Глава 2

ОСНОВНЫЕ ПОЛОЖЕНИЯ ТЕОРИИ НЕЙРОННЫХ СЕТЕЙ

Под нейронными сетями (НС) подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга. Адаптируемые и обучаемые, они представляют собой распараллеленные системы, способные к 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.

Термин «нейронные сети» сформировался в 40-х годах XX века в среде исследователей, изучавших принципы организации и функционирования биологических нейронных сетей. Основные результаты, полученные в этой области, связаны с именами американских исследователей У. Маккалоха, Д. Хебба, Ф. Розенблатта, М. Минского, Дж. Хопфилда и др.

Представим некоторые проблемы, решаемые в контексте НС и представляющие интерес для пользователей.

Классификация образов. Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), 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 приме-

нения кластеризации для извлечения знаний, сжатия данных и исследования свойств данных.

Аппроксимация функций. Предположим, что имеется обучающая выборка $\big((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\big)$ (пары данных вход-выход), которая генерируется неизвестной функцией $F(x)$, искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной функции $F(x)$. Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

Предсказание/прогноз. Пусть заданы n дискретных отсчетов $\{y(t_1), y(t_2), \ldots, y(t_k)\}$ в последовательные моменты времени $t_1, t_2, \ldots, t_k$. Задача состоит в предсказании значения $y(t_{k+1})$ в некоторый будущий момент времени t_{k+1}. Предсказание/прогноз имеет значительное влияние на принятие решений в бизнесе, науке и технике. Предсказание цен на фондовой бирже и прогноз погоды являются типичными приложениями техники предсказания/прогноза.

Оптимизация. Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию. Известная задача коммивояжера является классическим примером задачи оптимизации.

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. Ассоциативная память, или память, адресуемая по содержанию, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному входу или искаженному содержанию. Ассоциативная память чрезвычайно желательна при создании мультимедийных информационных баз данных.

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. Примером является оптимальное управление двигателем.

2.1. 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 нашей кожи, ушей и глаз к мозгу, процессы мышления и управления действиями — все это реализовано в живом организме как передача электрических импульсов между нейронами.

Нейрон (нервная клетка) является особой биологической клеткой, которая обрабатывает информацию (рис. 2.1). Он состоит из тела и отростков нервных волокон двух типов — дендритов, по

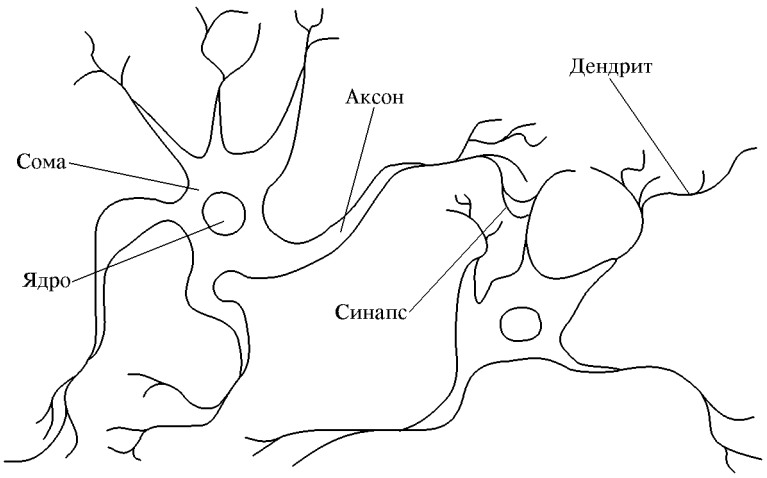

Рис. 2.1. Взаимосвязь биологических нейронов

которым принимаются импульсы, и единственного аксона, по которому нейрон может передавать импульс. Тело нейрона включает ядро, 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 (передатчик), который в конце разветвляется на волокна. На окончаниях этих волокон находятся специальные образования — синапсы, которые влияют на силу импульса.

Синапс является элементарной структурой и функциональным узлом между двумя нейронами (волокно аксона одного нейрона и

дендрит другого). Когда импульс достигает синаптического окончания, высвобождаются определенные химические вещества, называемые нейротрансмиттерами. Нейротрансмиттеры диффундируют через синаптическую щель, возбуждая или затормаживая, в зависимости от типа синапса, способность нейрона-приемника генерировать электрические импульсы. Результативность синапса 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, что изменяет и поведение соответствующего нейрона.

