В.В.Круглов
В.В.Борисов

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Теория и практика

Горячая линия - Телеком

В.В.Круглов
В.В.Борисов

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Теория и практика

Москва
Горячая линия - Телеком
2001

ББК 30.17
К 84
УДК 681.322

Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. – М.: Горячая линия - Телеком, 2001. – 382 с.: ил.

ISBN 5-93517-031-0

Книга посвящена одному из современных направлений в области информатики и вычислительной техники – нейрокомпьютерным технологиям. Достоинством книги является то, что в ней рассмотрены не только вопросы теории искусственных нейронных сетей, но и большое внимание уделено современным программным оболочкам-имитаторам нейронных сетей, а также решению с их помощью практических задач распознавания образов, кластеризации, прогнозирования, оптимизации, построения и использования нейросетевых экспертных систем. Книга содержит обширный справочный материал.

Для научных и инженерно-технических работников в области информатики и вычислительной техники, занимающихся созданием и использованием интеллектуальных систем, а также аспирантов и студентов разных специальностей в области компьютерных технологий.

Круглов Владимир Васильевич
Борисов Вадим Владимирович

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ. ТЕОРИЯ И ПРАКТИКА

Печатается в авторской редакции с оригинал-макета, подготовленного авторами

ЛР № 071825 от 16.03.99
ЛР № 071334 от 22.08.96

Подписано в печать 01.10.2000 Формат 60×88 $^{1}/_{16}$ Печать офсетная
Бумага газетная Гарнитура Arial Печ. л. 24,0
Тираж 2500 Заказ 7161 Изд. № 90 Издательский дом "ГРААЛЬ"
г. Пушкино Московской обл., ул. Лесная, д. 5

Отпечатано в Производственно-издательском комбинате ВИНИТИ,
140010, г. Люберцы Московской обл., Октябрьский пр-т, 403.
Тел. 554-21-86

ISBN 5-93517-031-0

Введение

Искусственные нейронные сети (ИНС) строятся по принципам организации и функционирования их биологических аналогов. Они способны решать широкий круг задач распознавания образов, идентификации, прогнозирования, оптимизации, управления сложными объектами. Дальнейшее повышение производительности компьютеров все в большой мере связывают с ИНС, в частности, с нейрокомпьютерами (НК), основу которых составляет искусственная нейронная сеть.

Термин «нейронные сети» сформировался к середине 50-х годов XX века. Основные результаты в этой области связаны с именами У. Маккалоха, Д. Хебба, Ф. Розенблатта, М. Минского, Дж. Хопфилда. Приведем краткую историческую справку.

1943 г. У. Маккалох (W. McCulloch) и У. Питтс (W. Pitts) предложили модель нейрона и сформулировали основные положения теории функционирования головного мозга.

1949 г. Д. Хебб (D. Hebb) высказал идеи о характере соединений нейронов мозга и их взаимодействии (клеточные ансамбли, синаптическая пластичность). Впервые предложил правила обучения нейронной сети

1957 г. Ф. Розенблатт (F. Rosenblatt) разработал принципы организации и функционирования персептронов, предложил вариант технической реализации первого в мире нейрокомпьютера Mark.

1959 г. Д. Хьюбел (D. Hubel) и Т. Визель (T. Wiesel) показали распределенный и параллельный характер хранения и обработки информации в биологических нейронных сетях.

1960–1968 гг. Активные исследования в области искусственных нейронных сетей, например, АДАЛИНА и МАДАЛИНА В. Уидроу (W. Widrow) (1960–1962 гг.), ассоциативные матрицы К. Штайнбуха (K. Steinbuch) (1961 г.).

1969 г. Публикация книги М. Минского (M. Minsky) и С. Пейперта (S. Papert) «Персептроны», в которой доказывается принципиальная ограниченность возможностей персептронов. Угасание интереса к искусственным нейронным сетям.

1970–1976 гг. Активные разработки в области персептронов в СССР (основные заказчики – военные ведомства).

Конец 1970-х гг. Возобновление интереса к искусственным нейронным сетям как следствие накопления новых знаний о деятельности мозга, а также значительного прогресса в области микроэлектроники и компьютерной техники.

1982–1985 гг. Дж. Хопфилд (J. Hopfield) предложил семейство оптимизирующих нейронных сетей, моделирующих ассоциативную память.

1985 г. Появление первых коммерческих нейрокомпьютеров, например, Mark III фирмы TRW (США).

1987 г. Начало широкомасштабного финансирования разработок в области ИНС и НК в США, Японии и Западной Европе (японская программа «Human Frontiers» и европейская программа «Basic Research in Adaptive Intelligence and Neurocomputing»).

1989 г. Разработки и исследования в области ИНС и НК ведутся практически всеми крупными электротехническими фирмами. Нейрокомпьютеры становятся одним из самых динамичных секторов рынка (за два года объем продаж вырос в пять раз). Агентством DARPA (Defence Advanced Research Projects Agency) министерства обороны США начато финансирование программы по созданию сверхбыстродействующих образцов НК для разнообразных применений.

1990 г. Активизация советских исследовательских организаций в области ИНС и НК (Институт кибернетики им. Глушкова в Киеве, Институт многопроцессорных вычислительных систем в Таганроге, Институт нейрокибернетики в Ростове-на-Дону). Общее число фирм, специализирующихся в области ИНС и НК, достигает трехсот.

1991 г. Годовой объем продаж на рынке ИНС и НК приблизился к 140 млн. долларам. Создаются центры нейрокомпьютеров в Москве, Киеве, Минске, Новосибирске, С.-Петербурге.

1992 г. Работы в области ИНС находятся стадии интенсивного развития. Ежегодно проводится десятки международных конференций и форумов по нейронным сетям, число специализированных периодических научных изданий по указанной тематике достигло двух десятков наименований.

1996 г. Число международных конференций по ИНС и НК достигло ста.

1997 г. Годовой объем продаж на рынке ИНС и НК превысил 2 млрд. долларов, а ежегодный прирост составил 50%.

2000 г. Переход на субмикронные и нанотехнологии, а также успехи молекулярной и биомолекулярной технологии приводят к принципиально новым архитектурным и технологическим решениям по созданию нейрокомпьютеров.

Глубокое изучение ИНС требует знания нейрофизиологии, науки о познании, психологии, физики (статистической механики), теории управления, теории вычислений, проблем искусственного интеллекта, статистики/математики, распознавания образов, компьютерного зрения, параллельных вычислений и аппаратных средств (цифровых и аналоговых). С другой стороны, ИНС также стимулируют эти дисциплины, обеспечивая их новыми инструментами и представлениями. Этот симбиоз жизненно необходим для исследования нейронных сетей.

Представим некоторые проблемы, решаемые искусственными нейронными сетями.

Классификация образов. Задача состоит в указании принадлежности входного образа, представленного вектором признаков, одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание речи, классификация сигнала электрокардиограммы, классификация клеток крови.

Кластеризация/категоризация. При решении задачи кластеризации, которая известна также как классификация образов без учителя, отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на подобии образов и размещает близкие образы в один кластер. Известны случаи применения кластеризации для извлечения знаний, сжатия данных и исследования свойств данных.

Аппроксимация функций. Предположим, что имеется обучающая выборка $((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N))$, которая генерируется неизвестной функцией, искаженной шумом. Задача аппроксимации состоит в нахождении оценки этой функции.

Предсказание/прогноз. Пусть заданы N дискретных отсчетов $\{y(t_1), y(t_2), \ldots, y(t_N)\}$ в последовательные моменты времени $t_1, t_2, \ldots, t_N$. Задача состоит в предсказании значения $y(t_{N+1})$ в момент t_{N+1}. Прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике.

Оптимизация. Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей оптимизации является нахождение решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию.

Память, адресуемая по содержанию. В модели вычислений фон Неймана обращение к памяти доступно только посредством адреса, который не зависит от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Память, адресуемая по содержанию, или ассоциативная память, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному или искаженному содержанию. Ассоциативная память чрезвычайно желательна при создании перспективных информационно-вычислительных систем.

Управление. Рассмотрим динамическую систему, заданную совокупностью $\{u(t), y(t)\}$, где $u(t)$ является входным управляющим воздействием, а $y(t)$ – выходом системы в момент времени t. В системах управления с эталонной моделью целью управления является расчет такого входного воздействия $u(t)$, при котором система следует по желаемой траектории, диктуемой эталонной моделью.

Каким образом нейронная сеть решает все эти, часто неформализуемые или трудно формализуемые задачи? Как известно, для решения таких задач традиционно применяются два основных подхода. Первый, основанный на правилах (rule-based), характерен для экспертных систем. Он базируется на описании предметной области в виде набора правил (аксиом) «если ..., то ...» и правил вывода. Искомое знание представляется в этом случае теоремой, истинность которой доказывается посредством построения цепочки вывода. При этом подходе, однако, необходимо заранее знать весь набор закономерностей, описывающих предметную область. При использовании другого подхода, основанного на примерах (case-based), надо лишь иметь достаточное количество примеров для настройки адаптивной системы с заданной степенью достоверности. Нейронные сети представляют собой классический пример такого подхода.

Книга состоит из трех частей и приложений. Первая часть посвящена вопросам теории искусственных нейронных сетей, вторая – программным оболочкам-имитаторам нейронных сетей, в третьей приведены конкретные примеры применения нейросетевого подхода для решения практических задач. Приложение содержат данные справочного характера.

Несмотря на огромный интерес, проявляемый к искусственным нейронным сетям, литература по этому направлению в нашей стране издается весьма малыми тиражами и является дефицитной, а зачастую слишком узкоспециализированной и поэтому трудной для понимания. Целью настоящей книги, в связи с этим, является знакомство широкого круга заинтересованных лиц с основными понятиями и методами исследования и применения нейронных сетей.

Авторы подчеркивают, что изданием данной книги они не преследуют коммерческие интересы и выражают глубокую благодарность всем, чьи материалы были использованы в работе.

Часть I

ТЕОРИЯ

Глава 1

ОСНОВНЫЕ ПОЛОЖЕНИЯ ТЕОРИИ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Под нейронными сетями подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга. Они представляют собой распределенные и параллельные системы, способные к адаптивному обучению путем анализа положительных и отрицательных воздействий. Элементарным преобразователем в данных сетях является искусственный нейрон или просто нейрон, названный так по аналогии с биологическим прототипом.

К настоящему времени предложено и изучено большое количество моделей нейроподобных элементов и нейронных сетей, ряд из которых рассмотрен в настоящей главе.

1.1. Биологический нейрон

Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами. Нервные волокна способны передавать электрические импульсы между нейронами. Все процессы передачи раздражений от кожи, ушей и глаз к мозгу, процессы мышления и управления действиями – все это реализовано в живом организме как передача электрических импульсов между нейронами.

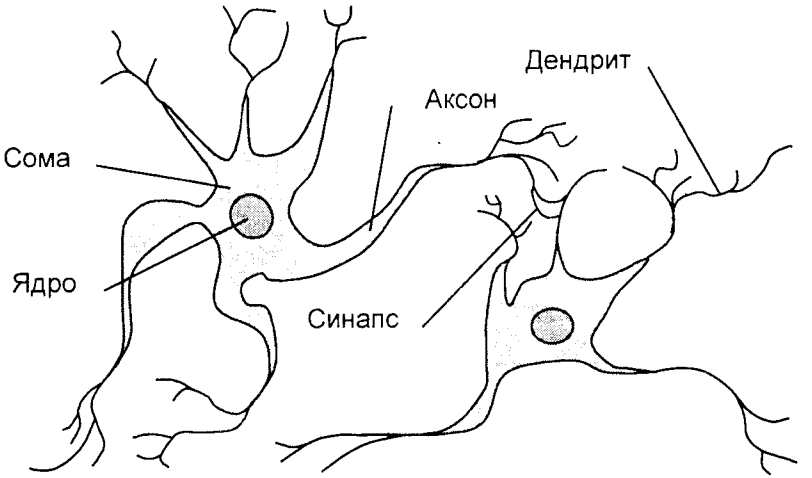


Рис. 1.1. Взаимосвязь биологических нейронов

Нейрон (нервная клетка) является особой биологической клеткой, которая обрабатывает информацию (рис. 1.1). Он состоит из тела (cell body), или *сомы* (soma), и отростков нервных волокон двух типов – *дендритов* (dendrites), по которым принимаются импульсы, и единственного *аксона* (axon), по которому нейрон может передавать импульс. Тело нейрона включает *ядро* (nucleus), которое содержит информацию о наследственных свойствах, и *плазму*, обладающую молекулярными средствами для производства необходимых нейрону материалов. Нейрон получает сигналы (импульсы) от аксонов других нейронов через дендриты (приемники) и передает сигналы, сгенерированные телом клетки, вдоль своего аксона (передатчика), который в конце разветвляется на волокна (strands). На окончаниях этих волокон находятся специальные образования – *синапсы* (synapses), которые влияют на величину импульсов.

Синапс является элементарной структурой и функциональным узлом между двумя нейронами (волокно аксона одного нейрона и дендрит другого). Когда импульс достигает синаптического окончания, высвобождаются химические вещества, называемые нейротрансмиттерами. Нейротрансмиттеры диффундируют через синаптическую щель, возбуждая или затормаживая, в зависимости от типа синапса, способность нейрона-приемника генерировать

электрические импульсы. Результативность передачи импульса синапсом может настраиваться проходящими через него сигналами так, что синапсы могут обучаться в зависимости от активности процессов, в которых они участвуют. Эта зависимость от предыстории действует как память, которая, возможно, ответственна за память человека. Важно отметить, что веса синапсов могут изменяться со временем, а значит, меняется и поведение соответствующих нейронов.

Кора головного мозга человека содержит около 10^{11} нейронов и представляет собой протяженную поверхность толщиной от 2 до 3 мм с площадью около 2200 см^2. Каждый нейрон связан с 10^3–10^4 другими нейронами. В целом мозг человека содержит приблизительно от 10^{14} до 10^{15} взаимосвязей.

Нейроны взаимодействуют короткими сериями импульсов продолжительностью, как правило, несколько миллисекунд. Сообщение передается посредством частотно-импульсной модуляции. Частота может изменяться от нескольких единиц до сотен герц, что в миллион раз медленнее, чем быстродействующие переключательные электронные схемы. Тем не менее сложные задачи распознавания человек решает за несколько сотен миллисекунд. Эти решения контролируются сетью нейронов, которые имеют скорость выполнения операций всего несколько миллисекунд. Это означает, что вычисления требуют не более 100 последовательных стадий. Другими словами, для таких сложных задач мозг «запускает» параллельные программы, содержащие около 100 шагов. Рассуждая аналогичным образом, можно обнаружить, что количество информации, посылаемое от одного нейрона другому, должно быть очень малым (несколько бит). Отсюда следует, что основная информация не передается непосредственно, а захватывается и распределяется в связях между нейронами.

1.2. Структура и свойства искусственного нейрона

Нейрон является составной частью нейронной сети. На рис. 1.2 показана его структура. Он состоит из элементов трех типов: умножителей (синапсов), сумматора и нелинейного преобразователя. Синапсы осуществляют связь между нейронами, умножают входной сигнал на число, характеризующее силу связи, (вес синапса). Сумматор выполняет сложение сигналов, поступающих по синаптическим связям от других нейронов, и внешних входных сигналов. Нелинейный преобразователь реализует нелинейную функцию одного аргумента – выхода сумматора. Эта функция называется *функцией активации* или *передаточной функцией* нейрона. Нейрон в целом реализует скалярную функцию векторного аргумента. Математическая модель нейрона:

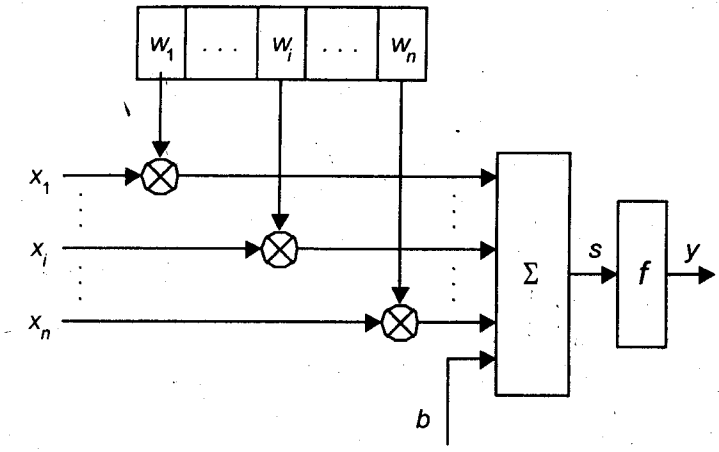


Рис. 1.2. Структура искусственного нейрона

$$s = \sum_{i=1}^{n} w_i x_i + b, \qquad (1.1)$$

$$y = f(s), \qquad (1.2)$$

где w_i – вес (weight) синапса, $i = 1...n$; b – значение смещения (bias); s – результат суммирования (sum); x_i – компонент входного вектора (входной сигнал), $i = 1...n$; y – выходной сигнал нейрона; n – число входов нейрона; f – нелинейное преобразование (функция активации).

В общем случае входной сигнал, весовые коэффициенты и смещение могут принимать действительные значения, а во многих практических задачах – лишь некоторые фиксированные значения. Выход (y) определяется видом функции активации и может быть как действительным, так и целым.

Синаптические связи с положительными весами называют *возбуждающими*, с отрицательными весами – *тормозящими*.

Описанный вычислительный элемент можно считать упрощенной математической моделью биологических нейронов. Чтобы подчеркнуть различие нейронов биологических и искусственных, вторые иногда называют *нейроноподобными элементами* или *формальными нейронами*.

На входной сигнал (s) нелинейный преобразователь отвечает выходным сигналом $f(s)$, который представляет собой выход y

нейрона. Примеры активационных функций представлены в табл. 1.1 и на рис. 1.3.

Таблица 1.1

Функции активации нейронов

Название	Формула	Область значений
Линейная	$f(s)=k\,s$	$(-\infty, \infty)$
Полулинейная	$f(s)=\begin{cases} k\,s, & s>0, \\ 0, & s\le 0 \end{cases}$	$(0, \infty)$
Логистическая (сигмоидальная)	$f(s)=\dfrac{1}{1+e^{-as}}$	$(0, 1)$
Гиперболический тангенс (сигмоидальная)	$f(s)=\dfrac{e^{as}-e^{-as}}{e^{as}+e^{-as}}$	$(-1, 1)$
Экспоненциальная	$f(s)=e^{-as}$	$(0, \infty)$
Синусоидальная	$f(s)=\sin(s)$	$(-1, 1)$
Сигмоидальная (рациональная)	$f(s)=\dfrac{s}{a+\lvert s\rvert}$	$(-1, 1)$
Шаговая (линейная с насыщением)	$f(s)=\begin{cases} -1, & s\le -1, \\ s, & -1<s<1, \\ 1, & s\ge 1 \end{cases}$	$(-1, 1)$
Пороговая	$f(s)=\begin{cases} 0, & s<0, \\ 1, & s\ge 0 \end{cases}$	$(0, 1)$
Модульная	$f(s)=\lvert s\rvert$	$(0, \infty)$
Знаковая (сигнатурная)	$f(s)=\begin{cases} 1, & s>0, \\ -1, & s\le 0 \end{cases}$	$(-1, 1)$
Квадратичная	$f(s)=s^2$	$(0, \infty)$

Одной из наиболее распространенных является нелинейная функция активации с насыщением, так называемая логистическая функция или сигмоид (функция *S*-образного вида):

$$f(s)=\frac{1}{1+e^{-as}}. \qquad (1.3)$$

При уменьшении a сигмоид становится более пологим, в пределе при $a = 0$ вырождаясь в горизонтальную линию на уровне 0,5, при увеличении a сигмоид приближается к виду функции единичного скачка с порогом Θ. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне (0, 1). Одно из ценных свойств сигмоидальной функции – простое выражение для ее производной, применение которой будет рассмотрено в дальнейшем:

$$f'(s)=a\,f(s)\left[1-f(s)\right]. \qquad (1.4)$$

Следует отметить, что сигмоидальная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того, она обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон.

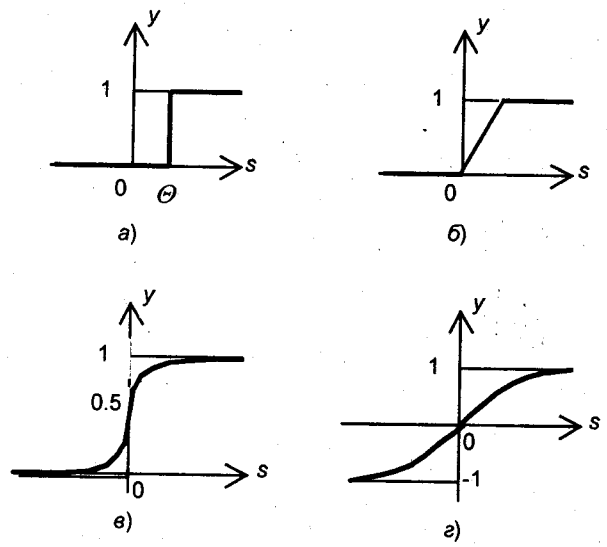


Рис. 1.3. Примеры активационных функций:

а – функция единичного скачка; *б* – линейный порог (гистерезис); *в* – сигмоид (логистическая функция); *г* – сигмоид (гиперболический тангенс)

1.3. Классификация нейронных сетей и их свойства

Нейронная сеть представляет собой совокупность нейроподобных элементов, определенным образом соединенных друг с другом и с внешней средой с помощью связей, определяемых весовыми коэффициентами. В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

нейрона. Примеры активационных функций представлены в табл. 1.1 и на рис. 1.3.

Таблица 1.1

Функции активации нейронов

Название	Формула	Область значений
Линейная	$f(s)=k\,s$	$(-\infty, \infty)$
Полулинейная	$f(s)=\begin{cases} k\,s, & s>0, \\ 0, & s\le 0 \end{cases}$	$(0, \infty)$
Логистическая (сигмоидальная)	$f(s)=\dfrac{1}{1+e^{-as}}$	$(0, 1)$
Гиперболический тангенс (сигмоидальная)	$f(s)=\dfrac{e^{as}-e^{-as}}{e^{as}+e^{-as}}$	$(-1, 1)$
Экспоненциальная	$f(s)=e^{-as}$	$(0, \infty)$
Синусоидальная	$f(s)=\sin(s)$	$(-1, 1)$
Сигмоидальная (рациональная)	$f(s)=\dfrac{s}{a+\lvert s\rvert}$	$(-1, 1)$
Шаговая (линейная с насыщением)	$f(s)=\begin{cases} -1, & s\le -1, \\ s, & -1<s<1, \\ 1, & s\ge 1 \end{cases}$	$(-1, 1)$
Пороговая	$f(s)=\begin{cases} 0, & s<0, \\ 1, & s\ge 0 \end{cases}$	$(0, 1)$
Модульная	$f(s)=\lvert s\rvert$	$(0, \infty)$
Знаковая (сигнатурная)	$f(s)=\begin{cases} 1, & s>0, \\ -1, & s\le 0 \end{cases}$	$(-1, 1)$
Квадратичная	$f(s)=s^2$	$(0, \infty)$

Одной из наиболее распространенных является нелинейная функция активации с насыщением, так называемая логистическая функция или сигмоид (функция *S*-образного вида):

$$f(s)=\frac{1}{1+e^{-as}}. \qquad (1.3)$$

При уменьшении a сигмоид становится более пологим, в пределе при $a = 0$ вырождаясь в горизонтальную линию на уровне 0,5, при увеличении a сигмоид приближается к виду функции единичного скачка с порогом Θ. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне (0, 1). Одно из ценных свойств сигмоидальной функции – простое выражение для ее производной, применение которой будет рассмотрено в дальнейшем:

$$f'(s)=a\,f(s)\left[1-f(s)\right]. \qquad (1.4)$$

Следует отметить, что сигмоидальная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того, она обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон.

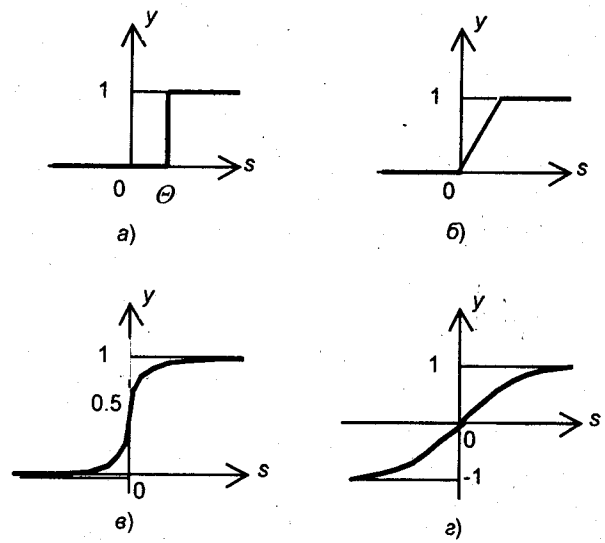


Рис. 1.3. Примеры активационных функций:

а – функция единичного скачка; *б* – линейный порог (гистерезис); *в* – сигмоид (логистическая функция); *г* – сигмоид (гиперболический тангенс)

1.3. Классификация нейронных сетей и их свойства

Нейронная сеть представляет собой совокупность нейроподобных элементов, определенным образом соединенных друг с другом и с внешней средой с помощью связей, определяемых весовыми коэффициентами. В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

- *входные нейроны*, на которые подается вектор, кодирующий входное воздействие или образ внешней среды; в них обычно не осуществляется вычислительных процедур, а информация передается с входа на выход путем изменения их активации;
- *выходные нейроны*, выходные значения которых представляют выходы нейронной сети; преобразования в них осуществляются по выражениям (1.1) и (1.2);
- *промежуточные нейроны*, составляющие основу нейронных сетей, преобразования в которых выполняются также по выражениям (1.1) и (1.2).

В большинстве нейронных моделей тип нейрона связан с его расположением в сети. Если нейрон имеет только выходные связи, то это входной нейрон, если наоборот – выходной нейрон. Однако возможен случай, когда выход топологически внутреннего нейрона рассматривается как часть выхода сети. В процессе функционирования сети осуществляется преобразование входного вектора в выходной, некоторая переработка информации. Конкретный вид выполняемого сетью преобразования данных обусловливается не только характеристиками нейроподобных элементов, но и особенностями ее архитектуры, а именно топологией межнейронных связей, выбором определенных подмножеств нейроподобных элементов для ввода и вывода информации, способами обучения сети, наличием или отсутствием конкуренции между нейронами, направлением и способами управления и синхронизации передачи информации между нейронами.

С точки зрения топологии можно выделить три основных типа нейронных сетей:

- полносвязные (рис. 1.4, *а*);
- многослойные или слоистые (рис. 1.4, *б*);
- слабосвязные (с локальными связями) (рис. 1.4, *в*).

В *полносвязных нейронных сетях* каждый нейрон передает свой выходной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются всем нейронам. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети.

В *многослойных нейронных сетях* нейроны объединяются в слои. Слой содержит совокупность нейронов с едиными входными сигналами. Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях. В общем случае сеть состоит из Q слоев, пронумерованных слева направо. Внешние входные сигналы подаются на входы нейронов входного слоя (его часто нумеруют как нулевой), а выходами сети являются выходные

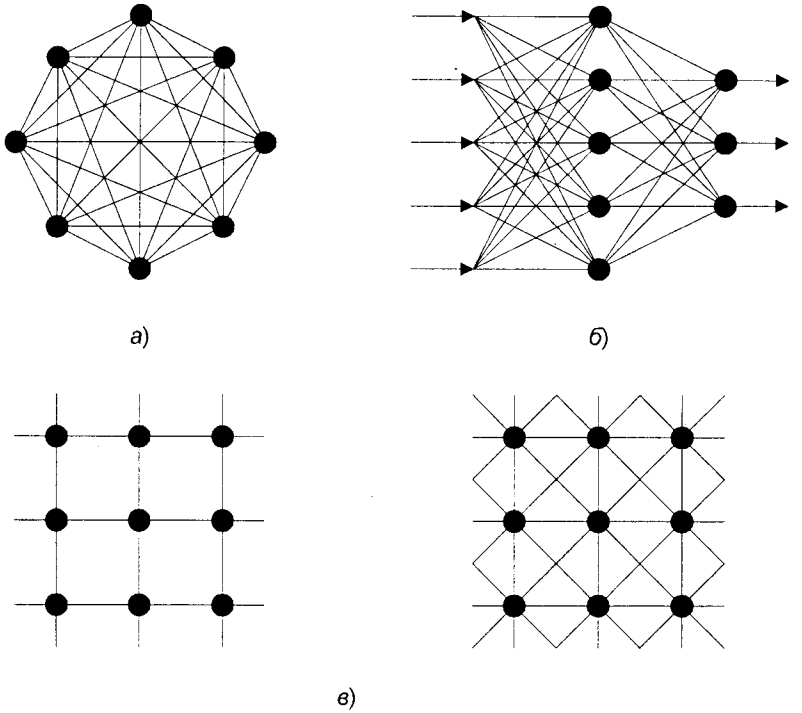


Рис. 1.4. Архитектуры нейронных сетей:

а – полносвязная сеть, *б* – многослойная сеть с последовательными связями, *в* – слабосвязные сети

сигналы последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько скрытых слоев. Связи от выходов нейронов некоторого слоя q к входам нейронов следующего слоя (q+1) называются последовательными.

В свою очередь, среди многослойных нейронных сетей выделяют следующие типы.

1) *Монотонные.*

Это частный случай слоистых сетей с дополнительными условиями на связи и нейроны. Каждый слой кроме последнего (выходного) разбит на два блока: возбуждающий и тормозящий. Связи между блоками тоже разделяются на тормозящие и возбуждающие. Если от нейронов блока А к нейронам блока В ведут только возбуждающие связи, то это означает, что любой выходной сигнал

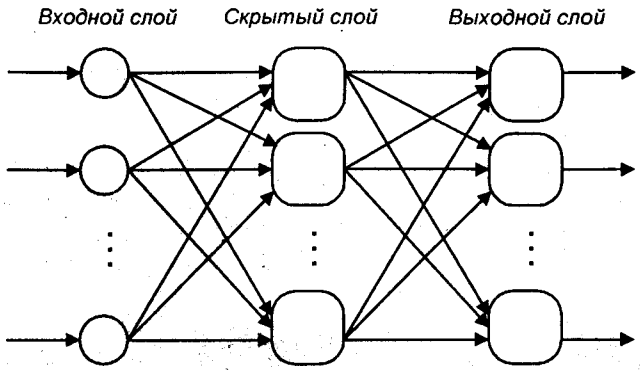


Рис. 1.5. Многослойная (двухслойная) сеть прямого распространения

блока является монотонной неубывающей функцией любого выходного сигнала блока А. Если же эти связи только тормозящие, то любой выходной сигнал блока В является невозрастающей функцией любого выходного сигнала блока А. Для нейронов монотонных сетей необходима монотонная зависимость выходного сигнала нейрона от параметров входных сигналов.

2) *Сети без обратных связей.* В таких сетях нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам первого скрытого слоя, и так далее вплоть до выходного, который выдает сигналы для интерпретатора и пользователя. Если не оговорено противное, то каждый выходной сигнал q-го слоя подастся на вход всех нейронов $(q+1)$-го слоя; однако возможен вариант соединения q-го слоя с произвольным $(q+p)$-м слоем.

Среди многослойных сетей без обратных связей различают полносвязанные (выход каждого нейрона q-го слоя связан с входом каждого нейрона $(q+1)$-го слоя) и частично полносвязанные. Классическим вариантом слоистых сетей являются полносвязанные сети прямого распространения (рис. 1.5).

3) *Сети с обратными связями.* В сетях с обратными связями информация с последующих слоев передается на предыдущие. Среди них, в свою очередь, выделяют следующие:

- слоисто-циклические, отличающиеся тем, что слои замкнуты в кольцо: последний слой передает свои выходные сигналы первому; все слои равноправны и могут как получать входные сигналы, так и выдавать выходные;

- слоисто-полносвязанные состоят из слоев, каждый из которых представляет собой полносвязную сеть, а сигналы передаются как от слоя к слою, так и внутри слоя; в каждом слое цикл работы распадается на три части: прием сигналов с предыдущего слоя, обмен сигналами внутри слоя, выработка выходного сигнала и передача к последующему слою;
- полносвязанно-слоистые, по своей структуре аналогичные слоисто-полносвязанным, но функционирующим по-другому: в них не разделяются фазы обмена внутри слоя и передачи следующему, на каждом такте нейроны всех слоев принимают сигналы от нейронов как своего слоя, так и последующих.

В качестве примера сетей с обратными связями на рис. 1.6 представлены частично-рекуррентные сети Элмана и Жордана.

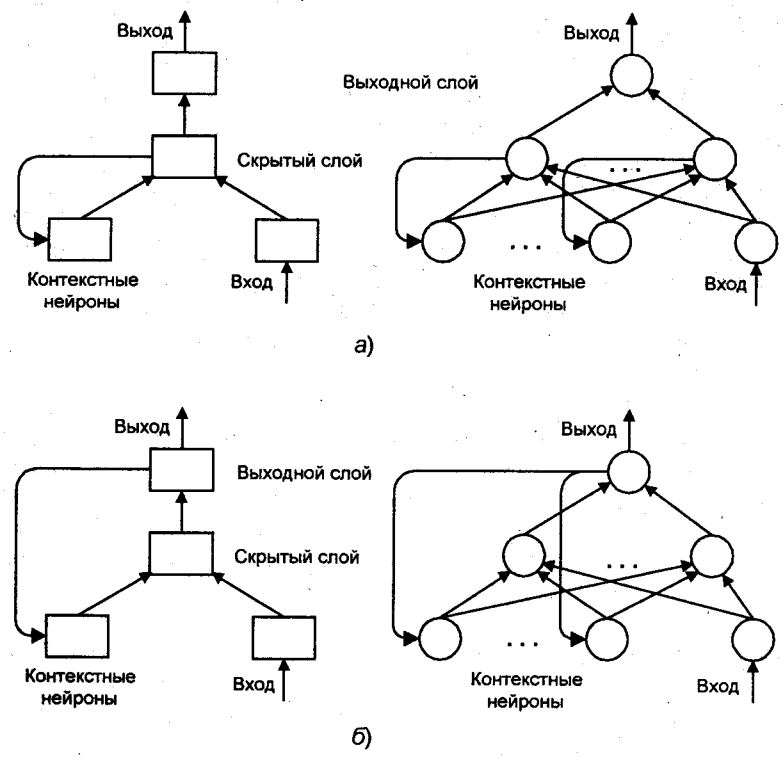


Рис. 1.6. Частично-рекуррентные сети:

а – Элмана, *б* – Жордана

В *слабосвязных нейронных сетях* нейроны располагаются в узлах прямоугольной или гексагональной решетки. Каждый нейрон связан с четырьмя (окрестность фон Неймана), шестью (окрестность Голея) или восемью (окрестность Мура) своими ближайшими соседями.

Известные нейронные сети можно разделить по типам структур нейронов на гомогенные (однородные) и гетерогенные. Гомогенные сети состоят из нейронов одного типа с единой функцией активации, а в гетерогенную сеть входят нейроны с различными функциями активации.

Существуют бинарные и аналоговые сети. Первые из них оперируют только двоичными сигналами, и выход каждого нейрона может принимать значение либо логического ноля (заторможенное состояние) либо логической единицы (возбужденное состояние).

Еще одна классификация делит нейронные сети на синхронные и асинхронные. В первом случае в каждый момент времени лишь один нейрон меняет свое состояние, во втором – состояние меняется сразу у целой группы нейронов, как правило, у всего слоя. Алгоритмически ход времени в нейронных сетях задается итерационным выполнением однотипных действий над нейронами. Далее будут рассматриваться только синхронные сети.

Сети можно классифицировать также по числу слоев. Теоретически число слоев и число нейронов в каждом слое может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированных микросхем, на которых обычно реализуется нейронная сеть. Чем сложнее сеть, тем более сложные задачи она может решать.

Выбор структуры нейронной сети осуществляется в соответствии с особенностями и сложностью задачи. Для решения отдельных типов задач уже существуют оптимальные конфигурации, описанные в приложении. Если же задача не может быть сведена ни к одному из известных типов, приходится решать сложную проблему синтеза новой конфигурации. При этом необходимо руководствоваться следующими основными правилами:

- возможности сети возрастают с увеличением числа нейронов сети, плотности связей между ними и числом слоев;
- введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети;
- сложность алгоритмов функционирования сети, введение нескольких типов синапсов способствует усилению мощности нейронной сети.

Вопрос о необходимых и достаточных свойствах сети для решения задач того или иного рода представляет собой целое направление нейрокомпьютерной науки. Так как проблема синтеза нейронной сети сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора, хотя в литературе приведены доказательства того, что для любого алгоритма существует нейронная сеть, которая может его реализовать. Остановимся на этом подробнее.

Многие задачи распознавания образов (зрительных, речевых), выполнения функциональных преобразований при обработке сигналов, управления, прогнозирования, идентификации сложных систем, сводятся к следующей математической постановке. Необходимо построить такое отображение $X \rightarrow Y$, чтобы на каждый возможный входной сигнал X формировался правильный выходной сигнал Y. Отображение задается конечным набором пар (<вход>, <известный выход>). Число этих пар (обучающих примеров) существенно меньше общего числа возможных сочетаний значений входных и выходных сигналов. Совокупность всех обучающих примеров носит название обучающей выборки.

В задачах распознавания образов X – некоторое представление образа (изображение, вектор), Y – номер класса, к которому принадлежит входной образ.

В задачах управления X – набор контролируемых параметров управляемого объекта, Y – код, определяющий управляющее воздействие, соответствующее текущим значениям контролируемых параметров.

В задачах прогнозирования в качестве входных сигналов используются временные ряды, представляющие значения контролируемых переменных на некотором интервале времени. Выходной сигнал – множество переменных, которое является подмножеством переменных входного сигнала.

При идентификации X и Y представляют входные и выходные сигналы системы соответственно.

Вообще говоря, большая часть прикладных задач может быть сведена к реализации некоторого сложного функционального многомерного преобразования.

В результате отображения $X \rightarrow Y$ необходимо обеспечить формирование правильных выходных сигналов в соответствии:

- со всеми примерами обучающей выборки;
- со всеми возможными входными сигналами, которые не вошли в обучающую выборку.

Второе требование в значительной степени усложняет задачу формирования обучающей выборки. В общем виде эта задача в настоящее время еще не решена, однако во всех известных случаях может быть найдено частное решение.

1.3.1. Теорема Колмогорова–Арнольда

Построить многомерное отображение $X \to Y$ – это значит представить его с помощью математических операций над не более, чем двумя переменными.

Проблема представления функций многих переменных в виде суперпозиции функций меньшего числа переменных восходит 13-й проблеме Гильберта. В результате многолетней научной полемики между А. Н. Колмогоровым и В. И. Арнольдом был получен ряд важных теоретических результатов, опровергающих тезис непредставимости функции многих переменных функциями меньшего числа переменных:

- теорема о возможности представления непрерывных функций нескольких переменных суперпозициями непрерывных функций меньшего числа переменных (1956 г.);
- теорема о представлении любой непрерывной функции трех переменных в виде суммы функций не более двух переменных (1957 г.);
- теорема о представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения (1957 г.).

1.3.2. Работа Хехт-Нильсена

Теорема о представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения в 1987 году была переложена Хехт-Нильсеном для нейронных сетей.

Теорема Хехт-Нильсена доказывает представимость функции многих переменных достаточно общего вида с помощью двухслойной нейронной сети с прямыми полными связями с n нейронами входного слоя, $(2n+1)$ нейронами скрытого слоя с заранее известными ограниченными функциями активации (например, сигмоидальными) и m нейронами выходного слоя с неизвестными функциями активации.

Теорема, таким образом, в неконструктивной форме доказывает решаемость задачи представления функции произвольного вида на нейронной сети и указывает для каждой задачи минимальные числа нейронов сети, необходимых для ее решения.

1.3.3. Следствия из теоремы Колмогорова–Арнольда – Хехт-Нильсена

Следствие 1. Из теоремы Хехт-Нильсена следует представимость любой многомерной функции нескольких переменных с помощью нейронной сети фиксированной размерности. Неизвестными остаются следующие характеристики функций активации нейронов:

- ограничения области значений (координаты асимптот) сигмоидальных функций активации нейронов скрытого слоя;
- наклон сигмоидальных функций активации;
- вид функций активации нейронов выходного слоя.

Про функции активации нейронов выходного слоя из теоремы Хехт-Нильсена известно только то, что они представляют собой нелинейные функции общего вида. В одной из работ, продолжающих развитие теории, связанной с рассматриваемой теоремой, доказывается, что функции активации нейронов выходного слоя должны быть монотонно возрастающими. Это утверждение в некоторой степени сужает класс функций, которые могут использоваться при реализации отображения с помощью двухслойной нейронной сети.

На практике требования теоремы Хехт-Нильсена к функциям активации удовлетворяются следующим образом. В нейронных сетях как для первого (скрытого), так и для второго (выходного) слоя используют сигмоидальные передаточные функции с настраиваемыми параметрами. То есть в процессе обучения индивидуально для каждого нейрона задается максимальное и минимальное значение, а также наклон сигмоидальной функции.

Следствие 2. Для любого множества пар (X^k, Y^k) (где Y^k – скаляр) существует двухслойная однородная (с одинаковыми функциями активации) нейронная сеть первого порядка с последовательными связями и с конечным числом нейронов, которая выполняет отображение $X \to Y$, выдавая на каждый входной сигнал X^k правильный выходной сигнал Y^k. Нейроны в такой двухслойной нейронной сети должны иметь сигмоидальные передаточные функции.

К сожалению, эта теорема не конструктивна. В ней не заложена методика определения числа нейронов в сети для некоторой конкретной обучающей выборки.

Для многих задач единичной размерности выходного сигнала недостаточно. Необходимо иметь возможность строить с помощью нейронных сетей функции $X \to Y$, где Y имеет произвольную размерность. Следующее утверждение является теоретической

основой для построения таких функций на базе однородных нейронных сетей.

Утверждение. Для любого множества пар входных-выходных векторов произвольной размерности $\{(X^k, Y^k), k = 1...N\}$ существует однородная двухслойная нейронная сеть с последовательными связями, с сигмоидальными передаточными функциями и с конечным числом нейронов, которая для каждого входного вектора X^k формирует соответствующий ему выходной вектор Y^k.

Таким образом, для представления многомерных функций многих переменных может быть использована однородная двухслойная нейронная сеть с сигмоидальными передаточными функциями.

Для оценки числа нейронов с скрытых слоях однородных нейронных сетей можно воспользоваться формулой для оценки необходимого числа синаптических весов L_w в многослойной сети с сигмоидальными передаточными функциями:

$$\frac{mN}{1+\log_2 N} \le L_w \le m\left(\frac{N}{m}+1\right)(n+m+1)+m, \qquad (1.5)$$

где n – размерность входного сигнала, m – размерность выходного сигнала, N – число элементов обучающей выборки.

Оценив необходимое число весов, можно рассчитать число нейронов в скрытых слоях. Например, для двухслойной сети это число составит:

$$L = \frac{L_w}{n+m}. \qquad (1.6)$$

Известны и другие формулы для оценки, например:

$$2(n + L + m) \le N \le 10(n + L + m), \qquad (1.7)$$

$$\frac{N}{10} - n - m \le L \le \frac{N}{2} - n - m. \qquad (1.8)$$

Точно так же можно рассчитать число нейронов в сетях с большим числом слоев.

Иногда целесообразно использовать сети с большим числом слоев. Такие многослойные нейронные сети могут иметь меньшие размерности матриц синаптических весов нейронов одного слоя, чем двухслойные сети, реализующие то же самое отображение. Однако строгой методики построения таких сетей пока нет.

Аналогичная ситуация складывается и с многослойными нейронными сетями, в которых помимо последовательных связей используются и прямые (связи от слоя с номером q к слою с номером $(q+p)$, где $p > 1$). Нет строгой теории, которая показывала бы возможность и целесообразность построения таких сетей.

Наибольшие проблемы возникают при использовании сетей циклического функционирования. К этой группе относятся многослойные сети с обратными связями (от слоя с номером q к слою с номером $(q+p)$, где $p < 0$), а также полносвязные сети. Для успешного функционирования таких сетей необходимо соблюдение условий динамической устойчивости, иначе сеть может не сойтись к правильному решению, либо, достигнув на некоторой итерации правильного значения выходного сигнала, после нескольких итераций уйти от этого значения. Проблема динамической устойчивости подробно исследована, пожалуй, лишь для одной модели из рассматриваемой группы – нейронной сети Хопфилда.

Отсутствие строгой теории для перечисленных моделей нейронных сетей не препятствует исследованию возможностей их применения.

Отметим, что отечественному читателю приведенные результаты известны в более фрагментарной форме – в виде так называемой *теоремы о полноте*.

Теорема о полноте.

Любая непрерывная функция на замкнутом ограниченном множестве может быть равномерно приближена функциями, вычисляемыми нейронными сетями, если функция активации нейрона дважды непрерывно дифференцируема и непрерывна.

Таким образом, нейронные сети являются универсальными структурами, позволяющими реализовать любой вычислительный алгоритм.

1.4. Постановка и возможные пути решения задачи обучения нейронных сетей

Очевидно, что процесс функционирования нейронной сети, сущность действий, которые она способна выполнять, зависит от величин синаптических связей. Поэтому, задавшись определенной структурой сети, соответствующей какой-либо задаче, необходимо найти оптимальные значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными).

Этот этап называется обучением нейронной сети, и от того, насколько качественно он будет выполнен, зависит способность сети решать поставленные перед ней проблемы во время функционирования.

В процессе функционирования нейронная сеть формирует выходной сигнал Y в соответствии с входным сигналом X, реализуя некоторую функцию g: $Y = g(X)$. Если архитектура сети задана,

то вид функции g определяется значениями синаптических весов и смещений сети. Обозначим через G множество всех возможных функций g, соответствующих заданной архитектуре сети.

Пусть решение некоторой задачи есть функция r: $Y = r(X)$, заданная парами входных-выходных данных (X^1, Y^1), ..., (X^k, Y^k), для которых $Y^k = r(X^k)$, $k = 1...N$. E – функция ошибки (функционал качества), показывающая для каждой из функций g степень близости к r.

Решить поставленную задачу с помощью нейронной сети заданной архитектуры – это значит построить (синтезировать) функцию $g \in G$, подобрав параметры нейронов (синаптические веса и смещения) таким образом, чтобы функционал качества обращался в оптимум для всех пар (X^k, Y^k).

Таким образом, задача обучения нейронной сети определяется совокупностью пяти компонентов:

$$<X, Y, r, G, E>.$$

Обучение состоит в поиске (синтезе) функции g, оптимальной по E. Оно требует длительных вычислений и представляет собой итерационную процедуру. Число итераций может составлять от 10^3 до 10^8. На каждой итерации происходит уменьшение функции ошибки.

Функция E может иметь произвольный вид. Если выбраны множество обучающих примеров и способ вычисления функции ошибки, обучение нейронной сети превращается в задачу многомерной оптимизации, для решения которой могут быть использованы следующие методы:

- локальной оптимизации с вычислением частных производных первого порядка;
- локальной оптимизации с вычислением частных производных первого и второго порядка;
- стохастической оптимизации;
- глобальной оптимизации.

К первой группе относятся: градиентный метод (наискорейшего спуска); методы с одномерной и двумерной оптимизацией целевой функции в направлении антиградиента; метод сопряженных градиентов; методы, учитывающие направление антиградиента на нескольких шагах алгоритма.

Ко второй группе относятся: метод Ньютона, методы оптимизации с разреженными матрицами Гессе, квазиньютоновские методы, метод Гаусса–Ньютона, метод Левенберга–Маркардта.

Стохастическими методами являются: поиск в случайном направлении, имитация отжига, метод Монте-Карло (численный метод статистических испытаний).

Задачи глобальной оптимизации решаются с помощью перебора значений переменных, от которых зависит целевая функция.

Для сравнения методов обучения нейронных сетей необходимо использовать два критерия:

- количество шагов алгоритма для получения решения;
- количество дополнительных переменных для организации вычислительного процесса.

Для иллюстрации термина «дополнительные переменные» приведем следующий пример. Пусть p_1 и p_2 – некоторые параметры нейронной сети с заданными значениями. В процессе обучения сети на каждом шаге, по меньшей мере, два раза потребуется выполнить умножение p_1p_2. Дополнительная переменная требуется для сохранения значение произведения после первого умножения.

Предпочтение следует отдавать тем методам, которые позволяют обучить нейронную сеть за небольшое число шагов и требуют малого числа дополнительных переменных. Это связано с ограничением ресурсов вычислительных средств. Как правило, для обучения используются персональные компьютеры.

Пусть нейронная сеть содержит P изменяемых параметров (синаптических весов и смещений). Существует лишь две группы алгоритмов обучения, которые требуют менее $2P$ дополнительных параметров и при этом дают возможность обучать нейронные сети за приемлемое число шагов. Это алгоритмы с вычислением частных производных первого порядка и, возможно, одномерной оптимизации. Именно эти алгоритмы и будут рассмотрены в следующих разделах.

Хотя указанные алгоритмы дают возможность находить только локальные экстремумы, они могут быть использованы на практике для обучения нейронных сетей с многоэкстремальными целевыми функциями (функциями ошибки), так как экстремумов у целевой функции, как правило, не очень много. Достаточно лишь раз или два вывести сеть из локального минимума с большим значением целевой функции для того, чтобы в результате итераций в соответствии с алгоритмом локальной оптимизации сеть оказалась в локальном минимуме со значением целевой функции, близким к нулю. Если после нескольких попыток вывести сеть из локального минимума нужного эффекта добиться не удается, необходимо увеличить число нейронов во всех слоях с первого по предпослед-

ний и присвоить случайным образом их синаптическим весам и смещениям значения из заданного диапазона.

Эксперименты по обучению нейронных сетей показали, что совместное использование алгоритма локальной оптимизации, процедуры вывода сети из локального минимума и процедуры увеличения числа нейронов приводит к успешному обучению нейронных сетей.

Кратко опишем недостатки других методов. Стохастические алгоритмы требуют очень большого числа шагов обучения. Это делает невозможным их практическое использование для обучения нейронных сетей больших размерностей. Экспоненциальный рост сложности перебора с ростом размерности задачи в алгоритмах глобальной оптимизации при отсутствии априорной информации о характере целевой функции также делает невозможным их использование для обучения нейронных сетей больших размерностей. Метод сопряженных градиентов очень чувствителен к точности вычислений, особенно при решении задач оптимизации большой размерности. Для методов, учитывающих направление антиградиента на нескольких шагах алгоритма, и методов, включающих вычисление матрицы Гессе, необходимо более чем $2P$ дополнительных переменных. В зависимости от способа разрежения, вычисление матрицы Гессе требует от $2P$ до P^2 дополнительных переменных.

Рассмотрим один из самых распространенных алгоритмов обучения – алгоритм обратного распространения ошибки.

1.4.1. Обучение с учителем. Алгоритм обратного распространения ошибки

В многослойных нейронных сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, неизвестны. Трех- или более слойный персептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах сети.

Один из вариантов решения этой проблемы – разработка наборов выходных сигналов, соответствующих входным, для каждого слоя нейронной сети, что, конечно, является очень трудоемкой операцией и не всегда осуществимо. Второй вариант – динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный метод, несмотря на кажущуюся простоту, требует громоздких рутинных вычислений. И, наконец, третий, более приемлемый вариант – распространение сигналов ошибки от выходов нейронной сети к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения получил название процедуры *обратного распространения ошибки* (error back propagation). Именно он рассматривается ниже.

Алгоритм обратного распространения ошибки – это итеративный градиентный алгоритм обучения, который используется с целью минимизации среднеквадратичного отклонения текущих от требуемых выходов многослойных нейронных сетей с последовательными связями.

Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки нейронной сети является величина:

$$E(w) = \frac{1}{2}\sum_{j,k}(y_{j,k}^{(Q)} - d_{j,k})^2, \qquad (1.9)$$

где $y_{j,k}^{(Q)}$ – реальное выходное состояние нейрона j выходного слоя нейронной сети при подаче на ее входы k-го образа; $d_{j,k}$ – требуемое выходное состояние этого нейрона.

Суммирование ведется по всем нейронам выходного слоя и по всем обрабатываемым сетью образам. Минимизация методом градиентного спуска обеспечивает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(q)} = -\eta \frac{\partial E}{\partial w_{ij}}, \qquad (1.10)$$

где w_{ij} – весовой коэффициент синаптической связи, соединяющей i-й нейрон слоя $(q-1)$ с j-м нейроном слоя q; η – коэффициент скорости обучения, $0 < \eta < 1$.

В соответствии с правилом дифференцирования сложной функции:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j}\frac{dy_j}{ds_j}\frac{\partial s_j}{\partial w_{ij}}, \qquad (1.11)$$

где s_j – взвешенная сумма входных сигналов нейрона j, т. е. аргумент активационной функции. Так как производная активационной функции должна быть определена на всей оси абсцисс, то функция единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемых нейронных сетей. В них применяются такие гладкие функции, как гиперболический тангенс или классический сигмоид с экспонентой (см. табл. 1.1). Например, в случае гиперболического тангенса:

$$\frac{dy}{ds} = 1 - s^2. \qquad (1.12)$$

Третий множитель $\partial s_j / \partial w_{ij}$ равен выходу нейрона предыдущего слоя $y_i^{(q-1)}$.

Что касается первого множителя в (1.11), он легко раскладывается следующим образом:

$$\frac{\partial E}{\partial y_j} = \sum_r \frac{\partial E}{\partial y_r} \frac{dy_r}{ds_r} \frac{\partial s_r}{\partial y_j} = \sum_r \frac{\partial E}{\partial y_r} \frac{dy_r}{ds_r} w_{jr}^{(q+1)}. \qquad (1.13)$$

Здесь суммирование по r выполняется среди нейронов слоя $(q+1)$.

Введя новую переменную:

$$\delta_j^{(q)} = \frac{\partial E}{\partial y_j} \frac{dy_j}{ds_j}, \qquad (1.14)$$

получим рекурсивную формулу для расчетов величин $\delta_j^{(q)}$ слоя q из величин $\delta_i^{(q+1)}$ более старшего слоя $(q+1)$:

$$\delta_j^{(q)} = \left[\sum_r \delta_r^{(q+1)} w_{jr}^{(q+1)} \right] \frac{dy_j}{ds_j}. \qquad (1.15)$$

Для выходного слоя:

$$\delta_j^{(Q)} = (y_j^{(Q)} - d_j) \frac{dy_j}{ds_j}. \qquad (1.16)$$

Теперь можно записать (1.10) в раскрытом виде:

$$\Delta w_{ij}^{(q)} = -\eta \, \delta_j^{(q)} \, y_i^{(q-1)}. \qquad (1.17)$$

Иногда для придания процессу коррекции весов некоторой инерционности, сглаживающей резкие скачки при перемещении по поверхности целевой функции, (1.17) дополняется значением изменения веса на предыдущей итерации:

$$\Delta w_{ij}^{(q)}(t) = -\eta \, (\mu \, \Delta w_{ij}^{(q)}(t-1) + (1-\mu) \, \delta_j^{(q)} \, y_i^{(q-1)}), \qquad (1.18)$$

где μ – коэффициент инерционности; t – номер текущей итерации.

Таким образом, полный алгоритм обучения нейронной сети с помощью процедуры обратного распространения строится следующим образом.

ШАГ 1. Подать на входы сети один из возможных образов и в режиме обычного функционирования нейронной сети, когда сигналы распространяются от входов к выходам, рассчитать значения последних. Напомним, что:

$$s_j^{(q)} = \sum_{i=0}^{L} y_i^{(q-1)} w_{ij}^{(q)}, \qquad (1.19)$$

где L – число нейронов в слое $(q-1)$ с учетом нейрона с постоянным выходным состоянием +1, задающего смещение; $y_i^{(q-1)} w_{ij}^{(q)}$ – i-й вход нейрона j слоя q,

$$y_j^{(q)} = f(s_j^{(q)}), \qquad (1.20)$$

где $f(\bullet)$ – сигмоид,

$$y_r^{(0)} = x_r, \qquad (1.21)$$

где x_r – r-я компонента вектора входного образа.

ШАГ 2. Рассчитать $\delta^{(Q)}$ для выходного слоя по формуле (1.16).

Рассчитать по формуле (1.17) или (1.18) изменения весов $\Delta w^{(Q)}$ слоя Q.

ШАГ 3. Рассчитать по формулам (1.15) и (1.17) (или (1.15) и (1.18)) соответственно $\delta^{(q)}$ и $\Delta w^{(Q)}$ для всех остальных слоев, $q = (Q-1) \ldots 1$.

ШАГ 4. Скорректировать все веса в нейронной сети:

$$w_{ij}^{(q)}(t) = w_{ij}^{(q)}(t-1) + \Delta w_{ij}^{(q)}(t). \qquad (1.22)$$

ШАГ 5. Если ошибка сети существенна, перейти на шаг 1. В противном случае – конец.

Сети на шаге 1 попеременно в случайном порядке предъявляются все тренировочные образы, чтобы сеть, образно говоря, не забывала одни по мере запоминания других.

Из выражения (1.17) следует, что когда выходное значение $y_i^{(q-1)}$ стремится к нулю, эффективность обучения заметно снижается. При двоичных входных векторах в среднем половина весовых коэффициентов не будет корректироваться, поэтому область возможных значений выходов нейронов (0, 1) желательно сдвинуть в пределы (–0,5, 0,5), что достигается простыми модификациями логистических функций. Например, сигмоид с экспонентой преобразуется к виду:

$$f(s) = -0{,}5 + \frac{1}{1 + e^{-\alpha s}}. \qquad (1.23)$$

Рассмотрим вопрос о емкости нейронной сети, т. е. числа образов, предъявляемых на ее входы, которые она способна научиться распознавать. Для сетей с числом слоев больше двух, этот вопрос остается открытым. Для сетей с двумя слоями, детерминистская емкость сети C_d оценивается следующим образом:

$$\frac{L_w}{m} < C_d < \frac{L_w}{m} \log\left(\frac{L_w}{m}\right), \qquad (1.24)$$

где L_w – число подстраиваемых весов, m – число нейронов в выходном слое.

Данное выражение получено с учетом некоторых ограничений. Во-первых, число входов n и нейронов в скрытом слое L должно удовлетворять неравенству $(n+L) > m$. Во-вторых, $L_w/m > 1000$. Однако приведенная оценка выполнена для сетей с пороговыми активационными функциями нейронов, а емкость сетей с гладкими активационными функциями, например (1.23), обычно больше. Кроме того, термин детерминистский означает, что полученная оценка емкости подходит для всех входных образов, которые могут быть представлены n входами. В действительности распределение входных образов, как правило, обладает некоторой регулярностью, что позволяет нейронной сети проводить обобщение и, таким образом, увеличивать реальную емкость. Так как распределение образов, в общем случае, заранее не известно, можно говорить о реальной емкости только предположительно, но обычно она раза в два превышает детерминистскую емкость.

Вопрос о емкости нейронной сети тесно связан с вопросом о требуемой мощности выходного слоя сети, выполняющего окончательную классификацию образов. Например, для разделения множества входных образов по двум классам достаточно одного выходного нейрона. При этом каждый логический уровень («1» и «0») будет обозначать отдельный класс. На двух выходных нейронах с пороговой функцией активации можно закодировать уже четыре класса. Для повышения достоверности классификации желательно ввести избыточность путем выделения каждому классу одного нейрона в выходном слое или, что еще лучше, нескольких, каждый из которых обучается определять принадлежность образа к классу со своей степенью достоверности, например: высокой, средней и низкой. Такие нейронные сети позволяют проводить классификацию входных образов, объединенных в нечеткие (размытые или пересекающиеся) множества. Это свойство приближает подобные сети к реальным условиям функционирования биологических нейронных сетей.

Рассматриваемая нейронная сеть имеет несколько «узких мест». Во-первых, в процессе большие положительные или отрицательные значения весов могут сместить рабочую точку на сигмоидах нейронов в область насыщения. Малые величины производной от логистической функции приведут в соответствии с (1.15) и (1.16) к остановке обучения, что парализует сеть. Во-вторых, применение метода градиентного спуска не гарантирует нахождения глобального минимума целевой функции. Это тесно связано вопросом выбора скорости обучения. Приращения весов и, следовательно, скорость обучения для нахождения экстремума должны быть бесконечно малыми, однако в этом случае обучение будет

происходить неприемлемо медленно. С другой стороны, слишком большие коррекции весов могут привести к постоянной неустойчивости процесса обучения. Поэтому в качестве коэффициента скорости обучения η обычно выбирается число меньше 1 (например, 0,1), которое постепенно уменьшается в процессе обучения. Кроме того, для исключения случайных попаданий сети в локальные минимумы иногда, после стабилизации значений весовых коэффициентов, η кратковременно значительно увеличивают, чтобы начать градиентный спуск из новой точки. Если повторение этой процедуры несколько раз приведет сеть в одно и то же состояние, можно предположить, что найден глобальный минимум.

Существует другой метод исключения локальных минимумов и паралича сети, заключающийся в применении стохастических нейронных сетей.

Дадим изложенному геометрическую интерпретацию.

В алгоритме обратного распространения вычисляется вектор градиента поверхности ошибок. Этот вектор указывает направление кратчайшего спуска по поверхности из текущей точки, движение по которому приводит к уменьшению ошибки. Последовательность уменьшающихся шагов приведет к минимуму того или иного типа. Трудность здесь представляет вопрос подбора длины шагов.

При большой величине шага сходимость будет более быстрой, но имеется опасность перепрыгнуть через решение или в случае сложной формы поверхности ошибок уйти в неправильном направлении, например, продвигаясь по узкому оврагу с крутыми склонами, прыгая с одной его стороны на другую. Напротив, при небольшом шаге и верном направлении потребуется очень много итераций. На практике величина шага берется пропорциональной крутизне склона, так что алгоритм замедляет ход вблизи минимума. Правильный выбор скорости обучения зависит от конкретной задачи и обычно делается опытным путем. Эта константа может также зависеть от времени, уменьшаясь по мере продвижения алгоритма.

Обычно этот алгоритм видоизменяется таким образом, чтобы включать слагаемое импульса (или инерции). Это способствует продвижению в фиксированном направлении, поэтому, если было сделано несколько шагов в одном и том же направлении, то алгоритм увеличивает скорость, что иногда позволяет избежать локального минимума, а также быстрее проходить плоские участки.

На каждом шаге алгоритма на вход сети поочередно подаются все обучающие примеры, реальные выходные значения сети сравниваются с требуемыми значениями, и вычисляется ошибка. Значение ошибки, а также градиента поверхности ошибок исполь-

зуется для корректировки весов, после чего все действия повторяются. Процесс обучения прекращается либо когда пройдено определенное количество эпох, либо когда ошибка достигнет некоторого определенного малого уровня, либо когда ошибка перестанет уменьшаться.

Рассмотрим проблемы обобщения и переобучения нейронной сети более подробно. Обобщение – это способность нейронной сети делать точный прогноз на данных, не принадлежащих исходному обучающему множеству. Переобучение же представляет собой чрезмерно точную подгонку, которая имеет место, если алгоритм обучения работает слишком долго, а сеть слишком сложна для такой задачи или для имеющегося объема данных.

Продемонстрируем проблемы обобщения и переобучения на примере аппроксимации некоторой зависимости не нейронной сетью, а посредством полиномов, при этом суть явления будет абсолютно та же.

Графики полиномов могут иметь различную форму, причем, чем выше степень и число членов, тем более сложной может быть эта форма. Для исходных данных можно подобрать полиномиальную кривую (модель) и получить, таким образом, объяснение имеющейся зависимости. Данные могут быть зашумлены, поэтому нельзя считать, что лучшая модель в точности проходит через все имеющиеся точки. Полином низкого порядка может лучше объяснять имеющуюся зависимость, однако, быть недостаточно гибким средством для аппроксимации данных, в то время как полином высокого порядка может оказаться чересчур гибким, но будет точно следовать данным, принимая при этом замысловатую форму, не имеющую никакого отношения к настоящей зависимости.

Нейронные сети сталкивается с такими же трудностями. Сети с большим числом весов моделируют более сложные функции и, следовательно, склонны к переобучению. Сети же с небольшим числом весов могут оказаться недостаточно гибкими, чтобы смоделировать имеющиеся зависимости. Например, сеть без скрытых слоев моделирует лишь обычную линейную функцию.

Как же выбрать правильную степень сложности сети? Почти всегда более сложная сеть дает меньшую ошибку, но это может свидетельствовать не о хорошем качестве модели, а о переобучении сети.

Выход состоит в использовании контрольной *кросс-проверки*. Для этого резервируется часть обучающей выборки, которая используется не для обучения сети по алгоритму обратного распространения ошибки, а для независимого контроля результата в ходе алгоритма. В начале работы ошибка сети на обучающем и

контрольном множествах будет одинаковой. По мере обучения сети ошибка обучения убывает, как и ошибка на контрольном множестве. Если же контрольная ошибка перестала убывать или даже стала расти, это указывает на то, что сеть начала слишком близко аппроксимировать данные (переобучилась) и обучение следует остановить. Если это случилось, то следует уменьшить число скрытых элементов и/или слоев, ибо сеть является слишком мощной для данной задачи. Если же обе ошибки (обучения и кросс-проверки) не достигнут достаточного малого уровня, то переобучения, естественно не произошло, а сеть, напротив, является недостаточно мощной для моделирования имеющейся зависимости.

Описанные проблемы приводят к тому, что при практической работе с нейронными сетями приходится экспериментировать с большим числом различных сетей, порой обучая каждую из них по несколько раз и сравнивая полученные результаты. Главным показателем качества результата является здесь контрольная ошибка. При этом, в соответствии с общесистемным принципом, из двух сетей с приблизительно равными ошибками контроля имеет смысл выбрать ту, которая проще.

Необходимость многократных экспериментов приводит к тому, что контрольное множество начинает играть ключевую роль в выборе модели и становится частью процесса обучения. Тем самым ослабляется его роль как независимого критерия качества модели. При большом числе экспериментов есть большая вероятность выбрать удачную сеть, дающую хороший результат на контрольном множестве. Однако для того чтобы придать окончательной модели должную надежность, часто (когда объем обучающих примеров это позволяет) поступают следующим образом: резервируют *тестовое* множество примеров. Итоговая модель тестируется на данных из этого множества, чтобы убедиться, что результаты, достигнутые на обучающем и контрольном множествах примеров, реальны, а не являются артефактами процесса обучения. Разумеется, для того чтобы хорошо играть свою роль, тестовое множество должно быть использовано *только один раз*: если его использовать повторно для корректировки процесса обучения, то оно фактически превратится в контрольное множество.

С целью ускорения процесса обучения сети предложены многочисленные модификации алгоритма обратного распространения ошибки, связанные с использованием различных функций ошибки, процедур определения направления и величин шага.

1) *Функции ошибки*:

- интегральные функции ошибки по всей совокупности обучающих примеров;

- функции ошибки целых и дробных степеней.

2) *Процедуры определения величины шага на каждой итерации*:

- дихотомия;
- инерционные соотношения (см. выше);
- отжиг.

3) *Процедуры определения направления шага*:

- с использованием матрицы производных второго порядка (метод Ньютона);
- с использованием направлений на нескольких шагах (партан метод).

1.4.2. Обучение без учителя

Рассмотренный алгоритм обратного распространения ошибки подразумевает наличие некоего внешнего звена, предоставляющего нейронной сети, кроме входных, целевые выходные образы. Алгоритмы, основанные на подобной концепции, называются алгоритмами обучения с учителем. Для их успешного функционирования необходимо наличие экспертов, задающих на предварительном этапе для каждого входного образа эталонный выходной. Так как создание интеллектуальных систем базируется, во многом, на биологических прототипах, до сих пор не прекращается спор о том, можно ли считать алгоритмы обучения с учителем натуральными или же они полностью искусственны. Например, обучение человеческого мозга, на первый взгляд, происходит без учителя: на зрительные, слуховые, тактильные и прочие рецепторы поступает информация извне, и внутри нервной системы происходит некая самоорганизация. Однако, нельзя отрицать и того, что в жизни человека немало учителей (и в буквальном, и в переносном смысле), которые координируют внешние воздействия. Вместе в тем, чем бы ни закончился спор приверженцев этих концепций обучения, они обе имеют право на существование.

Главная черта, делающая обучение без учителя привлекательным, это его самостоятельность. Процесс обучения, как и в случае обучения с учителем, заключается в подстраивании весов синапсов. Некоторые алгоритмы, правда, изменяют и структуру сети, т. е. количество нейронов и их взаимосвязи, но такие преобразования правильнее назвать более широким термином – *самоорганизацией*. Очевидно, что подстройка весов синапсов может проводиться только на основании информации о состоянии нейронов и уже имеющихся весовых коэффициентов). На этом, в частности, по аналогии с известными принципами самоорганизации нервных клеток, построены алгоритмы обучения Хебба.

Сигнальный метод обучения Хебба заключается в изменении весов по следующему правилу:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha\, y_i^{(q-1)} y_j^{(q)}, \qquad (1.25)$$

где $y_i^{(q-1)}$ – выходное значение нейрона i слоя $(q-1)$, $y_j^{(q)}$ – выходное значение нейрона j слоя q; $w_{ij}(t)$ и $w_{ij}(t-1)$ – весовой коэффициент синапса, соединяющего эти нейроны, на итерациях t и $(t-1)$ соответственно; α – коэффициент скорости обучения. Здесь и далее, для общности, под q подразумевается произвольный слой сети.

При обучении по данному методу усиливаются связи между возбужденными нейронами.

Существует также и дифференциальный метод обучения Хебба, определяемый соотношением:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \left(y_i^{(q-1)}(t) - y_i^{(q-1)}(t-1)\right)\cdot\left(y_j^{(q)}(t) - y_j^{(q)}(t-1)\right). \qquad (1.26)$$

Здесь $y_i^{(q-1)}(t)$ и $y_i^{(q-1)}(t-1)$ – выходное значение нейрона i слоя $(q-1)$ соответственно на итерациях t и $(t-1)$; $y_j^{(q)}(t)$ и $y_j^{(q)}(t-1)$ – то же самое для нейрона j слоя q. Как видно из формулы (1.26), более интенсивно обучаются синапсы, соединяющие нейроны, выходы которых наиболее динамично изменились в сторону увеличения.

Полный алгоритм обучения с применением вышеприведенных формул будет выглядеть следующим образом.

ШАГ 1. На стадии инициализации всем весовым коэффициентам присваиваются небольшие случайные значения.

ШАГ 2. На входы сети подается входной образ, и сигналы возбуждения распространяются по всем слоям согласно принципам классических сетей прямого распространения (feedforward). При этом для каждого нейрона рассчитывается взвешенная сумма его входов, к которой затем применяется активационная функция нейрона, в результате чего получается его выходное значение $y_i^{(q)}$, $i = 1 \ldots L_q$, где L_q – число нейронов в слое q, $q = 1 \ldots Q$; Q – число слоев в сети.

ШАГ 3. На основании полученных выходных значений нейронов по формуле (1.25) или (1.26) проводится изменение весовых коэффициентов.

ШАГ 4. Цикл с шага 2, пока выходные значения сети не стабилизируются с заданной точностью. Применение этого способа определения момента завершения обучения, отличного от использовавшегося для сети обратного распространения, обусловлено тем, что подстраиваемые значения синапсов фактически не огра-

ничены. На шаге 2 цикла попеременно предъявляются все образы из входного набора.

Следует отметить, что вид откликов на каждый класс входных образов заранее неизвестен и представляет собой произвольное сочетание состояний нейронов выходного слоя, обусловленное случайным распределением весов на стадии инициализации. Вместе с тем, сеть способна обобщать схожие образы, относя их к одному классу. Тестирование обученной сети позволяет определить топологию классов в выходном слое. Для приведения откликов обученной сети к удобному представлению можно дополнить сеть одним слоем, который, например, по алгоритму обучения однослойного персептрона необходимо заставить отображать выходные реакции сети в требуемые образы.

Другой алгоритм обучения без учителя – алгоритм Кохонена – предусматривает подстройку синапсов на основании их значений на предыдущей итерации:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha\left(y_i^{(q-1)} - w_{ij}(t-1)\right). \qquad (1.27)$$

Из выражения (1.27) видно, что обучение сводится к минимизации разницы между входными сигналами нейрона, поступающими с выходов нейронов предыдущего слоя $y_i^{(q-1)}$, и весовыми коэффициентами его синапсов.

Полный алгоритм обучения имеет примерно такую же структуру, как в методах Хебба, но на шаге 3 из всего слоя выбирается нейрон, значения синапсов которого максимально походят на входной образ, и подстройка весов по формуле (1.27) проводится только для него. Эта, так называемая, аккредитация может сопровождаться торможением всех остальных нейронов слоя и введением выбранного нейрона в насыщение. Выбор такого нейрона может осуществляться, например, расчетом скалярного произведения вектора весовых коэффициентов с вектором входных значений. Максимальное произведение дает выигравший нейрон.

Другой вариант состоит в расчете расстояния между этими векторами в R-мерном пространстве:

$$D_j = \sqrt{\sum_{i=1}^{R}(y_i^{(q-1)} - w_{ij})^2}\,, \qquad (1.28)$$

где R – размерность векторов; j – индекс нейрона в слое q; i – индекс суммирования по нейронам слоя $(q-1)$; w_{ij} – вес синапса, соединяющего нейроны; выходы нейронов слоя $(q-1)$ являются входными значениями для слоя q.

В данном случае, побеждает нейрон с наименьшим расстоянием. Иногда слишком часто получающие аккредитацию нейроны

принудительно исключаются из рассмотрения, чтобы уравнять права всех нейронов слоя. Простейший вариант такого алгоритма заключается в торможении только что выигравшего нейрона.

С целью сокращения длительности процесса обучения при использовании алгоритма Кохонена существует практика нормализации входных образов и начальных значений весовых коэффициентов на стадии инициализации в соответствии с выражениями:

$$x'_i = x_i \Big/ \sqrt{\sum_{j=1}^{n} x_j^2}\,, \qquad (1.29)$$

$$w_o = \frac{1}{\sqrt{n}}. \qquad (1.30)$$

где x_i – i-й компонент вектора входного образа или вектора весовых коэффициентов, а n – его размерность.

Инициализация весовых коэффициентов случайными значениями может привести к тому, что различные классы, которым соответствуют плотно распределенные входные образы, сольются или, наоборот, раздробятся на дополнительные подклассы в случае близких образов одного и того же класса. Для избежания такой ситуации используется метод выпуклой комбинации. Суть его сводится к тому, что входные нормализованные образы подвергаются преобразованию:

$$x_i = \alpha(t)\,x_i + (1-\alpha(t))\frac{1}{\sqrt{n}}, \qquad (1.31)$$

где $\alpha(t)$ – коэффициент, изменяющийся в процессе обучения от нуля до единицы, в результате чего вначале на входы сети подаются практически одинаковые образы, а с течением времени они все больше сходятся к исходным.

После выбора из слоя q нейрона j с минимальным расстоянием D_j по формуле (1.28) производится обучение по формуле (1.27) не только этого нейрона, но и его соседей, расположенных в окрестности Ω. Величина Ω для первых итераций очень большая, так что обучаются все нейроны, но с течением времени она уменьшается до нуля. Таким образом, чем ближе окончание обучения, тем точнее определяется группа нейронов, отвечающих каждому классу образов.

На основе рассмотренного выше метода строятся *самоорганизующиеся нейронные сети* (self-organizing feature maps). Этот устоявшийся перевод с английского представляется не вполне удачным, так как речь идет не об изменении структуры сети, а только о подстройке синапсов нейронов.

1.5. Настройка числа нейронов в скрытых слоях многослойных нейронных сетей в процессе обучения

Для того, чтобы многослойная нейронная сеть реализовывала заданное обучающей выборкой отображение, она должна иметь достаточное число нейронов в скрытых слоях. В настоящее время нет формул для точного определения необходимого числа нейронов в сети по заданной обучающей выборке. Однако предложены способы настройки числа нейронов в процессе обучения, которые обеспечивают построение нейронной сети для решения задачи и дают возможность избежать избыточности. Эти способы настройки можно разделить на две группы: *алгоритмы сокращения* (pruning algorithms) и *конструктивные алгоритмы* (constructive algorithms).

1.5.1. Алгоритмы сокращения

В начале работы алгоритма обучения с сокращением число нейронов в скрытых слоях сети заведомо избыточно. Затем из нейронной сети постепенно удаляются синапсы и нейроны. Существуют два подхода к реализации алгоритмов сокращения.

1) *Метод штрафных функций.*

В целевую функцию алгоритма обучения вводится штрафы за то, что значения синаптических весов отличны от нуля, например:

$$C = \sum_{i=1}^{L}\sum_{j=1}^{n} w_{ij}^2, \qquad (1.32)$$

где w_{ij} – синаптический вес, i – номер нейрона, j – номер входа; L – число нейронов скрытого слоя, n – размерность входного сигнала.

2) *Метод проекций.*

Синаптический вес обнуляется, если его значение попало в заданный диапазон:

$$w_{ij} = \begin{cases} 0, & |w_{ij}| \le \varepsilon, \\ w_{ij}, & |w_{ij}| > \varepsilon, \end{cases} \qquad (1.33)$$

где ε – некоторая константа.

Алгоритмы сокращения имеют, по крайней мере, два недостатка:

- нет методики определения числа нейронов скрытых слоев, которое является избыточным, поэтому перед началом работы алгоритма нужно угадать это число;

- в процессе работы алгоритма сеть содержит избыточное число нейронов, поэтому обучение идет медленно.

1.5.2. Конструктивные алгоритмы

Предшественником конструктивных алгоритмов можно считать методику обучения многослойных нейронных сетей, включающую в себя следующие шаги:

ШАГ 1. Выбор начального числа нейронов в скрытых слоях.

ШАГ 2. Инициализация сети, заключающаяся в присваивании синаптическим весам и смещениям сети случайных значений из заданного диапазона.

ШАГ 3. Обучение сети по заданной выборке.

ШАГ 4. Завершение в случае успешного обучения. Если сеть обучить не удалось, то число нейронов увеличивается и повторяются шаги со второго по четвертый.

В конструктивных алгоритмах число нейронов в скрытых слоях также изначально мало и постепенно увеличивается. В отличие от этой методики, в конструктивных алгоритмах сохраняются навыки, приобретенные сетью до увеличения числа нейронов.

Конструктивные алгоритмы различаются правилами задания значений параметров в новых, добавленных в сеть, нейронах:

- значения параметров являются случайными числами из заданного диапазона;
- значения синаптических весов нового нейрона определяются путем *расщепления* (splitting) одного из старых нейронов.

Первое правило не требует значительных вычислений, однако его использование приводит к некоторому увеличению значения функции ошибки после каждого добавления нового нейрона. В результате случайного задания значений параметров новых нейронов может появиться избыточность в числе нейронов скрытого слоя. Расщепление нейронов лишено двух указанных недостатков. Суть алгоритма расщепления проиллюстрирована на рис. 1.7.

На этом рисунке показан вектор весов нейрона скрытого слоя на некотором шаге обучения и векторы изменения весов, соответствующие отдельным обучающим примерам. Векторы изменений имеют два преимущественных направления и образуют в пространстве область, существенно отличающуюся от сферической. Суть алгоритма заключается в выявлении и расщеплении таких нейронов. В результате расщепления вместо одного исходного в сети оказывается два нейрона. Первый из этих нейронов имеет вектор весов, представляющий из себя сумму вектора весов исходного нейрона и векторов изменений весов одного из преиму-

1.5. Настройка числа нейронов в скрытых слоях многослойных нейронных сетей в процессе обучения

Для того, чтобы многослойная нейронная сеть реализовывала заданное обучающей выборкой отображение, она должна иметь достаточное число нейронов в скрытых слоях. В настоящее время нет формул для точного определения необходимого числа нейронов в сети по заданной обучающей выборке. Однако предложены способы настройки числа нейронов в процессе обучения, которые обеспечивают построение нейронной сети для решения задачи и дают возможность избежать избыточности. Эти способы настройки можно разделить на две группы: *алгоритмы сокращения* (pruning algorithms) и *конструктивные алгоритмы* (constructive algorithms).

1.5.1. Алгоритмы сокращения

В начале работы алгоритма обучения с сокращением число нейронов в скрытых слоях сети заведомо избыточно. Затем из нейронной сети постепенно удаляются синапсы и нейроны. Существуют два подхода к реализации алгоритмов сокращения.

1) *Метод штрафных функций.*

В целевую функцию алгоритма обучения вводится штрафы за то, что значения синаптических весов отличны от нуля, например:

$$C = \sum_{i=1}^{L}\sum_{j=1}^{n} w_{ij}^2, \qquad (1.32)$$

где w_{ij} – синаптический вес, i – номер нейрона, j – номер входа; L – число нейронов скрытого слоя, n – размерность входного сигнала.

2) *Метод проекций.*

Синаптический вес обнуляется, если его значение попало в заданный диапазон:

$$w_{ij} = \begin{cases} 0, & |w_{ij}| \le \varepsilon, \\ w_{ij}, & |w_{ij}| > \varepsilon, \end{cases} \qquad (1.33)$$

где ε – некоторая константа.

Алгоритмы сокращения имеют, по крайней мере, два недостатка:

- нет методики определения числа нейронов скрытых слоев, которое является избыточным, поэтому перед началом работы алгоритма нужно угадать это число;

- в процессе работы алгоритма сеть содержит избыточное число нейронов, поэтому обучение идет медленно.

1.5.2. Конструктивные алгоритмы

Предшественником конструктивных алгоритмов можно считать методику обучения многослойных нейронных сетей, включающую в себя следующие шаги:

ШАГ 1. Выбор начального числа нейронов в скрытых слоях.

ШАГ 2. Инициализация сети, заключающаяся в присваивании синаптическим весам и смещениям сети случайных значений из заданного диапазона.

ШАГ 3. Обучение сети по заданной выборке.

ШАГ 4. Завершение в случае успешного обучения. Если сеть обучить не удалось, то число нейронов увеличивается и повторяются шаги со второго по четвертый.

В конструктивных алгоритмах число нейронов в скрытых слоях также изначально мало и постепенно увеличивается. В отличие от этой методики, в конструктивных алгоритмах сохраняются навыки, приобретенные сетью до увеличения числа нейронов.

Конструктивные алгоритмы различаются правилами задания значений параметров в новых, добавленных в сеть, нейронах:

- значения параметров являются случайными числами из заданного диапазона;
- значения синаптических весов нового нейрона определяются путем *расщепления* (splitting) одного из старых нейронов.

Первое правило не требует значительных вычислений, однако его использование приводит к некоторому увеличению значения функции ошибки после каждого добавления нового нейрона. В результате случайного задания значений параметров новых нейронов может появиться избыточность в числе нейронов скрытого слоя. Расщепление нейронов лишено двух указанных недостатков. Суть алгоритма расщепления проиллюстрирована на рис. 1.7.

На этом рисунке показан вектор весов нейрона скрытого слоя на некотором шаге обучения и векторы изменения весов, соответствующие отдельным обучающим примерам. Векторы изменений имеют два преимущественных направления и образуют в пространстве область, существенно отличающуюся от сферической. Суть алгоритма заключается в выявлении и расщеплении таких нейронов. В результате расщепления вместо одного исходного в сети оказывается два нейрона. Первый из этих нейронов имеет вектор весов, представляющий из себя сумму вектора весов исходного нейрона и векторов изменений весов одного из преиму-

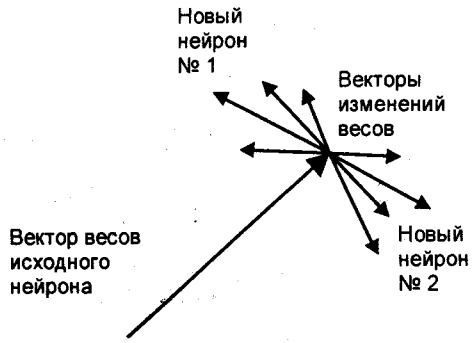


Рис. 1.7. Вектор весов нейрона скрытого слоя и изменения, соответствующие отдельным обучающим примерам

щественных направлений. В результате суммирования векторов изменений весов другого преимущественного направления и вектора весов исходного нейрона получают синаптические веса второго нового нейрона.

Расщеплять нейроны, векторы изменений которых имеют два преимущественных направления, необходимо потому, что наличие таких нейронов приводит к осцилляциям при обучении методом обратного распространения. При обучении методом с интегральной функцией ошибки наличие таких нейронов приводит к попаданию в локальный минимум с большим значением ошибки.

Алгоритм расщепления включает в себя:

- построение ковариационной матрицы векторов изменений синаптических весов;
- вычисление собственных векторов и собственных значений полученной матрицы с помощью итерационного алгоритма Ойа (Oja), в соответствии с которым выполняется стохастический градиентный подъем и ортогонализация Грамма–Шмидта.

Недостатком алгоритма является экспоненциальный рост времени вычислений при увеличении размерности сети.

Для преодоления указанного недостатка предложен *упрощенный алгоритм расщепления*, который не требует значительных вычислений. Общее число нейронов в сети, построенной с помощью этого алгоритма по заданной обучающей выборке, может быть несколько больше, чем у сети, построенной с помощью исходного алгоритма.

В упрощенном алгоритме для расщепления выбирается нейрон с наибольшим значением функционала:

$$F_i = \frac{\sum_{k=1}^{N} \left| \delta w_i^k \right|}{\left| \sum_{k=1}^{N} \delta w_i^k \right|}, \qquad (1.34)$$

где δw – вектор изменений синаптических весов нейрона; i – номер нейрона; $i = 1 \ldots L$, L – число нейронов; k – номер обучающего примера; N – число примеров в обучающей выборке.

Таким образом, в качестве критерия выбора нейрона для расщепления используется отношение суммы длин векторов изменений синаптических весов нейрона, соответствующих различным обучающим примерам, к длине суммы этих векторов.

В результате расщепления вместо исходного нейрона в сеть вводятся два новых нейрона. Значение каждого синаптического веса нового нейрона есть значение соответствующего веса старого нейрона плюс некоторый небольшой шум. Величины весов связей выходов новых нейронов и нейронов следующего слоя равны половине величин весов связей исходного нейрона с соответствующими весами следующего слоя. Упрощенный алгоритм, как и исходный, гарантирует, что функция ошибки после расщепления увеличиваться не будет.

Кроме описанных способов выбора нейронов для расщепления, может быть использован анализ чувствительности, в процессе которого строятся матрицы Гессе для вторых производных функции ошибки по параметрам сети. По величине модуля второй производной судят о важности значения данного параметра для решения задачи. Параметры с малыми значениями вторых производных обнуляют. Анализ чувствительности требует больших вычислительных ресурсов.

Дополнительная информация об алгоритмах обучения приведена в приложении.

1.6. Краткое обобщение материалов главы

1.6.1. Как построить нейронную сеть

Практически любую задачу можно свести к задаче, решаемой нейронной сетью. В табл. 1.2 показано, каким образом следует сформулировать в терминах нейронной сети задачу распознавания рукописных букв.

Таблица 1.2

Задача распознавания рукописных букв в терминах нейронной сети

Задача распознавания рукописных букв	
Дано: растровое черно-белое изображение буквы размером 30x30 пикселов	*Необходимо*: распознать предъявленную букву (в алфавите 33 буквы)
Формулировка задачи для нейронной сети	
Дано: входной вектор из 900 двоичных символов (900 = 30x30)	*Необходимо*: построить нейронную сеть с 900 входами и 33 выходами, которые помечены буквами. Номер нейрона с максимальным значением выходного сигнала должен соответствовать предъявленному изображению буквы.

Интерес представляет не аналоговое значение выхода, а номер класса (номер буквы в алфавите). Поэтому каждому классу сопоставляется свой выходной нейрон, а ответом сети считается номер класса, соответствующий нейрону, на чьем выходе уровень сигнала максимален. Значение же уровня сигнала на выходе характеризует достоверность того, что на вход была подана соответствующая рукописная буква.

Нейронная сеть строится в два этапа.

1) Выбор типа (архитектуры) сети.

2) Подбор весов (обучение) сети.

На первом этапе необходимо определить следующее:

- какие нейроны использовать (число входов, функции активации);
- каким образом следует соединить нейроны между собой;
- что взять в качестве входов и выходов сети.

Нет необходимости придумывать нейронную сеть «с нуля», так как существуют несколько десятков различных нейросетевых архитектур, причем эффективность многих из них доказана математически. Наиболее популярные и изученные из них – это многослойный персептрон, нейронная сеть с общей регрессией, сети Кохонена и другие, которые будут рассмотрены ниже.

На втором этапе производится обучение выбранной сети посредством настройки ее весов. Количество весов может быть велико, поэтому обучение представляет собой сложный и длительный процесс. Для многих архитектур разработаны специальные алгоритмы обучения, наиболее популярный из которых алгоритм обратного распространения ошибки.

1.6.2. Обучение нейронной сети

Обучить нейронную сеть это значит, сообщить ей, чего от нее добиваются. Этот процесс похож на обучение ребенка алфавиту. Показав ребенку изображение буквы и получив неверный ответ, ему сообщается тот ответ, который хотят получить. Ребенок запоминает этот пример вместе с верным ответом и в его памяти происходят некоторые изменения в нужном направлении.

Процесс обучения заключается в последовательном предъявлении букв (рис. 1.8).

При предъявлении изображения каждой буквы на входе сеть выдает некоторый ответ, не обязательно верный. Известен и верный (желаемый) ответ. В данном случае желательно, чтобы на выходе соответствующего нейрона уровень сигнала был максимален. Обычно в качестве желаемого в задаче классификации берут набор, где «1» стоит на выходе этого нейрона, а «0» – на выходах всех остальных нейронов. Одну и ту же букву (а также различные изображения одной и той же буквы) можно предъявлять сети много раз.

После многократного предъявления примеров веса сети стабилизируются, причем сеть дает правильные ответы на все (или почти все) примеры из базы данных. В таком случае говорят, что сеть обучена. В программных реализациях можно видеть, что в процессе обучения величина ошибки (сумма квадратов ошибок по всем выходам) постепенно уменьшается. Когда величина ошибки

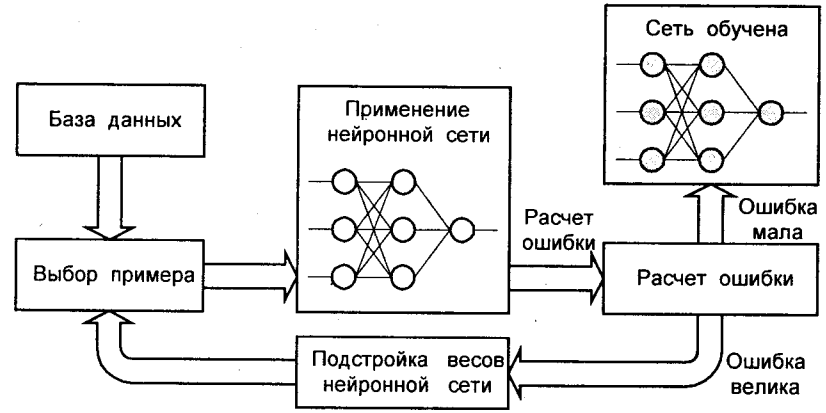


Рис. 1.8. Процесс обучения нейронной сети

достигает нуля или приемлемо малого уровня, обучение останавливают, и сеть готова к распознаванию.

Важно отметить, что вся информация, которую сеть приобретает о задаче, содержится в наборе примеров. Поэтому качество обучения сети зависит от количества примеров в обучающей выборке, а также от того, насколько полно эти примеры описывают задачу. Считается, что для полноценной тренировки требуется хотя бы несколько десятков (а лучше сотен) примеров.

1.6.3. Применение обученной нейронной сети

Важнейшая особенность человеческого мозга состоит в том, что, однажды обучившись определенному процессу, он может верно действовать и в тех ситуациях, которым он не обучался. Так же и обученная нейронная сеть может с большой вероятностью правильно реагировать на новые, не предъявленные ей ранее данные. Например, можно нарисовать букву другим почерком, а затем предложить нейронной сети классифицировать новое изображение. Веса обученной сети хранят достаточно много информации о сходстве и различиях букв, поэтому можно рассчитывать на правильный ответ и для нового варианта изображения.

Отметим, что задачи классификации (типа распознавания букв) очень плохо алгоритмизируются. Если в случае распознавания букв верный ответ очевиден заранее, то в более сложных практических задачах обученная нейронная сеть выступает как эксперт, обладающий большим опытом и способный дать ответ на трудный вопрос. Примером такой задачи служит медицинская диагностика, где сеть может учитывать большое количество числовых параметров (энцефалограмма, давление, вес). Конечно, «мнение» сети в этом случае нельзя считать окончательным.

Классификация предприятий по степени их перспективности – это уже привычный способ использования нейронных сетей в практике западных компаний. При этом сеть также использует множество экономических показателей, сложным образом связанных между собой.

Нейросетевой подход особенно эффективен в задачах экспертной оценки по той причине, что он сочетает в себе способность компьютера к обработке чисел и способность мозга к обобщению и распознаванию. Говорят, что у хорошего врача способность к распознаванию в своей области столь велика, что он может провести приблизительную диагностику уже по внешнему виду пациента. Можно согласиться также, что опытный трейдер чувствует направление движения рынка по виду графика. Однако в первом

случае все факторы наглядны, т. е. характеристики пациента мгновенно воспринимаются мозгом как «бледное лицо», «блеск в глазах». Во втором же случае учитывается только один фактор – курс за определенный период времени. Нейронная сеть позволяет обрабатывать огромное количество факторов (до нескольких тысяч), независимо от их наглядности. Это универсальный «хороший врач», который может поставить свой диагноз в любой области.

Помимо задач классификации, нейронные сети широко используются для поиска зависимостей в данных и кластеризации. Кластеризация – это разбиение набора примеров на несколько компактных областей (кластеров), причем число кластеров заранее неизвестно. Кластеризация позволяет представить неоднородные данные в более наглядном виде и использовать далее для исследования каждого кластера различные методы. Например, нейронная сеть на основе методики использования МГУА (метода группового учета аргументов) позволяет по обучающей выборке построить зависимость одного параметра от других в виде полинома. Такая сеть может не только мгновенно выучить таблицу умножения, но и найти сложные скрытые зависимости в данных (например, финансовых), которые не обнаруживаются стандартными статистическими методами, быстро выявить фальсифицированные страховые случаи или недобросовестные предприятия.

Особенно важны для практики, в частности, для финансовых приложений, задачи прогнозирования, поэтому поясним способы применения нейронных сетей в этой области более подробно.

Рассмотрим задачу прогнозирования курса акций на день вперед. Пусть имеется база данных, содержащая значения курса за последние 300 дней. Построим прогноз завтрашней цены на основе курсов за последние несколько дней. Понятно, что прогнозирующая нейронная сеть должна иметь всего один выход и столько входов, сколько предыдущих значений мы хотим использовать для прогноза, например, четыре последних значения. Составить обучающий пример очень просто, входными значениями будут курсы за четыре последних дня, а желаемым выходом – известный курс в следующий за ними день.

Если нейронная сеть совместима с какой-либо системой обработки электронных таблиц (например, Excel), то подготовка обучающей выборки состоит из следующих операций:

- занести значения курса акций последовательно в столбец таблицы;
- скопировать значения котировок в 4 соседних столбца;

Входы				Выход
1.81	3.21	3.22	1.90	0.48
3.21	3.22	1.90	0.48	0.29
3.22	1.90	0.48	0.29	1.59
1.90	0.48	0.29	1.59	3.41
0.48	0.29	1.59	3.41	**4.42**
0.29	1.59	3.41	4.42	4.10
1.59	3.41	4.42	4.10	**3.10**
3.41	4.42	4.10	3.10	2.49
4.42	4.10	3.10	2.49	2.80
4.10	3.10	2.49	2.80	3.61
3.10	2.49	2.80	3.61	4.20
2.49	2.80	3.61	4.20	4.30
2.80	3.61	4.20	4.30	4.29
3.61	4.20	4.30	4.29	4.69
4.20	4.30	4.29	4.69	5.41
4.30	4.29	4.69	5.41	5.82
4.29	4.69	5.41	5.82	5.41

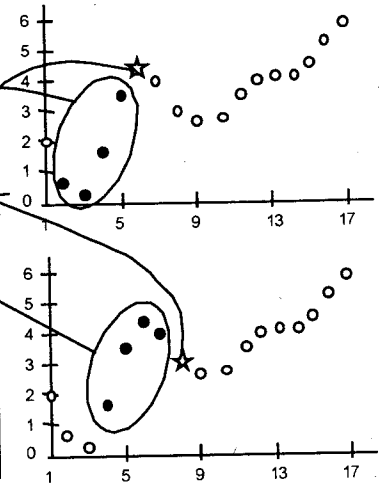


Рис. 1.9. Подготовка данных для нейронной сети в Excel

• сдвинуть второй столбец на 1 ячейку вверх, третий столбец – на 2 и т. д. (рис.1.9).

Смысл этой подготовки состоит в том, что каждая строка таблицы теперь представляет собой обучающий пример, где первые четыре числа – входные значения сети, а пятое число – желаемое значение выхода. Исключение составляют последние четыре строки, где данных недостаточно. Поэтому эти строки не учитываются при обучении. Заметим, что в четвертой снизу строке заданы все четыре входных значения, но неизвестно значение выхода. Именно при применении к этой строке обученной сети и можно получить прогноз на следующий день.

Как видно из этого примера, объем обучающей выборки зависит от выбранного количества входов. Если сделать 299 входов, то такая сеть потенциально могла бы строить лучший прогноз, чем сеть с 4 входами, однако в этом случае имеется всего один обучающий пример, и обучение бессмысленно. Это следует учитывать при выборе числа входов, выбирая разумный компромисс между глубиной предсказания (число входов) и качеством обучения (объем обучающей выборки).

Укажем в заключение, что ряд практических примеров использования нейронных сетей приведен в третьей части книги.

Глава 2

ОСНОВНЫЕ КОНЦЕПЦИИ НЕЙРОННЫХ СЕТЕЙ

Кроме рассмотренных выше многослойных нейронных сетей, обучаемых по алгоритму обратного распространения ошибки, известно много разновидностей специфических нейронных сетей, реализующих различные свойства биологических систем и, прежде всего, свойства ассоциативной памяти.

2.1. Ассоциативная память нейронных сетей

В биологических системах, обладающих памятью, изменение нервной активности системы под влиянием внешних раздражителей зависит от воздействия предшествующих событий и от информации, хранящейся в памяти. Процесс запоминания связан с образованием следа (узора, энграммы) в мозговых структурах. Поток нервных импульсов, несущих информацию о запоминаемом объекте, проходит через нейронные сети, возбуждая на своем пути нервные клетки, из которых формируется нейронный след. Проторенный нервный путь обладает меньшим сопротивлением по отношению к другим возможным путям. Повышение производительности возникшего нейронного следа возможно вследствие свойства нейронов достаточно быстро адаптироваться к повторно проходящим нервным импульсам. Механизмы памяти обеспечивают длительное сохранение увеличенной проводимости нейронов, вовлеченных в образованный узор.

Следы памяти, хранящие образы объектов, отражены в сложных параллельно-последовательных нейронных сетях, обладающих большой избыточностью. В организованных случайным образом нейронных сетях следы памяти распределяются по пространству мозга также случайно.