2.2. Структура и свойства искусственного нейрона

Нейрон — это составная часть нейронной сети. На рис. 2.2 показана его структура.

В состав нейрона входят умножители (синапсы), сумматор и нелинейный преобразователь. Синапсы осуществляют связь между нейронами и умножают входной сигнал на число, характеризую-

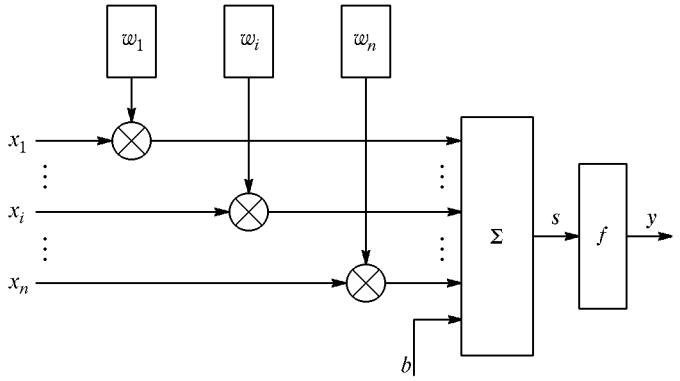

Рис. 2.2. Структура искусственного нейрона

щее 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 «функция активации» или «передаточная функция» нейрона. Нейрон в целом реализует скалярную функцию векторного аргумента. Математическая модель

нейрона описывается соотношениями

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

где w_i — вес синапса ($i = 1, \ldots, n$); b — значение смещения; s — результат суммирования; x_i — компонента входного вектора (входной сигнал) ($i = 1, \ldots, n$); y — выходной сигнал нейрона; n — число входов нейрона; f — нелинейное преобразование (функция активации или передаточная функция).

Таблица 2.1. **Перечень функций активации нейронов**

Название	Формула	Область значений
Пороговая	$f(s) = \begin{cases} 0, & s < \theta, \\ 1, & s \geqslant \theta \end{cases}$	0, 1
Знаковая (сигнатурная)	$f(s) = \begin{cases} 1, & s > 0, \\ -1, & s \leqslant 0 \end{cases}$	-1, 1
Сигмоидальная (логистическая)	$f(s) = \dfrac{1}{1 + e^{-s}}$	(0, 1)
Полулинейная	$f(s) = \begin{cases} s, & s > 0, \\ 0, & s \leqslant 0 \end{cases}$	$(0, \infty)$
Линейная	$f(s) = s$	$(-\infty, \infty)$
Радиальная базисная (гауссова)	$f(s) = e^{-s^2}$	(0, 1)
Полулинейная с насыщением	$f(s) = \begin{cases} 0, & s \leqslant 0, \\ s, & 0 < s < 1, \\ 1, & s \geqslant 1 \end{cases}$	(0, 1)
Линейная с насыщением	$f(s) = \begin{cases} -1, & s \leqslant -1, \\ s, & -1 < s < 1, \\ 1, & s \geqslant 1 \end{cases}$	$(-1, 1)$
Гиперболический тангенс (сигмоидальная)	$f(s) = \dfrac{e^s - e^{-s}}{e^s + e^{-s}}$	$(-1, 1)$
Треугольная	$f(s) = \begin{cases} 1 - \lvert s \rvert, & \lvert s \rvert \leqslant 1, \\ 0, & \lvert s \rvert > 1 \end{cases}$	(0, 1)

В общем случае входной сигнал, весовые коэффициенты и значения смещения могут принимать действительные значения. Выход (y) определяется видом функции активации и может быть как

действительным, так и целым. Во многих практических задачах входы, веса и смещения могут принимать лишь некоторые фиксированные значения.

Синаптические связи с положительными весами называют возбуждающими, с отрицательными весами — тормозящими.

Таким образом, нейрон полностью описывается своими весами w_i и передаточной функцией $f(s)$. Получив набор чисел (вектор) x_i в качестве входов, нейрон выдает некоторое число y на выходе.

Описанный вычислительный элемент можно считать упрощенной математической моделью биологических нейронов — клеток, из которых состоит нервная система человека и животных.

Чтобы подчеркнуть различие нейронов биологических и математических, вторые иногда называют нейроноподобными элементами или формальными нейронами.

На входной сигнал (s) нелинейный преобразователь отвечает выходным сигналом $f(s)$, который представляет собой выход нейрона y. Примеры активационных функций представлены в табл. 2.1 и на рис. 2.3.

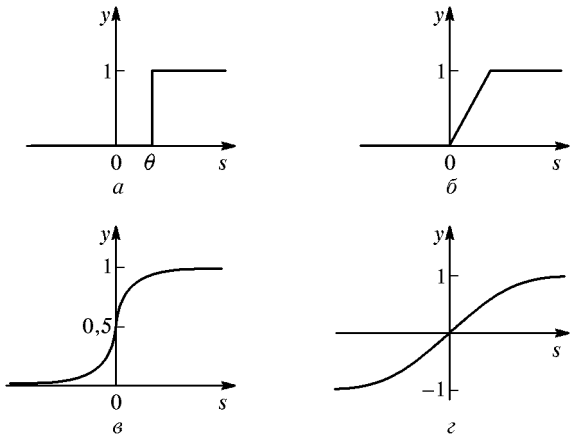

Рис. 2.3. Примеры активационных функций: *а* — функция единичного скачка; *б* — линейный порог (гистерезис); *в* — сигмоид (гиперболический тангенс); *г* — сигмоид (логистическая)

Одной из наиболее распространенных является нелинейная функция с насыщением, так называемая логистическая функция или сигмоид (т. е. функция S-образного вида)

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

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 по внешнему виду к функции единичного скачка с порогом θ в точке $s = 0$. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне $[0, 1]$. Одно из ценных свойств сигмоидной функции — простое выражение для ее производной, применение которого будет рассмотрено в дальнейшем:

$$f'(s) = af(s)\big(1 - f(s)\big).$$

Следует отметить, что сигмоидная 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.

Возвращаясь к общим чертам, присущим всем НС, отметим принцип параллельной обработки сигналов, который достигается путем объединения большого числа нейронов в так называемые слои и соединения определенным образом нейронов различных слоев, а также, в некоторых конфигурациях, и нейронов одного слоя между собой, причем обработка взаимодействия всех нейронов ведется послойно.

2.3. Классификация нейронных сетей и их свойства

Как отмечалось, искусственная нейронная сеть (ИНС, нейросеть) — это набор нейронов, соединенных между собой. Как правило, передаточные (активационные) функции всех нейронов в сети фиксированы, а веса являются параметрами сети и могут изменяться. Некоторые входы нейронов помечены как внешние входы сети, а некоторые выходы — как внешние выходы сети. Подавая любые числа на входы сети, мы получаем какой-то набор чисел на выходах сети. Таким образом, работа нейросети состоит в преобразовании входного вектора $\mathbf{X}$ в выходной вектор $\mathbf{Y}$, причем это преобразование задается весами сети.

Практически любую задачу можно свести к задаче, решаемой нейросетью. В табл. 2.2 показано, каким образом следует сформулировать в терминах нейросети задачу распознавания рукописных букв.

Поясним, зачем требуется выбирать выход с максимальным уровнем сигнала. Дело в том, что уровень выходного сигнала, как правило, может принимать любые значения из какого-то отрезка.

Однако в данной задаче нас интересует не аналоговый ответ, а всего лишь номер категории (номер буквы в алфавите). Поэтому используется следующий подход — каждой категории сопоставляется свой выход, а ответом сети считается та категория, на чьем выходе уровень сигнала максимален. В определенном смысле уровень сигнала на выходе «А» — это достоверность того, что на вход была подана рукописная буква «А». Задачи, в которых нужно отнести входные данные к одной из известных категорий, называются задачами классификации. Изложенный подход — стандартный способ классификации с помощью нейронных сетей.