Ситуация еще больше усложняется тем, что одни и те же нейроны участвуют в хранении образов различных запоминаемых

объектов. Это значит, что след, возникающий при запоминании одного объекта информации, может иметь общие звенья нейронной сети со следами от других объектов. Поэтому нельзя определенно указать, в каком участке мозга будет находиться след конкретного объекта информации – образа.

В этом случае механизм доступа к информации базируется не на указании места хранения информации в логико-запоминающей среде, а на анализе свойств самой искомой информации.

В биологических системах обработки данных таким механизмом является механизм ассоциаций. Впервые термин «ассоциация» был введен Дж. Локком в 1698 г. и определен как «связь, возникающая при определенных условиях между двумя или более психическими образованиями – ощущениями, актами, восприятиями, идеями».

Применительно к системам обработки данных, в том числе к искусственным нейронным сетям, ассоциация трактуется как взаимосвязь между информацией (образом) на входе логико-запоминающей среды и информацией (образом), хранящейся в логико-запоминающей среде. Способ доступа к информации в запоминающей среде, базирующийся на механизме ассоциации, получил название *ассоциативного способа доступа*. Ассоциативный способ доступа к информации обеспечивает:

- практически одновременный доступ ко всей хранящейся в памяти информации;
- относительную независимость времени поиска информации от емкости памяти;
- внесение элементов обработки информации непосредственно в процесс самого доступа;
- обработку информации непосредственно в среде ее хранения.

Эти, а также ряд других отличительных особенностей ассоциативного способа доступа к информации делает его чрезвычайно перспективным в системах обработки данных.

2.1.1. Ассоциации

Существуют различные концепции ассоциативной памяти. Однако все они предполагает наличие следующих элементов:

- логико-запоминающей среды, являющейся носителем информации;
- множества записанных в памяти информационных объектов;

- структуры взаимосвязей между информационными объектами;
- механизма информационных взаимодействий в логико-запоминающей среде.

Вышеперечисленные элементы концепции ассоциативной памяти позволяют определить основной подход, в соответствии с которым ассоциации между информационными объектами (образами) трактуются как некая абстрактная структура взаимозависимостей (отношений), неявно закодированная в информационных объектах и в соответствующих связях между ними или в формах их представлений.

Эти отношения содержат компоненты двух типов, первые из которых задают сами информационные объекты, а вторые – вид отношений. Признаки этих отношений могут характеризовать свойства объектов, действия над ними, подчиненность, временные признаки и т. д. Очевидно, что такое представление ассоциации – наиболее простая конструкция, на основе которой можно построить структуры отношений произвольной сложности.

В зависимости от условий формирования отношений между ассоциируемыми объектами, ассоциации могут устанавливаться по критериям сходства объектов, контраста, по смежности проявления объектов во времени или в пространстве, а также в рамках определенной совокупности свойств, например, при обеспечении заданных пространственно-временных соотношений.

Логика взаимодействий между информационными объектами может быть реализована на основе двух концепций:

- формирование отношений ассоциаций между однородными объектами;
- концепция «составного отношения», где отношения ассоциаций устанавливаются между качественно различными объектами.

Вид связи между ассоциируемыми объектами может быть различным:

- символьная – связь между объектами устанавливается «на основании соглашения»;
- индексная – связь ассоциируется в силу существующих отношений между объектами;
- иконическая – связь между объектами устанавливается на основе фактического сходства;
- гибридная – сочетает особенности различных видов связи.

В зависимости от степени соответствия ассоциируемых объектов различают *автоассоциации* и *гетероассоциации*. Автоассоциации реализуются при условии соответствия соотносимых объектов. Причем объект отыскивается по его произвольным частям, имеющим большую или меньшую корреляцию с искомым объектом, или по его фрагментам в случае, если они достаточны для того, чтобы отличить этот объект от остальных. Для гетероассоциаций характерно то, что инициируемый объект структурно не соответствует любому из поисковых объектов и формируется как ответ на специфический ключевой объект.

Ассоциируемые объекты могут быть представлены либо прямыми, либо косвенными (непрямыми) ассоциациями. Прямые ассоциации по способу представления в виде функциональной зависимости могут инициироваться либо логически детерминированной последовательностью, либо на основе ассоциативной связи посредством прямых указателей. Представление непрямых ассоциаций в виде функциональной зависимости образуется на основе ассоциативных связей посредством перекрестных ссылок.

В контексте трактовки понятия ассоциации как структуры взаимозависимостей между информационными объектами оно отражает наличие взаимосвязей между данными и не имеет отношения к самому механизму хранения информации.

В рамках подхода, рассматривающего ассоциации как коллективные или интегральные изменения в нейронной сети (логико-запоминающей среде), ассоциативные свойства сети могут быть рассмотрены:

- во-первых, с точки зрения возможности коллективного доступа ко всей распределенной в нейронной сети информации, а также параллельной обработки и одновременного преобразования всех данных непосредственно в нейронной среде;
- во-вторых, с точки зрения практической реализации отношений между размещенными в нейронной сети, ассоциируемыми информационными объектами. В этом случае свойства сети могут существенным образом влиять на интерпретацию отношений между информационными объектами, позволяя по-новому подойти к исследованию этих объектов и взаимодействиям между ними.

Именно этот подход позволяет выделить типы нейронных сетей, эффективно используемых для реализации различных задач ассоциативной памяти.

2.1.2. Модели ассоциативной памяти

Ассоциативная память может быть определена как система для записи, хранения, поиска, обработки и считывания информации, в которой данные (знания) об объекте могут быть инициализированы по заданному фрагменту этих данных (знаний), используемому в качестве поискового.

Исходя из этого определения, можно сформулировать решаемые ассоциативной памятью задачи:

- соотнесение поисковой информации с хранимой и дополнение ее (инициализация) до точного описания объекта, т. е. всей информации, которая доступна ассоциативной памяти;
- фильтрация (коррекция) поисковой информации относительно всего объема хранимой в ассоциативной памяти информации, выделение недостоверной и на основании оставшейся решение первой задачи.

Процессы, аналогичные биологическим механизмам запоминания и обработки информации, можно представить с помощью различных моделей ассоциативной памяти, позволяющих отобразить отношения (ассоциации) произвольной сложности между информационными объектами. Однако все эти отношения могут быть реализованы в виде простых конструкций – троек компонентов: упорядоченной пары информационных объектов O и V, и типа отношения A: $O \xleftrightarrow{A} V$. Одна из простейших моделей ассоциативной памяти для отображения таких отношений показана на рис. 2.1.

Модель состоит из ассоциативной логико-запоминающей среды (нейронной сети), связанной с двумя каналами ввода и одним каналом вывода информации.

На этапе записи (обучения) из первого канала ввода на вход K подается входная информация, а по второму каналу – признаковая информация C, представляющая контекст, в котором входная информация записывается в память.

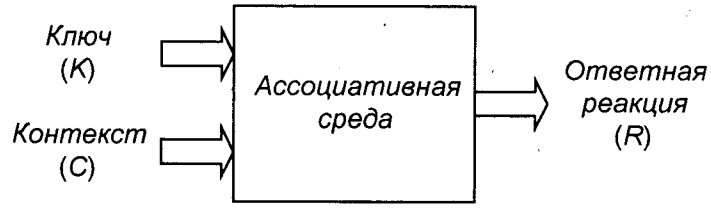


Рис. 2.1. Модель ассоциативной памяти

Рис. 2.2. Модель ассоциативной памяти с обратной связью

На этапе функционирования (считывания) при появлении ключа K (или его фрагмента) на выходе памяти формируется ответная реакция R, связанная с ключом K. Таким образом, записанная в память информация может быть получена с использованием любых ее фрагментов, используемых в качестве поисковых. Задавая различный контекст C, можно точнее конкретизировать информацию, которую необходимо получить.

Рассмотрим на примере, каким образом должен быть организован процесс накопления и поиска информации в ассоциативной памяти, чтобы обеспечить цикличность процесса, при котором выбранный элемент информации становится ключом для поиска новой информации.

По трем входным каналам одновременно могут вводиться наборы значений. По первому каналу в момент времени t подается адресная информация $K(t)$, а по второму – признак $C(t)$. Отклик $R(t)$ по каналу обратной связи подается также на вход ассоциативной среды. Выходной канал служит для выдачи информации.

При функционировании такой ассоциативной памяти ключи $K(t)$ и признаки $C(t)$ подаются через интервалы времени, соответствующие задержке канала обратной связи.

Процесс работы памяти будем рассматривать в предположении, что тройка $[K(t), C(t), R(t-\Delta)]$ представляет собой единый статический образ, заданный в момент времени t, причем возможна его одновременная запись в память за одну операцию. Допустим также, что на этапе записи $R(t)$ и $K(t)$ одинаковы.

На этапе записи на входы ассоциативной памяти поступают $K(t)$ и $C(t)$, при этом на выходе формируется $R(t)$, идентичный $K(t)$. После этого с задержкой Δ на входе формируется $R(t-\Delta)$. Каждая новая тройка, появляющаяся на входах, записывается в память.

На этапе получения данных из ассоциативной памяти на вход подается ключ K, связанный с контекстной информацией C, после чего K можно снять с входа. В результате на выходе в качестве отклика появляется копия K. Когда на входе памяти появится задержанный сигнал $R(t-\Delta)$, новым ключевым признаком становится пара (C, R), приводящая появлению на выходе следующего образа $R(t)$ и т. д. Таким образом, выбирается вся записанная последовательность образов вместе с контекстной информацией.

Рассмотренная системная модель реализует ассоциативную память, пригодную для записи и выборки структурированных знаний.

Модели ассоциативной памяти, реализуемые нейронными сетями, могут быть гораздо сложнее. Кроме того, память может иметь несколько входов и выходов, состоять из нескольких подсистем. Данные в одном канале могут порождать контекстную информацию для другого канала. Этап записи может выполняться отдельно от выборки или быть совмещенным и т. д.

2.2. Персептроны

Систематическое изучение искусственных нейронных сетей было начато Маккалохом и Питтсом в 1943 году. Позднее они исследовали нейросетевые парадигмы для распознавания изображений, подвергаемых сдвигам и поворотам, используя при этом нейронную модель, показанную на рис. 2.3. Элемент Σ умножает каждый вход x_i на вес w_i и суммирует взвешенные входы. Если эта сумма больше заданного порогового значения, выход равен единице, в противном случае – нулю. Эти системы (и множество им подобных) получили название *персептронов*. Они состоят из одного слоя искусственных нейронов, соединенных с помощью весовых коэффициентов с множеством входов (рис. 2.4), хотя в прин-

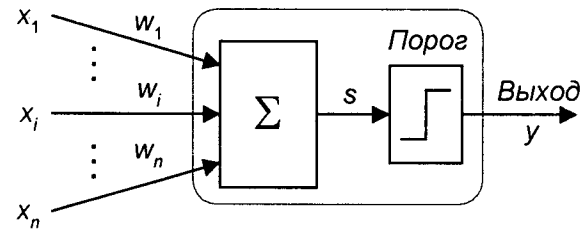


Рис. 2.3. Персептронный нейрон

ципе описываются и более сложные системы.

В 60-е годы персептроны вызвали большой интерес. Розенблатт доказал теорему об обучении персептронов. Уидроу продемонстрировал возможности систем персептронного типа. Однако дальнейшие исследования показали, что персептроны не способны обучиться решению ряда простых задач. Минский строго проанализировал эту проблему и показал, что существуют жесткие ограничения на то, что могут выполнять однослойные персептроны, и, следовательно, на то, чему они могут обучаться. Так как в то время методы обучения многослойных сетей не были известны, исследования в области нейронных сетей пришли в упадок. Возрождение интереса к нейронным сетям связано в большей степени со сравнительно недавним открытием с таких методов.

Работа Минского возможно и охладила пыл первых исследователей нейронных сетей, однако обеспечила необходимое время для развития лежащей в их основе теории. Важно отметить, что анализ Минского не был опровергнут и до сих пор остается весьма существенным.

Несмотря на ограничения, персептроны широко изучались. Теория персептронов является основой для изучения многих других типов искусственных нейронных сетей.

Рассмотрим в качестве примера трехнейронный персептрон (рис. 2.4), нейроны которого имеют активационную функцию в виде единичного скачка.

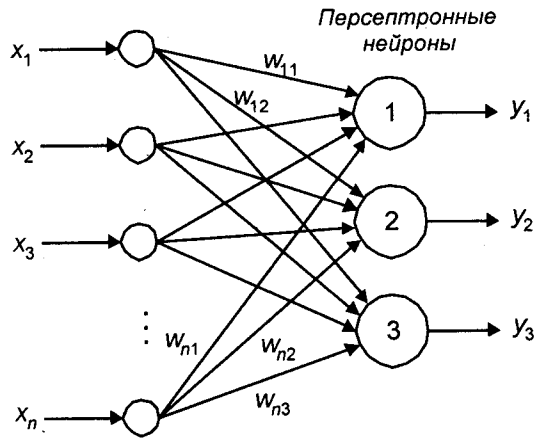


Рис. 2.4. Персептрон со многими выходами

На n входов подаются входные сигналы, поступающие далее по синапсам на три нейрона, которые образуют единственный слой этой сети. На выходах сети формируются сигналы:

$$y_j = f\left(\sum_{i=1}^{n} x_i\, w_{ij}\right),\ j = 1\ldots3. \qquad (2.1)$$

Весовые коэффициенты синапсов одного слоя нейронов можно свести в матрицу $\boldsymbol{W}$, в которой каждый элемент w_{ij} задает величину i-ой синаптической связи j-го нейрона. Таким образом, процесс, происходящий в нейронной сети, может быть записан в матричной форме:

$$\boldsymbol{Y} = F(\boldsymbol{XW}), \qquad (2.2)$$

где $\boldsymbol{X}$ и $\boldsymbol{Y}$ – соответственно входной и выходной векторы (под вектором понимается вектор-строка), $F(\boldsymbol{S})$ – активационная функция, применяемая поэлементно к компонентам вектора $\boldsymbol{S}$.

На рис. 2.5 представлен двухслойный персептрон, образованный из однослойного добавлением второго слоя, состоящего из двух нейронов.

Отметим важную роль нелинейности активационной функции, так как, если бы она не обладала данным свойством, результат функционирования любой Q-слойной нейронной сети с весовыми матрицами $\boldsymbol{W}^{(q)}$ для каждого слоя $q = 1\ldots Q$ свелся бы к перемножению входного вектора сигналов $\boldsymbol{X}$ на матрицу:

$$\boldsymbol{W}^{(\Sigma)} = \boldsymbol{W}^{(1)} \ldots \boldsymbol{W}^{(q)} \ldots \boldsymbol{W}^{(Q)}. \qquad (2.3)$$

Фактически такая Q-слойная нейронная сеть эквивалентна сети с одним скрытым слоем и с весовой матрицей единственного слоя $\boldsymbol{W}^{(\Sigma)}$:

$$\boldsymbol{Y} = \boldsymbol{XW}^{(\Sigma)}. \qquad (2.4)$$

Работа персептрона сводится к классификации (обобщению) входных сигналов, принадлежащих n-мерному гиперпространству, по некоторому числу классов. С математической точки зрения это происходит путем разбиения гиперпространства гиперплоскостями. Для однослойного персептрона:

$$\sum_{i=1}^{n} x_i\, w_{ir} = \theta_r,\ r = 1\ldots m.$$

Каждая полученная область является областью определения отдельного класса. Число таких классов для персептрона не превышает 2^m, где m – число его выходов. Однако не все из классов могут быть разделимы данной нейронной сетью.

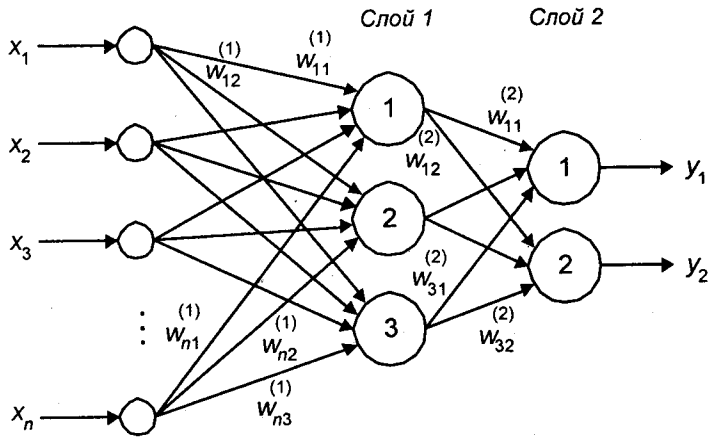


Рис. 2.5. Двухслойный персептрон

Например, однослойный персептрон, состоящий из одного нейрона с двумя входами, не может реализовать логическую функцию «Исключающее ИЛИ», т. е. не способен разделить плоскость (двумерное гиперпространство) на две полуплоскости так, чтобы осуществить классификацию входных сигналов по классам *A* и *B* (см. табл. 2.1).

Таблица 2.1

Логическая функция «Исключающее ИЛИ»

x_1 \ x_2	0	1
0	*B*	*A*
1	*A*	*B*

Уравнение сети для этого случая:

$$x_1 w_1 + x_2 w_2 = \theta \qquad (2.5)$$

является уравнением прямой (одномерной гиперплоскости), которая ни при каких условиях не может разделить плоскость так, чтобы точки из множества входных сигналов, принадлежащие разным классам, оказались по разные стороны от прямой (рис. 2.6). Не-

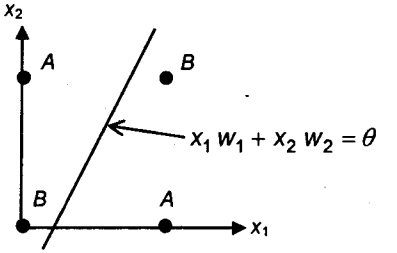


Рис. 2.6. Линейная неразделимость функции *Исключающее ИЛИ*

возможность реализации однослойным персептроном этой функции получила название *проблемы «Исключающего ИЛИ»*.

Отметим, что функции, которые не реализуются однослойным персептроном, называются линейно неразделимыми. Решение задач, подпадающих под это ограничение, заключается в применении 2-х и более слойных сетей или сетей с нелинейными синапсами, однако и тогда существует вероятность, что корректное разделение некоторых входных сигналов на классы невозможно.

Рассмотрим более подробно алгоритм обучения с учителем персептрона на примере, представленном на рис. 2.4.

ШАГ 1. Проинициализировать элементы весовой матрицы небольшими случайными значениями.

ШАГ 2. Подать на входы один из входных векторов, которые сеть должна научиться различать, и вычислить ее выход.

ШАГ 3. Если выход правильный, перейти на шаг 4. Иначе – вычислить разницу между требуемым и полученным значениями выхода:

$$\delta = d - Y .$$

Модифицировать веса в соответствии с формулой:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\ \delta\ x_i ,$$

где t и $(t+1)$ – номера текущей и следующей итераций; η – коэффициент скорости обучения, $0 < \eta < 1$; i – номер входа; j – номер нейрона в слое.

Очевидно, что если $d > Y$, то весовые коэффициенты будут увеличены и, тем самым, уменьшат ошибку. В противном случае они будут уменьшены, и Y тоже уменьшится, приближаясь к d.

Шаг 4. Цикл с шага 2, пока сеть не перестанет ошибаться.

На шаге 2 на разных итерациях поочередно в случайном порядке предъявляются все возможные входные вектора. К сожале-

нию, нельзя заранее определить число итераций, которые потребуется выполнить, а в некоторых случаях и гарантировать полный успех. Этот вопрос будет затронут в дальнейшем.

Сходимость рассмотренной процедуры устанавливается следующими теоремами.

Теорема 2.1.

Класс элементарных персептронов, для которых существует решение для любой задуманной классификации, не является пустым.

Эта теорема утверждает, что для любой классификации обучающей выборки можно подобрать такой набор (из бесконечного набора) элементарных нейронов, в котором будет осуществлено разделение обучающей последовательности при помощи линейного решающего правила.

Теорема 2.2.

Если для некоторой классификации решение существует, то в процессе обучения персептрона с коррекцией ошибок, начинающегося с произвольного исходного состояния, это решение будет достигнуто в течение конечного промежутка времени.

Смысл теоремы состоит в том, что если относительно задуманной классификации можно найти набор элементов, в котором существует решение, то в рамках этого набора оно будет достигнуто за конечный промежуток времени.

Интересную область исследований представляют многослойные персептроны и персептроны с перекрестными связями, однако теория этих систем практически не разработана.

2.3. Нейронные сети встречного распространения

Объединение разнотипных нейронных структур в единой архитектуре зачастую приводит к свойствам, которых нет у них по отдельности. Причем именно каскадные соединения нейронных структур, специализирующихся на решении различных задач, позволяют решить проблему комплексно.

Нейронные сети встречного распространения, состоящие из входного слоя нейронов и слоев нейронов Кохонена и Гроссберга, по своим характеристикам существенно превосходят возможности сетей с одним скрытым слоем нейронов. Так, время их обучения задачам распознавания и кластеризации более, чем в сто раз меньше времени обучения аналогичным задачам сетей с обратным распространением ошибки. Это может быть полезно в тех приложениях, где долгая обучающая процедура невозможна.

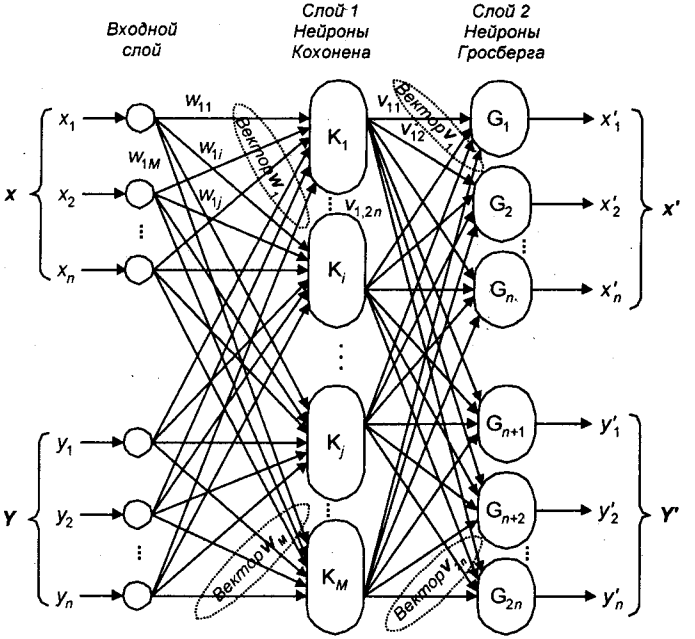


Рис. 2.7. Структура нейронной сети встречного распространения

Одними из определяющих характеристик сети встречного распространения являются ее хорошие способности к обобщению, позволяющие получать правильный выход даже при неполным или зашумленном входном векторе. Это существенно для эффективного использования данной сети для распознавания и восстановления образов, а также для усиления сигналов.

На рис. 2.7 показана структура сети встречного распространения. Нейроны входного слоя служат для передачи входных сигналов на все нейроны слоя Кохонена с соответствующими весовыми коэффициентами w_{ij}. Весовые коэффициенты входов нейронов j ($j = 1 \ldots M$) слоя Кохонена образуют соответствующие весовые векторы $\boldsymbol{w}_j = (w_{1j}, \ldots, w_{2n\,j})$ для $j = 1 \ldots M$.

Каждый нейрон слоя Кохонена соединен с каждым нейроном из слоя Гроссберга весами v_{jk}. Веса входов нейронов k ($k = 1 \ldots 2n$) слоя Гроссберга образуют соответствующие весовые векторы $\boldsymbol{v}_k = (v_{k1}, \ldots, v_{k,2n})$ для $k = 1 \ldots 2n$.

Нейроны слоя Кохонена реализуют функцию порогового суммирования взвешенных входов. Однако, в отличие от осталь-

ных слоев, нейрон слоя Кохонена с максимальным значением взвешенной суммы (на заданный входной вектор) является «победителем». На его выходе формируется уровень логической «1», на выходах остальных нейронов слоя – «0».

Нейроны же слоя Гроссберга на выходах выдают величины весов v_{ij}, которые связывают их с нейроном–«победителем» из слоя Кохонена.

Процесс обучения нейронной сети встречного распространения различен для слоев нейронов Кохонена и Гроссберга. Рассмотрим вопросы, возникающие при обучении каждого из этих слоев.

Перед обучением (самообучением) слоя Кохонена, протекающим без учителя, необходимо выполнить предварительную нормализацию входных $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k, \ldots, \boldsymbol{x}^N\}$ ($\{\boldsymbol{y}^1, \ldots, \boldsymbol{y}^k, \ldots, \boldsymbol{y}^N\}$) и весовых векторов $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M\}$.

Нормализация входных векторов осуществляется с целью их преобразования в единичные векторы с теми же направлениями перед предъявлением сети в соответствии со следующим выражением:

$$x^*_i = \frac{x_i}{\sqrt{x_1^2 + \ldots + x_i^2 + \ldots x_n^2}}.$$

Нормализация же начальных случайных значений весовых векторов приближает их к окончательным значениям, сокращая тем самым время обучения. Эти окончательные значения весовых векторов совпадают с нормализованными значениями входных векторов.

При обучении слоя Кохонена на вход подается нормализованный входной вектор. На выходе нейрона с максимальным значением взвешенной суммы формируется уровень логической «1».

При этом процесс обучения после выбора нейрона–«победителя» с весовым вектором, наиболее близким к входному вектору, состоит в дальнейшей подстройке (приближении) компонентов весового вектора выбранного нейрона к предъявленному входному вектору в соответствии с выражением:

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + \eta\,(\boldsymbol{x}^k - \boldsymbol{w}_i(t)),$$

где $\boldsymbol{w}_i(t+1)$, $\boldsymbol{w}_i(t)$ – соответственно новое и предыдущее значения вектора весов выигравшего i-го нейрона–«победителя» для предъявленного входного вектора $\boldsymbol{x}^k$ ($k = 1 \ldots N$); η – коэффициент скорости обучения.

Каждый коэффициент из весового вектора нейрона–«победителя» изменяется пропорционально разности между его величиной и величиной входа, к которому он присоединен. Знак изменения минимизирует разность между весовым коэффициентом и его входом. По мере обучения коэффициент η постепенно уменьшается.

В результате обучения нейрон–«победитель» будет активизироваться для совокупности ассоциированных с ним входных векторов, средняя величина которых совпадает с вектором весов этого нейрона.

Однако существует ряд проблем обучения слоя Кохонена, от решения которых зависит эффективность использования нейронной сети встречного распространения в целом.

Прежде всего, из-за того, что нормализованные входные векторы, как правило, неравномерно распределены по поверхности гиперсферы, большинство весовых векторов (изначально равномерно распределенных рандомизацией весов) будут значительно удалены от любого входного вектора. Поэтому на выходах соответствующих им нейронов постоянно будет установлен уровень логического «0», и эти нейроны окажутся бесполезными. С другой стороны, активных нейронов может оказаться недостаточно для эффективного разделения близкорасположенных входных векторов.

Еще одна проблема заключается в сложности разделения на различные классы множеств сходных входных векторов в случае, если изначальная плотность весовых векторов в окрестности обучающих векторов будет недостаточной.

Проблему может представлять также излишне высокая плотность весовых векторов вблизи несущественно различающихся входных векторов. Что может привести к активизации нескольких нейронов слоя Кохонена, т. е. к формированию ложных классов входных векторов.

Решением перечисленных проблем является распределение весовых векторов в соответствии со сложностью входных векторов. Существует несколько путей приближенного достижения этой цели, из которых наиболее популярны следующие четыре.

1) *Метод выпуклой комбинации* (*convex combination method*). Первоначально всем весовым коэффициентам присваиваются одинаковые значения $1/\sqrt{n}$. Каждый компонент входного вектора модифицируется в соответствии с правилом: $x'_i = \eta\, x_i + \frac{1-\eta}{\sqrt{n}}$. Сначала η мало, и длина всех входных векторов близка к векторам

весов. Затем в процессе обучения η постепенно увеличивается до единицы, что позволяет правильно разделить входные векторы.

Недостатком данного метода является увеличение времени обучения из-за необходимости подстройки весовых коэффициентов к постоянно изменяющимся значениям входных векторов.

2) *Зашумление входных векторов*. Входные векторы подвергаются случайным изменениям, благодаря чему захватывается ближайший весовой вектор. Этот метод более медленный, чем метод выпуклой комбинации.

3) *Изменение числа корректируемых нейронов*. На начальной стадии процесса обучения подстраиваются веса всех нейронов слоя, а не только нейрона–«победителя». Далее подстройка весов производится лишь для ближайших к выигравшему нейронах. Число этих нейронов по мере обучения сокращается. И в конце остается лишь один нейрон.

4) *Наделение нейронов «чувством справедливости»*. Если нейрон становится «победителем», то порог его срабатывания временно увеличивается. Это дает возможность обучаться и другим нейронам.

Назначением нейронов слоя Гроссберга является формирование требуемых выходных векторов после того, как нейроны слоя Кохонена разделили входные векторы на классы. Фактически каждый нейрон слоя Гроссберга лишь выдает значение веса, который связывает этот нейрон с нейроном–«победителем» слоя Кохонена.

В отличие от самообучающегося слоя Кохонена, слой Гроссберга обучается с учителем. Различие же со стандартной обучающей процедурой заключается в том, что подстройке подвергаются только те веса нейронов слоя Гроссберга, которые соединены с ненулевым нейроном Кохонена. Используется следующее правило:

$$v_{ij}(t+1) = v_{ij}(t) + \eta\,(y_j - v_{ij}(t))K_i,$$

где K_i – выход i-го нейрона Кохонена; y_j – j-й компонент требуемого выходного вектора. Первоначально η равен 0,1 и уменьшается в процессе обучения.

Отличие нейронной сети встречного распространения от других заключается также в особенностях функционирования. В соответствии с приведенной на рис. 2.7 структурой на вход сети подаются нормализованные единичные векторы $\boldsymbol{X}$ и $\boldsymbol{Y}$, а на выходе формируются их нормализованные аппроксимации $\boldsymbol{X'}$ и $\boldsymbol{Y'}$.

При обучении векторы $\boldsymbol{X}$ и $\boldsymbol{Y}$ подаются одновременно и как входные, и как требуемые выходные. При этом вектор $\boldsymbol{X}$ является входным для вектора $\boldsymbol{X'}$, а вектор $\boldsymbol{Y}$ – для вектора $\boldsymbol{Y'}$. В результате

такого обучения получается однозначное отображение векторов $\boldsymbol{X}$ и $\boldsymbol{Y}$ на их копии.

После обучения в нейронной сети встречного распространения реализуется свойство ассоциативной памяти, заключающееся в том, что предъявление на вход только вектора $\boldsymbol{X}$ (или $\boldsymbol{Y}$) при отсутствии другого приводит к порождению на выходе как вектора $\boldsymbol{X'}$, так и $\boldsymbol{Y'}$.

2.4. Оптимизирующие нейронные сети

2.4.1. Нейронные сети Хопфилда

Нейронная сеть Хопфилда реализует существенное свойство автоассоциативной памяти – восстановление по искаженному (зашумленному) образу ближайшего к нему эталонного. В этом случае входной вектор используется как начальное состояние сети, и далее сеть эволюционирует согласно своей динамике. Причем любой пример, находящийся в области притяжения хранимого образца, может быть использован как указатель для его восстановления. Выходной (восстановленный) образец устанавливается, когда сеть достигает равновесия.

Структура сети Хопфилда приведена на рис. 2.8. Она состоит из одного слоя нейронов, число которых определяет число входов и выходов сети. Выход каждого нейрона соединен с входами всех остальных нейронов. Подача входных векторов осуществляется через отдельные входы нейронов.

Сети Хопфилда отличаются от ранее рассмотренных типов нейронных сетей следующими существенными признаками:

- наличие обратных связей, идущих с выходов сетей на их входы по принципу «со всех на все»;
- расчет весовых коэффициентов нейронов проводится на основе исходной информации лишь перед началом функционирования сети, и все обучение сети сводится именно к этому расчету без обучающих итераций;
- при предъявлении входного вектора, сеть «сходится» к одному из запомненных в сети эталонов, представляющих множество равновесных точек, которые являются локальными минимумами функции энергии, содержащей в себе всю структуру взаимосвязей в сети.

Проблема устойчивости сети Хопфилда была решена после того, как Кохеном и Гроссбергом была доказана теорема, определяющая достаточное условие устойчивости сетей с обратными связями, а именно, сеть с обратными связями является устойчи-

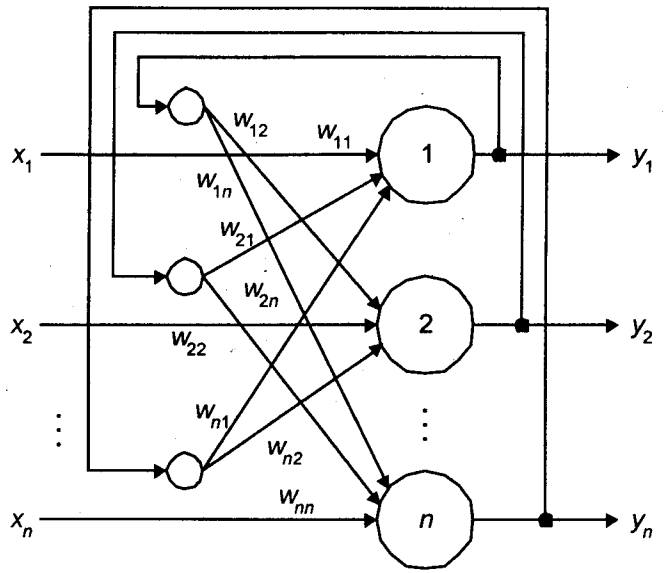


Рис. 2.8. Структура нейронной сети Хопфилда

вой, если матрица ее весов симметрична ($w_{ij} = w_{ji}$) и имеет нули на главной диагонали ($w_{ii} = 0$).

Динамическое изменение состояний сети может быть выполнено, по крайней мере, двумя способами: синхронно и асинхронно. В первом случае все элементы модифицируются одновременно на каждом временном шаге, во втором – в каждый момент времени выбирается и подвергается обработке один элемент. Этот элемент может выбираться случайно.

Рассмотрим синхронную бинарную сеть Хопфилда, представляющую собой пример сети *с дискретными состояниями и дискретным временем*, и сформулируем решаемую ею задачу следующим образом.

В качестве нейронов сети рассмотрим нейроны с пороговой функцией активации, выходы которых принимают значение либо «0», либо «1» при превышении взвешенной суммой значений входов некоторого порогового уровня.

Предварительно в сети матрицей весовых коэффициентов задан набор эталонов. Каждый эталон при этом является точкой из конечного множества равновесных точек, характеризующих минимум энергии сети (функции Ляпунова):

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i y_j - \sum_{j=1}^{n} x_j y_j + \sum_{j=1}^{n} \theta_j y_j ,$$

где E – искусственная энергия сети; w_{ij} – вес от выхода i-го ко входу j-го нейрона; x_j, y_j – вход и выход j-го нейрона; θ_j – порог j-го нейрона.

Главное свойство энергетической функции состоит в том, что в процессе эволюции состояний нейронной сети согласно уравнению она уменьшается и достигает локального минимума (аттрактора), в котором сохраняет постоянную энергию. Это позволяет решать задачи комбинаторной оптимизации, если они могут быть сформулированы как задачи минимизации энергии.

Обозначим вектор, описывающий k-й эталон, через $\boldsymbol{X}^k = \{x_i^k\}$, $k = 1 \ldots K$, K – число эталонов.

На вход сети подается произвольный вектор $\boldsymbol{X} = \{x_i\}$.

В результате серии итераций сеть должна выделить эталон, соответствующий входному вектору, или дать заключение о том, что входные данные не соответствуют ни одному из эталонов.

После отдельной итерации общее изменение энергии сети, вызванное изменением состояний всех нейронов, составит:

$$\Delta E = -\sum_{j=1}^{n}\left[\sum_{i \neq j}(w_{ij} y_i) + x_j - \theta_j\right]\Delta y_j ,$$

где Δy_j – изменение выхода j-го нейрона после итерации.

Анализ этого выражения показывает, что любое изменение состояния нейронов либо уменьшит значение E, либо оставит его без изменения. Второй случай указывает на достижение сетью устойчивого состояния и выделение ею эталона, наилучшим образом сочетающимся с входным вектором.

При распознании входного вектора (частично представленного или искаженного) выходы сети будут содержать соответствующий эталон, т. е. $\boldsymbol{Y} = \boldsymbol{X}^k$, где $\boldsymbol{Y} = \{y_i\}$ – выходной вектор. В противном случае, выходной вектор не совпадет ни с одним эталоном.

Для безошибочной работы сети Хопфилда число запоминаемых эталонов N не должно превышать $0{,}15n$.

Кроме того, в случае высокой степени корреляции нескольких эталонов возможно возникновение перекрестных ассоциаций при их предъявлении на входах сети. Требование достаточного (но не необходимого) условия слабой коррелируемости образов можно представить как выполнение следующего неравенства:

$$\sum_{k \neq j}^{N} \left|\left(x^k, x^j\right)\right| < n , \quad j = 1 \ldots n,$$

или в виде более сильного условия:

$$\left|\left(x^k, x^j\right)\right| < \frac{n}{N}, \quad k = 1 \ldots n, \quad k \neq j.$$

Нейронные сети Хопфилда *с непрерывными состояниями* отличается от вышерассмотренной сети непрерывной активационной функцией нейронов, в качестве которой наиболее часто выбирают сигмоидальную или логистическую функцию. Концептуально, вопросы организации и функционирования этих сетей схожи.

Недостатком классического варианта сетей Хопфилда является их тенденция к стабилизации в локальных, а не глобальных минимумах сети. Предложены *статистические сети Хопфилда*, в которых этот недостаток преодолевается за счет задания статистических, а не детерминистских правил изменения состояний нейронов.

Для этого для каждого j-го нейрона вводится вероятность изменения его состояния ρ_j как функция от величины, на которую выход нейрона s_j превышает его порог θ_j (для бинарной сети Хопфилда):

$$\rho_j = \frac{1}{1 + \exp\left(\frac{-\Delta E_j}{\phi}\right)},$$

где $\Delta E_j = s_j - \theta_j$, ϕ – параметр, изменяемый в процессе стабилизации сети.

Тогда после начального задания весовых коэффициентов нейронов процедура поиска минимума энергии сети для установленного входного вектора выглядит следующим образом.

ШАГ 1. Задать большое значение параметра ϕ.

ШАГ 2. Установить на выходе j-го нейрона значение «1» с вероятностью ρ_j и значение «0» с вероятностью $1-\rho_j$.

ШАГ 3. Постепенно уменьшать значение параметра ϕ, пока не будет достигнуто равновесие.

2.4.2. Нейронные сети Хэмминга

В случае, если необходимо определить номер эталона, ближайший к предъявленному входному вектору, может быть использована сеть Хэмминга. Преимуществами этой сети по сравнению с сетью Хопфилда являются меньшие затраты на память и объем вычислений.

Нейронная сеть Хэмминга (рис. 2.9) состоит из входного, скрытого и выходного слоев нейронов. Скрытый и выходной слои содержат по K нейронов, где K – число эталонов. Нейроны скрытого слоя n синапсами соединены с выходами нейронов входного слоя сети. Выходы нейронов выходного слоя связаны со входами остальных нейронов этого слоя отрицательными обратными (ингибиторными) связями. Единственная положительная обратная связь подается с выхода для каждого нейрона выходного слоя на его же вход.

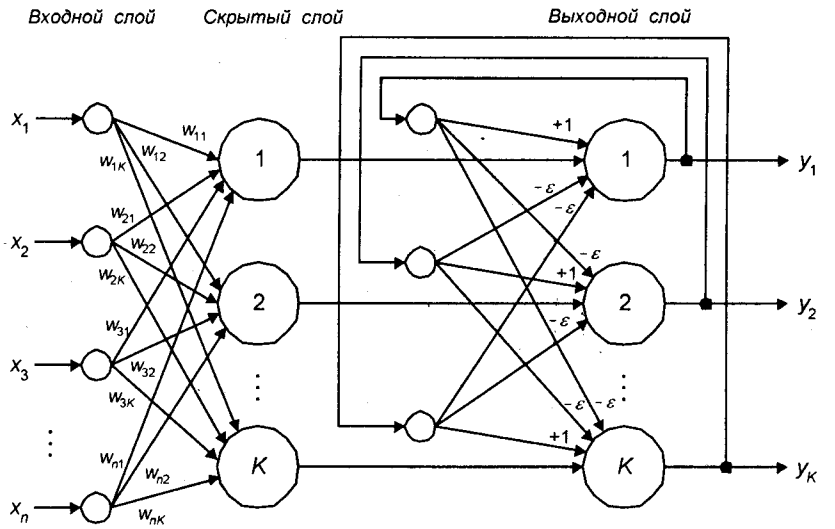


Рис. 2.9. Структура нейронной сети Хэмминга

Сеть выбирает эталон с минимальным хэммингловым расстоянием от предъявленного входного вектора путем активизации только одного выхода сети (нейрона выходного слоя), соответствующего этому эталону.

Хэммингово расстояние представляет собой пример меры сходства или, вернее, различия, первоначально введенной для бинарных функций в диадном пространстве. Она применима для сравнения любых упорядоченных наборов, принимающих дискретные значения и, вероятно, является наилучшей из известных мер сходства между цифровыми кодами. Для бинарных последовательностей $x = (x_1, \ldots, x_n)$ и $x' = (x'_1, \ldots, x'_n)$ хэммингово расстояние можно определить:

$$\rho(x, x') = bc\{(\overline{x_i} \wedge x'_i) \vee (x_i \wedge \overline{x'_i}) \mid i = 1 \ldots n\}.$$

Здесь функция $bc\{\bullet\}$ определяется как число элементов набора $\{\bullet\}$, принимающих значение логической «1».

На этапе настройки сети Хэмминга устанавливаются следующие значения весов нейронов скрытого слоя и порога их активационной функции:

$$w_{ik} = \frac{x_i^k}{2}, \quad \theta_k = n/2,$$

где x_i^k – i-й компонент k-го эталона; $i = 1...n$, $k = 1...K$.

Коэффициенты отрицательных обратных связей нейронов выходного слоя задают равными некоторой величине из интервала $0 < \varepsilon < 1/K$, а коэффициенты положительной обратной связи – +1 .

Рассмотрим алгоритм функционирования сети Хэмминга.

ШАГ 1. На нейроны входного слоя подается вектор $X = \{x_i\}$, $i = 1...n$. На их выходах формируются следующие значения (верхний индекс указывает номер слоя):

$$y_k^{(1)} = s_k^{(1)} = \sum_{i=1}^{n} w_{ik} x_i + \theta_j , \; k = 1...K.$$

В соответствии с этим устанавливаются значения на выходах нейронов выходного слоя:

$$y_k^{(2)} = y_k^{(1)}, \; k = 1...K.$$

ШАГ 2. В результате новой (t+1)-й итерации определяются новые состояния нейронов выходного слоя:

$$s_k^{(2)}(t+1) = y_k^{(2)}(t) - \varepsilon \sum_{\substack{j=1 \\ j \neq k}}^{n} y_j^{(2)}(t), \; k = 1... K ,$$

$$y_k^{(2)}(t+1) = f\left[s_k^{(2)}(t+1)\right] \; k = 1... K .$$

Активационная функция f имеет вид порога (рис. 1.3, *б*), причем величина «ступеньки» должна быть достаточно большой, чтобы возможные значения $s_k^{(2)}$ не приводили к насыщению.

ШАГ 3. Проверка изменения состояний нейронов выходного слоя за последнюю итерацию. И переход к шагу 2 в случае, если наблюдались изменения. Иначе – окончание процедуры.

Роль нейронов входного слоя весьма условна: воспользовавшись один раз на шаге 1 значениями его весовых коэффициентов, сеть больше не обращается к нему, поэтому этот слой может быть вообще исключен из сети (заменен на матрицу весовых коэффициентов).

В заключении можно сделать следующее обобщение. Сети Хопфилда и Хэмминга позволяют просто и эффективно решить

задачу автоассоциативной памяти: воссоздания образов по неполной и искаженной информации. Невысокая емкость сетей (число запоминаемых образов) объясняется тем, что сети не просто запоминают образы, а позволяют проводить их обобщение, например, с помощью сети Хэмминга возможна классификация по критерию максимального правдоподобия. Вместе с тем, легкость построения программных и аппаратных моделей делают эти сети привлекательными для многих практических применений.

В случае, если необходимо определить эталон, ближайший к предъявленному входному вектору (например, на основе хэммингова расстояния), часто возникают проблемы, связанные с различием длин или с ограничениями на длину последовательностей или количество компонентов в наборах.

Отметим один важный аспект проблемы сравнения векторов различной длины. Так, последовательность большей длины может представлять собой функцию в пространстве с числом измерений большим, чем число измерений другой функции на разницу числа параметров в длинах последовательностей. Это определяет возможность перехода к оценке не только количественного, но и качественного сходства векторов, к оперированию в терминах вероятности, нечетких оценок, отнесению вектора к какому-либо классу.

Эти различия и ограничения сравнения векторов различной длины можно устранить различными способами, например, с помощью *меры сходства Танимото*.

Авторами предложены модели оптимизирующих нейронных сетей, реализующие восстановление по искаженному (зашумленному) образу ближайший к нему эталонный, позволяющие снять ограничения на различия и ограничения длин сравниваемых векторов с сохранением высокой достоверности распознавания.

2.5. Двунаправленная ассоциативная память

Как и сети Хопфилда и Хэмминга, двунаправленная ассоциативная память (ДАП) или, иначе, нейронная сеть Коско (B. Kosko), способна к обобщению, вырабатывая правильные реакции, несмотря на искаженные входы. Однако в отличие от этих типов нейронных сетей, ДАП решает задачи гетероассоциативной памяти, т. е. входной образ может быть ассоциирован с другим, не коррелирующим с ним образом. Реализуется это вследствие того, что выходной вектор формируется на другом наборе нейронов, нежели соответствующий ему входной.

На рис. 2.10 рассмотрена базовая структура двунаправленной ассоциативной памяти, состоящая из входного, скрытого и вы-

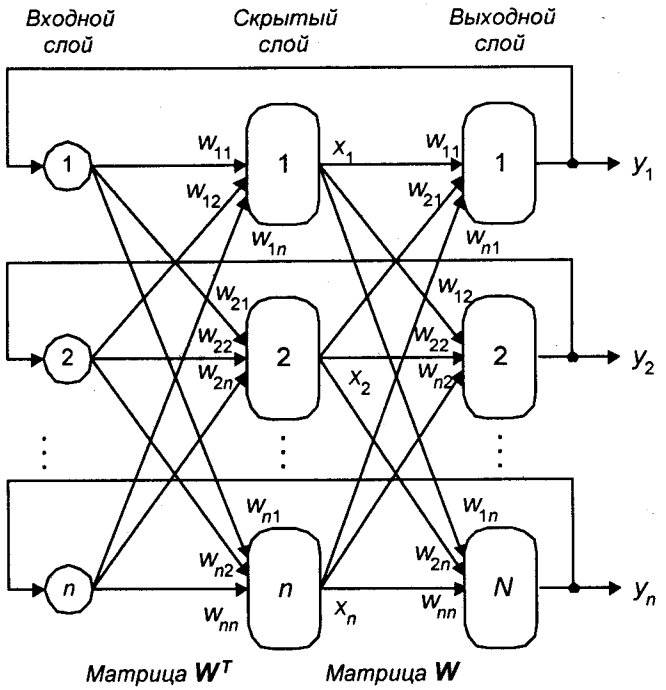


Рис. 2.10. Структура двунаправленной ассоциативной памяти

ходного слоев нейронов. Нейроны скрытого и выходного слоев выполняют функцию взвешенного суммирования входов с сигмоидальной (логистической) активационной функцией F. В упрощенном виде в качестве активационной выбирают пороговую функцию.

До начала функционирования нейронная сеть обучается с использованием набора пар векторов $\boldsymbol{X}$ и $\boldsymbol{Y}$ путем вычисления значений весовых коэффициентов $\boldsymbol{W}$ и $\boldsymbol{W}^T$, реализующих отношения ассоциаций. При этом весовая матрица $\boldsymbol{W}$ вычисляется как сумма произведений всех пар векторов N из обучающей выборки:

$$\boldsymbol{W} = \sum_{i=1}^{N} \boldsymbol{X}_i^T \boldsymbol{Y}_i, \quad i = 1 \ldots N.$$

При решении задачи восстановления запомненных ассоциаций вектор $\boldsymbol{X}$ или его часть кратковременно устанавливается на выходах нейронов скрытого слоя. Вектор $\boldsymbol{X}$ обрабатывается матрицей весовых коэффициентов $\boldsymbol{W}$ нейронов выходного слоя. Затем вектор $\boldsymbol{X}$ удаляется, и сеть вырабатывает ассоциированный вектор

$\boldsymbol{Y}$ на выходе нейронов выходного слоя, поступающий на входы нейронов входного слоя и обрабатывающийся транспонированной матрицей $\boldsymbol{W}^T$ весов нейронов скрытого слоя.

$$\boldsymbol{Y} = F(\boldsymbol{XW}),$$
$$\boldsymbol{X} = F(\boldsymbol{YW}^T).$$

При этом в каждом цикле происходит уточнение выходного вектора. Процесс повторяется до достижения устойчивого состояния сети, при котором вектор $\boldsymbol{X}$ и $\boldsymbol{Y}$ не изменяются.

Двунаправленная ассоциативная память функционирует в направлении минимизации энергии сети (функции Ляпунова) в соответствии со значениями весов. Устойчивость двунаправленной ассоциативной памяти гарантируется транспонированием матрицы весовых коэффициентов.

Проводя аналогию с биологическими системами, можно отметить, что значения весовых коэффициентов $\boldsymbol{W}$ и $\boldsymbol{W}^T$ образуют долговременную память, а состояние нейронов – кратковременную память. Весовые коэффициенты могут изменяться на более длительном интервале времени на основе представленных далее методов с целью вывода сети из локального и достижения глобального оптимума для первоначально установленного вектора $\boldsymbol{X}$.

Двунаправленная ассоциативная память сводится к сети Хопфилда, если матрица весовых коэффициентов $\boldsymbol{W}$ является квадратной и симметричной, т. е. $\boldsymbol{W} = \boldsymbol{W}^T$.

Двунаправленная ассоциативная память имеет ограничения на максимальное количество хранимых ассоциаций, при превышении которого сеть может выработать неверный выходной сигнал, воспроизводя ассоциации, которым не обучена.

Для безошибочной работы бинарной ДАП число запоминаемых векторов N не должно превышать $\dfrac{n}{2\log_2 n}$.

Известны оценки, в соответствии с которыми ДАП может иметь до 2^n стабильных состояний, если для каждого нейрона выбирается свое ненулевое пороговое значение θ_i. Такая сеть, названная *негомогенной двунаправленной ассоциативной памятью*, является расширением гомогенной ДАП, в которой все пороги нулевые. В этом случае выход нейрона принимает следующий вид:

$$y_i(t+1) = 1, \text{ если } s_i(t) > \theta_i,$$
$$y_i(t+1) = 1, \text{ если } s_i(t) < \theta_i,$$
$$y_i(t+1) = y_i(t), \text{ если } s_i(t) = \theta_i.$$

где $y_i(t)$ – выход нейрона i в момент времени t.

Однако выбор этих состояний определяется жесткой процедурой. Если выбрано N состояний случайным образом, причем $N < \frac{0{,}68\,n^2}{(\log_2 n + 4)^2}$, и если каждый вектор имеет $(4 + \log_2 n)$ компонентов, равных «1», (остальные компоненты равны «0»), то можно сформировать негомогенную ДАП, хранящую $0{,}98N$ этих векторов в качестве стабильных состояний.

Разработано много разновидностей двунаправленной ассоциативной памяти, основными из которых являются: непрерывная ДАП (с сигмоидами в качестве функций активации нейронов), адаптивная ДАП (с изменением весов в процессе функционирования сети), конкурирующая ДАП (с конкуренцией нейронов внутри каждого слоя).

Основными достоинствами двунаправленной ассоциативной памяти являются следующие:

- структурная простота сети, позволяющая реализовать ее в виде СБИС и УБИС;
- совместимость с аналоговыми схемами и оптическими системами;
- быстрая сходимость процесса обучения и восстановления информации.

2.6. Сети адаптивной резонансной теории

Феномен памяти человека исследован явно недостаточно. Среди классов памяти (генетической, врожденной и прижизненной), наибольший интерес для специалистов в области искусственных нейронных сетей представляет исследование прижизненной памяти, обеспечивающей возможность приспособления организма к окружающей среде путем накопления информации в процессе обучения, функционирования, самоорганизации. Один из самых интересных вопросов заключается в следующем: каким образом память сохраняет пластичность, т. е. способность к восприятию новых образов, оставаясь при этом устойчивой к разрушению ранее запомненных образов. И что является стимулом для запоминания новой информации.

Решение проблемы стабильности–пластичности для различных парадигм нейронных сетей решается по-разному. Так, в случае фиксированного набора обучающих векторов они могут предъявляться циклически, нейтрализуя необратимый характер модификаций весов в результате обучения новому образу. Для сетей с обратным распространением при существенном изменении весов

зачастую требуется полное переобучение. Кроме того, на практике сети может быть никогда не предъявлен один и тот же обучающий вектор дважды. Поэтому сеть будет непрерывно изменять веса, характеризуясь временной нестабильностью.

Для решения задач векторной классификации Карпентером и Гроссбергом предложен класс нейронных сетей, реализующих модели *адаптивной резонансной теории* (АРТ) (ART–1, ART–2 и ARTMAP), сохраняющих пластичность для запоминания новых образов и, в то же время, предотвращающих изменение ранее запомненных образов.

Для иллюстрации основных положений адаптивной резонансной теории рассмотрим далее сеть АРТ (ART–1) для классификации двоичных векторов, заметив, что такая нейронная сеть может классифицировать также и непрерывные векторы (ART–2).

На рис. 2.11 показана структура сети АРТ, состоящая из блока сравнения, блока распознавания, схемы определения сходства векторов, а также из двух вспомогательных схем ИЛИ 1 и ИЛИ 2.

Нейроны слоя сравнения блока сравнения осуществляют функцию мажоритарного срабатывания по правилу «2 из 3» (выход нейрона равен единице только в том случае, если, как минимум, два из трех его входов равны единице).

В блоке распознавания осуществляется классификация входных векторов. Слой нейронов распознавания этого блока состоит из нейронов с весовыми векторами $\boldsymbol{W}_k = \{w_{ik}\}$, взаимодействующих по латерально-тормозящей схеме: в каждый момент времени возбуждается только один нейрон с наибольшим уровнем активации. Это правило («победитель забирает все») реализуется за счет введения связей с отрицательными весами с выхода нейрона на входы остальных нейронов слоя. Для простоты на рисунке не показаны латерально-тормозящие связи. Кроме того, каждый нейрон имеет положительную обратную связь с выхода на собственный вход, усиливающую и поддерживающую единичный выходной уровень. Число нейронов этого слоя соответствует числу запомненных образов K (категорий классификации).

Схема определения сходства векторов определяет степень сходства между векторами $\boldsymbol{X}$ и $\boldsymbol{Y}$. В случае если степень отличия в соответствии с выбранным критерием сходства превышает некоторый заданный порог, этой схемой вырабатывается сигнал сброса возбужденного нейрона в слое распознавания.

Рассмотрим основные этапы функционирования сети АРТ.

Инициализация. До начала обучения случайным образом устанавливаются значения весовых векторов $\boldsymbol{W}_k$ нейронов слоя распознавания. Они должны удовлетворять следующему условию:

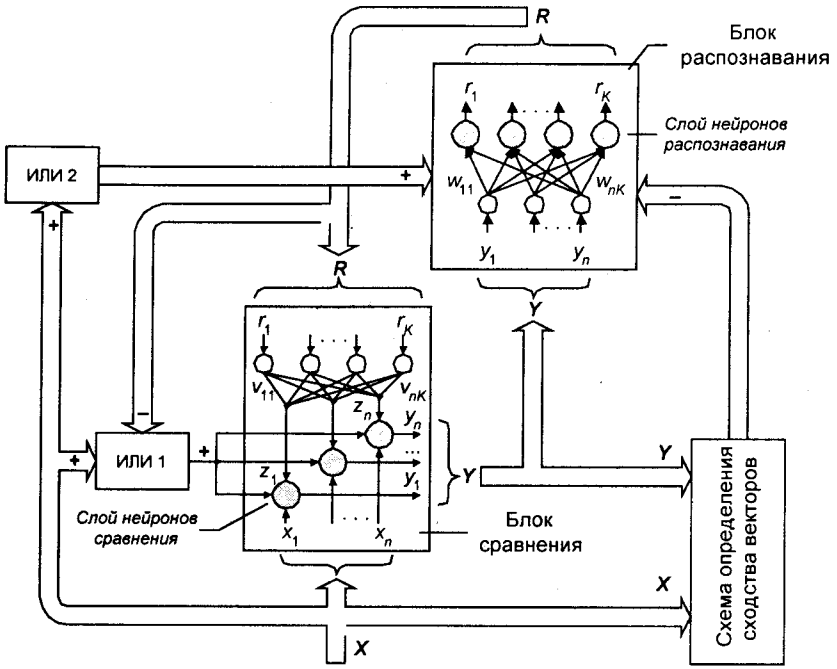


Рис. 2.11. Структура нейронной сети, реализующей адаптивную резонансную теорию

$$w_{ik} < \frac{H}{H-1+n}, \quad i = 1...n, \quad k = 1...K,$$

где n – число компонентов входного вектора (число нейронов слоя сравнения); P – число нейронов слоя распознавания; H – константа со значением в диапазоне от 1 до 2.

Такая установка этих весов гарантирует, что несвязанные нейроны не будут возбуждены более, нежели обученные нейроны в слое распознавания.

Значения всех компонентов весовых векторов $\boldsymbol{V_k}$ устанавливаются равными единице.

Значение параметра сходства ρ устанавливается в диапазоне от 0 до 1 в зависимости от заданной степени сходства входного и запомненных образов.

Обучение. Процесс обучения АРТ-сетей является обучением без учителя. Различают два вида обучения АРТ-сетей: медленное и быстрое. При медленном обучении входной вектор предъявляется на вход сети кратковременно, и весовые коэффициенты не достигают своих асимптотических значений в результате одного предъявления. В этом случае значения весов определяются статистическими характеристиками всех входных векторов, а не характеристиками отдельного входного вектора. Динамика процесса медленного обучения описывается дифференциальными уравнениями.

В случае рассматриваемого ниже быстрого обучения входной вектор устанавливается на входе сети на достаточно длительный интервал времени, что позволяет весовым коэффициентам w_{ik} приблизиться к своим окончательным значениям. Процесс быстрого обучения описывается алгебраическими уравнениями. Другим отличием быстрого обучения от медленного является то, что компоненты весовых коэффициентов $\boldsymbol{V_k}$ принимают бинарные значения, в отличие от непрерывного диапазона значений.

В ходе обучения входные векторы последовательно подаются на входы сети, и весовые векторы $\boldsymbol{W_k}$ изменяются таким образом, чтобы сходные векторы активизировали соответствующие нейроны слоя распознавания. Веса вычисляются по следующему правилу:

$$w_{ik} = \frac{H y_i}{H-1+\sum_{j=1}^{n} y_j},$$

где y_i – i-й компонент вектора $\boldsymbol{Y}$; k – номер возбудившегося нейрона в слое распознавания.

Компоненты вектора весов $\boldsymbol{V_k}$, связанные с новым запоминаемым образом, изменяются таким образом, что $v_{ik} = y_i$ для всех i.

Распознавание. В исходном положении на выходе схемы ИЛИ 2 установлен уровень нуля, обнуляющий выходы всех нейронов распознающего слоя.

Затем на вход сети подается ненулевой вектор $\boldsymbol{X}$, устанавливающий уровень единицы на выходе схемы ИЛИ 2. Таким образом, обеспечивается прохождение входного вектора $\boldsymbol{X}$ на выходы нейронов слоя сравнения без изменений, т. е. $\boldsymbol{Y} = \boldsymbol{X}$.

Далее для каждого k-го нейрона слоя распознавания вычисляется свертка его весового вектора $\boldsymbol{W_k}$ с вектором $\boldsymbol{Y}$. Выход нейрона с максимальным значением свертки, т. е. наиболее «близкого» к входному вектору, переходит в активное (единичное) состоя-

ние, т. е. «резонирует», тормозя остальные нейроны этого слоя, которые установятся в нуль.

Сравнение. Единица с выхода k-го возбужденного нейрона распознающего слоя подается на каждый i-й нейрон в слое сравнения со своим весом, устанавливая на входах z_i нейронов слоя сравнения уровень либо нуля, либо единицы.

Обратная связь от ненулевого вектора $\boldsymbol{R}$ устанавливает выход схемы ИЛИ 1 в нуль. И теперь в слое сравнения могут возбудиться лишь те нейроны, на входах которых соответствующие компоненты x_i и z_i одновременно равны единице.

Другими словами, целью обратной связи от нейронов слоя распознавания является установка компонентов выходного вектора $\boldsymbol{Y}$ в нуль в случае, если входной вектор $\boldsymbol{X}$ не соответствует хранимому образу, т. е. если векторы $\boldsymbol{X}$ и $\boldsymbol{Z}$ не имеют совпадающих компонентов.

Процедура определения сходства, осуществляемая схемой определения сходства векторов, заключается в определении отношения (S) числа единиц в векторах $\boldsymbol{Y}$ и $\boldsymbol{X}$, где вектор $\boldsymbol{Y}$ на этапе распознавания представляет собой логическое произведение входного вектора $\boldsymbol{X}$ и вектора $\boldsymbol{Z}$, который равен весовому вектору $\boldsymbol{W_k}$ выигравшего нейрона.

При существенном отличии векторов $\boldsymbol{X}$ и $\boldsymbol{Z}$ выходной вектор $\boldsymbol{Y}$ будет содержать много нулей в компонентах, где вектор $\boldsymbol{X}$ содержит единицы. Это означает, что установленный вектор $\boldsymbol{Z}$ не является искомым, и возбужденный нейрон в слое распознавания должен быть заторможен схемой определения сходства векторов. Процедура торможения заключается в установке в нуль выхода возбужденного нейрона в процессе текущей классификации.

Поиск. В случае совпадения или удовлетворения условий близости векторов $\boldsymbol{X}$ и $\boldsymbol{Y}$ процесс классификации завершается. Признаком этого является отсутствие на выходе схемы определения сходства векторов сигнала торможения (сброса) возбужденного в текущем цикле классификации нейрона слоя распознавания. В противном случае осуществляется поиск среди других запомненных образов для определения наиболее близкого к входному. При торможении возбужденного нейрона вектор $\boldsymbol{R}$ обнуляется, на выходе схемы ИЛИ 1 устанавливается уровень единицы, и вновь подготавливается прохождение входного вектора $\boldsymbol{X}$ на выходы нейронов слоя сравнения без изменений $\boldsymbol{Y} = \boldsymbol{X}$. В результате этой установки в слое распознавания возбуждается другой нейрон, и другой запомненный образ $\boldsymbol{Z}$ поступает на нейроны слоя сравнения. В случае несоответствия $\boldsymbol{Z}$ и $\boldsymbol{X}$, возбужденный нейрон в слое распознавания опять тормозится.

Процесс повторяется до тех пор, пока не будет найден запомненный образ, степень близости с которым у вектора $\boldsymbol{X}$ не меньше заданной. В этом случае осуществляется цикл дополнительного обучения с целью модификации весовых векторов k-го возбужденного нейрона слоя распознавания: $\boldsymbol{W_k}$ и $\boldsymbol{V_k}$.

Если определено, что ни один из запомненных векторов не соответствует входному, то вводится новый (K+1)-й нейрон в распознающем слое и его весовые векторы $\boldsymbol{W_k}$ и $\boldsymbol{V_k}$ устанавливаются в соответствии с новым входным вектором. Так как изначально все веса этого нейрона установлены в единичное значение, то выработанный в слое сравнения вектор $\boldsymbol{Y}$ будет идентичен входному вектору $\boldsymbol{X}$, и отношение S будет равно единице, удовлетворяя заданной степени сходства.

С целью ускорения процесса поиска вычисление сверток входного вектора с весовыми векторами $\boldsymbol{W_k}$ и определение сходства с запомненными образами может быть осуществлено параллельно.

Характеристики АРТ-сетей:

- быстрый доступ к предварительно запомненным образам, обусловленный тем, что после стабилизации процесса обучения предъявление одного из обучающих векторов (или вектора с существенными характеристиками категории) будет активизировать требуемый нейрон слоя распознавания без поиска;
- устойчивость процесса поиска, так как после возбуждения одного нейрона не будет возбуждений других нейронов в распознающем слое без сигнала с выхода схемы определения сходства векторов;
- конечность процесса обучения, обусловленная стабильным набором весов; повторяющиеся последовательности обучающих векторов не будут приводить к циклическому изменению весов.

В заключение отметим, что АРТ-сети организованы по принципу подобия с биологическими прототипами, а процессы, происходящие в них, подобно механизмам мозга, позволяют решить проблему стабильности–пластичности.

Недостатком АРТ-сетей является недостаточная надежность сохранения информации. Так, в случае «потери» одного образа разрушается вся память.

2.7. Когнитрон

Способы определения сходства и различия образов, правила распознавания, используемые компьютерными системами, за-

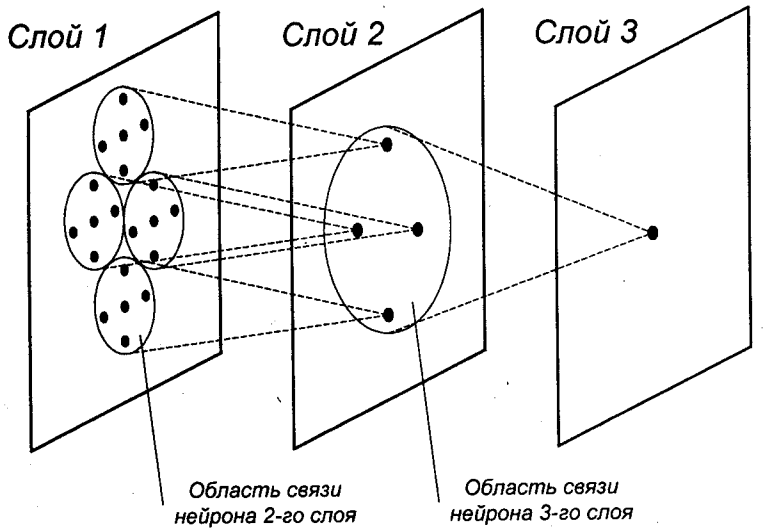


Рис. 2.12. Структура когнитрона

висят от особенностей их организации и функционирования, а также от выбора методов, адекватных решаемым задачам.

Переход при создании перспективных технических систем к биологическим адаптивным моделям восприятия и распознавания, которые характеризуются самоорганизацией в результате воздействия окружающей среды, обусловлен поразительной способностью человека к решению сложных задач.

К. Фукушима (K. Fukushima) в 1975 году разработал *когнитрон* – гипотетическую модель биологической системы восприятия и распознавания, инвариантную к поворотам, перемещениям, изменениям масштабов образов. Возможности когнитрона к адаптивному распознаванию образов стимулировали, в свою очередь, исследования механизмов мозга.

Когнитрон (рис. 2.12) организован подобно зрительной коре мозга человека, состоящей нескольких слоев нейронов. Несмотря на то, что слои организованы однотипно, каждый из них, подобно отдельным слоям зрительной коры, реализует различные уровни обобщения. Например, если входной слой нейронов распознает лишь простые образы (линии) и их ориентацию, то последующие слои способны к все более сложному обобщению, качество которого не зависит от положений распознаваемых образов.

Нейрон из последующего слоя связан с ограниченным набором нейронов предыдущего. Подобное правило организации при переходе от слоя к слою позволяет каждому нейрону выходного слоя реагировать на все входное поле при наличии ограниченного количества слоев нейронов. Однако при незначительном фиксированном размере области связи нейронов требуется большое число промежуточных слоев для перекрытия входного поля выходными нейронами. Количество требуемых слоев может быть уменьшено за счет расширения области связи в последующих слоях. Результатом такого расширения может явиться значительное перекрытие областей связи, приводящее к одинаковой реакции нейронов выходного слоя. Для решения этой проблемы может быть увеличен размер области конкуренции, из которой на поданный входной образ активизируется только один нейрон выходного слоя, а влияние малой разницы в реакциях нейронов усиливается.

В качестве альтернативного варианта организации межслойных взаимосвязей в когнитроне с целью предоставления каждому нейрону выходного слоя возможности реагировать на полное входное поле при наличии ограниченного количества слоев, области связи нейронов могут быть сформированы с учетом вероятностного распределения синаптических связей нейронов.

Каждый слой когнитрона (рис. 2.13) содержит два типа нейронов: возбуждающие и тормозящие. Возбуждающие нейроны одного слоя стремятся вызвать активизацию соединенного с ними нейрона следующего слоя. Тормозящие нейроны нейтрализуют это возбуждение. Возбуждение нейрона определяется значением нелинейной функции активации от взвешенной суммы его возбуждающих и тормозящих входов.

Каждый возбуждаемый нейрон последующего слоя связан с ограниченным числом возбуждающих нейронов предыдущего слоя (из области его связи), что согласуется с анатомическими представлениями о зрительной коре головного мозга. Аналогично, в предыдущем слое существуют тормозящие нейроны, соответствующие тем же областям связи.

Возбуждающие нейроны. Значение выхода возбуждающего нейрона определяется отношением взвешенных сумм возбуждающих и тормозящих входов:

$$y^e = F\left(\frac{1+\sum_i a_i u_i}{1+\sum_j b_j v_j} - 1\right) = F\left(\frac{\sum_i a_i u_i - \sum_j b_j v_j}{1+\sum_j b_j v_j}\right),$$

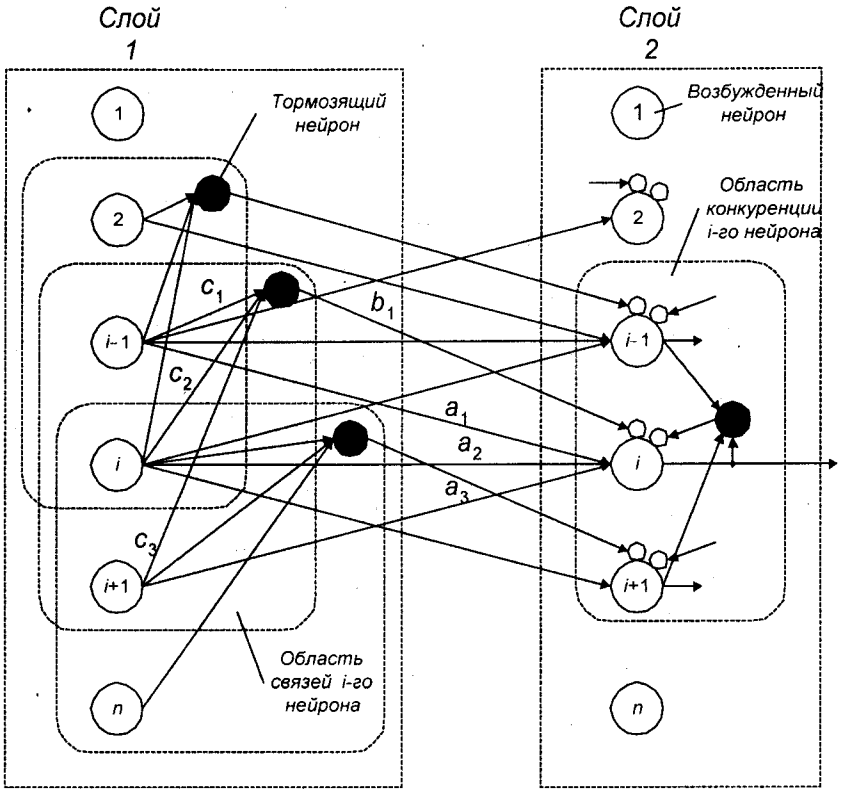


Рис. 2.13. Взаимосвязь слоев когнитрона

$$F(\Psi) = \begin{cases} \Psi, & \text{при} \quad \Psi \geq 0, \\ 0, & \text{при} \quad \Psi < 0, \end{cases}$$

где a_i – вес i-го возбуждающего входа; u_i – выход i-го возбуждающего нейрона предыдущего слоя; b_j – вес j-го тормозящего входа; v_j – выход j-го тормозящего нейрона предыдущего слоя; веса имеют только положительные значения.

При выполнении условия $\dfrac{1+\sum_i a_i u_i}{1+\sum_j b_j v_j} > 1$, выход нейрона принимает следующее значение:

$$y^e = \frac{\sum_i a_i u_i - \sum_j b_j v_j}{1+\sum_j b_j v_j}.$$

Если $\sum_j b_j v_j << 1$, то значение y^e может быть аппроксимировано $y^e = \sum_i a_i u_i - \sum_j b_j v_j$, что соответствует функции линейного порогового элемента с нулевым порогом.

Особенностью когнитрона является то, его весовые коэффициенты могут только возрастать в процессе обучения. Причем этот рост не ограничен. Однако условием ограничения значения y^e является одинаковый диапазон изменения значений возбуждающих и тормозящих входов:

$$y^e = \frac{\sum_i a_i u_i}{\sum_j b_j v_j} - 1, \quad \text{при} \quad \sum_i a_i u_i >> 1 \quad \text{и} \quad \sum_j b_j v_j >> 1.$$

В этом случае $\sum_i a_i u_i = \varepsilon\Psi$ и $\sum_j b_j v_j = \sigma\Psi$, где ε, σ – константы, Ψ – диапазон изменения значений возбуждающих и тормозящих входов.

$$y^e = \frac{(\varepsilon-\sigma)\Psi}{1+\sigma\Psi} = \frac{\varepsilon-\sigma}{2\sigma}\left(1+\tanh\frac{\log\sigma\Psi}{2}\right).$$

Значение выхода нейрона изменяется по закону Вебера-Фехнера, используемого в нейрофизиологии для аппроксимации нелинейных соотношений значений на входах/выходах сенсорных нейронов. При использовании этого соотношения нейрон когнитрона достаточно точно моделирует поведение биологического прототипа.

Тормозящие нейроны. Тормозящий нейрон соответствует области связи возбуждающего нейрона. Веса тормозящих нейронов устанавливаются заранее таким образом, чтобы их сумма была равна единице. Кроме того, они не изменяются в процессе обучения. Значение на выходе тормозящего нейрона представляет собой среднее арифметическое значений выходов возбуждающих нейронов из соответствующей области связи.

$$y^m = \sum_i c_i y_i^e,$$

где $\sum_i c_i = 1$, c_i – вес i-го возбуждающего входа.

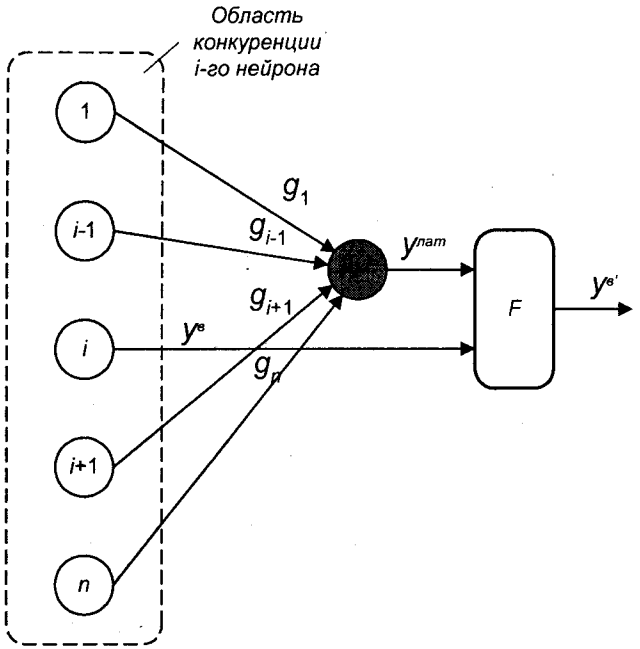


Рис. 2.14. Нейрон когнитрона с ускоренным торможением

Латеральное торможение. на каждый нейрон 2-го слоя оказывают латеральное торможение нейроны из области его конкуренции (рис. 2.13). Соответствующий тормозящий нейрон суммирует выходы всех нейронов из этой области и вырабатывает сигнал, тормозящий воздействие на целевой нейрон. Данный подход эффективен для нейрофизиологического моделирования, однако требует существенных вычислительных затрат из-за возможного большего числа итераций при стабилизации состояния нейрона.

Для реализации ускоренного латерального торможения может быть предложено решение (рис. 2.14), позволяющее выполнить все вычисления за одну итерацию.

На выходе дополнительного нейрона латерального торможения формируется сигнал:

$$y^{лат} = \sum_i g_i y_i ,$$

где y_i – выход i-го нейрона в области конкуренции; g_i – вес связи от i-го нейрона к нейрону латерального торможения; $\sum_i g_i = 1$.

Обучение когнитрона. Процесс обучения когнитрона представляет собой обучение без учителя, в результате которого при получении обучающего набора входных образов сеть самоорганизуется за счет изменения весовых коэффициентов.

При инициализации значения на всех выходах нейронов идентичны.

Соседние нейроны слоя (из области конкуренции) конкурируют между собой. Это обеспечивается за счет значительного перекрытия областей связей соседних нейронов.

В результате обучения в заданной области слоя возбуждается только один нейрон, который будет оказывать латерально-тормозящее воздействие на соседние нейроны из области его конкуренции. В результате обучения синапсы возбужденного нейрона будут усиливаться, а синапсы соседних нейронов останутся неизменными.

Возбуждающие веса данного нейрона изменяются следующим образом:

$$\delta a_i = \eta c_j u_j ,$$

где c_j – вес тормозящей связи j-го нейрона с тормозящим нейроном i; u_j – выход j-го нейрона; a_i – вес i-го возбуждающего входа; q – коэффициент скорости обучения.

Изменение же значений тормозящих весов этого нейрона вычисляется по формуле:

$$\delta b_i = \frac{\eta \sum_j a_j u_j}{2 y_i^m} .$$

В случае, когда отсутствуют возбужденные нейроны в области конкуренции (например, на этапе инициализации процесса обучения), изменения весов вычисляются следующим образом:

$$\delta a_i = \eta' c_j u_j , \quad \delta b_i = \eta' y_i^m ,$$

где η' – положительный коэффициент скорости обучения ($\eta' < \eta$).

Благодаря рассмотренной процедуре обучения у активизированных нейронов возбуждающие веса увеличиваются сильнее, чем тормозящие. И наоборот, у нейронов, которые проиграли конкуренцию, возбуждающие веса возрастают незначительно, а тормозящие – сильнее.

При обучении веса нейронов 2-го (и последующих) слоев настраиваются так, чтобы активные сигналы на их выходах соответствали векторам, которые предъявлялись в процессе обучения.

2.8. Неокогнитрон

Развитием когнитрона является *неокогнитрон*, представляющий собой многоуровневую иерархическую нейронную сеть, организация и принципы функционирования которой наиболее соответствуют модели зрительной коры головного мозга. Неокогнитрон достаточно универсален и находит широкое применение не только для обработки визуальных данных, но и в качестве обобщенной системы распознавания образов.

Он имеет иерархическую структуру, состоящую из последовательности слоев нейронов (рис. 2.15). Входной образ подается на первый слой и передается далее до достижения выходного слоя, в котором он распознается.

Входной слой неокогнитрона распознает линии и углы определенной ориентации. Каждый нейрон в слое, близком к входному, реагирует на определенные образы в определенном месте с определенной ориентацией. Каждый последующий слой имеет более абстрактную, менее специфическую реакцию по сравнению с предыдущим. В последующих слоях распознаются все более сложные образы независимо от их положения, размера, ориентации и искажений.

Каждый слой неокогнитрона состоит из двух плоскостей, разделенных на двумерные массивы нейронов. Первая плоскость, содержащая простые нейроны, получает сигналы с выходов сложных нейронов предыдущего слоя, выделяет определенные образы и затем передает их во вторую плоскость данного слоя, содержащую сложные нейроны, где образы обрабатываются таким образом, чтобы обеспечить их меньшую позиционную зависимость.

Внутри отдельного слоя массивы простых и сложных нейронов соответствуют друг другу.

Рецептивное поле каждого нейрона от слоя к слою возрастает, количество же нейронов в слое при этом уменьшается. Наконец, в каждом массиве выходного слоя имеется только один сложный нейрон, который реагирует на определенный входной образ. В процессе распознавания входной образ подается на вход неокогнитрона, а вычисления осуществляются слой за слоем. Так как только небольшая часть входного образа подается на вход каждого простого нейрона входного слоя, некоторые простые нейроны реагируют на наличие характеристик, которым они обучены, и возбуждаются. В следующих слоях выделяются более сложные характеристики как определенные комбинации выходов сложных нейронов, и уменьшается позиционная зависимость.

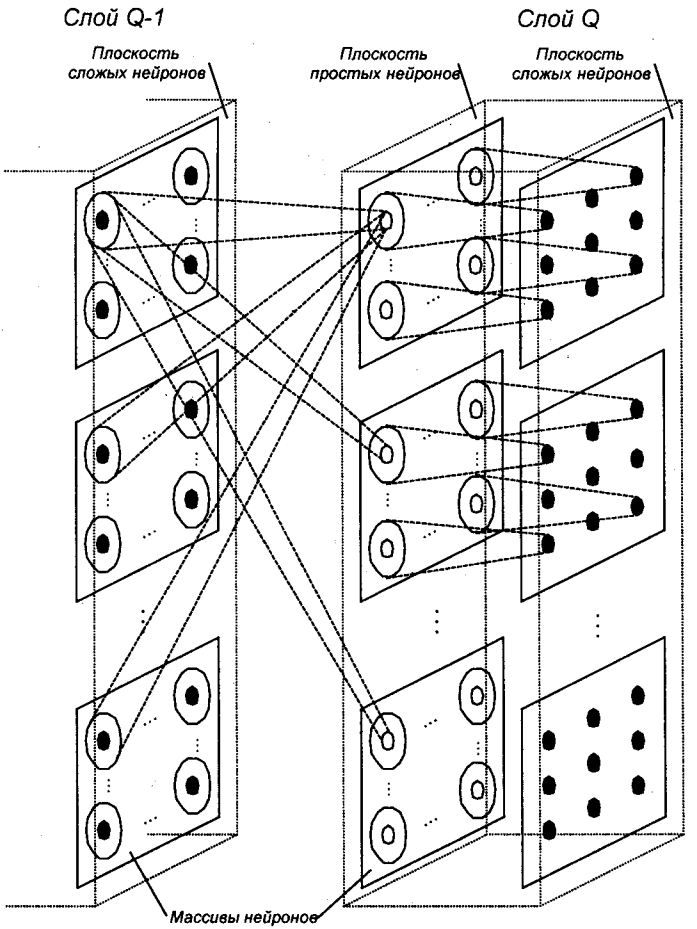


Рис. 2.15. Структура неокогнитрона

Если используется латеральное торможение, то возбуждается только один нейрон выходного слоя с максимальным значением выхода. Однако это часто является не лучшим вариантом.

Обычно используется подход, при котором будут активизироваться несколько нейронов с различной степенью возбуждения, и входной образ должен быть определен с учетом соотношения их выходов. Это позволяет улучшить точность распознавания.

Простые нейроны. Отдельный массив плоскости простых нейронов настраивается на один специфический входной образ. Каждый простой нейрон массива реагирует на ограниченную область входного образа, называемую его рецептивной областью. Нейрон реагирует, если часть образа, на которую он настроен, встречается во входном образе и обнаружена в его рецептивной области. Другие массивы простых нейронов первой плоскости в этом слое могут быть настроены, например, на повороты образов. Причем для выделения каждого дополнительного образа (или его версии) требуется дополнительная плоскость.

Рецептивные области простых нейронов в каждом массиве первой плоскости перекрываются для покрытия всего входного поля этого слоя. Каждый такой нейрон получает сигналы от соответствующих рецептивных областей всех массивов второй плоскости из предыдущего слоя. Следовательно, простой нейрон реагирует на появление своего образа в любой сложной плоскости предыдущего слоя, если он окажется внутри его рецептивной области.

Простые нейроны неокогнитрона имеют такие же свойства, что и в когнитроне, и для определения их выхода используются те же формулы.

Простые нейроны в отличие от сложных имеют настраиваемые веса связей, соединяющих простой нейрон со сложными нейронами в предыдущем слое, настраиваемые таким образом, чтобы выработать максимальную реакцию на определенные образы. Помимо возбуждающих синапсов, к простому нейрону подключены тормозящие, стремящиеся уменьшить значение на его выходе.

Сложные нейроны. Сложные нейроны решают задачу уменьшения позиционной зависимости реакции неокогнитрона на образы. Для этого на входы каждого сложного нейрона подаются выходные сигналы с набора простых нейронов из соответствующего множества первой плоскости того же слоя.

Активизация любого простого нейрона из рецептивной области сложного нейрона является достаточным условием для возбуждения данного сложного нейрона. Таким образом, сложный нейрон реагирует на тот же образ, что и простые нейроны в соответствующем ему массиве, но он менее чувствителен к позиции образа, чем любой из них.

Каждый слой сложных нейронов реагирует на все большую область входного образа, по сравнению с предшествующими слоями, что приводит к требуемому уменьшению позиционной зависимости реакции неокогнитрона на образы в целом.

Помимо активизирующих, в плоскости сложных нейронов присутствуют тормозящие нейроны, которые вырабатывают выходные сигналы, пропорциональные квадратному корню из взвешенной суммы квадратов их входных сигналов. При этом на входы тормозящего нейрона подаются сигналы с выходов сложных нейронов из соответствующей рецептивной области для заданного простого нейрона следующего слоя.

В символьном виде:

$$v = \sqrt{\sum_i (b_i u_i)^2} ,$$

где v – выход тормозящего нейрона; i – индекс сложного нейрона, с которым связан тормозящий нейрон; b_i – вес i-й синаптической связи от сложного нейрона к простому тормозящему нейрону; u_i – выход i-го сложного нейрона. Веса b_i монотонно уменьшаются с увеличением расстояния от центра области реакции, при этом $\sum_i b_i = 1$. Однако в процессе обучения эти веса не изменяются. Изменяется только вес тормозящего входа простого нейрона, к которому подключен выход тормозящего нейрона из предыдущего слоя.

На рис. 2.16 показана организация взаимосвязей между простым нейроном и сложными нейронами из одного из массивов предыдущего слоя.

Обучение. Как и для когнитрона, процесс обучения неокогнитрона представляет собой обучение без учителя, в результате которого сеть самоорганизуется. При этом на вход неокогнитрона подается образ, который необходимо распознать, и веса синапсов настраиваются слой за слоем. Значение веса от каждого сложного нейрона к заданному простому увеличивается, когда удовлетворяются следующие два условия:

- активизируется сложный нейрон;
- реакция одного из простых нейронов больше, чем у его соседей из любой из области конкуренции.

Это приводит к тому, что простой нейрон обучается реагировать более сильно на образы, появляющиеся наиболее часто в его рецептивной области. Если распознаваемый образ отсутствует на входе, тормозящий нейрон предохраняет от случайной активизации соответствующий простой нейрон.

Процедура обучения и подход при реализации латерального торможения когнитрона и неокогнитрона аналогичны. При этом выходы простых и сложных нейронов являются непрерывными, неотрицательными и изменяются по линейному закону.

При срабатывании на входной образ простого нейрона его веса должны быть увеличены. Также увеличиваются веса всех

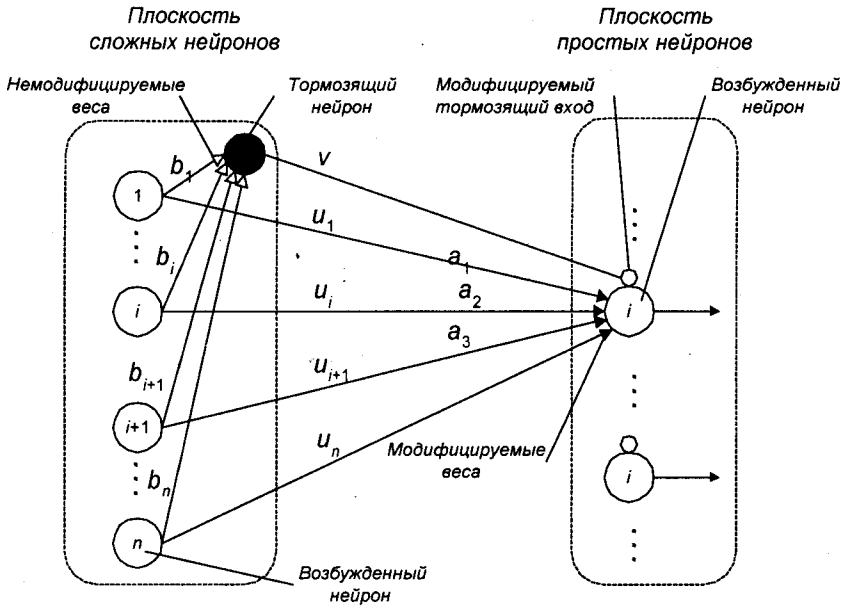


Рис. 2.16. Взаимосвязь простого нейрона
со сложными нейронами из предыдущего слоя

простых нейронов из данного массива для этого самого образа. Таким образом, все нейроны в массиве обучаются распознавать одни и те же свойства образа, и после обучения будут делать это независимо от позиции образа в поле сложных нейронов из предшествующего слоя.

Это определяет способность неокогнитрона к самовосстановлению. Так, если активизируемый нейрон выйдет из строя, среди других выбирается другой, реагирующий наиболее сильно, который и будет обучен распознаванию входного образа, заменяя отказавший нейрон.

При обучении с учителем требуемые значения выходов нейронов каждого слоя определяется заранее. Их веса настраиваются с использованием обычных процедур. Например, входной слой настраивался для распознавания отрезков линий в различных ориентациях. Последующие слои обучаются реагировать на более сложные свойства до тех пор, пока в выходном слое требуемый образ не будет выделен.

Глава 3

НЕЧЕТКИЕ НЕЙРОННЫЕ СЕТИ И ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Аппарат нечетких множеств и нечеткой логики уже давно с успехом применяется для решения задач, в которых исходные данные являются ненадежными и слабо формализованными. Сильные стороны такого подхода:

- описание условий и метода решения задачи на языке, близком к естественному;
- универсальность: согласно теореме FAT (Fuzzy Approximation Theorem), доказанной Б. Коско (B. Kosko) в 1993 г., любая математическая система может быть аппроксимирована системой, основанной на нечеткой логике;
- эффективность (связана с универсальностью), поясняемая рядом теорем, аналогичных теоремам о полноте для искусственных нейронных сетей, например, теоремой вида: для каждой вещественной непрерывной функции g, заданной на компакте U и для произвольного $\varepsilon > 0$ существует нечеткая экспертная система, формирующая выходную функцию $f(\mathbf{x})$ такую, что

$$\sup_{x \in U} \|g(x) - f(x)\| \leq \varepsilon,$$

где $\|\bullet\|$ – символ принятого расстояния между функциями.

Вместе с тем, для нечетких систем характерны и определенные недостатки:

- исходный набор постулируемых нечетких правил формулируется экспертом-человеком и может оказаться неполным или противоречивым;
- вид и параметры функций принадлежности, описывающих входные и выходные переменные системы, выбираются субъективно и могут оказаться не вполне отражающими реальную действительность.

Для устранения, по крайней мере, частично, указанных недостатков было предложено создавать нечеткие системы адаптивными, корректируя, по мере их работы, правила и параметры функций принадлежности. Одними из самых удачных примеров таких систем являются нечеткие нейронные сети.

Нечеткая нейронная сеть формально по структуре идентична многослойной нейронной сети с обучением, например, по алгоритму обратного распространения ошибки, но скрытые слои в ней соответствуют этапам функционирования нечеткой системы:

- первый слой нейронов выполняет функцию введения нечеткости (fuzzification) на основе заданных функций принадлежности входов;
- второй слой отображает совокупность нечетких правил;
- третий слой выполняет функцию приведения к четкости (defuzzification).

Каждый из этих слоев характеризуется набором параметров (функциями принадлежности, нечеткими решающими правилами, активационными функциями, весами связей), настройка которых производится, по сути, так же, как и для обычных нейронных сетей.

Ниже рассматриваются теоретические аспекты создания подобных сетей, а именно, аппарат нечеткой логики и собственно нечеткие нейронные сети применительно к задачам принятия решений в условиях неопределенности.

Кроме того, в этой главе существенное внимание уделено рассмотрению генетических алгоритмов, которые как и нечеткие нейронные сети относятся к классу гибридных систем. Наиболее востребованным является приложение, в котором генетические алгоритмы используются в процессе обучения нейронных сетей, в том числе и нечетких, для поиска оптимальной структуры и набора весовых коэффициентов.

3.1. Нечеткая информация

Пожалуй, наиболее поразительным свойством человеческого интеллекта является способность принимать правильные решения в обстановке неполной и нечеткой информации. Построение моделей приближенных рассуждений человека и использование их в интеллектуальных компьютерных системах представляет сегодня одно из самых перспективных направлений развития современной вычислительной техники.

Значительный вклад в это направление внес Л. Заде (L. Zadeh). Его работа «Fuzzy Sets», опубликованная в 1965 г. в журнале «Information and Control», явилась толчком к развитию новой математической теории. Заде расширил классическое понятие множества, допустив, что характеристическая функция (функция принадлежности элемента множеству) может принимать любые значения в интервале (0, 1), а не только значения 0 либо 1. Такие множества были названы им *нечеткими* (*fuzzy*). Заде определил также ряд операций над нечеткими множествами и предложил обобщение известных методов логического вывода modus ponens и modus tollens.

Введя затем понятие *лингвистической переменной* и допустив, что в качестве ее значений (термов) выступают нечеткие множества, Заде предложил аппарат для описания процессов интеллектуальной деятельности, включая нечеткость и неопределенность выражений. Это позволило создать фундамент теории нечетких множеств и нечеткой логики, а также предпосылки для внедрения методов нечеткого управления в инженерную практику.

Смещение центра исследований нечетких систем в сторону практических приложений привело к постановке целого ряда проблем таких, как новые архитектуры компьютеров для нечетких вычислений, элементная база нечетких компьютеров и контроллеров, инструментальные средства разработки, инженерные методы расчета и разработки нечетких систем управления и многое другое.

Математическая теория нечетких множеств позволяет описывать нечеткие понятия и знания, оперировать этими знаниями и делать нечеткие выводы. Нечеткое управление оказывается особенно полезным, когда исследуемые процессы являются слишком сложными для анализа с помощью общепринятых методов, или когда доступные источники информации интерпретируются качественно, неточно или неопределенно. Нечеткая логика, предоставляющая эффективные средства отображения неопределенностей и неточностей реального мира, и на которой основано нечеткое управление, ближе к человеческому мышлению и естественным языкам, чем традиционные логические системы.

3.1.1. Нечеткие множества

Пусть E – универсальное множество, x – элемент E, а G – некоторое свойство. Обычное (четкое) подмножество A универсального множества E, элементы которого удовлетворяют свойству G, определяется как множество упорядоченных пар:

$$A = \{\mu_A(x)/x\},$$

где $\mu_A(x)$ – *характеристическая функция*, принимающая значение 1, если x удовлетворяет свойству G, и 0 – в противном случае.

Нечеткое подмножество отличается от обычного тем, что для элементов x из E нет однозначного ответа «да–нет» относительно свойства G. В связи с этим нечеткое подмножество A универсального множества E определяется как множество упорядоченных пар:

$$A = \{\mu_A(x)/x\},$$

где $\mu_A(x)$ – характеристическая функция принадлежности (или просто функция принадлежности), принимающая значения в некотором вполне упорядоченном множестве M (например, $M = [0, 1]$).

Функция принадлежности указывает степень (или уровень) принадлежности элемента x подмножеству A. Множество M называют множеством принадлежностей. Если $M = \{0, 1\}$, то нечеткое подмножество A может рассматриваться как обычное или четкое множество.

Примеры записи нечеткого множества

Пусть $E = \{x_1, x_2, x_3, x_4, x_5\}$, $M = [0, 1]$; A – нечеткое множество, для которого $\mu_A(x_1) = 0{,}3$; $\mu_A(x_2) = 0$; $\mu_A(x_3) = 1$; $\mu_A(x_4) = 0{,}5$; $\mu_A(x_5) = 0{,}9$.

Тогда A можно представить в виде: $A = \{0{,}3/x_1;\ 0/x_2;\ 1/x_3;\ 0{,}5/x_4;\ 0{,}9/x_5\}$ или $A = 0{,}3/x_1 + 0/x_2 + 1/x_3 + 0{,}5/x_4 + 0{,}9/x_5$.

Замечание. Здесь знак «+» не является обозначением операции сложения, а имеет смысл объединения.

Основные характеристики нечетких множеств

Пусть $M = [0, 1]$ и A – нечеткое множество с элементами из универсального множества E и множеством принадлежностей M.

- Величина $\sup\limits_{x \in U} \mu_A(x)$ называется *высотой* нечеткого множества A. Нечеткое множество A *нормально*, если его высота равна 1, т. е. верхняя граница его функции принадлежности равна 1 ($\sup\limits_{x \in U} \mu_A(x) = 1$). При $\sup\limits_{x \in U} \mu_A(x) < 1$ нечеткое множество называется *субнормальным*.
- Нечеткое множество *пусто*, если $\forall x \in E\ \mu_A(x) = 0$. Непустое субнормальное множество можно нормализовать по формуле:

$$\mu_A(x) := \frac{\mu_A(x)}{\sup\limits_{x \in U} \mu_A(x)}.$$

- Нечеткое множество *унимодально*, $\mu_A(x) = 1$ только на одном x из E.

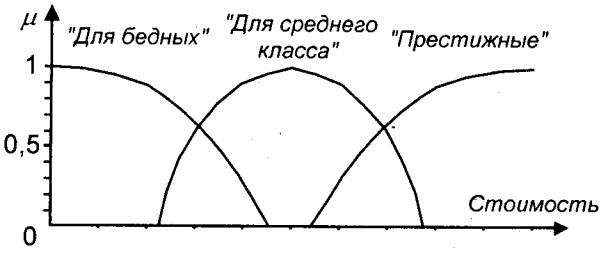


Рис. 3.1. Примеры функций принадлежности

- *Носителем* нечеткого множества A является обычное подмножество со свойством $\mu_A(x) > 0$, т. е. *носитель* $A = \{x/\mu_A(x) > 0\}$, $\forall x \in E$.
- Элементы $x \in E$, для которых $\mu_A(x) = 0{,}5$ называются *точками перехода* множества A.

Примеры нечетких множеств

Пример 1. Пусть $E = \{0, 1, 2, \ldots, 10\}$, $M = [0, 1]$. Нечеткое множество «*несколько*» можно определить следующим образом: «*несколько*» = 0,5/3 + 0,8/4 + 1/5 + 1/6 + 0,8/7 + 0,5/8; его характеристики: *высота* = 1, *носитель* = {3, 4, 5, 6, 7, 8}, *точки перехода* – {3,8}.