Таблица 2.2. **Задача распознавания рукописных букв в терминах нейросети**

Задача распознавания рукописных букв	
Дано: растровое черно-белое изображение буквы размером 30×30 пикселов	Надо: определить, какая это буква (в алфавите 33 буквы)
Формулировка для нейросети	
Дано: входной вектор из 900 двоичных символов ($900 = 30 \times 30$)	Надо: построить нейросеть с 900 входами и 33 выходами, которые помечены буквами. Если на входе сети — изображение буквы «А», то максимальное значение выходного сигнала достигается на выходе «А». Аналогично сеть работает для всех 33 букв

Теперь, когда стало ясно, что́ именно мы хотим построить, мы можем переходить к вопросу «как строить такую сеть». Этот вопрос решается в два этапа.

1. Выбор типа (архитектуры) сети.
2. Подбор весов (обучение) сети.

На первом этапе следует выбрать следующее:

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

Эта задача на первый взгляд кажется необозримой, но, к счастью, нам необязательно придумывать нейросеть «с нуля» — существует несколько десятков различных нейросетевых архитектур, причем эффективность многих из них доказана математически. Наиболее популярные и изученные архитектуры — это многослойный персептрон, нейросеть с общей регрессией, сети Кохонена и другие, которые будут рассмотрены ниже.

На втором этапе нам следует «обучить» выбранную сеть, т. е. подобрать такие значения ее весов, чтобы сеть работала нужным образом. Необученная сеть подобна ребенку — ее можно научить чему угодно. В используемых на практике нейросетях количество весов может составлять несколько десятков тысяч, поэтому обучение — действительно сложный процесс. Для многих архитектур разработаны специальные алгоритмы обучения, которые позволяют настроить веса сети определенным образом.

В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

- входные нейроны — это нейроны, на которые подается входной вектор, кодирующий входное воздействие или образ внешней среды; в них обычно не осуществляется вычислительных процедур, информация передается с входа на выход нейрона путем изменения его активации;
- выходные нейроны — это нейроны, выходные значения которых представляют выход сети;
- промежуточные нейроны — это нейроны, составляющие основу искусственных нейронных сетей.

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 может встретиться случай, когда выход топологически внутреннего нейрона рассматривается как часть выхода сети. В процессе функционирования (эволюции состояния) сети осуществляется преобразование входного вектора в выходной, т. е. некоторая переработка информации. Конкретный вид выполняемого сетью преобразования информации обусловливается не только характеристиками нейроподобных элементов (функциями активации и т. п.), но и особенностями ее архитектуры, а именно, той или иной 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.

Топология нейронных сетей. С точки зрения топологии, среди нейронных сетей, сформированных на основе нейроподобных элементов, можно выделить три основных типа:

- полносвязные сети (рис. 2.4*а*);
- многослойные или слоистые сети (рис. 2.4*б*);
- слабосвязные сети (нейронные сети с локальными связями) (рис. 2.4*в*).

Полносвязные сети представляют собой ИНС, каждый нейрон которой 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.

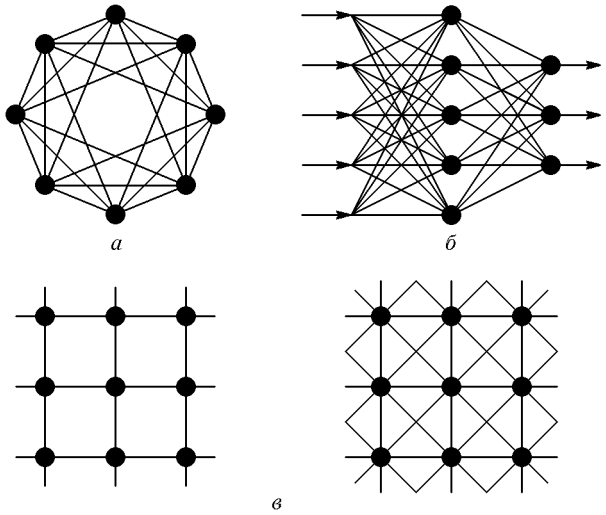

Рис. 2.4. Архитектуры нейронных сетей: *а* — полносвязная сеть; *б* — многослойная сеть с последовательными связями; *в* — слабосвязные сети

В многослойных сетях нейроны объединяются в слои. Слой содержит совокупность нейронов с едиными входными сигналами. Число нейронов в каждом слое может быть любым и никак заранее не связано с количеством нейронов в других слоях. В общем случае сеть состоит из Q слоев, пронумерованных слева направо. Внешние входные сигналы подаются на входы нейронов первого слоя (входной слой часто нумеруют как нулевой), а выходами сети являются выходные сигналы последнего слоя. Вход нейронной сети можно рассматривать как выход «нулевого слоя» вырожденных нейронов, которые служат лишь в качестве распределительных точек, суммирования и преобразования сигналов здесь не производится. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько промежуточных (скрытых) слоев. Связи от выходов нейронов некоторого слоя q к входам нейронов следующего слоя $(q+1)$ называются последовательными.