Пример 2. Пусть $E = \{0, 1, 2, 3, \ldots, n, \ldots\}$. Нечеткое множество «*малый*» можно определить:

$$\text{«малый»} = \mu_{\text{малый}}(n) = \frac{1}{1 + \left(\frac{n}{10}\right)^2} / n.$$

Пример 3. Пусть $E = \{1, 2, 3, \ldots, 100\}$ и соответствует понятию «*возраст*», тогда нечеткое множество «*молодой*», может быть определено следующим образом:

$$\mu_{\text{молодой}}(x) = \begin{cases} 1, & x \in [1, 25], \\ \dfrac{1}{1 + \left(\dfrac{x - 25}{5}\right)^2}, & x > 25. \end{cases}$$

Нечеткое множество «*молодой*» на универсальном множестве E' = {*Иванов, Петров, Сидоров, ...*} задается с помощью функции принадлежности $\mu_{\text{молодой}}(x)$ на $E = \{1, 2, 3, \ldots, 100\}$ (возраст),

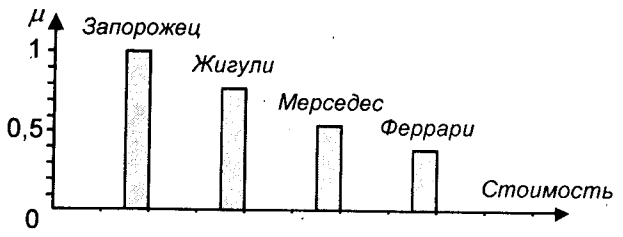


Рис. 3.2. Пример задания нечеткого множества

называемой по отношению к E' функцией совместимости, при этом:

$\mu_{молодой}(Сидоров) := \mu_{молодой}(x)$, где x – возраст Сидорова.

Пример 4. Пусть E = {*Запорожец, Жигули, Мерседес, ...*} – множество марок автомобилей, а E' = [0, ∞) – универсальное множество «*стоимость*», тогда на E' можно определить нечеткие множества типа: «*для бедных*», «*для среднего класса*», «*престижные*», с функциями принадлежности вида рис. 3.1.

• Имея эти функции и зная стоимости автомобилей из E в данный момент времени, тем самым можно определить на E' нечеткие множества с этими же названиями.

Так, например, нечеткое множество «*для бедных*», заданное на универсальном множестве E = {*Запорожец, Жигули, Мерседес, ...*} выглядит так, как показано на рис. 3.2.

Аналогично можно определить Нечеткое множество «*скоростные*», «*средние*», «*тихоходные*» и т. д.

Пример 5. Пусть E – множество целых чисел:

$$E = \{-8, -5, -3, 0, 1, 2, 4, 6, 9\}.$$

Тогда нечеткое подмножество чисел, по абсолютной величине близких к нулю можно определить, например, так:

A = {0/–8 + 0,5/–5 + 0,6/–3 + 1/0 + 0,9/1 + 0,8/2 + 0,6/4 + 0,3/6 + 0/9}.

Методы построения функций принадлежности нечетких множеств

Существуют прямые и косвенные методы построения функций принадлежности.

При использовании *прямых методов* эксперт просто задает для каждого $x \in E$ значение $\mu_A(x)$. Как правило, прямые методы задания функции принадлежности используются для измеримых понятий, таких как скорость, время, расстояние, давление, температура и т. д., или когда выделяются полярные значения.

Во многих задачах при характеристике объекта можно выделить набор признаков и для каждого из них определить полярные значения, соответствующие значениям функции принадлежности, 0 или 1. Для конкретного объекта эксперт, исходя из приведенной шкалы, задает $\mu_A(x) \in [0, 1]$, формируя векторную функцию принадлежности $\{\mu_A(x_1), \mu_A(x_2), ..., \mu_A(x_n)\}$.

Разновидностью прямых методов построения функций принадлежности являются *прямые групповые методы*, когда, например, группе экспертов предъявляют конкретный объект, и каждый должен дать один из двух ответов: принадлежит или нет этот объект к заданному множеству. Тогда число утвердительных ответов, деленное на общее число экспертов, дает значение функции принадлежности объекта к данному нечеткому множеству.

Косвенные методы определения значений функции принадлежности используются в случаях, когда нет измеримых элементарных свойств, через которые определяется нечеткое множество. Как правило, это методы попарных сравнений. Если бы значения функций принадлежности были известны, например, $\mu_A(x_i) = w_i$, $i = 1, 2, ..., n$, то попарные сравнения можно представить матрицей отношений $\boldsymbol{A} = \{a_{ij}\}$, где $a_{ij} = w_i/w_j$ (операция деления).

На практике эксперт сам формирует матрицу $\boldsymbol{A}$, при этом предполагается, что диагональные элементы равны 1, а для элементов, симметричных относительно главной диагонали, $a_{ij} = 1/a_{ij}$, т. е. если один элемент оценивается в α раз значимее чем другой, то этот последний должен быть в $1/\alpha$ раз значимее, чем первый. В общем случае задача сводится к поиску вектора $\boldsymbol{w}$, удовлетворяющего уравнению вида $\boldsymbol{Aw} = \lambda_{max}\boldsymbol{w}$, где λ_{max} – наибольшее собственное значение матрицы $\boldsymbol{A}$. Поскольку матрица $\boldsymbol{A}$ положительна по построению, решение данной задачи существует и является положительным.

Использование типовых форм кривых для задания функций принадлежности (в форме (L–R)-типа – см. ниже) с уточнением их параметров в соответствии с данными эксперимента.

Использование относительных частот по данным эксперимента в качестве значений принадлежности.

3.1.2. Операции над нечеткими множествами

Логические операции

Включение

Пусть A и B – нечеткие множества на универсальном множестве E. Тогда A содержится в B, если $\forall x \in E \ \mu_A(x) \le \mu_B(x)$.

Обозначение: $A \subset B$.

Иногда используют термин «*доминирование*», т. е. в случае когда $A \subset B$, говорят, что B доминирует A.

Равенство

A и B равны, если $\forall x \in E \ \ \mu_A(x) = \mu_B(x)$.

Обозначение: $A = B$.

Дополнение

Пусть $M = [0, 1]$, A и B – нечеткие множества, заданные на E. A и B дополняют друг друга, если

$\forall x \in E \ \ \mu_A(x) = 1 - \mu_B(x)$.

Обозначение: $B = \overline{A}$ или $A = \overline{B}$.

Очевидно, что $\left(\overline{\overline{A}}\right) = A$ (дополнение определено для $M = [0, 1]$, но очевидно, что его можно определить для любого упорядоченного M).

Пересечение

$A \cap B$ – наибольшее нечеткое подмножество, содержащееся одновременно в A и B:

$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$.

Объединение

$A \cup B$ – наименьшее нечеткое подмножество, включающее как A, так и B, с функцией принадлежности:

$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$.

Разность

$A - B = A \cap \overline{B}$ с функцией принадлежности:

$\mu_{A-B}(x) = \mu_{A \cap \overline{B}}(x) = \min(\mu_A(x), 1 - \mu_B(x))$.

Дизъюнктивная сумма

$A \oplus B = (A - B) \cup (B - A) = (A \cap \overline{B}) \cup (\overline{A} \cap B)$ с функцией принадлежности:

$\mu_{A-B}(x) = \max\{[\min\{\mu_A(x), 1 - \mu_B(x)\}]; [\min\{1 - \mu_A(x), \mu_B(x)\}]\}$

Наглядное представление логических операций над нечеткими множествами

Для нечетких множеств можно строить визуальное представление. Рассмотрим прямоугольную систему координат, на оси ординат которой откладываются значения $\mu_A(x)$, на оси абсцисс в произвольном порядке расположены элементы E. Если E по своей природе упорядочено, то этот порядок желательно сохранить в расположении элементов на оси абсцисс. Такое представление делает наглядными простые логические операции над нечеткими множествами (рис. 3.3).

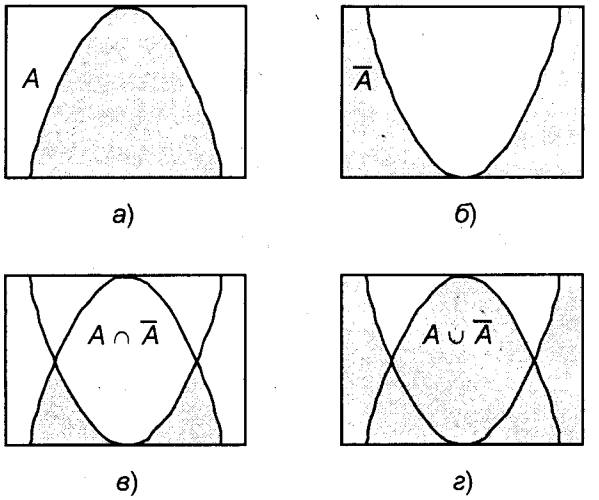


Рис. 3.3. Графическая интерпретация нечетких логических операций:

а – нечеткое множество A; *б* – $\overline{A}$; *в* – $A \cap \overline{A}$; *г* – $A \cup \overline{A}$

Свойства операций объединения и пересечения

Пусть A, B, C – нечеткие множества, тогда выполняются следующие свойства:

- $\left.\begin{array}{l} A \cap B = B \cap A \\ A \cup B = B \cup A \end{array}\right\}$ – коммутативность;
- $\left.\begin{array}{l} (A \cap B) \cap C = A \cap (B \cap C) \\ (A \cup B) \cup C = A \cup (B \cup C) \end{array}\right\}$ – ассоциативность;
- $\left.\begin{array}{l} A \cap A = A \\ A \cup A = A \end{array}\right\}$ – идемпотентность;
- $\left.\begin{array}{l} A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \\ A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \end{array}\right\}$ – дистрибутивность;
- $A \cup \varnothing = A$, где $\varnothing$ – пустое множество, т.е. $\mu_{\varnothing}(x) = 0 \ \forall x \in E$;
- $A \cap \varnothing = \varnothing$;
- $A \cap E = A$, где E – универсальное множество;
- $A \cup E = E$;
- $\left.\begin{array}{l} \overline{A \cap B} = \overline{A} \cup \overline{B} \\ \overline{A \cup B} = \overline{A} \cap \overline{B} \end{array}\right\}$ – формулы де Моргана.

В отличие от четких множеств, для нечетких множеств в общем случае:

$$A \cap \overline{A} \neq \varnothing,$$
$$A \cup \overline{A} \neq E$$

Замечание. Введенные выше операции над нечеткими множествами основаны на использовании операций max и min. В теории нечетких множеств рассмотрены вопросы построения обобщенных, параметризованных операторов пересечения, объединения и дополнения, позволяющих учесть разнообразные смысловые оттенки соответствующих им связок «И», «ИЛИ», «НЕ».

Один из подходов к обобщению операторов пересечения и объединения заключается в их определении в классе треугольных норм и конорм.

Треугольной нормой (*t-нормой*) называется двуместная действительная функция T: [0, 1]×[0, 1]→[0, 1], удовлетворяющая следующим условиям:

1) $T(0, 0)=0$; $T(\mu_A, 1) = \mu_A$; $T(1, \mu_A) = \mu_A$ – *ограниченность*;
2) $T(\mu_A, \mu_B) \le T(\mu_C, \mu_D)$, если $\mu_A \le \mu_C$, $\mu_B \le \mu_D$ – *монотонность*;
3) $T(\mu_A, \mu_B) = T(\mu_B, \mu_A)$ – *коммутативность*;
4) $T(\mu_A, T(\mu_B, \mu_C)) = T(T(\mu_A, \mu_B), \mu_C)$ – *ассоциативность*;

Примеры t-норм:

$\min(\mu_A, \mu_B)$
произведение $\mu_A \cdot \mu_B$
$\max(0, \mu_A + \mu_B - 1)$.

Треугольной конормой (*t-конормой*) называется двуместная действительная функция S:[0, 1]×[0, 1]→ [0, 1], со свойствами:

1) $S(1, 1) = 1$; $S(\mu_A, 0) = \mu_A$; $S(0, \mu_A) = \mu_A$ – *ограниченность*;
2) $S(\mu_A, \mu_B) \ge S(\mu_C, \mu_D)$, если $\mu_A \ge \mu_C$, $\mu_B \ge \mu_D$ – *монотонность*;
3) $S(\mu_A, \mu_B) = S(\mu_B, \mu_A)$ – *коммутативность*;
4) $S(\mu_A, S(\mu_B, \mu_C)) = S(S(\mu_A, \mu_B), \mu_C)$ – *ассоциативность*.

Примеры t-конорм:

$\max(\mu_A, \mu_B)$
$\mu_A + \mu_B - \mu_A \cdot \mu_B$
$\min(1, \mu_A + \mu_B)$.

Алгебраические операции над нечеткими множествами

Алгебраическое произведение A и B обозначается $A \cdot B$ и формируется следующим образом:

$\forall x \in E \quad \mu_{A \cdot B}(x) = \mu_A(x)\mu_B(x)$.

Алгебраическая сумма этих множеств обозначается $A \hat{+} B$ и определяется как:

$\forall x \in E \quad \mu_{A \hat{+} B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$.

Для операций $\{\cdot, \hat{+}\}$ справедливы следующие свойства:

- $\left.\begin{array}{l} A \cdot B = B \cdot A \\ A \hat{+} B = B \hat{+} A \end{array}\right\}$ – коммутативность;
- $\left.\begin{array}{l} (A \cdot B) \cdot C = A \cdot (B \cdot C) \\ (A \hat{+} B) \hat{+} C = A \hat{+} (B \hat{+} C) \end{array}\right\}$ – ассоциативность;
- $A \cdot \varnothing = \varnothing$, $A \hat{+} \varnothing = A$, $A \cdot E = A$, $A \hat{+} E = E$;
- $\left.\begin{array}{l} \overline{A \cdot B} = \overline{A} \hat{+} \overline{B} \\ \overline{A \hat{+} B} = \overline{A} \cdot \overline{B} \end{array}\right\}$ – формулы де Моргана.

Не выполняются:

- $\left.\begin{array}{l} A \cdot A = A \\ A \hat{+} A = A \end{array}\right\}$ – идемпотентность;
- $\left.\begin{array}{l} A \cdot (B \hat{+} C) = (A \cdot B) \hat{+} (A \cdot C) \\ A \hat{+} (B \cdot C) = (A \hat{+} B) \cdot (A \hat{+} C) \end{array}\right\}$ – дистрибутивность;
- $A \cdot \overline{A} = \varnothing$, $A \hat{+} \overline{A} = E$.

При совместном использовании операций $\{\cup, \cap, \hat{+}, \cdot\}$ справедливы свойства:

- $A \cdot (B \cup C) = (A \cdot B) \cup (A \cdot C)$;
- $A \cdot (B \cap C) = (A \cdot B) \cap (A \cdot C)$;
- $A \hat{+} (B \cup C) = (A \hat{+} B) \cup (A \hat{+} C)$;
- $A \hat{+} (B \cap C) = (A \hat{+} B) \cap (A \hat{+} C)$.

На основе операции алгебраического произведения определена операция *возведения в степень* α нечеткого множества A, где α – положительное число. Нечеткое множество A_α определяется функцией принадлежности $\mu_A{}^\alpha = \mu^\alpha{}_A(x)$. Частным случаем возведения в степень являются:

- $\mathrm{CON}(A) = A^2$ – операция *концентрирования (уплотнения)*,
- $\mathrm{DIL}(A) = A^{0,5}$ – операция *растяжения*,

которые используются при работе с лингвистическими неопределенностями (рис. 3.4).

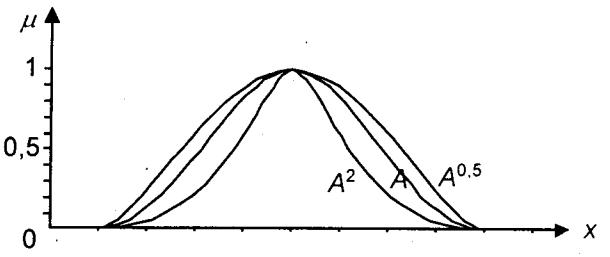


Рис. 3.4. Операции концентрирования (уплотнения) и растяжения

Умножение на число. Если α – положительное число, такое, что $\alpha \max\limits_{x \in A} \mu_A(x) \le 1$, то нечеткое множество αA имеет функцию принадлежности:

$$\mu\alpha A(x) = \alpha\mu A(x).$$

Выпуклая комбинация нечетких множеств. Пусть A_1, A_2, …, A_n – нечеткие множества универсального множества E, а ω_1, ω_2, …, ω_n – неотрицательные числа, сумма которых равна 1.

Выпуклой комбинацией A_1, A_2, …, A_n называется нечеткое множество A с функцией принадлежности:

$$\forall x \in E \quad \mu_A(x_1, x_2, ..., x_n) = \omega_1\mu_{A1}(x) + \omega_2\mu_{A2}(x) + ... + \omega_n\mu_{Ai}(x).$$

Декартово (прямое) произведение нечетких множеств. Пусть A_1, A_2, …, A_n – нечеткие подмножества универсальных множеств E_1, E_2, …, E_n соответственно. Декартово или прямое произведение $A = A_1 \times A_2 \times ... \times A_n$ является нечетким подмножеством множества $E = E_1 \times E_2 \times ... \times E_n$ с функцией принадлежности:

$$\mu_A(x_1, x_2, ..., x_n) = \min\{\mu_{A1}(x_1), \mu_{A2}(x_2), ..., \mu_{Ai}(x_n)\}.$$

Оператор увеличения нечеткости используется для преобразования четких множеств в нечеткие и для увеличения нечеткости нечеткого множества.

Пусть A – нечеткое множество, E – универсальное множество и для всех $x \in E$ определены нечеткие множества $K(x)$. Совокупность всех $K(x)$ называется ядром оператора увеличения нечеткости H. Результатом действия оператора H на нечеткое множество A является нечеткое множество вида:

$$H(A, K) = \bigcup_{x \in E} \mu_A(x)K(x),$$

где $\mu_A(x)K(x)$ – произведение числа на нечеткое множество.

Пример. Пусть $E = \{1, 2, 3, 4\}$; $A = 0{,}8/1 + 0{,}6/2 + 0/3 + 0/4$;
$K(1) = 1/1 + 0{,}4/2$; $K(2) = 1/2 + 0{,}4/1 + 0{,}4/3$; $K(3) = 1/3 + 0{,}5/4$;
$K(4) = 1/4$.

Тогда $H(A, K) = \mu_A(1)\,K(1) \cup \mu_A(2)K(2) \cup \mu_A(3)K(3) \cup \mu_A(4)K(4) =$
$= 0{,}8(1/1 + 0{,}4/2) \cup 0{,}6(1/2 + 0{,}4/1 + 0{,}4/3) =$
$= 0{,}8/1 + 0{,}6/2 + 0{,}24/3.$

Четкое множество α-уровня (или уровня α). Множеством α-уровня нечеткого множества A универсального множества E называется *четкое* подмножество $A\alpha$ универсального множества E, определяемое в виде:

$$A\alpha = \{x/\mu_A(x) \ge \alpha\}, \text{ где } \alpha \le 1.$$

Пример. Пусть $A = 0{,}2/x_1 + 0/x_2 + 0{,}5/x_3 + 1/x_4$
Тогда $A_{0,3} = \{x_3, x_4\}$, $A_{0,7} = \{x_4\}$.
Свойство множества α-уровня: если $\alpha_1 \ge \alpha_2$, то $A_{\alpha 1} \le A_{\alpha 2}$.

3.1.3. Нечеткие и лингвистические переменные

Понятия нечеткой и лингвистической переменных используются при описании объектов и явлений с помощью нечетких множеств.

Нечеткая переменная характеризуется тройкой параметров $<\alpha, X, A>$, где

α – наименование переменной,

X – универсальное множество (область определения α),

A – нечеткое множество на X, описывающее ограничения (т. е. $\mu_A(x)$) на значения нечеткой переменной α.

Лингвистическая переменная (ЛП) характеризуется набором параметров $<\beta, T, X, G, M>$, где

β – наименование лингвистической переменной;

T – множество ее значений (терм-множество), представляющих наименования нечетких переменных, областью определения каждой из которых является множество X. Множество T называется базовым терм-множеством лингвистической переменной;

G – синтаксическая процедура, позволяющая оперировать элементами терм-множества T, в частности, генерировать новые термы (значения). Множество $T \cup G(T)$, где $G(T)$ – множество сгенерированных термов, называется расширенным терм-множеством лингвистической переменной;

M – семантическая процедура, позволяющая превратить каждое новое значение лингвистической переменной, образуемое процедурой G, в нечеткую переменную, т. е. сформировать соответствующее нечеткое множество.

Замечание. Для избежания большого количества символов:

- символ β используют как для названия самой переменной, так и для обозначения всех ее значений;

• один и тот же символ используется для обозначения нечеткого множества и его названия, например, терм *«молодой»*, являющийся значением лингвистической переменной β = *«возраст»*, одновременно есть и нечеткое множество M (*«молодой»*).

Присвоение нескольких значений символам предполагает возможность решения неопределенностей с помощью контекста.

Пример. Пусть эксперт определяет толщину выпускаемого изделия с помощью понятий *«малая толщина»*, *«средняя толщина»* и *«большая толщина»*, при этом минимальная толщина равна 10 мм, а максимальная – 80 мм.

Формализация такого описания может быть проведена с помощью следующей лингвистической переменной $<\beta, T, X, G, M>$, где

β – толщина изделия;

T – {*«малая толщина»*, *«средняя толщина»*, *«большая толщина»*};

X – [10, 80];

G – процедура образования новых термов с помощью связок *«и»*, *«или»* и модификаторов типа *«очень»*, *«не»*, *«слегка»*. Например: *«малая или средняя толщина»*, *«очень малая толщина»*;

M – процедура задания на X = [10, 80] нечетких подмножеств A_1 = *«малая толщина»*, A_2 = *«средняя толщина»*, A_3 = *«большая толщина»*, а также нечетких множеств для термов из $G(T)$ в соответствии с правилами трансляции нечетких связок и модификаторов *«и»*, *«или»*, *«не»*, *«очень»*, *«слегка»* и др. над нечеткими множествами вида: $A \cap B$, $A \cup B$, $\overline{A}$, CON $A = A^2$, DIL $A = A^{0,5}$ и т. п.

Замечание. Наряду с рассмотренными выше базовыми значениями лингвистической переменной *«толщина»* (T = {*«малая толщина»*, *«средняя толщина»*, *«большая толщина»*}) возможны значения, зависящие от области определения X. В данном случае значения лингвистической переменной «толщина изделия» могут быть определены как *«около 20 мм»*, *«около 50 мм»*, *«около 70 мм»*, т. е. в виде *нечетких чисел*.

Терм-множество и расширенное терм-множество в условиях примера можно характеризовать функциями принадлежности, приведенными на рис. 3.5 и рис. 3.6.

Нечеткие числа

Нечеткие числа – нечеткие переменные, определенные на числовой оси, т. е. нечеткое число определяется как нечеткое множество A на множестве действительных чисел R с функцией принадлежности $\mu_A(x) \in [0, 1]$, где $x \in R$.

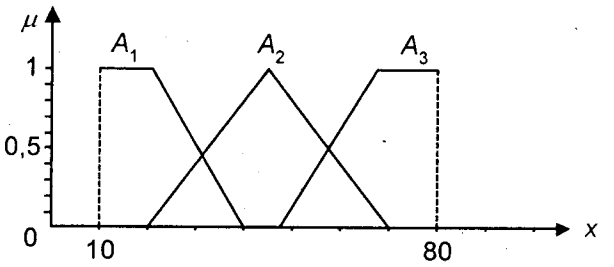


Рис. 3.5. Функции принадлежности нечетких множеств:
A_1 – *«малая толщина»*; A_2 – *«средняя толщина»*; A_3 – *«большая толщина»*

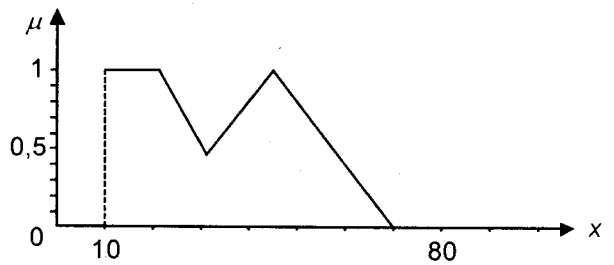


Рис. 3.6. Функция принадлежности нечеткого множества
$A_1 \cup A_2$ – *«малая или средняя толщина»*

Нечеткое число A *нормально*, если $\max\limits_x \mu_A(x) = 1$, и *выпуклое*, если для любых $x \le y \le z$ выполняется

$$\mu_A(x) \ge \mu_A(y) \wedge \mu_A(z).$$

Множество α-уровня нечеткого числа A определяется как

$$A\alpha = \{x / \mu_A(x) \ge \alpha\}.$$

Подмножество $S_A \subset R$ называется *носителем* нечеткого числа A, если:

$$S_A = \{x / \mu_A(x) > 0\}.$$

Нечеткое число A *унимодально*, если условие $\mu_A(x) = 1$ справедливо только для одной точки действительной оси.

Выпуклое нечеткое число A называется *нечетким нулем*, если справедливо:

$$\mu_A(0) = \sup\limits_x (\mu_A(x)).$$

Нечеткое число A *положительно*, если $\forall x \in S_A$, $x > 0$, и *отрицательно*, если $\forall x \in S_A$, $x < 0$.

Операции над нечеткими числами

Расширенные бинарные арифметические операции (сложение, умножение и др.) для нечетких чисел определяются через соответствующие операции для четких чисел с использованием принципа обобщения следующим образом.

Пусть A и B – нечеткие числа, и $\tilde{*}$ – нечеткая операция, соответствующая операции $*$ над обычными числами. Тогда (используя здесь и в дальнейшем обозначения $\bigvee\limits_x$ вместо $\max\limits_x$ и $\bigwedge\limits_x$ вместо $\min\limits_x$) можно записать:

$$C = A \tilde{*} B \Leftrightarrow \mu_C(z) = \bigvee_{Z=X*Y} (\mu_A(x) \wedge \mu_B(y))).$$

Отсюда:

$$C = A \tilde{+} B \Leftrightarrow \mu_C(z) = \bigvee_{Z=X+Y} (\mu_A(x) \wedge \mu_B(y))),$$

$$C = A \tilde{-} B \Leftrightarrow \mu_C(z) = \bigvee_{Z=X-Y} (\mu_A(x) \wedge \mu_B(y))),$$

$$C = A \tilde{\cdot} B \Leftrightarrow \mu_C(z) = \bigvee_{Z=X\cdot Y} (\mu_A(x) \wedge \mu_B(y))),$$

$$C = A \tilde{\div} B \Leftrightarrow \mu_C(z) = \bigvee_{Z=X\div Y} (\mu_A(x) \wedge \mu_B(y))),$$

$$C = \tilde{\max}(A, B) \Leftrightarrow \mu_C(z) = \bigvee_{Z=\max(X, Y)} (\mu_A(x) \wedge \mu_B(y))),$$

$$C = \tilde{\min}(A, B) \Leftrightarrow \mu_C(z) = \bigvee_{Z=\min(X, Y)} (\mu_A(x) \wedge \mu_B(y))).$$

Нечеткие числа (L–R)-типа

Нечеткие числа (L–R)-типа – это разновидность нечетких чисел специального вида, задаваемых по определенным правилам с целью снижения объема вычислений при операциях над ними.

Функции принадлежности нечетких чисел (L–R)-типа задаются с помощью не возрастающих на множестве неотрицательных действительных чисел функций действительного переменного $L(x)$ и $R(x)$, удовлетворяющих свойствам: *а*) $L(-x) = L(x)$, $R(-x) = R(x)$; *б*) $L(0) = R(0)$.

Очевидно, что к классу (L–R)-функций относятся функции, графики которых имеют вид, представленный на рис. 3.7.

Примерами аналитического задания (L–R)-функций могут быть функции:

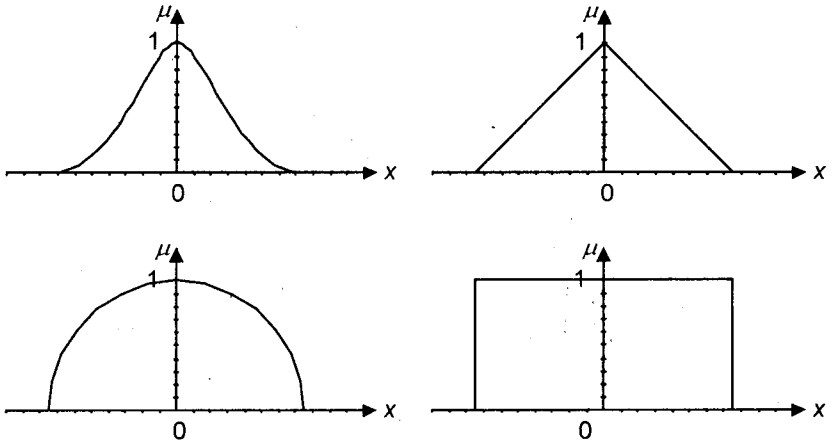


Рис. 3.7. Возможный вид (L–R)-функций

$$L(x) = e^{-|x|^p},\ p \geq 0;$$

$$R(x) = \frac{1}{1+|x|^p},\ p \geq 0.$$

Пусть $L(y)$ и $R(y)$ – функции (L–R)-типа (конкретные). Унимодальное нечеткое число A с *модой* a (т. е. $\mu_A(a) = 1$) с помощью $L(y)$ и $R(y)$ задается следующим образом:

$$\mu_A(x) = \begin{cases} L\left(\dfrac{a-x}{\alpha}\right), & \text{если} \quad x \leq a, \\ R\left(\dfrac{x-a}{\beta}\right), & \text{если} \quad x > a, \end{cases}$$

где a – мода; $\alpha > 0$, $\beta > 0$ – левый и правый коэффициенты нечеткости.

Таким образом, при заданных $L(y)$ и $R(y)$ нечеткое число (унимодальное) задается тройкой $A = (a, \alpha, \beta)$.

Толерантное нечеткое число задается, соответственно, четверкой параметров $A = (a_1, a_2, \alpha, \beta)$, где a_1 и a_2 – границы толерантности, т. е. в промежутке $[a_1, a_2]$ значение функции принадлежности равно 1.

Примеры графиков функций принадлежности нечетких чисел (L–R)-типа приведены на рис. 3.8.

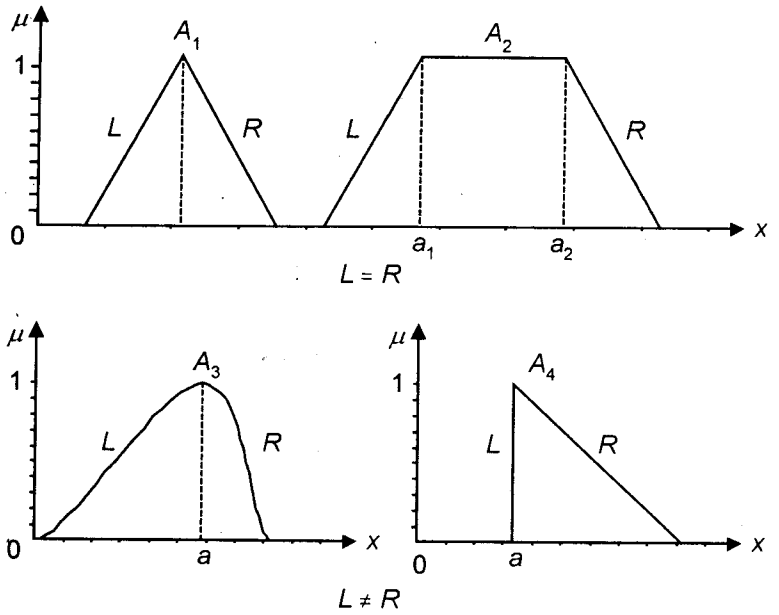


Рис. 3.8. Примеры графиков функций принадлежности нечетких чисел (L–R)-функций

Отметим, что в конкретных ситуациях функции $L(y)$, $R(y)$, а также параметры α, β нечетких чисел (a, α, β) и $(a_1, a_2, \alpha, \beta)$ должны подбираться таким образом, чтобы результат операции (сложения, вычитания, деления и т. д.) был точно или приблизительно равен нечеткому числу с теми же $L(y)$ и $R(y)$, а параметры α' и β' результата не выходили за рамки ограничений на эти параметры для исходных нечетких чисел, особенно, если результат будет участвовать в дальнейших операциях.

Замечание. Моделирование сложных систем с применением аппарата нечетких множеств требует выполнения большого объема операций над разного рода лингвистическими и другими нечеткими переменными. Для удобства исполнения операций, а также для ввода-вывода и хранения данных желательно выбирать функции принадлежности стандартного вида.

Нечеткие множества, которыми приходится оперировать в большинстве задач, являются, как правило, унимодальными и нормальными. Одним из возможных методов аппроксимации унимодальных нечетких множеств является аппроксимация с помощью функций (L–R)-типа. Примеры (L–R)-представлений некоторых лингвистических переменных: приведены в табл.3.1.

Таблица 3.1

Возможное (L–R)-представление некоторых лингвистических переменных

Терм лингвистической переменной	(L–R)-представление	Графическое представление
Средний	$A = (a, \alpha, \beta)_{LR}$ $\alpha = \beta > 0$	α β
Малый	$A = (a, \infty, \beta)_{LR}$ $\alpha = \infty$	$\alpha = \infty$ β
Большой	$A = (a, \alpha, \infty)_{LR}$ $\beta = \infty$	α $\beta = \infty$
Приблизительно в диапазоне	$A = (a_1, a_2, \alpha, \beta)_{LR}$ $\alpha = \beta > 0$	α β a_1 a_2
Определенный	$A = (a, 0, 0)_{LR}$ $\alpha = \beta = 0$	$\alpha = 0$ $\beta = 0$
Разнообразный: зона полной неопределенности	$A = (a, \infty, \infty)_{LR}$ $\alpha = \beta = \infty$	$\alpha = \beta = \infty$

3.1.4. Нечеткие отношения

Пусть $E = E_1 \times E_2 \times \ldots \times E_n$ – прямое произведение универсальных множеств и M – некоторое множество принадлежностей (например, $M = [0, 1]$). Нечеткое n-арное отношение определяется как нечеткое подмножество R на E, принимающее свои значения в M. В случае $n = 2$ и $M = [0, 1]$, нечетким отношением R между множествами $X = E_1$ и $Y = E_2$ будет называться функция R: $(X, Y) \rightarrow [0, 1]$, которая ставит в соответствие каждой паре элементов $(x, y) \in X \times Y$ величину $\mu_R(x, y) \in [0, 1]$.

Обозначение: нечеткое отношение на $X \times Y$ записывается в виде: $x \in X$, $y \in Y$: x R y. В случае, когда $X = Y$, т. е. X и Y совпадают, нечеткое отношение R: $X \times X \rightarrow [0, 1]$ называется нечетким отношением на множестве X.

Пример 1. Пусть $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3, y_4\}$, $M = [0, 1]$. Нечеткое отношение R = X R Y может быть задано, например, табл. 3.2.

Таблица 3.2.

Задание нечеткого отношения

	y_1	y_2	y_3	y_4
x_1	0	0	0,1	0,3
x_2	0	0,8	1	0,7
x_3	1	0,5	0,6	1

Пример 2. Пусть $X = Y = (-\infty, \infty)$, т. е. множество всех действительных чисел. Отношение $x >> y$ можно задать функцией принадлежности:

$$\mu_R = \begin{cases} 0, & \text{если } x \le y, \\ \dfrac{1}{1+\dfrac{1}{(x-y)^2}}, & \text{если } y > x. \end{cases}$$

Пример 3. Отношение R, для которого $\mu_R(x, y) = e^{-k(x-y)^2}$, при достаточно больших k можно интерпретировать так: «x и y близкие друг к другу числа».

Операции над нечеткими отношениями

Объединение

Объединение двух отношений обозначается $R_1 \cup R_2$ и определяется выражением:

$$\mu_{R1 \cup R2}(x, y) = \mu_{R1}(x, y) \vee \mu_{R2}(x, y).$$

Пересечение

Пересечение двух отношений R_1 и R_2 обозначается $R_1 \cap R_2$ и определяется выражением:

$$\mu_{R1 \cap R2}(x, y) = \mu_{R1}(x, y) \wedge \mu_{R2}(x, y).$$

Алгебраическое произведение

Алгебраическое произведение двух отношений R_1 и R_2 обозначается $R_1 \cdot R_2$ и определяется выражением:

$$\mu_{R1 \cdot R2}(x, y) = \mu_{R1}(x, y) \cdot \mu_{R2}(x, y).$$

Алгебраическая сумма

Алгебраическая сумма двух отношений R_1 и R_2 обозначается $R_1 \tilde{+} R_2$ и определяется выражением:

$$\mu_{R1 \tilde{+} R2}(x, y) = \mu_{R1}(x, y) + \mu_{R2}(x, y) - \mu_{R1}(x, y) \cdot \mu_{R2}(x, y).$$

Для введенных операций справедливы следующие свойства дистрибутивности:

$$R_1 \cap (R_2 \cup R_3) = (R_1 \cap R_2) \cup (R_1 \cap R_3),$$
$$R_1 \cup (R_2 \cap R_3) = (R_1 \cup R_2) \cap (R_1 \cup R_3),$$
$$R_1 \cdot (R_2 \cup R_3) = (R_1 \cdot R_2) \cup (R_1 \cdot R_3),$$
$$R_1 \cdot (R_2 \cap R_3) = (R_1 \cdot R_2) \cap (R_1 \cdot R_3),$$
$$R_1 \tilde{+} (R_2 \cup R_3) = (R_1 \tilde{+} R_2) \cup (R_1 \tilde{+} R_3),$$
$$R_1 \tilde{+} (R_2 \cap R_3) = (R_1 \tilde{+} R_2) \cap (R_1 \tilde{+} R_3).$$

Дополнение отношения

Дополнение отношения R обозначается $\overline{R}$ и определяется функцией принадлежности:

$$\mu_{\overline{R}}(x, y) = 1 - \mu_R(x, y)$$

Дизъюнктивная сумма

Дизъюнктивная сумма двух отношений R_1 и R_2 обозначается $R_1 \oplus R_2$ и определяется выражением:

$$R_1 \oplus R_2 = (R_1 \cap \overline{R}_2) \cup (\overline{R}_1 \cap R_2).$$

Обычное отношение, ближайшее к нечеткому

Пусть R – нечеткое отношение с функцией принадлежности $\mu_R(x, y)$. Обычное отношение, ближайшее к нечеткому, обозначается $\underline{R}$ и определяется выражением:

$$\mu_{\underline{R}}(x, y) = \begin{cases} 0, & \text{если } \mu_R(x, y) < 0{,}5, \\ 1, & \text{если } \mu_R(x, y) > 0{,}5, \\ 0 \text{ или } 1, & \text{если } \mu_R(x, y) = 0{,}5. \end{cases}$$

По договоренности принимают $\mu_R(x, y) = 0$ при $\mu_R(x, y) = 0{,}5$.

Композиция (свертка)

Пусть R_1 – нечеткое отношение R_1: $(X \times Y) \to [0, 1]$ между X и Y, и R_2 – нечеткое отношение R_2: $(Y \times Z) \to [0, 1]$ между Y и Z. Нечеткое отношение между X и Z, обозначаемое $R_1 \bullet R_2$, определенное через R_1 и R_2 выражением:

$$\mu_{R1 \bullet R2}(x, z) = \bigvee_y [\mu_{R1}(x, y) \wedge \mu_{R1}(y, z)],$$

называется (max–min)-композицией ((max–min)-сверткой) отношений R_1 и R_2.

Пример. Пусть

R_1	y_1	y_2	y_3
x_1	0,1	0,7	0,4
x_2	1	0,5	0

R_2	z_1	z_2	z_3	z_4
y_1	0,9	0	1	0,2
y_2	0,3	0,6	0	0,9
y_3	0,1	1	0	0,5

Тогда

$R_1 \bullet R_2$	z_1	z_2	z_3	z_4
x_1	0,3	0,6	0,1	0,7
x_2	0,9	0,5	1	0,5

При этом $\mu_{R1 \bullet R2}(x_1, z_1) = [\mu_{R1}(x_1, y_1) \wedge \mu_{R2}(y_1, z_1)] \vee$
$\vee [\mu_{R1}(x_1, y_2) \wedge \mu_{R2}(y_2, z_1)] \vee [\mu_{R1}(x_1, y_3) \wedge \mu_{R2}(y_3, z_1)] =$
$= (0{,}1 \wedge 0{,}9) \vee (0{,}7 \wedge 0{,}3) \vee (0{,}4 \wedge 0{,}1) = 0{,}1 \vee 0{,}3 \vee 0{,}1 = 0{,}3;$
$\mu_{R1 \bullet R2}(x_1, z_2) = (0{,}1 \wedge 0) \vee (0{,}7 \wedge 0{,}6) \vee (0{,}4 \wedge 1) = 0 \vee 0{,}6 \vee 0{,}4 = 0{,}6;$

$$\mu_{R1 \bullet R2}(x_1, z_3) = 0{,}1;$$

...

$$\mu_{R1 \bullet R2}(x_2, z_5) = 0{,}5.$$

Замечание. В этом примере сначала использован «аналитический» способ композиции отношений R_1 и R_2, т. е. i-я строка R_1 «умножается» на j-й столбец R_2 с использованием операции $\wedge$, затем полученный результат «свертывается» с использованием операции $\vee$ в $\mu(x_i, z_j)$.

Свойства max–min композиции

Операция (max-min)-композиции ассоциативна, т. е.

$$R_3 \bullet (R_2 \bullet R_1) = (R_3 \bullet R_2) \bullet R_1,$$

дистрибутивна относительно объединения, но недистрибутивна относительно пересечения:

$$R_3 \bullet (R_2 \cup R_1) = (R_3 \bullet R_2) \cup (R_3 \bullet R_1),$$

$$R_3 \bullet (R_2 \cap R_1) \neq (R_3 \bullet R_2) \cap (R_3 \bullet R_1).$$

Кроме того, для (max–min)-композиции выполняется следующее важное свойство: если $R_1 \subset R_2$, то $R_3 \bullet R_1 \subset R_3 \bullet R_2$.

(max–*)-композиция

В выражении $\mu_{R1 \bullet R2}(x, z) = \bigvee\limits_y [\mu_{R1}(x, y) \wedge \mu_{R2}(y, z)]$ для (max–min)-композиции отношений R_1 и R_2 операцию $\wedge$ можно заменить любой другой, для которой выполняются те же ограничения, что и для $\wedge$: ассоциативность и монотонность (в смысле неубывания) по каждому аргументу. Тогда

$$\mu_{R1 \bullet R2}(x, z) = \bigvee\limits_y [\mu_{R1}(x, y) * \mu_{R2}(y, z)].$$

В частности, операция $\wedge$ может быть заменена алгебраическим умножением, тогда говорят о (max–prod)-композиции.

3.2. Нечеткий логический вывод

Используемый в различного рода экспертных и управляющих системах механизм нечетких выводов в своей основе имеет базу знаний, формируемую специалистами предметной области в виде совокупности нечетких предикатных правил вида:

Π_1: если x есть A_1, то y есть B_1,

Π_2: если x есть A_2, то y есть B_2,

. . .

Π_n: если x есть A_n, то y есть B_n,

где x – входная переменная (имя для известных значений данных); y – переменная вывода (имя для значения данных, которое будет

вычислено); A и B – функции принадлежности, определенные соответственно на x и y.

Приведем более детальное пояснение. Знание эксперта $A \rightarrow B$ отражает нечеткое причинное отношение предпосылки и заключения, поэтому его можно назвать нечетким отношением и обозначить через R:

$$R = A \rightarrow B,$$

где «$\rightarrow$» называют нечеткой импликацией.

Отношение R можно рассматривать как нечеткое подмножество прямого произведения $X \times Y$ полного множества предпосылок X и заключений Y. Таким образом, процесс получения (нечеткого) результата вывода B' с использованием данного наблюдения A' и знания $A \rightarrow B$ можно представить в виде композиционного правила *нечеткий «modus ponens»*:

$$B' = A' \bullet R = A' \bullet (A \rightarrow B),$$

где «$\bullet$» – введенная выше операция свертки.

Как операцию композиции, так и операцию импликации в алгебре нечетких множеств можно реализовывать по-разному (при этом будет отличаться и получаемый результат), но в любом случае общий логический вывод осуществляется за следующие четыре этапа.

1) Введение нечеткости (*фаззификация*, *fuzzification*). Функции принадлежности, определенные на входных переменных, применяются к их фактическим значениям для определения степени истинности каждой предпосылки каждого правила.

2) Логический вывод. Вычисленное значение истинности для предпосылок каждого правила применяется к заключениям каждого правила. Это приводит к одному нечеткому подмножеству, которое будет назначено каждой переменной вывода для каждого правила. В качестве правил логического вывода обычно используются только операции min (МИНИМУМ) или prod (УМНОЖЕНИЕ). В логическом выводе МИНИМУМА функция принадлежности вывода «отсекается» по высоте, соответствующей вычисленной степени истинности предпосылки правила (нечеткая логика «И»). В логическом выводе УМНОЖЕНИЯ функция принадлежности вывода масштабируется при помощи вычисленной степени истинности предпосылки правила.

3) Композиция. Все нечеткие подмножества, назначенные к каждой переменной вывода (во всех правилах), объединяются вместе, чтобы сформировать одно нечеткое подмножество для всех переменных вывода. При подобном объединении обычно используются операции max (МАКСИМУМ) или sum (СУММА). При

композиции МАКСИМУМА комбинированный вывод нечеткого подмножества конструируется как поточечный максимум по всем нечетким подмножествам (нечеткая логика «ИЛИ»). При композиции СУММЫ комбинированный вывод нечеткого подмножества формируется как поточечная сумма по всем нечетким подмножествам, назначенным переменной вывода правилами логического вывода.

4) Приведение к четкости (*дефаззификация*, *defuzzification*) используется, если требуется преобразовать нечеткий набор выводов в четкое число. Существует большее количество методов приведения к четкости, некоторые из которых рассмотрены ниже.

Пример. Пусть некоторая система описывается следующими нечеткими правилами:

$П_1$: если x есть A, то w есть D,
$П_2$: если y есть B, то w есть E,
$П_3$: если z есть C, то w есть F,

где x, y и z – имена входных переменных, w – имя переменной вывода, а A, B, C, D, E, F – заданные функции принадлежности (треугольной формы).

Процедура получения логического вывода иллюстрируется рис. 3.9. Предполагается, что заданы конкретные (четкие) значения входных переменных: x_0, y_0 и z_0.

На первом этапе на основании данных значений и, исходя из функций принадлежности A, B, C, находятся степени истинности $\alpha(x_0)$, $\alpha(y_0)$ и $\alpha(z_0)$ для предпосылок каждого из трех приведенных правил.

На втором этапе происходит «отсекание» функций принадлежности заключений правил (D, E, F) на уровнях $\alpha(x_0)$, $\alpha(y_0)$, $\alpha(z_0)$.

На третьем этапе рассматриваются функции принадлежности, усеченные на предыдущем этапе, и производится их объединение с использованием операции max, в результате чего получается комбинированное нечеткое подмножество, описываемое функцией принадлежности $\mu_\Sigma(w)$ и соответствующее логическому выводу для выходной переменной w.

Наконец, на четвертом этапе находится, при необходимости, четкое значение выходной переменной, например, с применением центроидного метода: четкое значение выходной переменной определяется как центр тяжести для кривой $\mu_\Sigma(w)$:

$$w_0 = \frac{\int\limits_\Omega w \cdot \mu_\Sigma(w)dw}{\int\limits_\Omega \mu_\Sigma(w)dw}.$$

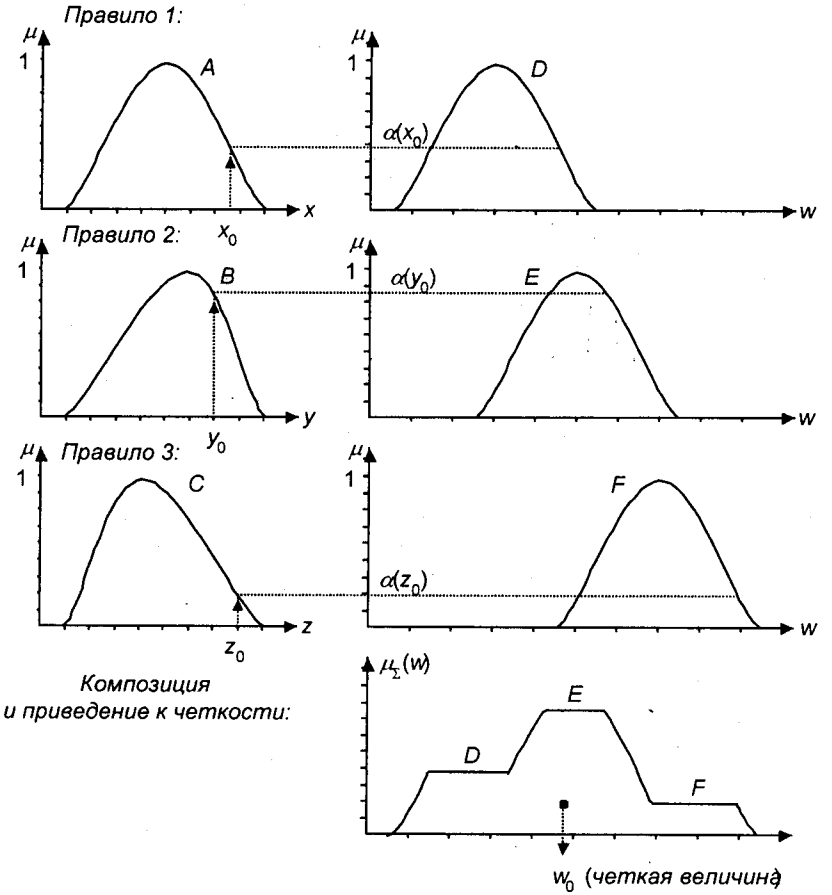


Рис. 3.9. Иллюстрация процедуры логического вывода

Рассмотрим следующие наиболее употребительные модификации алгоритма нечеткого вывода, полагая, для простоты, что базу знаний организуют два нечетких правила вида:

$П_1$: если x есть A_1 и y есть B_1, то z есть C_1,
$П_2$: если x есть A_2 и y есть B_2, то z есть C_2,

где x и y – имена входных переменных, z – имя переменной вывода, A_1, A_2, B_1, B_2, C_1, C_2 – некоторые заданные функции принадлежности.

При этом четкое значение z_0 необходимо определить на основе приведенной информации и четких значений x_0 и y_0.

Алгоритм Mamdani

Данный алгоритм соответствует рассмотренному примеру и рис. 3.9. В рассматриваемой ситуации он математически может быть описан следующим образом:

1) Введение нечеткости. Находятся степени истинности для предпосылок каждого правила: $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$.

2) Логический вывод. Находятся уровни «отсечения» для предпосылок каждого из правил (с использованием операции МИНИМУМ):

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0),$$
$$\alpha_2 = A_2(x_0) \wedge B_2(y_0),$$

где через «$\wedge$» обозначена операция логического минимума (min).

Затем находятся «усеченные» функции принадлежности:

$$C'_1(z) = (\alpha_1 \wedge C_1(z)),$$
$$C'_2(z) = (\alpha_2 \wedge C_2(z)).$$

3) Композиция. Производится объединение найденных усеченных функций с использованием операции МАКСИМУМ (max, далее обозначаемой как «$\vee$»), что приводит к получению итогового нечеткого подмножества для переменной выхода с функцией принадлежности:

$$\mu_\Sigma(z) = C(z) = C'_1(z) \vee C'_2(z) = (\alpha_1 \wedge C_1(z)) \vee (\alpha_2 \wedge C_2(z)).$$

4) Приведение к четкости. Проводится для нахождения z_0, например, центроидным методом.

Алгоритм Tsukamoto

Исходные посылки – как у предыдущего алгоритма, но здесь предполагается, что функции $C_1(z)$, $C_2(z)$ являются монотонными.

1) Введение нечеткости (как в алгоритме Mamdani).

2) Нечеткий вывод. Сначала находятся уровни «отсечения» α_1 и α_2 (как в алгоритме Mamdani), а затем решением уравнений:

$$\alpha_1 = C_1(z_1), \quad \alpha_2 = C_2(z_2)$$

определяются четкие значения (z_1 и z_2) для каждого исходного правила.

3) Определяется четкое значение переменной вывода (как взвешенное среднее z_1 и z_2):

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2};$$

в общем случае (дискретный вариант центроидного метода):

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i z_i}{\sum_{i=1}^{n} \alpha_i}.$$

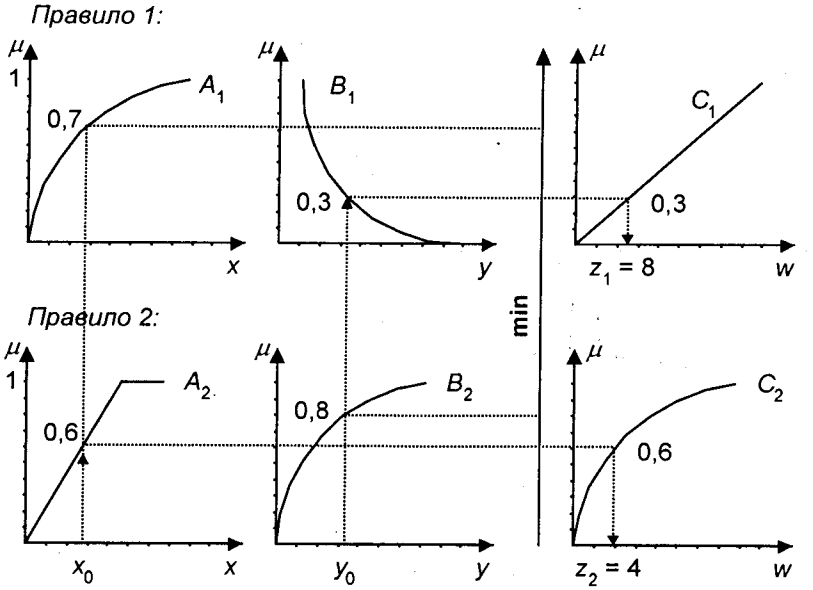


Рис. 3.10. Иллюстрация к алгоритму Tsukamoto

Пример. Пусть заданы $A_1(x_0) = 0{,}7$, $A_2(x_0) = 0{,}6$, $B_1(y_0) = 0{,}3$, $B_2(y_0) = 0{,}8$, соответствующие уровни отсечения:

$$\alpha_1 = \min\{A_1(x_0), B_1(y_0)\} = \min\{0{,}7, 0{,}3\} = 0{,}3,$$
$$\alpha_2 = \min\{A_2(x_0), B_2(y_0)\} = \min\{0{,}6, 0{,}8\} = 0{,}6.$$

и значения $z_1 = 8$ и $z_2 = 4$, найденные в результате решения уравнений (рис. 3.10):

$$C_1(z_1) = 0{,}3, \quad C_2(z_2) = 0{,}6$$

При этом четкое значение переменной вывода:

$$z_0 = (8 \cdot 0{,}3 + 4 \cdot 0{,}6)/(0{,}3 + 0{,}6) = 6.$$

Алгоритм Sugeno

Sugeno и Takagi использовали набор правил в следующей форме (как и ранее, приведем пример двух правил):

$П_1$: если x есть A_1 и y есть B_1, то $z_1 = a_1x + b_1y$,

$П_2$: если x есть A_2 и y есть B_2, то $z_2 = a_2x + b_2y$.

Представление алгоритма (рис. 3.11).

1) Введение нечеткости (как в алгоритме Mamdani).

2) Нечеткий вывод. Находятся $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$ и индивидуальные выходы правил:

$$z^*_1 = a_1x_0 + b_1y_0,$$
$$z^*_2 = a_2x_0 + b_2y_0.$$

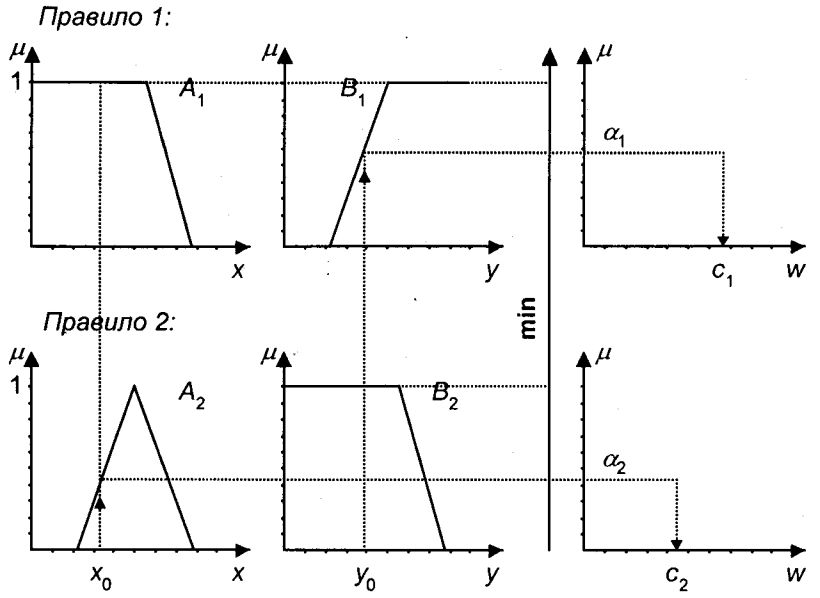


Рис. 3.13. Иллюстрация к упрощенному алгоритму нечеткого логического вывода

Методы приведения к четкости

1) Выше уже был рассмотрен один из данных методов – *центроидный*. Приведем соответствующие формулы еще раз.

В общем случае:

$$z_0 = \frac{\int_{\Omega} z \cdot C(z)dz}{\int_{\Omega} C(z)dz};$$

для дискретного варианта:

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i z_i}{\sum_{i=1}^{n} \alpha_i}.$$

2) *Первый максимум* (*First-of-Maxima*). Четкая величина вывода находится как наименьшее значение, при котором достигается максимум итогового нечеткого множества (рис. 3.14, *а*):

$$z_0 = \min\{z \mid C(z) = \max_u C(u)\}.$$

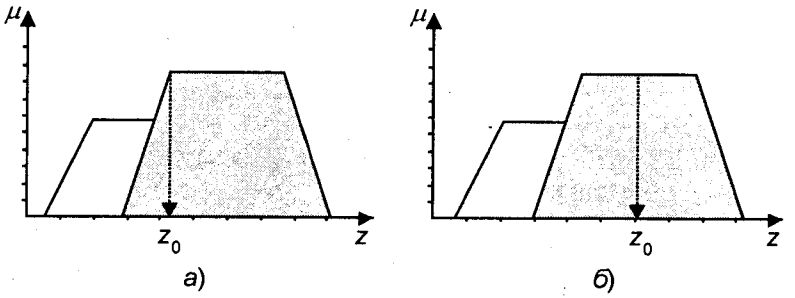


Рис. 3.14. Иллюстрация к методам приведения к четкости:

а – первый максимум; *б* – средний максимум

3) *Средний максимум* (*Middle-of-Maxima*). Четкое значение находится по формуле:

$$z_0 = \frac{\int_G zdz}{\int_G dz},$$

где G – подмножество элементов, максимизирующих C (рис. 3.14, *б*).

Для дискретного варианта (C дискретно):

$$z_0 = \frac{1}{n}\sum_{j=1}^{n} z_j.$$

4) *Критерий максимума* (*Max-Criterion*). Четкое значение выбирается произвольно среди множества элементов, для которых C достигает максимума:

$$z_0 \in \{z \mid C(z) = \max_u C(u)\}.$$

5) *Высотная дефаззификация* (*Height defuzzification*). Элементы области определения Ω, для которых значения функции принадлежности меньше, чем некоторый уровень α в расчет не принимаются, и четкое значение рассчитывается в соответствии с выражением:

$$z_0 = \frac{\int_{C\alpha} z \cdot C(z)dz}{\int_{C\alpha} C(z)dz},$$

где C_α – нечеткое множество α-уровня (см. выше).

Нисходящие нечеткие логические выводы

Ранее рассмотренные нечеткие логические выводы являются восходящими выводами от предпосылок к заключениям. В диагностических нечетких системах часто применяются нисходящие выводы. Рассмотрим механизм подобного вывода.

Пусть задано полное пространство предпосылок $X = \{x_1, \ldots, x_m\}$ и полное пространство заключений $Y = \{y_1, \ldots, y_n\}$.

Между x_i и y_j существуют нечеткие причинные отношения $x_i \to y_j$, которые можно представить в виде некоторой матрицы R с элементами $r_{ij} \in [0, 1]$, $i = 1 \ldots m$, $j = 1 \ldots n$. Предпосылки и заключения можно рассматривать как нечеткие множества A и B на пространствах X и Y, отношения которых можно представить в виде:

$$B = A \bullet \mathrm{R},$$

где «•», как и раньше, обозначает правило композиции нечетких выводов, например max–min-композицию.

В данном случае направление выводов является обратным для правил, т. е. задана матрица R (знания эксперта), наблюдаются выходы B (заключения) и определяются входы A (предпосылки).

Пусть задана модель диагностики системы, состоящая из двух предпосылок и трех заключений: $X = \{x_1, x_2\}$, $Y = \{y_1, y_2, y_3\}$, а матрица нечетких отношений имеет вид:

$$\mathrm{R} = \begin{bmatrix} 0{,}8 & 0{,}2 & 0{,}3 \\ 0{,}7 & 0{,}4 & 0{,}5 \end{bmatrix}.$$

Допустим в результате диагностики системы были получены следующие заключения:

$$B = 0{,}8/y_1 + 0{,}2/y_2 + 0{,}3/y_3.$$

Необходимо найти приведшие к этому предпосылки:

$$A = a_1/x_1 + a_2/x_2.$$

С учетом конкретных данных отношения между предпосылками и заключениями будут представлены следующим образом:

$$\begin{bmatrix} 0{,}8 & 0{,}2 & 0{,}3 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \bullet \begin{bmatrix} 0{,}8 & 0{,}2 & 0{,}3 \\ 0{,}7 & 0{,}4 & 0{,}5 \end{bmatrix},$$

либо в транспонированном виде:

$$\begin{bmatrix} 0{,}8 \\ 0{,}2 \\ 0{,}3 \end{bmatrix} = \begin{bmatrix} 0{,}8 & 0{,}7 \\ 0{,}2 & 0{,}4 \\ 0{,}3 & 0{,}5 \end{bmatrix} \bullet \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}.$$

При использовании max–min-композиции последнее соотношение преобразуется к виду:

$$0{,}8 = (0{,}8 \wedge a_1) \vee (0{,}7 \wedge a_2),$$
$$0{,}2 = (0{,}2 \wedge a_1) \vee (0{,}4 \wedge a_2),$$

$$0{,}3 = (0{,}3 \wedge a_1) \vee (0{,}5 \wedge a_2),$$

При решении данной системы заметим, что в первом уравнении второй член правой части не влияет на левую часть, поэтому:

$$0{,}8 = 0{,}8 \wedge a_1, \quad a_1 \geq 0{,}8.$$

Из второго уравнения получим:

$$0{,}2 \geq 0{,}4 \wedge a_2, \quad a_2 \leq 0{,}2.$$

Полученное решение удовлетворяет третьему уравнению. Таким образом:

$$0{,}8 \leq a_1 \leq 1, \quad 0 \leq a_2 \leq 0{,}2$$

При решении практических задач могут одновременно использоваться различные правила композиции нечетких выводов, сама схема выводов может быть многокаскадной. В настоящее время общих методов решения подобных задач, по-видимому, не существует.

3.3. Эффективность нечетких систем принятия решений

Возможность использования аппарата нечеткой логики базируется на следующих результатах.

1) В 1992 г. Wang показал, что нечеткая система является универсальным аппроксиматором, т. е. может аппроксимировать любую непрерывную функцию на компакте U с произвольной точностью, если использует набор n ($n \to \infty$) правил:

Π_i: если x_i есть A_i и y_i есть B_i, тогда z_i есть C_i, $i = 1, \ldots, n$,

при следующих условиях:

- гауссовых функциях принадлежности:

$$A_i(x) = \exp\left[-\frac{1}{2}\left(\frac{x - \alpha_{i1}}{\beta_{i1}}\right)^2\right], \quad B_i(y) = \exp\left[-\frac{1}{2}\left(\frac{y - \alpha_{i2}}{\beta_{i2}}\right)^2\right],$$

$$C_i(z) = \exp\left[-\frac{1}{2}\left(\frac{z - \alpha_{i3}}{\beta_{i3}}\right)^2\right],$$

- композиции в виде произведения:

$$[A_i(x) \text{ and } B_i(y)] = A_i(x)\, B_i(y),$$

- импликации в форме (Larsen):

$$[A_i(x) \text{ and } B_i(y)] \to C_i(z) = A_i(x)\, B_i(y)\, C_i(z),$$

- центроидном методе приведения к четкости:

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_{i3} A_i B_i}{\sum_{i=1}^{n} A_i B_i},$$

где α_{i3} – центры C_i.

Иначе говоря, Wang доказал теорему: для каждой вещественной непрерывной функции g, заданной на компакте U и для произвольного $\varepsilon > 0$ существует нечеткая система, формирующая выходную функцию $f(x)$ такую, что

$$\sup_{x \in U} \|g(x) - f(x)\| \le \varepsilon,$$

где $\|\bullet\|$ – символ принятого расстояния между функциями.

2) В 1995 г. Castro показал, что логический контроллер Mamdani также является универсальным аппроксиматором при:

- симметричных треугольных функциях принадлежности:

$$A_i(x) = \begin{cases} 1 - |a_i - x| / \alpha_i, & \text{если } |a_i - x| \le \alpha_i, \\ 0, & \text{если } |a_i - x| > \alpha_i, \end{cases}$$

$$B_i(y) = \begin{cases} 1 - |b_i - y| / \beta_i, & \text{если } |b_i - y| \le \beta_i, \\ 0, & \text{если } |b_i - y| > \beta_i, \end{cases}$$

$$C_i(z) = \begin{cases} 1 - |c_i - z| / \gamma_i, & \text{если } |c_i - z| \le \gamma_i, \\ 0, & \text{если } |c_i - z| \le \gamma_i, \end{cases}$$

- композиции с использованием операции min:

$$[A_i(x) \text{ and } B_i(y)] = \min\{A_i(x), B_i(y)\},$$

- импликации в форме Mamdani и центроидного метода приведения к четкости:

$$z_0 = \frac{\sum_{i=1}^{n} c_i \min\{A_i(x), B_i(y)\}}{\sum_{i=1}^{n} \min\{A_i(x), B_i(y)\}},$$

где c_i – центры C_i.

Вообще говоря, системы с нечеткой логикой *целесообразно* применять в следующих случаях:

- для сложных процессов, когда нет простой математической модели;
- если экспертные знания об объекте или о процессе можно сформулировать только в лингвистической форме.

Системы, базирующиеся на нечеткой логике, применять *нецелесообразно*:

- если требуемый результат может быть получен каким-либо другим (стандартным) путем;
- когда для объекта или процесса уже найдена адекватная и легко исследуемая математическая модель.

Отметим, что *основными недостатками* систем с нечеткой логикой являются то, что:

- исходный набор постулируемых нечетких правил формулируется экспертом-человеком и может оказаться неполным или противоречивым;
- вид и параметры функций принадлежности, описывающих входные и выходные переменные системы, выбираются субъективно и могут оказаться не вполне отражающими реальную действительность.

3.4. Синтез нечетких нейронных сетей

Различные типы интеллектуальных систем имеет свои особенности, например, по возможностям обучения, обобщения и выработки результатов, что делает их наиболее пригодными для решения одних классов задач и менее пригодными – для других.

Например, нейронные сети хороши для задач распознавания образов, но весьма неудобны для объяснения, *как они такое распознавание осуществляют*. Они могут автоматически приобретать знания, но процесс их обучения зачастую происходит достаточно медленно, а анализ обученной сети весьма сложен (обученная сеть представляет обычно черный ящик для пользователя). При этом какую-либо априорную информацию (знания эксперта) для ускорения процесса ее обучения в нейронную сеть ввести невозможно.

Системы с нечеткой логикой, напротив, хороши для объяснения получаемых с их помощью выводов, но *они не могут автоматически приобретать знания* для использования их в механизмах выводов. Необходимость разбиения универсальных множеств на отдельные области, как правило, ограничивает количество входных переменных в таких системах небольшим значением.

Вообще говоря, теоретически, системы с нечеткой логикой и искусственные нейронные сети подобны друг другу, однако, в соответствии с изложенным выше, на практике у них имеются свои собственные достоинства и недостатки. Данное соображение легло в основу создания аппарата нечетких нейронных сетей, в которых выводы делаются на основе аппарата нечеткой логики, но со-

ответствующие функции принадлежности подстраиваются с использованием алгоритмов обучения нейронных сетей, например, алгоритма обратного распространения ошибки. Такие системы не только используют априорную информацию, но могут приобретать новые знания, являясь логически прозрачными.

3.4.1. Основные понятия и определения нечетких нейронных сетей

Для пояснения сущности нечетких нейронных сетей, рассмотрим простую нейронную сеть, состоящую из одного нейрона с двумя входами. Входные сигналы x_i «взаимодействуют» с синаптическими весами w_i:

$$p_i = w_i x_i, \quad i = 1, 2.$$

Эти частные произведения суммируются, образуя значение *net* нейрона:

$$net = p_1 + p_2 = w_1 x_1 + w_2 x_2.$$

Выход нейрона образуется в результате преобразования значения *net* некоторой активационной функцией *f*:

$$y = f(net) = f(w_1 x_1 + w_2 x_2),$$

Рассмотренная однонейронная сеть, в которой используются операции умножения, суммирования и сигмоидная функция активации является стандартной нейронной сетью.

В случае применения вместо операций умножения, суммирования и активации таких операций, как *t-норма* или *t-конорма* (см. разд. 3.1.2) данную нейронную сеть будем называть *нечеткой*.

Нечеткая нейронная сеть – это нейронная сеть с четкими сигналами, весами и активационной функцией, но с объединением x_i и w_i, p_1 и p_2 с использованием операций *t-нормы, t-конормы* или некоторых других непрерывных операций. Входы, выходы и веса нечеткой нейронной сети – вещественные числа, принадлежащие отрезку [0, 1].

Рассмотрим примеры элементарных нечетких нейронных сетей.

Нечеткий нейрон «И». Сигналы x_i и веса w_i в данном случае объединяются с помощью *t-конормы*:

$$p_i = S(w_i, x_i), \quad i = 1, 2,$$

а выход образуется с применением *t-нормы* (рис. 3.15, *а*):

$$y = AND(p_1, p_2) = T(p_1, p_2) = T(S(w_1, x_1), S(w_2, x_2)).$$

Если принять T = min, S = max, то нечеткий нейрон «И» реализует композицию min–max:

$$y = \min\{w_1 \vee x_1, w_2 \vee x_2\}.$$

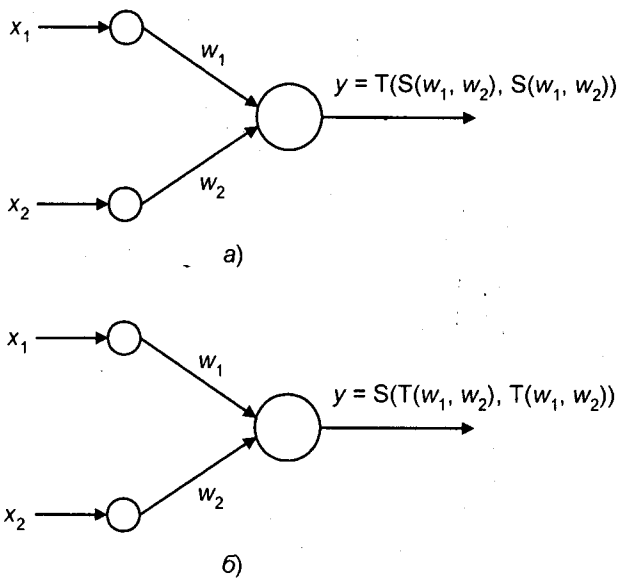


Рис. 3.15. Структуры нечетких нейронов:

а – «И»; *б* – «ИЛИ»

Нечеткий нейрон «ИЛИ». Сигналы x_i и веса w_i здесь объединяются с помощью *t-нормы*:

$$p_i = T(w_i, x_i), \quad i = 1, 2,$$

а выход образуется с применением *t-конормы* (рис. 3.15, *б*):

$$y = OR(p_1, p_2) = S(p_1, p_2) = S(T(w_1, x_1), T(w_2, x_2)).$$

Если принять T = min, S = max, то нечеткий нейрон «ИЛИ» реализует композицию max–min:

$$y = \max\{w_1 \wedge x_1, w_2 \wedge x_2\}.$$

3.4.2. Алгоритмы обучения и использования нечетких нейронных сетей

Опишем типовой подход к построению алгоритмов обучения и использования нечетких нейронных сетей.

Предположим, что нечеткой нейронной сетью должно быть реализовано (неизвестное) отображение:

$$y^k = f(\mathbf{x}^k) = f(x_1^k, x_2^k, \ldots, x_n^k), \quad k = 1, \ldots, N,$$

при наличии обучающего множества:

$$\{(\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^N, y^N)\}.$$

Для моделирования неизвестного отображения f используем упрощенный алгоритм нечеткого вывода (см. разд. 3.2), применяя следующую форму записи предикатных правил:

Π_i: если x_1 есть A_{i1} и x_2 есть A_{i2} и ... и x_n есть A_{in}, то $y = z_i$,

$$i = 1, \ldots, m,$$

где A_{ij} – нечеткие числа треугольной формы, z_i – вещественные числа.

Степень истинности i-го правила определяется с помощью операции умножения (Larsen):

$$\alpha_i = \prod_{j=1}^{n} A_{ij}(x_j^k)$$

(можно использовать и другие представления для моделирования логического оператора «И»).

Выход нечеткой системы определяется в соответствии с центроидным методом (дискретный вариант):

$$o^k = \frac{\sum_{i=1}^{m} \alpha_i z_i}{\sum_{i=1}^{m} \alpha_i}.$$

Введение функции ошибки для k-го предъявленного образца вида:

$$E_k = \frac{1}{2}\left(o^k - y^k\right)^2$$

позволяет, далее, как в обычных нейронных сетях использовать градиентный метод для подстройки параметров заданных предикатных правил. Так, величины z_i можно корректировать по соотношению:

$$z_i := z_i - \eta \frac{\partial E_k}{\partial z_i} = z_i - \eta(o^k - y^k)\frac{\alpha_i}{\alpha_1 + \alpha_2 + \ldots + \alpha_m}, \quad i = 1, \ldots, m,$$

где η – константа, характеризующая скорость обучения.

Более детально алгоритм настройки рассмотрим на примере системы, включающей два правила:

Π_1: если x есть A_1, то $y = z_1$,

Π_2: если x есть A_2, то $y = z_2$,

при этом предполагается, что нечеткие понятия A_1 («малый») и A_2 («большой») имеют сигмоидные функции принадлежности:

$$A_1(x) = \frac{1}{1 + \exp(b_1(x - a_1))}, \quad A_2(x) = \frac{1}{1 + \exp(b_2(x - a_2))},$$

характеризующиеся параметрами a_1, a_2, b_1, b_2.

Степени истинности правил определяются в данном случае соотношениями:

$$\alpha_1 = A_1(x) = \frac{1}{1 + \exp(b_1(x - a_1))},$$

$$\alpha_2 = A_2(x) = \frac{1}{1 + \exp(b_2(x - a_2))},$$

а выход системы – выражением:

$$o = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} = \frac{A_1(x) z_1 + A_2(x) z_2}{A_1(x) + A_2(x)}.$$

Предположим, что имеется обучающее множество $\{(x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)\}$, отображающее неизвестную функцию f.

Требуется: осуществить такую настройку параметров системы a_1, a_2, b_1, b_2, z_1, z_2, при которой обеспечивается наилучшая аппроксимация данной функции.

Решение. Для данного случая функция ошибки может быть записана в форме:

$$E_k = E_k(a_1, b_1, a_2, b_2, z_1, z_2) = \frac{1}{2}\left(o^k(a_1, b_1, a_2, b_2, z_1, z_2) - y^k\right)^2.$$

Используя далее тот же подход, как и при выводе алгоритма обратного распространения ошибки (см. разд. 1.4), запишем:

$$z_1 := z_1 - \eta \frac{\partial E_k}{\partial z_1} = z_1 - \eta(o^k - y^k)\frac{\alpha_1}{\alpha_1 + \alpha_2} =$$

$$= z_1 - \eta(o^k - y^k)\frac{A_1(x^k)}{A_1(x^k) + A_2(x^k)},$$

$$z_2 := z_2 - \eta \frac{\partial E_k}{\partial z_2} = z_2 - \eta(o^k - y^k)\frac{\alpha_2}{\alpha_1 + \alpha_2} =$$

$$= z_2 - \eta(o^k - y^k)\frac{A_2(x^k)}{A_1(x^k) + A_2(x^k)}.$$

Аналогичным образом могут быть получены выражения для коррекции коэффициентов a_1, a_2, b_1, b_2. Исходные соотношения таковы:

$$a_1 := a_1 - \eta \frac{\partial E_k}{\partial a_1}, \quad a_2 := a_2 - \eta \frac{\partial E_k}{\partial a_2},$$

$$b_1 := b_1 - \eta \frac{\partial E_k}{\partial b_1}, \quad b_2 := b_2 - \eta \frac{\partial E_k}{\partial b_2}.$$

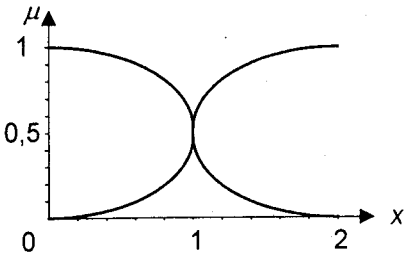


Рис. 3.16. Симметричные функции принадлежности

Конечные выражения являются достаточно громоздкими, но могут быть упрощены в случае, если функции принадлежности имеют вид:

$$A_1(x) = \frac{1}{1+\exp(-b(x-a))}, \quad A_2(x) = \frac{1}{1+\exp(b(x-a))}.$$

Данные функции характеризуются всего двумя параметрами (a и b), в определенном смысле являются симметричными (рис. 3.16) и удовлетворяют уравнению:

$$A_1(x) + A_2(x) = 1.$$

Заметим, что из последнего и ранее полученных уравнений следует:

$$z_1 := z_1 - \eta\frac{\partial E_k}{\partial z_1} = z_1 - \eta(o^k - y^k)\frac{\alpha_1}{\alpha_1+\alpha_2} = z_1 - \eta(o^k - y^k)A_1(x^k),$$

$$z_2 := z_2 - \eta\frac{\partial E_k}{\partial z_2} = z_2 - \eta(o^k - y^k)\frac{\alpha_2}{\alpha_1+\alpha_2} = z_2 - \eta(o^k - y^k)A_2(x^k).$$

Последующие выкладки таковы:

$$a := a - \eta\frac{\partial E_k(a,b)}{\partial a},$$

где

$$\frac{\partial E_k(a,b)}{\partial a} = (o^k - y^k)\frac{\partial o^k}{\partial a} = (o^k - y^k)\frac{\partial}{\partial a}\left[z_1 A_1(x^k) + z_2 A_2(x^k)\right] =$$

$$= (o^k - y^k)\frac{\partial}{\partial a}\left[z_1 A_1(x^k) + z_2(1 - A_1(x^k))\right] =$$

$$= (o^k - y^k)(z_1 - z_2)\frac{\partial A_1(x^k)}{\partial a} =$$

$$= (o^k - y^k)(z_1 - z_2)b\frac{\exp(b(x^k - a))}{\left[1+\exp(b(x^k - a))\right]^2} =$$

$$= (o^k - y^k)(z_1 - z_2)bA_1(x^k)(1 - A_1(x^k)) =$$

$$= (o^k - y^k)(z_1 - z_2)bA_1(x^k)A_1(x^k)$$

и

$$b := b - \eta\frac{\partial E_k(a,b)}{\partial b},$$

где

$$\frac{\partial E_k(a,b)}{\partial b} = (o^k - y^k)(z_1 - z_2)\frac{\partial}{\partial b}\left[\frac{1}{1+\exp(b(x^k - a))}\right] =$$

$$= (o^k - y^k)(z_1 - z_2)(x^k - a)A_1(x^k)(1 - A_1(x^k)) =$$

$$= (o^k - y^k)(z_1 - z_2)(x^k - a)A_1(x^k)A_1(x^k).$$

Приведенные выкладки, как представляется, полностью иллюстрируют идеи алгоритмов обучения и использования нечеткой нейронной сети.

Рассмотрим другой пример нечеткой системы, имеющей следующую базу знаний:

Π_1: если x_1 есть L_1 и x_2 есть L_2 и x_3 есть L_3, то y есть G,

Π_2: если x_1 есть H_1 и x_2 есть H_2 и x_3 есть L_3, то y есть P,

Π_3: если x_1 есть H_1 и x_2 есть H_2 и x_3 есть H_3, то y есть R,

где x_1, x_2, x_3 – входные переменные, y – выход системы, L_1, L_2, L_3, H_1, H_2, H_3, G, P, R – некоторые нечеткие множества с функциями принадлежности сигмоидного типа:

$$L_j(x_i) = \frac{1}{1+\exp(b_j(x_i - c_j))}, \quad H_j(x_i) = \frac{1}{1+\exp(-b_j(x_i - c_j))},$$

$$j = 1, 2, 3,$$

$$G(z_1) = \frac{1}{1+\exp(-b_4(z_1 - c_4 + c_5))}, \quad P(z_2) = \frac{1}{1+\exp(-b_4(z_2 - c_4))},$$

$$R(z_3) = \frac{1}{1+\exp(b_4(z_3 - c_4))}.$$

Для определения выходной переменной используется алгоритм вывода Tsukamoto (см. разд. 3.2), в соответствии с которым:

- подсчитываются значения истинности предпосылок для каждого правила:

$$\alpha_1 = L_1(x_1) \wedge L_2(x_2) \wedge L_3(x_3),$$

$$\alpha_2 = H_1(x_1) \wedge H_2(x_2) \wedge L_3(x_3),$$

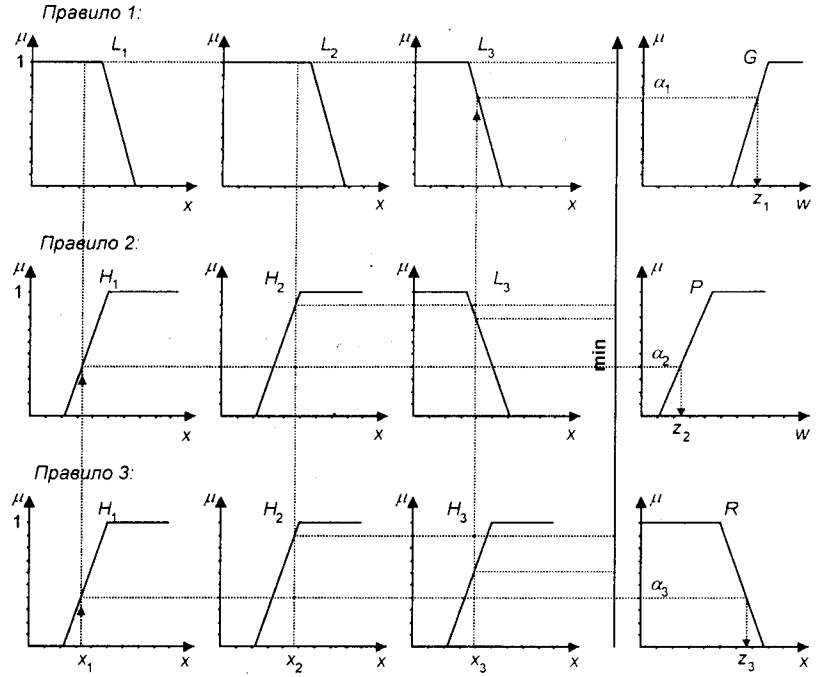


Рис. 3.17. Иллюстрация алгоритма вывода Tsukamoto в нечеткой нейронной сети

$$\alpha_3 = H_1(x_1) \wedge H_2(x_2) \wedge H_3(x_3),$$

где x_1, x_2, x_3 – текущие значения входов системы;

- для каждого правила определяются частные выходы:

$$z_1 = G^{-1}(\alpha_1) = c_4 + c_5 + \frac{1}{b_4} \ln \frac{1-\alpha_1}{\alpha_1},$$

$$z_2 = P^{-1}(\alpha_2) = c_4 + \frac{1}{b_4} \ln \frac{1-\alpha_2}{\alpha_2},$$

$$z_3 = R^{-1}(\alpha_3) = c_4 + \frac{1}{b_4} \ln \frac{1-\alpha_3}{\alpha_3};$$

- находится общий выход системы:

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3}{\alpha_1 + \alpha_2 + \alpha_3}.$$

Изложенный процесс иллюстрируется рис. 3.17.

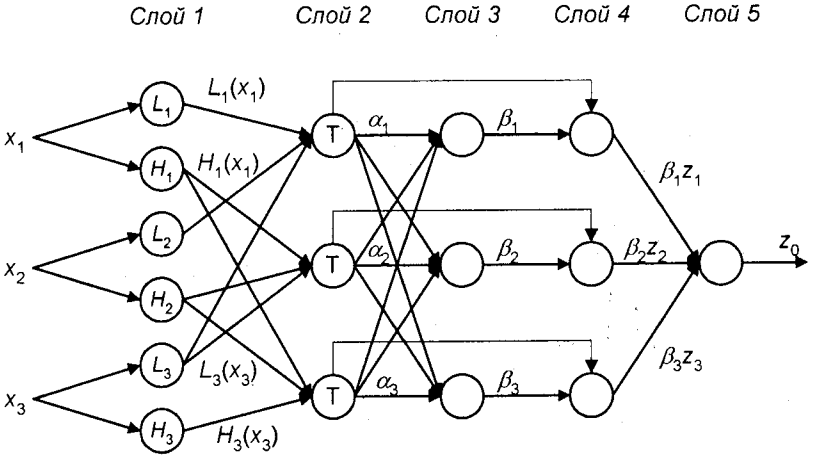


Рис. 3.18. Структура нечеткой нейронной сети (архитектура ANFIS)

Нечеткая нейронная сеть, реализующая приведенный механизм вывода, представлена на рис. 3.18. Заметим, что сети с подобной архитектурой в англоязычной литературе получили название ANFIS (Adaptive Neuro-Fuzzy Inference System).

Данная сеть может быть описана следующим образом.

- *Слой 1.* Выходы нейронов этого слоя представляют собой значения функций принадлежности при конкретных (заданных) значениях входов.
- *Слой 2.* Выходами нейронов этого слоя являются степени истинности предпосылок каждого правила базы знаний системы, вычисляемые по формулам:

$$\alpha_1 = L_1(x_1) \wedge L_2(x_2) \wedge L_3(x_3),$$

$$\alpha_2 = H_1(x_1) \wedge H_2(x_2) \wedge L_3(x_3),$$

$$\alpha_3 = H_1(x_1) \wedge H_2(x_2) \wedge H_3(x_3).$$

Все нейроны слоя обозначены буквой T, что означает, что они могут реализовывать произвольную *t*-норму для моделирования операции «И».

- *Слой 3.* Нейроны этого слоя вычисляют величины:

$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3}, \quad \beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2 + \alpha_3}, \quad \beta_3 = \frac{\alpha_3}{\alpha_1 + \alpha_2 + \alpha_3}.$$

- *Слой 4.* Нейроны данного слоя выполняют операции:

$$\beta_1 z_1 = \beta_1 G^{-1}(\alpha_1), \quad \beta_2 z_2 = \beta_2 P^{-1}(\alpha_2), \quad \beta_3 z_3 = \beta_3 R^{-1}(\alpha_3).$$

• *Слой 5.* Единственный нейрон этого слоя вычисляет выход сети:

$$z_0 = \beta_1 z_1 + \beta_2 z_2 + \beta_3 z_3.$$

Корректировка параметров системы для функций принадлежности G, P и R производится в соответствии с ранее рассмотренным подходом. Так, например, настройка коэффициентов для функций принадлежности b_4, c_4 и c_5 осуществляется по формулам:

$$b_4 := b_4 - \eta \frac{\partial E_k}{\partial b_4} = b_4 - \frac{\eta}{b_4^2} \delta_k \frac{\alpha_1 + \alpha_2 - \alpha_3}{\alpha_1 + \alpha_2 + \alpha_3},$$

$$c_4 := c_4 - \eta \frac{\partial E_k}{\partial c_4} = c_4 + \eta \delta_k \frac{\alpha_1 + \alpha_2 + \alpha_3}{\alpha_1 + \alpha_2 + \alpha_3} = c_4 + \eta \delta_k,$$

$$c_5 := c_5 - \eta \frac{\partial E_k}{\partial c_5} = c_5 + \eta \delta_k \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3},$$

где $\delta_k = y^k - o^k$.

Соответствующие выражения могут быть получены и для остальных коэффициентов: b_i и c_i, $i = 1, 2, 3$.

Генерация нечетких правил

Можно выделить два подхода к модификации топологии нечеткой нейронной сети на этапах обучения и использования. Первый, традиционный подход основан на введении дополнительных продукционных правил в базу знаний системы. При этом следует учитывать непротиворечивость ее пополнения.

Другой подход предполагает генерацию новых продукционных правил, не противоречащих правилам из базы знаний системы, исходя из анализа экспериментальных данных об объекте на основе предложенной одним из авторов книги процедуры, которая рассмотрена ниже.

Предположим, что исследуемый объект имеет n входов (иначе, векторный вход $\boldsymbol{x}$) и один выход y и имеет «истинное» (неизвестное) описание:

$$y = f(\boldsymbol{x}) + e,$$

где $f(\boldsymbol{x})$ – функция неизвестного вида, e – случайная аддитивная помеха (отражающая действие не учитываемых факторов) с нулевым средним значением и произвольным (неизвестным) распределением на $(-\varepsilon_m, \varepsilon_m)$.

Предположим далее, что на объекте может быть реализован эксперимент, заключающийся в регистрации N пар значений

$\langle \boldsymbol{x}_i, y_i \rangle$, $i = 1, \ldots, N$, при этом величины (векторы) $\boldsymbol{x}_i$ измеряются без ошибок; значение N при необходимости допускает модификацию.

Алгоритм построения системы может быть теперь описан следующим образом.

ШАГ 1. Из m ($m < N$) произвольных значений $\langle \mathbf{x}_i, y_i \rangle$ составляется начальная база знаний модели, отображаемая матрицей $\boldsymbol{U}_{m\times(n+1)}$ со строками вида $\langle \boldsymbol{x}_i^{\mathrm{T}}, y_i \rangle = \langle x_{i1}, x_{i2}, \ldots, x_{in}, y_i \rangle$.

Такое представление, очевидно, эквивалентно набору продукционных правил вида:

$$\Pi_i: \text{ если } x_1 \text{ есть } A_{i1} \text{ и } x_2 \text{ есть } A_{i2} \text{ и } \ldots \text{ и } x_n \text{ есть } A_{in}, \text{ то } y = y_i, \quad i = 1, \ldots, m,$$

ШАГ 2. Для каждой новой экспериментальной точки $\langle \boldsymbol{x}, y \rangle$ рассчитывается прогнозируемое значение по формуле, соответствующей рассмотренному центроидному методу:

$$\hat{y} = \frac{\sum_{i=1}^{m} y_i \varphi(\|\mathbf{x} - \mathbf{x}_i\|)}{\sum_{i=1}^{m} \varphi(\|\mathbf{x} - \mathbf{x}_i\|)},$$

где $\varphi(\bullet)$ – функция колоколообразной или экспоненциальной формы:

$$\varphi(\|\mathbf{x} - \mathbf{x}_i\|) = \exp(-\lambda \sum_{j=1}^{n} |x_j - x_{ij}|),$$

λ – параметр функций.

ШАГ 3. Проверяется неравенство:

$$\left| y - \hat{y} \right| > d,$$

где d – заданная константа, определяющая погрешность аппроксимации.

При выполнении неравенства база знаний системы пополняется путем расширения матрицы $\boldsymbol{U}$ (добавлением строки $\langle \boldsymbol{x}^{\mathrm{T}}, y \rangle$). В противном случае матрица $\boldsymbol{U}$ остается без изменений.

ШАГ 4. Проверяется правило останова. В данном варианте алгоритма построение модели считается законченным, если в соответствии с шагами 2 и 3 перебраны все N экспериментальных точек (без учета значений начальной базы знаний).

Если не все экспериментальные точки использованы, то осуществляется переход к шагу 2, в противном случае – останов.

В процессе реализации алгоритма параметры λ и d считаются априори заданными.

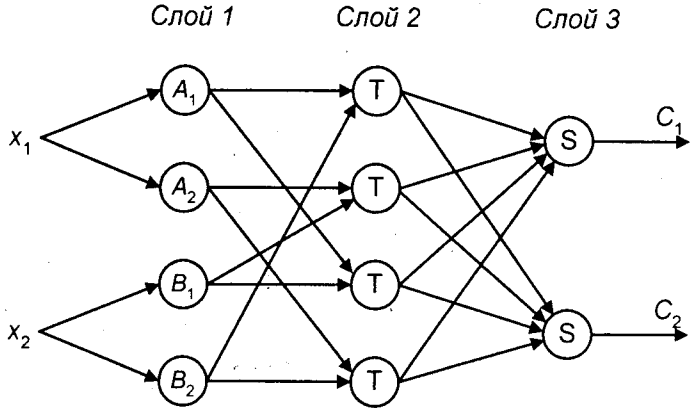


Рис. 3.19. Нечеткая нейронная сеть для решения задач классификации

При использовании системы заданными считаются матрица U (на этапе использования модели она не изменяется), отмеченные параметры λ и d, и расчет $\hat{y}$ производится в соответствии с шагом 2 приведенного алгоритма.

Нетрудно видеть, что описанный алгоритм, в сущности, соответствует упрощенному алгоритму нечеткого логического вывода (см. разд. 3.2), но отличается от последнего тем, что база знаний не остается фиксированной, а модернизируется по мере поступления экспериментальных данных. Причем непротиворечивость нового продукционного правила относительно набора правил из базы знаний гарантируется предложенной процедурой ее пополнения.

3.5. Нечеткий классификатор

Рассмотрим, как с помощью нечеткой нейронной сети может быть решена задача классификации. Допустим, что объекты характеризуется двумя признаками x_1 и x_2 и относится к одному из двух классов – C_1 или C_2. Каждый вход представляется двумя лингвистическими понятиями, что позволяет ограничиться всего четырьмя правилами.

Одна из возможных структур нечеткой нейронной сети для решения подобной задачи приведена на рис. 3.19.

Сеть состоит из трех слоев нейронов.

- *Слой 1.* Выходы нейронов данного слоя определяют степени принадлежности входных переменных к соответствующим нечетким множествам A_1, A_2, B_1, B_2.

Здесь выбраны функции принадлежности колоколообразного вида. Например,

$$A_j(x_1) = \exp\left[-\frac{1}{2}\left(\frac{x_1 - a_{j1}}{b_{j1}}\right)^2\right],\ B_j(x_2) = \exp\left[-\frac{1}{2}\left(\frac{x_2 - a_{j2}}{b_{j2}}\right)^2\right],\ j = 1, 2,$$

с набором параметров a_{i1}, a_{i2}, b_{i1}, b_{i2}.

Значения этих параметров корректируются в процессе обучения сети, основанном на градиентном методе.

- *Слой 2.* Каждый нейрон этого слоя является нечетким нейроном «И».
- *Слой 3.* Нейроны данного слоя являются обычными нейронами, осуществляющими взвешенное суммирование значений выходов нейронов предыдущего слоя. А их выходы формируются с использованием активационных функций, например, сигмоидного типа. Эти выходы трактуются как степени принадлежности предъявленного объекта первому или второму классу.

Алгоритм обучения данной сети не отличается от ранее рассмотренного.

3.6. Генетические алгоритмы

3.6.1. Естественный отбор в природе

Согласно эволюционной теории каждый биологический вид целенаправленно развивается и изменяется для того, чтобы наилучшим образом приспособиться к окружающей среде. Эволюция в этом смысле представляет процесс оптимизации всех живых организмов. Природа решает эту задачу оптимизации путем естественного отбора. Его суть состоит в том, что более приспособленные особи имеют больше возможностей для выживания и размножения и, следовательно, приносят больше потомства, чем плохо приспособленные особи. При этом благодаря передаче генетической информации (*генетическому наследованию*) потомки наследуют от родителей основные их качества. Таким образом, потомки сильных особей также будут относительно хорошо приспособленными, а их доля в общей массе особей будет возрастать. После смены нескольких десятков или сотен поколений средняя приспособленность особей данного вида заметно возрастает.

Дадим краткую справку о том, как устроены механизмы генетического наследования. В каждой клетке любого животного содержится вся генетическая информация данной особи. Эта информация записана в виде набора молекул ДНК, каждая из кото-

рых представляет собой цепочку, состоящую из молекул *нуклеотидов* четырех типов, обозначаемых A, T, C и G. Собственно информацию несет порядок следования нуклеотидов в ДНК. Таким образом, генетический код особи – это длинная строка, где используются всего 4 символа. В животной клетке каждая молекула ДНК окружена оболочкой, такое образование называется *хромосомой*.

Каждое врожденное качество особи (цвет глаз, наследственные болезни, тип волос и т. д.) кодируется определенной частью хромосомы, которая называется *геном* этого свойства. Например, ген цвета глаз содержит информацию, кодирующую определенный цвет глаз. Различные значения гена называются его *аллелями*.

При размножении особей происходит слияние двух родительских половых клеток, и их ДНК взаимодействуют, образуя ДНК потомка. Основной способ взаимодействия – *кроссовер (crossover, скрещивание)*. При кроссовере ДНК предков делятся на две части, а затем обмениваются своими половинками.

При наследовании возможны мутации, в результате которых могут измениться некоторые гены в половых клетках одного из родителей. Измененные гены передаются потомку и придают ему новые свойства. Если эти новые свойства полезны, они, скорее всего, сохранятся в данном виде. При этом произойдет скачкообразное повышение приспособленности вида.

3.6.2. Что такое генетический алгоритм

Пусть дана некоторая сложная целевая функция, зависящая от нескольких переменных, и требуется решить задачу оптимизации, т. е. найти такие значения переменных, при которых значение функции максимально или минимально.

Эту задачу можно решить, применяя известные биологические эволюционные подходы к оптимизации. Будем рассматривать каждый вариант (набор значений переменных) как особь, а значение целевой функции для этого варианта – как приспособленность данной особи. Тогда в процессе эволюции приспособленность особей будет возрастать, а значит, будут появляться все более и более оптимальные варианты. Остановив эволюцию в некоторый момент и выбрав лучший вариант, можно получить достаточно хорошее решение задачи.

Генетический алгоритм (ГА) – это последовательность управляющих действий и операций, моделирующая эволюционные процессы на основе аналогов механизмов генетического наследо-

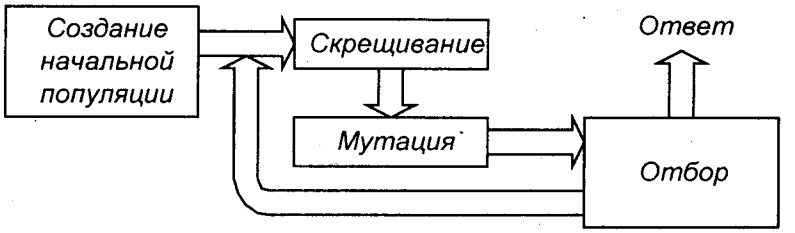


Рис. 3.20. Вариант структуры генетического алгоритма

вания и естественного отбора. При этом сохраняется биологическая терминология в упрощенном виде.