В свою очередь, среди слоистых сетей выделяют следующие типы.

1. *Монотонные.* Это специальный частный случай слоистых сетей с дополнительными условиями на связи и элементы. Каждый слой, кроме последнего (выходного), разбит на два блока: возбуждающий (В) и тормозящий (Т). Связи между блоками тоже разделяются на тормозящие и возбуждающие. Если от блока А к блоку С ведут только возбуждающие связи, то это означает, что любой выходной сигнал блока является монотонной неубывающей функцией любого выходного сигнала блока А. Если же эти связи только тормозящие, то любой выходной сигнал блока С является невозрастающей функцией любого выходного сигнала блока А. Для элементов монотонных сетей необходима монотонная зависимость выходного сигнала элемента от параметров входных сигналов.

2. *Сети без обратных связей.* В таких сетях нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам 1-го скрытого слоя, далее срабатывает 1-й скрытый слой и т. д. до Q-го, который выдает выходные сигналы для интерпретатора и пользователя. Если не оговорено противное, то каждый

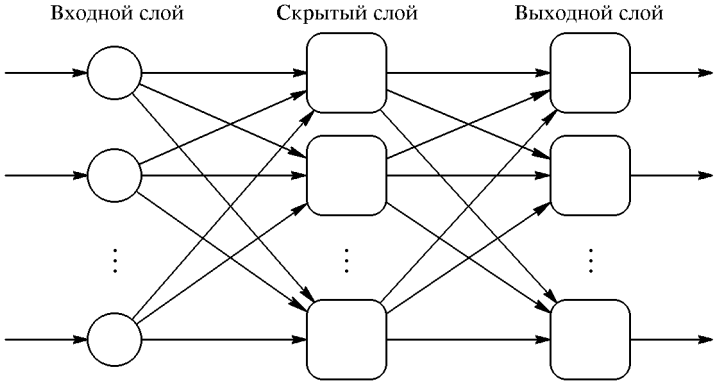

Рис. 2.5. Многослойная (двухслойная) сеть прямого распространения

выходной сигнал i-го слоя подастся на вход всех нейронов $(q+l)$-го слоя; однако возможен вариант соединения q-го слоя с произвольным $(q+p)$-м слоем.

Следует отметить, что классическим вариантом слоистых сетей являются сети прямого распространения (рис. 2.5).

3. *Сети с обратными связями.* Это сети, у которых информация с последующих слоев передается на предыдущие.

В качестве примера сетей с обратными связями на рис. 2.6 представлены так называемые частично-рекуррентные сети Элмана и Жордана.

Известные сети можно разделить по принципу структуры нейронов в них на гомогенные или однородные и гетерогенные. Гомогенные сети состоят из нейронов одного типа с единой функцией