Хромосома – вектор (последовательность) из нулей и единиц, каждая позиция (бит) которого называется *геном*.

Особь (индивидуум) = генетический код – набор хромосом = вариант решения задачи.

Кроссовер – операция, при которой две хромосомы обмениваются своими частями.

Мутация – случайное изменение одной или нескольких позиций в хромосоме.

Генетические алгоритмы представляют собой скорее подход, чем единые алгоритмы. Они требуют содержательного наполнения для решения каждой конкретной задачи.

На рис. 3.20 показан один из вариантов структуры генетического алгоритма. Вначале генерируется случайная популяция – несколько особей со случайным набором хромосом (числовых векторов). Генетический алгоритм имитирует эволюцию этой популяции как циклический процесс скрещивания особей, мутации и смены поколений (отбора).

В течение жизненного цикла популяции, т. е. в результате нескольких случайных скрещиваний (посредством кроссовера) и мутаций, к ней добавляется какое-то количество новых вариантов. Далее происходит отбор, в результате которого из старой популяции формируется новая, после чего старая популяция погибает. После отбора к новой популяции опять применяются операции кроссовера и мутации, затем опять происходит отбор, и так далее.

Отбор в генетическом алгоритме тесно связан с принципами естественного отбора следующим образом:

- приспособленность особи соответствует значению целевой функции на заданном варианте;

• выживание наиболее приспособленных особей соответствует тому, что популяция следующего поколения вариантов формируется с учетом целевой функции. Чем приспособленнее особь, тем больше вероятность ее участия в кроссовере, т. е. в размножении.

Таким образом, модель отбора определяет, как следует строить популяцию следующего поколения. Как правило, вероятность участия особи в скрещивании берется пропорциональной ее приспособленности. Часто используется так называемая *стратегия элитизма*, при которой несколько лучших особей переходят в следующее поколение без изменений, не участвуя в кроссовере и отборе. В любом случае каждое следующее поколение будет в среднем лучше предыдущего. Когда приспособленность особей перестает заметно увеличиваться, процесс останавливают и в качестве решения задачи оптимизации берут наилучший из найденных вариантов.

Математически, изложенное можно представить следующим образом. Пусть имеется некоторая целевая функция от многих переменных, у которой необходимо найти глобальный максимум или минимум:

$$f(x_1, x_2, x_3, \ldots, x_n).$$

Представим независимые переменные в виде хромосом. Для этого выполним кодирование независимых переменных либо в двоичном формате, либо в формате с плавающей запятой.

В случае двоичного кодирования используется n бит для каждого параметра, причем, n может быть различным. Если параметр может изменяться между минимальным (min) и максимальным (max) значениями, можно использовать следующие формулы для преобразования:

$$r = \frac{g(\max - \min)}{(2^n - 1)} + \min, \quad g = \frac{(r - \min)}{(\max - \min)(2^n - 1)},$$

где g – значение параметра в двоичном формате, r – значение параметра в формате с плавающей запятой.

Хромосомы в формате с плавающей запятой задаются путем последовательного размещения закодированных параметров друг за другом.

Наиболее хорошие результаты дает вариант представления хромосом в двоичном формате (особенно, при использовании кодов Грея). Однако в этом случае необходимо постоянно осуществлять кодирование/декодирование параметров (генов).

Рассмотрим работу генетического алгоритма более подробно. Заранее подбирается некоторое представление для рассматриваемого решения, размер и структура популяции. В первом поколении случайным образом генерируется популяция хромосом. Обычно, размер популяции постоянен. Определяется «полезность» хромосом. После чего генетический алгоритм может начинать генерировать новую популяцию.

Далее осуществляется репродуцирование, состоящее из:

• селекции;

• и трех генетических операторов: кроссовера, мутации и инверсии, порядок применения которых не важен.

Из трех генетических операторов кроссовер является наиболее важным. Он генерирует новую хромосому потомка, объединяя генетический материал двух родителей. Существует несколько вариантов кроссовера. Наиболее простым является одноточечный, в котором берутся две хромосомы и «перерезаются» в случайно выбранной точке. Хромосома потомка получается из начала одной и конца другой родительских хромосом:

0011001011100101|1000 → 0011001011100101***11100***
11010110110100011100

Мутация представляет собой случайное изменение хромосомы (обычно простым изменением состояния одного из битов на противоположное). Данный оператор позволяет, во-первых, более быстро находить локальные экстремумы и, во-вторых, перескочить в другой локальный экстремум:

0011001011100***1***011000 → 0011001011100***1***111000

Инверсия изменяет порядок бит в хромосоме путем циклической перестановки (случайное количество раз). Многие модификации ГА обходятся без данного генетического оператора:

0011001011100101***11000*** → ***11000***0011001011100010

По скорости определения оптимума целевой функции генетические алгоритмы на несколько порядков превосходят случайный поиск. Причина этому заключается в том, что большинство систем имеют довольно независимые подсистемы. Вследствие чего при обмене генетическим материалом от каждого из родителей берутся гены, соответствующие наиболее удачному варианту определенной подсистемы (неудачные варианты постепенно погибают). Генетический алгоритм позволяет накапливать удачные решения для таких систем в целом.

Генетические алгоритмы менее применимы для систем, которые сложно разбить на подсистемы. Кроме того, они могут давать сбои из-за неудачного порядка расположения генов (например, если рядом расположены параметры, относящиеся к различным подсистемам), при котором преимущества обмена генетическим материалом сводятся к нулю. Это замечание несколько сглаживается в системах с диплоидным (двойным) генетическим набором.

Данные, которые закодированы в генотипе, могут представлять собой команды какой-либо виртуальной машины. В таком случае можно говорить об эволюционном или генетическом программировании. В простейшем случае, можно ничего не менять в генетическом алгоритме. Однако в таком случае, длина получаемой последовательности действий (программы) получается не отличающейся от той (или тех), которая является «затравкой» на этапе инициализации. Современные алгоритмы генетического программирования функционируют в системах с переменной длиной генотипа.

3.6.3. Обучение нечетких нейронных сетей на основе генетических алгоритмов

Одной из наиболее востребованных областей применения генетических алгоритмов являются задачи обучения нейронных сетей, в том числе и нечетких, путем подбора адекватных параметров. Общими этапами такого обучения являются следующие:

ШАГ 1. Выделение управляющих параметров задачи обучения.

ШАГ 2. Получение решения при фиксированных значениях параметров.

ШАГ 3. Определение рассогласованности полученного и требуемого решений.

ШАГ 4. Выбор новых значений параметров на основе работы генетического алгоритма.

ШАГ 5. Останов в случае получения удовлетворительной рассогласованности решения, иначе – переход к шагу 2.

В качестве управляющих параметров обучения нечетких нейронных сетей, влияющих на качество решения, могут быть выбраны параметры функций принадлежности (см. разд. 3.4.2), а также различная формализация логических правил.

3.6.4. Особенности генетических алгоритмов

Генетические алгоритмы – не единственный способ решения задач оптимизации. Кроме него существуют два основных подхода для решения таких задач – переборный и локально-градиентный, каждый из которых имеет свои достоинства и недостатки.

Сравним стандартные подходы с генетическими алгоритмами на примере *задачи коммивояжера* (TSP – Travelling Salesman Problem), суть которой состоит в нахождении кратчайшего замкнутого пути обхода городов, заданных своими координатами.

Уже для 30 городов поиск оптимального пути представляет собой сложную задачу, побудившую развитие новых методов (в том числе нейронных сетей и генетических алгоритмов).

Каждый вариант решения (для 30 городов) – это числовая строка, где на j-м месте стоит номер j-го по порядку обхода города. Таким образом, в этой задаче 30 параметров, причем, не все комбинации значений допустимы.

Переборный метод наиболее прост в программировании. Для поиска оптимального решения (максимума целевой функции) требуется последовательно вычислить значения целевой функции во всех возможных точках, запоминая максимальное из них. Недостатком метода является большая вычислительная сложность: требуется просчитать длины более 10^{30} вариантов путей, что совершенно нереально. Однако, если перебор всех вариантов за разумное время возможен, то найденное решение является оптимальным.

Второй подход основан на методе градиентного спуска. Вначале выбираются некоторые случайные значения параметров, а затем эти значения постепенно изменяют, добиваясь наибольшей скорости роста целевой функции. При достижении локального максимума такой метод останавливается, поэтому для поиска глобального оптимума требуются дополнительные меры.

Градиентные методы работают быстро, но не гарантируют оптимальности найденного решения. Они идеальны для применения в так называемых *унимодальных* задачах, где целевая функция имеет единственный локальный максимум (он же – глобальный). Однако задача коммивояжера таковой не является.

Практические задачи, как правило, *мультимодальны* и многомерны, т. е. содержат много параметров. Для них не существует универсальных методов, позволяющих достаточно быстро найти абсолютно точные решения. Комбинируя переборный и градиентный методы, можно получить приближенные решения, точность которых будет возрастать с увеличением времени расчета.

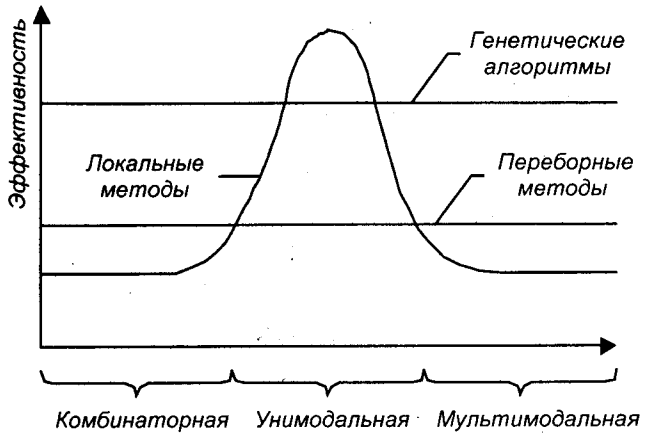


Рис. 3.21. Эффективность генетических алгоритмов

Генетический алгоритм представляет собой именно такой комбинированный метод. Механизмы скрещивания и мутации в каком-то смысле реализуют переборную часть метода, а отбор лучших решений – градиентный спуск. На рис. 3.21 показано, что такое сочетание обеспечивает устойчиво хорошую эффективность генетического поиска для любых типов оптимизационных задач.

Таким образом, если на некотором множестве задана сложная функция от нескольких переменных, то генетический алгоритм за разумное время находит значение функции достаточное близкое к оптимальному. Задавая время расчета, можно получить одно из лучших решений, которые реально получить за это время.

Часть II

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Глава 4

ОСНОВНЫЕ ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ ПРОГРАММ МОДЕЛИРОВАНИЯ НЕЙРОННЫХ СЕТЕЙ

Меньше всего авторам хотелось бы, чтобы после прочтения первой части данной книги у читателя сложилось впечатление о нейронных сетях как о какой-то непостижимой вещи, требующей как серьезного знания математики, так и программирования.

На самом деле, для использования аппарата нейронных сетей ни того, ни другого не требуется. Единственное, что нужно после формулировки задачи, это подготовить, например, в форме табл. 4.1 данные для обучения сети, а далее – использовать какую-либо из известных программ моделирования нейронных сетей и следовать инструкциям этой программы.

Таблица 4.1

Исходные данные для обучения нейронной сети

x_1	x_2	...	x_n	y_1	y_2	...	y_m
x_{11}	x_{21}	...	x_{n1}	y_{11}	y_{21}	...	y_{m1}
x_{12}	x_{22}	...	x_{n2}	y_{12}	y_{22}	...	y_{m2}
...	...	...	...	...	...	...	...
x_{1N}	x_{2N}	...	x_{nN}	y_{1N}	y_{2N}	...	y_{mN}

Первая (верхняя) строка таблицы соответствует обозначениям входных и выходных переменных поставленной задачи, каждая последующая строка – значениям входного и целевого (требуемого) выходного вектора.

Но какие же программы моделирования нейронных сетей известны, и какие из являются наилучшими? Остановимся на этом подробнее.

4.1. Общие сведения о программах моделирования нейронных сетей

Программу моделирования нейронной сети обычно называют программой–имитатором или нейропакетом, понимая под этим программную оболочку, эмулирующую для пользователя среду нейрокомпьютера на обычном компьютере.

В настоящее время на рынке программного обеспечения имеется множество самых разнообразных программ для моделирования нейронных сетей. Можно выделить несколько основных функций, которые реализованы во всех этих программах.

Формирование (создание) нейронной сети

Для решения разных практических задач требуются различные модели нейронных сетей. Модель нейронной сети определяется моделями нейронов и структурой связей сети.

Программы–имитаторы в зависимости от структуры связей реализуют следующие группы нейронных сетей.

- *Многослойные нейронные сети.* Нейроны в таких сетях делятся на группы с общим входным сигналом – слои. Различают несколько типов связей между слоями с номерами q и $(q + p)$:
  - последовательные ($p = 1$);
  - прямые ($p > 1$);
  - обратные ($p < 0$).

Связи между нейронами одного слоя называют латеральными (боковыми).

- *Полносвязные нейронные сети.* Каждый нейрон в полносвязных сетях связан со всеми остальными. На каждом такте функционирования сети на входы нейронов подается внешний входной сигнал и выходы нейронов предыдущего такта.
- *Нейронные сети с локальными связями.* Нейроны в таких сетях располагаются в узлах прямоугольной или гексагональной решетки. Каждый нейрон связан с небольшим числом (4, 6 или 8) своих топологических соседей.

- *Неструктурированные нейронные сети.* К этой группе относятся все модели нейронных сетей, которые нельзя отнести ни к одной из предыдущих групп.

Модели реализуемых программами–имитаторами нейронов чрезвычайно разнообразны. В простейшем случае нейроны первого порядка выполняет взвешенное суммирование компонентов входного вектора и нелинейное преобразование результата суммирования. Нейроны более высоких порядков осуществляют перемножение двумерных матриц и многомерных тензоров.

В моделях нейронов используются различные варианты нелинейных преобразований. Наиболее часто используются сигмоидальные, кусочно-линейные и жесткие пороговые функции активации. В сети все нейроны могут иметь как одинаковые (гомогенная сеть), так и различные функции активации (гетерогенная сеть).

Для построения нейронной сети, ориентированной на решение конкретной задачи, используются процедуры формирования нейронных сетей, которые обеспечивают ввод указанных характеристик моделей нейронов и структур нейронных сетей.

Каждая группа моделей нейронных сетей может быть использована для решения лишь некоторого ограниченного класса практических задач. Так, многослойные и полносвязные нейронные сети с сигмоидальными передаточными функциями используются для распознавания образов и адаптивного управления; нейронные сети с локальными связями – для обработки изображений и некоторых других частных задач. Для решения задач линейной алгебры используются многослойные сети с особыми передаточными функциями.

Лишь для небольшого числа моделей нейронных сетей существует строгое математическое обоснование возможности их применения для решения конкретных практических задач. В наибольшей степени теоретически проработаны двухслойные нейронные сети с сигмоидальными передаточными функциями. На основе приведенной в гл. 1 теоремы Колмогорова–Арнольда доказано, что такие сети могут реализовывать любые отображения входного сигнала в выходной. К построению многопараметрических отображений сводится большинство задач распознавания, управления, идентификации.

Обучение нейронной сети

В большинстве программ–имитаторов предлагаются стандартные процедуры обучения нейронных сетей, ориентированные на конкретные нейропарадигмы.

Как правило, в нейропакетах реализуется возможность задания различных типов данных и различных размерностей входных и

выходных сигналов в зависимости от решаемой задачи. В качестве входных данных в обучающей выборке могут использоваться растровые изображения, таблицы чисел, распределения. Типы входных данных – бинарные (0 и 1), биполярные (–1 и +1) числа, целые или действительные числа из некоторого диапазона. Выходные сигналы сети – векторы целых или действительных чисел.

Для решения практических задач часто требуются обучающие выборки большого объема. Поэтому в ряде нейропакетов предусмотрены средства, облегчающие процесс формирования и использования обучающих примеров. Однако в настоящее время отсутствует универсальная методика построения обучающих выборок и набор обучающих примеров, как правило, формируется индивидуально для каждой решаемой задачи.

В качестве функции ошибки, численно определяющей сходство всех текущих выходных сигналов сети и соответствующих требуемых выходных сигналов обучающей выборки, в большинстве случаев используется среднеквадратичное отклонение. Однако в ряде нейроимитаторов существует либо возможность выбора, либо задания своей функции ошибки.

Реализуемые в нейропакетах алгоритмы обучения нейронных сетей можно разделить на три группы: градиентные, стохастические, генетические. Градиентные алгоритмы (первого и второго порядков) основаны на вычислении частных производных функции ошибки по параметрам сети. В стохастических алгоритмах поиск минимума функции ошибки ведется случайным образом. Генетические алгоритмы комбинируют свойства стохастических и градиентных алгоритмов: на основе аналога генетического наследования реализуют перебор вариантов, а на основе аналога естественного отбора – градиентный спуск.

При обучении нейронных сетей, как правило, используются следующие критерии останова:

- при достижении некоторого малого значения функции ошибки;
- в случае успешного решения всех примеров обучающей выборки (при неизменности выходных сигналов сети).

В нейроимитаторах предусмотрено наличие специальных процедур инициализации перед обучением сети, т. е. присваивания параметрам сети некоторых малых случайных значений.

Обучение представляет собой итерационную процедуру, которая при реализации на персональных компьютерах требует значительного времени. Скорость сходимости алгоритма обучения является одной из самых важных характеристик программ для моделирования нейронных сетей.

Тестирование обученной нейронной сети

Для проверки правильности обучения построенной нейронной сети в нейроимитаторах предусмотрены специальные средства ее тестирования. В сеть вводится некоторый сигнал, который, как правило, не совпадает ни с одним из входных сигналов примеров обучающей выборки. Далее анализируется получившийся выходной сигнал сети.

Тестирование обученной сети может проводиться либо на одиночных входных сигналах, либо на тестовой выборке, которая имеет структуру, аналогичную обучающей выборке, и также состоит из пар (<вход>, <требуемый выход>). Обычно, обучающая и тестовая выборки не пересекаются. Тестовая выборка строится индивидуально для каждой решаемой задачи.

4.2. Характеристики современных нейропакетов

В настоящее время известно большое количество нейропакетов, выпускаемых рядом фирм и отдельными исследователями и позволяющих конструировать, обучать и использовать нейронные сети для решения практических задач. Характеристики некоторых из них приведены в табл. 4.2.

Рассмотрим несколько нейропакетов, предназначенных для реализации на персональных компьютерах в различных операционных средах, по степени их универсальности, а также с точки зрения простоты использования и наглядности представления информации.

Таблица 4.2

Характеристики некоторых современных нейропакетов

Производитель	Наименование продукта	Платформа	Функциональные характеристики
1	2	3	4
AbTech (Шалетсвилл, шт. Вайоминг, США); http://www.abtech.com/MQ1.HTM	ModelQuest	Windows	Интегральная среда для прогнозирования, принятия решений и управления. В основе лежит концепция «статистических сетей» (Statistical Networks) как сплава НС и статистических методов обработки
AIWare Inc.	Process Advisor	Windows	Нейропакет для решения задач управления динамическими процессами

Продолжение табл. 4.2

1	2	3	4
Attar Software (Ланкашир, Великобритания); http://www.attar.com	XpertRule Analyser	Windows	Пакет для построения моделей данных и выявления скрытых закономерностей на основе вероятностных правил, генетических алгоритмов или НС
BioComp Systems (Редмонд, шт. Вашингтон, США); http://www.biocomp.com	NeuroGenetic Optimizer (NGO)	Windows	Инструментальная среда для оптимизации входных сигналов и структуры НС на основе генетических алгоритмов
California Scientific Software (Невада-Сити, шт. Калифорния, США); http://www.calsci.com	BrainMaker	Windows, Macintosh	Инструментальная среда для разработки приложений на основе НС для распознавания образов, прогнозирования и нейросетевой памяти
Megaputer Intelligence (Москва, Россия); http://mosca.sai.msu.su/-mp/megapute.html	PolyAnalyst	OS/2 Warp, Windows	Объектно-ориентированная среда для анализа данных, поиска закономерностей и представления их в символическом виде
NCS (Саутгемптон, Великобритания); http://www.demon.co.uk/skylake/software.html	NeuFrame	Windows	Пакет для разработки приложений на основе НС. Реализует комбинированные алгоритмы НС и нечеткой логики.
NeuroDimension, Inc. (Сиэтл, США) http://www.nd.com	NeuroSolutions	HP, Sun, Windows	Нейропакет с широкими средствами визуализации, для конструирования НС с произвольной топологией и процедурами обучения
NeoVista (Купертино, шт. Калифорния, США); http://www.neovista.com	Decision Series	HP, Sun, Digital	Система для автоматического поиска знаний в коммерческих базах данных. Позволяет строить прогнозы, используя технологию обучаемых НС
NeuralWare (Питсбург, шт. Пенсильвания, США); http://www.neuralware.com	NeuralWorks Professional II/ Plus	PC, Sun, IBM RS/6000, Apple Macintosh, SGI, Digital, HP	Инструментальная среда для разработки приложений на основе 25 моделей НС с полный набором средств для обучения и тестирования НС
Promised Land Technologies, Inc. http://promland.com/demo.htm	Braincel	Windows	Нейропакет для моделирования НС прямого распространения с быстрым алгоритмом обучения

Продолжение табл. 4.2

1	2	3	4
StatSoft	Statistica Neural Networks	Windows	Пакет нейросетевого анализа для конструирования и применения НС
Southern Scientific CC, South Africa http://www.simtel.net/simtel.net/win3/neural-pre.html	Neural10	Windows	Нейропакет, реализующий сеть прямого распространения с алгоритмом обратного распространения ошибки для решения задач распознавания
SPSS (Чикаго, шт. Иллинойс, США) http://www.spss.com	Neural Connection	Windows	Среда для решения задач классификации, прогнозирования и анализа временных рядов на основе 4-х архитектур НС
Thinking Machines (Бэдфорд, шт. Массачусетс, США); http://www.think.com	Darwin	Аппаратные платформы от ПК до суперкомпьютеров	Инструментальная среда для анализа данных. StarNet – модуль для моделирования НС
Universal Problem Solvers, Inc. http://www.simtel.net/simtel.net/win3/neural-pre.html	MPIL (Multi-Pass Instance-Based Learning)	Windows	Нейропакет для кластерного анализа извлечения знаний из экспериментальных данных
Ward Systems Group	NeuroShell 2	Windows	Нейропакет для моделирования наиболее известных нейропарадигм (многослойных НС, сетей Кохонена и др.)
Ward Systems Group	NeuroShell 2	Windows	Нейропакет для моделирования наиболее известных нейропарадигм (многослойных НС, сетей Кохонена и др.)
C. Jensen, США http://www.simtel.net/simtel.net/win95/neural-pre.html	QwikNet	Windows	Нейропакет, моделирующий многослойную сеть прямого распространения и различные модификации алгоритма обратного распространения ошибки
S.Wolstenholme, UK http://www.simtel.net/simtel.net/win3/neural-pre.html	Neural Planner	Windows	Нейропакет для моделирования нейронных сетей различной конфигурации для решения задач классификации объектов, анализа случайных процессов, создания эффективных экспертных систем.

Окончание табл. 4.2

1	2	3	4
АОЗТ «Альфа Систем» (ЛЭТИ, С.- Петербург); dorv@actor.ru	Neuro Office	Windows	Пакет для проектирования интеллектуальных программных модулей на основе НС с ядерной организацией
Институт вычислительного моделирования СО РАН, Красноярск,Россия; Царегородцев В. Г. tsar@cc.krascience.rssi.ru	NeuroPro	Windows	Нейропакет для извлечения знаний из таблиц данных
ООО «НейрОК» (МГУ, Москва, Россия); www.neurok.ru	Excel Neural Package	Windows	Нейропакет для статистического прогнозирования и анализа многомерных данных
НейроКомп (ВЦ СО РАН, Красноярск, Россия); amse@cc.krascience.rssi.ru	Глаз	Windows	Распознавание визуальных образов. Используется для обработки аэрокосмической информации
НейроКомп (ВЦ СО РАН, Красноярск, Россия); amse@cc.krascience.rssi.ru	Клаб	Windows	Среда для решения задач классификации, в том числе в медицинской, психологической и технической диагностике
НейроКомп (ВЦ СО РАН, Красноярск, Россия); amse@cc.krascience.rssi.ru	MultiNeuron	DOS, Windows	Программный нейроимитатор для решения задач из различных предметных областей
АО «Нейрома-РД» (Москва, Россия); http://www.nbdatyner.com/nbd	«Нейроимитатор»	Windows	Пакет программ моделирования биологических нейронных сетей
СНИЦ «Нейросистемы» АН Татарстана	NeuralMaker	Windows	Пакет для разработки прикладных НС
ТОО НПИЦ «Микросистемы» (Москва, Россия)	TextAnalyst	Windows	Нейросетевая система автоматического смыслового анализа текстов
Аргусофт (Москва, Россия)	Neuroline	Windows	Инструментальная среда на основе НС
ЗАО «Интраст» (Москва, Россия)		Windows	Комплекс прогнозирования курса акций и финансового анализа

Проведем анализ, используя данные журнала «Нейрокомпьютер», наиболее мощных зарубежных нейропакетов:

1) NeuroSolutions фирмы NeuroDimension Inc.;
2) NeuralWorks Professional II/ Plus с модулем UDND фирмы NeuralWare Inc.;
3) Process Advisor фирмы AIWare Inc.;
4) NeuroShell 2 фирмы Ward Systems Group;
5) BrainMaker Pro фирмы California Scientific Software.

В качестве тестовой рассмотрена задача прогнозирования многомерного временного ряда. В качестве архитектуры взята многослойная нейронная сеть с различными критериями и алгоритмами обучения.

В результате тестирования все нейропакеты показали практически одинаковые результаты по времени обучения с помощью алгоритма обратного распространения ошибки. Поэтому оценка производилась по показателям нейропакетов, связанным с возможностями использования различных нейронных структур, критериев оптимизации и алгоритмов обучения сетей, а также с простотой использования нейропакетов и наглядностью представляемой информации.

При тестировании учитывались и возможности использования нейропакетов для разработки нейронных систем для решения прикладных задач. В результате были определены следующие показатели сравнения, отражающие возможности нейропакетов:

- простота создания и обучения нейронной сети, интуитивно понятный интерфейс;
- простота подготовки обучающей выборки;
- наглядность и полнота представления информации в процессе создания и обучения нейронной сети;
- количество реализуемых стандартных нейропарадигм, критериев и алгоритмов обучения нейронной сети;
- возможность создания собственных нейронных структур;
- возможность использования собственных критериев оптимизации;
- возможность использования собственных алгоритмов обучения нейронной сети;
- простота обмена информацией между нейропакетом и другими приложениями операционной системы;
- открытость архитектуры, т. е. возможность расширения нейропакета за счет собственных программных модулей;
- возможность генерации исходного кода;

• наличие макроязыка для ускорения работы с нейропакетом.

Первые три показателя важны для начинающих пользователей нейропакетов, 3–8 – для опытных пользователей, решающих конкретные прикладные задачи, показатели же 7–11 являются определяющими при создании интегрированных. нейронных инструментальных систем на базе нейропакетов и важны для профессиональных разработчиков и программистов.

Оценка проведена по десятибалльной шкале и сопровождена комментариями, призванными помочь составить собственное мнение о протестированных нейропакетах.

1) Нейропакет NeuroSolutions фирмы NeuroDimension Inc.

NeuroSolutions предназначен для моделирования большого набора нейронных сетей. Основное его достоинство состоит в гибкости: помимо традиционных нейросетевых парадигм (полносвязных и многослойных НС, самоорганизующихся карт Кохонена) нейропакет включает в себя мощный редактор визуального проектирования нейронных сетей, позволяющий создавать любые нейронные структуры и алгоритмы их обучения, а также вводить собственные критерии обучения. NeuroSolutions имеет хорошие средства визуализации структур, процессов и результатов обучения и функционирования нейронных сетей. Это ставит данный нейропакет на уровень CAD-систем (систем автоматизированного проектирования) проектирования и моделирования НС.

Пакет предназначен для работы Windows. Помимо средств взаимодействия с операционной системой (OLE), нейропакет снабжен генератором исходного кода и позволяет использовать внешние модули при создании и обучении нейронной сети. Пакет поддерживает программы, написанные на языке C++ для компиляторов Microsoft Visual C++ и Borland C++, а также в виде DLL-кода. Таким образом, NeuroSolutions является гибкой открытой системой, которую можно при необходимости дополнять и модифицировать. Пакет содержит встроенный макроязык, позволяющий производить практически любую настройку под конкретную задачу.

В пакете реализуется большой перечень нейронов, включая взвешенный сумматор (нейрон первого порядка), нейроны высших порядков (с перемножением входов), а также непрерывный интегрирующий нейрон. Функция активации нейрона может быть выбрана из пяти стандартных (кусочно-линейная, функция знака и три типа сигмоидальных) функций, а также задана пользователем. Связи между нейронами задаются произвольно на этапе проектирования и могут быть изменены в процессе работы. Поддерживаются все типы связей: прямые, перекрестные и обратные. При этом хорошо реализована схема организации связей: можно задать одну векторную связь с заданной весовой матрицей, а не набор скалярных связей с весовыми коэффициентами.

Нейропакет NeuroSolutions содержит мощные средства для организации обучающих выборок. Встроенные конверторы данных поддерживают графические изображения в формате BMP, текстовые файлы с числовыми или символьными данными, а также функции непрерывного аргумента (например, времени), заданные в аналитическом виде или в виде выборки значений. Нейропакет позволяет использовать любые внешние конверторы данных.

На этапе обучения может быть использован широкий круг критериев обучения, как дискретных, так и непрерывных. Помимо этого можно вводить собственные критерии. Можно использовать как встроенный алгоритм обучения типа back-propagation или дельта-правила, так и использовать собственный. Система визуализации процесса обучения позволяет проводить анализ изменения весов непосредственно в процессе обучения и вносить коррективы. Может быть введена шумовая характеристика как при тестировании, так и при обучении нейронной сети. Можно задать аддитивный белый шум, шум произвольной природы, а также любой заданный тип шума (например, белый мультипликативный).

Neurosolutions содержит генератор (мастер) стандартных нейросетевых архитектор (Neural Wizard), с помощью которого быстро задается архитектура, подбирается обучающая выборка, критерии и методы обучения нейронной сети.

Оценка нейропакета

• *Простота использования – 9.*

Прост в использовании, имеет хороший интуитивно понятный интерфейс с возможностями настройки. Неудобством является несколько непривычная терминология, примененная разработчиками, за что и снят один балл.

• *Простота формирования обучающей выборки – 9.*

Обучающая выборка может быть сформирована либо на этапе создания нейронной сети средствами нейропакета, либо задана для уже созданной НС. Поддерживаются основные типы данных: текстовые данные в формате ASCII, бинарные данные в виде исполняемого модуля и изображения в формате BMP. При необходимости может быть подключен внешний конвертор данных. Балл снят за отсутствие встроенных конверторов некоторых популярных форматов данных, в частности звуковых файлов (в формате WAV).

- *Наглядность представления информации – 10.*

NeuroSolutions является лучшим из сравниваемых пакетов по наглядности представления информации. Система визуализации очень гибкая и хорошо продумана.

- *Реализация стандартных нейронных парадигм и алгоритмов обучения – 8.*

Поддерживает основные нейропарадигмы (сеть обратного распространения, сети Кохонена, оптимизирующие нейронные сети и т. д.). Реализуется также широко используемый набор алгоритмов обучения: обратного распространения ошибки, градиентные методы, обучение без учителя и т. д. Два балла сняты за отсутствие некоторых популярных нейропарадигм, в частности, сетей Хопфилда.

- *Возможность создания собственных нейронных структур – 10.*

В нейропакет встроен мощный нейроконструктор, позволяющий создавать любые нейронные структуры.

- *Возможность использования собственных критериев обучения – 8.*

Позволяет задавать собственные критерии обучения нейронной сети, что, однако, связано с необходимостью подключения внешних модулей. За что и сняты два балла.

- *Возможность использования собственных алгоритмов обучения – 10.*

Позволяет использовать любые собственные алгоритмы обучения в виде внешних программных модулей.

- *Обмен информацией между нейропакетом и операционной системой – 10.*

Имеет развитые средства обмена с другими приложениями операционной системы. Поддерживается OLE и технология Drag-and-Drop.

- *Открытость архитектуры – 10.*

Представляет собой открытую систему, позволяющую подключать собственные программные модули, реализующие нейронные структуры, критерии и алгоритмы обучения, данные для обучения.

- *Генератор исходного кода – 10.*

Поддерживается генератор исходного кода на языке C++ для компиляторов Microsoft Visual C++ и Borland C++.

- *Наличие макроязыка – 10.*

Встроенный макроязык облегчает настройку нейропакета.

2) Нейропакет NeuralWorks Professional II/Plus фирмы NeuralWare Inc.

NeuralWorks Professional является мощным средством для моделирования нейронных сетей. В нем реализованы 28 нейронных парадигм, а также большое количество алгоритмов обучения. Дополнительный модуль UDND (User Define Neural Dynamics) позволяет создавать собственные нейронные структуры.

Как и NeuroSolutions, NeuralWorks Professional имеет хорошую систему визуализации данных: структуры нейронной сети, изменения ошибки обучения, изменения весов и их корреляции в процессе обучения. Последнее является уникальным свойством пакета и полезна при анализе поведения сети.

В NeuralWorks Professional можно интегрировать внешние программные модули. Он имеет встроенный генератор кода, поддерживающий компилятор Microsoft Visual C++.

Способ представления информации незначительно отличается от NeuroSolutions

Оценка нейропакета

- *Простота использования – 9.*

Прост в использовании. Единственным недостатком является отсутствие возможности настройки интерфейса.

- *Простота формирования обучающей выборки – 9.*

Обучающая выборка формируется достаточно просто. Поддерживается текстовый формат данных ASCII. Пакет позволяет подключать свои конверторы данных. Балл снят за отсутствие встроенных конверторов популярных форматов данных.

- *Наглядность представления информации – 9.*

Имеет мощные и хорошо продуманные средства визуализации данных и нейронных структур. Балл снят за несколько непривычный визуальный интерфейс.

- *Реализация стандартных нейронных парадигм и алгоритмов обучения – 10.*

В пакете NeuralWorks Professional реализованы практически все известные и описанные в литературе нейронные парадигмы и алгоритмы их обучения.

- *Возможность создания собственных нейронных структур – 8 (только с модулем UDND).*

Модуль UDND позволяет формировать собственные нейронные структуры в виде внешних программных модулей. Два балла сняты за то, что модуль UDND является внешней программой к пакету.

• *Возможность использования собственных критериев обучения – 7.*

Введение собственных критериев обучения предусмотрено только с использованием модуля UDND. Два балла сняты за то, что модуль UDND является внешней программой к пакету, и еще один – за сложность введения собственного критерия обучения.

• *Возможность использования собственных алгоритмов обучения – 7.*

Три балла сняты за то же, что и ранее.

• *Обмен информацией между нейропакетом и операционной системой – 8.*

За счет мощной системы визуализации большого обмена данными не требуется, поэтому имеющихся в нейропакете средств обмена вполне достаточно. Два балла сняты за отсутствие технологии Drag-and-Drop и OLE.

• *Открытость архитектуры – 10.*

NeuralWorks Professional представляет собой открытую систему, позволяющую подключать внешние программные модули. Пакетом поддерживается компилятор Microsoft Visual C++.

• *Генератор исходного кода – 10.*

Имеется генератор исходного кода на языке C++ (Microsoft Visual C++).

• *Наличие макроязыка – 0 (отсутствует).*

3) Нейропакет Process Advisor фирмы AIWare Inc.

Process Advisor предназначен для решения задач управления динамическими процессами (в частности, технологическими процессами). Однако он может считаться универсальным нейропакетом. В нем реализована только многослойная нейронная сеть прямого распространения, обучаемая с помощью модифицированного алгоритма обратного распространения ошибки. В пакет введена возможность работы с входными сигналами как с функциями времени, а не дискретным набором точек. Такой возможностью помимо Process Advisor обладает только NeuroSolutions. Кроме того, нейропакет Process Advisor позволяет осуществлять управление внешними аппаратными контроллерами, подключаемыми к компьютеру. Именно эти две особенности делают нейропакет Process Advisor примечательным.

Оценка нейропакета

• *Простота использования – 8.*

Достаточно прост в использовании. Имеет Excel-подобный интерфейс как для задания структуры нейронной сети и данных

для обучения, так и для отображения графической информации. Балл снят за отсутствие возможностей настройки интерфейса и еще один балл за использование Excel-подобного интерфейса.

• *Простота формирования обучающей выборки – 7.*

Позволяет обрабатывать данные, представленные в текстовом виде. Редактирование данных осуществляется подобно Excel-таблице. Однако нейропакет не позволяет подключать внешние конверторы данных. Балл снят за отсутствие встроенных конверторов популярных форматов данных, и еще два – за невозможность подключения внешних конверторов.

• *Наглядность представления информации – 7.*

Наглядность такая же, как в Excel. Возможно построение двух- и трехмерных графиков. В процессе обучения отображается только изменение ошибки работы нейронной сети. Три балла сняты за использование псевдографического интерфейса, который проигрывает в сравнении с интерфейсами NeuroSolutions и NeuralWorks.

• *Реализация стандартных нейронных парадигм и алгоритмов обучения – 5.*

Реализована только многослойная нейронная сеть, обучаемая с помощью модифицированного алгоритма обратного распространения. Нейропакет позволяет работать с динамическими функциями времени, поступающими в качестве входных сигналов. Оценка обусловлена тем, что многослойная нейронная сеть представляет собой половину всех используемых нейронных парадигм.

• *Возможность создания собственных нейронных структур – 5.*

Позволяет изменять только число слоев и количество нейронов в слоях.

• *Возможность использования собственных критериев обучения – 0 (отсутствует).*

Позволяет использовать только критерий средней квадратичной ошибки без возможности его изменения.

• *Возможность использования собственных алгоритмов обучения – 3.*

При обучении возможно только менять параметры алгоритма обратного распространения ошибки.

• *Обмен информацией между нейропакетом и информационной системой – 5.*

Обмен реализуется только через буфер обмена или внешние файлы, представленные в текстовом виде.

• *Открытость архитектуры – 3.*

Позволяет осуществлять управление внешними контроллерами, подключаемыми к компьютеру. Возможность подключения внешних программных модулей отсутствует.

- *Генератор исходного кода – 0 (отсутствует).*
- *Наличие макроязыка – 0 (отсутствует).*

4) Нейропакет NeuroShell 2 фирмы Ward Systems Group

NeuroShell 2 является одной из трех программ, входящих в состав пакета The AI Trilogy и представляет собой универсальный нейропакет для моделирования нескольких наиболее известных нейронных парадигм: многослойных сетей, сетей Кохонена и т. д.

NeuroShell 2 сильно проигрывает по сравнению с NeuroSolutions и NeuralWorks. Он имеет много мелких недостатков, существенно замедляющих подготовку и работу в среде нейропакета. Кроме недостаточно продуманного интерфейса нейропакет NeuroShell имеет и усложненную систему визуализации данных. Из-за отсутствия единого интегрального контроля данных в процессе обучения или работы нейронной сети часто приходится переключаться из одного режима в другой, что неудобно в использовании.

Для NeuroShell характерна жесткая последовательность действий при работе с нейронной сетью. Это удобно для начинающих пользователей. Однако, для того, чтобы внести небольшое изменение приходится выполнять заново всю последовательность действий.

NeuroShell предоставляет хорошие средства обмена данными с другими приложениями. Он обеспечивает обмен данными, представленными в текстовом бинарном виде, а также в наиболее популярных финансовых форматах MataStock и DowJones. Нейропакет имеет генератор исходного кода на языках Visual C и Visual Basic.

Оценка нейропакета

- *Простота использования – 10.*

NeuroShell прост в использовании за счет жесткой последовательности действий по созданию и обучению нейронной сети.

- *Простота формирования обучающей выборки – 8.*

Обучающая выборка формируется достаточно просто, но из-за непродуманности интерфейса приходится выполнять слишком много лишних действий. Балл снят за отсутствие встроенных конверторов популярных форматов данных (в частности, BMP) и еще балл – за непродуманность интерфейса.

- *Наглядность представления информации – 6.*

Отображать и контролировать можно многие параметры, однако, невозможно контролировать все их одновременно. Три балла сняты за использование псевдографического интерфейса.

- *Возможность создания собственных нейронных структур – 5.*

Создание собственных нейронных структур сводится к возможности изменения числа слоев и количество нейронов в слоях.

- *Возможность использования собственных критериев обучения – 0 (отсутствует).*
- *Возможность использования собственных алгоритмов обучения – 0 (отсутствует).*
- *Обмен информацией между нейропакетом и операционной системой – 8.*

Обмен информацией реализован достаточно хорошо. Нейропакет имеет несколько встроенных конверторов для чтения данные в текстовом и бинарном форматах, а также в популярных форматах представления финансовых данных. Встроенный генератор кода позволяет получать исходные тексты программ на языках C и Basic. Два балла сняты за отсутствие технологии Drag-and-Drop и OLE.

- *Открытость архитектуры – 2.*

Не позволяет подключать внешние программные модули.

- *Генератор исходного кода – 10.*

Имеет встроенный генератор исходного кода на языках C и Basic.

- *Наличие макроязыка – 0 (отсутствует).*

5) Нейропакет BrainMaker Pro фирмы California Scientific Software

BrainMaker Pro является простым нейропакетом для моделирования многослойных нейронных сетей, обучаемых с помощью алгоритма обратного распространения ошибки. Основным его достоинством является большое число параметров настройки алгоритма обучения. В остальном BrainMaker Pro уступает NeuroSolutions и NeuralWorks, особенно, в наглядности представляемой информации и простоты интерфейса.

Оценка нейропакета

- *Простота использования – 6.*

Хотя BrainMaker Pro представляет собой достаточно простой нейропакет, однако, использовать его несколько сложнее, чем NeuroSolutions и NeuralWorks. Сложности использования BrainMaker Pro связаны с неудобством интерфейса, а также с наличием

большого количества модулей, не интегрированных в единую оболочку. Три балла сняты за использование псевдографического интерфейса, и еще один – за отсутствие интегральной оболочки.

- *Простота формирования обучающей выборки – 7.*

Этап формирования обучающей выборки достаточно прост. К сожалению, нейропакет поддерживает только текстовый формат представления данных и не позволяет использовать внешние конверторы данных. Балл снят за отсутствие встроенных конверторов популярных форматов данных, и еще два – за невозможность подключения внешних конверторов.

- *Наглядность представления информации – 4.*

Примитивные возможности отображения различной информации о состоянии нейронной сети. Шесть баллов сняты за непродуманность системы визуализации и отсутствие даже псевдографического интерфейса.

- *Реализация стандартных нейронных парадигм и алгоритмов обучения – 6.*

Реализована только одна нейронная парадигма – многослойная нейронная сеть и только один алгоритм ее обучения – метод обратного распространения ошибки. Лишний балл добавлен за большое количество параметров настройки алгоритма обучения.

- *Возможность создания собственных нейронных структур – 5.*

С помощью модуля NetMaker имеется возможность создания собственных многослойных нейронных сетей. При этом можно только задавать количество слоев, количество нейронов в слоях и функции активации нейронов. По заданным параметрам нейропакет генерирует полносвязную нейронную сеть прямого распространения, структуру которой изменить невозможно.

- *Возможность использования собственных критериев обучения – 0 (отсутствует).*

Используется только средняя квадратичная ошибка.

- *Возможность использования собственных алгоритмов обучения – 4.*

Реализован только алгоритм обратного распространения ошибки, который можно настроить путем изменения параметров скорости и точности обучения. По сравнению с Process Advisor один балл добавлен за большее количество настраиваемых параметров алгоритма обучения.

- *Обмен информацией между нейропакетом и операционной системой – 5.*

Обмен данными реализуется только через буфер обмена или внешние файлы, записанные в текстовом виде.

- *Открытость архитектуры – 0 (отсутствует).*

Не имеет средств для подключения внешних модулей.

- *Генератор исходного кода – 0 (отсутствует).*
- *Наличие макроязыка – 0 (отсутствует).*

Суммарная оценка

1) NeuroSolutions – 104;
2) NeuralWorks Professional II/ Plus с модулем UDND – 87;
3) Process Advisor – 43;
4) NeuroShell 2 – 57;
5) BrainMaker Pro 37.

Приведенные основные характеристики универсальных зарубежных нейропакетов и их сравнение с точки зрения простоты использования и спектра предоставляемых услуг для моделирования искусственных нейронных сетей дают возможность из большого количества существующих нейропакетов выбрать наиболее пригодный для практической работы. Суммарная оценка дает представление о нейропакете в целом.

Глава 5

ПРОГРАММЫ МОДЕЛИРОВАНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Помня золотое методическое правило «от простого – к сложному», рассмотрение нейропакетов начнем с простых и доступных для пользователя.

5.1. Нейропакет Neural10

5.1.1. Общая характеристика

Бесплатная версия пакета доступна через сеть Интернет по адресу: http://www.simtel.net/simtel.net/win3/neural-pre.html. Разработан пакет в 1992 г. (в Southern Scientific CC, South Africa) под Windows. Основной исполняемый файл – Slug.exe.

Пакет реализует одну нейросетевую парадигму – двухслойную сеть прямого распространения с одним алгоритмом обучения (обратного распространения ошибки). Активационная функция нейронов скрытого слоя – сигмоид, выходных нейронов – линейная.

5.1.2. Создание, обучение и работа нейронной сети

Данные для обучения должны быть подготовлены в виде файла текстового формата с расширением по умолчанию *.dat (вообще, допускаются любые расширения).

Процедура работы с пакетом такова.

1) Обучающая выборка подготавливается в виде, аналогичном отображаемому табл. 4.1, но не в виде таблицы, а в виде колонок цифр, например, исходные данные для обучения нейронной сети решению задачи реализации функции «Исключающее ИЛИ» будут иметь вид:


```
#
#
1.0000  1.0000  0.0000
0.0000  0.0000  0.0000
1.0000  0.0000  1.0000
0.0000  1.0000  1.0000
```

Первые две строки – для комментариев, первый и второй столбцы отображают значения первой (x_1) и второй(x_2) входных переменных, третий столбец – значения выходной переменной (y); разделителями являются пробелы или табуляции.

Выборка сохраняется в виде текстового файла (с рекомендуемым расширением *.dat), например, xor.dat.

2) Подготавливается пустой текстовый файл протокола (с расширением по умолчанию *.log), например, xor.log.

3) Запускается программа Slug.exe, после чего на экране появляется контрольная панель программы (рис. 5.1), а также меню с опциями **File**, **Parameters**, **Data**, **Train**, **Run**.

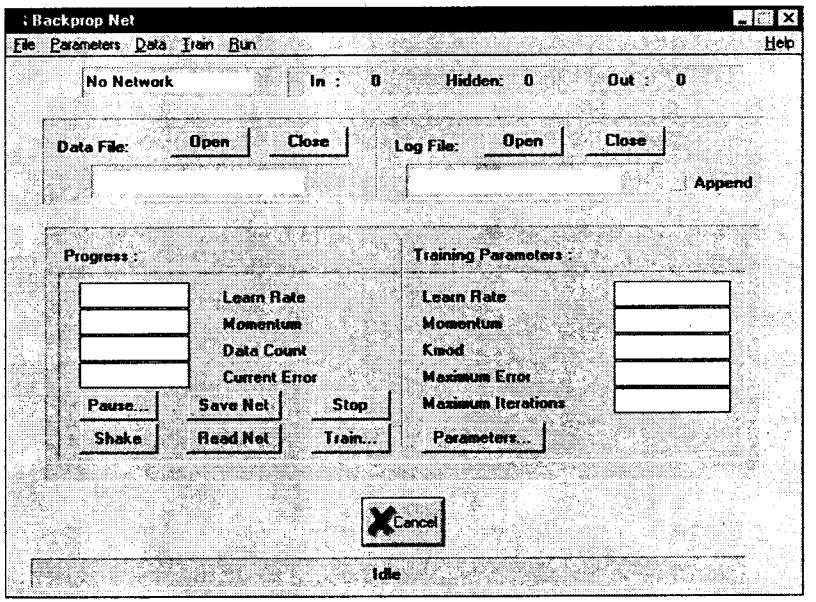


Рис. 5.1. Контрольная панель программы Slug.exe

Для создания нейронной сети выбирается опция меню **File**. В выпадающем подменю (содержащем опции **New**, **Open**, **Save**, **Save as**) выбирается пункт **New** (Новая), после чего появляется окно конструирования сети (рис. 5.2), где необходимо указать:

- число входных нейронов (Number of neurons in input layer), равное числу входных переменных; в нашем примере – 2;
- число нейронов в выходном слое (Number of neurons in output layer), равное числу выходных переменных – 1;
- число нейронов в скрытом слое (Number of neurons in hidden layer), выбираемое в соответствии с рекомендациями, приведенными в гл. 1 (формулы (1.5)–(1.8)) – 2.

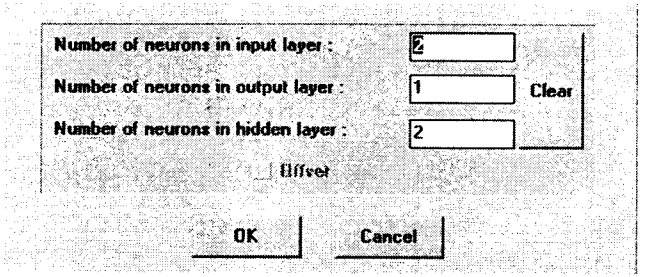


Рис. 5.2. Окно конструирования конфигурации нейронной сети

Задав данные значения, подтверждаем это нажатием кнопки **OK**, после чего опять появится вид контрольной панели рис. 5.1.

4) Теперь необходимо открыть подготовленные файлы с обучающей выборкой и протоколом. Сделать это можно или с помощью опций меню **Data**, **Input File** (для файла обучающей выборки) и **Data**, **Output File** (для файла протокола), или с помощью кнопок панели **Data File**, **Open** и **Log File**, **Open**. И в том, и в другом случае будет появляться окно вида рис. 5.3 с файлами данных с расширениями *.dat и с файлами протокола с расширениями *.log.

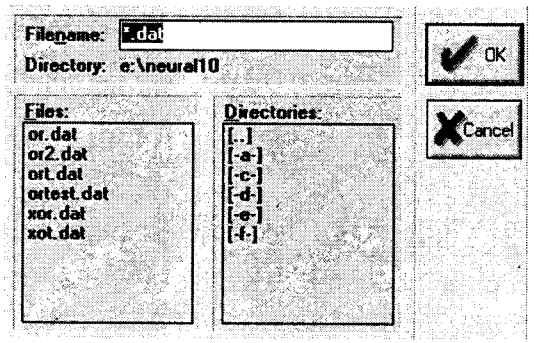


Рис. 5.3. Окно загрузки файла данных

Откроем подготовленные файлы xor.dat и xor,log.

5) Параметры процесса обучения задаются либо с помощью опции меню **Parameters**, либо с помощью одноименной клавиши в правой нижней части контрольной панели. При этом появляется окно диалога вида рис. 5.4.

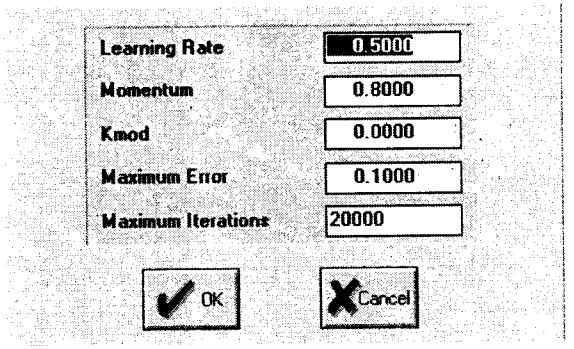


Рис. 5.4. Окно установки параметров процесса обучения нейронной сети

Параметры процесса обучения следующие: коэффициент скорости обучения Learning Rate; импульс (Momentum); максимально допустимая ошибка (Maximum Error); максимальное число итераций в процессе обучения (об этих параметрах более подробно – в разд. 1.4. и в прил. 3). На рис. 5.4 показаны значения данных параметров, устанавливаемые по умолчанию (параметр Kmod в программе не используется). Если нет каких-либо личных соображений, можно согласиться со значениями и подтвердить это нажатием кнопки **OK**. Окно при этом закроется, а панель приобретет вид рис. 5.5.

6) При нажатии кнопки **Train** можно наблюдать процесс обучения: изменяются цифры в окошках Data Count (общее количество итераций) и в Current Error (текущая ошибка). Количество итераций, естественно, все время увеличивается, а текущая ошибка должна изменяться в сторону уменьшения. Обучение заканчивается, если текущая ошибка становится меньше заданной или, если общее количество итераций достигает заданного максимального числа (процесс обучения можно принудительно приостановить нажатием кнопки **Pause**, а затем возобновить нажатием появляющейся кнопки **Resume** или вообще прекратить нажатием кнопки **Stop**).

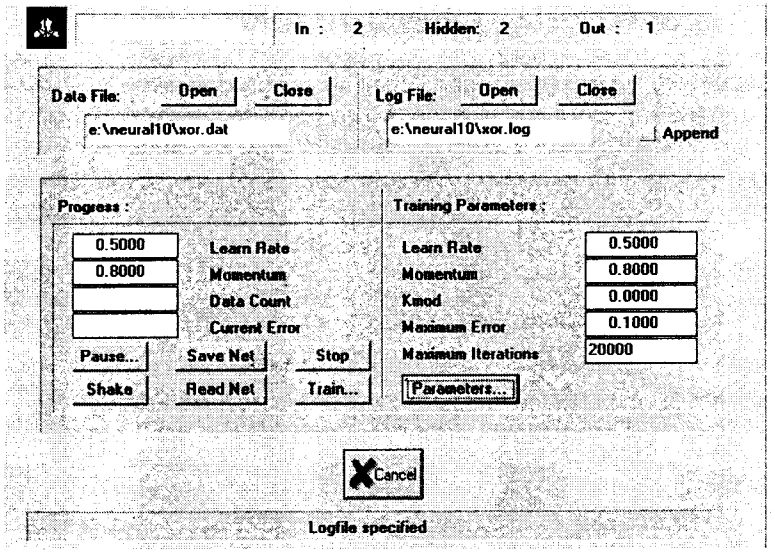


Рис. 5.5. Контрольная панель после подготовки к процессу обучения

Останов по второму критерию не очень хорош – он свидетельствует, что сеть не обучилась должным образом. Что же делать в такой ситуации?

Вариантов действий здесь четыре:

- изменить (увеличить, если возможно) объем обучающей выборки;
- изменить структуру сети (увеличить число скрытых нейронов);
- изменить параметры процесса обучения;
- и, наконец, «потрясти» сеть нажатием кнопки **Shake**. При этом небольшим случайным образом изменяются веса сети, и часто это помогает выбираться из локальных минимумом ошибки в процессе обучения.

Но в нашем случае все будет удачно, процесс обучения закончится с итоговыми цифрами, например,

Data Count 258;
Current Error 0.0575.

В нижней части контрольной панели при этом появится надпись «Trained!» (обучена).

7) Теперь с помощью кнопки **Save net** или опций меню **File**, **Save** или **File**, **Save as** можно сохранить обученную сеть под каким-либо именем с расширением *.ann, например, как xor.ann, и с

помощью соответствующих кнопок **Close** (рис. 5.5) закрыть файлы обучающей выборки и протокола.

Файл протокола содержит достаточно интересную информацию. Откроем его, например, с помощью текстового редактора. Для рассматриваемого примера содержимое файла xor.log примет следующий вид:

```
IO MATRIX:
 1.0000  1.0000  0.0000
 0.0000  0.0000  0.0000
 1.0000  0.0000  1.0000
 0.0000  1.0000  1.0000

DESIRED MATRIX:
 0.0000
 0.0000
 1.0000
 1.0000

INPUT MATRIX:
 1.0000  1.0000
 0.0000  0.0000
 1.0000  0.0000
 0.0000  1.0000

Event # 10   2.654183
Event # 20   2.087392
    . . .
Event # 240  1.285267
Event # 250  1.110611

Network response:
 inputvec :  1.00  1.00 response :  0.009
 inputvec :  0.00  0.00 response :  0.046
 inputvec :  1.00  0.00 response :  1.020
 inputvec :  0.00  1.00 response :  1.013

Final Weights:
 0.0000  0.0000 -4.3897 -1.6988  0.0000  0.0000
 0.0000  0.0000 -6.5033 -1.7948  0.0000  0.0000
 0.0000  0.0000  0.0000  0.0000 -2.9008  0.0000
 0.0000  0.0000  0.0000  0.0000  2.7870  0.0000
 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
 0.0000  0.0000  0.6582  1.9572 -0.4847  0.0000
```

Вначале повторяется обучающая выборка (IO MATRIX), затем требуемые (целевые) значения выхода этой выборки (DESIRED MATRIX), затем – значения входов в обучающей выборке (INPUT MATRIX), далее – набор данных, характеризующих изменение ошибки сети в процессе ее обучения (например, запись вида Event # 250 1,110611 означает номер итерации процесса обучения и соответствующую ошибку нейронной сети), рассчитанные значения выхода сети по наборам входных данных (Network response) и, наконец, приводится итоговая матрица весов сети (Final Weights). Каждая строка данной матрицы (кроме нижней) соответствует одному из нейронов: первая строка – первому (входному) нейрону, вторая – второму входному нейрону, третья – первому нейрону скрытого слоя, четвертая – второму нейрону того же слоя, пятая – единственному выходному нейрону и шестая (последняя) – смещениям. Первый столбец матрицы соответствует первому из отмеченных нейронов, второй – второму и т. п. Элементы матрицы (кроме нижней строки) являются весами связей между нейронами. Так, в рассматриваемом примере видно, что первый входной нейрон соединен с третьим и четвертым (т. е. со

всеми нейронами скрытого слоя) с весами, соответственно, –4,3897 и –1,6988, третий нейрон соединен с выходным с весом –2,9008 и т. п. По данным весам можно судить о значимости отдельных входов и связей и попытаться оптимизировать структуру сети, о чем подробнее будет рассказано ниже.

8) Использовать обученную сеть достаточно просто. Подготовим два текстовых файла: первый, содержащий только наборы интересующих нас значений входов, для которых с помощью обученной сети требуется дать оценку выхода или выходов (с расширением *.dat), другой – пустой (с расширением *.log) для сохранения ответа сети, например, файл xortest.dat, с данными:

```
#
#
0.9  1.0
0.0  0.2
1.0  0.1
0.0  1.0
```

и файл xortest.log.

Запустим программу Slug.exe, откроем файл сохраненной и обученной сети (xor.ann) и оба подготовленных файла. После нажатия кнопки **Run** контрольная панель примет вид рис. 5.6.

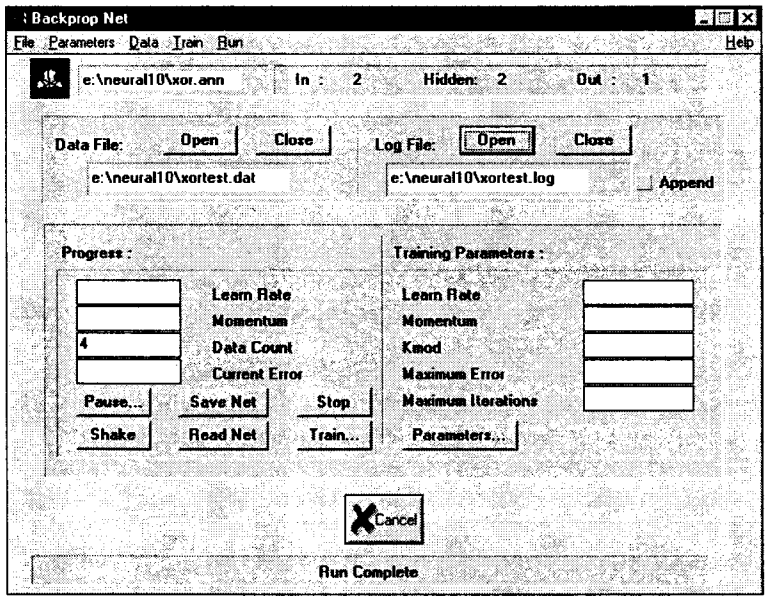


Рис. 5.6. Контрольная панель программы Slug.exe после опроса обученной сети

Что изменилось? В окошке Data Count появилась цифра 4 – по числу предъявленных входных наборов в файле xortest.dat, а также надпись Run Complete (выполнение завершено). Теперь необходимо закрыть оба файла и ознакомиться с результатами работы сети, сохраненными в файле xortest.log:

```
DATA MATRIX:
 0.9000  1.0000
 0.0000  0.2000
 1.0000  0.1000
 0.0000  1.0000

  0.90   1.00   0.055
  0.00   0.20   0.977
  1.00   0.10   0.881
  0.00   1.00   1.006
```

Желаемый ответ содержится в третьем столбце нижней матрицы. Комментарии, по-видимому, излишни.

Завершая описание работы с программой укажем на возможность «дозаписи» информации в уже имающийся (и заполненный) log-файл – с помощью кнопки **Append**. Это требуется в случае, когда имеется несколько файлов с данными, но желательно использовать один и тот же файл протокола.

Что можно отметить в заключение? Нейропакет крайне прост, но, тем не менее, позволяет решать достаточно сложные задачи с использованием нейросетевого подхода.

5.2. Нейропакет НейроПро (NeuroPro)

5.2.1. Общая характеристика

Программа NeuroPro является свободно распространяемой альфа-версией нейросетевого программного продукта для работы с искусственными нейронными сетями и извлечения знаний из таблиц данных с помощью нейронных сетей в среде Windows. Разработчик – В. Г. Царегородцев, Институт вычислительного моделирования СО РАН, 660036, Красноярск-36, Академгородок, e-mail: tsar@cc.krascience.rssi.ru.

Возможности программы:

1) Работа (чтение, запись, редактирование) с файлами данных, представленными в форматах *.dbf (СУБД dBase, FoxPro, Clipper) и *.db (СУБД Paradox).

Создание слоистых нейронных сетей для решения задач прогнозирования:

- число слоев нейронов – до 10;
- число нейронов в слое – до 100;
- нейроны: с нелинейной сигмоидной функцией активации $f(A) = A/(|A| + c)$, крутизна сигмоиды может задаваться отдельно для каждого слоя нейронов.

Нейронная сеть может одновременно решать несколько задач прогнозирования; для каждого из выходных сигналов могут быть установлены свои требования к точности прогнозирования.

2) Обучение нейронной сети производится по принципу двойственного функционирования с применением одного из следующих методов оптимизации:

- градиентного спуска;
- модифицированного ParTan-метода;
- метода сопряженных градиентов.

3) Тестирование нейронной сети.

4) Вычисление и вывод значимости входных сигналов сети.

5) Внесение случайных возмущений в веса синапсов сети.

6) Упрощение (контрастирование) нейронной сети:

- сокращение числа входных сигналов сети;
- сокращение числа синапсов сети;
- сокращение числа неоднородных входов (порогов) нейронов сети;
- равномерное прореживание структуры синапсов сети;
- бинаризация весов синапсов сети;
- генерация вербального описания нейронной сети.

Установка нейропакета производится после распаковки содержимого архива NeuroPro.zip на жесткий диск (доступен в Интернет по адресу: http://www.bmstu.ru/facult/iu/iu4/rus/stat/book4/NEUROPRO.ZIP). Для установки программы необходимо запустить программу-инсталлятор Setup.exe, после чего будет произведено копирование следующих файлов:

- NeuroPro.exe – исполняемый файл;
- NeuroPro.hlp – файл справки;
- NeuroPro.cnt – служебный файл для справки;
- Readme.doc – краткое описание программы;
- UserGuide.doc – руководство пользователя;
- Boltz.db – демонстрационный файл данных;
- Election.db – демонстрационный файл данных;
- Распространяемые файлы Borland Database Engine.

Применение данного программного продукта возможно в традиционных областях, а именно, в медицине, экологии, климатологии, метеорологии, при построении моделей технических объектов и их идентификации, в экономике (прогнозирование курсов валют, акций и т. д.) и вообще, для решения любой задачи классификации или прогноза, которая решается при наличии выборки данных и для решения которой ранее использовались традиционные математические методы (регрессионный анализ, непараметрическая статистика), однако не была достигнута требуемая точность прогноза.

5.2.2. Главное меню

Меню программы содержит следующие пункты, относящиеся к нейронным сетям и работе с ними (рис. 5.7):

Файл – базовые операции с файлами, включая стандартные опции: создать, открыть, сохранить, сохранить как, выход.

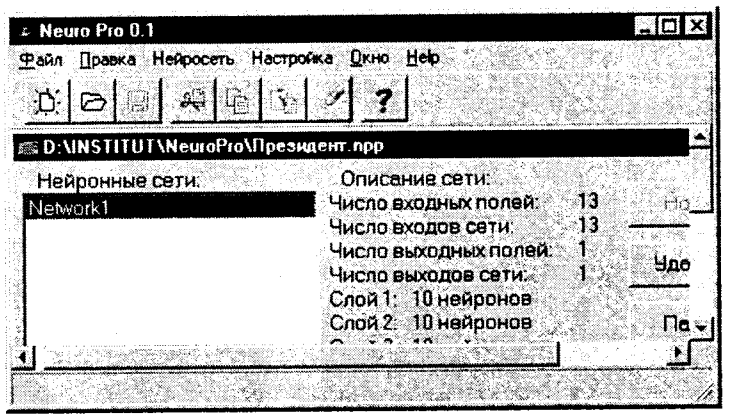


Рис. 5.7. Интерфейс программы NeuroPro

Нейросеть – операции с нейронными сетями. Операция выполняется над активной в данный момент в нейропроекте нейронной сетью (рис. 5.8) и включает в себя следующие опции:

- **Обучение** – обучение нейронной сети;
- **Тестирование** – тестирование нейронной сети;
- **Сокращение числа входных сигналов** – удаление наименее значимых входных сигналов;
- **Сокращение числа синапсов** – удаление наименее значимых синапсов сети;

- **Сокращение числа неоднородных входов** – удаление наименее значимых неоднородных входов нейронов сети;
- **Равномерное упрощение сети** – сокращение числа приходящих на нейроны сети сигналов до заданного пользователем;
- **Бинаризация синапсов сети** – приведение значений весов синапсов и неоднородных входов нейронов к значениям –1 и 1;
- **Вербализация** – генерация вербального описания нейронной сети;
- **Значимость входов** – подсчет и отображение значимости входных сигналов нейронной сети;
- **Возмущение весов синапсов** – добавление случайных поправок к весам синапсов сети.

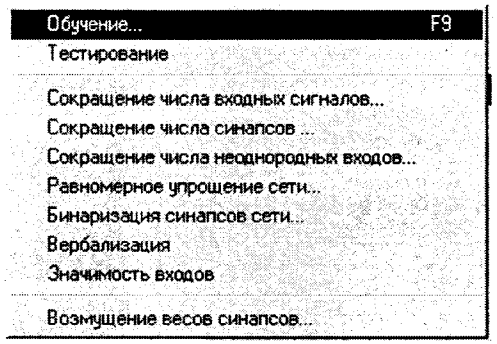


Рис. 5.8. Операции пункта **Нейросеть** главного меню

Настройка – операции по настройке нейронной сети. Настройки действуют в пределах нейропроекта, сохраняются в файле нейропроекта и восстанавливаются при его чтении программой (рис. 5.9 и рис. 5.10).

Опции данного пункта меню:

- **Метод оптимизации** – выбор метода оптимизации для обучения сети. Из трех реализованных в настоящее время в программе методов (градиентный спуск, модифицированный партан-метод (ParTan) и метод сопряженных градиентов) при создании нейропроекта автоматически предлагается ParTan (см. прил. 3).
- **Норма накопления значимости** – выбор нормы накопления градиента при подсчете показателей значимости, иначе говоря, показатель совокупной ошибки сети. При создании нейропроекта автоматически выбирается норма в виде суммы модулей.

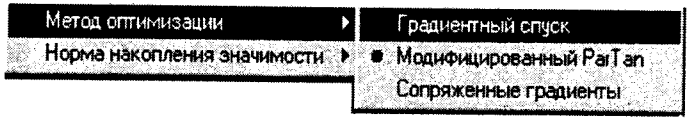


Рис. 5.9 Опции пункта **Настройка** главного меню

Рис. 5.10. Опции пункта **Настройка** главного меню

5.2.3. Создание нейропроекта

Работа с нейронными сетями возможна только в рамках некоторого нейропроекта. Для того, чтобы создать нейропроект, необходимо выбрать пункт меню **Файл/Создать** или нажать кнопку **Создать** на панели. После создания нейропроекта в него можно вставлять нейронные сети и работать с ними. Созданный нейропроект может быть сохранен при помощи команд меню **Файл/Сохранить**, **Файл/Сохранить как**, или нажатием кнопки **Сохранить** (рис. 5.11).

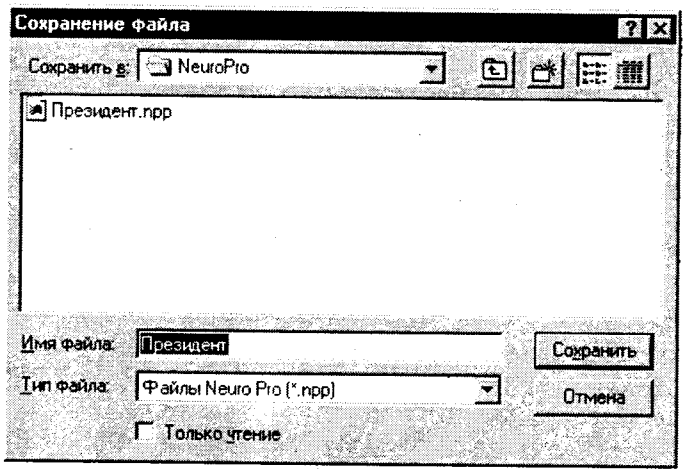


Рис. 5.11. Окно сохранения нейропроекта

В дальнейшем возможна работа с сохраненными файлами нейропроекта. Для этого необходимо выбрать пункт меню

Файл/Открыть или нажать кнопку **Открыть** и далее выбрать в диалоговом окне имя нужного нейропроекта (рис. 5.12).

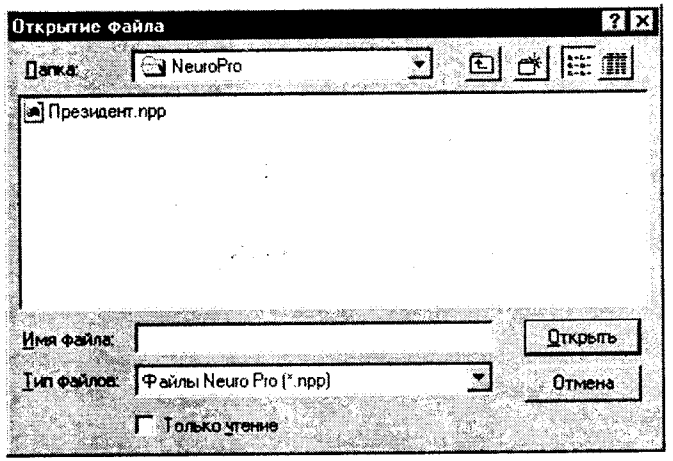


Рис. 5.12. Окно открытия нейропроекта

Большинство операций с нейронными сетями требуют присутствия подключенного к нейропроекту файла данных. Для подключения файла данных или его замены необходимо открыть файл данных в окне нейропроекта или выбрать имя необходимого файла данных. Открытый файл данных отображается в собственном окне, где предоставляется возможность его редактирования. При закрытии окна файла данных подключение к нейропроекту завершается.

При подключенном файле данных можно проводить операции создания новых нейронных сетей, их обучения, тестирования и упрощения.

5.2.4. Создание нейронной сети

Для создания новой нейронной сети необходимо нажать кнопку **Новая сеть** в окне нейропроекта и заполнить **Диалог создания нейронной сети**.

Диалог создания нейронной сети предназначен для задания спецификаций для создаваемой нейронной сети. Элементы диалога (рис. 5.13):

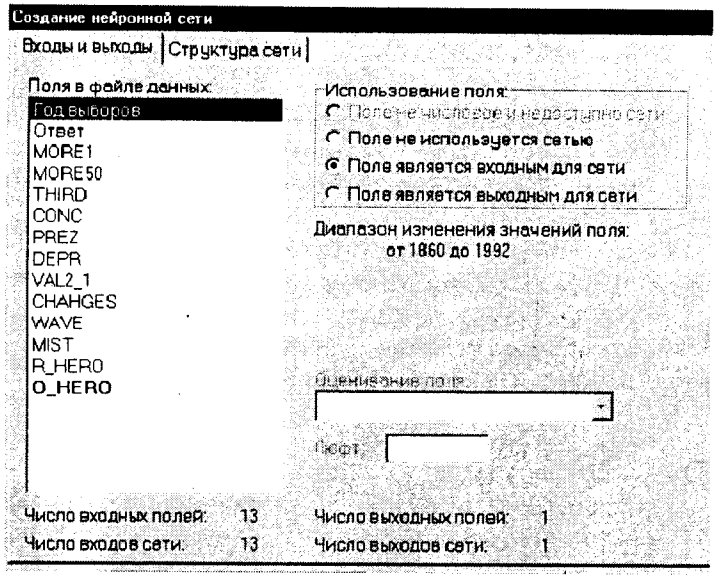


Рис. 5.13. Окно создания нейронной сети

Входы и выходы – окно для определения использования нейронной сетью имеющихся в файле данных полей.

Поля в файле данных – список полей в файле данных.

Использование поля – использование текущего поля нейронной сетью:

- поле не числовое и недоступно сети – поле не является числовым и не может обрабатываться нейронной сетью;
- поле не используется сетью – данное числовое поле не используется сетью;
- поле является входным для сети – значения данного поля подаются на входы сети;
- поле является выходным для сети – нейронная сеть обучается прогнозировать значения этого поля.

Диапазон изменения значений поля – минимальное и максимальное значение поля в файле данных.

Оценивание поля – способ оценивания выходного поля при обучении сети:

- МНК – используется оценка МНК;
- МНК с люфтом – используется оценка МНК вида:

$$H = \sum_i P\left(\frac{Y_i - Y_i'}{\varepsilon}\right),$$

где

$$P(\Delta)=\begin{cases}(|\Delta|-1)^2, & \text{если } |\Delta|\geq 1,\\ 0, & \text{если } |\Delta|<1,\end{cases}$$

Y_i и Y_i' – соответственно, выход по обучающей выборке и выход нейронной, ε – параметр алгоритма (люфт).

Люфт может изменяться от 0 до границ диапазона изменения значений этого поля. По умолчанию люфт определяется в 10% от диапазона, при этом сеть должна обучиться предсказывать значения данного поля с точностью ±10% от диапазона изменения значений. Чем меньше величина люфта, тем более точно должна сеть предсказывать известные значения.

Параметры, автоматически определяемые программой:

- **Число входных полей** – число полей в файле данных, используемых сетью в качестве входных.
- **Число входов сети** – число входных сигналов сети.
- **Число выходных полей** – число полей в файле данных, используемых сетью в качестве выходных.
- **Число выходов сети** – число выходных сигналов сети.
- **Структура сети** – окно для задания структуры нейронной сети (рис. 5.14). Характеризуется следующими параметрами.

Рис. 5.14. Окно **Структура сети**

- **Число слоев нейронов** – число слоев нейронов в сети. Изменяется от 1 до 10. Дополнительно после последнего слоя нейронов создается слой выходных сумматоров с числом сумматоров, равным числу выходных сигналов сети. По умолчанию предлагается 3 слоя нейронов. Для каждого слоя нейронов возможно задание ниже рассмотренных характеристик.
- **Число нейронов** – число нейронов в слое. Изменяется от 1 до 100. По умолчанию предлагается 10 нейронов в слое.
- **Нелинейность** – вид функции активации нейронов данного слоя. На данный момент реализована только сигмоидная нелинейность вида $f(A)=A/(c+|A|)$, где c – характеристика нейрона.
- **Характеристика** – значение не обучаемой константы c, используемой нелинейным преобразователем. Может изменяться в диапазоне от 0,0001 до 1 для описанной выше сигмоидной нелинейности. По умолчанию предлагается значение характеристики, равное 0,1. Чем больше значение характеристики, тем лучше интерполяционные и экстраполяционные способности обученной сети, но, как правило, это требует более длительного обучения.

За последним слоем нейронов строится слой выходных сумматоров по числу выходных сигналов сети.

- **Монотонность** – создание сети монотонной структуры.
- **Имя сети** – имя нейронной сети в списке нейропроекта.

Опции **Создать/Изменение** и **Отменить** позволяют соответственно создать нейронную сеть или внести изменения в ее параметры, или же отменить эти действия.

После создания нейронной сети она появляется в списке сетей нейропроекта и становится активной. Созданную нейронную сеть можно далее обучать, тестировать, упрощать и сохранять на диске вместе с нейропроектом.

5.2.5. Обучение нейронной сети

Для обучения активной в данный момент нейронной сети необходимо выбрать пункт меню **Нейросеть/Обучение**. Если подключенный к нейропроекту файл данных не содержит необходимых полей (это возможно, когда сеть создается по одному файлу данных, а дальнейшее ее обучение, тестирование или упрощение – по данным из другого файла), то выдается сообщение о несовместимости нейронной сети и файла данных. Если же в файле данных имеются все необходимые поля и он не пустой, то запускается процесс обучения сети. При этом на экран выводится **Окно обучения и упрощения сети**, где имеется возможность наблюдать процесс обучения и при необходимости самостоятельно за-

вершить обучение нажатием кнопки **Завершить**, заменяющейся в случае удачного обучения кнопкой **Готово** (рис. 5.15).

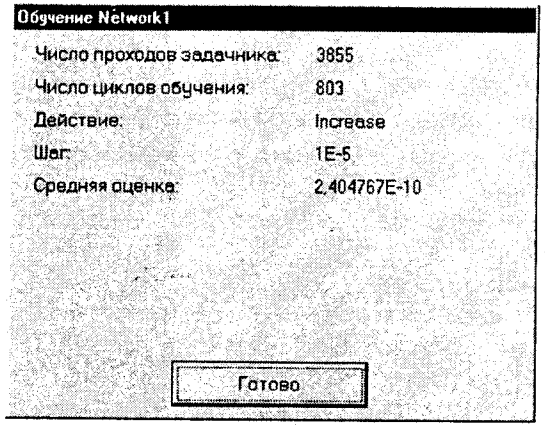


Рис. 5.15. Окно обучения нейронной сети

Обучение прекращается при достижении нулевого значения средней оценки на задачнике, в случае невозможности дальнейшего улучшения оценки либо при аварийных ситуациях (нулевой или бесконечный шаг в направлении оптимизации).

Окно обучения и упрощения сети отображает следующие характеристики (рис. 5.16):

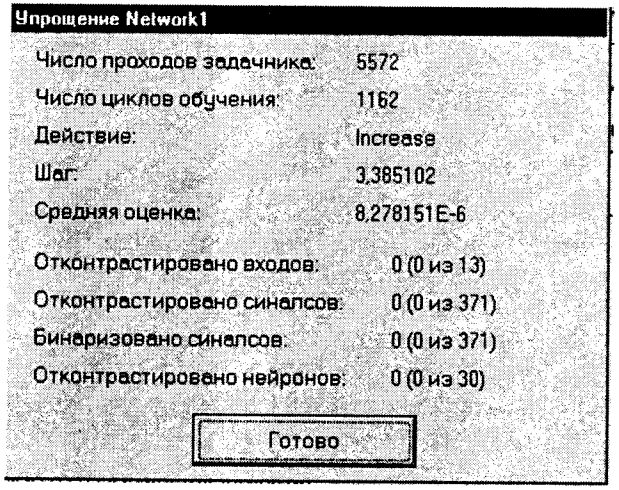


Рис. 5.16. Окно упрощения нейронной сети

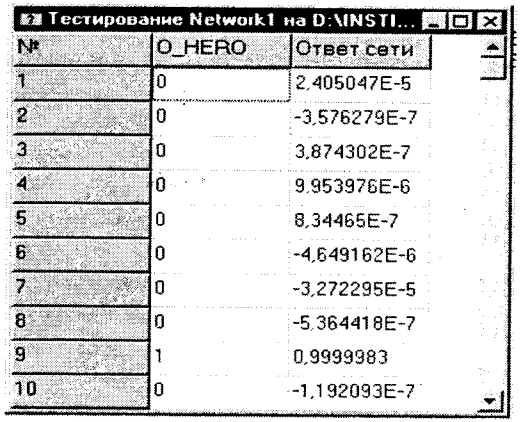
- **Число проходов задачника** – общее число просмотров обучающего множества.
- **Число циклов обучения** – общее число шагов обучения (шагов градиентного спуска, ParTan-шагов или шагов метода сопряженных градиентов).
- **Действие** – текущее действие.
- **Градиент** – вычисление градиента функции оценки.
- **Increase** – увеличение шага в направлении оптимизации.
- **Decrease** – уменьшение шага в направлении оптимизации.
- **Zero step** – нулевой шаг в направлении оптимизации, прекращает процесс обучения или контрастирования сети.
- **Zero divide** – нулевой делитель при параболической аппроксимации, прекращает процесс обучения или контрастирования сети.
- **Шаг** – величина шага в направлении оптимизации.
- **Средняя оценка** – средняя оценка на обучающем множестве.
- **Отконтрастировано входов** – число отконтрастированных входных сигналов сети. Здесь и далее информация выводится в виде *X*(*Y* из *Z*), где *X* – число отконтрастированных входных сигналов (синапсов, нейронов) на текущем этапе упрощения, *Y* – общее число отконтрастированных входных сигналов (синапсов, нейронов) у сети, *Z* – число имеющихся у сети входных сигналов (синапсов, нейронов).
- **Отконтрастировано синапсов** – число отконтрастированных синапсов и неоднородных входов нейронов сети.
- **Бинаризовано** синапсов – число бинаризованных синапсов и неоднородных входов нейронов сети.
- **Отконтрастировано нейронов** – число отконтрастированных в сети нейронов.

Завершить/Готово – кнопка принудительного завершения или подтверждения прекращения процесса обучения, или упрощения сети.

5.2.6. Тестирование нейронной сети

После обучения нейронной сети можно провести тестирование ее прогностических возможностей. Для этого нужно выбрать пункт меню **Нейросеть/Тестирование**. Результат тестирования сети выводится в **Окно тестирования сети** (рис. 5.17) и представ-

ляет собой выходные данные для нейронной сети, а также значения прогноза этих полей нейронной сетью. Если для какой-либо строки в файле неизвестно значение выходного поля, то для этой записи будет выведен только прогноз сети. Если у строки отсутствует хотя бы одно входное поле, прогноз сети будет отсутствовать.



N#	O_HERO	Ответ сети
1	0	2,405047E-5
2	0	-3,576279E-7
3	0	3,874302E-7
4	0	9,953976E-6
5	0	8,34465E-7
6	0	-4,649162E-6
7	0	-3,272295E-5
8	0	-5,364418E-7
9	1	0,9999983
10	0	-1,192093E-7

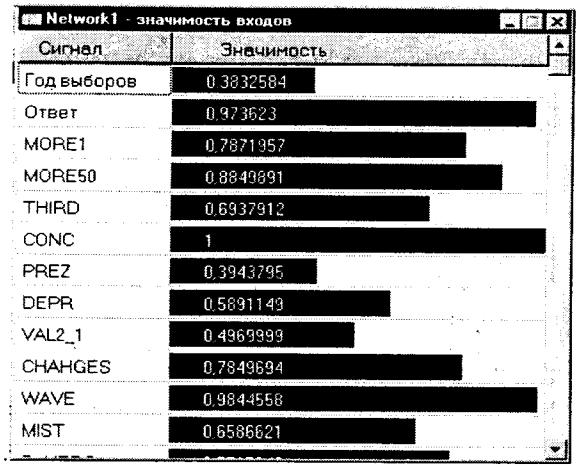
Рис. 5.17. Окно тестирования нейронной сети

5.2.7. Вычисление показателей значимости входных сигналов сети

Для вычисления показателей значимости входных сигналов нейронной сети необходимо выбрать пункт меню **Нейросеть/Значимость входов**. Вычисленные показатели значимости выводятся в **Окно значимости входов** (рис. 5.18).

Данное окно отображает текущую значимость входных сигналов для принятия сетью правильного решения. Значимость (в относительных единицах) отображается в виде гистограммы, где наиболее значимому входу соответствует наиболее длинный столбец гистограммы. Для отконтрастированных входных сигналов значимость равна 0.

Если отмеченное окно отображено на экране, и для данной нейронной сети запускается процесс упрощения, то это окно динамически изменяет отображаемые данные после каждого пересчета показателей значимости входных сигналов (на каждом шаге упрощения, т. е. при исключении входного сигнала или контрастировании/бинаризации синапса).



Сигнал	Значимость
Год выборов	0,3832584
Ответ	0,973623
MORE1	0,7871957
MORE50	0,8849891
THIRD	0,6937912
CONC	1
PREZ	0,3943795
DEPR	0,5891149
VAL2_1	0,4969999
CHAHGES	0,7849694
WAVE	0,9844558
MIST	0,6586621

Рис. 5.18. Окно значимости входов нейронной сети

5.2.8. Упрощение нейронной сети

Не все входные сигналы сети и синапсы нейронов необходимы для правильного решения сетью задачи. Часто можно достаточно существенно упростить сеть без ухудшения точности решения задачи. Основными результатами проведения процесса упрощения сети являются следующие:

- сокращается число входных сигналов сети. Если правильно решить задачу можно на основе меньшего набора входных данных, то это может в дальнейшем сократить временные и материальные затраты на сбор информации;
- нейронную сеть более просто можно будет реализовать на аппаратной платформе;
- сеть может приобрести логически прозрачную структуру. Известно, что почти невозможно понять, как обученная нейронная сеть решает задачу. После упрощения нейронная сеть становится достаточно обозримой, и можно попытаться построить алгоритм решения сетью задачи на основе графического представления или вербального описания структуры сети.

Для упрощения нейронной сети имеются следующие операции в меню **Нейросеть**:

- **Сокращение числа входных сигналов** – удаление наименее значимых входных сигналов.
- **Сокращение числа синапсов** – удаление наименее значимых синапсов сети.

- **Сокращение числа неоднородных входов** – удаление наименее значимых неоднородных входов нейронов сети.
- **Равномерное упрощение сети** – сокращение числа приходящих на нейроны сети сигналов до задаваемого пользователем.
- **Бинаризация синапсов сети** – приведение значений весов синапсов и неоднородных входов нейронов к значениям –1 и 1.

Упрощение нейронной сети проводится до тех пор, пока возможно обучение нейронной сети до нулевой средней оценки. Текущая информация выводится в **Окно обучения и упрощения сети**. Упрощение может прекратиться, когда уже все синапсы, подлежащие контрастированию или бинаризации, соответственно отконтрастированы или бинаризованы.

Нейрон сети считается отконтрастированным, когда у него нет ни одного входного сигнала или сигнал данного нейрона не используется нейронами следующего слоя. Поэтому нейроны сети можно контрастировать, не вводя специальной операции, а пользуясь только контрастированием сигналов и синапсов.

Если пользователя не удовлетворяет число отконтрастированных входов/синапсов, то можно попробовать дообучить сеть с более сильными требованиями к точности решения задачи и, вернувшись к прежней точности, запустить процесс упрощения вновь. Часто это помогает отконтрастировать большее число нейронов.

Как показывает опыт, простое сокращение числа синапсов может удалять как наименее значимые входы, так и «лишние» нейроны сети. Однако для более простого понимания решения задачи часто бывает необходимо упрощать сеть по более сложным правилам, в первую очередь, равномерно прореживая структуру синапсов сети, а только потом удалять наименее значимые входы и нейроны сети.

Диалог равномерного упрощения сети содержит следующие элементы:

- **Максимальное число сигналов на нейрон** – максимальное число сигналов, приходящих на нейрон. У каждого нейрона сети последовательно будут контрастироваться наименее значимые синапсы до тех пор, пока число оставшихся у нейрона синапсов не будет превышать заданного числа. По умолчанию максимальное число сигналов на нейрон принимается равным 3.
- **Упрощение** – запускает процесс равномерного упрощения сети с заданным максимальным числом сигналов.
- **Отменить** – отменяет запуск процесса упрощения.

5.2.9. Вербализация нейронной сети

Для получения вербального описания текущей нейронной сети необходимо выбрать пункт меню **Нейросеть/Вербализация**. Вербальное описание сети выводится в **Окно вербального описания сети** (рис. 5.19). На основе вербального описания можно попытаться восстановить набор правил, используемых сетью для правильного решения задачи.

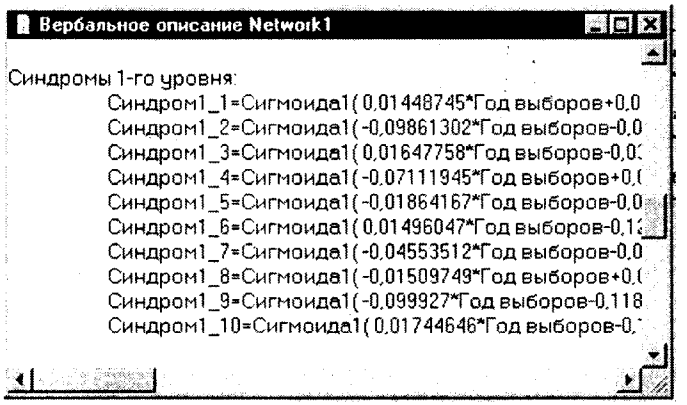


Рис. 5.19. Окно вербального описания нейронной сети

Окно вербального описания сети содержит текст, описывающий нейронную сеть. Отконтрастированные входные сигналы, синапсы и нейроны сети в тексте не описываются. В текст включены следующие разделы:

- поля базы данных БД (исходные симптомы) – имена полей файла данных, которые используются сетью в качестве входных;
- поля базы данных (конечные синдромы) – имена полей БД, значения которых прогнозирует нейронная сеть;
- предобработка входных полей БД для подачи сети – правила нормирования входных сигналов в диапазон [–1, 1] для подачи нейронной сети;
- функциональные преобразователи – описания используемых на каждом слое сети функциональных преобразователей нейронов;
- синдромы первого уровня – описание преобразования входных сигналов сети нейронами первого слоя сети;

• синдромы второго уровня – описание преобразования входных сигналов сети нейронами второго слоя сети (если у сети два и более слоев нейронов);

• конечные синдромы – описание вычисления значений прогнозируемых полей файла данных;

• постобработка конечных синдромов – правила нормирования выходных сигналов сети из диапазона [–1, 1] в диапазон истинных значений.

Данное вербальное описание нейронной сети может быть сохранено на диске в текстовом файле.

5.2.10. Правила работы с нейропакетом

Рассмотрим теперь правила работы с нейропакетом на примере задачи «Исключающее ИЛИ». Последовательность действий при этом такова.

1) Подготовка исходных данных (обучающей выборки, задачника). Ранее отмечалось, что пакет NeuroPro использует данные, представленные в виде электронных таблиц форматов *.dbf (СУБД dBase, FoxPro, Clipper) и *.db (СУБД Paradox). А что делать, если отмеченные СУБД на компьютере пользователя не установлены? Срочно устанавливать? Необходимости в этом нет при наличии Excel. Подготовим исходные данные в Excel, снабдив все столбцы соответствующими именами (рис. 5.20).

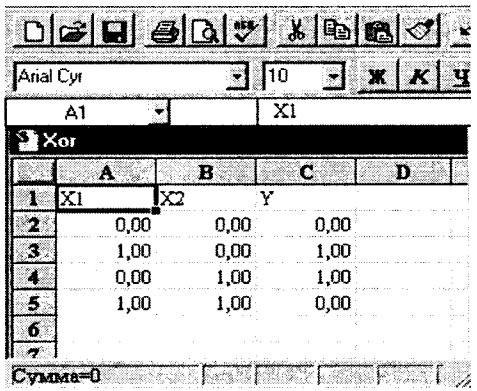


Рис. 5.20. Таблица с данными обучающей выборки

Теперь выделим все ячейки, укажем их формат как числовой, например, с двумя знаками после запятой и после этого сохраним таблицу, используя опцию меню **Файл/Сохранить как**, в формате dBase, например, под именем Xor.dbf.

2) Создание проекта. Запустим нейропакет. В появившемся окне программы с помощью опции меню **Файл/Создать** перейдем к окну вида рис. 5.21.

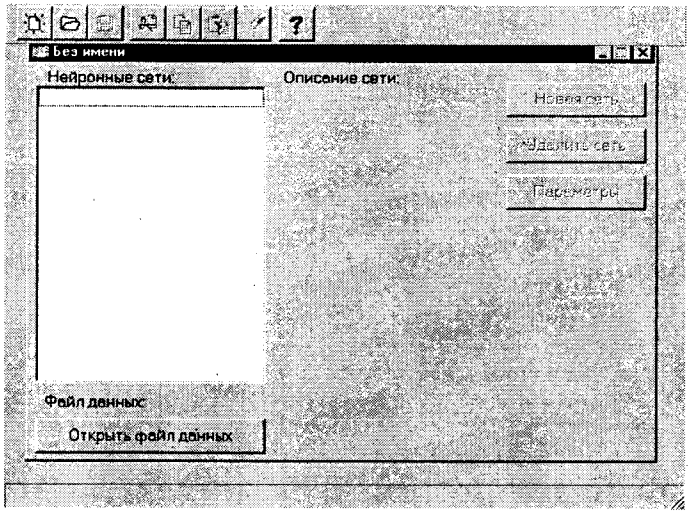


Рис. 5.21. Окно создание новой нейронной сети

Нажмем кнопку **Открыть файл данных** и выберем файлы с расширением *.dbf и среди них – файл Xor.dbf. Откроем его, при этом открытые окна приобретают вид рис. 5.22.

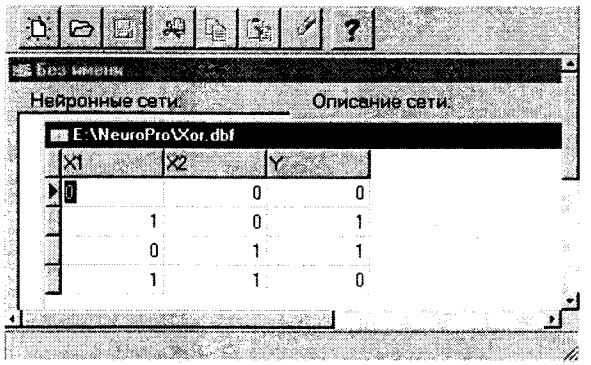


Рис. 5.22. Вид окон нейропакета при подключении файла данных

Нажмем кнопку **Новая сеть**, подтвердим, что *X*1 и *X*2 – входы, а *Y* – выход сети, выберем МНК для оценивания поля, присвоим новой сети имя Xor и нажмем кнопку **Структура сети**. В появившемся окне укажем число слоев нейронов – 1, число нейронов в слое – 2, после чего нажмем кнопку **Создать**. Окно программы примет вид рис. 5.23.

3) Обучение нейронной сети. Теперь с помощью опции меню **Нейросеть/Обучение** или путем нажатия клавиши F9 запустим режим обучения нейронной сети. По его окончании увидим окно, аналогичное показанному на рис. 5.15. Нажмем кнопку **Готово**.

4) Тестирование нейронной сети. Через пункты меню **Нейросеть/Тестирование** запустим режим тестирования. Результаты в появляющемся окне (рис. 5.24) говорят о достаточно приемлемом качестве работы нейронной сети.

Можно теперь делать что угодно: выяснять значимость входов, пытаться упростить сеть, сохранить ее и т. п. Но можно перейти и к режиму ее использования, все же вначале сохранив обученную сеть (например, под именем Xor.npp).

5) Использование обученной сети. Подготовим какой-либо набор входных данных, также используя Excel; столбец для выходных данных снабдим только названием, но заполнять не будем (рис. 5.25). Сохраним файл под именем Xortest в формате dbf.

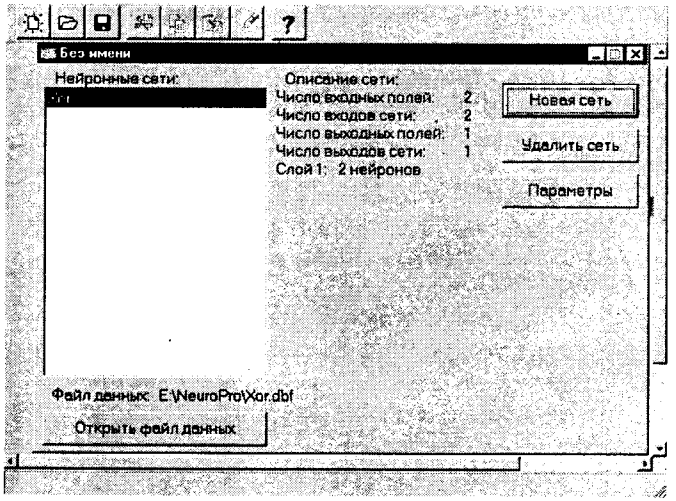


Рис. 5.23. Вид окна программы после создания структуры нейронной сети

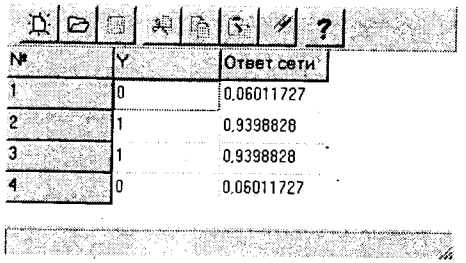


Рис. 5.24. Результаты тестирования обученной нейронной сети

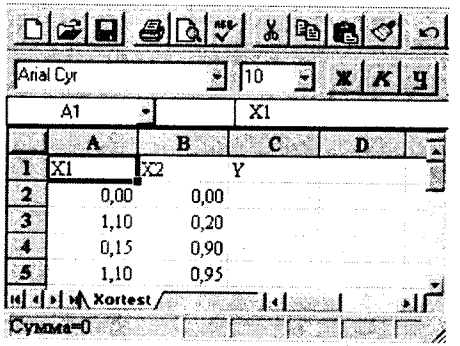


Рис. 5.25. Данные для опроса обученной нейронной сети

Запустим, далее, нейропакет и откроем (как описано выше) сохраненный нейропроект. В появившемся окне нажмем кнопку **Открыть файл данных** и далее откроем файл Xortest.dbf. Перейдем в окно **Нейронные сети** и в меню выберем пункт **Нейросеть/Тестирование**. Полученный результат приведен на рис. 5.26.

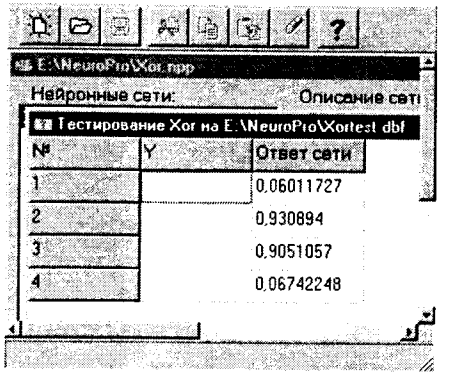


Рис. 5.26. Результаты опроса нейронной сети

Как видно, данные результаты также можно считать удовлетворительными по точности.

5.2.11. Общее суждение о нейропакете

Рассмотренный нейропакет несомненно удобен для изучения и использования по следующим причинам:

- является русскоязычным;
- прост в изучении;
- имеет такие достоинства, как возможности упрощения сети и выявление наиболее (или наименее) значимых входов.

В то же время есть и ряд недостатков, из которых наиболее существенным представляется (по крайней мере, в имеющейся альфа-версии) невозможность сохранения результатов опроса обученной сети. К другим недостаткам можно отнести скромные интерфейсные и сервисные удобства.

5.3. Нейропакет QwikNet32

Нейропакет QwikNet32 разработан C. Jensen (9935 NE 125th Ln #4, Kirkland, WA 98034, e-mail: cjensen@kagi.com); trial-версия (версия для ознакомления с ограниченным сроком использования) пакета доступна в Интернет по адресу: http://www.simtel.net/simtel.net/win95/neural-pre.html.

5.3.1. Общая характеристика и интерфейс

Нейропакет QwikNet32 (версия 2.1) предназначен для работы в среде Windows и особых требований к вычислительным ресурсам не предъявляет. В QwikNet реализуется лишь один тип нейронной сети – многослойная сеть прямого распространения с числом скрытых слоев до 5 и с набором из 6 алгоритмов обучения (модификации алгоритма обратного распространения ошибки). Вид контрольной панели программы после ее запуска приведен на рис. 5.27.

В верхней ее части расположено меню, включающее следующие пункты:

File (Файл) с подпунктами: **New Network** (Новая сеть); **Open Network** (Открыть сеть); **Save Network** (Сохранить сеть); **Save Network as ...** (Сохранить сеть как ...) – сохранение структуры сети но не ее весов; **Write network as C code** (Записать сеть в кодах языка C); **Exit** (Выход).

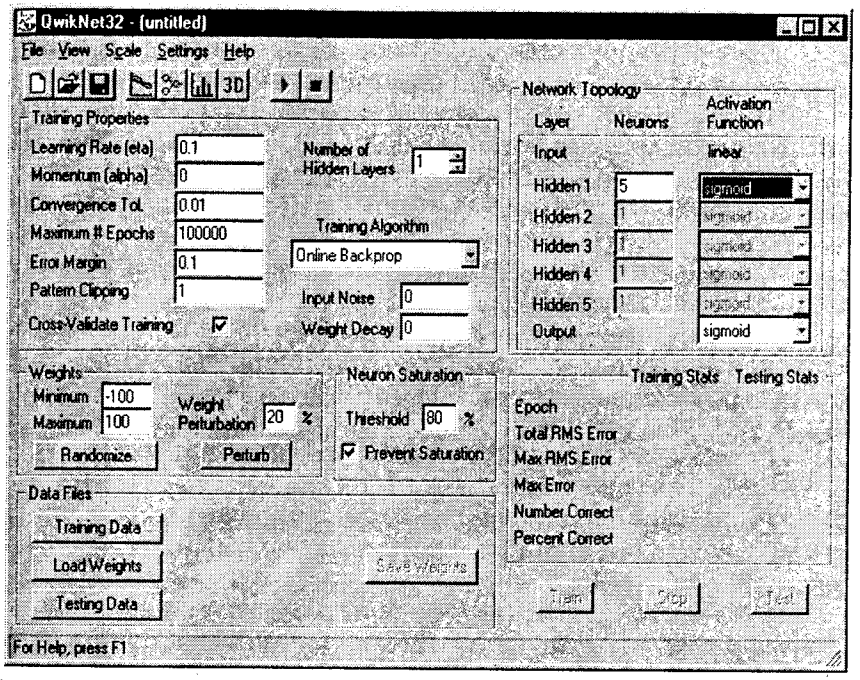


Рис. 5.27. Контрольная панель нейропакета QwikNet

View (Вид) с подпунктами:

- **Network** (Сеть) – графическое изображение созданной сети;
- **Training Error Plot** (График ошибки обучения) – график, показывающий изменение общей ошибки нейронной сети в процессе обучения;
- **Contour Plot** (Контурный график) – трехмерный график, отображающий выход сети в зависимости от двух выбранных входов – для нормированных значений;
- **Network Analysis Plot** (График анализа сети) – гистограмма, показывающая уровни средних ошибок по каждому образцу обучающей или тестовой последовательности;
- **Training History** (История обучения) – протокол обучения сети – текстовая информация.

Scale (Масштабирование) имеет единственную опцию **Scale Data File** (Масштабировать файл данных) – используется для масштабирования данных (см. ниже).

Settings (Установочные параметры) также имеет только одну опцию **Advanced Settings** (Расширенные установочные параметры), позволяющую задать параметры процесса обучения.

Help (Справка) с подпунктами: **Help** (Справка); **Registration Info** (Информация о регистрации программы); **Register QwikNet** (Регистрация QwikNet) – для регистрации легальной копии нейропакета; **About** (О программе) – информация о программе и вычислительных ресурсах компьютера.

Ряд подпунктов меню дублируется кнопками в левой верхней части контрольной панели.

Правее от отмеченных расположены еще две кнопки – **Train Network** (Запуск обучения) и **Stop Training Network** (Принудительное прекращение обучения). Эти кнопки продублированы в правой нижней части панели (кнопки **Train** и **Stop**). Справа от них расположена кнопка **Test** (Запуск тестирования сети).

Назначение других кнопок:

- **Training Data** и **Testing Data** позволяют подключать к программе (загружать) файлы с обучающей и тестовой последовательностями данных;
- **Load Weights** и **Save Weights** позволяют загрузить и сохранить в текстовом файле (например, для последующего анализа) веса и смещения обученной сети;
- **Randomize** производит установку начальных (случайных) значений весов сети перед ее обучением;
- **Perturb** позволяет случайным образом варьировать веса сети в процессе ее обучения и, тем самым, избегать останова процесса в точках локального, а не глобального минимума ошибки.

Диалоговые мини-окна контрольной панели (рис. 5.27) позволяют задать структуру нейронной сети, выбрать алгоритм и параметры ее обучения.

В окне **Number of Hidden Layers** (Число скрытых слоев) задается число скрытых слоев (до пяти) сети. В правой части контрольной панели – **Network Topology** (Топология сети) – для каждого скрытого слоя отдельно указывается число нейронов в нем (окна **Hidden 1 – Hidden 5/ Neurons**) и вид функции активации (окна **Hidden 1 – Hidden 5/ Activation Function**). В нейропакете реализован следующий набор функций активации:

- sigmoid (сигмоидная),
- tanh (гиперболический тангенс),
- linear (линейная),
- Gaussian (функция Гаусса).

В центральной части панели диалоговое окно **Training Algorithm** (Алгоритм обучения) позволяет выбрать один из шести алгоритмов обучения нейронной сети:

- Online Backprop;
- Online Backprop-Rand;
- Batch Backprop;
- Delta-Bar-Delta;
- RPROP;
- QuickProp;

описание которых можно найти в прил. 3. По умолчанию устанавливается алгоритм Online Backprop – алгоритм обучения по методу обратного распространения ошибки в режиме реального времени – модификация алгоритма обучения по методу обратного распространения ошибки, когда веса и смещения сети корректируются после предъявления каждого нового образа (вектора) обучающей выборки.

Группа диалоговых окон в левой части панели **Training Properties (Опции обучения)** позволяют установить следующие параметры алгоритма обучения:

- **Learning Rate** – коэффициент скорости обучения (темп обучения), который определяет скорость изменений величин весов и смещений сети в процессе ее обучения, обычно при использовании алгоритма обратного распространения ошибки. Чем больше темп обучения, тем быстрее обучается сеть. Допустимые значения параметра – от 0,0 до 1,0; хорошим начальным приближением считается величина 0,1. Если данный параметр велик, процесс обучения может потерять устойчивость.
- **Momentum** – коэффициент импульса – константа, используемая в методе импульса (по умолчанию – 0).
- **Convergence Tol.** (convergence tolerance) – это максимально допустимая величина ошибки во время обучения. Если tolerance = 0, то выходной результат (output), выдаваемый нейронной сетью, должен абсолютно точно совпадать с образцом для обучения (pattern). В большинстве случаев это нереально. При tolerance = 0,1 значение выхода output будет рассматриваться как корректное, если оно отличается не более чем на 10% (в среднем квадратическом смысле) от заданного значения (pattern). Процесс обучения будет продолжаться до тех пор, пока значение ошибки не снизится до установленного параметром tolerance предела (по умолчанию – 0,01).
- **Maximum # Epoch** (maximum number of epochs) – максимальное число периодов, которые нужно использовать для обуче-

ния (по умолчанию 10000). Один период эквивалентен одному полному представлению всех образцов обучающей выборки.

- **Error Margin** – граница ошибки (по умолчанию – 0,01). Выходы сети, для которых ошибка меньше данной величины будут считаться «обученными», корректными.
- **Pattern Clipping** – «отсечение» образцов – определяет степень участия «обученных» образцов в последующем обучении сети. Образец считается «обученным», когда связанная с ним ошибка сети менее заданной в опции Error Margin. Значение параметра от 0,0 до 1,0. По умолчанию – 1.
- **Cross-Validate Training** (обучение с перекрестным пересечением) – когда эта опция установлена, обучающий набор данных автоматически делится на два набора: 90 % – для обучения и 10% – для тестирования. Набор для тестирования используется для проверки качества обучения сети.

Два последних мини-окна в рассматриваемой части контрольной панели **Input Noise** (Входной шум) и **Weight Decay** (Разрушение веса) используются для задания параметров, в ряде случаев улучшающих свойства обученной сети (свойства обобщения); задание первого ненулевого параметра обеспечивает добавление гауссовского шума небольшой интенсивности к данным обучающей выборки, задание второго ограничивает большие весовые коэффициенты.

Диалоговые окна в части контрольной панели **Weights** (Веса) относятся к ранее описанным кнопкам **Randomize** и **Perturb** и позволяют задавать:

- диапазон случайных вариаций начальных значений весов сети в процентах от некоторой автоматически определяемой программой величины, связанной с количеством входов нейрона (по умолчанию – 100%);
- диапазон вариации весов сети при повторном обучении.

Диалоговые окна в части панели **Neuron Saturation** (Насыщение нейрона) позволяют устранить насыщение нейронов в процессе обучения сети (у насыщенного нейрона сумма входов, умноженных на веса такова, что значение аргумента активационной функции велико и работа проходит на пологом, практически горизонтальном участке данной функции, так что большие изменения входов приводят к незначительному изменению выхода нейрона, т. е. нейрон становится малочувствительным к входным сигналам).

Окно **Threshold** (Порог) определяет минимальный процент примеров за эпоху (по умолчанию – 80%), которые должны насытить нейрон. Нейрон считается насыщенным, когда данный процент примеров обеспечивает выход не менее, чем 99% от максимально возможной величины. Если это происходит, веса входов нейрона уменьшаются на 90%.

Окно **Prevent Saturation** позволяет установить или убрать режим автоматического предотвращения насыщения нейронов.

Сообщения в правой нижней части контрольной панели **Training Stats** (Статистика обучения) и **Testing Stats** (Статистика тестирования) отображают информацию во время работы программы в соответствующих режимах.

Заметим, что по всем элементам интерфейса и режимам работы нейропакета имеются достаточно информативные и удобно организованные разделы справки на английском языке.

5.3.2. Правила работы с нейропакетом

Работу с нейропакетом рассмотрим на примере все той же задачи моделирования логической функции «Исключающее ИЛИ».

1) Подготовка исходных данных. Исходные данные готовятся в формате табл. 4.1 и сохраняются в виде текстового файла (разделители – пробелы или табуляции). Файл для обучения нейронной сети имеет расширение *.prn и следующее содержание:

```
* Комментарий 1
* ...
* Комментарий n
[INPUTS] 2
[OUTPUTS] 1
0 0 0
0 1 1
1 0 1
1 1 0
```

Несколько первых строк, начинающихся звездочкой, отводятся под комментарии; далее находится строка, указывающее число входов – [INPUTS] 2, затем – строка, указывающее число выходов – [OUTPUTS] 1. Вообще-то эти параметры можно и не указывать, они все равно будут запрошены программой.

В нашем примере как входы, так и выход изменяются от нуля до единицы, но в общем случае диапазоны изменений могут быть произвольными, в связи с чем для работы с данным нейропакетом рекомендуется сначала провести масштабирование данных. Дело в том, что активационные функции чаще всего имеют ограниченный диапазон значений: сигмоидальная функция – (0, 1), гиперболический тангенс – (–1, 1), функция Гаусса – (0, 1); для линейной

функции ограничений нет. Соответственно, в обучающих и тестирующих файлах выходные данные должны быть приведены к тому или иному масштабу. Для входных данных отсутствие масштабирования является не обязательным, но желательным условием, поскольку уменьшает возможность насыщения нейронов. Рекомендуемые диапазоны для масштабирования:

- для всех входов от 0 до 1 или от –1 до +1,
- для выходов – в случае сигмоидальной или гауссовой функций активации – от 0,1 до 0,9, а в случае гиперболического тангенса – от –0.9 до +0.9.

Нейропакет имеет опцию, позволяющую проводить соответствующее масштабирование и обратное преобразование к естественным, исходным значениям.

Подготовим текстовый файл, содержащий только таблицу, без комментариев и указания количества входов:

```
0 0 0
0 1 1
1 0 1
1 1 0
```

и сохраним его под именем xor.raw.

Запустим нейропакет. Выберем в меню (рис. 5.27) пункт **Scale/ Scale Data File**. При этом появится окно диалога (рис. 5.28).

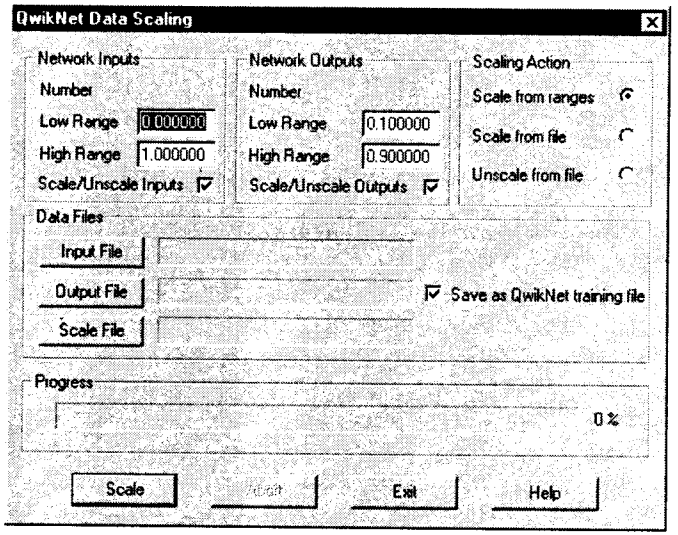


Рис. 5.28. Диалоговое окно при выполнении масштабирования данных

Полагая, что в дальнейшем будем строить сеть с сигмоидальными активационными функциями, оставим предлагаемые по умолчанию установки масштабирования и нажмем кнопку **Input File** (Входной файл). В последующем диалоге укажем файл xor.raw. После этого появится окно, требующее указать число входов и выходов в таблице обучающей выборки (рис. 5.29).

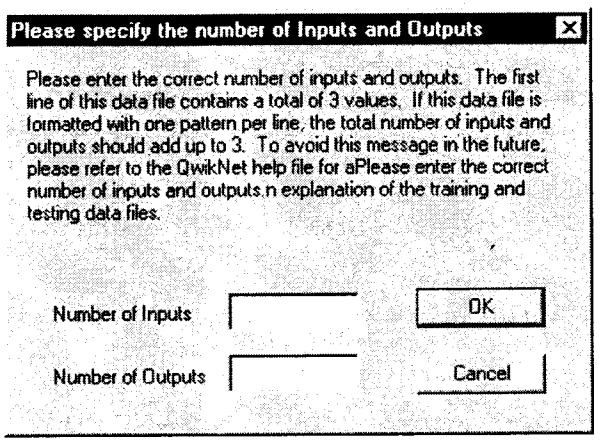


Рис. 5.29. Диалоговое окно задания числа входов и выходов

Введем 2 в строку **Number of Inputs** (Число входов) и 1 в строку **Number of Outputs** (Число выходов). Подтвердим ввод нажатием кнопки **OK**, после чего вернемся к окну рис. 5.28.

Нажмем кнопку **Output File** (Выходной файл) и в ответ на подсказку введем имя файла xor.trn. Нажмем далее кнопку **Scale File** (Файл масштаба) и в ответ на подсказку введем имя xor. Данный файл будет сохранен с расширением по умолчанию scl.

Выполним операцию масштабирования **Scale** и по нажатию кнопки **Exit** (Выход) вернемся к контрольной панели на рис. 5.27. Созданный файл xor.trn с обучающей выборкой будет иметь структуру, со строками [INPUTS] 2, [OUTPUTS] 1 и с диапазонами изменений входов и выходов согласно отмеченным рекомендациям:

```
[INPUTS] 2
[OUTPUTS] 1
0.000000  0.000000  0.100000
0.000000  1.000000  0.900000
1.000000  0.000000  0.900000
1.000000  1.000000  0.100000
```

2) Задание структуры сети и параметров обучения. Используя кнопки и диалоговые мини-окна контрольной панели, зададим следующую структуру нейронной сети:

- количество скрытых слоев (Number of Hidden Layers) – 1;
- число нейронов в скрытом слое (Hidden 1/Neurons) – 2;
- функции активации нейронов скрытого слоя (Hidden 1/ Activation Function) – сигмоидные (Sigmoid);
- функция активации выходного нейрона (Output/Activation Function) – сигмоидная (Sigmoid).

Сохраним остальные параметры процесса обучения по умолчанию как на рис. 5.27, убрав лишь флажок в окне **Cross-Validate Training** (Обучение с перекрестным пересечением). Далее нажатием кнопок **Training Data** и **Testing Data** выберем и в качестве обучающего и в качестве тестирующего один и тот же файл – xor.trn.

3) Обучение нейронной сети. Используя кнопку с треугольным значком **Train Network** (Обучение сети) в верхней части контрольной панели или кнопку **Train** (Обучение) в ее правой нижней части, запустим процесс обучения. Данный процесс будет сопровождаться изменением цифр в разделах панели **Training Stats** (Статистика обучения) и **Testing Stats** (Статистика тестирования). По истечении процесса обучения панель программы примет вид, соответствующий рис. 5.30.

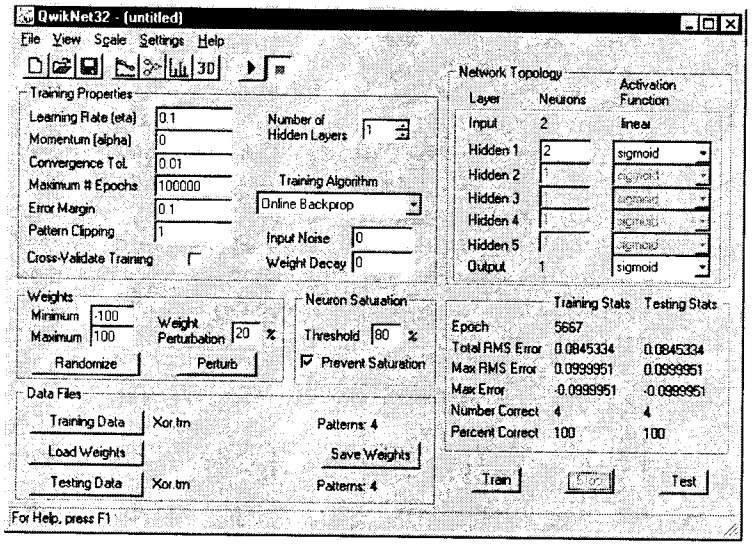


Рис. 5.30. Контрольная панель по завершении процесса обучения нейронной сети

Судя по результату, обучение прошло успешно (правильно оценены все 4 образца, т. е. 100% образцов, количество периодов (Epoch) обучения – 5667 – меньше предельно заданного (10000) значения, величины ошибок также в заданных пределах.

Теперь можно в полной мере оценить графические возможности пакета. Рассмотрим, например, графическое изображение сети, используя кнопку **View Network** или соответствующие опции меню (рис. 5.31).

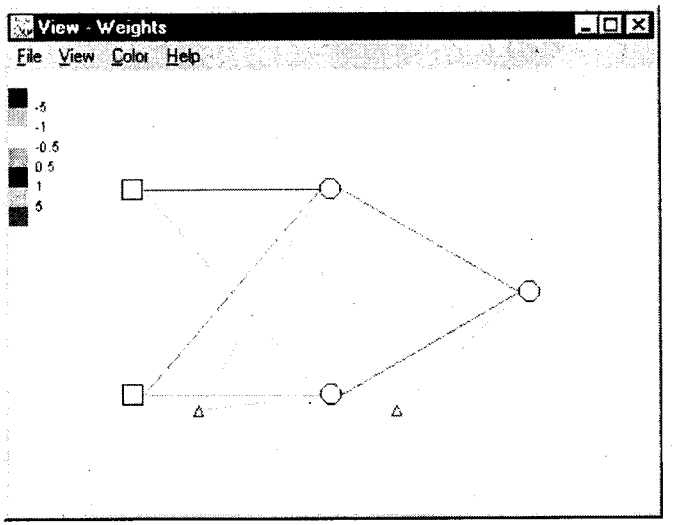


Рис. 5.31. Вид структуры нейронной сети

Можно просмотреть график анализа сети (**Network Analysis Plot**) с разнообразными опциями его представления (рис. 5.32), а также, используя пункты меню **View/Contour Plot** или кнопку **3D**, увидеть трехмерное изображение выхода сети (рис. 5.33) и даже повращать его. Можно просмотреть файл протокола обучения сети (опция меню **View/Training History**) и еще многое другое.

4) Сохранение результатов. С помощью опций меню **File/Save as ...** сохраним полученную сеть, например, под именем xor (с расширением по умолчанию *.net).

При работе с нейропакетом необходимо отдельно сохранять веса и смещения обученной сети (с помощью кнопки **Save Weights**) в файле с расширением по умолчанию *.wts, например, с именем xor.wts. Он, кстати, имеет текстовый формат, что удобно

для просмотра). Закроем программу, теперь обученную нейронную сеть можно использовать для прогнозирования.

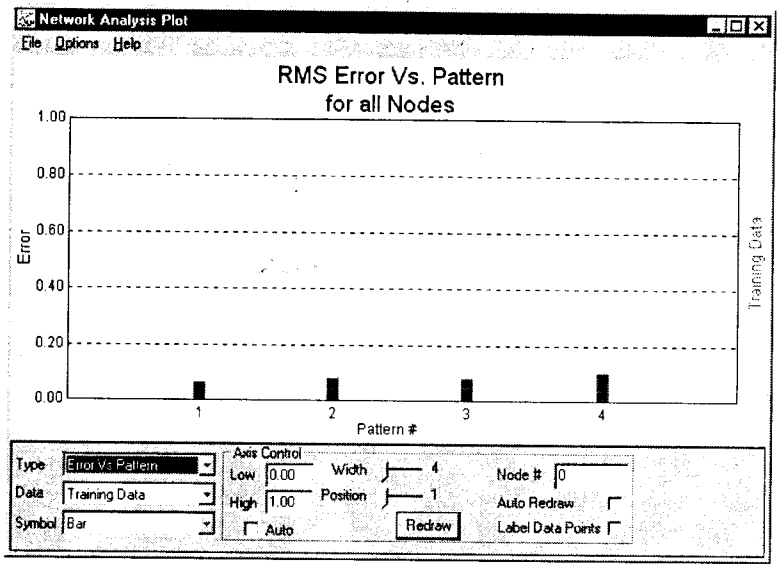


Рис. 5.32. График анализа нейронной сети

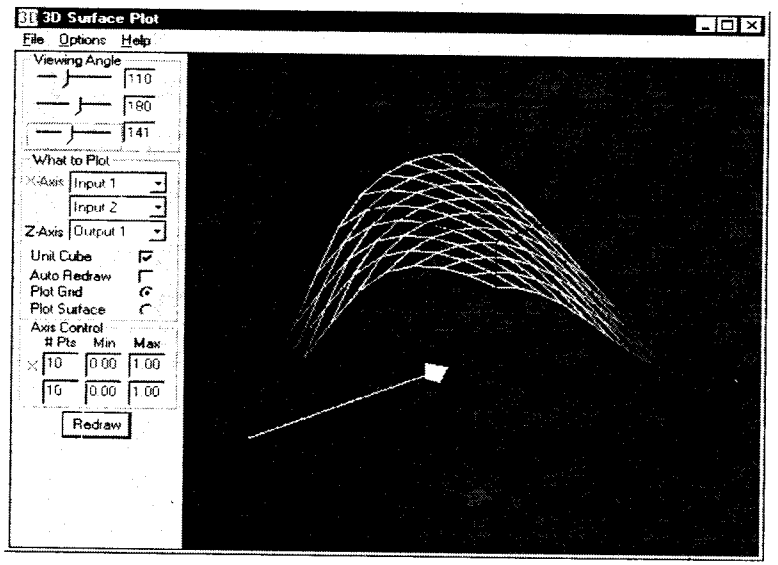


Рис. 5.33. Трехмерное представление выхода обученной нейронной сети

5) Опрос сети. Для выполнения прогноза, подготовим файл текстового формата со следующим содержимым:

```
[INPUTS] 2
0.000000  0.000000
0.000000  1.000000
1.000000  0.000000
1.000000  1.000000
```

т. е. включающим только значения входов. Сохраним его под именем xortest.tst.

Запустим нейропакет, откроем файл xor.net с сохраненной структурой сети и файл xor.wts с ее весами. Далее нажатием кнопки **Testing Data** в качестве тестового файла укажем созданный файл xortest.tst. После чего нажмем кнопку **Test** в правом нижнем углу контрольной панели (рис. 5.30). При этом появляется диалоговое окно (рис 5.34), предлагающее указать имя выходного файла (с расширением по умолчанию *.out).

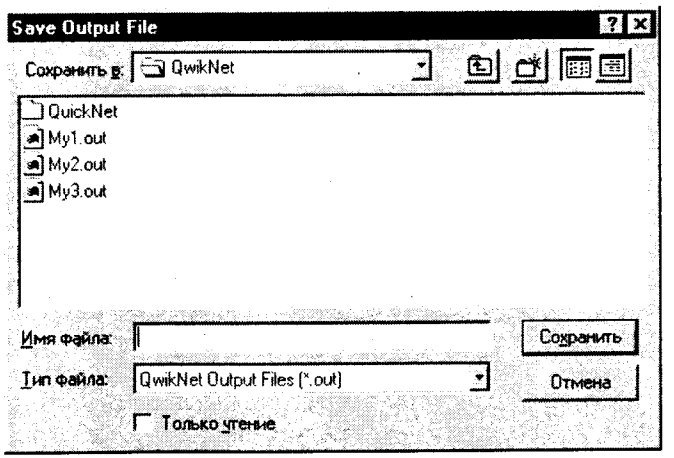


Рис. 5.34. Диалоговое окно для задания имени выходного файла

Укажем имя хог и нажмем кнопку **Сохранить**, после чего появляется новое окно (рис. 5.35), запрашивающее запись имен столбцов (**Write column headers**) и выходов–образцов тестового файла (**Write test file outputs (targets)**).

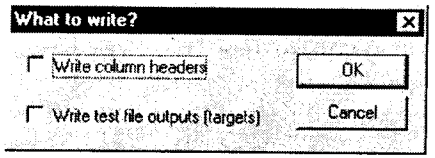


Рис. 5.35. Запрос на вид выходного файла

В нашем случае выходов у тестового файла нет, поэтому укажем только запись имен, подтвердив это нажатием кнопки **OK**. После этого в рабочей директории нейропакета появится файл xor.out с содержимым:

```
Output1
  0.160291
  0.822393
  0.821254
  0.199978
```

Цифры, в общем, близки к цифрам последнего столбца файла xor.trn (см. выше); не надо забывать, что к тому же выход сети здесь представлен в масштабируемом виде.

Можно перейти к естественному масштабу. Для этого вначале несколько изменим содержание фала xor.out, добавив звездочку в начало первой строки (т. е. превратив заголовок в комментарий), затем с помощью пункта меню **Scale/Scale Data File** перейдем к окну масштабирования вида рис. 5.28. Укажем в нем в качестве входного файл (модифицированный) xor.out. Далее в появившемся окне диалога вида рис. 5.29 укажем число входов – 0, число выходов – 1. После возвращения к окну масштабирования, зададим имя выходного файла (**Output File**), например xorout.trn, укажем в качестве файла масштаба (**Scale File**) ранее созданный xor.scl, установим опцию **Unscale from file** (в правой верхней части окна) и нажмем кнопку **Scale**. После завершения процесса преобразования данных можно закрыть окно масштабирования.

В итоге, получим текстовый файл xorout.trn с содержимым:

```
[OUTPUTS] 1
0.075364
0.902991
0.901568
0.124972
```

которое и является прогнозируемым выходом нашей сети в естественном масштабе. Как видно, погрешность здесь все же не очень мала. Можно попытаться повторить процесс обучения, задать другие параметры (меньшую величину ошибки и т. п.), при этом результат, естественно, будет точнее.

5.3.3. Общее суждение

Рассмотренный пакет имеет много привлекательных черт: он компактен, достаточно прост в изучении и обращении, имеет хорошие иллюстративные графические возможности, позволяет сохранять обученную нейронную сеть в кодах языка С, имеет достаточно хорошие возможности по моделированию и обучения нейронных сетей.

К недостаткам пакета, на наш взгляд, следует отнести необходимость использования, по сути, ручного масштабирования данных и опроса сети.

5.4. Нейропакет Neural Planner

5.4.1. Общая характеристика

Neural Planner – программная оболочка, позволяющая моделировать нейронные сети различной конфигурации. Может работать под Windows как в сети, так и на локальном компьютере и не предъявляет особых требований к оборудованию. Neural Planner предназначен для решения различных задач классификации объектов, обработки значений случайных процессов, решения некоторых математических задач, создания эффективных экспертных систем.

Neural Planner использует один из двух реализованных алгоритмов обучения с учителем, которые будут описаны ниже.

Основными разделами пакета являются встроенный графический редактор и аналог табличного редактора. Графический редактор позволяет легко и наглядно создавать и редактировать нейронные сети без необходимости их описания в текстовом виде. Аналог табличного редактора позволяет создавать и редактировать обучающие векторы и векторы опроса, а также управлять процессом обучения.

Условно-бесплатная (shareware) версия пакета доступна в Интернет по адресу: http://www.simtel.net/simtel.net/win3/neural-pre.html (файл np452s.zip). Разработчик – S. Wolstenholme (18 Seymour Road, Cheadle Hulme, Cheshire, SK8 6LR, UK; e-mail: steve@tropheus.demon.co.u).

5.4.2. Форматы файлов

Создаваемый пакетом файл, в котором хранится графическое изображение нейронной сети, имеет расширение *.nnp. Кроме этого, в процессе обучения Neural Planner автоматически создает дополнительные файлы: сеть с самой низкой ошибкой (с расширением *.lnp) и сеть с наилучшим результатом теста (с расширением *.bnp). Обучение, опрос и тестирование сети осуществляется при помощи отдельного файла, имеющего расширение *.tti (от первых букв английских слов training, testing, interrogating – обучение, тестирование, опрос). Этот файл состоит из трех разделов, в которых содержатся соответствующие векторы обучения, опроса и тестирования сети (в табличной форме). Каждый раздел создается и редактируется отдельно в своем диалоговом окне.

Кроме этого, автоматически создается резервная копия сетевого файла, имеющего расширение *.tbu.

5.4.3. Команды основного меню программы

Основное меню программы включает в себя следующие пункты: **File** (Файл), **Edit** (Редактирование), **View** (Просмотр), **Control** (Управление), **Action** (Действие), **Display** (Дисплей), **Options** (Опции). Окно, появляющееся при запуске Neural Planner, приведено на рис. 5.36.

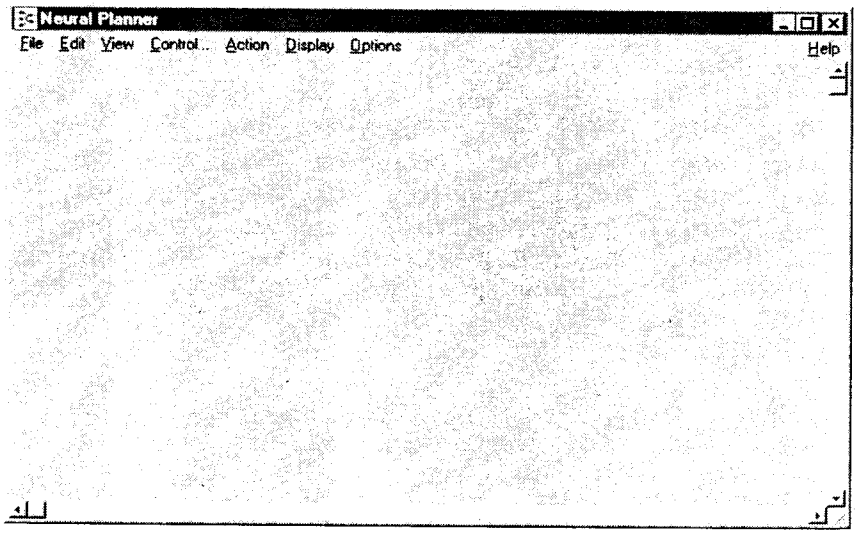


Рис. 5.36. Основное окно программы.

Пункт меню **File** (Файл) содержит стандартные базовые операции с файлами: **New** (Новый), **Open** (Открыть), **Save...** (Сохранить...), **Save As...** (Сохранить как...), **Exit** (Выход).

Команды пункта меню **Edit** (Редактирование):

- **Add Input Neuron(s)** – добавить входной нейрон.
- **Add Hidden Neuron(s)** – добавить скрытый нейрон.
- **Add Output Neuron(s)** – добавить выходной нейрон.
- **Delete Neuron(s)** – удалить нейрон.
- **Add Synapse(s)** – добавить синапс.
- **Deletes Synapse(s)** – удалить синапс.
- **Connect Layers** – соединить слои (команда применяется для полного соединения двух слоев нейронов).
- **Disconnect Layers** – разъединить слои (команда применяется для полного разъединения двух слоев нейронов).
- **Add Zones(s)** – добавить зоны (команда добавляет прямоугольные зоны).
- **Delete Zones(s)** – удалить зоны (команда удаляет зоны).
- **Call associated tti editor** – вызвать ассоциированный tti редактор. Команда вызывает какой-либо текстовый редактор или систему электронных таблиц для редактирования tti файлов.

Пункт меню **View** (Просмотр) содержит команды для увеличения/уменьшения размеров отображаемых нейронов и синапсов **Zoom In/Zoom Out**), а также для задания максимального/минимального масштабов изображений (**Max Zoom/Min Zoom**).

Команда **Control (Управление)** вызывает диалоговое окно, позволяющее наблюдать за процессом обучения сети и управлять им (рис. 5.37).

Рассмотрим опции окна.

- **Learning rate** – коэффициент скорости обучения, параметр алгоритма обучения по методу обратного распространения ошибки.
- **Momentum** – импульс (или коэффициент импульса) определяет изменение веса синапса в текущем цикле обучения относительно изменения в предыдущем цикле.
- **Target Error** – целевая ошибка.

Следующие три опции активируются, если предусматривается тестирование сети.

- **Cycles Per Test** – определяет количество циклов обучения после каждого цикла теста нейронной сети.
- **Cycles Before Test** – определяет количество циклов обучения, которое должно быть выполнено перед каждым тестированием нейронной сети.

- **Target % Correct or in Range +/– %** – обучение остановится, если полученный в процессе обучения процент правильных результатов больше заданного в данной опции теста.
- **Cycles Per Refresh** – определяет количество циклов обучения, которые должны быть завершены между каждой регенерацией дисплея.

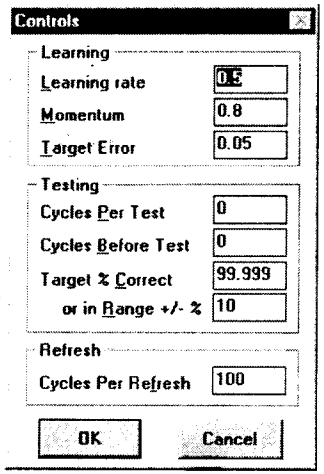


Рис. 5.37. Диалоговое окно задания параметров процесса обучения

Команды пункта меню **Action (Действия)**.

- **Interrogate** – позволяет опросить обученную сеть.
- **Learn from File...** –позволяет обучить сеть на основе векторов обучения, записанных в разделе Training tti файла.
- **Smart-Start** – интеллектуальный старт. Эта команда аналогична предыдущей, но здесь Neural Planner автоматически устанавливает параметры обучения, при которых средняя ошибка после одного цикла обучения будет наименьшей.
- **Create Network** – создать сеть на основе существующего tti файла.
- **Forget Learning** – забыть обучение. Эта команда используется, если необходимо переобучить сеть, задав другие параметры обучения.
- **Reset** – сброс. При выполнении этой команды сбрасываются опции Lowest Error (Наименьшая ошибка) и Best Test (Наилучший тест).

Команды пункта меню **Display** (Дисплей) рассмотрены ниже.

Команды пункта **Options (Опции)**.

- **Auto Refresh Display** – автообновление дисплея, т. е. автоматическая регенерация дисплея при редактировании сети. Если эта опция отключена, результаты каких-либо действий не будут отображаться, пока не будут сохранены результаты изменений.
- **Auto Delete Synapses** – автоматическое удаление синапсов (входов), веса которых близки к нулю.
- **Stop on 100% Cases Under Target** – остановка, если 100% случаев меньше целевой ошибки. Эту опцию необходимо установить, если векторы обучения сети должны совпадать с векторами опроса, например, когда все возможные задачи, решаемые такой сетью использовались и для ее обучения, в частности, при решении логических задач. Когда сеть решает задачи одного класса, но не все возможные варианты вопросов входят в обучающую выборку, опция должна быть отключена.
- **Learning Algorithm** – алгоритм обучения. Опция позволяет выбрать один из двух обучающих алгоритмов, реализованных в Neural Planner: On-Line Back Propagation и Batch Back Propagation.
- **Training Set Defaults** – обучающая последовательность по умолчанию. Если установлена эта опция, то при пропуске каких-либо компонент векторов обучения или опроса, они автоматически заменяются на минимальные, максимальные или средние значения из существующих компонентов.
- **Graph** – график. Опции этого пункта определяют вид графика изменения средней ошибки при обучении сети.
- **Plot from start of learning** – рисовать с начала обучения. Если установлена эта опция, график начинает изменяться как только начинается процесс обучения (установлена по умолчанию).
- **Restart if scale change** – перезапуск при изменении масштаба. Эта опция позволяет в процессе обучения автоматически перезапустить график, если изменился его масштаб. При этом предыдущие значения стираются. Если эта опция не установлена, то изменение масштаба показывается вертикальной линией от места его изменения.
- **Step «X» only if «Y» changes** – шаг по «X», если изменилось «Y». Если установлена эта опция движение по оси «X» осуществляется только в том случае, если изменилось значение «Y».

5.4.4. Работа с пакетом

1) Создание и редактирование нейронной сети.

Создать сеть в Neural Planner можно двумя способами. В первом, сначала создается nnp-файл, а затем на его основе tti-файл. Второй способ позволяет сделать наоборот, сначала соз-

дать tti-файл, а затем программа автоматически по этому файлу строит сеть, что часто оказывается более удобным.

Для того, чтобы воспользоваться первым способом, необходимо выбрать команду **New/Network file** из меню **File**. Для более удобного расположения нейронов, можно включить линии сетки на экране командой **Background Grid** из меню **Display**. Для непосредственного построения или редактирования сети необходимо далее воспользоваться командами меню **Edit**. Все действия по редактированию сети выполняются левой кнопкой мыши.

Пример сети приведен на рис. 5.38.

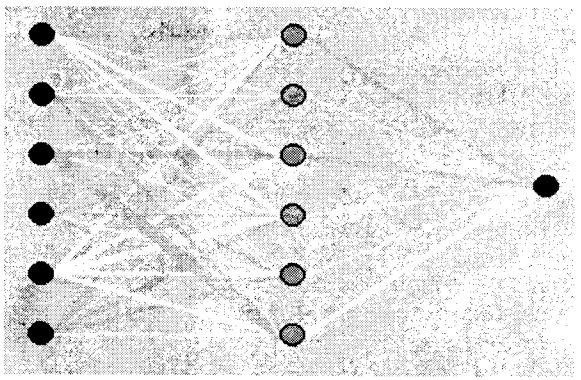


Рис. 5.38. Пример нейронной сети

При помощи команды **Add Input Neuron(s)** можно добавить входной нейрон (маркируется красным цветом). После выполнения этой команды нажатием левой кнопкой мыши добавляется входной нейрон на место, указанное курсором.

Аналогично при помощи команды **Add Hidden Neuron(s)** можно добавить нейрон скрытого слоя (маркируется зеленым цветом), а при помощи команды **Add Output Neuron(s)** – выходной нейрон (маркируется синим цветом).

Для удаления любого нейрона используется команда **Delete Neuron(s)**. При этом для удаления нейрона сначала необходимо выбрать данную команду а затем пометить удаляемый нейрон.

При помощи команды **Add Synapse(s)** добавляется синапс между парой нейронов. Для этого сперва выбирается первый (левый) нейрон из пары, после этого помечается другой нейрон, и производится соединение. Любой нейрон остается выбранным, до тех пор, пока он не помечен второй раз.

Для удаления синапса необходимо воспользоваться командой **Delete Synapse(s)**. При удалении синапса сначала помечается нейрон–источник, затем нейрон, с которым он соединен.

Если необходимо осуществить полносвязное соединение двух слоев нейронов, то можно воспользоваться командой **Connect Layers**. Для нормального, без обратной связи, соединения сначала выбирается предыдущий слой, затем последующий. Синапсы, обеспечивающие обратную связь, могут соединять слои только справа налево и только между скрытыми или выходными нейронами.

Для разъединения слоев нужно воспользоваться командой **Disconnect Layers**. Для разъединения нормального соединения сначала выбирается правый уровень, для разъединения обратной связи – левый уровень.

Для построения более сложной сети можно применить ее деление на зоны при помощи команды **Add Zones(s)**. При помощи зон можно разбивать существующую нейронную сеть на несколько отдельных частей и затем рассматривать их как независимые (и обучать, и опрашивать отдельно). При создании или добавлении зоны вначале помечается левый верхний угол выделяемой области, затем правый нижний.

Для удаления зоны используется команда **Delete Zones(s)**. После выбора этой команды достаточно щелкнуть мышью внутри зоны и она автоматически удалится.

После того, как сеть будет создана, ее необходимо сохранить при помощи стандартных команд **Save** или **Save As**. При этом файлу автоматически присваивается расширение *.nnp.

Для того, чтобы создать сеть вторым способом, необходимо воспользоваться командой **Create Network** из меню **Action**. Для этого предлагается выбрать какой-либо tti-файл (как создавать такие файлы будет рассмотрено ниже). После выбора файла на его основе автоматически будет создана сеть, по умолчанию содержащая только один слой скрытых нейронов. Созданную таким образом сеть необходимо сохранить под тем же именем, что и tti-файл.

Заметим, что всю необходимую информацию о сети в целом и по каждому нейрону и синапсу в отдельности можно при помощи команд меню **Display**:

- **Neuron Labels** – метки нейронов. При выборе этой команды на экране появляются метки всех нейронов в текущей загруженной сети, но при этом обязательно должен существовать соответствующий tti файл.

• **Neuron Activations** – активации нейронов. При выборе этой команды появляются уровни активации всех нейронов. Чем длиннее полоса, тем выше уровень активации.

• **Neuron Details** – позволяет отобразить всю информацию о любом выбранном в сети нейроне (рис 5.39). Для того, чтобы это сделать, необходимо сначала выбрать команду, затем выбрать нейрон. Кроме того, можно, не покидая окна, выбрать другой нейрон, соединенный с предыдущим синапсом, нажав кнопку **Next Synapse**, или соседний нейрон кнопкой **Next Neuron**.

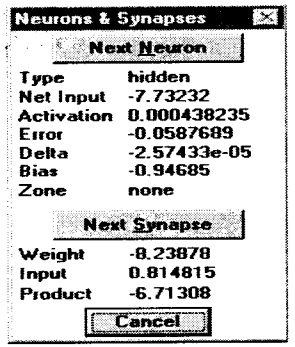


Рис. 5.39. Диалоговое окно **Neuron Details**

В окне отображается следующая информация:

• Type – тип выбранного нейрона (input – входной, output – выходной, hidden – скрытый);

• Net Input – уровень входного сигнала;

• Activation – уровень активации в относительных единицах;

• Error – ошибка вычисления;

• Delta – разность между полученным выходом и требуемым (целевым);

• Bias – смещение нейрона;

• Zone – в какой зоне расположен выбранный нейрон;

• Weight – вес синапса в относительных единицах;

• Input – сумма всех весов синапсов, умноженных на входы нейрона;

• Product – вес синапса умноженный на входное значение.

• **Synapse Weights** – веса синапсов (входов). При выборе этой команды отображаются веса всех синапсов сети. При этом

ширина синапса показывает его вес относительно всех других. Синапсы с положительным весом выделяются желтым цветом, с отрицательным – синим. Черным цветом выделяются синапсы, веса которые близки к нулю и подлежат удалению, поскольку не оказывают никакого влияния на вычисления.

• **Network Details** – детали сети. Эта команда позволяет в отдельном диалоговом окне отобразить всю информацию о любом выбранном нейроне (рис. 5.40):

- Neurons – количество нейронов;
- Synapse connections – количество соединений;
- Average error – средняя ошибка;
- Lowest error – наименьшая ошибка;
- Best test results –лучший результат тестирования;
- Total cycles – количество циклов обучения.

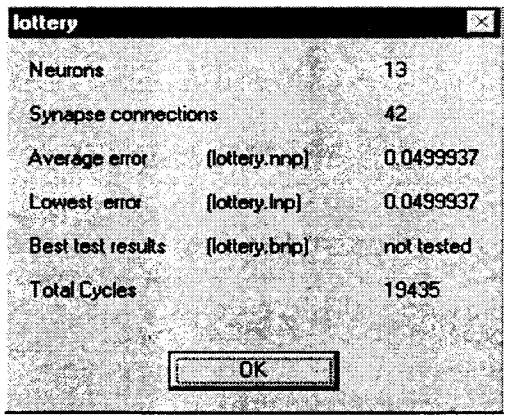


Рис. 5.40. Окно Network Details

• **Network Zones** – зоны сети. При выборе этой команды показываются все зоны, если сеть на них разбита.

• **Warning Messages** – предупреждающие сообщения. При выборе этой команды подавляются все предупреждающие сообщения, выводимые программой.

2) Обучение нейронной сети

В Neural Planner, так же, как и в ранее рассмотренных нейропакетах, предполагается процесс обучения с учителем, при котором сети для ее обучения необходимо предъявлять обучающую выборку из векторов, содержащих известные входные и выходные значения. Эта выборка записывается в специальный текстовый

файл с расширением *.tti. Файл состоит из четырех разделов: обучения, опроса, тестирования и ограничений.

Для того, чтобы создать новый файл обучения, тестирования и опроса, необходимо выбрать команду **New/Training, Testing, Interrogating file** (Новый файл обучения, тестирования, опроса) из меню **File**. После этого появится диалоговое окно, приведенное на рис. 5.41, в котором необходимо указать метки входных и выходных нейронов (разделы **Inputs** и **Outputs**).

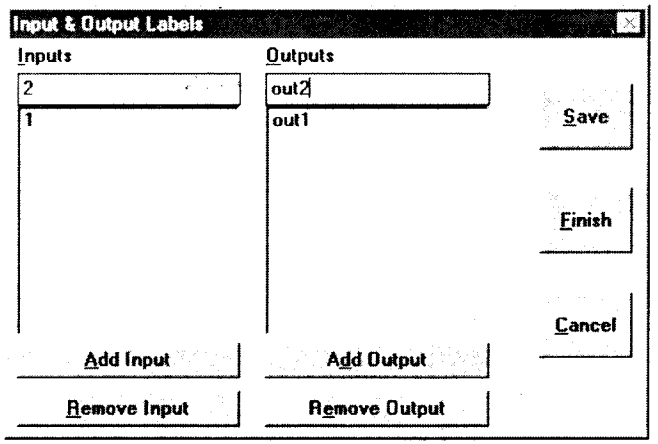


Рис. 5.41. Диалоговое окно ввода меток нейронов

Метками могут служить любые символы или группы символов. После ввода каждой метки нажимается кнопка **Add Input** или **Add Output**. Удалить метку можно при помощи кнопки **Remove**. После ввода всех меток необходимо нажать кнопку **Finish** и сохранить файл. Кнопка **Save** используется, если редактируется уже существующий файл. В итоге формируется каркас tti-файла.

Для ввода в tti-файл векторов обучения нужно открыть требуемый раздел командой **Open/Training File Section** из меню **File**. В левую часть появившегося диалогового окна (рис 5.42) необходимо занести имя вводимого вектора, а в правую – значения его компонентов. Порядок ввода следующий: вводится имя вектора, нажимается кнопка **New**, затем в правой части окна символы «?» заменяются на значения компонентов, после ввода всех компонентов нажимается кнопка **Replace**. Кнопка **Copy** создает копию текущего вектора, указанного курсором. Кнопка **Delete** удаляет текущий вектор. После ввода всех векторов необходимо нажать кнопку **Finish**, и ввод будет завершен.

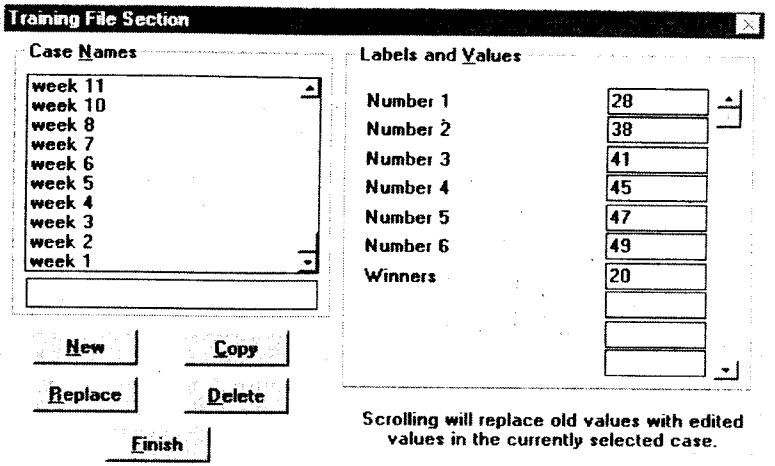


Рис. 5.42. Диалоговое окно ввода векторов обучения.

Следует обратить внимание на автоматически появляющиеся вектора с именами Highs (Верхний), Lows (Нижний) и Dummy (Шаблон, Болванка). Компонентами двух первых векторов будут присвоены максимальные и минимальные значения из всех введенных, а компоненты вектора Dummy по умолчанию – нулевые (данный вектор следует удалить).

Перед началом процесса обучения необходимо выбрать один из двух обучающих алгоритмов. Алгоритм On-Line Back Propagation предусматривает вычисление ошибки после предъявления каждого вектора обучающей выборки внутри цикла обучения. Алгоритм Batch Back Propagation предусматривает вычисление ошибки после завершения каждого цикла обучения. Общих рекомендаций по выбору того, или иного алгоритма не существует.

Процессом обучения можно достаточно эффективно управлять при помощи опций диалогового окна **Control** (см. рис. 5.37). Рассмотрим эти опции и их влияние на процесс обучения.

Опция **Learning rate** (коэффициент скорости обучения) определяет скорость изменения веса синапсов в течение каждого цикла обучения сети. Значение **Learning rate** может изменяться в пределах от 0,1 до 1. Например, если задано значение 0,5, а первоначальное значение веса какого-либо синапса было 5, то в следующем цикле обучения значение веса может измениться не более, чем на 5 + 0,5 · 5, или не менее, чем на 5 – 0,5 · 5. Малый коэффициент замедляет процесс обучения, но делает его более на-

дежным, в то время как высокий коэффициент может заставить обучение отклониться от оптимального направления спуска к нулевой ошибке. Коэффициент может быть изменен в течение обучения.

Опция **Momentum** (импульс) определяет величину изменения веса синапса относительно изменения в предыдущем цикле обучения. Значение **Momentum** может изменяться в пределах от 0 до 1. Например, если задано значение 0,5, а в предыдущем цикле обучения вес синапса изменился на 2,5, то в следующем цикле вес не может измениться более чем на 1,25. Маленький импульс будет замедлять обучение, но также повышает надежность сходимости процесса обучения, в то время как высокий импульс может приводить к колебаниям относительно оптимального направления спуска к нулевой ошибке. Очень большой импульс может иногда привести процесс обучения к локальному минимуму. Импульс может быть изменен в течение обучения.

При помощи приведенных двух опций можно достаточно эффективно управлять процессом обучения сети.

Опция **Target Error** задает так называемую целевую ошибку. Обучение останавливается, если средняя ошибка **Average Error**, которая уменьшается в процессе обучения, достигнет значения, меньшего, чем заданная целевая ошибка. Очевидно, что чем меньшее значение этой опции задано, тем более длительным будет процесс обучения.

Опция **Cycles Per Refresh** определяет количество циклов обучения, которые должны быть завершены между каждой регенерацией дисплея.

После того, как установлены все опции окна **Control** и выбран обучающий алгоритм, необходимо запустить процесс обучения сети при помощи команды **Learning From File** из меню **Action**. При выполнении команды появляется стандартное диалоговое окно, в котором необходимо выбрать соответствующий tti-файл. Перед началом обучения сеть автоматически проверяется на совместимость с tti-файлом. Критерием совместимости служат метки нейронов. Если метки в сети и в файле не совпадают, выдается сообщение об ошибке.

Одновременно с запуском процесса обучения на экране появляется окно, в котором отражаются все процессы, протекающие в сети при обучении (рис. 5.43).

Рассмотрим опции окна. В заголовке приводится тип обучающего алгоритма, далее – имя сетевого файла и имя обучающего файла. Ниже показывается график средней ошибки. Ось «X» графика является безразмерной, ось «Y» соответствует величине

средней ошибки. График можно отключить при помощи кнопки **Graph**.

Еще ниже в числовом виде отображаются средняя и целевая ошибки (Neural Planner периодически запоминает в файле с расширением lnp сеть с наименьшей к текущему моменту ошибкой, показанной в опции **Lowest Saved**), процент векторов обучения, при прогнозе которых средняя ошибка меньше заданной целевой (**Cases < Target**).

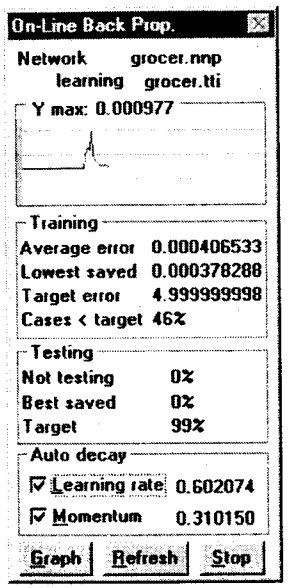


Рис. 5.43. Окно, отображающее процесс обучения

Отметим, что для обучения сети можно воспользоваться также командой **Smart-Start** из меню **Action**. Эта команда аналогична команде **Learning From File**, но здесь Neural Planner автоматически установит такие параметры окна **Control**, при которых обеспечивается минимальная средняя ошибка за один цикл обучения.

Редактирование tti-файла при помощи системы электронных таблиц MS Excel.

Файл обучения, тестирования и опроса по своей структуре является текстовым файлом и редактируется при помощи встроенного текстового редактора Neural Planner. Однако данный редактор обладает малыми возможностями и работать с ним крайне не-

удобно, особенно если требуется вводить большое количество векторов. Поэтому рекомендуется в Neural Planner создавать только каркас tti-файла, а затем заносить в него значения при помощи любой системы электронных таблиц, в частности, Excel. При открытии tti-файла в Excel, он автоматически преобразуется в нужный формат, аналогичный приведенному в табл. 5.1.

Таблица 5.1

Пример редактирования tti-файла

[LABELS]	[X11]	[X12]	[X21]	[X22]	[D]
[END]					
[TRAINING]	>	>	>	>	<
[dummy]	0	0	0	0	0
[1]	1	0	0	1	1
[2]	3	-1	0	-2	-6
[3]	2	1	4	2	0
[4]	-1	-1	-2	0	-2
[5]	0	4	-1	5	4
[END]					
[INTERROGATING]	>	>	>	>	<
[dummy]	0	0	0	0	0
[1]	3	-1	0	-2	-6
[2]	-1	-1	-2	0	-2
[3]	0	4	-1	5	4
[END]					
[LIMITS]	>	>	>	>	<
[highs]	3	4	4	5	4
[lows]	-1	-1	-2	-2	-6
[END]					

Вначале в таблице выводятся названия разделов и имена векторов для каждого раздела, в верхней строке выводятся метки нейронов. Значения векторов вводятся в ячейки на пересечении имени вектора и соответствующей метки нейрона. Следует обратить внимание на то, что для входных значений символы–разделители разделов имеют вид «>», а для выходных – «<». После того, как файл будет отредактирован, необходимо сохранить его под тем же именем и с тем же расширением, не преобразовывая в формат *.xls.

3) Тестирование нейронной сети.

Напомним, что тестирование нейронной сети – это ее кратковременный опрос в процессе обучения. В процессе такого опроса сети предъявляются векторы тестирования, вычисляются выходные значения и находится текущая ошибка на данном шаге обучения. Тестирование осуществляется при помощи векторов, занесенных в раздел теста tti-файла. Векторы тестирования могут совпадать с векторами обучения или быть другими, но с заранее известными выходными значениями. Раздел теста открывается при помощи команды **Open/Testing File Section** из меню **File**. Ввод векторов в этот раздел аналогичен вводу векторов обучения и опроса. Управлять процессом тестирования можно при помощи опций окна **Control** (рис. 5.37). Опция **Cycles Per Test** определяет количество циклов обучения, между каждым циклом теста нейронной сети (данной опции должно быть присвоено значение ноль, если тестирование не предусматривается). Опция **Cycles Before Test** определяет количество циклов обучения, которое должно быть завершено перед первым циклом теста нейронной сети (значение опции желательно устанавливать достаточно высоким, чтобы позволить сети находить оптимальное направление спуска к нулевой ошибке прежде, чем будет проведено тестирование; в противном случае тестирование не даст положительных результатов). Опция **Target % Correct or in Range +/– %** устанавливает целевой процент правильных результатов, при достижении которого обучение останавливается.

После каждого тестирования сеть автоматически записывается в файл с расширением *.bnp. Если результаты текущего теста лучше полученных ранее (т. е. ошибка меньше), то файл обновляется.

4) Опрос нейронной сети.

Для того, чтобы опросить сеть, необходимо предварительно ввести входные значения в специальный раздел tti-файла. Данный раздел открывается при помощи команды **Open/Interrogating File Section** из меню **File**. Ввод векторов опроса аналогичен вводу векторов обучения, но в разделе выходных значений сети необходимо оставить символ «?».

Опрос сети осуществляется командой **Interrogate** из меню **Action**. Перед опросом нужно загрузить раздел **Interrogating** (Опрос) tti-файла при помощи стандартного окна **Open File**, появляющегося при выборе команды. Результаты опроса, отображаются в соответствующем диалоговом окне, приведенном на рис. 5.44.

Окно содержит средства управления, позволяющие переключаться между входными векторами и изменять их значения. Входные значения сети можно изменить, используя опции **Increase** (Увеличение), **Decrease** (Уменьшение), **Change** (Изменение), **Min** (Минимальное) и **Max** (Максимальное) значение. Переключаться между векторами опроса можно при помощи кнопок **Next** (Сле-

дующий), **Back** (Предыдущий), **First** (Первый) и **Last** (Последний). Кнопки **Seek High** и **Seek Low** позволяют найти входные значения текущего вектора опроса, приводящие к самому высокому или низкому значению компонентов выходных векторов (при такой операции необходимо установить флажки в опциях **Cycle** (Цикл) и **Real** (Действительное значение).

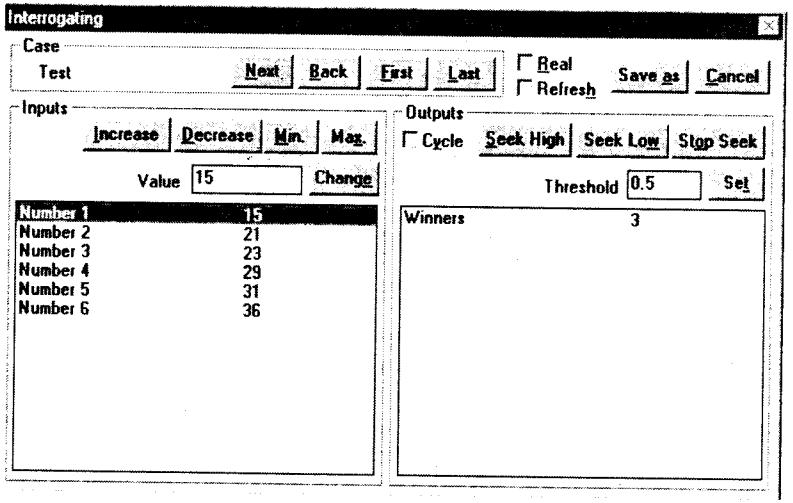


Рис. 5.44. Диалоговое окно **Interrogating**.

Любой протокол опроса можно записать в отдельный текстовый файл при помощи команды **Save as**.

5.4.5. Впечатления от работы с нейропакетом

Нейропакет Neural Planner 4.52 обладает некоторыми преимуществами по сравнению с ранее рассмотренными, в частности, он позволяет создавать нейронную сеть достаточно сложной структуры (в том числе, с обратными связями). Более простыми являются организация опроса нейронной сети и подготовка данных для обучения и тестирования.

В то же время возможности пакета ограничены по числу алгоритмов обучения, а интерфейс излишне упрощен.

В общем, можно полагать, что данный нейропакет по классу примерно сопоставим с двумя предыдущими.

5.5. Нейропакет BrainMaker

Большое число модификаций весьма популярного в нашей стране, особенно среди экономистов, нейропакета BrainMaker разработано фирмой California Scientific Software, USA (адрес сайта в Интернет: http://www.calsci.com.products.html). Ниже рассмотрим особенности одной из распространенных версий данного пакета – BrainMaker 3.10, которую можно найти на многочисленных CD-дисках тематики «Математика, экономика, статистика».

5.5.1. Общая характеристика

Пакет предназначен для построения многослойных нейронных сетей с алгоритмом обучения обратного распространения ошибки. Он включает в себя программу подготовки и анализа исходных данных NetMaker, программу построения, обучения и запуска нейронных сетей BrainMaker, а также набор утилит широкого назначения.

Нейропакет ориентирован на широкий круг задач – от создания прогностических приложений до организации систем распознавания образов и нейросетевой памяти. Значительное количество функций программы ориентировано на специалистов в области исследования нейронных сетей. Следует отметить, что организация внутреннего представления нейросетевых моделей является «прозрачной» и легко доступной для программного наращивания. В программе BrainMaker предусмотрена система команд для пакетного запуска. Существует интерфейсная программа-функция для включения обученных сетей в программы пользователя. В целом пакет может быть легко интегрирован в программный комплекс целевого использования.

Программа NetMaker предназначена для ввода исходных данных вручную либо из файлов популярных форматов, статистико-математического анализа этих данных, проведения стандартных процедур их преобразования и создания входных файлов для программы BrainMaker. Программа также способна обрабатывать выходные данные нейронной сети, выводить статистику ее обучения и прогонки.

Программа BrainMaker предназначена для построения нейронной сети по заданным параметрам, ее обучения в различных режимах, модификации параметров сети. Программа имеет значительное количество контрольных функций для оптимизации процесса обучения. Помимо этого, она реализует ряд методов анализа чувствительности выходов сети к различным вариациям входных данных, при этом формируется подробный отчет, в соответст-

вии с которым можно дополнительно оценить степень функциональной зависимости входных и выходных значений.

Нейросетевой симулятор BrainMaker хорошо зарекомендовал себя в задачах прогнозирования, классификации, обработки некорректных данных. Вот некоторые области, где BrainMaker уже успешно используется: портфельная торговля, тотализатор на скачках, спортивный прогноз, оценка недвижимости, распознавание речи, ранняя диагностика рака.

5.5.2. Входные и выходные данные

Входные и выходные данные могут быть представлены в программе в числовом, символьном видах, а также в виде растровых изображений.

Символьные данные представляются набором строк, расположенных в списке, например, «красный» , «круглый» и т. п. Каждый символ соответствует одному входному нейрону, и присутствие данного символа во входных данных устанавливает этот нейрон во включенное состояние. Символ может быть взвешенным, например, характеризоваться степенью нечеткого включения.

Растровые изображения представляются в виде двумерных массивов точек, каждая из которых соответствует входному нейрону. По сути, это символьное представление двумерного массива данных. Выходной растр, кроме того, формирует «температуру» точек, т. е. способен представлять значимость выходов отдельных нейронов растра.

В одной сети символьные входы могут сочетаться с числовыми, однако, для входов, имеющих тип «растр», никакое сочетание невозможно.

5.5.3. Типы файлов

Основными типами файлов являются файлы определений, файлы фактов и файлы сети. Все они имеют текстовый формат и могут создаваться и редактироваться вручную.

Файл определений содержит всю необходимую информацию о создаваемой сети (количество слоев и нейронов в них, тип входных и выходных данных, представление информации о работе сети на экране, параметры обучения и т. п.). Данный тип файлов используется программой только в процессе первоначального построения сети. По умолчанию файл имеет расширение *.def.

Файл фактов содержит обучающие, тестирующие и рабочие факты, которые будут использоваться созданной сетью. По умолчанию они имеют расширения соответственно *.fct, *.tst, *.in.

Файл сети создается программой в процессе обучения и содержит текущие параметры такие, как веса сети, а также данные из файлов обучающих фактов и определений.

Файлы определений и фактов создаются программой NetMaker или вручную, за исключением случаев, когда входом или выходом является растр точек. В этом случае файл создается только вручную.

Помимо перечисленных основных файлов BrainMaker генерирует множество других типов файлов. Эти файлы могут содержать: выходные данные сети и сопровождающую информацию (*.out), статистику обучения (*.sts) и тестирования (*.sta), файлы отчетов по исследованиям зависимостей (*.rpt), отдельные параметры созданной сети (*.ext).

5.5.4. Создание нейросетевой модели

Непосредственно процессу создания нейросетевой модели предшествует процедура сбора, анализа и обработки исходных данных с целью наиболее адекватного представления моделируемого процесса.

Для предварительной обработки данных в пакете предусмотрена программа NetMaker. Данные экспортируются в нее из файлов табличных и текстовых форматов (dBase, Excel), что позволяет провести предобработку и в других приложениях.

После подготовки обучающей выборки определяются параметры нейросетевой модели. Ниже рассматриваются основные процедуры данного этапа.

Определение количества и размера скрытых слоев. Размерность входного сигнала и число нейронов выходного слоя в многослойных нейронных сетях определяются заданной обучающей выборкой. Определение числа нейронов в скрытых слоях представляет из себя нетривиальную задачу. В настоящий момент не разработано исчерпывающей и однозначной методики для определения количества скрытых слоев и количества нейронов в них. Согласно исследованиям, для задач, которые используют многослойные нейронные сети с алгоритмом обучения обратного распространения ошибки, достаточно не более двух скрытых слоев. При включении в сеть третьего слоя резко возрастает время обучения, а сходимость обучения – падает. В большинстве случаев достаточно одного скрытого слоя, и лишь при полностью тупиковой ситуации, в сеть может быть добавлен еще один слой. При этом результаты предыдущего обучения разрушаются.

Количество нейронов в скрытых слоях сильно зависит от размера обучающей выборки. Например, если обучающая выборка мала, а количество нейронов велико, то сеть начинает запоминать факты, тогда как в задачах прогнозирования требуется обобщение. Обратная ситуация может привести к тому, что сеть никогда не обучится. Для решения задачи о количестве нейронов в скрытом слое и размере обучающей выборки можно использовать методику, приведенную в гл. 1.

В программе BrainMaker по умолчанию количество нейронов в скрытых слоях устанавливается равным количеству входов, но не менее десяти.

Определение коэффициента скорости обучения. Коэффициент скорости обучения определяет степень изменений, которые претерпевают веса сети за один шаг обучения. Таким образом, чем больше этот коэффициент, тем грубее будут подстраиваться веса и тем сильнее сеть будет «рыскать» по поверхности ошибки. В начале обучения коэффициент должен быть велик (около 1) для того, чтобы сеть быстрее спустилась с пика поверхности ошибки. В дальнейшем коэффициент скорости обучения постепенно снижается до уровня 0,1. В программе BrainMaker предусмотрена функция постепенного снижения этого коэффициента в зависимости от степени обученности сети. Кроме того, можно установить линейную или экспоненциальную зависимость убывания коэффициента.

По окончании процесса обучения и тестирования сеть подготовлена к работе. При поступлении новых данных могут потребоваться дообучение и оптимизация модели ввиду заметного расхождения выходов сети и реальных величин. Ниже приводятся основные процедуры оптимизации, некоторые из которых могут быть применены и на этапе первоначального обучения.

Динамическое сокращение ошибки обучения. В процессе обучения нейронная сеть «рыскать» по поверхности ошибки. При этом она часто попадает в локальные минимумы сложной поверхности. Степень точности, которая допускается при сравнении реального выхода сети и обучающей матрицы, сильно влияет на скорость обучения. Чем больше допустимая ошибка, тем больше разброс величин коррекции весов сети от факта к факту. Например, при точности 0,25 и требуемом выходе 1,0 ответ сети 0,76 не вызовет необходимой коррекции, тогда как ответ 0,74 заставит сеть скорректировать веса в соответствии с разностью, равной 0,26, что может заставит сеть «скакнуть» далеко по гиперповерхности ошибки, возможно, в высокий локальный максимум, с которого сеть будет спускаться обратно в течение следующих нескольких циклов обучения. Визуально это можно наблюдать на графике

среднеквадратической ошибки в процессе обучения с высоким допустимым отклонением.

С другой стороны, на начальных этапах обучения, когда обучающая выборка имеет сильный разброс значений, для ускорения сходимости имеет смысл установить низкую точность проверки выхода сети. Далее, когда большинство или все обучающие примеры порождают выходные значения в пределах заданной точности, уровень точности снижают и продолжают процесс.

В BrainMaker существует опция параметров обучения, позволяющая постепенно снижать допустимую ошибку по некоторому закону. Таким образом, на начальном этапе сеть обучается с заданной точностью, а когда выполняется условие адекватности выходы, точность умножается на некоторый понижающий множитель. В результате осцилляция весов сети и, следовательно, средней ошибки постепенно уменьшается.

Контроль паралича сети. В процессе обучения возможен паралич сети, возникающий, когда большинство весов нейронов сети достигает больших значений. Что приводит, в свою очередь, к большим значениям выходов, при которых производные логистических функций крайне малы. Так как посылаемые обратно в процессе обучения ошибки пропорциональна этим производным, то процесс обучения может практически замереть. Обычно паралича сети избегают снижением шага обучения.

В программе BrainMaker предусмотрена функция визуального контроля за распределением весов нейронной сети в виде гистограмм для каждого скрытого и выходного слоев. По вертикали отложено общее количество весов, а по горизонтали – их величина в долях от диапазона. Перед обучением весам придаются случайные малые значения, так что гистограмма имеет форму центрального колокола с максимумом на нулевом значении весов. В процессе обучения веса претерпевают изменения, стремясь «расплыться» по всей гистограмме.

До тех пор, пока веса имеют подобное нормальному распределение, сеть имеет хорошие способности к обучению, ее «познавательные» ресурсы велики. Когда основная масса весов приближается к краям гистограммы, возникает возможность возникновения паралича. Это также сигнализирует о том, что дальнейшее обучение в большинстве случаев бесполезно.

Нет однозначных рекомендаций, что делать в таких ситуациях. Если задача не очень сложна, проще переобучить сеть в новых условиях. Если сеть сложна, следует внимательно проанализировать ее статистику. Как свидетельствует опыт, зачастую сети,

близкие к параличу, способны завершить процесс обучения и успешно функционировать.

В программе также существует опция, способная автоматически уменьшать изменения весов, близких к критическим значениям («тяжелых» весов).

Включение шумов в обучающий процесс. Нейронная сеть в процессе обучения стремиться подобрать некоторую многомерную функцию, удовлетворяющую всем примерам обучающей выборки. При этом с ростом необходимой точности обучения веса сети подгоняются все точнее и точнее. Одновременно с этим снижаются возможности сети по восприятию новых данных. Так, попытки заставить нейронную сеть как можно более точно воспроизводить выходные факты обучающей выборки в большинстве случаев приводят к значительному успеху. Однако впоследствии оказывается, что предъявление сети фактов, которые сеть еще не «видела», порождает большие ошибки. Причем, чем с большей точностью обучена сеть, тем значительней ошибки в тестовых испытаниях.

Таким образом, стремление обучить сеть на выборке из неограниченного и/или непрерывного множества с высокой точностью приводит к запоминанию ею лишь обучающей выборки, а не к желаемому обобщению предъявляемых данных.

Для избежания такой ситуации существует множество методик, основной из которых является внесение шумов в вектор входных величин. Это реализуется случайным непериодическим умножением отдельных величин вектора на малую константу, например, на 0,05. В результате формируется обучающая выборка, где один и тот же факт повторяется крайне редко. Ясно, что сеть никогда не обучиться всем фактам данной выборки с точностью, большей 5%. Однако это резко увеличивает способность сети к обобщению, так что при тестировании на фактах из всего множества подобных количество неправильных ответов резко снижается.

Программа BrainMaker позволяет добавлять шумы как в обучающие, так и тестовые и рабочие примеры.

Случайность процесса обучения и порядок предъявления обучающих примеров. Процесс обучения сети случаен. Даже при абсолютно одинаковых начальных условиях инициализация весов сети перед обучением носит случайный характер. Сильно влияет на процесс обучения порядок предъявления сети обучающих примеров. Для оптимизации результатов можно создать и обучить несколько сетей, отличающихся только порядком обучающей выборки. В результате будет получено несколько различных выходов, которые при усреднении дадут наиболее адекватный результат.

Существует ряд функций пакета для ручной оптимизации полученной модели на основе статистических данных обучения. Ниже рассматриваются эти функции.

Статистика процессов обучения и тестирования. Процесс обучения нейронной сети в программе BrainMaker возможно сочетать со многими контрольными функциями, основной из которых является тестирование сети в процессе обучения. Это означает, что набор обучающих примеров можно разбить в определенной пропорции (по умолчанию, 1/9) на тестирующий и обучающий наборы. С определенной периодичностью программа предъявляет сети набор тестовых примеров, на основе которых оценивается ее умение правильно обрабатывать факты, которых сеть никогда не «видела».

Максимальная частота тестирования – один раз за цикл обучения. Результаты каждого цикла обучения и тестирования могут быть записаны в файлы статистики (*.sts и *.sta). В эти файлы заносится следующая информация: номер цикла, общее число предъявленных примеров, количество правильных и ошибочных фактов, количество ошибочных выходов сети, общее число ошибочных фактов, коэффициент скорости обучения, точность обучения/тестирования, средняя ошибка, среднеквадратическая ошибка, время обучения. Файлы имеют текстовый формат и могут быть обработаны любым текстовым редактором. Помимо этого, файлы могут быть обработаны программой NetMaker. Данная программа позволяет применить к данным тестовых файлов все средства анализа, описанные выше.

Программа BrainMaker имеет также функцию периодического сохранения текущего состояния обучаемой сети (не реже одного раза за цикл). В сочетании с файлами статистики эта функция позволяет выбирать промежуточные состояния сети, удовлетворяющие нужным требованиям, например, минимальная ошибка тестирования, минимальное число ошибочных фактов тестирования.

Изменение скорости обучения вблизи глубокого минимума ошибки. В процессе обучения нейронная сеть проходит множество состояний, которые характеризуются, в частности, количеством ошибочных фактов и ошибкой выхода. Необязательно, что минимальная ошибка соответствует минимальному количеству ошибочных фактов. Более того, поведение указанных характеристик для обучения и тестирования может существенно расходиться. В случае постановки задачи с точки зрения оптимальности результатов тестирования необходимо выбрать такое состояние сети, при котором, например, среднеквадратическая ошибка тестирования

минимальна, а далее обучить сеть более «тонко». Здесь можно рекомендовать следующую методику.

На основе статистики тестирования и обучения выбирается промежуточное состояние сети, при котором фиксируется минимальное количество ошибочных тестовых фактов и малое значение среднеквадратической ошибки. Затем делается предположение, что в данном состоянии сеть находится вблизи глубокого минимума ошибки. Из сохраненного файла состояния извлекаются соответствующие промежуточные параметры сети, снижается коэффициент скорости обучения, задается новая частота тестирования и сохранения промежуточных состояний, и сеть запускается на дальнейшее обучение. В результате снижения скорости обучения нейронная сеть с большей точностью должна обходить рельеф поверхности ошибки, попадая в локальные минимумы, которые ранее «перепрыгивала».

Далее результаты обучения извлекаются и обрабатываются в соответствии с приведенной выше методикой. Если достигнутые оптимальные для данного множества состояний параметры не удовлетворяют с точки зрения точности, процедура повторяется. Таким образом, в соответствии с этой методикой, непрерывно уменьшая шаг обучения, можно достигнуть состояния сети, отражающего глубокий локальный (возможно глобальный) минимум ошибки.

Создание, обучение, тестирование и опрос сети в среде BrainMaker рассмотрим далее на примере уже рассмотренной задачи моделирования логической функции «Исключающее ИЛИ».

1) Подготовка исходных данных.

Подготовим текстовый файл (разделители – табуляции) следующего содержания:

X1	X2	Y
0	0	0
1	0	1
0	1	1
1	1	0

Сохраним его под именем xor.dat. Исходные данные можно подготовить и в других форматах также, как в пакете НейроПро.

Запустим NetMaker, при этом откроется меню программы (рис. 5.45). Выберем опцию **Read in Data File** (Читать файл данных). В окне диалога выберем и откроем созданный файл xor.dat. Выберем далее опцию **Manipulate Data** (Преобразование данных). В окне программы появится таблица, представленная на рис. 5.46.

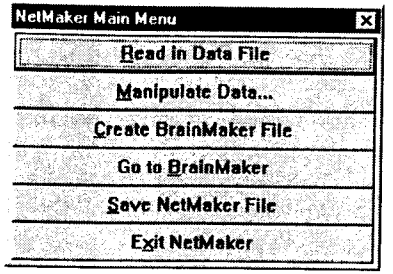


Рис. 5.45. Основное меню программы NetMaker

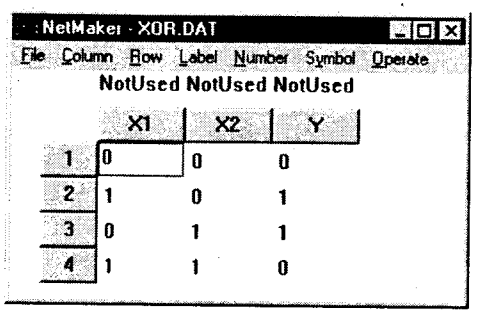


Рис. 5.46. Окно преобразования данных утилиты NetMaker

Нажмем мышью кнопку X1 и далее выберем из меню окна пункты **Label/Mark Column as Input** (Пометить столбец как вход). То же самое проделаем со столбцом X2, а столбец Y с помощью команды **Label/Mark Column as Pattern** (Пометить столбец как выход) укажем как выходной. В результате таблица преобразуется к виду рис. 5.47.

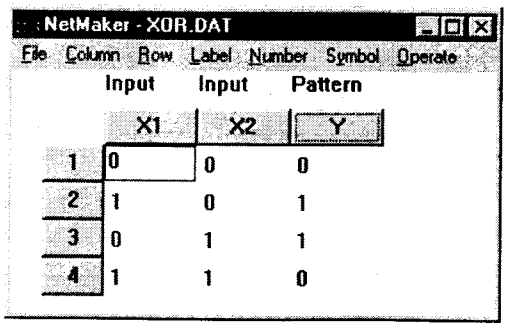


Рис. 5.47. Исходные данные после преобразования

Теперь с помощью команды **File/Create BrainMaker Files** перейдем к диалоговому окну вида рис. 5.48.

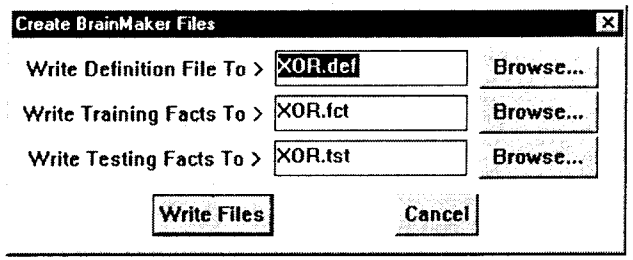


Рис. 5.48. Окно подтверждения записи образованных файлов

В нем появится сообщение, что будут подготовлены три файла BrainMaker с расширениями, соответственно, *.def, *.fct и *.tst. Согласимся с этим, нажав кнопку **Write Files** (Записать файлы), и выйдем из программы, выбрав команду **File/Exit**.

2) Задание структуры сети и параметров ее обучения.

Запустим программу BrainMaker, в меню рабочего окна программы выберем команду **File/Read Network** (Файл/Читать сетевой файл) и далее в окне диалога укажем и откроем файл xor.def. Окно нейропакета примет при этом вид рис. 5.49.

В верхней части окна можно выделить следующую информацию: открыт файл xor.fact (данные обучающей выборки во внутреннем формате BrainMaker), коэффициент скорости обучения (Learn) – 1, допустимый уровень ошибки (Tolerance) – 0,1. Следующая строка в верхней части окна описывает процесс обучения, и пока он не начат, все параметры в строке установлены нулевыми. Далее в окне указаны имена входных переменных (X1 и X2), выхода сети (Y, Out) и соответствующего образца (Y, Ptn) обучающей выборки.

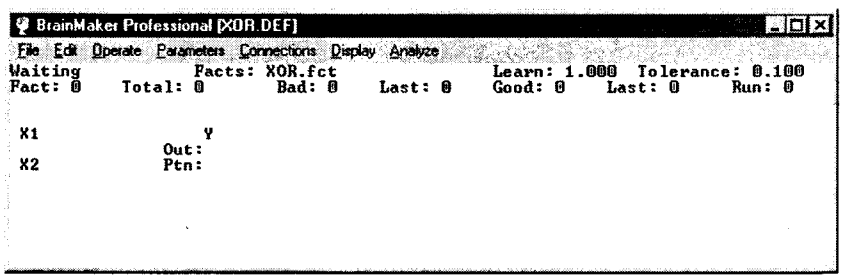


Рис. 5.49. Рабочее окно BrainMaker после открытия сетевого файла

Установим требуемую структуру сети, выбрав команду меню **Connections/Network Size**, и далее в появившемся диалоговом окне укажем 2 нейрона в первом (скрытом) слое (рис. 5.50) с подтверждением этого.

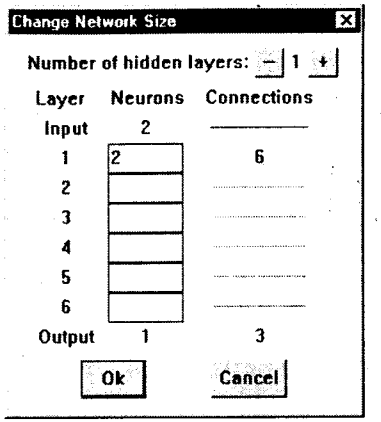


Рис. 5.50. Диалоговое окно задания структуры нейронной сети

Теперь можно либо скорректировать параметры обучения (через опции меню **Parameters**), либо, согласившись с их величинами по умолчанию, перейти к процессу обучения.

3) Обучение сети.

Выберем команду меню **Operate/Train Network**, запускающую процесс обучения. Спустя некоторое время процесс обучения остановится, и окно программы примет вид рис. 5.51.

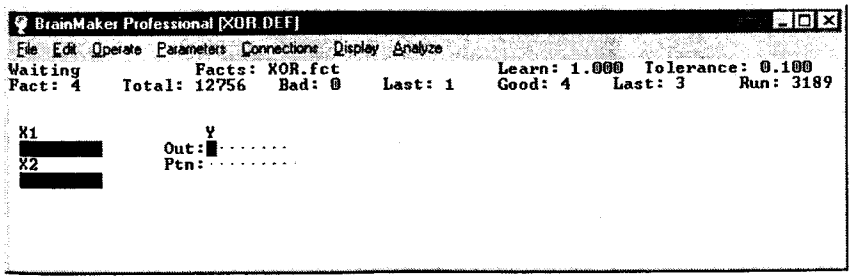


Рис. 5.51. Окно нейропакета после завершения процесса обучения

Заметим, что нейропакет предоставляет достаточно широкие возможности контроля качества обучения.

4) Опрос сети.

В приведенном окне значения входов, выхода сети и образца представлены в графической форме в виде столбиков («термометров»). Мышью можно регулировать их длину, тем самым плавно меняя значения входов и наблюдая реакцию обученной сети (Y, Out). Для представления данных в числовой форме, необходимо выбрать команду меню **Display/Edit Network Display**, после чего в появившемся диалоговом окне (рис. 5.52) задать опцию **Number** (Число).

После этого можно изменять числовые значения входов, щелкая мышью по соответствующему входу и далее устанавливая его требуемое числовое значение. Выход сети при этом также будет отображаться в числовой форме.

5) Сохранение результатов.

Для сохранения результатов достаточно выбрать команду меню **File/Save Network** и далее согласиться с предлагаемом именем сохраняемой сети (в нашем случае xor.net).

На самом деле команд меню нейропакета достаточно много, что создает достаточно комфортную среду для пользователя, но для понимания работы, как представляется, приведенной информации вполне достаточно.

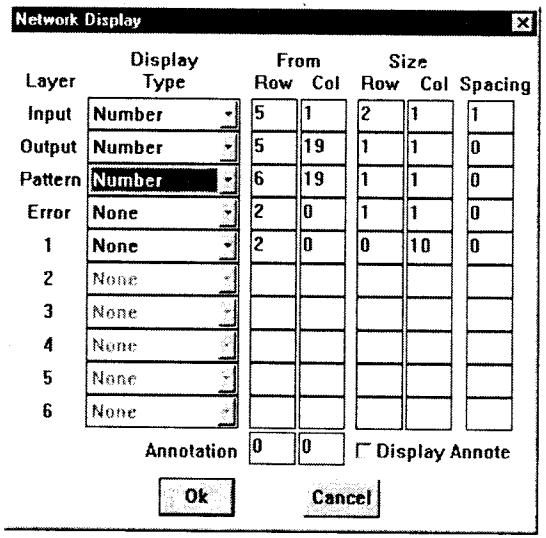


Рис. 5.52. Диалоговое окно задания вида отображения информации

5.5.5. Общее суждение о нейропакете

Процесс нейросетевого моделирования в среде BrainMaker производится полностью вручную, что при создании моделей средней сложности приводит к уже значительным трудозатратам. Кроме того, интерфейс пакета построен таким образом, что при длительной работе с моделью пользователь способен запутаться в последовательности обработки данных, что может привести к разрушению модели.

Пакет BrainMaker Professional обладает следующими характеристиками, делающими возможным создание на основе его интегрированных программных систем:

- открытая файловая система: вся информация, необходимая для построения, обучения, тестирования, оптимизации, анализа, редактирования и запуска нейросетевых моделей, создаваемых программой, доступна для прочтения и изменения в текстовом режиме и жестко форматирована; правила работы с ней легко формализуемы;
- исчерпывающий набор команд обучения, тестирования, анализа и запуска нейронных сетей из командной строки;
- возможность генерации кода для включения обученных сетей в программы пользователя.

К недостаткам можно отнести отсутствие полной документации по структуре внутреннего формата бинарных файлов программы. Другие достоинства и недостатки программы указаны в гл. 3.

5.6. Нейропакет MPIL

5.6.1. Общая характеристика

Пакет моделирования нейронных сетей MPIL (Multi-Pass Instance-Based Learning) представляет собой разработку фирмы Universal Problem Solvers, Inc.. Его демо-версия доступна в Интернет по адресу: http://www.simtel.net/simtel.net/win3/neural-pre.html.

Нейропакет работает под Windows и крайне нетребователен к вычислительным ресурсам. Хотя программа декларируется как нейропакет, на самом деле она не создает нейронную сеть в обычном понимании и не использует известные алгоритмы обучения. Принцип действия пакета основан на идеях кластерного анализа: на основе многократного перебора обучающей выборки формируется набор опорных элементов, с помощью которых мож-

но объяснить свойства всех остальных. Работа пакета возможна в двух режимах.

В первом режиме (MPIL–1) реализуется метод «*M* ближайших соседей»: для каждого предъявляемого входного вектора находятся *M* ближайших (например, в смысле евклидова расстояния) опорных векторов, ассоциированных с эталонными выходными, по которым и рассчитывается выход сети.

Во втором режиме (MPIL–2) пакет дополнительно генерирует логические правила причинно-следственных отношений между примерами обучающей выборки, т. е. позволяет извлекать знания из экспериментальных данных.

В этом, а также в очень быстром, практически мгновенном, обучении и состоят уникальные свойства пакета.

5.6.2. Интерфейс программы

После запуска программы появляется ее окно вида рис. 5.53.

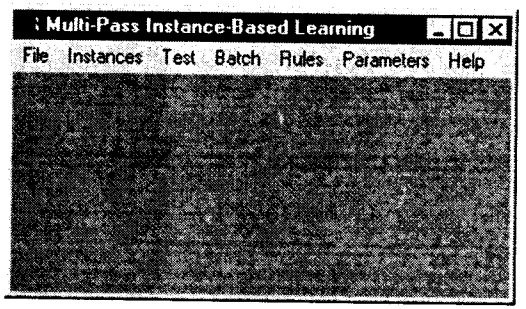


Рис. 5.53. Окно программы MPIL

Пункты и опции меню в верхней части окна следующие.

Пункт **File** (Файл) имеет только две опции: **Edit** (Редактировать) и **Exit** (Выход из программы).

Пункт **Instances** (Примеры) содержит четыре опции: **Create** (Создать), **Load** (Загрузить), **Save** (Сохранить), **Train** (Обучать).

Пункт **Test** (Тестирование) имеет две опции: **Check Accuracy** (Проверить точность), **Classify** (Классифицировать).

Также две опции содержит пункт меню **Batch** (Пакет): **Random Testing** (Случайное тестирование), **Training** (Обучение).

Опции пункта **Rules** (Правила) таковы: **Extract** (Извлечь), **Import** (Импортировать).

Наибольшее количество опций содержит пункт меню **Parameters** (Параметры):

- **Learn Mode** (Режим обучения) – при выборе опции появляется диалоговое окно (рис. 5.54), позволяющее установить требуемый режим обучения для MPIL–1 или MPIL–2.

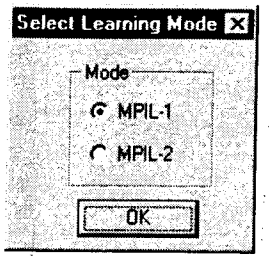


Рис. 5.54. Диалоговое окно для задания режима обучения

- **Distance Measure** (Мера расстояния) – мера сходства между примерами выборки; обращение к опции приводит к появлению диалогового окна вида рис. 5.55, позволяющего задать одно из трех возможных расстояний: Hamming (хэммингово), Euclidean (евклидово) или Square (модуль наибольшей разности между элементами двух векторов).

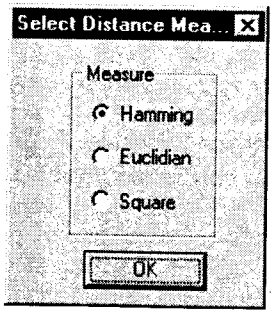


Рис. 5.55. Диалоговое окно для задания меры расстояния

- **Display Info** (Информация дисплея) – в случае установки флага на опции будет отображаться дополнительная информация во время подготовки, тестирования и других операций MPIL.
- **Show Files** (Показать файлы) – при установке флага на этой опции процессы обучения, тестирования и работы сети завершаются показом соответствующих файлов.
- **Learning** (Обучение) – при выборе этой опции появляется диалоговое окно вида рис. 5.56, позволяющее установить требуемые параметры обучения.

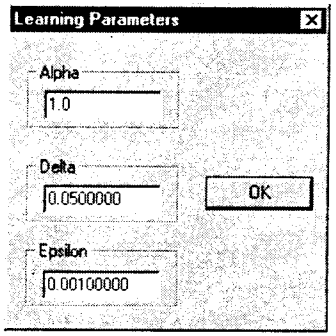


Рис. 5.56. Окно установки параметров обучения

- **Num. Batch Runs** (Количество прогонов пакета) – используется вместе с опцией **Random Testing** (Случайное тестирование) пункта меню **Batch** для организации случайного перекрестного тестирования сети.
- **Setup** (Установка) – имеет, в свою очередь, две опции: **Load** и **Save**, позволяющие загрузить или сохранить конфигурацию программы.

Наконец, пункт меню **Help** (Справка) содержит опцию запуска достаточно подробного файла справки и опцию запуска режима регистрации.

5.6.3. Правила работы с программой

Правила работы с программой рассмотрим на примере моделирования логической функции «Исключающее ИЛИ».

1) Подготовка исходных данных.

Подготовим два текстовых файла:

```
#
Ninputs 2 Noutputs 1
NTrainingPatterns 4
0.0000 0.0000  0.0000
1.0000 0.0000  1.0000
1.0000 0.0000  1.0000
1.0000 1.0000  0.0000

NTrainingPatterns 4
0.0000 0.0000  0.0000
1.0000 0.0000  1.0000
1.0000 0.0000  1.0000
1.0000 1.0000  0.0000
```


и сохраним их с именами xor.net и xor.tes. Первый файл будет использоваться для обучения, второй – для тестирования нейронной сети. Обязательная информация таких файлов содержится в их первых строках. Это: количество входов (Ninputs), в нашем случае – 2, количество выходов (Noutputs) – 1 и общее число примеров обучающей выборки (NTrainingPatterns) – 4.

2) Обучение сети.

Запустим программу MPIL, выберем пункты меню **Instances/Create** и в окне диалога откроем файл xor.net. Появится сообщение вида рис. 5.57, подтверждающее создание примеров. Нажмем кнопку **OK**.

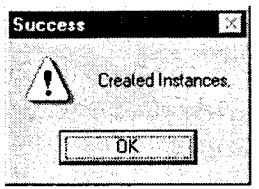


Рис. 5.57. Окно подтверждения создания примеров

Оставим большинство параметров обучения по умолчанию, а с помощью пунктов меню **Parameters/Learn Mode** выберем режим обучения MPIL–2.

Далее пунктом меню **Instances/Train** запустим процесс обучения. Появится окно (рис. 5.58) с информацией об обучении нейронной сети.

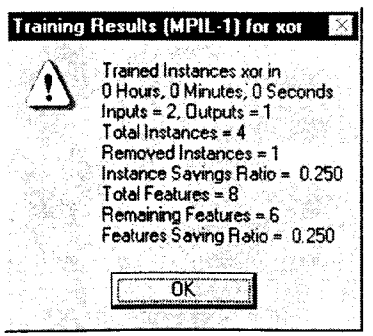


Рис. 5.58. Информация об обучении нейронной сети

Нажмем кнопку **OK**, после чего откроется блокнот Windows с информацией вида рис. 5.59.

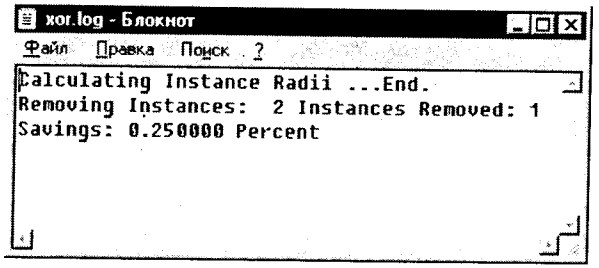


Рис. 5.59. Дополнительная информация об обучении нейронной сети

Данную информацию можно принять к сведению, после чего блокнот следует закрыть и перейти к тестированию выбором команд **Test/Check Accuracy** с указанием имени подготовленного тестового файла xot.tes и выходного файла (файла протокола) xor.out. Выполнив указанные действия, получим сообщение вида рис. 5.60, из которого следует, что тестирование успешно завершено.

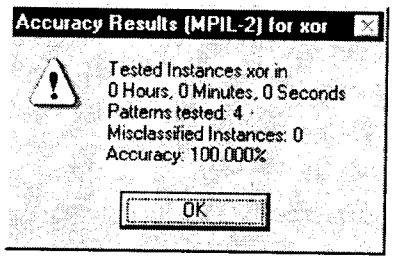


Рис. 5.60. Информация о результатах тестирования

Подтвердим это нажатием кнопки **OK**, в ответ на что получим открытую страничку блокнота (рис. 5.61). Закроем эту страницу и проверим, как программа извлекает знания. Выберем команду **Rules/Extract**, укажем в качестве имени файла протокола xor.rul. После выполнения команды получим информацию вида рис. 5.62.

Что здесь что? Через Feature_0 и Feature_1 обозначены входы, через Output – выход. Видно, что программа сгенерировала три правила, из которых первое (если X1 < 1, то Y = 0) и второе (если X1 > 0 и X2 < 1, то Y = 1) представляются корректными, а третье (если X2 > 0, то Y = 0) – не очень. Или мы что-то не так сделали, или программа все же не вполне в этом аспекте надежна. Может быть, что-то необходимо было изменить в параметрах обучения.

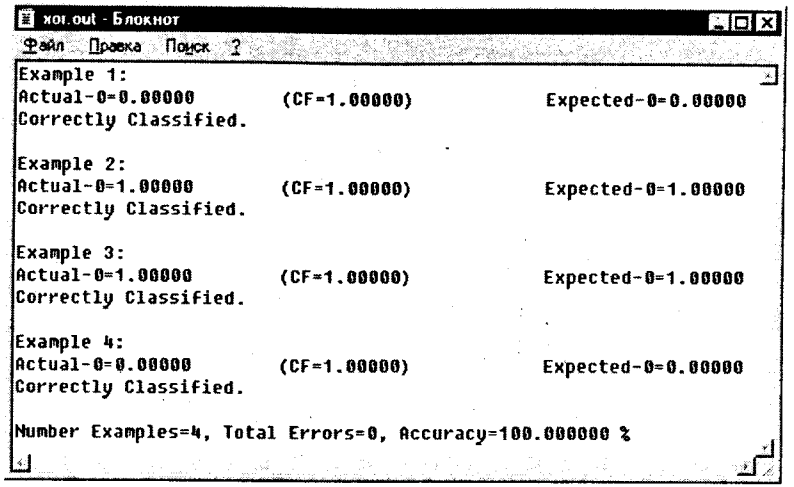


Рис. 5.61. Детальная информация о результатах тестирования

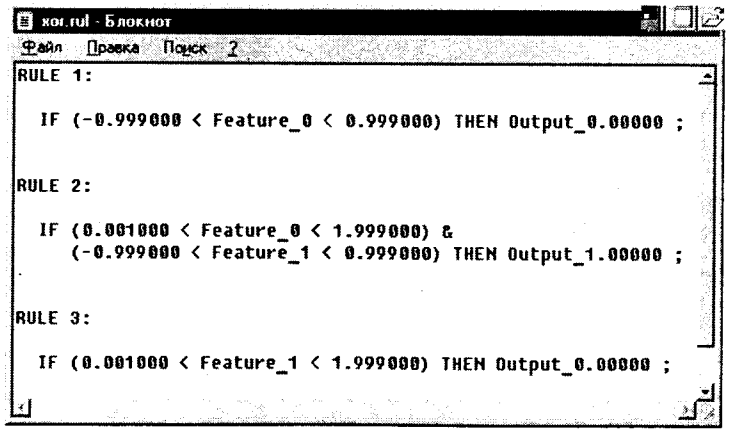


Рис. 5.62. Правила, извлеченные программой MPIL

Теперь можно закрыть блокнот и сохранить полученную сеть через пункты меню **Instances/Save** с указанием имени файла, например, xor.sav. На этом работа с программой закончена. При желании можно с помощью какого-либо текстового редактора просмотреть образованные программой файлы.

5.6.4. Впечатления от работы с пакетом

Подкупает простота программы и возможность автоматического извлечения знаний. Проверка на ряде примеров показала достаточно хорошие моделирующие возможности пакета MPIL.

5.7. Нейропакет Braincel

5.7.1. Общая характеристика

Программа Braincel разработана фирмой Promised Land Technologies, Inc. (адрес в Интернет демо-варианта программы http://promland.com/demo.htm) и представляет собой надстройку для табличных процессоров Excel, позволяющую реализовывать нейронную сеть прямого распространения с одним или двумя скрытыми слоями.

Braincel использует улучшенный алгоритм обучения BackPerc (BackPercolation – обратное процеживание), который позволяет проводить обучение сети примерно в 100 раз быстрее, чем стандартный алгоритм Backpropagation. Особенностью программы является также ее возможность оперировать не только с числовыми, но и с символьными (текстовыми) данными. Наконец, ее интегрированность с Excel (рис. 5.63) позволяет использовать все графические возможности табличного процессора.

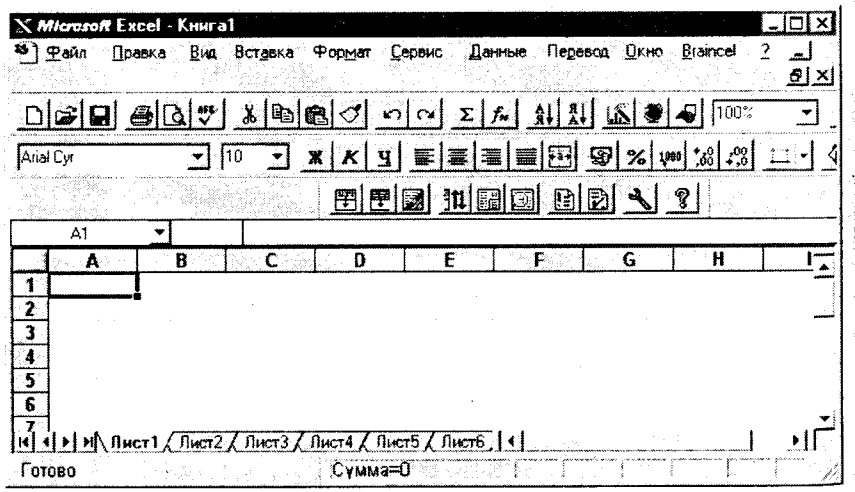


Рис. 5.63. Окно Excel с встроенной надстройкой Braincel

Процесс встраивания Braincel прост: после запуска Excel вызывается опция меню **Сервис/Надстройки**, далее нажимается кнопка **Обзор**, и находится надстройка Braincel (файл braincel.xll), если она имеется в какой-либо директории. При следующем запуске Excel в меню добавляется пункт Braincel с двумя опциями: **Braincel Menu** (Меню Braincel) и **Close Braincel** (Закрыть Braincel).

5.7.2. Интерфейс программы

Для описания и изучения интерфейса надстройки, выберем опцию меню **Braincel/Braincel Menu**. После этого главное окно примет вид рис. 5.64.

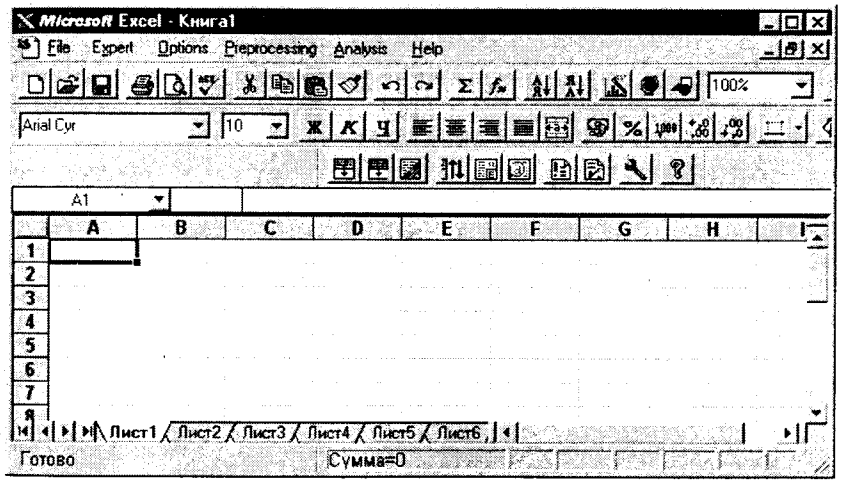


Рис. 5.64. Окно Excel при работе с надстройкой Braincel

Меню программы Excel заменяется при этом на меню надстройки. Пункты данного меню следующие.

Пункт **File** (Файл) имеет подпункты: **New Expert** (Новый эксперт – новая нейронная сеть); **Open Expert** (Открыть эксперта); **Save** (Сохранить); **Save As ...** (Сохранить как ...); **Return to Excel** (Возвратиться в Excel).

Пункт **Expert** (Эксперт) содержит 11 опций: **New Train** (Новое обучение); **Continue Train on Prior Data** (Продолжать обучение на прежних данных); **Continue Train on New Data** (Продолжать обучение на новых данных); **Randomize & Retrain** (Случайное перемешивание и переобучение); **Search For Best Net** (Поиск лучшей сети); **Ask Expert** (Опрос нейронной сети); **Ask Multiple Experts** (Опрос ряда обученных нейронных сетей); **Enable Ask Link** (Уста-

новить связь опроса); **Reset Counter** (Сброс счетчика); **Display Weights** (Показать веса); **Reload Weights** (Перезагрузить веса).

Пункт **Options** (Опции) имеет подпункты: **Setup** (Установки); **Notes** (Заметки); **Status** (Состояние).

Три подпункта имеет пункт **Preprocessing** (Предобработка): **Chart Inputs** (Диаграмма входов); **Random Select And Move** (Случайное перемешивание и перемещение); **Cross Multiply** (Перекрестное перемножение).

Столько же подпунктов имеет и пункт **Analysis** (Анализ): **Chart Outputs** (Диаграмма выходов); **Isolate Bad Predictors ...** (Изолировать плохие прогнозы ...); **Measure Percent Correct ...** (Определить процент правильных ...).

5.7.3. Правила работы с пакетом

Работу с пакетом рассмотрим на примере изученной ранее задачи моделирования логической функции «Исключающее ИЛИ».

1) Подготовка исходных данных.

Запустим Excel и с его помощью подготовим таблицу, отображающую функцию «Исключающее ИЛИ» (рис. 5.65), после чего перейдем к меню Braincel (см. выше).

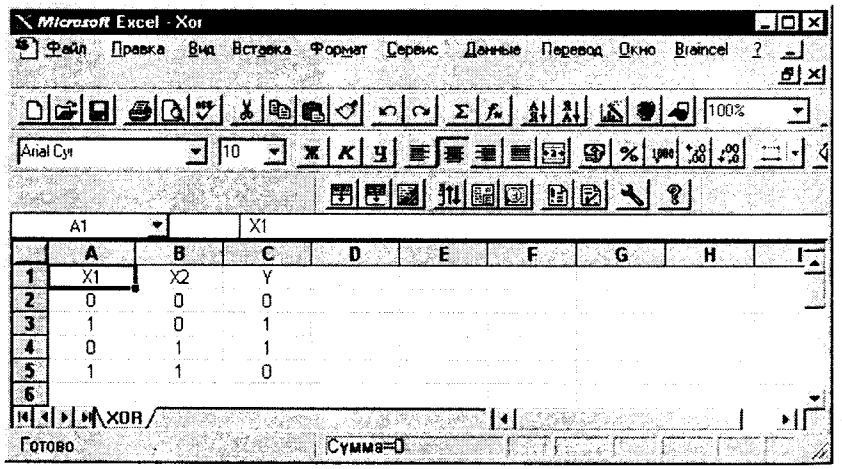


Рис. 5.65. Окно Excel с таблицей функции «Исключающее ИЛИ»

2) Подготовка структуры новой нейронной сети.

Выберем опции меню **File/New Expert**. В появившемся окне вида рис. 5.66 укажем имя создаваемой нейронной сети, количество входов и выходов сети, в данном случае, соответственно, 2 и 1.

Появятся подсказки программы (рис. 5.66):

- рекомендуем 1 скрытый слой с 2 нейронами;
- основываясь на размерности сети, рекомендуем обучающую выборку из 24 образцов и тестовую из 9 образцов.

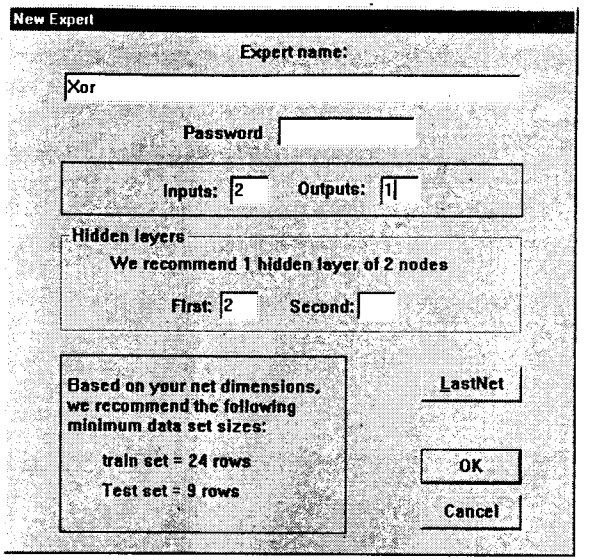


Рис. 5.66. Диалоговое окно создания структуры нейронной сети

Согласимся с предложенной структурой сети, вторую же рекомендацию выполнить принципиально невозможно. Нажмем кнопку **OK**. Появится диалоговое окно (рис. 5.67), позволяющее задать тип выходных нейронов: бинарный (для решения задач классификации), линейный (в случай непрерывного выхода) или логарифмический (промежуточный). Выберем линейный тип.

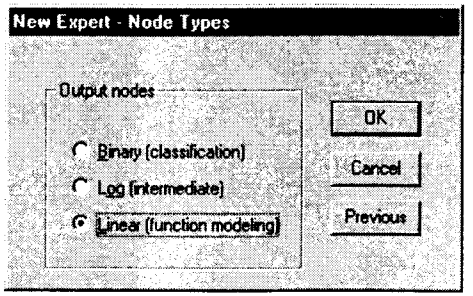


Рис. 5.67. Окно задания типа выходных нейронов

3) Обучение сети.

Выберем опцию меню **Expert/New Train**. В появившемся окне (рис. 5.68) укажем имя рабочего листа (XOR.XLS) и диапазон ячеек таблицы, используемых для обучения (зададим область ячеек A2:C5). Нажмем кнопку **OK**.

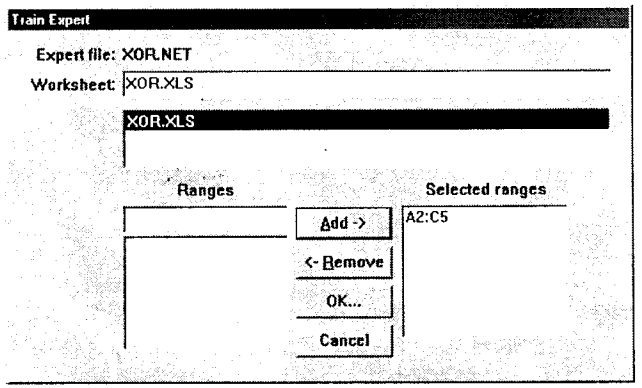


Рис. 5.68. Задание области данных для обучения сети

В ответ появится небольшое сообщение (рис. 5.69), рекомендующее использовать обучающую выборку большего объема. При нажатии кнопки **OK**, появится диалоговое окно, позволяющее задать опции обучения (рис. 5.70).

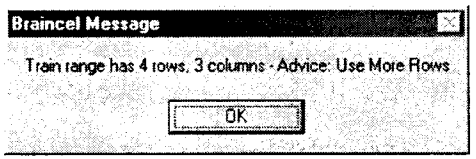


Рис. 5.69. Сообщение о необходимости обучающей выборки большего объема

Согласимся с предложенным и подтвердим это нажатием кнопки **OK**. То же сделаем и в следующем окне (рис. 5.71).

После этого начнется процесс обучения сети, свидетельством чему будет появление меняющихся цифр в нижней части основного окна программы. Достаточно быстро сеть обучится.

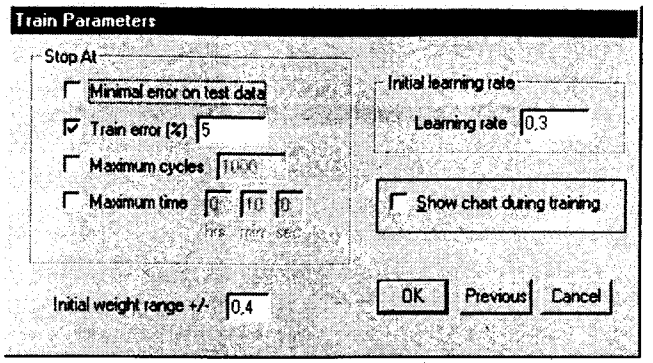


Рис. 5.70. Окно задания опций процесса обучения

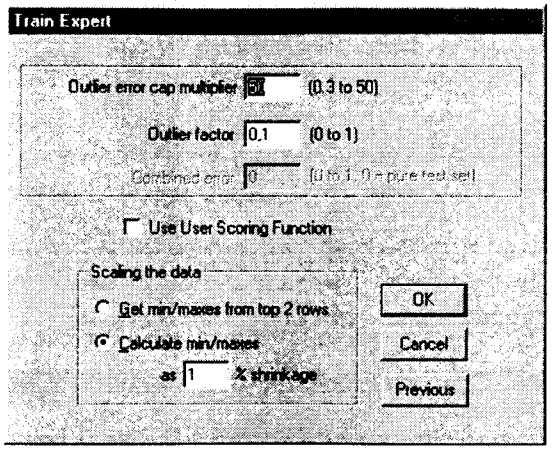


Рис. 5.71. Окно задания дополнительных опций процесса обучения

4) Опрос сети.

Выберем опцию меню **Expert/Ask Expert**. Появится диалоговое окно, идентичное приведенному на рис. 5.68. Зададим в нем значения, приведенные на рис. 5.72, после чего появится окно задания вывода результата (рис. 5.73).

Согласимся с предлагаемой по умолчанию стандартной формой и нажмем кнопку **OK**. Искомый результат появится практически мгновенно (рис. 5.74). Обратим внимание на то, *где именно* выведен ответ сети и сравним это с указанием диапазона ячеек на рис. 5.72.

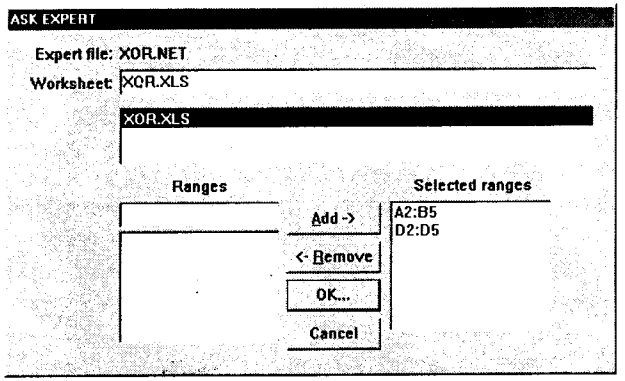


Рис. 5.72. Окно задания области расчета и вывода результата

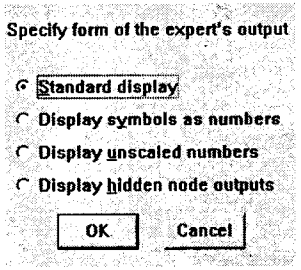
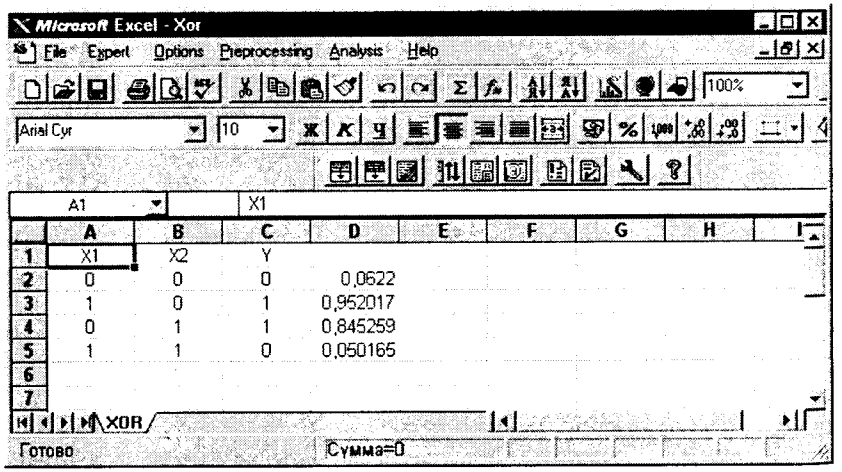


Рис. 5.73. Окно задания формата результата



	A	B	C	D
1	X1	X2	Y	
2	0	0	0	0,0622
3	1	0	1	0,952017
4	0	1	1	0,845259
5	1	1	0	0,050165

Рис. 5.74. Результат опроса нейронной сети

5) Сохранение результатов.

Здесь все осуществляется стандартным образом, через опции меню **File/Save** или **File/Save As ...** .

5.7.4. Дополнительные возможности

Надстройка Braincel предоставляет пользователю несколько поистине уникальных дополнительных возможностей: во-первых, определение значимости входов, во-вторых, возможность нахождения сети наилучшей структуры, в-третьих, использование для прогноза нескольких обученных сетей одновременно. Не останавливаясь детально на этих возможностях, отметим, что они реализуются через пункт меню **Expert**, требуют использования не только обучающей, но и тестовой выборки.

Стоит указать, что в пункте меню **Analysis** можно, после обучения сети, выбрать опции графического представления результатов.

5.7.5. Достоинства и недостатки программы

К несомненным достоинствам программы следует отнести ее интегрированность с Excel, простоту в обращении, отмеченные дополнительные возможности, а к недостаткам – ограниченные функциональные возможности (к слову сказать, у авторов выбор опций пункта меню **Analysis** все время приводил к появлению сообщения об ошибках в числовом формате – демо-версия есть демо-версия).

5.8. Нейропакет Excel Neural Package

5.8.1. Общая характеристика

Пакет Excel Neural Package разработан фирмой «НейрОК» (119899, Москва, Воробьевы горы, Научный Парк МГУ, 5 – 529, e-mail: info@neurok.ru, адрес в Интернет: www.neurok.ru; демонстрационную версию программы можно получить на сервере www.download.ru в разделе «Образование, наука, техника – Научно-технические программы») и расширяет функциональные возможности Excel, предоставляя в распоряжение пользователя алгоритмы обработки данных на основе теории нейронных сетей. Технически семейство продуктов реализовано как набор надстроек

(add-ins) над Microsoft Excel и не предъявляет особых требований к оборудованию.

Excel Neural Package состоит из двух независимых компонентов: Winnet и Kohonen Map.

Программа Winnet реализует многослойный персептрон и предназначена для поиска и моделирования скрытых зависимостей в больших массивах численных данных, для которых в явном виде аналитические зависимости не известны. Характеристики Winnet 3.0 приведены в табл. 5.2.

Таблица 5.2

Основные характеристики Winnet 3.0

Преобразования входных данных	Линейное, сигмоидальное, нормировка на среднее/дисперсию, нормировка на минимум/максимум
Возможность выявления релевантных переменных до стадии обучения нейронной сети	Обеспечивается благодаря применению энтропийного анализа (алгоритм Boxcounting)
Число слоев нейронной сети	до 5
Число нейронов в каждом слое	Зависит от количества свободной памяти
Используемые функции активации	Линейная, гиперболический тангенс
Возможность использования нейронов высоких порядков	Позволяет строить более сложные модели данных
Используемые алгоритмы обучения	Backpropagation с адаптивным выбором параметров обучения Rpropagation Алгоритмы обучения подключаются как внешние модули, поэтому их число не фиксировано

Winnet 3.0 имеет удобный графический интерфейс и обладает хорошими возможностями контроля за процессом обучения:

- отображение в процессе обучения графиков ошибок обучения и обобщения;
- отображение диаграмм рассеяния «реальное значение – предсказанное нейронной сетью» для каждого выхода сети;
- выбор тестового множества;
- остановка обучения по достижении различных критериев останова.

Kohonen Map представляет собой программный инструмент для построения и анализа самоорганизующихся карт Кохонена. Среди основных применений следует выделить задачи кластеризации и визуализации многомерной информации. Пользователь

может представить весь массив данных в виде двумерной цветной карты и визуализировать на ней интересующие его характеристики. Характеристики Kohonen Map 1.0 приведены в табл. 5.3.

Таблица 5.3

Основные характеристики Kohonen Map 1.0

Преобразования входных данных	Линейное, сигмоидальное, нормировка на среднее/дисперсию, нормировка на минимум/максимум
Дополнительная предобработка входных данных с целью понижения размерности	Метод главных компонент (Principal Components Analysis)
Размер карты Кохонена	Ограничен размерами свободной памяти
Число градаций цвета на карте	От 2 до 10
Возможность изменения цветовой палитры под конкретные задачи пользователя	Для улучшения наглядности при просмотре и выводе на печать
Возможность поиска по карте	Позволяет быстро находить положение любого примера на карте и просматривать содержимое ячеек карты

5.8.2. Установка нейропакета

Установка пакета требует выполнения следующих действий.

1) Создать директорию для файлов пакета. Например: C:\Program Files\Microsoft Office\ENP.

2) Скопировать в эту директорию все файлы из директории ...\Neural tools package.

3) Запустить Excel.

4) Выбрать **Меню/Сервис/Надстройки**. В всплывающем окне **Надстройки** нажать кнопку **Обзор**. Далее в открывшемся окне выбрать в созданной папке файл NPackage.xla и нажать кнопку **ОК**.

5) В списке окна **Надстройки** появится новый пункт **Neural tools Package**. Выбрать его и нажать кнопку **ОК**.

6) В открывшемся далее окне **Authorization checker** в поле ввода ввести персональный код (который содержится, например, в файле pwd) и нажать кнопку **ОК**.

7) В левом верхнем углу экрана появится панель инструментов **Neural Analysis**, на которой расположены кнопки доступных в данной версии Excel Neural Package нейроинструментов.

5.8.3. Работа с пакетом Winnet

Возможны два варианта работы с пакетом, во-первых, данные вводятся впервые, во-вторых, обученная ранее нейронная сеть используется для работы с новыми данными.

В первом варианте необходимо обучить нейронную сеть на заданном наборе данных. Сеть построит аппроксимацию многомерных данных, автоматически подбирая веса нейронов. Для этого необходимо сделать следующее:

- загрузить данные из книги Excel в систему;
- определить, что является входной информацией, а что – выходной;
- предобработать данные, т. е. осуществить их нормировку;
- оценить значимость входов для выходной информации и, если необходимо, изменить (удалить/добавить) входы;
- создать нейронную сеть;
- обучить нейронную сеть на заданном множестве примеров и оценить работу на тестовом множестве;
- сохранить предсказанные (обработанные) данные в книге Excel;
- сохранить обученную нейронную сеть для дальнейшей работы.

Разберем эти действия по шагам.

1) Для работы с данными выделите область на листе книги Excel. Данные на листе располагаются следующим образом: входы и выходы – столбцы, строки – обучающие примеры.

2) Щелкните мышью по кнопке с изображением нейрона на панели инструментов **Neural Analysis**. Появится диалоговое окно **Select data source**, предлагающее уточнить параметры области данных для работы (рис. 5.75).

3) В случае согласия с параметрами ввода нажмите **OK**.

4) Откроется основное окно Winnet 3.0. Оно содержит 4 табулированных листа **Data**, **Network**, **Training** и **Output**, ассоциированных с общими этапами работы (рис. 5.76).

5) Открывшийся лист **Data** позволяет определить и предобработать данные для последующего использования. Кроме того, с этого листа можно сохранить обученную нейронную сеть (**Save Project ...**) или загрузить уже сохраненный в прошлом проект (**Load Project ...**). Первое, что нужно сделать, это определить входы, для чего следует нажать кнопку **Select Inputs...**.

6) В открывшемся диалоговом окне **Select Inputs/All Data** выберете необходимые входы и с помощью кнопки **>>** переведите

их в окно списка **Inputs**. Корректировку выбранных входов можно провести, используя кнопку **<**.

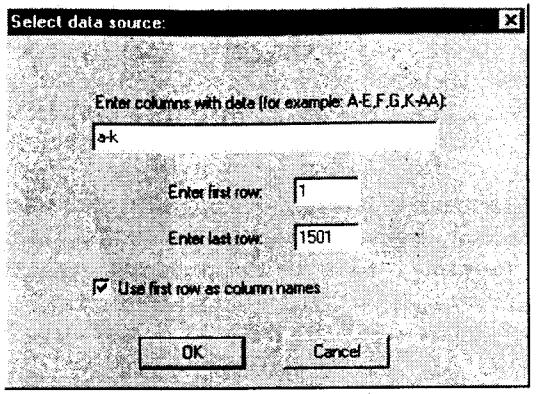


Рис. 5.75. Окно определения данных

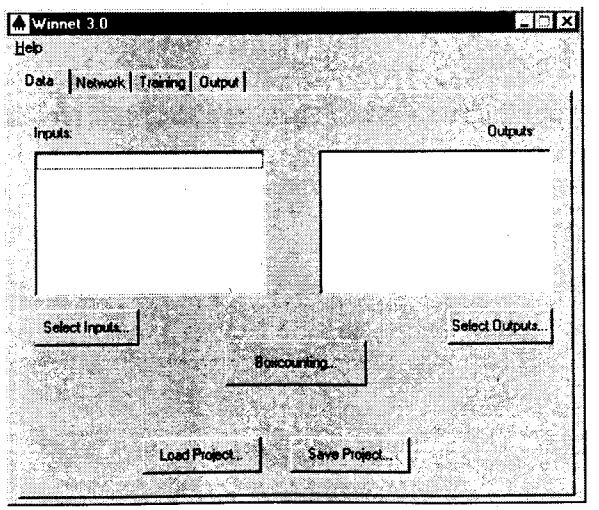


Рис. 5.76. Основное окно Winnet 3.0

7) Так как конкретные значения входов могут быть любыми, то рекомендуется их нормализовать. Для большинства случаев подходит нормировка входных значений **Mean/Variance**. При этом данные переводятся в безразмерную форму вычитанием среднего и нормированием на их дисперсию. Другие способы: линейное преобразование, приведение к диапазону (–1, +1) ([–1, +1] nor-

malization) и нелинейное преобразование гиперболическим тангенсом (than-normalization). Нажмите кнопку **Normalization...** и выберите в открывшемся окне **Inputs normalization** соответствующую позицию переключателя. Подтвердите выбор нажатием кнопки **OK**. Вернувшись в окно **Select Inputs** также подтвердите выбор нажатием кнопки **OK**.

8) Далее, нажав на кнопку **Select Outputs...**, аналогично пунктам 6 и 7 определите и нормализуйте, при необходимости, выходы системы.

9) Определение значимость входной информации для предсказания выходной начинается с нажатия кнопки **Boxcouting...**, по которому система, используя алгоритм Boxcouting, определит статистическую значимость входов для заданных выходов. В открывшемся окне **Boxcouting results** (рис. 5.77) представлена результирующая информация. Причем:

- чем выше столбец гистограммы для данного входа, тем значимее его информация;
- чем больше отношение Mean predictability/Variance отличается от 1, тем большего успеха можно достичь в предсказании выходной информации.

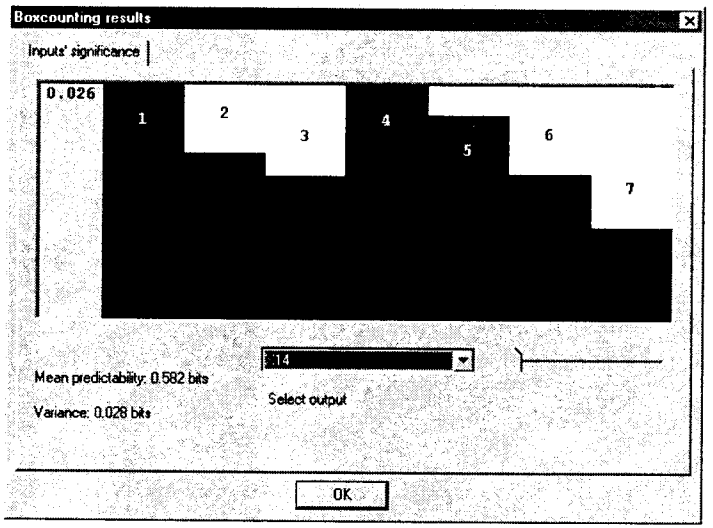


Рис. 5.77. Результат отбора наиболее существенных переменных

Не имеет смысла использовать для предсказания входы, значимость которых равна или близка к нулю. Поэтому, вернувшись в основное окно, удалите эти входы из списка Inputs. Данную

процедуру по определению значимости входов можно повторить несколько раз. Дело в том, что уменьшение количества входов позволит либо существенно сократить время обучения нейронной сети, либо даст возможность улучшить ее нелинейные свойства.

10) Следующий этап – создание нейронной сети (многослойного персептрона). Перейдите на закладку **Network** и нажмите на кнопку **Create Net...**. В открывшемся окне **Network Construction** (рис. 5.78) можно полностью определить структуру и топологию нейронной сети:

- количество слоев;
- тип активационной функции нейронов данного слоя;
- число нейронов в слое;
- порядок нелинейности нейронов данного слоя.

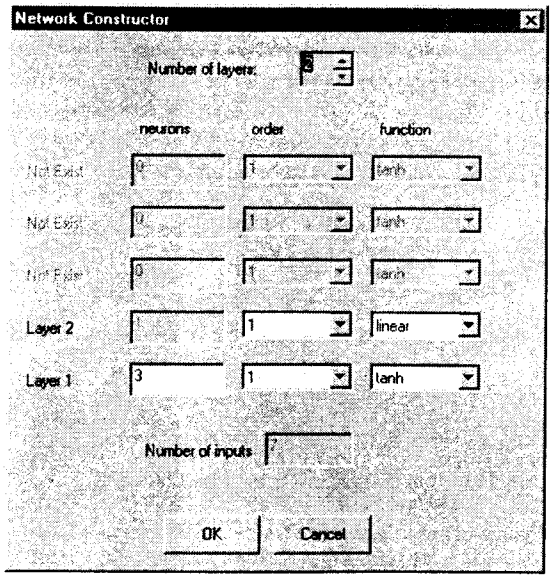


Рис. 5.78. Окно конструирования нейронной сети

При создании нейронной сети следует руководствоваться следующими простыми соображениями.

Во-первых, для хорошей (гладкой) аппроксимации данных общее число связей сети должно быть в несколько раз (лучше на порядок) меньше числа обучающих примеров. В противном случае построенная нейронная сеть просто запомнит данные, потеряв возможность делать статистически значимые предсказания на но-

вых данных (напомним, что это нежелательное явление называется переобучением).

Во-вторых, в большинстве задач аппроксимации нет смысла использовать архитектуру нейронной сети с количеством скрытых слоев более одного. Нелинейный же характер нейронной сети определяется количеством нейронов в этом скрытом слое.

В-третьих, использование порядка нелинейности нейрона более 1 рекомендуется только подготовленным пользователям.

Подтвердите выбранную конфигурацию нейронной сети нажатием кнопки **OK** и вернитесь в исходное окно, где конфигурация сети будет проиллюстрирована в графическом виде (рис. 5.79).

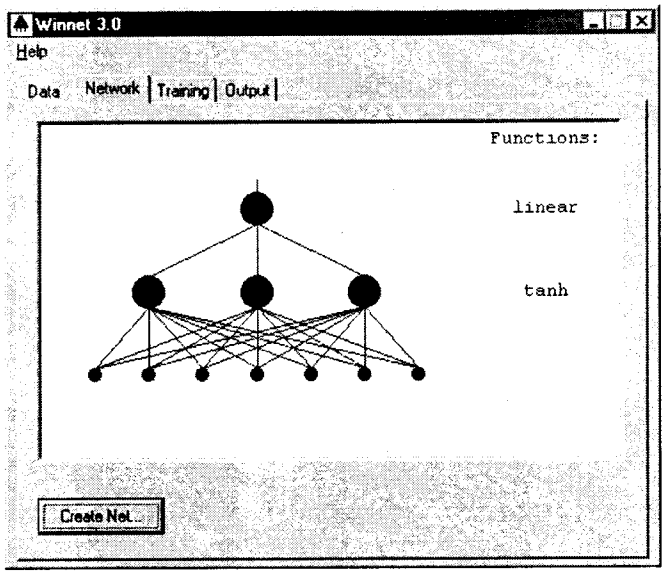


Рис. 5.79. Графическое представление структуры созданной нейронной сети

11) Для обучения созданной нейронной сети перейдите на следующую закладку **Training**. Перед обучением надо задать тестовое множество примеров из общей совокупности обучающих примеров. Эти примеры не будут участвовать в обучении. На них будут основываться оценки предсказательных свойств обученной нейронной сети. Щелкните по кнопке **Edit test set...** . В окне (рис. 5.80) можно задать размер и характер обучающей выборки.

Общие рекомендации:

- число примеров в тестовой выборке должно составлять 20–30%, но не менее нескольких десятков;

- при решении задач аппроксимации наиболее естественным является случайный выбор тестового множества. Отдельная опция **Random + Last examples** введена для задач прогнозирования временных рядов. При этом последние примеры всегда исключаются из обучения и являются, по сути, прогнозом.

Подтвердите выбор нажатием кнопки **OK** и вернитесь в основное окно.

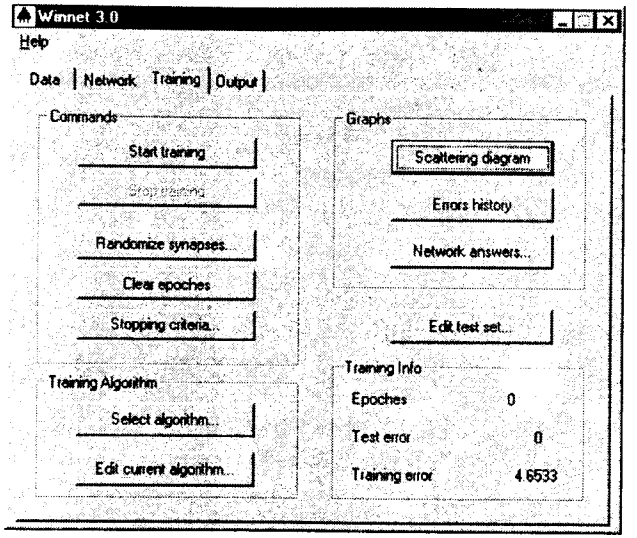


Рис. 5.80. Окно задания параметров обучения

12) Теперь можно начинать обучение. Нажав кнопку **Start training**, можно наблюдать за ходом обучения нейронной сети по изменению информации в области **Training Info** или в графическом виде в соответствующих окнах, вызываемых нажатием клавиш в области **Graphs**. Желательно остановить процесс обучения в момент, когда ошибки обучения и обобщения начнут сильно расходиться. Рост ошибки обобщения сигнализирует о начале переобучения.

13) По завершении процесса обучения его результаты можно визуально оценить на графике **Network answers...**, вызываемом по нажатию соответствующей кнопки.

14) Пакет позволяет управлять параметрами процесса обучения. Но данными возможностями рекомендуется пользоваться только подготовленным пользователям в исключительных случаях.

15) Теперь осталось сохранить проект (**Save Project...**) и экспортировать результаты назад в книгу Excel. Для экспорта результатов перейдите на закладку **Output**, задайте необходимые параметры и сохраните результаты нажатием кнопки **OK**.

16) Можете теперь закрыть окно программы Winnet 3.0. Дальнейший анализ полученных результатов удобнее проводить стандартными статистическими методами в Excel.

Если ранее созданная и обученная нейронная сеть применяется в работе с новыми данными для получения прогнозов, то эти данные, естественно, должны иметь тот же формат, как и те, по которым сеть обучалась. В этом случае необходимо проделать следующие операции:

- загрузить данные в систему;
- загрузить ранее созданный проект;
- сохранить предсказанные данные в книге Excel.

Последовательность действий такова.

1) Выделите необходимую для работы область данных на листе книги Excel. Прежние соглашения относительно расположения данных остаются в силе.

Необходимо отметить, что количество и порядок столбцов новых данных должно в совпадать с данными, использованными при создании проекта и обучении нейронной сети. С этой целью столбцы, используемые в качестве выходов, заполняются нулевыми значениями.

2) Загрузите созданный ранее проект, нажав на кнопку **Load Project...** . В открывшемся окне выберите нужный файл проекта, имеющий по умолчанию расширение *.wnp, и подтвердите выбор нажатием кнопки **Открыть**.

3) Перейдите на закладку **Output**, задайте необходимые параметры и сохраните результаты нажатием кнопки **OK**. Работу с программой можно считать завершенной.

Kohonen Map

Kohonen Map представляет собой удобный инструмент анализа многомерной информации, которая отображается в виде двумерной цветной карты, раскрашиваемой по любому интересующему параметру. Карта строится по специальному алгоритму, сохраняющему локальную близость данных: точки, близкие на карте будут близки и в исходном многомерном пространстве. Обратное, вообще говоря, неверно.

Если работа с данными выполняется впервые, то для проведения анализа необходимо сделать следующее:

- загрузить данные из книги Excel в систему;
- определить входы, которые необходимы для проведения кластеризации;
- предобработать данные – осуществить их нормировку;
- при желании выбрать нужное число главных компонентов;
- создать и обучить нейронную сеть Кохонена;
- провести анализ многомерных данных встроенными средствами Kohonen Map;
- сохранить обработанные данные и рисунки в книге Excel;
- если необходимо, сохранить обученную нейронную сеть для дальнейшей работы.

Разберем все эти этапы по шагам.

1) Для работы выделите область данных на листе книги Excel.

Данные на листе располагаются следующим образом: входы и выходы – столбцы, а строки – обучающие примеры. Выделяемая область может включать в первой строке названия входов как на русском, так и на английском языке. Кроме того, в первой колонке могут находиться данные, используемые не для обучения сети Кохонена, а для идентификации примеров при последующем анализе результатов.

2) Щелкните мышью по кнопке с цветной картой Кохонена на панели инструментов **Neural Analysis**. В ответ появится диалоговое окно **Select data source**, предлагающее уточнить параметры области данных для работы (рис. 5.75). В случае согласия с параметрами ввода нажмите кнопку **OK**.

3) Откроется основное окно программы Kohonen Map 1.0, которое содержит два листа **Project** и **Results** (рис. 5.81).

4) Открывшийся лист **Project** позволяет определить и предобработать данные для последующего использования. Кроме того, с этого листа можно сохранить обученную нейронную сеть (**Save Project ...**) или загрузить уже сохраненный в прошлом проект (**Load Project...**). Прежде всего нужно определить входы, для чего нажмите на кнопку **Create patterns ...** .

5) В открывшемся диалоговом окне **Select relevant columns** выберете в окне левого списка **All columns** необходимые входы и с помощью кнопок **>** или **>>** перейдите в окно списка **Selected**. Корректировку выбранных входов можно провести, используя кнопки **<** или **<<**. Отметим, что в отличие от Winnet 3.0 теперь все данные являются входными и участвуют в обучении.

6) Поскольку конкретные значения входов могут быть из любого диапазона, то рекомендуется провести их нормировку. Для

большинства случаев подходит нормировка входных значений **Mean/Variance**. Нажмите кнопку **Normalize...** и выберите в открывшемся окне **Inputs normalization** соответствующую позицию переключателя. Подтвердите выбор нажатием кнопки **OK** и вернитесь в окно **Select relevant columns**.

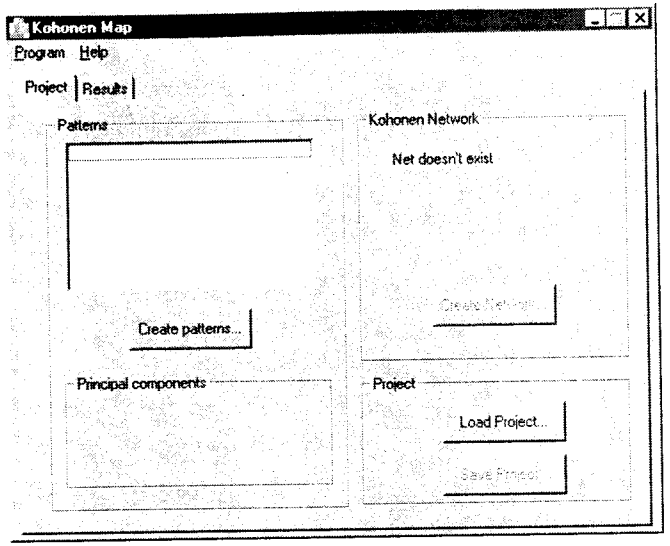


Рис. 5.81. Основное окно программы Kohonen Map 1.0

7) Очень часто при анализе используется много входов, имеющих существенную линейную зависимость друг от друга. В этих случаях реализованный в пакете Kohonen Map 1.0 *метод главных компонентов* (PCA) позволяет автоматически существенно понизить размерность пространства входных векторов. Для этого переключатель **Extract principal components** должен быть включен. Подтвердите выбор нажатием кнопки **OK**.

8) В открывшемся графическом окне **PC Analyzer** (рис. 5.82) можно контролируемо понизить размерность входов за счет уменьшения числа учитываемых при обучении главных компонентов без существенной потери информативности. С помощью соответствующего графика можно оценить потери информации. Обычно они не должны превышать 10%. Подтвердите выбор нажатием кнопки **OK** и вернитесь в основное окно программы.

9) Следующий этап – создание карты Кохонена. Нажатием кнопки **Create Network...** перейдите в диалоговое окно **Dialog** и

задайте параметры сети (число кластеров): число ячеек по горизонтали и вертикали (рис. 5.83).

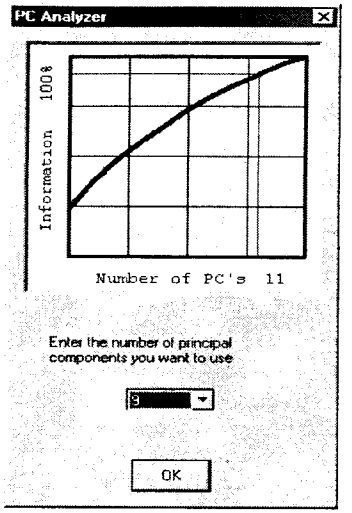


Рис. 5.82. Диалоговое окно для задания числа главных компонентов

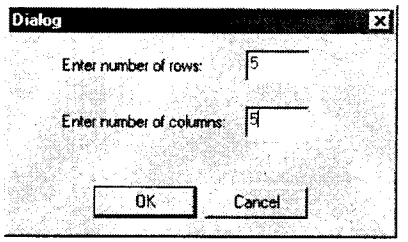


Рис. 5.83. Окно задания числа кластеров

10) При создании нейронной сети следует руководствоваться следующими простыми правилами:

- для хорошей аппроксимации сетью данных общее число нейронов сети должно быть, как минимум, на порядок меньше числа обучающих примеров;
- кластерную структуру данных, напротив, лучше исследовать, когда число ячеек сравнимо с числом примеров. При этом появятся пустые ячейки, разграничивающие области данных.

11) Подтвердите выбранную конфигурацию сети нажатием кнопки **OK**. Далее автоматически стартует процесс обучения.

12) По завершении процесса обучения для анализа результатов перейдите на закладку **Results** (рис. 5.84).

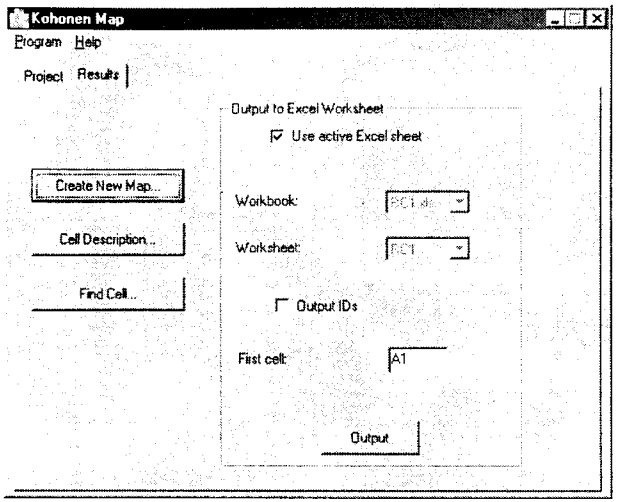


Рис. 5.84. Диалоговое окно формы вывода результатов

Здесь доступны следующие функции:

- **Create New Map...** – цветовая раскраска карты Кохонена по любому параметру с выбранной степенью градации;
- **Cell Description...** – определение усредненных значений входных параметров для данного кластера (ячейки) и принадлежащих ему примеров;
- **Find Cell...** – поиск кластера, которому принадлежит данный пример;
- **Output** – сохранение результатов в книге Excel.

13) Кроме того, программа позволяет управлять параметрами процесса обучения и изменения цветовой палитры раскраски карты Кохонена. Для этого выберите пункт меню **Program**, пункт **Preferences** и далее **Set Custom...** . В открывшемся окне **Program Preferences** на закладке **Colors** можно установить другие цвета градационной раскраски карты, а на странице **Training parameters** – параметры обучения сети (данными возможностями рекомендуется пользоваться подготовленным пользователям в исключительных случаях). Восстановить параметры программы по умолчанию можно, выбрав там же подпункт меню **Set Defaults**.

14) Создаваемая при нажатии кнопки **Create New Map...** карта (рис. 5.85) является активной: при двойном щелчке мыши на какой-либо ячейке открывается окно **Cell description**, в котором удобно проводить анализ усредненных значений параметров (рис. 5.86). Дополнительно предусмотрена возможность сохранения изображения карты раскраски в формате *.bmp для последующего экспорта через буфер обмена в любые документы MS Office. Для этого в окне карты выберите **Actions/Copy to Bitmap**. Далее вернитесь в документ MS Office и произведите вставку рисунка командой **Paste** меню **Edit**.

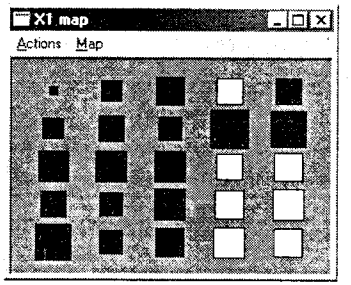


Рис. 5.85. Условный графический вид выявленных кластеров

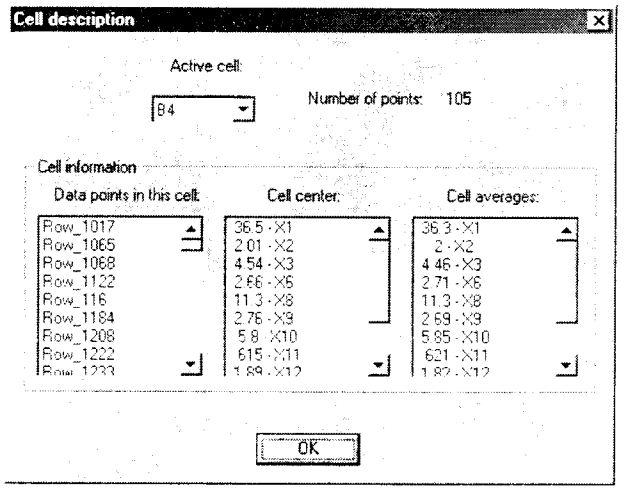


Рис. 5.86. Характеристики центра выбранного кластера

15) Осталось сохранить результаты работы. В программе предусмотрены функции сохранения проекта (кнопка **Save Project...**) и экспорта результатов назад в книгу Excel. Для экспорта

результатов перейдите на закладку **Project**, задайте необходимые параметры и сохраните результаты нажатием кнопки **OK**.

16) Можете закрыть окно программы Kohonen Map 1.0. Дальнейший анализ полученных результатов удобнее проводить стандартными статистическими методами в Excel.

Для использования при анализе новых данных ранее сохраненного проекта необходимо проделать следующие операции:

- загрузить данные в систему;
- загрузить созданный ранее проект;
- провести анализ результатов встроенными средствами Kohonen Map 1.0;
- сохранить данные в книге Excel.

Последовательность действий такова.

1) Выделите необходимую для работы область данных на листе книги Excel. Прежние соглашения относительно расположения данных остаются в силе.

Количество и расположение столбцов новых данных и описание первого столбца должно соответствовать параметрам, использованным при создании проекта и обучении сети Кохонена.

2) Загрузите созданный ранее проект, нажав на кнопку **Load Project...** . В открывшемся окне выберите нужный файл проекта, имеющий по умолчанию расширение *.kmp, и подтвердите выбор нажатием кнопки **Открыть**.

3) Перейдите на закладку **Results** и проведите анализ результатов встроенными средствами Kohonen Map 1.0. Затем, задайте необходимые параметры и сохраните результаты нажатием кнопки **Output**. Работу с программой можно считать завершенной.

5.8.4. Впечатления от работы с пакетом

Пакет не обладает какими-то особенными функциональными или сервисными возможностями, но, тем не менее, представляется очень удобной.

5.9. Пакет Fuzzy Logic Toolbox

5.9.1. Общая характеристика

Fuzzy Logic Toolbox используется в системе MATLAB и представляет собой пакет прикладных программ, относящихся к теории нечетких множеств и позволяющих конструировать так называемые нечеткие экспертные и/или управляющие системы.

Основные возможности пакета:

- построение систем нечеткого вывода (экспертных систем, нечетких регуляторов, аппроксиматоров зависимостей);
- построение адаптивных нечетких систем (нечетких нейронных сетей);
- интерактивное динамическое моделирование в Simulink.

Пакет позволяет работать в режиме графического интерфейса, в режиме командной строки, а также с использованием блоков и примеров пакета Simulink.

5.9.2. Состав графического интерфейса

В состав средств Fuzzy Logic Toolbox входят следующие основные программы, позволяющие работать в режиме графического интерфейса:

- редактор Fuzzy Inference System Editor (FIS Editor или FIS-редактор) вместе со вспомогательными программами: редактором функций принадлежности (Membership Function Editor), редактором правил (Rule Editor), просмотрщиком правил (Rule Viewer) и просмотрщиком поверхности отклика (Surface Viewer);
- редактор нечетких нейронных систем (ANFIS Editor или ANFIS-редактор);
- программа нахождения центров кластеров (Clustering – кластеризация).

Набор этих программ предоставляет пользователю максимальные удобства для создания, редактирования и использования различных систем нечеткого вывода. Далее остановимся только на вопросах построения, обучения и использования нечетких нейронных сетей с помощью пакета Fuzzy Logic Toolbox.

5.9.3. Создание нечеткой нейронной сети

Графический интерфейс нечетких нейронных систем вызывается функцией **anfisedit** из режима командной строки. Исполнение функции приводит к появлению окна редактора нечетких нейронных систем (ANFIS Editor), вид которого приведен на рис. 5.87.

С помощью данного редактора осуществляется создание или загрузка структуры нечеткой нейронной сети, просмотр структуры, настройка ее параметров, проверка качества функционирования такой сети.

Рассмотрим пример создания нечеткой нейронной сети, отображающей зависимость $y = x^2$ между переменными x и y, заданную с помощью табл. 5.4.

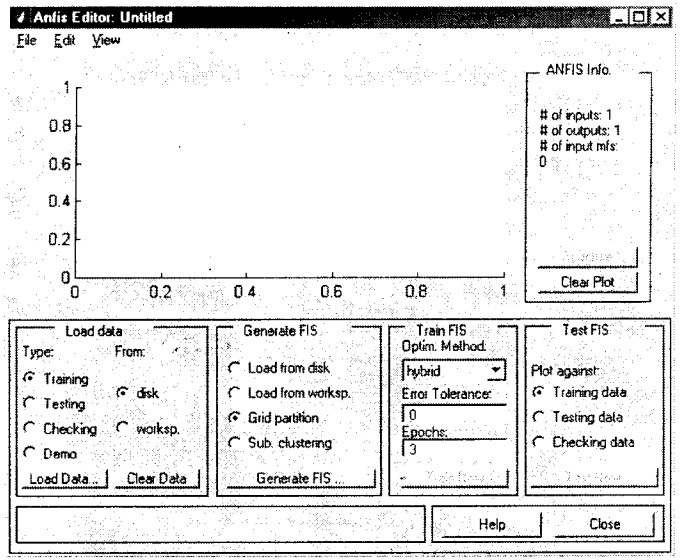


Рис 5.87. Окно редактора нечеткой нейронной сети

Таблица 5.4

Значения x и y

x	−1	−0,6	0	0,4	1
y	1	0,36	0	0,16	1

Создание структуры, настройка параметров и проверка осуществляются по обучающей (Training), проверочной (Checking) и тестирующей (Testing) выборкам, которые предварительно должны быть представлены в виде текстовых фалов (с расширением *.dat и разделителями–табуляциями), первые колонки которых соответствуют входным переменным, а последняя – единственной выходной переменной; количество строк в таких файлах равно количеству образцов (примеров). Так, обучающая выборка, сформированная по табл. 5.4, представляется в следующем виде:

```
-1      1
-0.6    0.36
0.0     0.00
0.4     0.16
1       1
```

Строгих рекомендаций по объемам этих выборок не существует, поэтому следует исходить из принципа «чем больше, тем лучше». Обучающая и проверочная выборки непосредственно задействуются в процессе настройки параметров нечеткой нейронной сети. Проверочная выборка используется для определения переобучения сети, при котором ошибка для обучающей последовательности стремится к нулю, а для проверочной – возрастает. Тестовая выборка применяется для проверки качества функционирования обученной сети.

Пункты меню **File** и **View**, идентичны пунктам FIS-редактора за тем исключением, что здесь работа может происходить только с алгоритмом нечеткого вывода Sugeno. Пункт меню **Edit** содержит единственный подпункт **Undo** (Отменить выполненное действие).

Набор опций **Load data** (Загрузить данные) в нижней левой части окна редактора включает:

- **Type** – тип загружаемых данных (Training – для обучения, Testing – для тестирования, Checking – для проверки, Demo – демонстрационные);
- **Disk** (Диск) или **Workspace** (Рабочая область) – место, откуда должны загружаться данные.

К этим опциям относятся два действия: **Load Data...** (Загрузить данные) и **Clear Data** (Стереть введенные данные).

Следующая группа опций ANFIS-редактора объединена под именем **Generate FIS** (Создание нечеткой системы вывода). Данная группа включает в себя опции:

- **Load from disk** – загрузка структуры системы с диска;
- **Load from worksp.** – загрузка структуры системы из рабочей области MATLAB;
- **Grid partition** – разбиение областей определения входных переменных (аргументов) на подобласти независимо для каждого аргумента;
- **Subtract clustering** или **Sub. clustering** – разбиение всей области определения входных переменных (аргументов) на подобласти в комплексе для всех аргументов;
- **Generate FIS** – создание нечеткой системы с точностью до ряда параметров.

Следующая группа опций **Train FIS** (Обучение нечеткой системы вывода) позволяет определить метод обучения (Optim. Method) нечеткой нейронной сети: гибридный (Hybrid) или обратного распространения ошибки (Backpropagation); установить уровень текущей суммарной, по всем образцам, ошибки обучения (Error Tolerance), при достижении которого процесс обучения заканчивается; количество циклов обучения (Epochs). Процесс обучения заканчивается либо при достижении уровня ошибки, либо при проведении заданного количество циклов.

Опция **Train Now** позволяет начать процесс обучения нечеткой нейронной сети.

В окне ANFIS-редактора выдаются данные (ANFIS Info.) о системе: количество входов, выходов, функций принадлежности входов; нажатие кнопки **Structure** (Структура) позволяет увидеть структуру сети в виде, аналогичном представленному на рис. 3.18. Кнопка **Clear** (Очистить) позволяет стереть все результаты.

Опции **Test FIS** позволяют провести проверку и тестирование созданной и обученной системы с выводом результатов в виде графиков (для обучающей выборки – Training data, тестирующей выборки – Testing data и проверочной выборки – Checking data). Кнопка **Test Now** позволяет запустить указанные процессы.

Работу с редактором рассмотрим на примере восстановления зависимости $y = x^2$ по данным табл. 5.4. Предположим, что эти данные сохранены в файле Proba.dat. Создание и проверку нечеткой нейронной сети проведем по этапам.

1) В окне ANFIS-редактора выберем тип загружаемых данных Training и нажмем кнопку **Load data**. В окне диалога укажем местоположение и имя файла. Его открытие приводит к появлению в графической части окна редактора набора точек, соответствующих введенным данным (рис. 5.88).

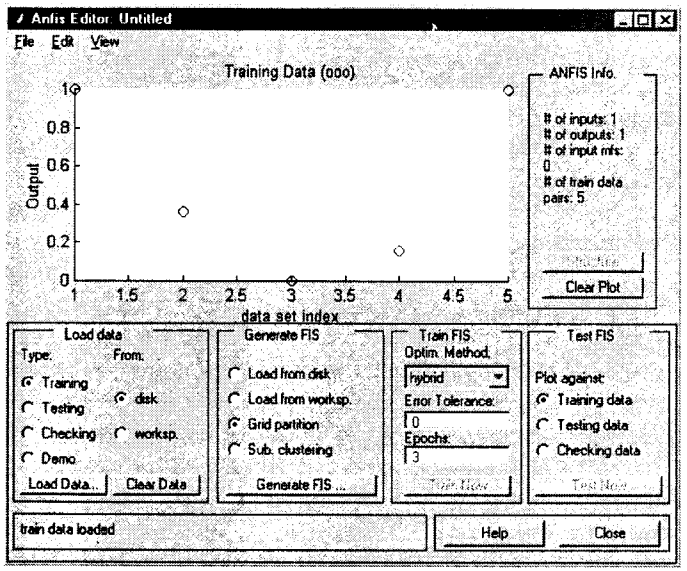


Рис. 5.88. Окно ANFIS-редактора после загрузки обучающей выборки

2) В группе опций **Generate FIS** по умолчанию активизирована опция **Grid partition**. Не будем ее изменять и нажмем кнопку **Generate FIS**, после чего появится диалоговое окно (рис. 5.89) для задания числа и типов функций принадлежности. Сохраним все установки по умолчанию, нажав **OK**. Произойдет возврат в основное окно ANFIS-редактора. Теперь структура нечеткой нейронной сети создана, и ее графический вид можно просмотреть с помощью кнопки **Structure** (рис. 5.90).

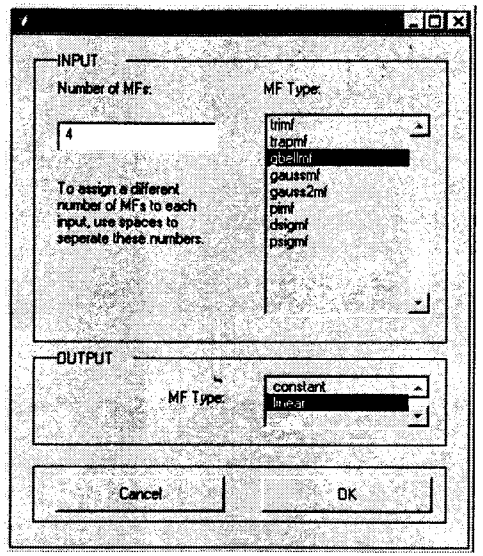


Рис. 5.89. Окно задания функций принадлежности

3) Перейдем к опциям **Train FIS**. Не будем менять задаваемые по умолчанию метод настройки параметров (Hybrid) и уровень ошибки (0), но количество циклов обучения изменим на 40, после чего нажмем кнопку начала процесса обучения (**Train Now**). Получившийся результат в виде графика ошибки сети в зависимости от числа циклов обучения, из которого следует, что обучение фактически закончилось после пятого цикла, представлен на рис. 5.91.

4) Теперь нажатием кнопки **Test Now** можно начать процесс тестирования обученной сети, но, поскольку использовалась только обучающая выборка, ничего особенно интересного ожидать не приходится. Действительно, выходы обученной сети практически совпадают со значениями обучающей выборки.

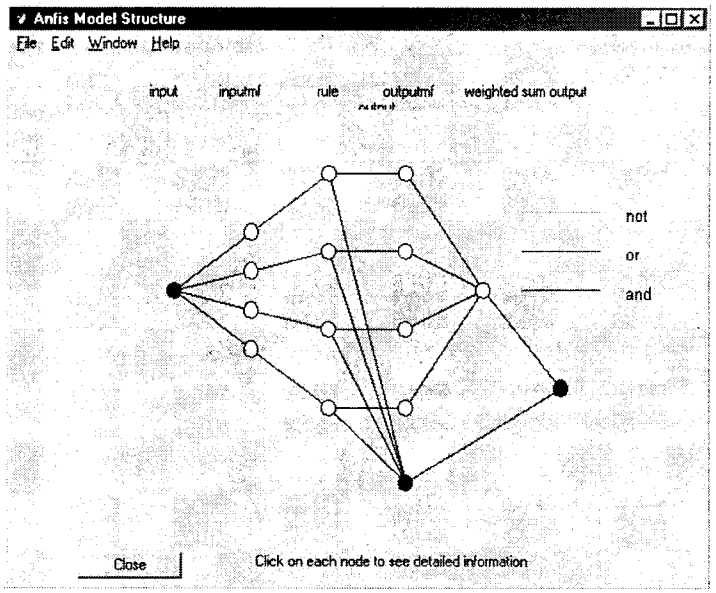


Рис. 5.90. Структура созданной гибридной сети

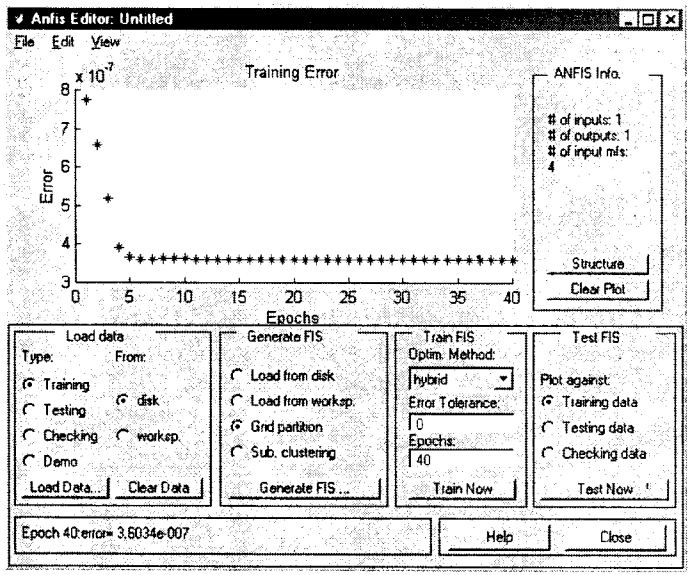


Рис. 5.91. Результат обучения сети

5) Сохраним разработанную нечеткую нейронную сеть с именем Proba1.fis и, для исследования разработанной системы средствами FIS-редактора, из командной строки MATLAB выполним команду **Fuzzy**, а затем через пункты меню **File/Open FIS from disk...** откроем созданный файл. С созданной сетью можно теперь выполнять все приемы редактирования и исследования.

5.9.4. Впечатления от работы с пакетом

Что можно сказать про эффективность использования нечетких нейронных сетей и ANFIS-редактора? В данном случае используется только один алгоритм нечеткого вывода – Sugeno (нулевого или первого порядков), может быть задана только одна выходная переменная, всем правилам приписывается один и тот же единичный вес. Кроме того, возникают значительные проблемы при количестве входных переменных большим, чем 5–6.

Несомненными достоинствами пакета является практически полная автоматизация процесса создания нечеткой нейронной сети, возможность просмотра сформированных правил и придания им лингвистической интерпретации, что позволяет рассматривать аппарат нечетких нейронных сетей как средство извлечения знаний из баз данных и существенно отличает данные сети от классических нейронных.

Рекомендуемые области применения: построение аппроксиматоров зависимостей по экспериментальным данным, систем классификации, извлечение знаний.

5.10. Совсем все просто

Хотелось бы, чтобы после прочтения данной главы у читателя сложилось бы совершенно правильное впечатление об относительной легкости работы с программами-нейроимитаторами. Если такую задачу авторам удалось решить, значит, их труд не пропал даром.

Часть III

ПРИМЕНЕНИЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Глава 6

ПРИМЕРЫ ПРИМЕНЕНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Рассмотрим несколько конкретных примеров применения аппарата искусственных нейронных сетей для решения популярных задач классификации, прогнозирования, аппроксимации, сжатия информации, построения экспертных систем и некоторых других.

6.1. Прогнозирование результатов выборов

Данная задача стала классической для демонстрации работы нейросетевого классификатора. Она компактна, значения всех обучающих параметров представляются в форме «Да–Нет», основана на реальных данных и дает хороший результат. Содержательная постановка задачи взята из книги: Горбань А. Н., Россиев Д. А. Нейронные сети на персональном компьютере. – Новосибирск: Наука, 1996 (см. список литературы).

6.1.1. Содержательная постановка задачи

Рассмотрим использование нейроимитатора на примере предсказания итогов выборов президента США.

На первый взгляд кажется, что итоги выборов зависит только от личностей кандидатов и от их программ. Однако и программы, и образы кандидатов создаются профессионалами. Оказывается, что если предвыборные компании всех кандидатов отработаны добросовестно и все участники сделали все возможное, то выбор практически предопределяется лишь объективными признаками сложившейся накануне выборов ситуации в стране. А кто победит, можно решать на основании ответов на следующие вопросы.

1) Правящая партия у власти более 1 срока?

2) Правящая партия получила больше 50 % на прошлых выборах?

3) В год выборов была активна третья партия?

4) Была серьезная конкуренция при выдвижении кандидата от правящей партии?

5) Кандидат от правящей партии был президентом в год выборов?

6) Был ли год выборов временем спада или депрессии?

7) Был ли рост среднего национального валового продукта на душу населения более 2,1%?

8) Произвел ли правящий президент существенные изменения в политике?

9) Во время правления были существенные социальные волнения?

10) Администрация правящей партии виновна в серьезной ошибке или скандале?

11) Кандидат правящей партии – национальный герой?

12) Кандидат оппозиционной партии – национальный герой?

Обучающая выборка состоит из 31 примера, каждый из которых представляет ситуацию выборов, начиная с 1864 г. (табл. 6.1), где ответы «Да» обозначены единицами, а ответы «Нет» – нулями.

Класс 1 означает, что в данной ситуации был избран кандидат правящей партии, класс 2 – кандидат оппозиционной партии. После обучения сеть должна предсказать ответ для ситуации, отраженной табл. 6.2, которая не входила в обучающую выборку (когда производились эксперименты, результат выборов 1992 г. еще не был известен).

Таблица 6.1

Обучающая выборка для прогнозирования результата выборов президента США

№	Год	Класс	Обучающие параметры											
			1	2	3	4	5	6	7	8	9	10	11	12
1	1864	1	0	0	0	0	1	0	0	1	1	0	0	0
2	1868	1	1	1	0	0	0	0	1	1	1	0	1	0
3	1872	1	1	1	0	0	1	0	1	0	0	0	1	0
4	1880	1	1	0	0	1	0	0	1	1	0	0	0	0
5	1888	1	0	0	0	0	1	0	0	0	0	0	0	0
6	1900	1	0	1	0	0	1	0	1	0	0	0	0	1
7	1904	1	1	1	0	0	1	0	0	0	0	0	1	0
8	1901	1	1	1	0	0	0	0	0	1	0	0	0	1
9	1916	1	0	0	0	0	1	0	0	1	0	0	0	0
10	1924	1	0	1	1	0	1	0	1	1	0	1	0	0
11	1921	1	1	1	0	0	0	0	1	0	0	0	0	0
12	1936	1	0	1	0	0	1	1	1	1	0	0	1	0
13	1940	1	1	1	0	0	1	1	1	1	0	0	1	0
14	1944	1	1	1	0	0	1	0	1	1	0	0	1	0
15	1948	1	1	1	1	0	1	0	0	1	0	0	0	0
16	1956	1	0	1	0	0	0	0	0	0	0	0	1	0
17	1964	1	0	0	0	0	1	0	1	0	0	0	0	0
18	1972	1	0	0	0	0	1	0	1	0	1	0	0	0
19	1860	2	1	0	1	1	0	0	1	0	1	0	0	0
20	1872	2	1	1	0	1	0	1	0	0	0	1	0	0
21	1884	2	1	0	0	1	0	0	1	0	1	0	1	0
22	1892	2	0	0	1	0	1	0	0	1	1	0	0	1
23	1896	2	0	0	0	1	0	1	0	1	1	0	1	0
24	1912	2	1	1	1	1	1	0	1	0	0	0	0	0
25	1920	2	1	0	0	1	0	0	0	1	1	0	0	0
26	1932	2	1	1	0	0	1	1	0	0	1	0	0	1
27	1952	2	1	0	0	1	0	0	1	0	0	1	0	1
28	1960	2	1	1	0	0	0	1	0	0	0	0	0	1
29	1968	2	1	1	1	1	0	0	1	1	1	0	0	0
30	1976	2	1	1	0	1	1	0	0	0	0	1	0	0
31	1980	2	0	0	1	1	1	1	0	0	0	1	0	1

Таблица 6.2

Пример выборной ситуации в США в 1992 г. (Д. Буш – Б. Клинтон)

32	1992	?	Нет	Да	Да	Да	Да	Да	Нет	Да	Да	Да	Нет	Нет

Ответ неизвестен.

6.1.2. Нейросетевое моделирование

Для решения поставленной задачи выберем в качестве программы-нейроимитатора нейропакет НейроПро (см. разд. 5.2). Решение будем проводить по этапам.

1) Подготовка исходных данных.

Используя табл. 6.1, подготовим в Excel обучающую выборку в виде табл. 6.3; сохраним эти данные в виде файла dBASE, как описано в разд. 5.2, с названием Выборы.dbf.

Таблица 6.3

Исходные данные (обучающая выборка) в виде таблицы

Y	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
1	0	0	0	0	1	0	0	1	1	0	0	0
1	1	1	0	0	0	0	1	1	1	0	1	0
1	1	1	0	0	1	0	1	0	0	0	1	0
1	1	0	0	1	0	0	1	1	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	1	0	0	0	0	1
1	1	1	0	0	1	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	1	0	0	0	0
1	0	1	1	0	1	0	1	1	0	1	0	0
1	1	1	0	0	0	0	1	0	0	0	0	0
1	0	1	0	0	1	1	1	1	0	0	1	0
1	1	1	0	0	1	1	1	1	0	0	1	0
1	1	1	0	0	1	0	1	1	0	0	1	0
1	1	1	1	0	1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	1	0	1	0	0	0	0	0
1	0	0	0	0	1	0	1	0	1	0	0	0
2	1	0	1	1	0	0	1	0	1	0	0	0
2	1	1	0	1	0	1	0	0	0	1	0	0
2	1	0	0	1	0	0	1	0	1	0	1	0
2	0	0	1	0	1	0	0	1	1	0	0	1
2	0	0	0	1	0	1	0	1	1	0	1	0
2	1	1	1	1	1	0	1	0	0	0	0	0
2	1	0	0	1	0	0	0	1	1	0	0	0
2	1	1	0	0	1	1	0	0	1	0	0	1
2	1	0	0	1	0	0	1	0	0	1	0	1
2	1	1	0	0	0	1	0	0	0	0	0	1
2	1	1	1	1	0	0	1	1	1	0	0	0
2	1	1	0	1	1	0	0	0	0	1	0	0
2	0	0	1	1	1	1	0	0	0	1	0	1

Подготовим данные для опроса в соответствии с табл. 6.2 в форме табл. 6.4 и сохраним под именем Прогноз.dbf.

Таблица 6.4

Исходные данные для прогноза

y	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
	0	1	1	1	1	1	0	1	1	1	0	0

2) Задание топологии нейронной сети.

Будем использовать нейронную сеть с одним скрытым слоем. Очевидно, число входных нейронов – 12, число выходных нейронов – 1. Для определения числа нейронов в скрытом слое воспользуемся рекомендациями, приведенными в разд. 1.3.

Использование формулы (1.5) при $N_x = 12$, $N_y = 1$, $N_p = 31$ дает минимальное значение для N_w – 5, и максимальное – 51; из формулы (1.6) следует, что число нейронов в скрытом слое должно быть от 1 до 4. Из выражения (1.8) видно, что это число не должно превышать 2,5. На основании этих результатов примем число нейронов в скрытом слое равным 2.

3) Обучение нейронной сети.

Используя нейропакет НейроПро и подготовленный файл Выборы.dbf, действуя в соответствии с правилами, изложенными в разд. 5.2, создадим нейронную сеть заданной топологии, проведем ее обучение и определим наиболее значащие признаки. Полученный результат отражен на рис. 6.1 и несколько отличается от приведенного в цитированном источнике. В нашем исследовании получилось, что наибольшее влияние на исход выборов оказывают ответы на вопросы 4, 8, 3 и 9.

Сохраним сеть и проект под именем «Выборы»; закроем программу.

4) Опрос обученной сети.

Вновь запустим программу НейроПро, откроем сохраненный проект и файл Прогноз.dbf. Выберем режим тестирования сети. Полученный при этом результат (в нашем случае – 2,01401, т. е. 2 при округлении) говорит о том, что на 1992 год прогнозируется победа кандидата от оппозиционной партии, т. е. Б. Клинтона. Как известно, так и произошло. Отметим, что весьма близкий выход сети к числу 2 говорит, пожалуй, что сделанному прогнозу следовало доверять с большой степенью уверенности.

К сожалению, крайне неясно, можно ли использовать аналогичный нейросетевой подход для прогноза выборов в условиях пока еще очень нестабильной России … .

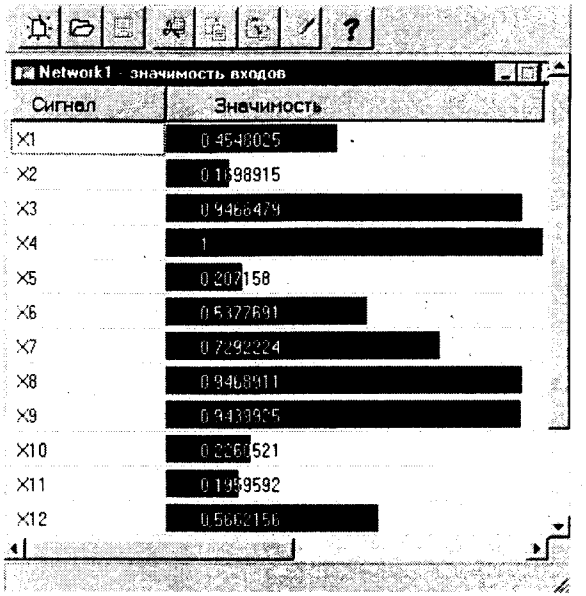


Рис. 6.1. Информативность параметров при выборе президента

6.2. Анализ данных социологического опроса

Постановка задачи. Исходные данные представляют собой материалы социологического опроса, проведенного в 1999 г. анкетированием 1500 респондентов одного из регионов России по 14 вопросам (признакам), отражающим социальный статус опрашиваемого:

x_1 – возраст, лет;
x_2 – пол;
x_3 – образование;
x_4 – базовая профессия;
x_5 – национальность;
x_6 – самооценка социального слоя;
x_7 – отношение к религии;
x_8 – род занятий;
x_9 – основное место работы;
x_{10} – сфера деятельности;
x_{11} – средний доход члена семьи, руб.;
x_{12} – самооценка уровня доходов;
x_{13} – тип населенного пункта, где проживает опрашиваемый;
x_{14} – политическая ориентация.

Среди перечисленных только два признака (x_1 и x_{11}) имеют количественный характер, три признака (x_4, x_5 и x_7) – чисто качественный, остальные – качественный, выраженный в псевдоколичественной форме.

Заметим, что признак x_{14} (политическая ориентация) представляется здесь «выходной» или «основной» следственной переменной, определяемой или формируемой другими (причинными) переменными $x_1, \ldots, x_{13}$. Действительно, можно предположить, что уровень доходов в семье формирует политическую ориентацию, обратное вряд ли имеет место, хотя и может иметь место; аналогичные соображения можно провести и по другим признакам.

В связи с этим, сформулируем следующую задачу исследований: *выявить причинно-следственные связи между политической ориентацией субъекта* (x_{14}) *и признаками, характеризующими его социальное положение* ($x_1, \ldots, x_{13}$).

Методы исследований. При выборе метода исследований необходимо учесть тот факт, что большинство из признаков – качественные, и поэтому имеющиеся данные требуют применения специальных приемов исследования. Между тем, современные инструментальные средства обработки статистической информации, например, пакет Statistica, возможностями такого рода практически не обладают (кроме использования дисперсионного анализа). При анализе же псевдоколичественных данных необходимо принимать во внимание невозможность в большинстве случаев установления между ними отношений эквивалентности и предпочтения, что влечет трудности в определении мер сходства и т. п. Это, в свою очередь, при применении к ним приемов и формул обработки, разработанных для количественных переменных (например, регрессионного, корреляционного или дискриминантного анализов) приводит к крайне низкой достоверности получаемых результатов.

Поэтому выберем в качестве методов исследования менее чувствительные к выполнению вероятностных предпосылок нейросетевые методы, а в качестве инструментов исследования возьмем нейропакеты НейроПро и Excel Neural Package. Такой подбор инструментальных средств обеспечит перекрестную проверку получаемых результатов.

Этапы исследований.

1) Определение признаков, наиболее существенно влияющих на выбранную выходную переменную (отклик).

2) Построение модели, отражающей причинно-следственные связи между откликом и входными признаками.

3) Интерпретация модели.

Полученные результаты.

1) Результаты первого этапа, характеризующие степень влияния факторов на отклик x_{14} (качественные признаки x_4, x_5, x_7 исключены из рассмотрения), представлены на рис. 6.2 (пакет НейроПро) и рис. 6.3 (пакет Excel Neural Package).

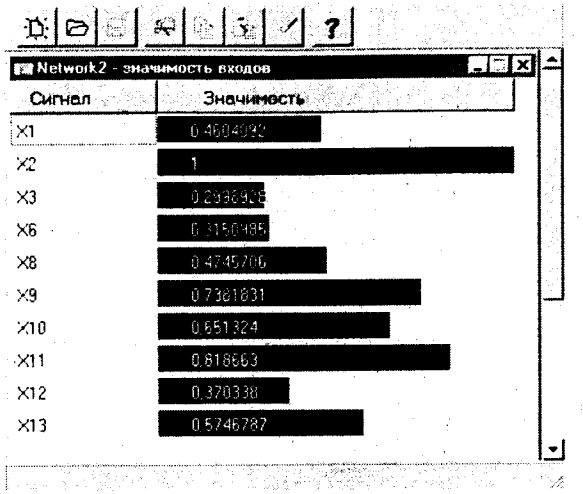


Рис. 6.2. Оценка значимости факторов (пакет НейроПро)

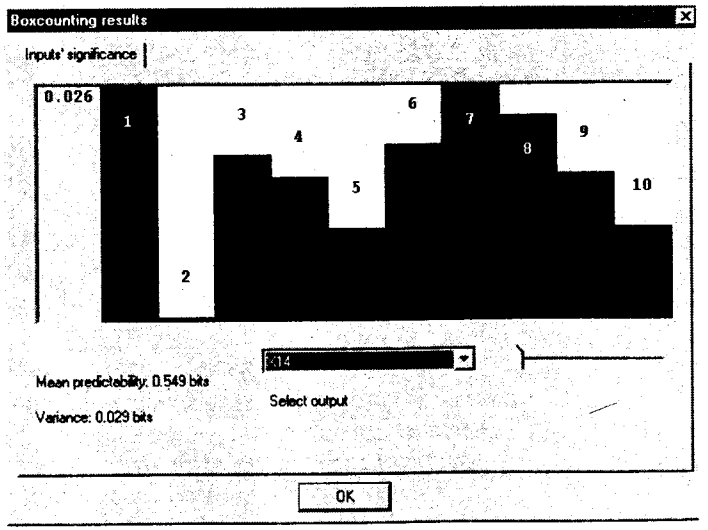


Рис. 6.3. Оценка значимости факторов (пакет Excel Neural Package)

Из рис. 6.2 следует, что наиболее значимыми факторами являются x_2, x_{11}, x_9, x_{10}, x_{13}, а из рис. 6.3 – x_1, x_{10} (седьмой по счету), x_{11}, x_3, x_6.

Обобщая результаты, получим следующую упорядоченную по степени влияния на x_{14} последовательность признаков: x_1 (возраст), x_{11} (доход), x_{10} (сфера деятельности), x_6 (самооценка социального слоя), x_{13} (тип населенного пункта проживания).

2) Результаты второго этапа исследований, характеризующие проверку возможностей использования различных моделей для описания имеющихся данных социологического опроса с учетом полученной значимости признаков, показали, что в качестве таких моделей *не подходят* модели количественного характера типа регрессионных или классических нейросетевых. Наиболее подходящей представляется модель в виде совокупности кластеров. Такие кластеры, соответствующая информация о которых представлена в табл. 6.5, выявлены с помощью пакета Excel Neural Package при использовании самоорганизующейся карты Кохонена.

Таблица 6.5

Информация о выявленных кластерах

Центр кластера	Кластер				
	1	2	3	4	5
x_1	39,2	39,6	32,3	40,5	66,9
x_2	1,45	1,78	1,56	1,48	1,51
x_3	4,1	4,1	3,3	2,38	2,76
x_6	3,25	2,3	2,5	1,58	1,75
x_8	10,4	11,7	3,58	3,24	9,88
x_9	4,05	3,06	3,54	3,72	1,07
x_{10}	3,84	4,15	4,7	3,15	1,05
x_{11}	1480	492	609	348	480
x_{12}	3,06	1,53	1,99	1,18	1,47
x_{13}	1,1	1,53	1,27	1,91	1,63
x_{14}	2,9	3,29	2,42	4,03	3,91
Количество элементов в кластере	204	377	279	256	384

С учетом выявленных кластеров и значимости факторов полученным результатам можно дать следующую интерпретацию.

Во-первых, по социальной ориентации (т.е. по усредненной величине показателя x_{14}) все опрашиваемые лица могут быть разделены на три группы:

- разделяющие социалистические и коммунистические идеи и взгляды;
- капиталистической ориентации;
- национально-патриотической ориентации, полагающие, что Россия должна развиваться своим особенным путем.

Во-вторых, первая группа, в свою очередь, включает в себя две подгруппы, соответствующие кластерам 4 и 5.

Кластер 5 – это пенсионеры (среднее значение x_1 = 66,9 лет), как мужчины, так и женщины, со средним или специальным средним образованием, имеющие невысокую пенсию (x_{11} = 480 руб.), проживающие, в основном, в небольших городах и поселках городского типа.

Кластер 4 образован лицами, наиболее активно поддерживающими коммунистические взгляды. Особенность его состава: средний возраст x_1 = 40,5 лет, мужчин и женщин – поровну, образовательный ценз – низкий, профессии – рабочие, заработки очень низкие (x_{11} = 348 руб.), проживающие в поселках городского типа.

Кластеры 4 и 5 объединяют примерно 40% опрошенных лиц.

В-третьих, вторая группа включает в себя кластеры 1 и 3.

Кластер 1: средний возраст около 40 лет, поровну мужчин и женщин, образование высшее или незаконченное высшее, специалисты или руководители, работающие на предприятиях, принадлежащих государству или городу, высокий средний доход (1480 руб.), проживание в городах.

Кластер 3: средний возраст около 30 лет, поровну мужчин и женщин, образование среднее специальное и/или высшее, работники сферы обслуживания и рабочие, средний доход – 609 руб., проживание – в городах. Лица данной подгруппы настроены наиболее «прокапиталистически».

Кластеры 1 и 3 объединяют чуть менее 40% опрошенных.

В-четвертых, третья группа образована лицами, отнесенными к *кластеру 2*. Его характеристики: средний возраст около 40 лет, в основном женщины с высшим и незаконченным высшим образованием, специалисты, с невысоким доходом (492 руб.), проживание – в городах и поселках городского типа.

В данный кластер входят несколько более 20% опрошенных.

Итак, к коммунистическому (социалистическому) электорату относятся, в основном, пенсионеры или люди среднего возраста с низкими доходами. В рассматриваемом регионе общее количество лиц данных категорий – около 40%.

Факторами, определяющими «прокапиталистические» взгляды, является высокий уровень образования и доходов.

Выводы.

Использование чисто статистических подходов для анализа социологических процессов представляется не вполне надежным.

Предсказание социальной ориентации отдельной персоны по косвенным показателям с удовлетворительной точностью сделать, по-видимому, невозможно. Гораздо легче прогнозировать поведение группы лиц.

Наиболее подходящей моделью для рассматриваемого типа задач является модель в виде совокупности кластеров.

Выявление наиболее влияющих на социальную ориентацию признаков и кластеров по группам лиц может быть использовано для прогнозирования их социального поведения.

6.3. Выявление показателей, влияющих на валовую прибыль предприятия

6.3.1. Постановка задачи

Требуется: на основании экспертных данных, отраженных в табл. 6.6, выявить факторы, наиболее влияющие на ежемесячную прибыль предприятия.

Таблица 6.6

Исходные данные к анализу (показатели за 1998 г.)

№	Наименование фактора	Един. измерения	1	2	3	4	5	6	7	8	9	10	11	12
1	Объем реализации (без НДС)	тыс. рублей	354	310	277	302	327	211	263	168	278	292	305	326
2	в том числе бюджет	тыс. рублей		20		37		27			30	18	10	19
3	Затраты, в том числе:	тыс. рублей	255	281	319	251	215	203	208	172	323	262	239	475
4	материалы	тыс. рублей	53	58	44	63	38	32	41	33	45	50	39	58
5	заработная плата	тыс. рублей	122	126	126	104	112	74	102	76	123	117	107	218
6	Численность	чел.	59	62	62	63	62	62	62	63	63	62	62	62
7	Производительность	руб/чел	6003	5002	4474	4798	5273	3410	4237	2673	4406	4711	4833	5253
8	Цена ед. продукции	руб.	0,08	0,08	0,08	0,06	0,07	0,06	0,08	0,08	0,12	0,10	0,15	0,15
9	Рентабельность	%	38,9	10,4	-	20.4	52,0	4,1	26,3	-	-	11,5	27,4	-
10	Курс $	руб.	6,0	6,1	6,1	6,1	6,2	6,2	6,2	7,9	16,1	16,0	17,9	20,1
11	Прибыль валовая	тыс. рублей	99	29	-42	51	112	8	55	-4	-45	30	66	-149

6.3.2. Анализ технического задания

Изучение содержательной части задачи. Учитывая малое количество (12 – за 12 месяцев 1998 г.) наборов экспериментальных данных, на первом этапе исследования было проведено изучение представленных показателей. В результате ряд следующих показателей был исключен из дальнейшего рассмотрения:

- объем реализации по линии бюджета – поскольку данные являются неполными, содержат пропуски;
- затраты – поскольку более целесообразно рассматривать основные составляющие затрат: затраты на материалы и заработную плату;
- объем реализованной продукции – так как он связан жесткой аналитической зависимостью с другими показателями (прибыль = объем реализованной продукции – затраты);
- численность – из-за постоянства этого показателя в течение года;
- цена единицы продукции – фактор малоинформативен;
- рентабельность – этот показатель исключен по тем же причинам, что объем реализованной продукции.

В результате для дальнейшего анализа оставлены факторы:

- затраты на материалы;
- объем заработной платы;
- производительность;
- курс доллара США.

Выбор метода исследования. Учитывая, что в рассматриваемой ситуации ни о каких вероятностных предпосылках говорить не приходится (данных мало, скорее всего по годам объект исследования – нестационарен, непонятно, как определить генеральную совокупность и соответствующий закон распределения), в данном случае неприменимы известные статистические методы исследования. В соответствии с этим, в качестве метода исследования принят нейросетевой подход с использованием в качестве инструментального средства пакета НейроПро.

6.3.3. Анализ данных

В соответствии с отобранными показателями и выходной величиной (валовой прибылью) с помощью MS Excel, составляется таблица вида табл. 6.7. При ее подготовке указывается числовой формат ячеек с двумя знаками после запятой, а сама таблица сохраняется в формате Dbase, например, с именем Предприятие.dbf.

Таблица 6.7

Таблица данных для анализа

Материалы	Зар. плата	Производство	Курс $	Прибыль
53,00	122,00	6003,00	6,00	99,20
58,00	126,00	5002,00	6,10	29,10
44,00	126,00	4474,00	6,10	-41,60
63,00	104,00	4798,00	6,10	51,30
38,00	112,00	5273,00	6,20	111,90
32,00	74,00	3410,00	6,20	8,40
41,00	102,00	4237,00	6,20	54,70
33,00	76,00	2673,00	7,90	-3,60
45,00	123,00	4406,00	16,10	-45,40
50,00	117,00	4711,00	16,00	30,10
39,00	107,00	4833,00	17,90	65,50
58,00	218,00	5253,00	20,10	-149,30

Структура нейронной сети определяется так же, как и в предыдущем примере: будем полагать что сеть содержит один скрытый слой при числе входов – 4, числе выходов – 1; требуемое число нейронов N рассчитывается по формулам (1.5), (1.6), где в данном случае N_p = 12, N_x = 4, N_y = 1. При расчете получаем значение N = 0,52; округляя, принимаем число нейронов N = 1.

Запускаем программу НейроПро. На экране появляется окно программы. В меню **Файл** выбираем опцию **Создать**. Появляется окно с заголовком **Без имени**, в котором нажимаем клавишу **Открыть файл данных**. Затем находим и открываем файл созданной таблицы (Предприятие.dbf). Далее в окне **Без имени** выбираем опцию **Новая сеть**. Появляется окно с **заголовком Создание нейронной сети**. В данном окне сначала проверяем правильность определения программой входных факторов и выхода, а затем выбираем опцию **Структура сети** с указанием числа слоев нейронов – 1, числа нейронов – 1. Далее в этом же окне указываем имя сети и нажимаем клавишу **Создать**.

Затем обучим созданную сеть, выбрав опции **Нейросеть** и **Обучение**. Откроется окно **Обучение Предприятие**, демонстрирующее процесс обучения нейронной сети. После его завершения нажимаем клавишу **Готово**. Затем в меню программы можно выбрать опции **Нейросеть** и **Тестирование**, чтобы убедиться в качестве работы обученной сети. В нашем случае это приведет к выдаче таблицы вида рис. 6.4.

В первой колонке – результаты, используемые при обучении, во второй – прогноз сети. Как видно, точность прогноза не очень высока, что объясняется малым объемом обучающей выборки из-за невыполнения условия значительной малости числа оставленных показателей по сравнению с объемом выборки.

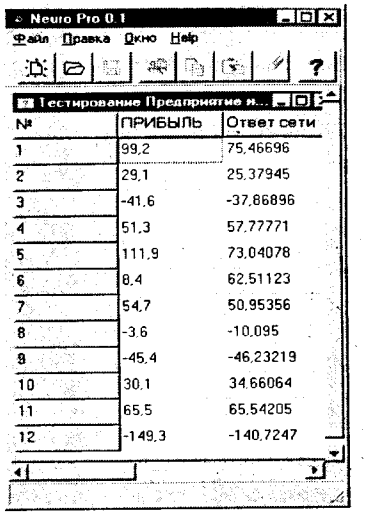


Рис. 6.4. Результаты тестирования нейронной сети

В комментариях полученное, по-видимому, не нуждается.

После этого опять необходимо щелкнуть мышью по окну **Без имени**, войти в меню программы и выбрать опции **Нейросеть**, **Значимость входов**. Получим окно, отражающее результаты анализа (рис. 6.5).

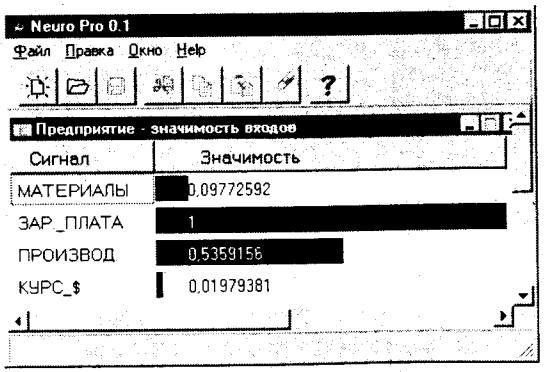


Рис. 6.5. Значимость факторов

Очевидно, наибольшее влияние на ежемесячную прибыль предприятия оказывает общий объем месячной заработной платы сотрудников. Вдвое меньше влияет производительность. Расходы

на материалы и курс доллара США на прибыль практически не влияют. Данный вывод завершает проведенное исследование и может быть использован руководством предприятия для принятия каких-либо решений.

6.4. Задача об ирисах Фишера

Данная задача относится к классическим задачам классификации; ее содержательная постановка взята из книги Бэстенс Д.-Э., ван ден Берг В.-М., Вуд Д. Нейронные сети и финансовые рынки: принятие решений в торговых операциях. – М.: ТВП, 1997.

6.4.1. Содержательная постановка задачи

Имеются данные измерений для трех видов ирисов (iris setosa, iris versicolor, iris virginica), в равных пропорциях (по 50 штук). Известны величины измерений четырех признаков: длина чашелистика (SL), ширина чашелистика (SW), длина лепестка (PL) и ширина лепестка (PW).

Требуется: используя все или часть этих данных и применяя нейросетевой подход, построить автоматический классификатор, относящий каждый вновь предъявляемый цветок к одному из трех видов на основании перечисленных признаков.

6.4.2. Построение нейросетевого классификатора

Выполним решение задачи, используя описанный выше нейропакет MPIL.

1) Подготовка исходных данных.

Используя 120 из 150 данных измерений, подготовим текстовый файл iris2.net для обучения нейронной сети. В соответствии с изложенным в разд. 5.6, начало файла будет иметь вид:

```
#
Ninputs 4 Noutputs 3
NTrainingPatterns 120
0.222 0.625 0.068 0.042  1.0 0.0 0.0
0.167 0.417 0.068 0.042  1.0 0.0 0.0
0.111 0.500 0.051 0.042  1.0 0.0 0.0
0.000 0.417 0.017 0.000  1.0 0.0 0.0
0.417 0.833 0.034 0.042  1.0 0.0 0.0
0.389 1.000 0.085 0.125  1.0 0.0 0.0
0.306 0.792 0.051 0.125  1.0 0.0 0.0
0.222 0.625 0.068 0.083  1.0 0.0 0.0
0.389 0.750 0.119 0.083  1.0 0.0 0.0
                . . .
```


Первые четыре столбца соответствуют числовым значениям четырех входных переменных, а три последующих – трем видам ириса. Единица означает принадлежность цветка к данному виду.

Используя все исходные данные, подготовим файл для тестирования сети (см. разд. 5.6) – iris.tes (отметим, что файлы iris2.net, iris.tes поставляются в комплекте с программой MPIL).

2) Обучение сети.

Запустим программу MPIL. Используя правила работы с программой, описанные в разд. 5.6, обучим нейронную сеть, сохраняя опции обучения по умолчанию.

Используя, далее, файл iris.tes, проанализируем точность классификации обученной сети, выбирая для этого опцию меню **Test/Check Accuracy** и указывая в диалоге в качестве тестирующего файла iris.tes, а в качестве файла протокола – iris.out. Результат тестирования отражается сообщением вида рис. 6.6.

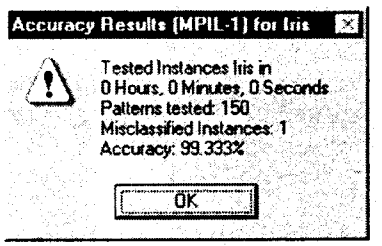


Рис. 6.6. Сообщение о результате тестирования обученной нейронной сети

Из данного сообщения следует, в частности (и это подтверждается при просмотре файла iris.out), что из 150 предъявленных ошибочно классифицирован только один пример. Полученную точность можно считать очень высокой.

Заметим, что при обучении сети в режиме Mpil–2 генерируется набор из 13 логических правил классификации ириса.

Теперь обученную сеть можно сохранить и использовать далее по назначению. Точно так же строится классификатор для других подобных задач.

6.5. Задача о землекопах

Задача о землекопах представляет собой простой пример нахождения оптимального решения с использованием нейросетевого подхода. Соответствующие файлы и постановка задачи приведены как иллюстрация для нейропакета Neural Planner.

6.5.1. Содержательное описание задачи

Предположим, что землекопам необходимо выкопать яму определенного диаметра. Возможное количество землекопов от 1 до 3, количество лопат у них от 1 до 4. Выходной величиной является производительность работы землекопов, выражающаяся в весе вынутого за час работы грунта, т. е. в кг/час. На производительность влияют не только количество землекопов и имеющихся лопат, но и диаметр ямы, состав грунта (камни, глина) и температура окружающей среды.

Действительно, землекопы работают лучше, если каждый из них имеет по лопате, чем когда имеется одна лопата на всех. Каменистую почву рыть труднее, но камни весят больше, чем земля, а в качестве показателя процесса выступает величина, связанная с весом, а не с объемом. Глинистую почву также копать труднее, но глина тоже имеет больший удельный вес, чем земля, хотя не настолько, сколько камни. Холодную почву рыть труднее; еще труднее рыть замерзшую почву. С другой стороны, землекопы нуждаются в большем числе перерывов на отдых, когда температура воздуха высока. Если диаметр ямы слишком мал, число работающих землекопов ограничено. С другой стороны землекопы могут отдыхать по очереди.

Таким образом, на производительность труда (Kgms/hour) в данном случае влияет, по крайней мере, шесть входных факторов: число землекопов (Diggers), число лопат (Spades), диаметр ямы (Diameter, Cms), скалистость грунта (Rocks, %), глинистость почвы (Clay, %), температура воздуха (Temp, ^{0}C).

Требуется: определить, при каких значениях перечисленных факторов из ряда возможных вариантов производительность труда будет наивысшей.

Заметим, что аналитическим путем задача вряд ли может быть решена с приемлемой точностью, однако, аппарат нейронных сетей позволяет получить решение достаточно легко.

6.5.2. Решение задачи

Для решения поставленной задачи воспользуемся нейропакетом Neural Planner и файлом diggers.tti, который, как отмечено, поставляется вместе с пакетом. Содержание файла приведено ниже; его структура соответствует описанию в разд. 5.4.

Как видно, обучающая выборка (секция файла [TRAINING]) содержит 17 образцов; секция опроса ([INTERROGATING]) – 11 возможных вариантов, из которых и нужно определить наилучший.


```
Digging holes for NPlan

[LABELS]:
[Diggers] [Spades] [Diameter (Cms)] [Rocks (%)] [Clay (%)] [Temp (C)] [Kgms/hour]
[END]

[TESTING]          >        >          >          >          >          >          <
[END]

[INTERROGATING]    >        >          >          >          >          >          <
[Hole a]           1        1          250        20         15         18         ?
[Hole b]           1        2          300        20         20         20         ?
[Hole c]           1        3          200        40         20         15         ?
[Hole d]           2        4          200        20         10         25         ?
[Hole e]           2        1          300        10         10         5          ?
[Hole f]           2        2          300        50         15         10         ?
[Hole g]           3        3          300        50         20         15         ?
[Hole h]           3        4          250        30         10         20         ?
[Hole i]           3        1          200        40         15         10         ?
[Hole j]           3        2          150        40         15         10         ?
[Hole k]           3        3          150        40         15         10         ?
[END]

[TRAINING]         >        >          >          >          >          >          <
[Hole 01]          1        1          200        20         10         3          1000
[Hole 02]          1        2          250        20         20         5          1100
[Hole 03]          1        3          150        10         20         20         1600
[Hole 04]          1        4          200        5          15         25         1500
[Hole 05]          1        1          300        30         10         20         1200
[Hole 06]          1        2          150        10         20         20         1500
[Hole 07]          2        3          200        50         0          18         1700
[Hole 08]          2        4          150        40         10         15         1500
[Hole 09]          2        1          200        10         50         15         1000
[Hole 10]          2        2          150        10         20         20         1900
[Hole 11]          3        3          200        20         10         10         2100
[Hole 12]          3        4          300        20         5          14         2600
[Hole 13]          3        1          200        10         20         15         1500
[Hole 14]          3        2          150        10         15         25         1600
[Hole 15]          1        4          200        20         10         3          1000
[Hole 16]          2        3          150        10         20         20         1900
[Hole 17]          3        4          200        20         10         10         2100
[END]

[LIMITS]           >        >          >          >          >          >          <
[highs]            3        4          300        50         50         25         2600
[lows]             1        1          150        5          0          3          1000
[END]
```

Определим топологию сети. Выберем нейронную сеть с одним скрытым слоем. По формулам (1.5), (1.6) определим, что при заданных значениях $N_x = 6$, $N_y = 1$, $N_p = 17$ число нейронов в скрытом слое $N < 4$.

Запустим программу Neural Planner. Используя правила работы с нейропакетом, изложенные в разд. 5.4, построим нейронную сеть вида рис. 6.7 и проведем ее обучение, используя установки по умолчанию.

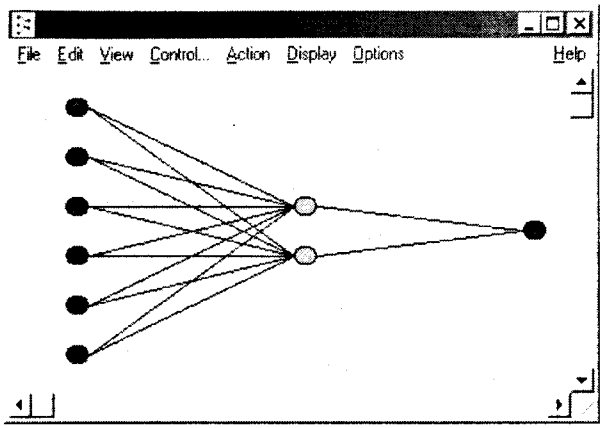


Рис. 6.7. Вид нейронной сети к задаче о землекопах

Выберем опцию меню **Action/Interrogate**. В появившемся окне установим флажок в окне **Circle**, выделим мышью заголовок в правой части окна Kgms/hour и нажмем кнопку **Seek High**. Это запустит циклический поиск наибольшего значения выхода. Результат проделанных действий отображается рис. 6.8.

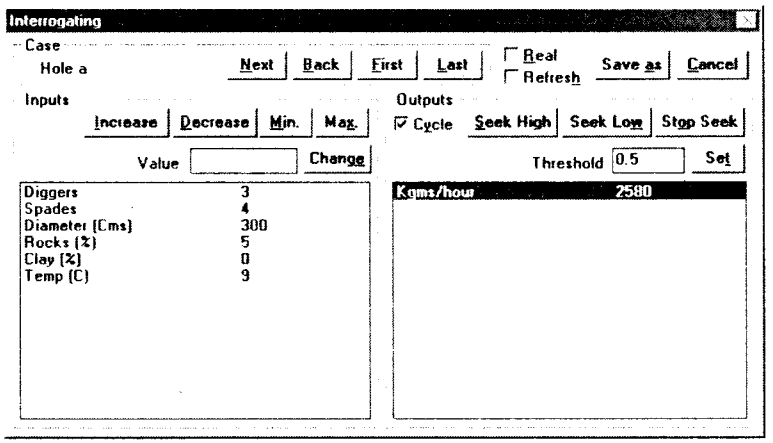


Рис. 6.8. Результат поиска оптимального решения

Как видно, наибольшая производительность труда (2580 кг/час) из заданных вариантов обеспечивается при следующих условиях: 3 землекопа, 4 лопаты, скалистость грунта 5%, отсутствие в почве глины, температура воздуха 9°С.

Таким образом, поставленная задача решена, и ее результаты (т. е. обученную нейронную сеть) можно сохранить.

Интересно заметить, что в примере решения данной задачи, поставляемом вместе с нейропакетом, нейронная сеть имеет 6 нейронов в скрытом слое и обеспечивает нахождение максимума в 2563 кг/час при другом варианте набора входных факторов.

6.6. Аппроксимация функции

Рассмотрим пример аппроксимации функции двух переменных с использованием нейропакета Neural Planner:

$$F(X, Y) = (1 - X^2) + 2(1 - Y)^2.$$

Ограничим диапазон изменения переменных X и Y интервалом (–1, 1).

Для решения задачи выберем топологию нейронной сети с двумя скрытыми слоями по четыре нейрона в каждом слое, с двумя входными и одним выходным нейроном (рис. 6.9).

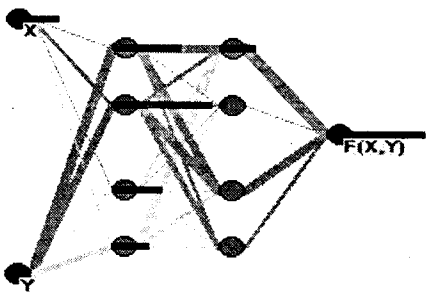


Рис. 6.9. Структура нейронной сети

Для обучения сети используем выборку из 20 векторов, приведенных в табл. 6.8. Время обучения сети при заданной ошибке 0,05 (остальные установки программы – по умолчанию) составило около 1 мин. Сеть обучилась за 4500 циклов.

Для проверки полученных результатов проведем опрос сети. Результаты опроса приведены в табл. 6.9.

Таблица 6.8

Содержание файла с обучающей выборкой

[LABELS]	[X]	[Y]	[F(X,Y)]
[END]			
[TRAINING]	>	>	<
[1]	-1	-1	8
[2]	-0.8	-1	8.129
[3]	-1	-0.8	6.48
[4]	-0.6	-0.5	4.909
[5]	-0.7	-0.6	5.38
[6]	-0.4	-0.5	5.2
[7]	-0.5	-0.3	3.94
[8]	-0.3	-0.2	3.708
[9]	-0.2	-0.1	3.342
[10]	-0.1	0	2.98
[11]	0	0	3
[12]	0.1	0	2.98
[13]	0.1	0.2	2.26
[14]	0.2	0.3	1.902
[15]	0.4	0.5	1.206
[16]	0.6	0.3	1.389
[17]	0.6	0.6	0.729
[18]	0.7	0.8	0.34
[19]	0.9	1	0.0361
[20]	1	1	0
[END]			
[LIMITS]	>	>	<
[highs]	1	1	8.129
[lows]	-1	-1	0
[END]			

Таблица 6.9

Результаты опроса сети

Case Name	X	Y	F(X,Y)
1	0.5	0.5	0.958084
2	0.00999999	0.00999999	2.40566
9	0.1	0.95	0.656095
4	0.3	0.3	1.42821
5	-0.9	-1	7.66945
6	-0.2	0.9	0.839218
7	0.1	-0.4	4.47549
8	-0.9	-0.3	3.93202
3	0	0.2	1.84074
10	-0.34	0	2.61335

Из табл. 6.9. видно, что результаты опроса не во всех случаях верны или ошибка намного больше заданной, например, при X = 0,5, Y = 0,5 (значения вектора 1) имеем $F(X, Y)$ = 0,958, а точное значение равно 1,0625, т. е. ошибка составляет примерно 0,1 (при заданной ошибке обучения 0,05). Заметим, что точность аппроксимации можно повысить, увеличив объем обучающей выборки.

6.7. Нейросетевая экспертная система

В настоящее время известно много удачных примеров применения нейросетевого подхода для построения интеллектуальных информационных систем и, в частности, экспертных систем.

Комбинированное использование экспертной системы и аппарата искусственных нейронных сетей обеспечивает необходимую гибкость и самообучение на основе знаний, в то же время, полученные от экспертов знания позволяют существенно упростить структуру нейронных сетей, уменьшить число нейронов и связей в сети.

Например, медицинские нейросетевые экспертные системы проявили себя как серьезный соперник традиционных экспертных систем, составляя конкуренцию квалифицированным экспертам. Исследования в Боткинской больнице (С.-Петербург), проведенные с использованием разработанной нейросетевой экспертной системы, показали ее превосходные возможности по диагностике некоторых классов болезней, которые плохо диагностируются врачами. Результаты проверки свидетельствуют о высокой достоверности результатов, достигаемых такими системами (до 94%).

Рассмотрим пример постановки задачи для экспертной системы. Многих людей беспокоят боли в спине, которые часто возникает внезапно и без определенной причины. Обычно трудно определять характер недомогания из-за отсутствия признаков какого-либо конкретного заболевания. Боль обычно исчезает после короткого отдыха. Врачи называют это неопределенными болями в спине. Такие недомогания – основная причина потерянных рабочих дней. Люди, занимающиеся тяжелым физическим трудом, связанным, например, с поднятием тяжестей более подвержены периодически возникающим болям в спине. Это может также беспокоить тех, кто проводит много времени без движения. В любом случае, боли в спине могут иметь много причин. Большое значение при этом имеет правильно поставленный диагноз.

Целью построения экспертной системы является диагностика заболевания, которое может быть причиной периодических болей в спине.

Принцип построения экспертной системы на базе нейронной сети состоит в следующем. Составляются вопросы, ответы на которые имеют бинарный вид, т. е. «Да» или «Нет». При составлении «вектора опроса», если при диагностике следует ответ «Да», то компоненту вектора присваивается 1, если «Нет», то – 0.

Согласно вышесказанному, решающее «дерево», приведенное на рис. 6.10, может быть записано в виде трех векторов:

{..., 1, 1, 0, ...}, {..., 1, 0, 1, ...} и {..., 0, ...}.

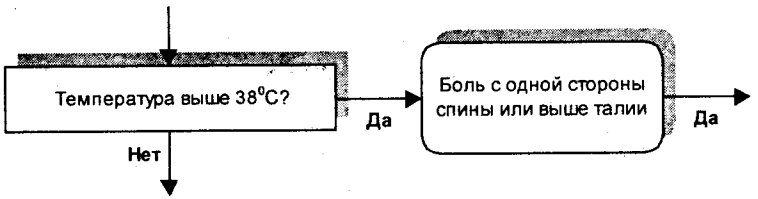


Рис. 6.10. Часть решающего дерева

Первые две записи (векторы) передают следующий смысл:

ЕСЛИ:

Пациент имеет температуру выше 38°С, и чувствует боль только с одной стороны спины

ИЛИ:

Пациент чувствует недомогание,

ТО: ...

Третья запись (вектор) передает:

ЕСЛИ:

Пациент имеет температуру меньше 38°С.

ТО: ...

Составим простые вопросы для диагностики возможных причин болей в спине. Входные векторы в лингвистической форме будут при этом иметь следующий вид:

1.1) Боль возникает после подъема тяжести?

И/ИЛИ:

1.2) После истощающего физического упражнения?

2) Температура выше 38 °С?

3.1) Пациент старше 60 лет?

И/ИЛИ:

3.2) Пациент провел несколько недель в кровати или в инвалидном кресле?

4) Пациент старше 45 лет?

5) Боль сильнее утром?

6.1) Пациенту мешает боль при ходьбе?

ИЛИ:

6.2) Боль чувствуется только в одной ноге?

7) Боль ограничена главным образом в спине и не распространяется где-нибудь еще?

8.1) Боль – только с одной стороны спины, выше талии?

И:

8.2) Чувство тошноты?

9) Боль намного сильнее с одной стороны позвоночника?

10) Обычно болит шея или спина между плечами?

Аналогично можно построить вектор выходных значений, руководствуясь тем же самым правилом, что и для входных. Если на выходе нейронной сети, соответствующем какому-либо диагнозу получаем 1, то на данный диагноз следует обратить внимание, так как он может быть причиной боли.

Вектор выходных переменных (диагноз) имеет следующие компоненты:

1) Ишиас, вызванный давлением на корень седалищного нерва; необходима консультация у врача.

2) Возможный люмбаго (прострел), вероятно вызванный сильным напряжением спины.

3) Возможна инфекция почек, или боли могут являться сопровождением общего вирусного воспаления; необходима срочная консультация врача.

4) Боль в спине (возможно очень сильная), может быть следствием какого либо вирусного заболевания, например гриппа; необходима консультация врача.

5) Возможно повреждение кости в результате какой либо травмы; необходима консультация врача.

6) Возможен артрит позвонков шеи.

7) Возможен остеоартрит в нижней части груди, почек или позвоночника.

8) Возможно хроническое воспаление суставов.

9) Причина боли не выяснена; необходима консультация врача.

Полное решающее «дерево» для экспертной системы приведено на рис. 6.11. Сеть, реализующая эту систему с помощью нейропакета Neural Planner, приведена на рис. 6.12. Сеть обучена при помощи 18 обучающих векторов, которые приведены в табл. 6.10. Для опроса сети необходимо закодировать в двоичном виде вопросы, а затем проделать обратную операцию с ответами.

Время обучения сети при заданной ошибке 0,05 составило около 2 мин. Сеть обучилась за 7500 циклов.

Для проверки полученных результатов был проведен следующий опрос сети.

Постановка и кодирование вопросов:

1.1) Боль возникает после подъема тяжести? Да – 1.

И/ИЛИ:

1.2) После истощающего физического упражнения? Да – 1.

2) Температура выше 38°С? Нет – 0.

3.1) Пациент старше 60 лет? Нет – 0.

И/ИЛИ:

3.2) Пациент провел несколько недель в кровати или в инвалидном кресле? Нет – 0.

4) Пациент старше 45 лет? Нет – 0.

5) Боль сильнее утром? Да – 1.

6.1) Пациенту мешает боль при ходьбе? Да – 1.

ИЛИ:

6.2) Боль чувствуется только в одной ноге? Нет – 0.

7) Боль ограничена главным образом в спине и не распространяется где-нибудь еще? Да – 1.

8.1) Боль – только с одной стороны спины, выше талии? Да – 1.

И:

8.2) Чувство тошноты? Нет – 0.

9) Боль намного сильнее с одной стороны позвоночника? Нет – 0.

10) Обычно болит шея или спина между плечами? Нет – 0.

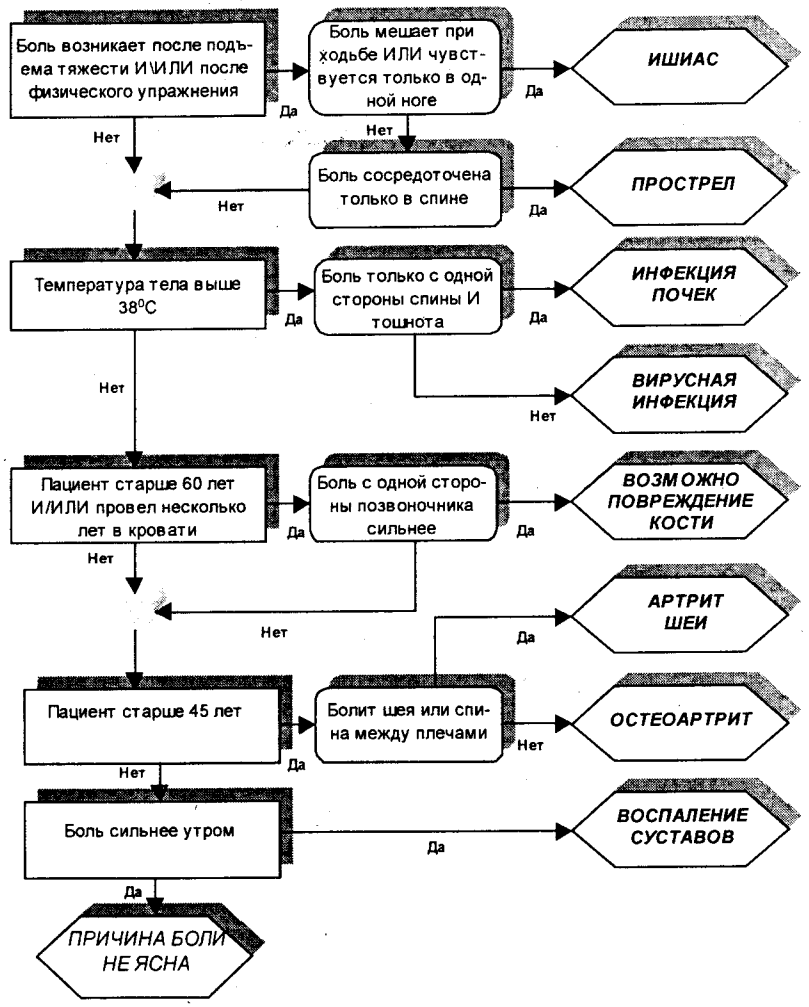


Рис. 6.11. Полное решающее дерево

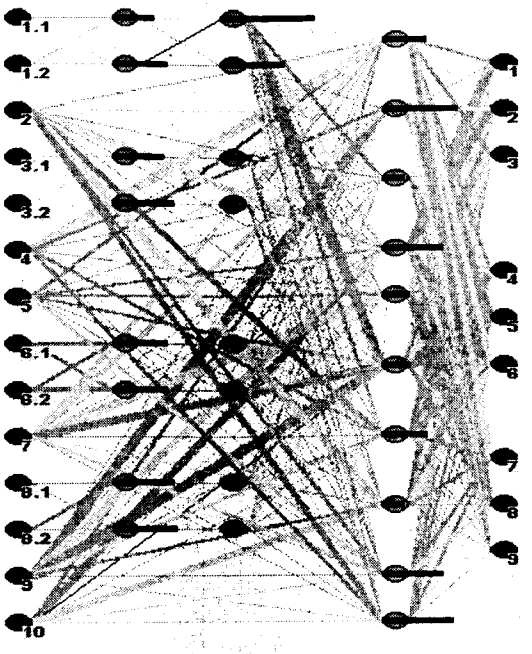


Рис. 6.12. Нейронная сеть, реализующая экспертную систему

В результате опроса сети получен следующий диагноз:

1) Ишиас, вызванный давлением на корень седалищного нерва; необходима консультация у врача. Нет – 0.

2) Возможный люмбаго (прострел), возможно вызванный сильным напряжением спины. Да – 1.

3) Возможна инфекция почек, или боли могут являться сопровождением общего вирусного воспаления; необходима срочная консультация врача. Нет – 0.

4) Боль в спине (возможно очень сильная), может быть следствием какого либо вирусного заболевания, например гриппа; необходима консультация врача. Нет – 0.

5) Возможно повреждение кости в результате какой либо травмы; необходима консультация врача. Нет – 0.

6) Возможен артрит позвонков шеи. Нет – 0.

7) Возможен остеоартрит в нижней части груди, почек или позвоночника. Нет – 0.

8) Возможно хроническое воспаление суставов. Нет – 0.

9) Причина боли не выяснена; необходима консультация врача. Нет – 0.

Следует отметить, что только врачи могут оценить правильность ответов подобной экспертной системы.

Таблица 6.10

Содержание файла с обучающей выборкой

LABELS	1.1	1.2	2	3.1	3.2	4	5	6.1	6.2	7	8.1	8.2	9	10	1	2	3	4	5	6	7	8	9
END																							
TESTING	>	>	>	>	>	>	>	>	>	>	>	>	>	>	<	<	<	<	<	<	<	<	<
dummy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
END																							
INTERROGATING	>	>	>	>	>	>	>	>	>	>	>	>	>	>	<	<	<	<	<	<	<	<	<
dummy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	1	1	0	1	1	0	0	0	?	?	?	?	?	?	?	?	?
END																							
TRAINING	>	>	>	>	>	>	>	>	>	>	>	>	>	>	<	<	<	<	<	<	<	<	<
dummy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
7	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
10	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
12	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
13	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
14	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
15	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
16	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
17	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
END																							
LIMITS	>	>	>	>	>	>	>	>	>	>	>	>	>	>	<	<	<	<	<	<	<	<	<
Highs	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Lows	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
END																							

6.8. Прогнозирование на финансовом рынке

Рассмотрим пример создания нейронной сети для финансового прогнозирования, разобрав методику использования средств пакета BrainMaker 3.1 (пример взят из статьи: Блинов С. Практикум применения пакета BrainMaker для прогнозирования на финансовых рынках //http://win.aha.ru/~mdo/office/bm_fin.htm).

6.8.1. Построение нейросетевой модели

Первым этапом при разработке нейронной сети является определение того, что же она должна делать. В рассматриваемом примере целью создания нейронной сети будет являться прогнозирование изменения значения индекса цен на акции вымышленной фирмы Bart-Davis-100 (BD100).

Вторым этапом является определение состава исходных данных и сбор этих данных. Предположим, что эти данные подготовлены в файле price1.dat. Файл представляет собой текстовую таблицу, в которой находятся упорядоченные по строкам и столбцам технические индикаторы, индексы и цены. Каждой строке соответствуют данные одного дня. В столбцах: day, BD100, price1, price2, index1D, index2D, line, streng, utilD, transpD содержатся данные одного типа.

Столбец day содержит порядковый номер дня и не используется в качестве входных данных. Записи сделаны за 185 дней.

Столбец BD100 содержит значения, которые одновременно являются как входными данными, так и результатом работы нейронной сети. Поэтому в дальнейшем придется добавить еще один столбец, содержащий будущие известные значения BD100, на которых можно обучать нейронную сеть.

Поскольку целью является ознакомление с приемами работы в BrainMaker, а данные взяты условно, смысл остальных величин не имеет значения.

На третьем этапе проводится анализ и предварительная обработка данных. Это делается в программе NetMaker.

Четвертый, пятый и шестой этапы – обучение, тестирование и использование нейронной сети будут выполнены в самом BrainMaker и рассмотрены позднее.

6.8.2. Предварительный анализ и подготовка данных

Загрузка данных.

1) Запустите NetMaker.

2) Выберите в главном меню опцию **Read in Data File**.

3) Загрузите файл price1.dat.

4) Перейдите в режим **Manipulate Data**.

Построение графика.

1) В пункте **Operate** главного меню выберите **Graph Columns**. Появится окно, в заголовке которого будет запрос **Select Columns To Graph**.

2) Щелкните мышью на столбец BD100. Обратите внимание, что теперь значение поля trace1 изменилось на BD100.

3) Выберите **Make Graph**.

4) Исправьте текст «Index» на «Fact».

5) Выберите **Trace ranges**, а затем **Create Plot**. В окне появится график, изображенный на рис. 6.13.

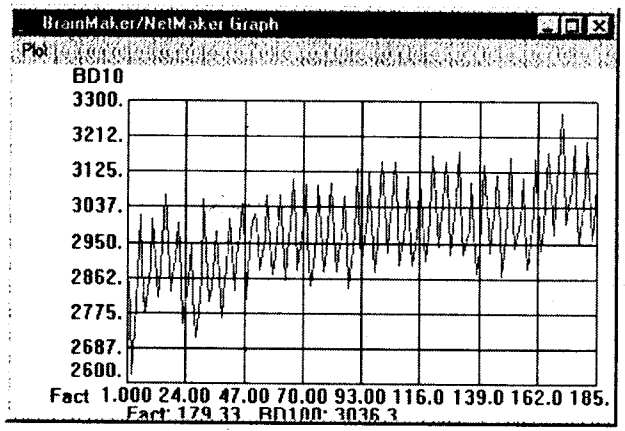


Рис. 6.13. График изменения BD100

На горизонтальной оси показано время, на вертикальной – значения BD100. Таким же образом можно построить графики price1 и price2. Значение BD100 сильно колеблется, но при этом общее движение цены (тренд) направлено вверх.

Создание столбца с изменениями цен.

Будем использовать не абсолютные значения BD100, а только их изменения по сравнению с предыдущим днем. То же самое относится к столбцам price1, price2. Таким образом, нужно добавить три новых столбца, в которых будут содержаться изменения абсолютных значений.

1) В пункте «Operate» главного меню выберите **Difference Columns**, а в подменю – **Simple**.

2) В ответ на запрос **Select Column** выберите BD100.

3) Задайте смещение, равное 1.

4) Пометьте новый столбец BD100D. Он будет размещен в крайней правой колонке таблицы.

5) Сделайте то же самое с price1 и price2 и обозначьте новые столбцы: price1D и price2D. Столбцы index1D, index2D, utilD, transpD уже представлены в виде приращений.

Примечание. Поскольку данные в столбцах с изменениями цен получаются в результате вычитания значений предыдущего

дня из значений текущего дня, первый ряд будет содержать некорректные значения. Поэтому в дальнейшем мы его удалим.

Анализ цикличности данных.

Для более корректного предсказания будущие значения BD100, попробуем выявить присущие данному рынку циклы и в дальнейшем будем строить прогноз на один цикл вперед.

1) В пункте **Operate** главного меню выберите **Cyclic Analysis**.

2) Щелкните мышкой на столбец BD100.

3) Значения количества рядов в периоде, метки осей и режим **Graph Only** должны быть уже заданы по умолчанию, поэтому нажмите **Analyze**.

4) Выберите **Trace ranges**, а затем **Create Plot** на появляющихся запросах. В окне появится график вида рис. 6.14.

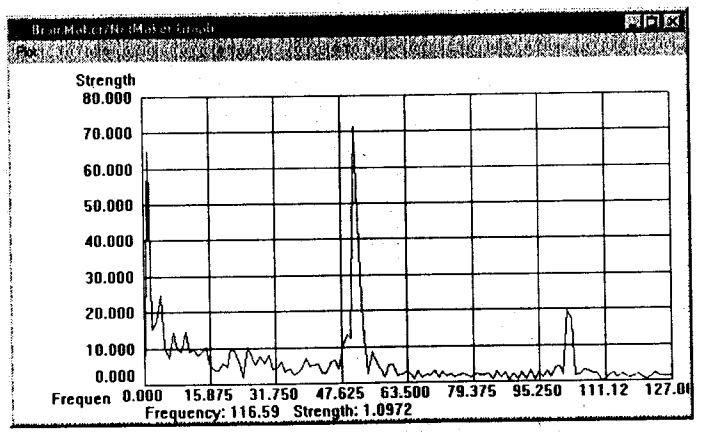


Рис. 6.14. График для определения взаимосвязи между переменными

На горизонтальной оси показана частота цикла, на вертикальной его «сила». Пик с частотой 1, т. е. цикл, появившийся один раз, соответствует общему движению рынка. Нам же нужно найти наиболее короткий ярко выраженный цикл. Этому циклу соответствует другой пик с частотой 51. Для того, чтобы его разглядеть получше, нужно изменить масштаб графика:

1) В меню окна с графиком нажмите **Plot**, выберите **Modify** в подменю, а затем перейдите в режим **Trace ranges**.

2) Измените **Plot Min** и **Plot Max** для **Frequency** на, скажем, 47 и 63.

3) После нажатия **Create Plot** можно разглядеть этот пик.

Найдем длину этого цикла. Общее количество дней – 185, частота – 51. Следовательно, длина цикла: 185/51 – приблизительно 4 дня.

Создание набора выходных данных для обучения.

Для обучения нейронной сети нужно для каждого набора входных данных, соответствующих одному дню, указать правильный результат прогнозирования – значение BD100D через 4 дня. Это делается путем смещения значений столбца BD100D вверх на 4 строки и записи результата в новый столбец.

1) В пункте **Column** главного меню выберите подпункт **Shift Column Up**.

2) Щелкните мышкой на столбец BD100D.

3) Укажите, что величина сдвига равна 4, а метка новой колонки – BD+4.

Сглаживание данных скользящей средней.

Для уменьшения случайного шума сгладим BD100D с помощью скользящего среднего.

1) В пункте **Operate** главного меню выберите **Moving Average**, а затем **Simple** в появившемся подменю.

2) Щелкните мышкой на столбец BD100D.

3) Для сглаживания рекомендуется использовать интервал, равный половине длины цикла. Поэтому укажите количество строк для построения скользящей средней равным 2, а новую колонку назовите Bdavg2.

Маркировка колонок.

Теперь надо указать назначение каждой колонки получившейся таблицы. Это делается или с помощью команд раздела **Label** главного меню или с клавиатуры. Ниже показано, каким образом должны быть маркированы колонки и какими комбинациями клавиш это можно сделать.

day	Annote	Ctrl+A
BD100	Not Used -	-
price1	Not Used -	-
price2	Not Used -	-
index1D	Input	Ctrl+I
index2D	Input	Ctrl+I
line	Input	Ctrl+I
streng	Input	Ctrl+I
utilD	Input	Ctrl+I
transpD	Input	Ctrl+I
BD100D	Input	Ctrl+I
price1D	Input	Ctrl+I
price2D	Input	Ctrl+I
BD+4	Pattern	Ctrl+P
Bdavg2	Input	Ctrl+I

Результат проделанных шагов запишите в файл price2.dat.

Анализ корреляции данных.

Для того, чтобы выявить взаимосвязи между входными данными и прогнозируемыми значениями BD100, проведем анализ корреляции между ними.

1) В пункте **Operate** меню войдите в **Data Correlator**.

2) На запрос о выборе колонок укажите index1D и BD100.

3) Значения количества рядов в периоде, метки осей и режим **Graph Only** должны быть уже заданы по умолчанию, поэтому нажмите **Correlate**.

4) Выберите **Trace ranges**, а затем **Create Plot** на появляющихся запросах. Появится график, изображенный на рис. 6.15.

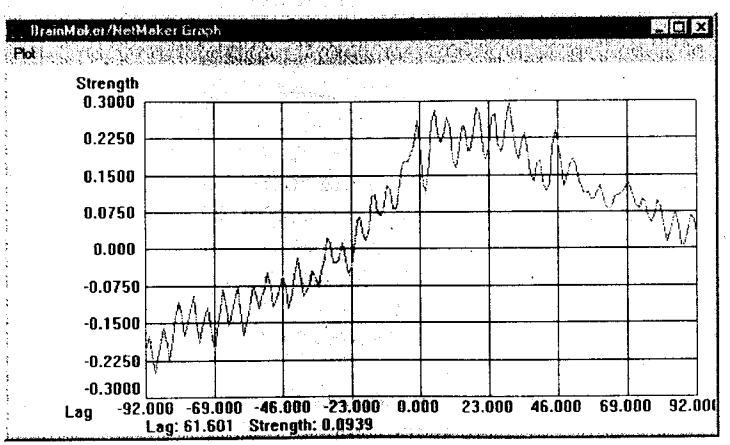


Рис. 6.15. График для выявления возможных циклов

На горизонтальной оси показано время. Единица времени (день) соответствует одной строке из входной таблицы. Нуль делит горизонтальную ось на будущее и прошедшее время относительно второй колонки (BD100), которая выбрана при построении графика.

На вертикальной оси представлена степень корреляции. Положительная корреляция означает, что если значения в первом столбце увеличиваются, то увеличиваются и значения во втором столбце, и, наоборот. Отрицательная корреляция означает, что если значения в первом столбце увеличиваются, то значения во втором столбце уменьшаются, и, наоборот. Степень корреляции может изменяться от –1 до 1.

Проанализируем полученный график. Обратите внимание на пик, расположенный немного левее нулевого значения на горизонтальной оси. Для того, чтобы его получше разглядеть, увеличьте масштаб графика. Временное смещение (лаг) равно –1, а степень корреляции – около 0,25.

Нулевой лаг означает, что изменения значений в первом и втором столбцах происходят одновременно. Отрицательный лаг означает, что сначала происходит изменение значений в первом столбце, а затем, через промежуток времени, равный лагу, изменяются значения второго столбца. Положительный лаг означает, что сначала происходит изменение значений во втором столбце, а затем, через промежуток времени, равный лагу, изменяются значения первого столбца. Таким образом, наибольший интерес представляет сильная корреляция с отрицательным лагом, при которой изменения входного параметра влекут за собой изменение прогнозируемой величины через какое-то время. Следовательно, можно использовать значения index1D для предсказания значений BD100 на 1 день вперед.

Примечание. Вообще-то нужна корреляция с отрицательным лагом не менее 4 дней, так как собираемся прогнозировать на 4 дня вперед.

Добавим столбец, в котором содержатся вчерашние значения index1D.

1) В пункте **Column** главного меню выберите **Shift Column Down**.

2) Щелкните мышью на столбец index1D.

3) Укажите, что величина сдвига равна 1, а метка новой колонки – ind-1.

Наводим порядок (1).

1) Удалите столбец BD100, так он больше не понадобится. Для этого выделив его, укажите **Column/Delete Column**.

2) Таким же образом удалите столбцы price1 и price2.

3) Пометьте столбец ind-1 как Input.

4) Переместите BD+4 в правый крайний столбец. Для этого выберите «Move Column» в разделе «Column» главного меню и укажите BD+4 и ind-1 на запрос об выборе колонок..

Входные данные будут выглядеть следующим образом:

day	Annote
index1D	Input
index2D	Input
line	Input
streng	Input
utilD	Input

transpD	Input
BD100D	Input
price1D	Input
price2D	Input
Bdavg2	Input
ind-1	Input
BD+4	Pattern

Результат проделанных шагов запишите в файл price3.dat.

Представление временных рядов.

К сожалению, нельзя просто задать столбец с данными за понедельник, вторник и т. д. Нейронная сеть «видит» только один факт, представленный текущей строкой. Поэтому, если требуется подать на вход данные ста последних торговых дней, то в каждой строке должно быть сто полей для задания этих значений. И неважно, что в предыдущей строке входного файла они уже есть. Придется указать их еще раз. При этом следует иметь ввиду, что при увеличении количества входов увеличивается сама сеть, что ухудшает ее способность к обучению. В то же время, если нейронная сеть будет «видеть» тренд, то она сделает лучший прогноз. Выбор оптимальных данных и их количества не имеет однозначного решения и на практике производится в процессе настройки сети.

Так как в нашем примере был найден цикл, равный четырем дням, то для информирования нейронной сети о тренде будем использовать текущие значения входного параметра и значения четырехдневной давности. Текущие значения уже есть, следовательно, надо создать столбцы с данными за четыре дня назад для всех входных столбцов, кроме ind-1.

1) Выберите **Repeat** в разделе **Operate** главного меню.

2) Укажите число повторений команды – 10.

3) Выберите **Shift Column Down** в разделе **Column** главного меню.

4) На запрос **Select Column** укажите Index1D.

5) Величина сдвига равна 4, а метка новой колонки – ind1-4.

6) В последующих запросах укажите метки: ind2-4, line-4, stren-4, util-4, tran-4, BD-4, pric1-4, pric2-4, Bdavg-4. Вы получили десять новых столбцов с данными.

Наводим порядок (2).

В десяти столбцах справа, вследствие сдвига на четыре строки вниз, первые пять строк идентичны друг другу. Кроме того, ранее мы делали сдвиг на одну строку вниз в столбцах BD100, price1D, price2D. Следовательно, надо удалить первые пять строк, которые не содержат корректные данные. Для этого можно воспользоваться командой **Repeat** или пять раз проделать следую-

щую операцию: выделив строку, выберите **Delete Row** в разделе **Row** главного меню и подтвердите удаление.

Для формирования BD+4 был осуществлен сдвиг вверх на 4 строки. Поэтому нужно удалить четыре нижние строки.

В результате всех этих операций осталось 176 фактов из 185. То есть, на подготовке данных в нашем примере потеряно около 5% входных данных. Эти потери следует учитывать при планировании количества данных при создании нейронной сети.

Результат всех проделанных операций запишите в файл price4.dat.

Создание файлов для BrainMaker.

Перед тем, как создавать входные файлы для BrainMaker, перемешаем строки таблицы, чтобы они располагались в случайном порядке. Это нужно для лучшего обучения нейронной сети. Таблицу с перемешанными строками сохранять не следует.

1) Выберите **Shiffle Rows** в разделе **Row** главного меню.

2) Сделайте это еще раз.

3) В разделе **Label** главного меню выберите **All Unmarked Columns/Inputs**.

4) Выберите **Create BrainMaker Files** в разделе **File** главного меню и подтвердите имена файлов, предлагаемые по умолчанию.

5) Выйдите из NetMaker. Подготовка данных завершена.

6.8.3 Обучение, тестирование и опрос нейронной сети

Обучение нейронной сети.

1) Запустите BrainMaker.

2) В пункте **File** главного меню выберите **Read Network** и загрузите файл price4.def.

3) Прогнозирование на финансовых рынках является достаточно сложной задачей. Поэтому будем считать точность прогноза 80% вполне приемлемой. В пункте **Parameters** главного меню выберите **Training Control Flow**. Измените значения Training tolerance и Testing tolerance на 0,2. Остальные параметры остаются без изменения. Нажмите **Ok**.

4) В пункте **Operatre** главного меню выберите **Train Network** В окне будет отображена информация вида рис. 6.16.

Горизонтальные столбики (термометры) показывают входные и выходные значения. Их использование облегчает наблюдение за быстро изменяющимися значениями. Обратите внимание, что BD+4 соответствует два термометра. Out показывает значение, которое генерирует нейронная сеть, а Ptn – то значение, которое должно быть предсказано (Pattern – образец).

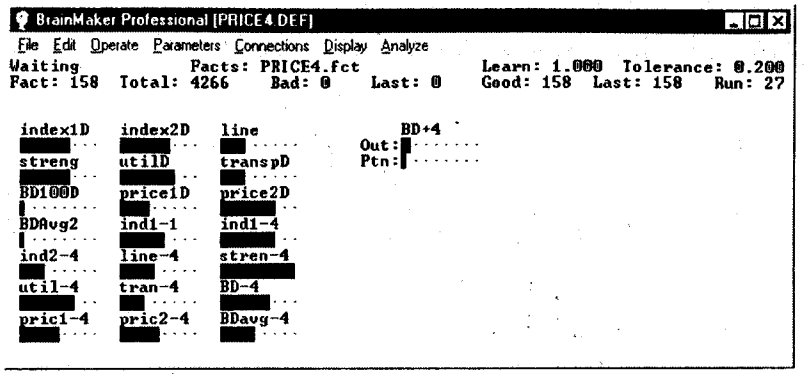


Рис. 6.16. Окно программы

Для остановки обучения выберите **Train Network** еще раз.

Для отображения данных в виде чисел, символов и др. войдите в пункт **Display** главного меню, а затем в подменю выберите **Edit Network Display**.

Для обучения сети используется 158 фактов из 176. 10% или 18 фактов зарезервированы для тестирования при формировании файлов в NetMaker.

Первая строка с данными под главным меню (status line) показывает, что в настоящее время делает BrainMaker и какие используются файлы.

Learning Rate используется для более точной настройки обучения нейронной сети и сейчас рассматриваться не будет.

Tolerance является максимальной допустимой величиной ошибки во время обучения. Если tolerance равна 0, то это означает, что выходной результат (output) , выдаваемый нейронной сетью, должен абсолютно точно совпадать с образцом для обучения (pattern). Для подавляющего большинства случаев это нереалистичное требование, особенно для финансовых приложений, которые считаются наиболее сложными. При tolerance равной 0,1 выход нейронной сети будет рассматриваться как корректный, если он отличается не более чем на 10% от заданного значения. Если же output будет вне 10% интервала, то BrainMaker внесет изменения в структуру нейронной сети таким образом, чтобы в следующей попытке получить более корректный результат. Этот процесс будет продолжаться до тех пор, пока значение ошибки не снизится до установленного параметром tolerance предела.

В строке statistics line, следующей за status line, представлена информация о ходе обучения или тестирования сети:

Fact – порядковый номер примера из обучающей или тестирующей последовательности (номер строки в таблице с исходными данными), который обрабатывается в данное время;

Total – общее количество примеров, обработанных на данный момент;

Bad – количество примеров, для которых получен некорректный результат в текущем проходе входной последовательности;

Last – количество примеров, для которых получен некорректный результат в предыдущем проходе;

Good – количество примеров, для которых получен корректный результат в текущем проходе;

Last – количество примеров, для которых получен корректный результат в предыдущем проходе;

Run – общее количество проходов входной последовательности, включая текущий.

Наблюдать за процессом обучения нейронной сети можно, включив режим **Network Progress Display** в разделе **Display** главного меню (рис. 6.17).

Гистограмма на верхнем графике отображает распределение ошибок за один проход обучающей выборки. На горизонтальной оси показана величина ошибки, а на вертикальной оси – количество ошибок. По мере обучения вертикальные столбики смещаются влево, отражая процесс уменьшения величины ошибок при обработке примеров. Обучение заканчивается, когда для всех примеров уровень ошибки будет меньше допустимого уровня – 0,2.

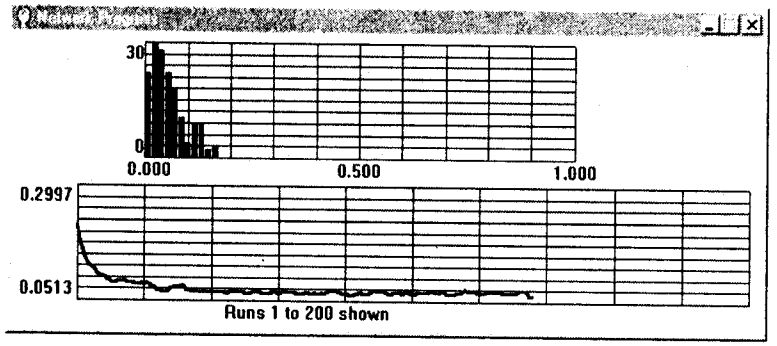


Рис. 6.17. Динамика процесса обучения

По графику можно определить достижимый уровень ограничения ошибок нейронной сети. Например, в рассматриваемом примере этот уровень можно довести до 0,125.

Нижний график отражает динамику изменения общей ошибки нейронной сети RMS Error (Root Mean Squared) по формуле:

$$RMSerror = \frac{\sqrt{\sum\left((O-P)^2\right)}}{N},$$

где O – output, P – pattern, N – количество фактов.

На горизонтальной оси показан номер прохода обучающей выборки, на вертикальной оси – величина ошибки.

Сохраните обученную нейронную сеть в файле price4.net.

Тестирование нейронной сети.

1) В пункте **Operate** главного меню выберите **Test Network**.

2) Нейронная сеть обработает примеры и сравнит полученные результаты (output) с образцами (pattern). Ошибки будут вычислены, но корректировка структуры нейронной сети производится при этом не будет.

3) В строке statistics line отображается результат тестирования – Good: 15 Bad: 3. Если осуществить настройку нейронной сети, то этот результат можно улучшить. Рассмотрим, как используется сеть для получения прогноза.

Получение результатов.

Запустите NetMaker и загрузите файл price5.dat. Этот файл содержит пять последних удаленных строк из исходного файла price1.dat и одну строку с новыми данными. С этим файлом нужно проделать те же манипуляции, что и с price1.dat, за исключением создания столбца с известными будущими значениями BD+4. Анализ циклов и корреляции данных делать также не нужно. После этого сохраните данный файл под именем price5.in, **выбрав Create Running Fact File** в разделе **File** главного меню.

1) Войдите в BrainMaker и загрузите нейронную сеть из файла price4.net.

2) Выберите **Select Fact File** в разделе **File** главного меню. Укажите **Read Running Facts** и прочитайте price5.in.

3) Выберите **Write Facts to File** в разделе **File** главного меню. Предложенные по умолчанию значения менять не нужно, поэтому сразу ответьте **Open File**.

4) Измените способ отображения output с thermometers на number. Для этого войдите в подраздел **Edit Network Display** пункта **Display** главного меню.

5) Выберете опцию **Run Trained Network** в разделе **Operate**.

6) На экране в поле Out появится результат прогноза.

7) Выйдите из BrainMaker и посмотрите файл price4.out. Число после графического символа является номером примера. В

следующей строке представлены входные данные. Третья строка содержит результат – прогноз изменения BD100 на 4 дня вперед.

6.8.4. Некоторые выводы

Процесс разработки нейросетевых приложений состоит из определенной последовательности операций, которая не зависит от выбранного пакета программ.

Большое значение имеет исходная формулировка проблемы.

Самая трудоемкая для аналитика часть работы включает сбор и предварительную обработку данных. По некоторым оценкам это может занимать до 90% времени.

Многие операции по подготовке данных удобнее выполнять не в NetMaker, а, например, в Excel. Более того, для интенсивного использования нейронной сети требуются специальные программы, которые выбирают из поступающего потока нужные данные, преобразуют их и передают на вход сети. Решение этой проблемы весьма актуально для хранилищ данных.

Нейронные сети позволяют строить комплексную систему прогноза как на базе технических, так и макроэкономических показателей, находя компромисс между сторонниками двух подходов.

6.9. Сжатие информации

Проиллюстрируем применение многослойных персептронов на примере решения задачи сжатия информации для представления исходных данных в компактной и удобной форме. Результаты этапа сжатия зачастую определяют успех решения задачи распознавания в целом.

Популярный метод сжатия информации был предложен в 1987 г. G. Cottrell, P. Munro, D. Zipser. Рассмотрим трехслойный персептрон, у которого число нейронов входного и выходного слоев одинаково, а число нейронов скрытого слоя значительно меньше. Предположим, что в результате обучения персептрон может воспроизводить на выходе тот же самый вектор X, который подается на входной слой. Такой персептрон автоматически осуществляет сжатие (компрессию) информации: на нейронах скрытого слоя возникает представление каждого вектора, которое значительно короче, чем длина вектора, подаваемого на вход. Предположим, что некоторый набор векторов нужно передавать по линии связи, предварительно сжимая информацию и, тем самым, уменьшая число каналов, необходимых для ее передачи. Поместим на одном конце линии входной и скрытый слои персептрона, а результат работы нейронов скрытого слоя (короткие векторы) будем передавать по линии связи. На другом конце линии поместим копию скрытого слоя и выходной слой, тогда переданный короткий вектор с нейронов скрытого слоя перейдет на нейроны выходного слоя, где и будет воспроизведен входной вектор (декомпрессия).

6.10. Компактное представление информации репликативными нейронными сетями

В процессе анализа или обработки удобно предположить, что анализируемые данные являются векторами многомерного, например, евклидова пространства, где они располагаются в соответствии с некоторой функцией распределения. В случае же, если многомерные данные не порождаются непосредственно пространственно-временной природой информации, то это также не мешает использовать такую модель для их представления. Примерами могут служить результаты измерений, полученные в ходе физических, биологических и других экспериментов, результаты медицинской диагностики, телеметрическая информация. При этом размерность пространства признаков (координат) n может достигать десятков тысяч.

Естественные координаты. Опыт показывает, что объекты, как правило, не заполняют все n-мерное пространство, а располагаются на некоторой поверхности, имеющей невысокую размерность (десятки, сотни). Это означает, что существуют обобщенные (естественные) признаки, образованные из комбинаций исходных признаков и наиболее точно характеризующие исходные объекты.

Доказана единственность системы естественных координат, а также известно, что эти координаты обладают рядом важных свойств, например, признаки объектов в естественных координатах являются попарно независимыми. Что делает естественные координаты весьма удобными для широкого класса вероятностных распределений.

Рассмотрим некий генератор данных в n-мерном пространстве. Предположим, что генерируемые им данные заполняют не все пространство, а лишь некоторое многообразие размерности m, где $m << n$. Многообразие, на котором расположены данные, можно представлять как сложную поверхность, которая в окрестности каждой своей точки похожа на поверхность m-мерной сферы. Утверждается, что на многообразии существует такая функция распределения, которая сколь угодно близка к исходной функции распределения генератора в смысле средних значений. Это позволя-

ет вместо исходных данных, представляемых большим числом признаков, рассматривать генератор данных в пространстве существенного меньшего числа измерений.

Естественные координаты вводятся следующим образом. Рассмотрим взаимно-однозначное непрерывное отображение m-мерного многообразия в m-мерный единичный куб. Это означает, что каждый вектор X, лежащий на m-мерном многообразии, представляется вектором с m координатами, причем каждая координата является числом, равномерно распределенным между нулем и единицей.

Естественные координаты зависят только от внутренней, заранее определенной, вероятностной структуры многообразия данных: равные объемы внутри единичного куба соответствуют множествам с равной вероятностью на многообразии данных, хотя их геометрические размеры могут значительно различаться. Естественные координаты могут отражать сложную вероятностную структуру многообразия данных.

Естественные координаты – это единственная координатная система из независимых компонент, которая обеспечивает оптимальное кодирование информации, с учетом вероятностной структуры генератора данных.

Репликативные нейронные сети. Для нахождения естественных координат можно использовать репликативные (копирующие) нейронные сети. Репликативная нейронная сеть представляет собой многослойный персептрон с тремя скрытыми слоями, число нейронов входного и выходного слоев которого одинаково. Первый и третий скрытые слои состоят из нейронов с сигмоидной активационной функцией. Размеры этих слоев подбираются в процессе обучения сети. Интересно отметить, что репликативные нейронные сети дают компактное и эффективное представление произвольных наборов векторов, имеющих сложное вероятностное распределение в пространстве, за счет того, что средний скрытый слой имеет меньше нейронов, чем входной и выходной слои.

Для вектора длины n, подаваемого на входной слой, строится его отображение f в единичный куб. Для обученной сети это отображение реализует представление исходного вектора в системе естественных координат u, которое воспроизводится в среднем скрытом слое. Дальнейшее прохождение информации от среднего скрытого слоя до выходного слоя дает отображение g (обратное f) из единичного куба в исходное n-мерное пространство с заданной функцией распределения $F(x)$.

Средний скрытый слой состоит из m нейронов, где m – предполагаемое число естественных координат. Передаточная функция нейронов среднего скрытого слоя имеет вид наклонной или ступенчатой функции.

Цель обучения репликативной нейронной сети состоит в том, чтобы вектор, воспроизводимый выходным слоем сети, совпадал с вектором, поданным на входной слой. Передаточная функция нейронов выходного слоя выбирается линейной. Обучение проводится на обучающей выборке, полученной с помощью генератора данных с функцией распределения $F(x)$. Утверждается, что обученная репликативная нейронная сеть строит в среднем скрытом слое представление исходных векторов в естественных координатах. Таким образом, входной вектор длины n передается на средний слой и там представляется естественными координатами в m-мерном единичном кубе ($m << n$). Дальнейшая передача информации по сети от среднего скрытого слоя к выходному дает обратное отображение: вектор в естественных координатах переходит в n-мерный вектор, расположенный близко к входному.

Один из подходов к обучению репликативной нейронной сети основан на том, что известно, каким должен быть выходной сигнал у нейронов среднего скрытого слоя – это должны быть естественные координаты. Таким образом, можно использовать такой метод обучения, благодаря которому нейроны среднего скрытого слоя более активно производят выходные сигналы, равномерно и плотно заполняющие внутреннюю часть m-мерного единичного куба, а также обладают свойствами естественных координат.

Удаление шума. Репликативная сеть способна удалять аддитивный шум, присутствующий в исходных данных. Предположим, что вектор данных состоит из двух слагаемых: информационной части вектора и шумового случайного компонента, выбираемого в каждой точке многообразия данных в соответствии с условной плотностью распределения.

Утверждается, что репликативная нейронная сеть приводит шумовой компонент к среднему значению, и результат, получаемый на выходе сети, является суммой информативной части вектора (входного вектора) и среднего значения шума в данной точке многообразия. Если среднее значение равно нулю, то выходной слой воспроизводит входной вектор, удаляя случайный шум.

Определение размерности. До сих пор полагалось, что размерность m заранее известна. Однако такое бывает достаточно редко. Тем не менее эту размерность можно оценить. Рассмотрим большое количество данных, произведенных генератором, и упорядочим их по возрастанию евклидова расстояния до некоторой

фиксированной точки из этого же набора. Такое упорядочение данных позволяет оценить размерность многообразия вблизи фиксированной точки.

Возьмем первые k векторов из упорядоченного набора, рассмотрим гауссовый ковариационный эллипсоид и определим количество не слишком коротких его осей. Эту величину назовем локальной размерностью и нарисуем график ее зависимости от числа k. Обычно этот график линейно возрастает при увеличении k, но при некотором его значении наклон графика резко уменьшается, образуя «колено». Соответствующую величину k будем рассматривать как аппроксимацию размерности в окрестности выбранной точки. Повторяя описанную процедуру определения локальной размерности для других точек, находим оценку размерности многообразия данных m, как наибольшее из значений локальной размерности. Найденное значение m используется в качестве размерности системы естественных координат. Если оно меньше, чем реальная размерность многообразия данных, то полученные естественные координаты будут образовывать решетку, заполняющую все пространство. С другой стороны, если значение m больше, чем реальная размерность, то координатные объемы в естественной системе координат будут «сплюснуты» в такие множества, у которых, по крайней мере, один характерный размер много меньше других. Таким образом, чтобы определить правильное значение размерности, нужно найти компромисс между полнотой заполнения пространства (когда величина размерности выбирается слишком малой) и сильной деформацией координатной решетки в естественных координатах (когда величина размерности выбрана слишком большой).

Знание размерности многообразия данных и выбор правильного значения m может помочь избежать неэффективного использования системы естественных координат. Аналогичное замечание справедливо и для «испорченных» шумом векторов для выбора нужного число нейронов в среднем скрытом слое, после чего сеть может быть натренирована для очистки шума.

Использование репликативных нейронных сетей. Решающим фактором в вопросе практического применения репликативных сетей является то, что большинство генераторов данных сильно структурированы и могут быть смоделированы посредством многообразия данных с относительно малой размерностью m. Такой подход заметно упрощает решение многих задач распознавания образов, управления, сжатия информации, поскольку вместо исходных векторов с большим числом признаков, могут использоваться естественные координаты с малым числом компонентов.

6.11. Кратко о других задачах

Ниже приведена информация аннотационного плана о ряде других задач, с успехом решаемых с помощью искусственных нейронных сетей. Несмотря на все различие в их содержательных трактовках, методика применения нейросетевого подхода аналогична той, которая была рассмотрена.

6.11.1. Обработка видеоизображений

Одной из наиболее сложных и актуальных проблем обработки видеоизображений, представленных последовательностью оцифрованных кадров, является проблема выделения и распознавания движущихся объектов в условиях действия различного рода помех и возмущений. Для этого должны быть решены задачи выделения изображений движущихся объектов на сложном зашумленном фоне, фильтрации помех, скоростной фильтрации, отделения объектов от фона, оценки скорости каждого объекта, его идентификации и сопровождения. Системы обработки видеоизображений, построенные с применением нейросетевых методов, представляют собой, как правило, программно-аппаратные комплексы на персональных компьютерах, позволяющие работать с данными телевизионной системы в реальном времени (не менее 25 кадров/с, 320x200 пикселей).

Выделение изображений движущихся объектов осуществляется путем построения оценки поля скоростей с помощью многослойной локально-связной нейронной сети. Размерность сети для изображения 320x200 пикселей составляет несколько миллионов нейронов и примерно вчетверо больше синапсов.

Распознавание выделенных силуэтов производится на самоорганизующейся нейронной сети, предварительно обученной на изображениях объектов рассматриваемых классов. Обеспечивается инвариантность к произвольному движению фона, устойчивость к зашумлению до 10%. Вероятность правильного распознавания не менее 90%.

6.11.2. Обработка статических изображений

Не менее сложными являются задачи выделения и распознавания объектов на статическом тоновом изображении. В частности, подобные задачи возникают при автоматической обработке спутниковых изображений земной поверхности. Для их решения разработан и реализован на персональных компьютерах ряд автоматизированных систем анализа изображений земной поверхно-

сти. Системы в автоматическом режиме обеспечивает выделение на обрабатываемых изображениях объектов заданных классов: дорожной сети, кварталов с характерной застройкой, аэродромов и стоящих на них самолетов. Нейросетевые принципы, заложенные в их основу, обеспечивают инвариантность к яркостным характеристикам выделяемых и распознаваемых объектов, а также позволяют проводить обучение и адаптацию систем.

6.11.3. Обнаружение и классификация объектов по звуковым и гидроакустическим сигналам

Использование нейросетевых технологий для анализа акустического излучения демонстрируют системы обнаружения и распознавания летательных аппаратов по звуку, а также надводных и подводных объектов по гидроакустическим сигналам. Сигналы от объектов подвергаются предобработке и в оцифрованном виде подаются на вход предварительно обученной нейронной сети для распознавания. Исследования систем показали высокую вероятность правильного распознавания (до 90%).

6.11.4. Задачи комбинаторной оптимизации

Высокая степень распараллеленности обработки информации позволяет успешно применять нейросетевые технологии для решения задач комбинаторной оптимизации, к которым, в первую очередь следует отметить задачи транспортно-ориентированной оптимизации (например, задача коммивояжера и ее модификации) и задачи распределения ресурсов (задача о назначениях, задача целераспределения и другие).

Решение таких задач традиционными методами математического программирования, большинство из которых изначально ориентировано на вычислительную технику с последовательной архитектурой, сопряжено с большими временными затратами, неприемлемыми для многих приложений. При соответствующей аппаратной поддержке нейросетевые методы позволяют значительно повысить оперативность решения данного класса задач, сохраняя высокую точность результата.

6.11.5. Медицинская диагностика

В настоящее время разработано достаточно много нейродиагностических комплексов, позволяющих сократить необходимое время диагностики различных заболеваний, а также снизить потребность в квалифицированном медицинском персонале.

Например, компанией «НейроПроект» создана система диагностики слуха у грудных детей. Общепринятая методика диагностики состоит в том, что в процессе обследования регистрируются отклики мозга в ответ на звуковые раздражители. Для достаточно уверенной диагностики слуха ребенка опытному эксперту-аудиологу необходимо провести около 2000 тестов, что занимает около часа. Нейронная сеть способна с той же достоверностью определить уровень слуха уже по 200 наблюдениям в течение всего нескольких минут, причем без участия квалифицированного персонала.

6.11.6. Распознавание речи

Распознавание речи – одно из наиболее популярных применений нейронных сетей. Достаточно квалифицированный пользователь может создать свою нейросетевую систему распознавания речи, используя, к примеру, двухкаскадную иерархическую нейронную сеть, где первый уровень осуществляет грубое распознавание слов, относя их к одному из классов, а второй уровень точно классифицирует слово внутри каждого из классов.

6.11.7. Обнаружение фальсификаций

В США введена в действие система обнаружения мошенничеств в области здравоохранения. Подсчитано, что потери бюджета от такого рода фальсификаций составляют около 730 млн. долларов в год. Создание нейросетевой специализированной системы заняло у фирмы ITC более года и обошлось в 2,5 млн. долларов. Тестирование показало, что нейронная сеть позволяет обнаруживать 38% мошеннических случаев, в то время как существующая экспертная система – только 14%. Для настройки системы были использованы также методы нечеткой логики и генетической оптимизации.

6.11.8. Анализ потребительского рынка

Фирма IBM Consulting создала нейросетевую систему, прогнозирующую свойства потребительского рынка. Одним из основных маркетинговых механизмов является распространение купонов, дающих право покупки определенного товара со скидкой. Так как затраты на рассылку купонов довольно велики, решающим фактором является эффективность рассылки, т. е. доля клиентов, воспользовавшихся скидкой.

Для повышения эффективности купонной системы важно было провести предварительную сегментацию рынка, а затем адресовать клиентам каждого сегмента именно те купоны, которыми они с большей вероятностью воспользуются. В терминах анализа данных здесь требовалось решить задачу кластеризации, что и было успешно проделано с помощью сетей Кохонена. На втором этапе для потребителей каждого из кластеров подбирались подходящие коммерческие предложения, а затем строился прогноз объема продаж для каждого сегмента.

Другой вариант решения этой же задачи избрала компания GoalAssist Corp., исследуя механизм предоставления поощрительных призов за приобретенные покупки. Обычные методы прогнозирования откликов потребителей оказались недостаточно точны. Так, спрос на одни призы оказался слишком велик, в то время как другие призы остались невостребованными. Для повышения точности прогнозирования было решено использовать нейронные сети. Первая нейронная сеть, построенная с помощью пакета NeuroShell Classifier, решала задачу классификации откликов. Вторая самоорганизующуюся сеть, реализованная в нейропакете NeuroShell Predictor, осуществляла количественное прогнозирование. Средняя ошибка прогноза составила всего около 4%.

Фирма Neural Innovation Ltd. использует при работе с маркетинговыми компаниями конкретную стратегию прямой рассылки. Вначале рассылается 25% от общего числа предложений и собирается информация об откликах потребителей. Затем эта информация обрабатывается нейронной сетью, которая осуществляет поиск оптимального сегмента потребительского рынка для данного товара. Затем остальные 75% предложений рассылаются в указанный сегмент. При этом эффективность рассылки существенно возрастает.

6.11.9. Проектирование и оптимизация сетей связи

Одна из важнейших задач в области телекоммуникаций – нахождение оптимального пути пересылки трафика между узлами – может быть успешно решена с помощью нейронных сетей. В данном случае важны две особенности: во-первых, решение должно быть адаптивным, то есть учитывать текущее состояние сети связи и наличие сбойных участков, а во-вторых, найти оптимальное решение нужно очень быстро, в реальном времени. Нейронные сети прекрасно приспособлены для решения такого рода задач.

Кроме управления маршрутизацией потоков, нейронные сети используются и для проектирования новых телекоммуникационных сетей. При этом удается получить очень эффективные решения.

6.11.10. Прогнозирование изменений котировок

Компания Alela Corp. занимается прогнозированием изменения биржевых индексов. Для предсказания знаков изменения индексов применяется нейронная сеть, использующая радиальные базисные функции. На сайте компании можно бесплатно получить прогнозы изменения индексов Dow Jones, S&P500 и Merval, а также убедиться, что доля верных предсказаний на основе нейронных сетей составляет не менее 80%. Создатели сайта предлагают всем желающим использовать эти прогнозы в качестве дополнительных индикаторов.

6.11.11. Управление ценами и производством

Руководители предприятий часто недооценивают потери от неоптимального планирования производства. Так как спрос и условия реализации зависят от времени, сезона, курсов валют и многих других факторов, то и объем производства следует гибко варьировать с целью оптимального использования ресурсов. Уже существуют удачные примеры нейросетевых систем планирования, которые применяются совместно со стандартными методами исследования операций, динамического программирования, а также с методами нечеткой логики.

Крупное английское издательство, выпускающее газеты, приобрело у фирмы Neural Innovation Ltd. систему планирования цен и затрат, основанную на нейронной сети с использованием генетических алгоритмов. На основе исторических данных система обнаруживает сложные зависимости между затратами на рекламу, объемом продаж, ценой газеты, ценами конкурентов, днем недели, сезоном и т. д. После этого возможен подбор оптимальной стратегии с точки зрения максимизации объема продаж или прибыли.

6.11.12. Исследование факторов спроса

Для увеличения прибыльности бизнеса в условиях конкуренции компании необходимо поддерживать обратную связь с потребителями. В частности, для этого проводятся опросы потребителей, позволяющие выяснить, какие факторы являются для них решающими при покупке данного товара или услуги, почему в некоторых случаях предпочтение отдается конкурентам, и какие улуч-

шения товара потребитель хотел бы увидеть в будущем. Анализ результатов такого опроса – достаточно сложная задача, так как здесь участвует большое количество связанных между собой параметров. Нейронные сети идеально подходят для решения этой задачи. Существующие нейросетевые методы позволяют выявлять сложные зависимости между факторами спроса, прогнозировать поведение потребителей при изменении маркетинговой политики, находить наиболее значимые факторы и оптимальные стратегии рекламы, а также очерчивать сегмент потребителей, наиболее перспективный для данного товара.

Примеры успешного использования нейросетевых технологий для построения эффективной маркетинговой политики:

- маркетинговая кампания «Tango Orange Man», проведенная английским производителем безалкогольных напитков, фирмой Britvic Soft Drinks совместно с фирмой Neural Technologies;
- исследование предпочтений потребителей различных сортов пива в зависимости от их возраста, дохода, семейного положения и других параметров, проведенное фирмой Neural Technologies.

6.11.13. Прогнозирование потребления энергии

Фирма ZSolutions разработала нейросетевую систему анализа данных об энергопотреблении, фиксирующую измерение потребляемой энергии для каждого клиента через каждые 15 минут, с учетом того, что некоторые из измерений ошибочны. Система позволяет выявлять ошибочные измерения, а также прогнозировать потребление энергии в каждый момент времени. Знание точного прогноза позволило энергетической компании использовать гибкую тарифную политику и увеличить свою прибыль.

6.11.14. Оценка недвижимости

Стоимость дома или квартиры зависит от большого числа факторов, таких как общая площадь, удаленность от центра, экологическая обстановка, престижность, тип дома. Так как вид этой зависимости неизвестен, то стандартные методы анализа, как правило, неэффективны. Обычно эта задача решается экспертами, работающими в агентстве по недвижимости. Недостатком такого подхода является субъективность оценщиков, а также возможные разногласия между ними. Существуют успешные примеры решения задачи объективной оценки с помощью нейронной сети. В частности, фирма Attrasoft приводит пример оценки стоимости домов в Бостоне с учетом 13 параметров.

6.11.15. Анализ страховых исков

Фирмой Neural Innovation Ltd. создана нейросетевая система Claim Fraud Analyser, позволяющая мгновенно выявлять подозрительные страховые иски, относящиеся к поврежденным автомобилям. На входы системы подаются такие параметры, как возраст и опыт водителя, стоимость автомобиля, наличие подобных происшествий в прошлом. В результате обработки определяется вероятность того, что данный иск связан с мошенничеством.

Такая система позволяет не только сэкономить за счет выявления фальсификаций, но и улучшить отношения с клиентами за счет более быстрого удовлетворения честных исков.

6.12. Краткое обобщение

Подводя итог, можно заметить, что нейросетевая методология находит все новые применения в практике принятия решений и управлении сложными организационно-техническими системами. Нейронные сети идеально приспособлены для обнаружения сложных зависимостей в отсутствие априорных знаний об исследуемой системе или процессе. Нейронные сети также можно использовать везде, где обычно применялись линейные методы и алгоритмы, и производилось оценивание при помощи статистических методов анализа, таких как регрессионный, кластерный, дискриминантный анализ, временные ряды.

Приведенные выше примеры показывают, что технологии нейронных сетей, нечеткой логики и генетических алгоритмов применимы практически в любой области. В некоторых задачах, таких как прогнозирование котировок или распознавание образов, нейронные сети стали уже привычным инструментом. Нет сомнений, что повсеместное проникновение новых технологий и в другие области – только вопрос времени.

ПРИЛОЖЕНИЯ

П.1. Основные парадигмы нейронных сетей

Таблица П.1.1

Наименования парадигм

Английское название	Русское название
Adaptive Neuro-Fuzzy Inference System (ANFIS)	Нечеткая нейронная сеть
ART–1, ART–1 Network	Искусственный резонанс – 1, 2
BAM Network	Двунаправленная ассоциативная память
Boltzmann Machine	Больцмановское обучение
Back Propagation	Обратное распространение
Cognitron	Когнитрон
Counter Propagation	Сеть встречного распространения
Delta-Bar-Delta (DBD) Network	Delta-Bar-Delta сеть
Elman and Jordan Network	Частично-рекуррентные сети Элмана и Жордана
Extended DBD Network	Расширенная DBD сеть
Feed-Forward MAXNET	Сеть поиска максимума с прямыми связями
Gaussian Classifier	Гауссовый классификатор
Genetic Algorithm	Генетический алгоритм
Hamming Network	Сеть Хэмминга
Hopfield Network	Сеть Хопфилда
Instar Network	Входная звезда
Kohonen Network	Сеть Кохонена
MAXNET	Сеть поиска максимума
Neocognitron	Неокогнитрон
Outstar Network	Выходная звезда
Radial Basis Function Network	Сеть радиального основания
Simulated Annealing	Обучение обжигом
Single Layer Perceptron	Однослойный персептрон

П.1.1. Искусственный резонанс – 1. ART–1 Network (Adaptive Resonance Theory Network – 1).

1) Название.

Adaptive Resonance Theory Network (сеть теории адаптивного резонанса).

Другие названия.

Сеть адаптивной резонансной теории – 1 (АРТ–1).

Carpenter–Grossberg Classifier (классификатор Карпентера–Гроссберга).

2) Авторы и история создания.

Разработана Карпентером и Гроссбергом в 1986 г.

3) Модель.

Сеть ART–1 (рис. П.1.1) обучается без учителя. Она реализует алгоритм кластеризации, подобный алгоритму «последовательного лидера» (Sequential Leader Clustering Algorithm). В соответствии с алгоритмом первый входной сигнал считается эталоном первого кластера. Следующий входной сигнал сравнивается с эталоном первого кластера. Говорят, что входной сигнал «следует за лидером» и принадлежит первому кластеру, если расстояние до эталона первого кластера меньше порога. В противном случае второй входной сигнал становится эталоном второго кластера. Процесс повторяется для всех следующих входных сигналов. Таким образом, число кластеров растет с течением времени и зависит как от значения порога, так и от меры сходства, использующейся для сравнения входных сигналов и эталонов классов.

Основная часть сети ART–1 схожа с сетью Хэмминга, которая дополнена полносвязной сетью MAXNET. С помощью после-

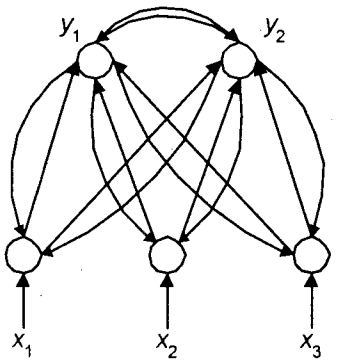


Рис. П.1.1. Основные компоненты классификатора Карпентера–Гроссберга

довательных связей высчитывается соответствие входных сигналов и образцов кластеров. Максимальное значение соответствия усиливается с помощью взаимного латерального торможения выходных нейронов. Сеть ART–1 отличается от сети Хэмминга обратными связями от выходных нейронов к входным. Кроме того, имеется возможность отключать выходной нейрон с максимальным значением соответствия и проводить пороговое тестирование соответствия входного сигнала и образцов кластеров, как того требует алгоритм «последовательного лидера».

Алгоритм функционирования сети.

ШАГ 1. Инициализация сети:

$$v_{ij}(0) = 1, \quad w_{ij}(0) = \frac{1}{N+1}, \quad i = 1,\ldots, N, \quad j = 1,\ldots, M, \quad 0 \le r \le L,$$

где $w_{ij}(t)$ – синаптический вес связи от i-го нейрона первого слоя к j-му нейрону второго слоя в момент времени t (bottom up); $v_{ij}(t)$ – синаптический вес связи от i-го нейрона второго слоя к j-му нейрону первого слоя в момент времени t (top down); r – значение порога, L – константа со значением в диапазоне от 1 до 2.

Веса $v_{ij}(t)$ и $w_{ij}(t)$ определяют эталон, соответствующий нейрону j. Порог r показывает, насколько должен входной сигнал совпадать с одним их запомненных эталонов, чтобы они считались похожими. Близкое к единице значение порога требует почти полного совпадения. При малых значениях порога даже сильно различающиеся входной сигнал и эталон считаются принадлежащими одному кластеру.

ШАГ 2. Предъявление сети нового бинарного входного сигнала первому слою нейронов аналогично тому, как это делается в сети Хэмминга.

ШАГ 3. Вычисление значений соответствия:

$$y_j = \sum_{i=1}^{N} w_{ij}(t)\, x_i, \qquad j = 1,\ldots, M.$$

Значения соответствия вычисляются параллельно для всех эталонов, запомненных в сети, аналогично сети Хэмминга.

ШАГ 4. Выбор образца с наибольшим соответствием:

$$y_j = \max_j (y_j).$$

Эта операция выполняется с помощью латерального торможения аналогично сети MAXNET.

ШАГ 5. Вычисление отношения скалярного произведения входного сигнала и эталона с наибольшим значением соответствия к числу единичных бит входного сигнала. Значение отношения сравнивается с порогом, введенном на шаге 1.

$$\|X\| = \sum_{i=1}^{N} x_i, \quad \|V \cdot X\| = \sum_{j=1}^{M} \sum_{i=1}^{N} v_{ij} \cdot x_{ij},$$

$$\textit{если} \quad \frac{\|V \cdot X\|}{\|X\|} > r, \quad \textit{то переход к шагу 7},$$

иначе – к шагу 6.

Если значение отношения больше порога, то входной сигнал считается похожим на эталон с наибольшим значением соответствия. В этом случае эталон модифицируется путем выполнения операции «Логического И» на входной сигнал.

Если значение отношения меньше порога, то считается, что входной сигнал отличается от всех эталонов и рассматривается как новый эталон. В сеть вводится нейрон, соответствующий новому эталону, и рассчитываются значения синаптических весов.

ШАГ 6. Исключение нейрона с наибольшим значением соответствия:

Выход нейрона с наибольшим значением соответствия временно устанавливается равным нулю и более не принимает участие в шаге 4.

ШАГ 7. Адаптация нейрона с наибольшим значением соответствия:

$$v_{ij}(t+1) = v_{ij}(t) \cdot x_i, \quad w_{ij}(t+1) = \frac{v_{ij}(t) \cdot x_i}{0{,}5 + \sum_{i=1}^{N} v_{ij}(t) \cdot x_i}.$$

ШАГ 8. Включение всех исключенных на шаге 6 эталонов. Возвращение к шагу 2.

Характеристики сети.

Тип входных сигналов – бинарные. Размерность входных и выходных сигналов ограничена при программной реализации только возможностями вычислительной системы, на которой моделируется нейронная сеть, при аппаратной реализации – технологическими возможностями. Емкость сети совпадает с числом нейронов второго слоя и может увеличиваться в процессе функционирования сети.

4) Области применения.

Распознавание образов, кластерный анализ.

5) Недостатки.

Неограниченное увеличение числа нейронов в процессе функционирования сети.

При наличии шума возникают значительные проблемы, связанные с неконтролируемым ростом числа эталонов.

6) Преимущества.

Обучение без учителя.

7) Модификации.

Модель ART–2 с непрерывными значениями входных сигналов.

П.1.2. Двунаправленная ассоциативная память. Bi-Directional Associative Memory (BAM).

1) Название.

Bi-Directional Associative Memory BAM (двунаправленная ассоциативная память ДАП).

2) Авторы и история создания.

Совокупность моделей двунаправленной ассоциативной памяти разработана под руководством Б. Коско (Университет Южной Калифорнии). Большая часть публикаций, посвященных этим моделям, датирована второй половиной 1980-х годов.

3) Модель.

Двунаправленная ассоциативная память является гетероассоциативной. Входной вектор поступает на один набор нейронов, а соответствующий выходной вектор вырабатывается на другом наборе нейронов. Входные образы ассоциируются с выходными.

Для сравнения: сеть Хопфилда является автоассоциативной. Входной образ может быть восстановлен или исправлен сетью, но не может быть ассоциирован с другим образом. В сети Хопфилда используется одноуровневая структура ассоциативной памяти, в которой выходной вектор появляется на выходе тех же нейронов, на которые поступает входной вектор.

Двунаправленная ассоциативная память, как и сеть Хопфилда, способна к обобщению, вырабатывая правильные выходные сигналы, несмотря на искаженные входы.

На рис. 2.10 приведена схема двунаправленной ассоциативной памяти. В результате обработки входного вектора $\boldsymbol{X}$ матрицей $\boldsymbol{W}$ весов (нейронами первого слоя) вырабатывается выходной вектор $\boldsymbol{Y} = F(\boldsymbol{XW})$, который затем обрабатывается транспонированной матрицей $\boldsymbol{W}^T$ весов (нейронами второго слоя) и преобразуется в новый входной вектор $\boldsymbol{X} = F(\boldsymbol{YW}^T)$. Этот процесс повторяется до достижения сетью стабильного состояния. В качестве функции активации используется экспоненциальная сигмоида.

Формула для вычисления значений синаптических весов:

$$\boldsymbol{W} = \sum_j \boldsymbol{X}_j^T \boldsymbol{Y}_j ,$$

где $\boldsymbol{X}_j$ и $\boldsymbol{Y}_j$ – входные и выходные сигналы обучающей выборки. Весовая матрица вычисляется как сумма произведений всех векторных пар обучающей выборки.

Системы с обратной связью имеют тенденцию к колебаниям. Они могут переходить от состояния к состоянию, никогда не достигая стабильности. Доказано, что ДАП безусловно стабильна при любых значениях весов сети.

4) Области применения.

Ассоциативная память, распознавание образов.

5) Недостатки.

Емкость ДАП жестко ограничена. Если n – количество нейронов в меньшем слое, то число векторов, которые могут быть запомнены в сети, не превышает:

$$L = \frac{n}{2\log_2 n}.$$

Так, если n = 1024, то сеть способна запомнить не более 25 образов. Кроме того, ДАП обладает некоторой непредсказуемостью в процессе функционирования. Возможны ложные ответы.

6) Преимущества.

По сравнению с автоассоциативной памятью (например, с сетью Хопфилда), двунаправленная ассоциативная память дает возможность строить ассоциации между векторами $\boldsymbol{X}$ и $\boldsymbol{Y}$, которые в общем случае имеют разные размерности. За счет таких возможностей гетероассоциативная память используется для более широкого класса приложений, чем автоассоциативная память.

Процесс формирования синаптических весов является простым и достаточно быстрым. Сеть быстро сходится в процессе функционирования.

Двунаправленная ассоциативная память – простая сеть, которая может быть реализована в виде отдельной СБИС или оптоэлектронным способом.

7) Модификации.

Предложена негомогенная двунаправленная ассоциативная память, в которой пороговые значения подбираются индивидуально для каждого нейрона (в исходной модели ДАП все нейроны имеют нулевые пороговые значения). Емкость негомогенной сети выше, чем исходной модели.

Сигналы в сети могут быть как дискретными, так и непрерывными. Для обоих случаев доказана стабильность сети.

Предложены модели двунаправленной ассоциативной памяти с обучением без учителя (адаптивная ДАП).

Введение латеральных связей внутри слоя дает возможность реализовать конкурирующую тип ДАП. Веса связей формируют матрицу с положительными значениями элементов главной диагонали и отрицательными значениями остальных элементов. Теорема Кохонена–Гроссберга доказывает, что такая сеть является стабильной.

П.1.3. Машина Больцмана (Boltzmann Machine)

1) Название.

Boltzmann Machine (машина Больцмана).

Другое название.

Больцмановское обучение.

2) Авторы и история создания.

Машина Больцмана была предложена и исследовалась во второй половине 1980-х годов.

3) Модель.

Алгоритм больцмановского обучения:

ШАГ 1. Определить переменную T, представляющую искусственную температуру.

ШАГ 2. Предъявить сети множество входов и вычислить выходы и целевую функцию.

ШАГ 3. Дать случайное изменение весу и пересчитать выход сети и изменение целевой функции в соответствии со сделанным изменением веса.

ШАГ 4. Если целевая функция улучшилась (уменьшилась), то сохранить изменение веса.

Если изменение веса приводит к увеличению целевой функции, то вероятность сохранения этого изменения вычисляется с помощью распределения Больцмана:

$$P(c) = \exp\left(-\frac{c}{kT}\right),$$

где $P(c)$ – вероятность изменения параметра c в целевой функции; k – константа, аналогичная константе Больцмана, выбираемая в зависимости от задачи.

Выбирается случайное число r из равномерного распределения от нуля до единицы. Если $P(c)$ больше, чем r, то изменение сохраняется, в противном случае величина веса возвращается к предыдущему значению.

Эта процедура дает возможность системе делать случайный шаг в направлении, «портящем» целевую функцию, позволяя ей тем самым выходить из локальных минимумов.

Шаги 3 и 4 повторяются для каждого из весов сети, постепенно уменьшая температуру T, пока не будет достигнуто допустимо низкое значение целевой функции. В этот момент предъявляется другой входной вектор и процесс обучения повторяется. Сеть обучается на всех векторах обучающего множества, пока целевая функция не станет допустимой для всех из них.

Скорость уменьшения температуры должна быть обратно пропорциональна логарифму времени. При этом сеть сходится к глобальному минимуму.

4) Области применения.

Распознавание образов, классификация.

5) Недостатки.

Медленный алгоритм обучения.

6) Преимущества.

Алгоритм дает возможность сети выбираться из локальных минимумов адаптивного рельефа.

7) Модификации.

Случайные изменения могут проводиться не только для отдельных весов, но и для всех нейронов слоя в многослойных сетях или даже для всех нейронов сети одновременно. Эти модификации алгоритма дают возможность сократить общее число итераций обучения.

П.1.4. Обратное распространение (Neural Network with Back Propagation Training Algorithm)

1) Название.

Нейронная сеть с обучением по методу обратного распространения ошибки.

Другие названия.

Backprop.

Back Propagation (Neural) Network.

Сеть обратного распространения (ошибки).

Multi-Layer Perceptron with Back Propagation Training Algorithm (многослойный персептрон с обучением по методу обратного распространения ошибки).

2) Авторы и история создания.

Многослойные персептроны были предложены и исследованы в 1960-х годах Розенблаттом, Минским, Пейпертом и др. Лишь в середине 1980-х годов был предложен эффективный алгоритм обучения многослойных персептронов, основанный на вычислении градиента функции ошибки и названный обратным распространением ошибки.

3) Модель.

Алгоритм обратного распространения – это итеративный градиентный алгоритм, который используется с целью минимизации среднеквадратического отклонения текущего выхода многослойного персептрона и требуемого выхода.

Он используется для обучения многослойных нейронных сетей с последовательными связями. Нейроны в таких сетях делятся на группы с общим входным сигналом – слои. На каждый нейрон первого слоя подаются все элементы внешнего входного сигнала. Все выходы нейронов q-го слоя подаются на каждый нейрон слоя $(q+1)$. Нейроны выполняют взвешенное суммирование входных сигналов. К сумме элементов входных сигналов, помноженных на соответствующие синаптические веса, прибавляется смещение нейрона. Над результатом суммирования выполняется нелинейное преобразование – функция активации (передаточная функция). Значение функции активации есть выход нейрона.

Характеристики сети.

Тип входных сигналов – целые или действительные. Тип выходных сигналов – действительные из интервала, заданного передаточной функцией нейронов. Тип передаточной функции – сигмоидальная. В нейронных сетях применяются несколько вариантов сигмоидальных передаточных функций.

Функция Ферми (экспоненциальная сигмоида):

$$f(s) = \frac{1}{1 + e^{-2as}},$$

где s – выход сумматора нейрона, a – некоторый параметр.

Рациональная сигмоида:

$$f(s) = \frac{1}{|s| + a}.$$

Гиперболический тангенс:

$$f(s) = \text{th}\left(\frac{s}{a}\right) = \frac{e^{\frac{s}{a}} - e^{-\frac{s}{a}}}{e^{\frac{s}{a}} + e^{-\frac{s}{a}}}.$$

Перечисленные функции относятся к однопараметрическим. Значение функции зависит от аргумента и одного параметра. Используются также и многопараметрические передаточные функции, например:

$$f(s) = p_1 \frac{s}{|s| + p_2} + p_3.$$

Сигмоидальные функции являются монотонно возрастающими и имеют отличные от нуля производные на всей области определения. Эти характеристики обеспечивают правильное функционирование и обучение сети.

Наиболее эффективной передаточной функцией является рациональная сигмоида. Для вычисления гиперболического тангенса требуются большие вычислительные затраты.

Функционирование многослойной нейронной сети осуществляется в соответствии с формулами:

$$s_{i_q} = \sum_{i_{q-1}}^{N_{q-1}} w_{i_q i_{q-1}} w_{i_{q-1}} - b_{i_q}, \quad i_q = 1, \ldots, N_q, \quad m = 1, \ldots, L,$$

$$y_{i_q} = f(s_{i_q}), \quad i_q = 1, \ldots, N_q, \quad m = 1, \ldots, L,$$

где s – выход сумматора, w – вес связи, y – выход нейрона, b – смещение, i – номер нейрона, N – число нейронов в слое, m – номер слоя, L – число слоев, f – функция активации.

Обучение сети разбивается на следующие этапы:

ШАГ 1. Инициализация сети.

Весовым коэффициентам и смещениям сети присваиваются малые случайные значения из установленных диапазонов.

ШАГ 2. Определение элемента обучающей выборки:

(<Текущий вход>, <требуемый выход>). Текущие входы $(x_1, \ldots, x_N)$, должны различаться для всех элементов обучающей выборки. При использовании многослойного персептрона в качестве классификатора требуемый выходной сигнал $(d_1, \ldots, d_N)$ состоит из нулей за исключением одного единичного элемента, соответствующего классу, к которому принадлежит текущий входной сигнал.

ШАГ 3. Вычисление текущего выходного сигнала.

Текущий выходной сигнал определяется в соответствии с традиционной схемой функционирования многослойной нейронной сети.

ШАГ 4. Настройка синаптических весов.

Для настройки весовых коэффициентов используется рекурсивный алгоритм, который сначала применяется к выходным нейронам сети, а затем проходит сеть в обратном направлении до первого слоя. Синаптические веса настраиваются в соответствии с формулой:

$$w_{ij} = (t+1) = w_{ij}(t) + \eta \delta_j x_i,$$

где $w_{ij}(t)$ – вес от нейрона i или от элемента входного сигнала i к нейрону j в момент времени t; x_i – выход нейрона i или i-й элемент входного сигнала, η – коэффициент скорости обучения; δ_j – значение ошибки для нейрона j.

Если нейрон с номером j принадлежит последнему слою, то:

$$\delta_j = y_j(1 - y_j)(d_j - y_j),$$

где d_j, y_j – соответственно требуемый и текущий выход j-го нейрона.

Если j-й нейрон принадлежит одному из слоев с первого по предпоследний, то:

$$\delta_j = x_j(1 - x_j)\sum_k \delta_k y_{jk},$$

где j-й нейрон принадлежит предыдущему слою, а индекс k пробегает все нейроны последующего слоя.

Смещения нейронов b настраиваются аналогичным образом.

4) Области применения.

Распознавание образов, классификация, прогнозирование, синтез речи, контроль, адаптивное управление, построение экспертных систем.

5) Недостатки.

Многокритериальная задача оптимизации в методе обратного распространения рассматривается как набор однокритериальных – на каждой итерации происходят изменения значений параметров сети, улучшающие работу лишь с одним примером обучающей выборки. Такой подход существенно уменьшает скорость обучения.

Классический метод обратного распространения относится к алгоритмам с линейной сходимостью. Для увеличения скорости сходимости необходимо использовать матрицы вторых производных функции ошибки.

6) Преимущества.

Обратное распространение – первый эффективный алгоритм обучения многослойных нейронных сетей. Один из самых популярных алгоритмов обучения, с его помощью решены и решаются многочисленные практические задачи.

7) Модификации.

Модификации алгоритма обратного распространения связаны с использованием различных функций ошибки, различных процедур определения направления и величины шага:

функции ошибки:

- интегральные функции ошибки по всей совокупности обучающих примеров;
- функции ошибки целых и дробных степеней;

процедуры определения величины шага на каждой итерации:

- дихотомия;

- инерционные соотношения, например,

$$w_{ij} = (t+1) = w_{ij}(t) + \eta\,\delta_j x_i + \alpha\left(w_{ij}(t) - w_{ij}(t-1)\right),$$

где α – некоторое положительное число, меньше единицы;

- отжиг;

процедуры определения направления шага:

- с использованием матрицы производных второго порядка (метод Ньютона и др.);
- с использованием направлений на нескольких шагах (партан метод и др.).

П.1.5. Сеть встречного распространения (Counter Propagation Network)

1) Название.

Counter Propagation Network (сеть встречного распространения).

Другое название.

Hecht-Nielsen Neurocomputer.

2) Авторы и история создания.

Разработаны Р. Хехт-Нильсенем (University of California, San Diego) в 1986 г.

3) Модель.

В сети встречного распространения объединены две нейропарадигмы: самоорганизующаяся карта Кохонена и звезда Гроссберга. Считается, что в мозге именно соединения модулей различной специализации позволяют выполнять требуемые вычисления.

В процессе обучения сети встречного распространения входные векторы ассоциируются с соответствующими выходными векторами. Эти векторы могут быть двоичными или непрерывными. После обучения сеть формирует выходные сигналы, соответствующие входным сигналам. Обобщающая способность сети дает возможность получать правильный выход, когда входной вектор неполон или искажен.

В режиме обучения на вход сети подается входной сигнал и веса корректируются, чтобы сеть выдавала требуемый выходной сигнал.

Слой Кохонена функционирует по правилу «победитель получает все». Для данного входного вектора только один нейрон этого слоя выдает логическую единицу, все остальные – нули. Выход каждого нейрона Кохонена является просто суммой взвешенных элементов входных сигналов:

Выходы нейронов слоя Гроссберга также являются взвешенными суммами выходов нейронов слоя Кохонена. Однако каждый нейрон слоя Гроссберга выдает величину веса, который связывает этот нейрон с единственным нейроном Кохонена, чей выход отличен от нуля.

На этапе предварительной обработки входных сигналов осуществляется нормализация входных векторов.

На этапе обучения слой Кохонена классифицирует входные векторы в группы схожих. Это достигается с помощью такой подстройки весов слоя Кохонена, что близкие входные векторы активируют один и тот же нейрон данного слоя. Какой именно нейрон будет активироваться при предъявлении конкретного входного сигнала, заранее трудно предсказать, так как слой Кохонена обучается без учителя.

Затем задачей слоя Гроссберга является получение требуемых выходов. Обучение слоя Гроссберга – это обучение с учителем. Выходы нейронов вычисляются как при обычном функционировании. Далее каждый вес корректируется лишь в случае, если он соединен с нейроном Кохонена, имеющим ненулевой выход. Величина коррекции веса пропорциональна разности между весом и требуемым выходом нейрона Гроссберга.

В режиме функционирования сети предъявляется входной сигнал и формируется выходной сигнал.

В полной модели сети встречного распространения имеется возможность получать выходные сигналы по входным и наоборот. Этим двум действиям соответствуют прямое и обратное распространение сигналов.

4) Области применения.

Распознавание и восстановление образов (ассоциативная память), сжатие данных (с потерями), статистический анализ.

5) Недостатки.

Сеть не дает возможности строить точные аппроксимации. В этом она значительно уступает сетям с обратным распространением ошибки.

Слабая теоретическая проработка модификаций этой сети.

6) Преимущества.

Сеть встречного распространения проста. Она дает возможность извлекать статистические характеристики из множеств входных сигналов. Кохоненом показано, что для полностью обученной сети вероятность того, что случайно выбранный входной вектор (в соответствии с функцией плотности вероятности входного множества) будет ближайшим к любому заданному весовому вектору, равна $1/l$, l – число нейронов Кохонена.

Сеть быстро обучается. Время ее обучения по сравнению с обратным распространением может быть в 100 раз меньше.

По своим возможностям строить отображения сеть встречного распространения значительно превосходит однослойные персептроны.

Сеть полезна для приложений, в которых требуется быстрая начальная аппроксимация.

Сеть дает возможность строить функцию и обратную к ней, что находит применение при решении практических задач.

7) Модификации.

Сети встречного распространения могут различаться способами определения начальных значений синаптических весов. Так, кроме случайных значений из заданного диапазона, могут быть использованы значения в соответствии с известным *методом выпуклой комбинации*.

Для повышения эффективности обучения применяется добавление шума к входным векторам.

Еще один метод повышения эффективности обучения – наделение каждого нейрона «чувством справедливости». Если нейрон становится победителем чаще, чем $1/l$, то ему временно увеличивают порог, предоставляя, тем самым, возможность обучаться и другим нейронам.

Кроме *метода аккредитации*, при котором для каждого входного вектора активируется лишь один нейрон Кохонена, может быть использован *метод интерполяции*, при использовании которого группа нейронов Кохонена, имеющих наибольшие выходы, может передавать свои выходные сигналы в слой Гроссберга. Этот метод повышает точность отображений, реализуемых сетью.

П.1.6. Delta Bar Delta сеть

1) Название.

Delta Bar Delta Network (DBD).

2) История создания.

Алгоритм Delta Bar Delta создан Якобсом с целью ускорения обучения сети за счет использования эвристического подхода. Алгоритм использует предыдущие значения градиента функции, на основе которых осуществляются изменения в пространстве весов с помощью ряда эвристических правил.

Опыт показывает, что размерности пространства весов могут значительно различаться с точки зрения общей поверхности ошибки. Якобс предложил ряд эвристик, суть которых в том, что каждый вес должен изменяться в соответствии с индивидуальной скоро-

стью обучения, так как размер шага обучения для одного веса не всегда подходит в качестве шага обучения для всех весов. Более того, этот размер может со временем изменяться.

Первые эвристики по изменению индивидуальных шагов обучения нейронов были введены Кестеном. Он предложил, что если последовательные изменения веса имеют противоположные знаки, то значит данный вес осциллирует, и, следовательно, скорость обучения должна быть уменьшена. Позднее Садирис ввел следующее правило: если серия последовательных изменений веса имеет одинаковые знаки, то скорость обучения должна быть увеличена.

3) Модель.

Изменение веса на последующем шаге:

$$w(t+1) = w(t) + a(t)g(t).$$

Расчет среднего изменения градиента на шаге t:

$$g_{av}(t) = (1 - convex)g(t) + g(t-1)convex.$$

Расчет изменения скорости обучения на шаге t:

$$da(t) = \begin{cases} k_1, & если \quad g_{av}(t-1)g_{av}(t) > 0, \\ k_2 a(t), & если \quad g_{av}(t-1)g_{av}(t) < 0, \\ 0, & если \quad g_{av}(t-1)g_{av}(t) = 0, \end{cases}$$

где $e(t)$ – ошибка обучения на шаге t; $w(t)$ – значение веса; $dw(t)$ – изменение веса; $a(t)$ – коэффициент скорости обучения; $da(t)$ – изменение скорости обучения; $g(t)$ – градиент изменения веса; $g_{av}(t)$ – взвешенное среднее изменение градиента; $convex$ – фактор выпуклости весов; k_1 – константа увеличения скорости обучения; k_2 – константа уменьшения скорости обучения.

Линейное увеличение изменения скорости позволяет избежать быстрого роста скорости. Геометрическое уменьшение позволяет проследить, что скорость обучения всегда положительная. Более того, скорость может уменьшаться более быстро на сильно нелинейных участках.

4) Области применения.

Распознавание образов, классификация.

5) Недостатки.

Стандартный алгоритм DBD не использует эвристики, основанные на моменте.

Даже небольшое линейное увеличение коэффициента может привести к значительному росту скорости обучения, что вызовет скачки в пространстве весов.

Геометрическое уменьшение коэффициента иногда оказывается недостаточно быстрым.

6) Преимущества.

Парадигма Delta Bar Delta является попыткой ускорить процесс конвергенции алгоритма обратного распространения за счет использования дополнительной информации об изменении параметров и весов во время обучения.

П.1.7. Расширенная DBD сеть (Extended Delta Bar Delta Network)

1) Название.

Extended Delta Bar Delta Network (EDBD).

2) История создания.

Сеть Extended Delta Bar Delta разработана Minai и Williams. Ими был использован параметр *момента связи* (momentum), представляющий собой некоторое число, пропорциональное предыдущему изменению веса. Они использовали значения момента для ускорения обучения с помощью ряда эвристических правил.

3) Модель.

Изменение веса на последующем шаге:

$$dw(t+1) = a(t)g(t) + m(t)dw(t),$$

$$w(t+1) = w(t) + dw(t+1).$$

Расчет среднего изменения градиента на шаге t:

$$g_{av}(t) = (1 - convex)g(t) + g(t-1)convex.$$

Расчет изменения скорости обучения на шаге t:

$$da(t) = \begin{cases} k_{a1} \exp\left(-y_a \left| g_{av}(t) \right|\right), & если \quad g_{av}(t-1)g_{av}(t) > 0, \\ -k_{a2} a(t), & если \quad g_{av}(t-1)g_{av}(t) < 0, \\ 0, & если \quad g_{av}(t-1)g_{av}(t) = 0. \end{cases}$$

Расчет изменения момента на шаге t:

$$dm(t) = \begin{cases} k_{m1} \exp\left(-y_m \left| g_{av}(t) \right|\right), & если \quad g_{av}(t-1)g_{av}(t) > 0, \\ -k_{m2} m(t), & если \quad g_{av}(t-1)g_{av}(t) < 0, \\ 0, & если \quad g_{av}(t-1)g_{av}(t) = 0, \end{cases}$$

где $e(t)$ – ошибка обучения на шаге t; $w(t)$ – значение веса; $dw(t)$ – изменение веса; $a(t)$ – коэффициент скорости обучения; $da(t)$ – изменение скорости обучения; $g(t)$ – градиент изменения веса; $g_{av}(t)$– взвешенное среднее изменение градиента; $convex$ – фактор выпуклости весов; $m(t)$ – значение момента; $dm(t)$ – изменение значения момента; k_{a1} – фактор масштабирования скорости обучения; k_{m1} – фактор масштабирования момента; y_a – экспоненциальный фактор скорости обучения; y_m – экспоненциальный фактор момен-

та; k_{a2} – фактор масштабирования скорости обучения; k_{m2} – фактор масштабирования момента; a_{max} – верхняя граница скорости обучения; m_{max} – верхняя граница момента.

Коэффициенты скорости обучения и скорости изменения момента имеют различные константы, контролирующие их увеличение и уменьшение.

Для всех связей принимаются следующие ограничения:

$$a(t) < a_{max},$$
$$m(t) < m_{max}.$$

Если текущая ошибка превышает минимальную предыдущую ошибку с учетом максимального отклонения, то все связи восстанавливаются для наилучшего варианта и коэффициенты обучения и момента уменьшаются.

4) Области применения.

Распознавание образов, классификация.

П.1.8. Сеть поиска максимума с прямыми связями (Feed-Forward MAXNET)

1) Название.

Feed-Forward MAXNET (сеть поиска максимума с прямыми связями).

Другое название.

Сеть поиска максимума, основанная на двоичном дереве и нейросетевых компараторах.

2) Авторы и история создания.

Сеть предложена в качестве дополнения к сети Хэмминга.

3) Модель.

Многослойная сеть с прямыми связями. Входные сигналы попарно сравниваются друг с другом. Наибольший сигнал в каждой паре передается на следующий слой для дальнейших сравнений. На выходе сети лишь один сигнал имеет ненулевое значение. Он соответствует максимальному сигналу на входе сети.

Основу сети составляет показанный на рис. П.1.2 нейросетевой компаратор, который выполняет сравнение двух аналоговых сигналов (x_1 и x_2). На выходе z – значение максимального сигнала (x_1 или x_2). Выходы y_1 и y_2 показывают, какой именно входной сигнал имеет максимальное значение. На рисунке проставлены значения синаптических весов. Смещения всех нейронов сети – нулевые. Нейроны, помеченные темным цветом, имеют жесткие пороговые передаточные функции, передаточные функции у остальных нейронов – линейные с насыщением.

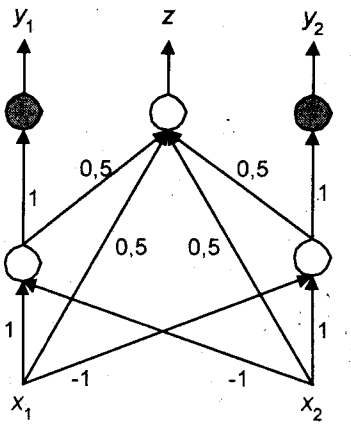


Рис. П.1.2. Нейросетевой компаратор

На рис. П.1.3 показан пример построения нейронной сети для поиска максимума с прямыми связями, которая дает возможность определять максимальный сигнал из восьми входных сигналов. Сеть состоит из нескольких компараторов и дополнительных нейронов и синаптических связей. Синаптические веса компараторов такие же, как и на рис. П.1.2. Веса других связей – единичные. Зачерненные нейроны имеют жесткие пороговые функции с нулевым смещением, активационные (передаточные) функции белых нейронов – линейные с насыщением. Активационные функции нейронов последнего слоя представляют собой пороговые функции со смещением 2,5.

Характеристики сети.

Типы входных сигналов – аналоговые (целые или действительные числа), тип выходных сигналов – целые, размерности входных и выходных сигналов совпадают и ограничены только возможностями реализуемой вычислительной системы. Число слоев в сети приблизительно равно $\log_2(n)$, где n – размерность входного сигнала.

4) Области применения.

Совместно с сетью Хэмминга, в составе нейросетевых систем распознавания образов.

5) Недостатки.

Число слоев сети растет с увеличением размерности входного сигнала.

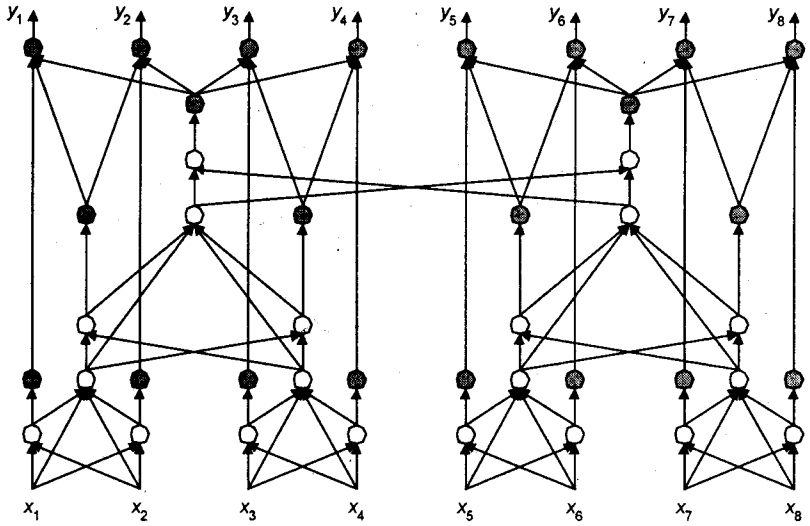


Рис. П.1.3. Сеть поиска максимума с прямыми связями

6) Преимущества.

В отличие от сети MAXNET циклического функционирования, в рассматриваемой модели заранее известен объем вычислений, который требуется для получения решения.

7) Модификации.

Для решения задачи выделения сигнала с максимальным значением из некоторого множества сигналов наиболее часто используется сеть MAXNET циклического функционирования.

П.1.9. Гауссов классификатор (Neural Gaussian Classifier)

1) Название.

Нейросетевой гауссов классификатор.

Другое название.

Gaussian Classifier Implemented Using the Perceptron Structure (гауссов классификатор, реализованный на персептроне).

2) Авторы и история создания.

Модель предложена Липпманом в 1987 г.

3) Модель.

Персептрон может быть использован для реализации гауссова классификатора по максимуму вероятности (Gaussian Maximum Likelihood Classifier).

В классическом алгоритме обучения персептрона не используются предположения относительно распределений примеров обучающих выборок, а рассматривается функция ошибки. Этот алгоритм работает более устойчиво, если входные сигналы формируются в результате нелинейных процессов и распределены несимметрично и не по гауссову закону.

В основе построения гауссова классификатора лежат предположения о распределениях входных сигналов. Считается, что эти распределения известны и соответствует закону Гаусса.

Формулы для расчета параметров нейросетевого гауссова классификатора определяются следующим образом. Пусть M_{Ai} и s_{Ai}^2 – среднее значение и отклонение (математическое ожидание и дисперсия) входного сигнала x_i, когда входной сигнал принадлежит классу A, M_{Bi} и s_{Bi}^2 – среднее значение и отклонение входного сигнала x_i, когда входной сигнал принадлежит классу B, и $s_{Ai}^2 = s_{Bi}^2 = s_i^2$. Тогда значения вероятности, используемые классификатором по максимуму вероятности пропорциональны следующим величинам:

$$L_A = -\sum_{i=1}^{N}\frac{(x_i - M_{Ai})^2}{s_i^2} = -\sum_{i=1}^{N}\frac{x_i^2}{s_i^2} + 2\sum_{i=1}^{N}\frac{M_{Ai}x_i}{s_i^2} - \sum_{i=1}^{N}\frac{M_{Ai}^2}{s_i^2},$$

$$L_B = -\sum_{i=1}^{N}\frac{(x_i - M_{Bi})^2}{s_i^2} = -\sum_{i=1}^{N}\frac{x_i^2}{s_i^2} + 2\sum_{i=1}^{N}\frac{M_{Bi}x_i}{s_i^2} - \sum_{i=1}^{N}\frac{M_{Bi}^2}{s_i^2}.$$

Классификатор по максимуму вероятности должен вычислять L_A L_B и выбирать класс с наибольшей вероятностью. Первые слагаемые в формулах идентичны. Поэтому их можно опустить. Вторые слагаемые могут быть вычислены путем умножения входных сигналов на синаптические веса. Третьи слагаемые являются константами, значения которых можно присвоить смещению нейрона.

Значения синаптических весов и смещения:

$$w_i = \frac{2(M_{Ai} - M_{Bi})^2}{s_i^2}, \qquad b = \sum_{i=1}^{N}\frac{M_{Ai}^2 - M_{Bi}^2}{s_i^2}.$$

Характеристики сети.

Тип входных сигналов – бинарные или аналоговые (действительные). Размерности входа и выхода ограничены при программной реализации только возможностями вычислительной системы, на которой моделируется нейронная сеть, при аппаратной реализации – технологическими возможностями.

4) Области применения.

Распознавание образов, классификация.

5) Недостатки.

Примитивные разделяющие поверхности (гиперплоскости) дают возможность решать лишь самые простые задачи распознавания. Считаются априорно известными распределения входных сигналов, соответствующие разным классам.

6) Преимущества.

Программные или аппаратные реализации модели очень просты. Простой и быстрый алгоритм формирования синаптических весов и смещений.

7) Модификации.

Адаптивный гауссов классификатор.

П.1.10. Генетический алгоритм (Genetic Algorithm)

1) Название.

Neural Network with Genetic Training Algorithm (нейронная сеть с генетическим алгоритмом обучения).

2) Авторы и история создания.

Впервые идея использования генетических алгоритмов для обучения (machine learning) была предложена Дж. Голландом (J. Holland) в 1970-е годы. Во второй половине 1980-х годов к этой идее вернулись в связи с обучением нейронных сетей.

3) Модель.

Использование механизмов генетической эволюции для обучения нейронных сетей кажется естественным, поскольку модели нейронных сетей разрабатываются по аналогии с мозгом и реализуют некоторые его особенности, появившиеся в результате биологической эволюции.

Основные компоненты генетических алгоритмов: стратегии репродукций, мутаций и отбор «индивидуальных» нейронных сетей (по аналогии с отбором индивидуальных особей).

Первая проблема построения алгоритмов генетической эволюции – это кодировка информации, содержащейся в модели нейронной сети. Коды называют хромосомами. Для фиксированной топологии (архитектуры) нейронной сети эта информация полностью содержится в значениях синаптических весов ($\boldsymbol{W}$) и смещений ($\boldsymbol{B}$). Набор ($\boldsymbol{W}$, $\boldsymbol{B}$) рассматривается как хромосома. Возможны более сложные способы кодирования информации.

Для реализации концепции отбора необходимо ввести способ сопоставления различных хромосом в соответствии с их возможностями решения поставленных задач. Для сетей с последовательными связями это может быть евклидово расстояние.

В отличие от большинства других алгоритмов обучения, для генетических алгоритмов формируется не один, а несколько наборов начальных значений параметров, которые называются популяцией хромосом. Популяция обрабатывается с помощью алгоритмов репродукции, изменчивости (мутаций), генетической композиции. Эти алгоритмы напоминают биологические процессы. Наиболее важные среди них: случайные мутации данных в индивидуальных хромосомах, переходы (кроссовер), рекомбинация генетического материала, содержащегося в индивидуальных родительских хромосомах (аналогично гетеросексуальной репродукции), миграция генов.

Генетический алгоритм работает следующим образом. Инициализируется популяция и все хромосомы сравниваются в соответствии с выбранной функцией оценки. Далее (возможно многократно) выполняется процедура репродукции популяции хромосом. Родители выбираются случайным образом в соответствии со значениями оценки (вероятность того, что данная хромосома станет родителем, пропорциональна полученной оценке). Репродукция происходит индивидуально для одного родителя путем мутации хромосомы либо для двух родителей путем кроссовера генов. Получившиеся потомки оцениваются в соответствии с заданной функцией и помещаются в популяцию.

В результате использования описанных операций на каждой стадии эволюции получаются популяции со все более совершенными вариантами.

4) Области применения.

Распознавание образов, классификация, прогнозирование.

5) Недостатки.

Сложны для понимания и программной реализации.

6) Преимущества.

Генетические алгоритмы особенно эффективны в поиске глобальных минимумов адаптивных рельефов, так как ими исследуются большие области допустимых значений параметров нейронных сетей.

Достаточно высокая скорость обучения, хотя и меньшая, чем скорость сходимости градиентных алгоритмов.

Генетические алгоритмы дают возможность оперировать дискретными значениями параметров нейронных сетей, что упрощает аппаратную реализацию нейронных сетей и приводит к сокращению общего времени обучения.

7) Модификации.

В рамках генетического подхода в последнее время разработаны многочисленные алгоритмы обучения нейронных сетей,

различающиеся способами представления данных нейронной сети в хромосомах, стратегиями репродукции, мутаций, отбора.

П.1.11. Сеть Хэмминга (Hamming Net)

1) Название.

Hamming Net сеть Хэмминга).

Другие названия.

Нейросетевая модель ассоциативной памяти, основанная на вычислении расстояния Хэмминга.

Классификатор по минимуму расстояния Хэмминга.

2) Авторы и история создания.

Нейросетевые модели, основанные на вычислениях расстояния Хэмминга в задачах передачи двоичных сигналов фиксированной длины, введены Липпманом в 1987 г.

3) Модель.

Расстояние Хэмминга между двумя бинарными векторами одинаковой длины – это число несовпадающих бит в этих векторах. Нейронная сеть, которая реализует параллельное вычисление расстояний Хэмминга от входного вектора до нескольких векторов–образцов, носит название сети Хэмминга.

Характеристики сети.

Тип входных сигналов – бинарные векторы; тип выходных сигналов – целые числа. Размерности входа и выхода ограничены при программной реализации только возможностями вычислительной системы, на которой моделируется нейронная сеть, при аппаратной реализации – технологическими возможностями. Размерности входных и выходных сигналов могут не совпадать. Тип передаточной функции – линейная с насыщением. Число синапсов в сети равно произведению числа нейронов в сети на размерность входного сигнала.

Формирование синаптических весов и смещений сети:

$$w_{ij} = x_i^j, \quad b_j = 0, \quad i = 1...N, \quad j = 1...M.$$

Функционирование сети:

$$y_j = f\left(\sum_{i=1}^{N} w_{ij} x_i - b_j\right), \quad j = 1...M,$$

где w_{ij} – i-й синаптический вес j-го нейрона; x_i – i-й элемент входного сигнала сети; y_j – выход j-го нейрона; b_j – смещение j-го нейрона; N – размерность входного сигнала; M – количество нейронов в сети; x_i^j – i-ый элемент j-го вектора–образца.

Наиболее часто рассматривается модель, синаптические веса и смещения в которой вычисляются по формулам:

$$w_{ij} = \frac{x_i^j}{2}, \quad b_j = \frac{N}{2}, \quad i = 1...N, \quad j = 1...M.$$

4) Области применения.

Распознавание образов, классификация, ассоциативная память, надежная передача сигналов в условиях помех.

5) Недостатки.

Сеть способна правильно распознавать (классифицировать) только слабо зашумленные входные сигналы. Возможность использования только бинарных входных сигналов существенно ограничивает область применения.

6) Преимущества.

Сеть работает предельно просто и быстро. Выходной сигнал (решение задачи) формируется в результате прохода сигналов всего лишь через один слой нейронов. Для сравнения: в многослойных сетях сигнал проходит через несколько слоев; в сетях циклического функционирования сигнал многократно проходит через нейроны сети, причем число итераций, необходимое для получения решения, бывает заранее не известно.

В модели использован один из самых простых алгоритмов формирования синаптических весов и смещений сети.

В отличие от сети Хопфилда, емкость сети Хэмминга не зависит от размерности входного сигнала, она в точности равна количеству нейронов. Сеть Хопфилда с входным сигналом размерностью 100 может запомнить 10 образцов, при этом у нее будет 10000 синапсов. У сети Хэмминга с такой же емкостью будет всего лишь 1000 синапсов.

7) Модификации.

Сеть Хэмминга может быть дополнена сетью MAXNET, которая определяет, какой из нейронов сети Хэмминга имеет выход с максимальным значением.

П.1.12. Сеть Хопфилда (Hopfield Network)

1) Название.

Hopfield Network (сеть Хопфилда).

Другие названия.

Ассоциативная память, адресуемая по содержанию.

Модель Хопфилда.

2) Авторы и история создания.

Сеть разработана Хопфилдом в 1982 г. С нее началось возрождение интереса к нейронным сетям. С тех пор были предложены ее многочисленные модификации, направленные на увеличение емкости, улучшение сходимости.

3) Модель.

Одна из первых предложенных сетей Хопфилда используется как автоассоциативная память. Исходными данными для расчета значений синаптических весов сети являются векторы–образцы классов. Выход каждого из нейронов подается на входы всех остальных нейронов.

Формирование синаптических весов сети:

$$w_{ij} = \begin{cases} \sum_{k=1}^{M} x_i^k x_j^k, & \text{если } i \neq j, \\ 0, & \text{если } i = j, \end{cases} \qquad i = 1...N, \quad j = 1...N.$$

Функционирование сети:

$$y_j(0) = x_j, \quad y_j(t+1) = f\left(\sum_{i=1}^{N} w_{ij} y_i(t)\right), \quad j = 1...N.$$

где w_{ij} – i-й синаптический вес j-го нейрона; x_i – i-й элемент входного сигнала сети; x_i^j – i-й элемент j-го вектора–образца, y_j – выход j-го нейрона; N – размерность входного сигнала; M – количество векторов–образцов.

Сеть функционирует циклически. В процессе функционирования уменьшается энергетическая функция:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} x_i x_j.$$

Критерием останова – неизменность выходов сети.

Характеристики сети.

Тип входных и выходных сигналов – биполярный (+1 и –1).

Размерности входа и выхода ограничены при программной реализации только возможностями вычислительной системы, на которой моделируется нейронная сеть, при аппаратной реализации – технологическими возможностями. Размерности входных и выходных сигналов совпадают.

Тип передаточной функции – жесткая пороговая; число синапсов в сети равно $M(M-1)$. Сеть, содержащая N нейронов, может запомнить не более $M = 0{,}15\,N$ образов. При этом запоминаемые образы не должны быть сильно коррелированы.

4) Области применения.

Ассоциативная память, адресуемая по содержанию; распознавание образов; задачи оптимизации (в том числе, комбинаторной оптимизации).

5) Недостатки.

Сеть обладает небольшой емкостью. Кроме того, наряду с запомненными образами в сети хранятся и их «негативы».

Размерность и тип входных сигналов совпадают с размерностью и типом выходных сигналов. Это существенно ограничивает применение сети в задачах распознавания образов.

При использовании сильно коррелированных векторов–образцов возможно зацикливание сети в процессе функционирования.

Квадратичный рост числа синапсов при увеличении размерности входного сигнала.

6) Преимущества.

Позволяет восстановить искаженные сигналы.

7) Модификации.

Существует различные модификации сети Хопфилда как с дискретными, так и с непрерывными состояниями и временем.

Для увеличения емкости сети и повышения качества распознавания образов используют мультипликативные нейроны. Сети, состоящие из таких нейронов называются сетями высших порядков.

Были предложены многослойные сети Хопфилда, которые обладают определенными преимуществами по сравнению с первоначальной моделью.

П.1.13. Входная звезда (Instar)

1) Название.

Instar (входная звезда).

2) Авторы и история создания.

Конфигурация Instar – фрагмент нейронных сетей, являющаяся моделью отдельных участков биологического мозга, была предложена и использована Гроссбергом во многих нейросетевых архитектурах.

3) Модель.

Входная звезда (на рис. П.1.4) представляет собой нейрон, на который подаются входные сигналы через синаптические веса.

Входная звезда реагирует на определенный входной вектор, которому она обучена. Это обеспечивается настройкой весов на соответствующий входной вектор. На выходе звезды формируется взвешенная сумма ее входов, представляющая свертку входного вектора с весовым вектором.

В процессе обучения осуществляется модификация весовых коэффициентов:

$$w_i(t+1) = w_i(t) + \eta\left(x_i - w_i(t)\right),$$

где η –коэффициент скорости обучения, равный в начальный момент 0,1 и уменьшающийся в процессе обучения до нуля.

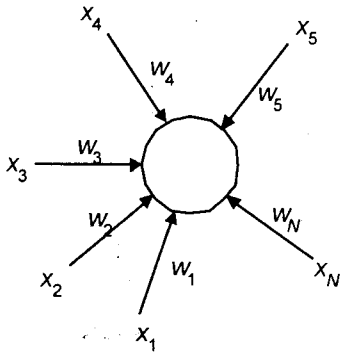


Рис. П.1.4. Сеть Instar

Входная звезда обладает способностью к обобщению, проявляющейся в возможности реагировать на незначительные изменения единичного входного вектора. Это достигается настройкой весов в процессе обучения таким образом, чтобы усреднить величины обучающих векторов, с целью реакции на любой вектор этого класса.

4) Области применения.

Рассмотренная конфигурация может быть использована в сетях распознавания образов.

5) Недостатки.

Каждая звезда в отдельности реализует слишком простую функцию. Из таких звезд невозможно построить нейронную сеть, которая реализовала бы любое заданное отображение. Это ограничивает практическое применение входных звезд.

6) Преимущества.

Входная звезда хорошо моделирует некоторые функции компонентов биологических нейронных сетей и может быть достаточно хорошей моделью отдельных участков мозга.

При решении практических задач входные звезды могут быть использованы для построения простых быстро обучаемых сетей.

7) Модификации.

Модели входных звезд могут использовать различные алгоритмы изменения с течением времени величин нормирующих коэффициентов обучения.

П.1.14. Сеть Кохонена (Kohonen's Neural Network)

1) Название.

Kohonen's Neural Network (сеть Кохонена)

Другое название.

Kohonen's Self Organizing Feature Map (SOFM) (самоорганизующаяся карта признаков Кохонена).

2) Авторы и история создания.

Предложена Кохоненом в 1984 г. К настоящему времени существует множество модификаций исходной модели с развитой математической теорией построения и функционирования.

3) Модель.

Хотя строение мозга в значительной степени предопределяется генетически, отдельные структуры мозга формируются в результате самоорганизации. Алгоритм Кохонена в некоторой степени подобен процессам, происходящим в мозге на основе самообучения.

Сеть Кохонена предназначена для разделения векторов входных сигналов на подгруппы. Сеть состоит из M нейронов, образующих прямоугольную решетку на плоскости (рис. П.1.5). Элементы входных сигналов подаются на входы всех нейронов сети. В процессе работы алгоритма настраиваются синаптические веса нейронов.

Входные сигналы (вектора действительных чисел) последовательно предъявляются сети, при этом требуемые выходные сигналы не определяются. После предъявления достаточного числа входных векторов, синаптические веса сети определяют кластеры. Кроме того, веса организуются так, что топологически близкие нейроны чувствительны к похожим входным сигналам.

Для реализации алгоритма необходимо определить меру соседства нейронов (меру близости). На рис. П.1.5 показаны зоны топологического соседства нейронов в различные моменты времени. $D_j(t)$ – множество нейронов, которые считаются соседями нейрона j в момент времени t. Размеры зоны соседства уменьшаются с течением времени.

Алгоритм Кохонена:

ШАГ 1. Инициализация сети: весовым коэффициентам сети, общее число которых равно MN, присваиваются малые случайные значения. $D_j(0)$ – начальная зона соседства.

ШАГ 2. Предъявление сети нового входного сигнала.

ШАГ 3. Вычисление расстояния d_j от входного сигнала до каждого нейрона j по формуле:

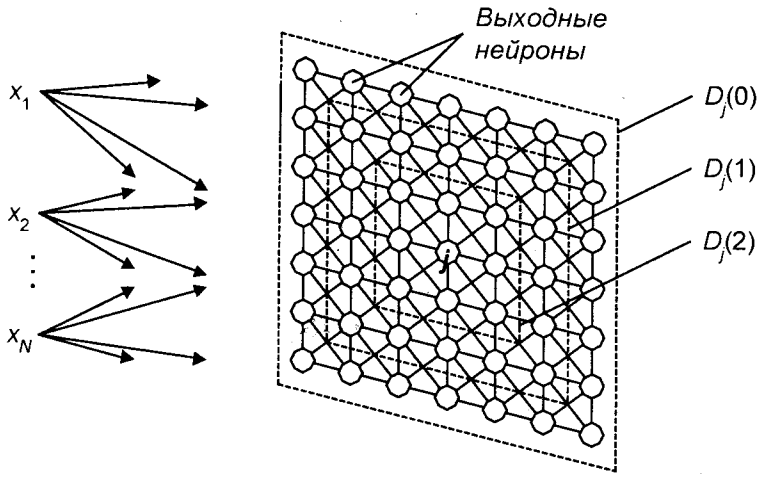


Рис. П.1.5. Зоны топологического соседства нейронов сети Кохонена

$$d_j = \sum_{i=1}^{N} \left(x_i(t) - w_{ij}(t)\right)^2,$$

где $x_i(t)$ – i-й элемент входного сигнала в момент времени t; $w_{ij}(t)$ – вес связи от i-го элемента входного сигнала к нейрону j.

Шаг 4. Выбор нейрон j^*, для которого расстояние d_j является наименьшим.

Шаг 5. Настройка весов для нейрона j^* и всех нейронов из его зоны соседства $D_{j^*}(t)$:

$$w_{ij}(t+1) = w_{ij} + \eta\left(x_i(t) - w_{ij}(t)\right),$$

где η – шаг обучения ($0 < \eta < 1$), уменьшающийся с течением времени до нуля.

Шаг 6. Возвращение к шагу 2.

4) Области применения.

Кластерный анализ, распознавание образов, классификация.

5) Недостатки.

Сеть может быть использована для кластерного анализа только в случае, если заранее известно число кластеров.

6) Преимущества.

В отличие от сети ART, сеть Кохонена способна функционировать в условиях помех, так как число классов фиксировано, веса модифицируются медленно, и настройка весов заканчивается после обучения (в сети ART настройка продолжается непрерывно).

7) Модификации.

Одна из модификаций состоит в добавлении к сети Кохонена сети MAXNET, которая определяет нейрон с наименьшим расстоянием до входного сигнала.

П.1.15. Сеть поиска максимума (MAXNET)

1) Название.

MAXNET (сеть поиска максимума).

Другие названия.

MAXNET with Competition through Lateral Inhibition (сеть поиска максимума с конкуренцией посредством латерального торможения).

Одна из сетей группы «winner takes all» («победитель получает все»).

Максимизатор.

2) Авторы и история создания.

Простая сеть, имеющая давнюю историю использования.

3) Модель.

MAXNET (рис. П.1.6) является сетью циклического функционирования. На каждой итерации большие сигналы на выходах нейронов подавляют слабые сигналы на выходах других нейронов. Это достигается за счет использования латеральных связей. Если в начале функционирования сети сигнал на выходе одного из нейронов имеет максимальное значение, то в конце функционирования на выходах всех нейронов, кроме максимального, формируются значения, близкие к нулю. Итерации сети завершаются после того, как выходные сигналы нейронов перестают меняться. Таким образом нейрон с наибольшим выходным сигналом «побеждает».

Характеристики сети.

Типы входных сигналов – целые или действительные; тип выходных сигналов – действительные. Размерности входа и выхода ограничены при программной реализации только возможностями вычислительной системы, на которой моделируется нейронная сеть, при аппаратной реализации – технологическими возможностями. Размерности входных и выходных сигналов совпадают. Тип передаточной функции – линейная с насыщением (используется линейный участок). Число синапсов в сети равно $N(N-1)$.

Формирование синаптических весов сети:

$$w_{ij} = \begin{cases} 1, & i = j, \\ -\varepsilon, & i \neq j,\ \varepsilon \leq \dfrac{1}{N}, \end{cases} \qquad i = 1 \ldots N, \quad j = 1 \ldots N,$$

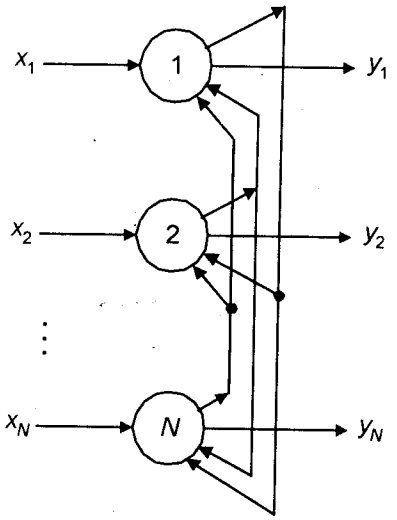


Рис. П.1.6. Структура нейронной сети MAXNET

где w_{ij} – i-й синаптический вес j-го нейрона; N – число элементов (размерность) входного сигнала, количество нейронов в сети.

Функционирование сети:

$$y_j(0) = x_j, \quad y_j(t+1) = f\left(y_j(t) - \varepsilon \sum_{i \neq j} y_i(t)\right), \quad i = 1...N, \quad i = 1...N,$$

где x_i – i-й элемент входного сигнала сети; y_j – выход j-го нейрона.

4) Области применения.

Совместно с сетью Хэмминга, в составе нейросетевых систем распознавания образов.

5) Недостатки.

Заранее не известно число итераций функционирования нейронной сети MAXNET. Она определяет, какой из входных сигналов имеет максимальное значение. Однако, в процессе функционирования сеть не сохраняет само значение максимального сигнала.

Квадратичный рост числа синапсов при увеличении размерности входного сигнала.

6) Преимущества.

Простота работы сети.

7) Модификации.

Для выделения сигнала с максимальным значением из некоторого множества сигналов может быть использована многослойная нейронная сеть с последовательными связями.

П.1.16. Выходная звезда (Outstar)

1) Название.

Outstar (выходная звезда).

2) Авторы и история создания.

Конфигурация Outstar – фрагмент нейронных сетей, предложенный и использованный Гроссбергом во многих нейросетевых архитектурах, и также, как и входная звезда, является моделью отдельных участков биологического мозга.

3) Модель.

Выходная звезда представляет собой нейрон, управляющий весовыми коэффициентами (рис. П.1.7). При возбуждении она вырабатывает требуемый сигнал для других нейронов.

При обучении веса выходной звезды настраиваются на множество векторов, близких к эталонному вектору в соответствии с выражением:

$$w_i(t+1) = w_i(t) + \eta\left(y_i - w_i(t)\right),$$

где η – коэффициент, изменяющийся в процессе обучения от единицы до нуля.

Выходной сигнал выходной звезды представляет собой статистическую характеристику обучающего набора.

Входные и выходные звезды могут быть взаимно соединены в сети любой сложности.

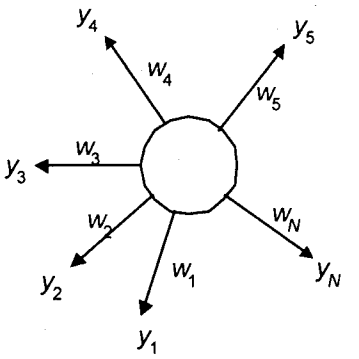


Рис. П.1.7. Сеть Outstar

4) Области применения.

Рассмотренная конфигурация может быть использована как компонент нейронных сетей для распознавания образов.

5) Недостатки.

Каждая звезда в отдельности реализует слишком простую функцию. Вычислительные возможности нейронных сетей, составленных из таких звезд, ограничены.

6) Преимущества.

При решении практических задач выходные звезды могут быть использованы для построения простых быстро обучаемых сетей.

7) Модификации.

В моделях выходных звезд могут быть использованы различные алгоритмы изменения с течением времени величин нормирующих коэффициентов обучения.

П.1.17. Сеть радиального основания (Radial Basis Function Network)

1) Название.

Radial Basis Function Network (RBFN).

Другое название.

Сеть радиальных базисных функций.

2) Авторы и история создания.

Под парадигмой RBFN понимается архитектура, предложенная Moody и Darken в 1989 г. К классу RBFN относят также вероятностные и регрессионные нейронные сети.

3) Модель.

В общем случае под термином Radial Basis Function Network понимается нейронная сеть, которая содержит слой скрытых нейронов с радиально симметричной активационной функцией, каждый из которых предназначен для хранения отдельного эталонного вектора (в виде вектора весов). Для построения RBFN необходимо выполнение следующих условий.

Во-первых, наличие эталонов, представленных в виде весовых векторов нейронов скрытого слоя. Во-вторых, наличие способа измерения расстояния входного вектора от эталона. Обычно это стандартное евклидово расстояние. В-третьих, специальная функция активации нейронов скрытого слоя, задающая выбранный способ измерения расстояния. Обычно используется функция Гаусса, существенно усиливающая малую разницу между входным и эталонным векторами.

Другими словами, выходной сигнал эталонного нейрона скрытого слоя y_i – это функция (гауссиан) только от расстояния ρ_i между входным и эталонным векторами:

$$\rho_i = \sqrt{\sum_{j=1}^{N}(x_j - c_{ij})^2}, \qquad y_i = \exp\left[-\left(\frac{\rho_i - R}{\sigma_i}\right)^2\right], \qquad i = 1, \ldots, L,$$

где $X = (x_1, \ldots, x_N)$ – входной вектор; $C_i = (c_{i1}, \ldots, c_{iN})$ – весовой вектор i-го эталонного нейрона скрытого слоя; R, σ_i – параметры активационной функции; L – число эталонов.

Обучение слоя образцов–нейронов сети подразумевает предварительное проведение кластеризации для нахождения эталонных векторов и определенных эвристик для определения значений σ_i.

Нейроны скрытого слоя соединены по полносвязной схеме с нейронами выходного слоя, которые осуществляют взвешенное суммирование. Для нахождения значения весов от нейронов скрытого к выходному слою используется линейная регрессия.

В общем случае активационные функции нейронов скрытого слоя могут отражать законы распределения случайных величин (вероятностные нейронные сети) либо характеризовать различные аналитические зависимости между переменными (регрессионные нейронные сети).

4) Области применения.

Распознавание образов, классификация.

5) Недостатки.

Заранее должно быть известно число эталонов, а также эвристики для построения активационных функций нейронов скрытого слоя.

6) Преимущества.

Отсутствие этапа обучения в принятом смысле этого слова.

7) Модификации.

В моделях RBFN могут быть использованы различные способы измерения расстояния между векторами, а также функции активации нейронов скрытого слоя.

П.1.18. Нейронные сети, имитирующие отжиг (Neural Networks with Simulated Annealing Training Algorithm)

1) Название.

Neural Networks with Simulated Annealing Training Algorithm (нейронные сети, обучаемые по методу имитации отжига).

2) Авторы и история создания.

В 50-е годы была разработана математическая модель отжига металла, согласно которой металл в процессе кристаллизации из жидкой фазы проходит через ряд состояний, характеризующихся различным значением энергии. Атомы металла стремятся к состоянию минимума энергии. При высоких температурах атомы могут совершать движения, приводящие к переходу в состояния с большими значениями энергии. В процессе постепенного охлаждения металла достигается глобальный минимум энергии.

Алгоритм имитации отжига – вариант итеративного решения оптимизационных задач, в соответствии с которым, как и в реальных условиях отжига, разрешаются шаги, повышающие значения функции ошибки (энергии).

На основе этой математической модели в 80-е годы был создан алгоритм оптимизации, обладающий высокой эффективностью при обучении нейронных сетей.

3) Модель.

Алгоритм имитации отжига может быть использован для обучения как многослойных, так и полносвязных сетей. При этом функции активации сети не обязательно должны быть непрерывно дифференцируемыми. В качестве функции ошибки можно использовать среднеквадратическое отклонение.

Используются градиентный и стохастический алгоритмы имитации отжига.

В градиентном алгоритме на каждой итерации вычисляется направление антиградиента адаптивного рельефа и делается шаг заданной величины. В процессе обучения величина шага уменьшается. Большие значения шага на начальных итерациях обучения могут приводить к возможному возрастанию значения функции ошибки. В конце обучения величина шагов мала и значение функции ошибки уменьшается на каждой итерации.

При обучении нейронной сети на основе стохастического алгоритма имитации отжига совершаются шаги по адаптивному рельефу в случайных направлениях. Пусть на итерации k система находится в точке S адаптивного рельефа, характеризующейся значением энергии E. При этом шаг из точки S в точку S' со значением энергии E' ($E' > E$), приводящий к увеличению значения функции ошибки (энергии) на величину $\Delta = E' - E$), допускается с заданной вероятностью.

Критерии останова функционирования сети:

- функционирование многослойных сетей без обратных связей заканчивается после получения выходных сигналов нейронов последнего слоя;

- для сетей циклического функционирования (полносвязных, многослойных с обратными связями и др.): останов после K итераций; останов после прекращения изменения выходных сигналов;
- достижение некоторого заданного значения функции ошибки.

Характеристики сети.

Типы входных и выходных сигналов – любые. Размерности входа и выхода – любые, ограничения связаны с заданной скоростью обучения (медленная сходимость для сетей большой размерности). Емкость сети в общем случае не определена. Тип передаточной функции – любая, ограниченная по области значений. Количество синапсов и смещений сети ограничено скоростью обучения. Для сетей с числом синапсов порядка нескольких сотен алгоритм имитации отжига очень эффективен. Для программно реализованных на персональном компьютере сетей с десятками тысяч настраиваемых параметров процесс обучения по методу отжига длится катастрофически долго.

4) Области применения.

С помощью алгоритма имитации отжига можно строить отображения векторов различной размерности. К построению таких отображений сводятся многие задачи распознавания образов, адаптивного управления, многопараметрической идентификации, прогнозирования и диагностики.

5) Недостатки.

Низкая скорость сходимости при обучении нейронных сетей большой размерности.

6) Преимущества.

«Тепловые флуктуации», заложенные в алгоритм, дают возможность избегать локальных минимумов. Показано, что алгоритм имитации отжига может быть использован для поиска глобального оптимума адаптивного рельефа нейронной сети.

7) Модификации.

Алгоритмы имитации отжига различаются структурами нейронных сетей, для обучения которых они используются, а также правилами, в соответствии с которыми допускаются шаги, увеличивающие энергию системы.

Модифицированные алгоритмы имитации отжига используются также для решения задач комбинаторной оптимизации.

П.1.19. Однослойный персептрон (Single Layer Perceptron)

1) Название.

Single Layer Perceptron (однослойный персептрон).

2) Авторы и история создания.

Разработан Ф. Розенблаттом в 1959 г.

3) Модель.

Однослойный персептрон способен распознавать простейшие образы. Отдельный персептронный нейрон вычисляет взвешенную сумму элементов входного сигнала, вычитает значение смещения и пропускает результат через жесткую пороговую функцию, выход которой равен +1 или –1 в зависимости от принадлежности входного сигнала к одному из двух классов.

Персептрон, состоящий из одного нейрона, формирует две решающие области, разделенные гиперплоскостью. Уравнение, задающее разделяющую гиперплоскостью (прямую – в случае двухвходового персептронного нейрона), зависит от значений синаптических весов и смещения.

Классический алгоритм настройки персептрона, предложенный Розенблаттом, заключается в следующем.

ШАГ 1.Инициализация синаптических весов $w_i(0)$ (i = 1, …, N) и смещения b некоторыми малыми случайным числами.

ШАГ 2. Предъявление персептрону нового входного вектора $x = (x_1, \ldots, x_N)$ и требуемого выходного сигнала d.

ШАГ 3. Вычисление выходного сигнала персептрона:

$$y(t) = f\left(\sum_{i=1}^{N} w_i(t)\, x_i(t) - b\right).$$

ШАГ 4. Настройка значений весов персептрона:

$$w_i(t+1) = w_i(t) + \eta\big(d - y(t)\big)x_i(t), \quad i = 1,\ldots,N.$$

где η – коэффициент скорости обучения ($0 < \eta < 1$).

Если решение правильное, веса не модифицируются.

ШАГ 5. Переход к шагу 2.

Характеристики сети.

Тип входных сигналов – бинарные или аналоговые (действительные). Размерности входа и выхода ограничены при программной реализации только возможностями вычислительной системы, на которой моделируется нейронная сеть, при аппаратной реализации – технологическими возможностями. Емкость сети совпадает с числом нейронов.

4) Области применения:

Распознавание образов, классификация.

5) Недостатки.

Простые разделяющие поверхности (гиперплоскости) дают возможность решать лишь несложные задачи распознавания.

6) Преимущества.

Программные или аппаратные реализации модели очень просты. Простой и быстрый алгоритм обучения.

7) *Модификации.*

Многослойные персептроны дают возможность строить более сложные разделяющие поверхности и поэтому находят более широкое применение при решении задач распознавания.

П.2. Алгоритмы обучения нейронных сетей

Процесс обучения нейронной сети может рассматриваться как настройка архитектуры и весов связей для эффективного выполнения поставленной задачи. Обычно итеративная настройка весов связей осуществляется в соответствии с обучающей выборкой. Свойство сети обучаться на примерах делает их более привлекательными по сравнению с системами, которые следуют системе правил функционирования, сформулированной экспертами.

Для организации процесса обучения, во-первых, надо иметь модель внешней среды, в которой функционирует нейронная сеть. Эта модель определяет парадигму обучения. Во-вторых, необходимо понять, как модифицировать весовые параметры сети.

Существуют три парадигмы обучения: *с учителем*, *без учителя* (самообучение) и *смешанная*. В первом случае на каждый входной пример существует требуемый ответ. Веса настраиваются таким образом, чтобы выходы сети как можно более близкие к требуемым ответам. Более «жесткий» вариант обучения с учителем предполагает, что известна только критическая оценка правильности выхода нейронной сети, а не сами требуемые значения выхода. Для обучения без учителя не нужно знания требуемых ответов на каждый пример обучающей выборки. В этом случае происходит распределение образцов по категориям (кластерам) в соответствии с внутренней структурой данных или степенью корреляции между образцами. При смешанном обучении весовые коэффициенты одной группы нейронов настраиваются посредством обучения с учителем, а другой группы – на основе самообучения.

В процессе обучения учитываются следующие свойства нейронных сетей: емкость сети, сложность образцов и вычислительная сложность. Под емкостью сети понимается число запоминаемых образцов, с учетом сформированных функций и границ принятия решений. Сложность образцов определяет число обу-

чающих примеров, необходимых для достижения способности сети к обобщению.

Известны четыре основных правила обучения, обусловленные связанными с ними архитектурами сетей (табл. П.2.1): *коррекция ошибки*, *правило Больцмана*, *правило Хебба* и *метод соревнования*.

Коррекция ошибки.

Для каждого входного примера задан требуемый выход d, который может не совпадать с реальным y. Правило обучения при коррекции по ошибке состоит в использовании разницы $(d - y)$ для изменения весов, с целью уменьшения ошибки рассогласования. Обучение производится только в случае ошибочного результата. Известны многочисленные модификации этого правила обучения.

Правило Больцмана.

Правило Больцмана является стохастическим правилом обучения, обусловленным аналогией с термодинамическими принципами. В результате его выполнения осуществляется настройка весовых коэффициентов нейронов в соответствии с требуемым распределением вероятностей. Обучение правилу Больцмана может рассматриваться как отдельный случай коррекции по ошибке, в котором под ошибкой понимается расхождение корреляций состояний в двух режимах.

Правило Хебба.

Правило Хебба является самым известным алгоритмом обучения нейронных сетей, суть которого заключается в следующем: если нейроны с обеих сторон синапса возбуждаются одновременно и регулярно, то сила синаптической связи возрастает. Важной особенностью является то, что изменение синаптического веса зависит только от активности связанных этим синапсом нейронов. Предложено большое количество разновидностей этого правила, различающихся особенностями модификации синаптических весов.

Метод соревнования.

В отличие от правила Хебба, в котором множество выходных нейронов могут возбуждаться одновременно, здесь выходные нейроны соревнуются между собой. И выходной нейрон с максимальным значением взвешенной суммы является «победителем» («победитель забирает все»). Выходы же остальных выходных нейронов устанавливаются в неактивное состояние. При обучении модифицируются только веса нейрона–«победителя» в сторону увеличения близости к данному входному примеру.

Таблица П.2.1

Известные алгоритмы обучения

Парадигма	Обучающее правило	Архитектура нейронной сети	Алгоритм обучения	Задачи
С учителем	Коррекция ошибки	Однослойный и многослойный персептрон	Алгоритмы обучения персептрона Обратное распространение; Adaline и Madaline	Классификация образов; Аппроксимация функций предсказание; управление
	Больцман	Рекуррентная	Алгоритм обучения Больцмана	Классификация образов
	Хебб	Многослойная прямого распространения	Линейный дискриминантный анализ	Анализ данных; классификация образов
	Соревнование	Соревнование	Векторное квантование	Категоризация внутри класса; сжатие данных
		Сеть ART	ART Map	Классификация образов
Без учителя	Коррекция ошибки	Многослойная прямого распространения	Проекция Саммона	Категоризация внутри класса; анализ данных
	Хебб	Прямого распространения или соревнование	Метод главных компонентов	Анализ данных; сжатие данных
		Сеть Хопфилда	Обучение ассоциативной памяти	Ассоциативная память
	Соревнование	Соревнование	Векторное квантование	Категоризация; сжатие данных
		SOM Кохонена	SOM Кохонена	Категоризация; анализ данных
		Сети ART	ART1, ART2	Категоризация
Смешанная	Коррекция ошибки и соревнование	Сеть RBFN	Алгоритм обучения RBFN	Классификация образов; аппроксимация функций; предсказание; управление

Это правило позволяет группировать входные данные на категории (кластеры) и представлять их отдельными выходными нейронами.

Нейронная сеть считается устойчивой, если после конечного числа итераций обучения ни один из примеров обучающей выборки не изменяет своей принадлежности в кластерах. Однако сеть не перестанет обучаться, если параметр скорости обучения не равен нулю. Но эта искусственная остановка обучения вызывает другую проблему, называемую пластичностью и связанную со способностью сети к адаптации к новым данным. Возникает дилемма стабильности–пластичности Гроссберга.

Список представленных в табл. П.2.1 алгоритмов обучения нейронных сетей не является исчерпывающим. В последней колонке перечислены задачи, для которых может быть применены эти алгоритмы. Каждый алгоритм обучения ориентирован на сеть определенной архитектуры и предназначен для ограниченного класса задач. Кроме рассмотренных, следует упомянуть некоторые другие алгоритмы: Adaline и Madaline, линейный дискриминантный анализ, проекции Саммона, метод главных компонентов.

П.3. Глоссарий

ADALINE – Адалина – одно из наименований для линейного нейрона: ADAptive LINear Element.

adaptive learning rate – адаптивный параметр обучения – параметр процедуры обучения, который изменяется по определенному алгоритму так, чтобы минимизировать время обучения.

architecture – архитектура – описание числа слоев в нейронной сети, передаточных функций каждого слоя, числа нейронов в каждом слое и связей между слоями.

artificial neural networks (ANN) – искусственные нейронные сети (ИНС).

average error – средняя ошибка сети по всему набору обучающих векторов.

backpropagation batch – разновидность алгоритма обучения с обратным распространением ошибки, когда коррекция весов и смещений производится один раз за период обучения – после предъявления всех векторов обучающей последовательности («пачки» векторов, batch).

backpropagation learning rule – обратного распространения правило – обучающее правило, в котором веса и смещения регулируются (подстраиваются) по производной ошибки от выхода сети через промежуточные слои к первому, в соответствии с выражением:

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}}(t) + \alpha \Delta w_{ij}(t-1),$$

где $E(t)$ – квадрат ошибки сети на t-м шаге обучения, η – коэффициент скорости обучения (learning rate), α – коэффициент импульса (momentum) или коэффициент инерционности.

Обычно применяется для обучения многослойных сетей прямого распространения. Иногда называется обобщенным дельта–правилом (generalized delta rule).

backpropagation online – обратное распространение в режиме реального времени – модификация алгоритма обучения по методу обратного распространения ошибки, когда веса и смещения сети корректируются после предъявления каждого нового образа (вектора) обучающей последовательности.

backtracking search – поиск с возвратом – одномерный поисковый алгоритм, который начинает поиск с единичным шагом и возвращается в исходную точку с уменьшением шага до тех пор, пока не будет получено приемлемое уменьшение целевой функции.

batch – пачка – матрица входных или целевых векторов, приложенных к сети одновременно, изменение (подстройка) весов и смещений сети происходит после просмотра (обработки) всех векторов входной матрицы.

batching – процесс представления матрицы (пачки) входных векторов для одновременного вычисления матрицы выходных векторов и/или новых весов и смещений.

Bayesian framework – байесовский подход – допущение, что веса и смещения сети являются случайными переменными с определенными распределениями.

BFGS quasi-Newton algorithm – разновидность оптимизационного алгоритма Ньютона, в котором аппроксимация матрицы Гессе (матрицы вторых производных) получается из градиентов, вычисленных на каждой итерации алгоритма.

bias – смещение – параметр нейрона, который суммируется со взвешенными входами нейрона, образуя входную величину (аргумент) для функции активации нейрона.

bias vector – вектор смещения – вектор-столбец величин смещений для слоя нейронов.

Brent's search – одномерный поисковый метод оптимизации, который является комбинацией метода золотого сечения и квадратичной интерполяции.

Charalambous' search – одномерный гибридный поисковый метод, использующий кубическую интерполяцию.

classification – классификация – ассоциация входного вектора с некоторым выходным (целевым).

competitive layer – конкурирующий слой – слой нейронов в которых только нейрон с максимальным входом имеет активизированный выход (например, выход 1), а все другие нейроны – выход 0. Нейроны конкурируют друг с другом за возможность реагировать на данный входной вектор.

competitive learning – конкурирующее обучение – обучение без учителя в слое конкурирующих нейронов. После обучения слой распределяет входные векторы среди своих нейронов.

competitive transfer function – конкурирующая передаточная (активационная) функция – преобразует входной (для слоя конкурирующих нейронов) вектор в нулевой для всех нейронов за исключением нейрона–«победителя», для которого выход будет равен единице.

concurrent input vectors – параллельные входные векторы – имя, данное матрице входных векторов, которые должны представляться в сети «одновременно». Все векторы в матрице используются для получении одного набора изменений в весах и смещениях нейронной сети.

conjugate gradient algorithm – метод сопряженных градиентов – разновидность градиентного метода поиска минимума функции.

connection – соединение – односторонняя связь между нейронами в сети.

connection strength – сила связи (соединения) – уровень связи между двумя нейронами в сети. Часто количественно выражается весом и определяет эффект влияния одного нейрона на другой.

convergence tolerance – это максимально допустимая величина ошибки во время обучения. Если tolerance = 0, то выходной результат (output), выдаваемый нейронной сетью, должен абсолютно точно совпадать с образцом для обучения (pattern). В большинстве случаев это нереально. При tolerance = 0,1 выход нейронной сети (output) будет рассматриваться как корректный, если он отличается не более чем на 10% (в среднем квадратическом смысле) от заданного значения (pattern). Процесс обучения будет продолжаться до тех пор, пока значение ошибки не снизится до установленного параметром tolerance предела.

cycle – цикл – однократное представление входного вектора, соответствующие ему вычисления выходного вектора и новых весов и смещений сети.

dead neurons – «мертвые» нейроны – нейроны конкурирующего слоя , которые никогда не выигрывают любую конкуренцию за входной вектор в процессе обучения.

decision boundary – граница решения – линия в гиперпространстве, определяемая векторами весов и смещений, для которой вход сети является нулем.

delta-bar-delta – метод обучения сети с адаптивным (и индивидуальным для каждого веса) подбором коэффициента скорости обучения (learning rate). Адаптация осуществляется по следующим формулам:

$$\Delta\eta(t) = \begin{cases} k, & \text{если } \bar{\delta}(e-1)\delta(t) > 0, \\ -\phi\,\eta(t), & \text{если } \bar{\delta}(e-1)\delta(t) < 0, \\ 0, & \text{если } \bar{\delta}(e-1)\delta(t) = 0, \end{cases}$$

где $\delta(t) = \dfrac{\partial E}{\partial w}(t)$ – производная ошибки сети по весовому коэффициенту;

$\bar{\delta}(t) = (1-\theta)\delta(t) + \theta\,\bar{\delta}(t-1)$ – взвешенное среднее.

Метод применяется в сочетании с алгоритмом **backpropagation batch.**

delta rule – дельта-правило – правило обучения Уидроу–Хоффа (the Widrow–Hoff rule). Как известно, возможности персептрона ограничены бинарными выходами. Уидроу и Хофф расширили алгоритм обучения персептрона для случая непрерывных выходов с использованием сигмоидальной функцией активации. Кроме того, они доказали, что сеть при определенных условиях будет сходиться к любой функции, которую она может представить. Их первая модель – Адалин – имеет один выходной нейрон, более поздняя модель – Мадалин – обобщена для случая с многими выходными нейронами.

delta vector – дельта-вектор – вектор производных ошибки выхода сети по отношению к выходам какого-либо слоя сети.

distance – расстояние – расстояние между нейронами, определяемое в каком-либо смысле.

early stopping – ранняя остановка – техника, базирующаяся на разбиении исходных данных на три подмножества. Первое подмножество является обучающим и используется для вычисления градиента и коррекции (обновления) весов и смещений сети. Второе подмножество используется для проверки правильности сделанных коррекций. Когда для данного подмножества и определенного количества итераций ошибка увеличивается, обучение прекращается, а веса и смещения принимаются соответствующими минимальной ошибке на этом подмножестве. Третье подмножество является тестирующим для обученной сети.

epoch – период (эпоха) – представление набора обучающих (входных и выходных – целевых) векторов сети и вычисление новых весов и смещений. Векторы могут предъявляться поочередно или все вместе, пакетом, «пачкой».

error jumping – скачок ошибки – внезапное возрастание суммарной квадратической ошибки сети в процессе ее обучения. Часто возникает из-за слишком большой величины параметра обучения (learning rate).

error margin – граница ошибки – выходы сети, для которых ошибка меньше данной величины будут считаться «правильными», корректными.

error ratio – коэффициент ошибки – один из параметров процедуры обучения в сетях с обратным распространением ошибки.

error vector – вектор ошибки – различие между целевым вектором и выходным вектором сети.

feedback network – сеть с обратной связью – сеть с соединениями с выхода сети на ее вход. Соединение обратной связи может охватывать различные слои.

feedforward network – сеть с прямой связи – многослойная сеть, в который каждый слой своими входами имеет выходы только предшествующих слоев.

Fletcher–Reeves update – метод Флетчера–Ривса – метод для вычисления сопряженных направлений, которые используются в оптимизационной процедуре метода сопряженных градиентов.

function approximation – аппроксимация функций – одна из задач, которую может выполнить нейронная сеть после ее обучения.

gaussian transfer function – активационная функция в виде функции Гаусса (см. radial basis transfer function).

generalization – обобщение – свойство сети устанавливать свой выход для нового входного вектора близким к выходам похожих на него (в смысле какого-либо расстояния) входных векторов из обучающей последовательности.

generalized regression network – обобщенная сетевая регрессия – приближение непрерывной функции с произвольной точностью с помощью нейронной сети, которое может быть получено при достаточном числе скрытых нейронов.

global minimum – глобальный минимум – наименьшее значение функции во всей области входных параметров. Методы градиентного спуска изменяют веса и смещения так, чтобы найти (достичь) глобальный минимум ошибки сети.

golden section search – метод золотого сечения – один из одномерных поиска экстремума функции, не требующий вычисления производной и отличающийся высокой скоростью сходимости.

gradient descent – градиентный спуск – процесс получения изменений (коррекций) в весах и смещениях сети, при котором данные изменения пропорциональны производным сетевой ошибки по этим весам и смещениям, приводящий к минимизации сетевой ошибки.

hard limit transfer function – пороговая активационная функция – отображает неотрицательные числа в единицу, отрицательные в ноль.

Hebb learning rule – Хебба правило обучения – исторически первое предложенное правило обучения нейронной сети. Веса корректируются пропорционально произведению выходов предыдущего и последующего нейронов.

hidden layer – скрытый слой – слой нейронов, непосредственно не соединенный с выходом сети.

home neuron – внутренний нейрон – нейрон в центре некоторой окрестности.

hybrid bisection-cubicsearch – гибридная одномерная поисковая процедура оптимизации, объединяющая методы половинного деления (бисекции) и кубической интерполяции.

initialization – инициализация – установка начальных значений весов и смещений нейронной сети, обычно, присваивание синаптическим весам и смещениям сети случайных значений из заданного диапазона.

input layer – входной слой – слой нейронов, на который непосредственно поступают входы сети.

input noise – входной шум – опция обучения, определяющая уровень нормально распределенного шума, добавляемого к входным обучающим образцам. При обучении с таким дополнительным шумом обычно

предотвращается «переобучение» сети (overfitting) и улучшается ее способность к обобщению.

input space – входное пространство – область всех возможных значений входного вектора.

input vector – входной вектор сети.

input weights – входные веса – веса, с которыми входные сигналы поступают в сеть.

input weight vector – вектор входных весов.

interrogate – опрос – предъявление обученной сети входного вектора и вычисление сетью соответствующего выходного.

Jacobian matrix – Якобиан – матрица первых производных ошибки сети по отношению к ее весам и смещениям.

Kohonen learning rule – правило обучения Кохонена – правило обучения, согласно которому веса нейронов выбираются в соответствии с элементами входного вектора.

layer – слой – группа нейронов, имеющие соединения с одними и теми же источниками-входами и посылающие свои выходные сигнала к одним и тем же потребителям.

layer diagram – диаграмма слоев – представление архитектуры нейронной сети в виде образующих ее слоев нейронов и матриц весовых коэффициентов. Активационные функции отображаются символически. Показываются размеры данных матриц. а также размеры входного и выходного векторов сети. Отдельные нейроны не отображаются.

layer weights – веса (весовые коэффициенты) слоя нейронов, соединяющие данный слой с другими. В случае рекуррентных связей должны иметь ненулевые задержки.

learning – обучение – процесс коррекции весов и смещений сети, при котором достигается ее желательное функционирование.

learning rate – коэффициент скорости обучения (параметр обучения), который определяет скорость изменения величин весов и смещений сети в процессе ее обучения, обычно при использовании алгоритма обратного распространения ошибки. Чем он больше, тем быстрее обучается сеть. Допустимые значения параметра от 0,0 до 1,0; хорошим начальным приближением считается величина 0,1. Если данный параметр велик, процесс обучения может потерять устойчивость.

learning rule – обучающее правило – метод модификации весов и смещений сети в процессе ее обучения.

Levenberg–Marquardt – алгоритм обучения Левенберга–Марквардта, обеспечивающий в 10–100 раз более быстрое обучения сети, чем алгоритм обратного распространения ошибки, использующий градиентную оптимизацию.

line search function – одномерная поисковая процедура – процедура поиска минимума функции по заданному направлению.

linear transfer function – линейная передаточная (активационная) функция.

local minimum – локальный минимум – минимальное значение функции в некоторой ограниченной области изменения аргумента. Не обязательно является глобальным минимумом.

log-sigmoid transfer function – лог-сигмоидная передаточная (активационная) функция, определяемая выражением:

$$f(s) = \frac{1}{1 + e^{-as}}.$$

Manhattan distance – расстояние Манхэттена – расстояние между двумя векторами ***x*** и ***y***, определяемое соотношением:

$$D = \sum_i |x_i - y_i|.$$

maximum number of epochs – максимальное число периодов (эпох), которые нужно использовать для обучения. Один период эквивалентен одному полному представлению всех образцов (векторов) обучающей выборки.

maximum step size – максимальный шаг – максимальный шаг изменения аргумента в одномерной поисковой процедуре. Величина вектора весовых коэффициентов в процессе обучения сети на одной итерации не может увеличиваться более, чем на размер максимального шага.

mean square error function – средняя квадратическая ошибка сети (средний квадрат разности между реальным и требуемым выходами).

momentum – импульс – метод ускорения процесса обучения для алгоритма обратного распространения. Заключается в добавлении к корректируемому весу числа, пропорционального предыдущему значению веса.

Используя метод импульса, сеть стремится идти по дну узких оврагов поверхности ошибки (если таковые имеются), а не двигаться от склона к склону. Метод хорошо работает на одних задачах, но может дать отрицательный эффект на других.

momentum constant – коэффициент импульса – константа, используемая в методе импульса и устанавливаемая обычно на уровне 0,9.

neighborhood – окрестность – в данном случае группа нейронов, находящихся в пределах определенного расстояния от выбранного нейрона.

net input vector – входной вектор сети.

neuron – нейрон – базовый элемент нейронных сетей. Имеет входы, снабженные весами, смещение, суммирующий элемент и выходную активационную функцию. Является аналогом биологического нейрона.

neuron diagram – диаграмма нейронов – сетевая архитектура, отображаемая фигурой, показывающей нейроны и веса связей между ними. Активационные функции нейронов отображаются символически.

neuron saturation – насыщение нейрона – состояние нейрона, когда значительные изменения его входов приводят к незначительному изменению выхода. Когда нейрон насыщается, процесс его обучения становится неэффективным.

number of hidden layers – число скрытых слоев в нейронной сети.

output layer – выходной слой – слой нейронов, выход которого является выходом всей сети.

output vector – выходной вектор – выход нейронной сети. Каждый элемент этого вектора является выходом одного из нейронов выходного слоя.

output weight vector – выходной вектор весовых коэффициентов – вектор-столбец весовых коэффициентов для выходов нейрона.

outstar learning rule – правило обучения выходной звезды – в то время как входная звезда возбуждается при предъявлении определенного входного вектора, выходная звезда имеет дополнительную функцию; она вырабатывает требуемый возбуждающий сигнал для других нейронов всякий раз, когда возбуждается. В процессе обучения нейрона выходной звезды, его веса настраиваются в соответствии с требуемым целевым вектором.

overfitting – «переобучение» – ситуация, когда на обучающей последовательности ошибки сети были очень малы, но на новых данных становятся большими.

partan method – партан–метод – улучшенная модификация градиентного метода (метода наискорейшего спуска) минимизации функций. Известны итерационный (*k*-партан) и модифицированный партан–методы. Итерационный партан–метод строится следующим образом. В начальной базовой точке вычисляется градиент и делается шаг наискорейшего спуска. Далее – снова наискорейший спуск и так *k* раз. После *k* шагов наискорейшего спуска проводится одномерная оптимизация с начальным шагом, равным единице в направлении между начальной базовой точкой и полученной после *k* шагов. После этого цикл повторяется.

pass – проход – каждое прохождение через все обучающие и целевые векторы.

pattern – образ (вектор).

pattern association – образов ассоциация – задача, решаемая обученной нейронной сетью и заключающаяся в соотнесении «правильного» выходного вектора каждому предъявляемому входному.

pattern recognition – образов распознавание – задача, решаемая обученной нейронной сетью и заключающаяся в отнесении предъявленного входного вектора (изображения) к одному из классов.

performance function – функция эффективности – обычно средняя квадратическая ошибка сети.

perceptron – персептрон – обычно однослойная сеть с пороговой активационной функцией нейронов. Обучение такой сети производится по специальному алгоритму.

perceptron learning rule – правило обучения персептрона. Гарантирует обучение однослойного персептрона с пороговой активационной функцией за конечное число шагов.

positive linear transfer function – положительно-линейная активационная функция – функция, значения которой равны нулю при отрицательных значениях аргумента и пропорциональны аргументу для его положительных значений.

postprocessing – послеобработка – преобразует нормализованное значение выхода сети в его естественное значение.

Powell–Beale restarts – метод Поуэлла–Била – метод определения сопряженных направлений. Используется в оптимизационном алгоритме сопряженных градиентов.

preprocessing – предобработка – выполнение некоторых преобразований входных или целевых (выходных) данных перед их представлением нейронной сети, обычно заключающаяся в приведении данных к некоторому одинаковому (единичному) масштабу.

principal component analysis – метод главных компонентов – ортогонализация компонентов входных векторов сети. Процедура может быть также использована для понижения (уменьшения) размерности входных векторов путем исключения неинформативных компонентов.

quasi-Newton algorithm – квази-ньютоновский алгоритм – класс алгоритмов оптимизации, основанных на методе Ньютона. Аппроксимация матрицы Гессе (матрицы вторых частных производных) вычисляется на каждой итерации с использованием градиента.

quickprop – быстрое распространение – алгоритм обучения сети, базирующийся на следующих допущениях:

- зависимость $E(w)$ ошибки сети от каждого весового коэффициента может быть аппроксимирована выпуклой параболой;
- изменение какого-либо веса в процессе настройки не оказывает влияние на другие веса.

Правило изменения (коррекции) весов определяется формулой:

$$\Delta w(t) = \frac{\delta(t)}{\delta(t-1) - \delta(t)} \Delta w(t-1) - \eta\delta(t),$$

где $\delta(t) = \frac{\partial E}{\partial w}(t)$ – производная ошибки сети по весовому коэффициенту на t-м шаге обучения.

Заметим, что знаменатель приведенной дроби представляет собой, по сути, оценку (с точностью до знака) второй производной ошибки по w, так что формула коррекции отображает метод Ньютона минимизации скалярной функции:

$$\varphi(x): \quad \Delta x = -\frac{\varphi'(x)}{\varphi''(x)}.$$

Параметр η, как и в других алгоритмах – это коэффициент скорости обучения (learning rate).

radial basis function networks (RBFN) – сеть радиального основания – любая сеть, которая содержит скрытый слой нейронов с радиально симметричной активационной функцией, каждый из которых предназначен для хранения отдельного эталонного вектора (в виде вектора весов). Для построения RBFN необходимо выполнение следующих условий:

- наличие эталонов, представленных в виде весовых векторов нейронов скрытого слоя;

- наличие способа измерения расстояния входного вектора от эталона; обычно это стандартное евклидово расстояние;
- наличие специальной функции активации нейронов скрытого слоя, задающей выбранный способ измерения расстояния.

radial basis transfer function – радиальная базисная функция активации (или функция Гаусса) – функция активации для радиального базисного нейрона, определяемая соотношением:

$$f(s) = e^{-s^2}.$$

randomize patterns – рандомизация (случайное перемешивание) образцов обучающей последовательности перед каждым периодом (epoch) обучения. Зачастую улучшает сходимость процесса обучения сети.

regularization – регуляризация – модификация функции эффективности сети (обычно являющейся средней квадратической ошибкой) путем добавления помноженной на некоторый коэффициент суммы квадратов весовых коэффициентов.

RMS error (root mean squared error) – средняя квадратическая ошибка сети.

RPROP (resilient propagation) – «упругое распространение» – один из алгоритмов обучения нейронных сетей, основанный на использовании не величины производной ошибки сети, а ее знака. Веса корректируются в соответствии с выражениями:

$$w_{ij}(t) = \begin{cases} -\Delta_{ij}(t), & \text{если } \frac{\partial E}{\partial w_{ij}}(t) > 0, \\ \Delta_{ij}(t), & \text{если } \frac{\partial E}{\partial w_{ij}}(t) < 0, \\ 0, & \text{если } \frac{\partial E}{\partial w_{ij}}(t) = 0, \end{cases}$$

$$\Delta_{ij}(t) = \begin{cases} \eta^+ \Delta_{ij}(t-1), & \text{если } \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) > 0, \\ \eta^- \Delta_{ij}(t-1), & \text{если } \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) < 0, \\ \Delta_{ij}(t-1), & \text{если } \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) = 0, \end{cases}$$

где $0 < \eta^- < 1 < \eta^+$.

saturating linear transfer function – линейная функция активации с насыщением – функция активации, линейная на интервале (–1, +1) и постоянная вне его.

sequential input vectors – последовательные входные векторы – комплект векторов, которые должны представляться в сети «один после другого». Веса и смещения сети корректируются после представление каждого входного вектора.

sigmoid – сигмоид – монотонная *S*-образная функция, преобразующая значения аргумента из интервала (–1, +1) в значения функции из конечного интервала, например, (–1, +1) или (0, 1).

simulation – моделирование – процесс определения сетью выходного вектора при заданном входном.

spread constant – константа распространения – расстояние между входным и весовым векторами нейрона, при котором выход равен 0,5.

squashing function – сжимающая функция – монотонно возрастающая функция, преобразующая значения аргумента из интервала (–1, +1) в значения функции, образующие интервал конечной длины.

star learning rule – правило обучения звезды – входная звезда обучается реагировать на определенный входной вектор и ни на какой другой. Это обучение реализуется путем настройки весов таким образом, чтобы они соответствовали входному вектору. Выход звезды определяется как взвешенная сумма ее входов. С другой точки зрения, выход можно рассматривать как свертку входного вектора с весовым вектором. Следовательно, нейрон должен реагировать наиболее сильно на входной образ, которому был обучен.

sum-squared error – суммарная квадратическая ошибка – сумма квадратов ошибок сети для предъявленного набора входных векторов.

supervised learning – обучение с учителем – процесс обучения сети, предполагающий, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход; вместе они называются обучающей парой. Обычно сеть обучается на некотором множестве таких обучающих пар.

symmetric hard limit transfer function – симметричная пороговая передаточная (активационная) функция – равна +1 при неотрицательном аргументе и равна –1 при отрицательном.

symmetric saturating linear transfer function – линейная функция активации с симметричным ограничением (насыщением) – пропорционально преобразует значения аргумента из интервала (–1, +1) в значения функции в таком же интервале. При значениях аргумента вне данного интервала значения функции равны знаку аргумента (т. е. –1 или +1).

tan-sigmoid transfer function – функция активации типа гиперболического тангенса.

tapped delay line – набор звеньев задержки – последовательный набор задержек с выходами от каждой задержки.

target error – допустимая ошибка – параметр, задаваемый при обучении сети; обучение прекращается, когда средняя (реальная) ошибка сети становится меньше заданной допустимой. Обычно устанавливается на уровне 0,05.

target vector – целевой вектор – требуемый выходной вектор для заданного входного.

training – обучение – процедура, посредством которой сеть настраивается на решение конкретной задачи; заключается в подстройке по определенному алгоритму весов и смещений сети.

training data – исходные данные для обучения сети (набор входных и соответствующих им выходных векторов).

training vector – обучающий вектор – входной и/или выходной (целевой) вектор, используемый для обучения сети.

transfer function – передаточная (активационная) функция – функция, которая преобразует сумму взвешенных входов нейрона в его выход.

underdetermined system – недоопределенная система – система, которая имеет больше переменных, чем связывающих их ограничений.

unsupervised learning – обучение без учителя – процесс обучения сети, при котором изменения (настройки, коррекции) весов и смещений сети происходят не по предъявлении эталонных образцов, а автоматически, в зависимости от характеристик входных и выходных векторов сети. Обучающий алгоритм подстраивает веса сети так, чтобы получились согласованные выходные векторы, т. е. чтобы предъявление достаточно близких входных векторов давало близкие (или одинаковые) выходы.

update – коррекция – изменения в весах и смещениях сети в процессе ее обучения. Коррекция может произойти после представления единственного входного вектора или после предъявления и соответствующей обработки группы векторов.

weighted input vector – взвешенный входной вектор – результат умножения входного вектора для слоя нейронов на их веса.

weight matrix – матрица весов – матрица, составленная из весовых коэффициентов слоя нейронов. Элемент w_{ij} данной матрицы (матрицы $\boldsymbol{W}$) отображает вес связи от i-го входа к j-му нейрону.

Widrow–Hoff learning rule – Уидроу–Хоффа правило обучения – правило обучения сети с одним скрытым слоем. Оно является предшественником алгоритма обратного распространения ошибки и на него иногда ссылаются как на дельта–правило. Как известно, персептрон ограничивается бинарными выходами. Уидроу и Хофф расширили алгоритм обучения персептрона на случай непрерывных выходов, используя сигмоидальную функцию. Кроме того, они доказали того, что сеть при определенных условиях будет сходиться к любой функции, которую она может представить. Их первая модель – Адалин – имеет один выходной нейрон, более поздняя модель – Мадалин – представляет расширение на случай с многими выходными нейронами.

автоассоциативная сеть – сеть (обычно многослойный персептрон), предназначенная для воспроизведения на выходе входной информации после сжатия данных в промежуточном слое, имеющем меньшую размерность. Используется для сжатия информации и понижения размерности данных.

алгоритм *K* ближайших соседей – алгоритм выбора отклонений для радиальных элементов. Каждое отклонение равно усредненному расстоянию до *K* ближайших точек.

алгоритм *K* средних – алгоритм, предназначенный для выбора *K* центров, представляющих кластеры в *N* точках ($K < N$). Начиная со случайной выборки из *N* точек, расположение центров кластеров последовательно корректируется таким образом, чтобы каждая из *N* точек относилась ровно к одному из *K* кластеров, и центр каждого кластера совпадал с центром тяжести относящихся к нему точек.

алгоритмы минимизации функции – алгоритмы, используемые для поиска минимума, в частности, в нелинейном оценивании, при этом здесь минимизируется заданная функция потерь.

алгоритмы минимизации функций, свободные от производных – алгоритмы минимизации функций, использующие различные стратегии поиска (которые не зависят от производных второго порядка) для нелинейного оценивания. Эти стратегии наиболее эффективны при минимизации функции потерь, имеющей локальные минимумы.

байесовы сети – сети, чей принцип действия основан на теореме Байеса, позволяющей сделать выводы о распределении вероятностей на основании имеющихся данных.

быстрое распространение – эвристическая модификация алгоритма обратного распространения, где для ускорения сходимости применяется простая квадратичная модель поверхности ошибок (которая вычисляется отдельно для каждого веса).

вероятностные нейронные сети (PNN) – вид нейронных сетей для задач классификации, где плотность вероятности принадлежности классам оценивается посредством ядерной аппроксимации. Один из видов так называемых байесовых сетей.

встряхивание весов – добавление к весам нейронной сети небольших случайных величин с целью обойти локальные минимумы в пространстве ошибок.

выбросы – нетипичные или редкие значения, которые существенно отклоняются от распределения остальных выборочных данных. Эти данные могут отражать истинные свойства изучаемого явления (переменной), а могут быть связаны с ошибками измерения или аномальными явлениями, и поэтому не должны включаться в модель.

гауссово распределение – то же, что и нормальное распределение (с формой колокола).

генетический алгоритм – алгоритм поиска оптимальной битовой строки, который случайным образом выбирает начальную популяцию таких строк и затем подвергает их процессу искусственных мутаций, скрещивания и отбора по аналогии с естественным отбором.

генетический алгоритм отбора входных данных – применение генетического алгоритма к нахождению оптимального набора входных переменных путем построения битовых масок, обозначающих, какие из переменных следует оставить на входе, а какие удалить. Этот алгоритм может служить этапом построения модели, на котором отбираются наиболее существенные переменные; затем отобранные переменные используются для построения обычной аналитической модели (например, линейной регрессии или нелинейного оценивания).

гетероассоциативная сеть – сеть, в которой устанавливаются соответствия между произвольно выбранными входными и выходными векторами.

гиперболический тангенс (tanh) – симметричная функция с *S*-образной (сигмоидальной) формой графика; используется как альтернатива логистической функции.

гиперплоскость – *N*-мерный аналог прямой линии или плоскости, делит $(N+1)$-мерное пространство на две части.

гиперсфера – *N*-мерный аналог окружности или сферы.

горизонт (для нейронных сетей) – у нейронных сетей для анализа временных рядов – число шагов по времени, считая от последнего входного значения, на которое нужно спрогнозировать значения выходной переменной.

градиентный спуск – совокупность методов оптимизации нелинейных функционалов (например, функции ошибок нейронной сети, когда веса сети рассматриваются как аргументы функции), где с целью поиска минимума происходит последовательное продвижение во все более низкие точки в пространстве поиска.

два значения (для нейронных сетей) – способ кодирования значений номинальных переменных, принимающих только два значения, при котором номинальной переменной соответствует один входной или выходной элемент, который может быть активен или неактивен.

дельта – дельта с чертой (delta-bar-delta) – эвристическая модификация алгоритма обратного распространения для нейронных сетей, имеющая целью автоматическую коррекцию скорости обучения по каждой из координатных осей в пространстве поиска с тем, чтобы учесть особенности его топологии

диаграмма кластеров (для нейронных сетей) – точечная диаграмма, на которой наблюдения из разных классов представлены на плоскости. Координаты на плоскости соответствуют выходным уровням некоторых нейронов сети.

интерполяция – восстановление значения функции в промежуточной точке по известным ее значениям в соседних точках.

квадратическая функция ошибок – функция ошибок, равная сумме (взятой по всем наблюдениям) квадратов разностей требуемых и реальных значений.

квази-ньютоновский метод – процедура нелинейного оценивания, вычисляющая на каждом шаге значения функции в различных точках для оценивания первой и второй производной, и использующая эти данные для определения направления изменения параметров и минимизации функции потерь.

классификация – отнесение наблюдения к одному из нескольких, заранее известных классов (представленных значениями номинальной выходной переменной).

кодирование *N*-в-один (для нейронных сетей) – для номинальных переменных с числом значений, большим двух, – способ представления

переменной с помощью одного элементов сети через его различные выходные значения.

кодирование один-из-*N* (для нейронных сетей) – представление номинальной переменной с помощью набора входных или выходных элементов – по одному на каждое возможное номинальное значение. Во время обучения сети один из этих элементов бывает активен, а остальные – неактивны.

Кохонена обучение – алгоритм, размещающий центры кластеров радиального слоя посредством последовательной подачи на вход сети обучающих наблюдений и корректировки положения центра выигравшего (ближайшего) радиального элемента и соседних с ним в сторону обучающего наблюдения.

Кохонена сети – нейронные сети, основанные на воспроизведении топологических свойств человеческого мозга. Известны также как самоорганизующиеся карты признаков (SOFM).

кросс-проверка – процедура оценки точности прогнозирования с помощью данных из специальной тестовой выборки (используется также термин «кросс-проверочная выборка») путем сравнения точности прогноза с той, что достигается на обучающей выборке. В идеале, когда имеется достаточно большая выборка, часть наблюдений (например, половину или две трети) можно использовать для обучающей выборки, а оставшиеся наблюдения – для тестовой. Если на тестовой выборке модель дает результаты того же качества, что и на обучающей выборке, то говорят, что модель хорошо прошла кросс-проверку.

Для выполнения кросс-проверки при малых объемах выборки разработаны специальные методы, в которых тестовая и обучающая выборки могут частично пересекаться.

кросс-проверка (для нейронных сетей) – то же самое, что и вообще кросс-проверка. Применительно к нейронным сетям заключается в использовании во время итерационного обучения дополнительного множества данных (контрольного множества). В то время, как обучающее множество используется для корректировки весов сети, контрольное множество служит для независимой проверки того, как нейронная сеть научилась обобщать информацию.

кросс-энтропия (для нейронных сетей) – функция ошибок, основанная на теоретико-информационных характеристиках. Особенно хорошо подходит для задач классификации. Имеется два варианта: для сетей с одним выходом и для сетей с несколькими выходами. В первом варианте используются логистические функции активации, во втором – так называемые функции софтмакс.

Левенберга–Марквардта алгоритм – алгоритм нелинейной оптимизации, использующий для поиска минимума комбинированную стратегию – линейную аппроксимацию и градиентный спуск. Переключение с одной стратегии на другую происходит в зависимости от того, была ли успешной линейная аппроксимация. Такой подход называется моделью доверительных областей.

линейная функция активации – тождественная функция активации: выходной сигнал элемента совпадает с его уровнем активации.

линейное моделирование – аппроксимация дискриминантной или регрессионной функции с помощью гиперплоскости. Для этой гиперплоскости с помощью простых вычислений может быть найден глобальный оптимум. Однако таким образом нельзя построить адекватные модели для многих реальных задач.

линейные нейроны – нейроны, имеющие линейную постсинаптическую (PSP) функцию. Уровень активации такого нейрона представляет собой взвешенную сумму его входов, из которой вычитается пороговое значение (это называется также скалярным произведением или линейной комбинацией). Этот тип нейронов обычно используется в многослойных персептронах . Несмотря на название, линейные нейроны могут иметь нелинейные функции активации.

логистическая функция – функция с *S*-образной (сигмоидной) формой графика, принимающая значения из интервала (0, 1).

локальные минимумы – в большинстве практических приложений локальные минимумы функции потерь приводят к неправдоподобно большим или неправдоподобно малым значениям параметров с очень большими стандартными ошибками. Симплекс-метод нечувствителен к таким минимумам, поэтому он может быть использован для отыскания подходящих начальных значений для сложных функций.

матрица несоответствий (для нейронных сетей) – в задачах классификации так иногда называют матрицу, в которой для каждого класса наблюдений приводится количество наблюдений, отнесенных сетью к этому и другим классам.

матрица потерь – квадратная матрица, при умножении которой на вектор вероятностей принадлежности к классам получается вектор оценок потерь от ошибок классификации. На основе этого вектора можно принимать решения, приводящие к наименьшим потерям.

метод наименьших квадратов – общий смысл оценивания по методу наименьших квадратов заключается в минимизации суммы квадратов отклонений наблюдаемых значений зависимой переменной от значений, предсказанных моделью.

метод Розенброка – метод нелинейного оценивания, вращающий пространство параметров, располагая одну ось вдоль «гребня» поверхности (он называется также методом вращения координат – method of rotating coordinates), при этом все другие остаются ортогональными относительно выбранной оси. Если поверхность графика функции потерь имеет одну вершину и различимые «гребни» в направлении минимума этой функции, то данный метод приводит к очень точным значениям параметров, минимизирующим функцию потерь.

метод сопряженных градиентов – быстрый метод обучения многослойных персептронов, осуществляющий последовательный линейный поиск в пространстве ошибок. Последовательные направления поиска выбираются сопряженными (не противоречащими друг другу).

метод Хука–Дживса – метод нелинейного оценивания, который при каждой итерации сначала определяет схему расположения параметров, оптимизируя текущую функцию потерь перемещением каждого параметра по отдельности. При этом вся совокупность параметров сдвигается в новое положение. Это новое положение в m-мерном пространстве параметров определяется экстраполяцией вдоль линии, соединяющей текущую базовую точку с новой. Размер шага этого процесса постоянно меняется для попадания в оптимальную точку. Этот метод обычно очень эффективен, и его следует использовать в том случае, когда ни квазиньютоновский, ни симплекс-метод не дают удовлетворительных оценок.

минимакс – алгоритм определения коэффициентов линейного масштабирования для набора чисел. Находятся минимальное и максимальное значения, затем масштабирующие коэффициенты выбираются таким образом, чтобы преобразованный набор данных имел заранее заданные минимальное и максимальное значения.

многослойные персептроны – нейронные сети с прямой передачей сигнала, линейными PSP-функциями и, как правило, нелинейными функциями активации.

нейрон – элемент нейронной сети.

нейронные сети – класс аналитических методов, построенных на (гипотетических) принципах функционирования мозга и позволяющих прогнозировать значения некоторых переменных в новых наблюдениях по данным других наблюдений (для этих же или других переменных) после прохождения этапа так называемого обучения на имеющихся данных.

нелинейное оценивание – используется при неадекватности линейной модели путем добавления в уравнение модели нелинейных членов. В нелинейном оценивании выбор характера зависимости остается за исследователем. Например, можно определить зависимую переменную как логарифмическую функцию от предикторной переменной, как степенную функцию или как любую другую композицию элементарных функций от предикторов.

неуправляемое обучение или обучение без учителя (для нейронных сетей) – алгоритмы обучения, в которых на вход нейронной сети подаются данные, содержащие только значения входных переменных. Такие алгоритмы предназначены для нахождения кластеров во входных данных.

номинальные переменные – переменные, которые могут принимать конечное множество значений. В нейронных сетях номинальные выходные переменные используются в задачах классификации, в отличие от задач регрессии.

нормировка – корректировка длины вектора посредством некоторой суммирующей функции (например, на единичную длину или на единичную сумму компонент).

обобщение (для нейронных сетей) – способность нейронной сети делать точный прогноз на данных, не принадлежащих исходному обучающему множеству (но взятых из того же источника). Обычно это качество сети достигается разбиением имеющихся данных на три подмножества;

первое из них используется для обучения сети, второе – для кросс-проверки алгоритма обучения во время его работы, и третье – для окончательного независимого тестирования.

обобщенно-регрессионная нейронная сеть (GRNN) – вид нейронной сети, где для регрессии используются ядерная аппроксимация. Один из видов так называемых байесовых сетей.

обратное распространение (backpropagation learning rule) – алгоритм обучения многослойных персептронов. Надежный и хорошо известный, однако существенно более медленный по сравнению с некоторыми современными алгоритмами.

окрестность (для нейронных сетей) – в обучении по Кохонену – «квадрат», составленный из нейронов, окружающих «выигравший» нейрон, которые одновременно корректируются при обучении.

отдельное наблюдение (для нейронных сетей) – наблюдение, данные которого вводятся с клавиатуры и которые затем подаются на вход нейронной сети отдельно (а не как часть какого-то файла данных; в обучении такие наблюдения не участвуют).

отклика поверхность – поверхность, изображенная в трехмерном пространстве, представляющая отклик одной или нескольких переменных (в нейронной сети) в зависимости от двух входных переменных при постоянных остальных.

отклонение – в радиальных элементах – величина, на которую умножается квадрат расстояния от элемента до входного вектора, в результате чего получается аргумент, который затем пропускается через функцию активации элемента.

отношение стандартных отклонений – в задачах регрессии – отношение стандартного отклонения ошибки прогноза к стандартному отклонению исходных выходных данных. Чем меньше отношение, тем выше точность прогноза. Эта величина равна единице минус объясненная доля дисперсии модели.

перемешивание данных (для нейронных сетей) – случайное разбиение наблюдений на обучающее и контрольное множества, таким образом, чтобы они (насколько это возможно) получились статистически несмещенными.

перемешивание, обратное распространение (для нейронных сетей) – подача обучающих наблюдений на каждой эпохе в случайном порядке с целью предотвращения различных нежелательных эффектов, которые могут иметь место без этого приема (например, осцилляции и сходимость к локальным минимумам).

переобучение (для нейронных сетей) – при итерационном обучении – чрезмерно точная подгонка, которая имеет место, если алгоритм обучения работает слишком долго (а сеть слишком сложна для такой задачи или для имеющегося объема данных).

пост-синаптическая потенциальная функция (PSP-функция) – функция, которая применяется к входным сигналам нейрона, его весам и порогам и выдает уровень активации этого нейрона. Наиболее часто используются линейные (взвешенная сумма входов минус порог) и радиаль-

ные (промасштабированный квадрат расстояния от вектора весов до входного вектора) PSP-функции.

присоединение сети – действие, позволяющее сделать из двух нейронных сетей (совместимых по выходному и входному слоям) одну составную сеть.

промежуточные (скрытые) слои (для нейронных сетей) – все слои нейронной сети, кроме входного и выходного; придают сети способность моделировать нелинейные явления.

прямой передачи сети – нейронные сети с различной структурой слоев, в которых все соединения ведут только в от входов к выходам. Иногда используется как синоним для многослойных персептронов.

псевдо-обратных метод – эффективный метод оптимизации линейных моделей; известен также под названием «сингулярное разложение матрицы».

радиальные базисные функции – вид нейронной сети, имеющей промежуточный слой из радиальных нейронов и выходной слой из линейных элементов. Сети этого типа довольно компактны и быстро обучаются.

расстояние «городских кварталов» (расстояние Манхэттена) – это расстояние является средним разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного евклидова расстояния. Однако отметим, что для этой меры влияние отдельных больших разностей (выбросов) уменьшается (так как они не возводятся в квадрат).

расстояние Махаланобиса – независимые переменные в уравнении регрессии можно представлять точками в многомерном пространстве (каждое наблюдение – точка). В этом пространстве можно построить «средняя точка» – центроид, т. е. центр тяжести. Расстояние Махаланобиса определяется как расстояние от наблюдаемой точки до центра тяжести в многомерном пространстве, определяемом коррелированными (неортогональными) независимыми переменными. Эта мера позволяет, в частности, определить является ли данное наблюдение выбросом по отношению к остальным значениям независимых переменных. Если независимые переменные некоррелированы, то расстояние Махаланобиса совпадает с евклидовым расстоянием.

регрессия – категория задач, где цель состоит оценке значений непрерывной выходной переменной по значениям входных переменных.

регуляризация (для нейронных сетей) – модификация алгоритмов обучения, имеющая цель предотвратить пере- и недо-подгонку на обучающих данных за счет введения штрафа за сложность сети (обычно штрафуются большие значения весов – они означают, что отображение, моделируемое сетью, имеет большую кривизну).

самоорганизующиеся карты признаков (SOFMs, сети Кохонена) – нейронные сети, основанные на воспроизведении топологических свойств человеческого мозга, известны также как сети Кохонена.

сигмоидная функция – функция, график которой имеет *S*-образную форму, дающая приблизительно линейный отклик в середине входного диапазона и эффект насыщения на его концах.

симплекс-метод – метод нелинейного оценивания, не использующий производные функции потерь. Вместо этого, при каждой итерации функция оценивается в (m+1) точках m-мерного пространства. Например, на плоскости (т. е. при оценивании двух параметров) программа будет вычислять значение функции потерь в трех точках в окрестности текущего минимума. Эти три точки определяют треугольник; в многомерном пространстве получаемая фигура называется симплексом.

скорость обучения (для нейронных сетей) – управляющий параметр некоторых алгоритмов обучения, который контролирует величину шага при итерационной коррекции весов.

софтмакс – функция активации, предназначенная для классификационных сетей с кодированием по методу один-из-N. Вычисляет нормированную экспоненту (т. е. сумма выходов равна единице). В сочетании с кросс-энтропийной функцией ошибок позволяет модифицировать многослойный персептрон для оценки вероятностей принадлежности классам.

сохранение лучшей сети – возможность автоматически запоминать лучшую из сетей, обнаруженных в процессе обучения, с тем, чтобы по окончании экспериментов восстановить ее.

среднего/стандартного отклонения алгоритм (для нейронных сетей) – алгоритм для определения коэффициентов линейного масштабирования набора чисел. Находятся среднее значение и стандартное отклонение данных, затем масштабирующие коэффициенты выбираются таким образом, чтобы преобразованный набор данных имел заранее заданные значения среднего и стандартного отклонения.

среднее – показывает «центральное положение» переменной.

среднеквадратическая (RMS) ошибка – для вычисления среднеквадратической ошибки все отдельные ошибки возводятся в квадрат, суммируются, сумма делится на общее число ошибок, затем из всего извлекается квадратный корень. Полученное в результате число характеризует суммарную ошибку.

управляемое обучение или обучение с учителем (для нейронных сетей) – алгоритмы обучения, в которых на вход нейронной сети подаются данные, содержащие известные значения выходных переменных, а корректировка весов сети производится по результатам сравнения фактических выходных значений с требуемыми.

условия остановки – для итерационного процесса (подгонки, поиска, обучения) – условия, при выполнении которых процесс завершается. Например, для нейронных сетей условиями остановки могут быть: максимальное число эпох, целевое значение ошибки или порог ее минимального улучшения.

функция активации нейронной сети – функция, которая используется для преобразования уровня активации нейрона в выходной сигнал. Вместе с PSP-функцией (которая применяется сначала) определяет тип нейрона сети.

функция ошибок (для нейронных сетей) – служит для определения качества работы нейронной сети во время итерационного обучения и

последующих рабочих запусков. Градиент функции ошибок используется в формулах алгоритмов итерационного обучения.

функция ошибок городских кварталов (для нейронных сетей) – вычисляет расстояние между двумя векторами как сумму модулей разностей их компонент. Менее чувствительна к выбросам, чем квадратическая функция ошибок, но при этом обычно дает худшие результаты обучения.

функция потерь – функция, которая минимизируется в процессе подгонки модели. Она представляет выбранную меру несогласия наблюдаемых данных и данных, предсказываемых подогнанной функцией. Например, в большинстве традиционных методов построения общих линейных моделей, функция потерь (часто называемая наименьшими квадратами) вычисляется как сумма квадратов отклонений от подогнанной линии или плоскости. Одним из свойств (которое обычно рассматривается как недостаток) этой распространенной функции потерь является высокая чувствительность к наличию выбросов.

Распространенной альтернативой минимизации функции потерь наименьших квадратов (см. выше) является максимизация функции правдоподобия или логарифма функции правдоподобия (или минимизация функции правдоподобия со знаком минус). Эти функции обычно используются для подгонки нелинейных моделей.

частота выигрышей (для нейронных сетей) – для радиальных элементов сети Кохонена – число раз, когда нейрон выигрывал при прогоне файла данных. Часто, выигрывавшие нейроны представляют центры кластеров на топологической карте.

чрезмерно близкая подгонка – при восстановлении функции по набору ее значений – построение кривой с большой кривизной, которая хорошо удовлетворяет заданным значениям, но плохо моделирует исходное отображение, поскольку форма кривой искажена помехами, присутствующими в данных.

шум, добавление (для нейронных сетей) – способ предотвращения переобучения при обучении нейронной сети методом обратного распространения. Во время обучения к данным входных наблюдений добавляется случайный шум (в результате чего обучающие данные «смазываются»).

эвристика – в противоположность алгоритму (который описывает вполне определенный набор операций для получения конкретного результата), эвристики – это общие рекомендации, основанные на статистической очевидности или теоретических рассуждениях.

экстраполяция – прогнозирование неизвестных значений путем продолжения функций за границы области известных значений.

эпоха (для нейронных сетей) – в итерационном обучении нейронной сети – один проход по всему обучающему множеству с последующей проверкой на контрольном множестве.

ядерные функции – функции известного типа (как правило, гауссовы), которые размещаются в соответствии с известными данными, которые затем суммируются, и, таким образом, строится аппроксимация выборочного распределения.

Список литературы

1. **Анил К. Джейн, Жианчанг Мао, Моиуддин К. М.** Введение в искусственные нейронные сети //Открытые системы. – 1997. – № 4.
2. **Блинов С.** BrainMaker – прогнозирование на финансовых рынках// Открытые системы. 1998. № 4.
3. **Блинов С.** Практикум применения пакета BrainMaker для прогнозирования на финансовых рынках//http:// win.aha.ru/~mdo/office/bm_fin.htm.
4. **Блум Ф., Лейзерсон А., Хофстедтер Л.** Мозг, разум и поведение. – М.: Мир, 1988.
5. **Борисов В. В., Круглов В. В., Харитонов Е. В.** Основы построения нейронных сетей. – Смоленск: Изд-во Военного ун-та войсковой ПВО ВС РФ, 1999.
6. **Борисов Ю., Кашкаров В., Сорокин С.** Нейросетевые методы обработки информации и средства их программно-аппаратной поддержки// Открытые системы. – 1997. – № 4.
7. **Бэстенс Д.-Э., ван ден Берг В.-М., Вуд Д.** Нейронные сети и финансовые рынки: принятие решений в торговых операциях. – М.: ТВП, 1997.
8. **Васильев В. И.** Распознающие системы. – Киев: Наукова думка, 1988.
9. **Галушкин А. И.** Синтез многослойных систем распознавания образов. – М.: Энергия, 1974.
10. **Галушкин А. И.** Современные направления развития нейрокомпьютеров// Зарубежная радиоэлектроника. Успехи современной радиоэлектроники. – 1998. – № 1.
11. **Гелиг А. Х.** Динамика импульсных систем и нейронных сетей. – Л.: Изд-во ЛГУ, 1982.
12. **Горбань А. Н.** Обучение нейронных сетей. – М.: СП ПараГраф, 1991.
13. **Горбань А. Н., Россиев Д. А.** Нейронные сети на персональном компьютере. — Новосибирск: Наука, 1996.
14. **Горбань А. Н., Дунин-Барковский В. Л., Миркес Е. М. и др.** Нейроинформатика. – Новосибирск: Наука, 1998.
15. **Дли М. И., Круглов В. В., Осокин М. В.** Локально-аппроксимационные модели социально-экономических систем и процессов. – М.: Наука. Физматлит, 2000.
16. **Дуда Р., Харт П.** Распознавание образов и анализ сцен. – М.: Мир, 1976.
17. **Змитрович А. И.** Интеллектуальные информационные системы. – Минск: НТООО ТерраСистемс, 1997.
18. **Короткий С.** Нейронные сети: алгоритм обратного распространения// http:// www.neuropower.de/rus/books/index.html.
19. **Короткий С.** Нейронные сети: обучение без учителя// http://www.neuropower.de/rus/books/index.html.

20. **Короткий С.** Нейронные сети: основные положения// http://www.neuropower.de/rus/books/index.html.
21. **Короткий С.** Нейронные сети Хопфилда и Хэмминга// http://www.neuropower.de/rus/books/index.html.
22. **Круглов В. В., Борисов В. В., Харитонов Е. В.** Нейронные сети: конфигурации, обучение, применение. – Смоленск: Изд-во Моск. энерг. ин-та, фил-л в г. Смоленске, 1998.
23. **Курейчик В. М.** Генетические алгоритмы. Обзор и состояние// Новости искусственного интеллекта. 1998. – № 3.
24. **Логовский А. С.** Зарубежные нейропакеты: современное состояние и сравнительные характеристики// Нейрокомпьютер. – 1998. – № 1–2.
25. **Миркес Е. М.** Нейрокомпьютер. Проект стандарта. – Новосибирск: Наука, 1998.
26. **Нейрокомпьютеры** и интеллектуальные роботы/ Под ред. Н. М. Амосова. – Киев.: Наукова думка, 1991.
27. **Нечеткие** множества в моделях управления и искусственного интеллекта/ Под ред. Д. А. Поспелова. – М.: Наука, 1986.
28. **Огнев И. В., Борисов В. В.** Ассоциативные среды. – М.: Радио и связь, 2000.
29. **Перцептрон** – система распознавания образов/ Под ред. А. Г. Ивахненко. – Киев: Наукова думка, 1975.
30. **Попов Э. В., Фоминых И. Б., Кисель Е. Б., Шапот М. Д.** Статистические и динамические экспертные системы. – М.: Финансы и статистика, 1996.
31. **Прикладные** нечеткие системы/ Под ред. Т. Тэрано, К. Асаи, М. Сугено. – М.: Мир, 1993.
32. **Розенблатт Ф.** Принципы нейродинамики: Персептроны и теория механизмов мозга. – М.: Мир, 1965.
33. **Силов В. Б.** Принятие стратегических решений в нечеткой обстановке. М.: ИНПРО – РЕС, 1995.
34. **Соколов Е. Н., Вайткявичус Г. Г.** Нейроинтеллект: от нейрона к нейрокомпьютеру. – М.: Наука, 1989.
35. **Сотник С.** Курс лекций по предмету «Основы проектирования систем с искусственным интеллектом» //http://www.neuropower.de/rus/books/index.html.
36. **Куффлер С., Николс Дж.** От нейрона к мозгу. – М.: Мир, 1979.
37. **Уоссермен Ф.** Нейрокомпьютерная техника. – М.: Мир, 1992. – 240 с.
38. **Хехт-Нильсен Р.** Нейрокомпьютинг: история, состояние, перспективы// Открытые системы. 1998. № 4.
39. **Хьюбел Д.** Глаз, мозг, зрение. – М.: Мир, 1990.

ОГЛАВЛЕНИЕ

Часть II. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Часть III. ПРИМЕНЕНИЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

ПРИЛОЖЕНИЯ

ИПРЖР принимает заказы на готовящиеся к выпуску книги, объединенные серией с общим названием «Нейрокомпьютеры и их применение», под редакцией докт. техн. наук проф. А. И. Галушкина:

Теория нейронных сетей.
Нейроуправление и его приложения (Пер. с англ. под ред. А.И. Галушкина, В.А. Птичкина).
Нейрокомпьютеры.
История развития нейрокомпьютеров (Под ред. А.И. Галушкина и акад. Я. З. Цыпкина).
Сборник статей под ред. А.И. Галушкина. «Теория нейронных сетей».
Сборник статей «Нейроматематика» (Под ред. А. И. Галушкина).
Сборник статей «Нейрокомпьютеры 90-х» (под ред. А. И. Галушкина).
Сборник статей «Применение нейрокомпьютеров» (под ред. А.И. Галушкина).
Нейросетевые системы управления.
Нейроинтеллект: теория и применение.
Алгоритмы обучения нейронных сетей.
Теория модульных нейронных сетей.
Нейросетевые алгоритмы обработки изображений.
Применение нейрокомпьютеров.
Ассоциативная память. Нейросетевой подход.
Интеллектуальные нейросистемы. (Под научной ред. А. И. Галушкина).
Нейронные сети: систематизированное введение. (Пер. с англ; под ред. А. И. Галушкина, Г. Г. Губайдуллина, Ю. И Зозули).
Нейрокомпьютеры в вертолетах.
Нейроуправляемые конструкции и системыа.
Нейросетевое управление антропоморфными роботами и манипуляторами.
Нейросетевые алгоритмы управления манипуляторами.
Систематехническое проектирование высокопроизводительных цифровых нейрокомпьютеров с параллельной структурой.
Нейропроцессорные системы активного гашения промышленных вибраций.
Применение нейросетевых методов в информационных и аналитических системах.
Автоматизированное проектирование нейроадаптивных систем активного управления волновыми полями.
Нейрокомпьютеры в космической технике.
Нейрокомпьютеры в авиации.

Книги рекомендованы учебно методическим советом в качестве учебного пособия "Прикладные математика и физика" для студентов вузов по направлению 511600 подготовки бакалавров и магистров.Заявки на приобретение книг направляйте в ИПРЖР тел. 921-48-37 или по эл. почте iprzhr@online.ru