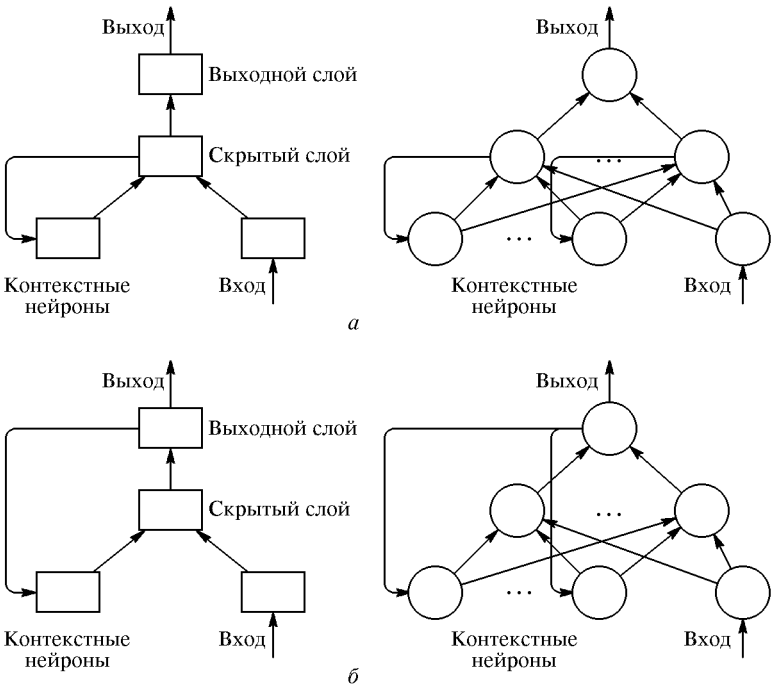

Рис. 2.6. Частично-рекуррентные сети: *а* — Элмана; *б* — Жордана

активации. В гетерогенную сеть входят нейроны с различными функциями активации.

Развивая дальше вопрос о возможной классификации НС, важно отметить существование бинарных и аналоговых сетей. Первые из них оперируют с двоичными сигналами, и выход каждого нейрона может принимать только два значения: логический ноль («заторможенное» состояние) и логическая единица («возбужденное» состояние).

Еще одна классификация делит НС на асинхронные и синхронные. В первом случае в каждый момент времени свое состояние

меняет лишь один нейрон. Во втором — состояние меняется сразу у целой группы нейронов, как правило, у всего слоя. Алгоритмически ход времени в НС задается итерационным выполнением однотипных действий над нейронами. Далее будут рассматриваться только синхронные НС.

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 специализированной микросхемы, на которых обычно реализуется ИНС. Чем сложнее ИНС, тем масштабнее задачи, подвластные ей.

Выбор структуры ИНС осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные, на сегодняшний день, конфигурации, описанные в приложении. Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации. При этом он руководствуется несколькими основополагающими принципами:

- возможности сети возрастают с увеличением числа слоев сети и числа нейронов в них;
- введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о так называемой динамической устойчивости сети;
- сложность алгоритмов функционирования сети (в том числе, например, введение нескольких типов синапсов — возбуждающих, тормозящих и др.) также способствует усилению мощи ИНС.

Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки. Так как проблема синтеза ИНС 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 и т.д.), выполнения функциональных преобразований при обработке сигналов, управления, прогнозирования, идентификации сложных систем и т.д., сводятся к следующей математической постановке. Необходимо построить отображение $\mathbf{X} \to \mathbf{Y}$ такое, чтобы на каждый возможный входной сигнал $\mathbf{X}$ формировался правильный выходной сигнал $\mathbf{Y}$. Отображение задается конечным

набором пар (⟨вход⟩, ⟨известный выход⟩). Число таких 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 $\mathbf{X}$ — некоторое представление образа (изображение, вектор чисел и т. д.), $\mathbf{Y}$ — номер класса, к которому принадлежит входной образ.

В задачах управления $\mathbf{X}$ — набор контролируемых параметров управляемого объекта, $\mathbf{Y}$ — 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 $\mathbf{X}$ и $\mathbf{Y}$ представляют входные и выходные сигналы системы соответственно.

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

В результате построения такого отображения (т.е. $\mathbf{X} \to \mathbf{Y}$) необходимо добиться того, чтобы:

- обеспечивалось формирование правильных выходных сигналов в соответствии со всеми примерами обучающей выборки;
- обеспечивалось формирование правильных выходных сигналов в соответствии 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 было найдено частное решение. Дальнейшие рассуждения предполагают, что обучающая выборка уже сформирована.

Отметим, что теоретической основой для построения нейронных сетей является следующее

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

Таким образом, для представления многомерных функций многих переменных может быть использована однородная нейронная сеть, имеющая всего один скрытый слой, с сигмоидальными передаточными функциями нейронов.

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

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

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, число нейронов в двухслойной сети составит:

$$L = \frac{L_w}{n+m}.$$

Известны и другие подобные формулы, например, вида

$$2(L+n+m) \leqslant N \leqslant 10(L+n+m),$$

$$\frac{N}{10} - n - m \leqslant L \leqslant \frac{N}{2} - n - m.$$

Точно так же можно рассчитать число нейронов в сетях с большим числом слоев, которые иногда целесообразно использовать: такие многослойные нейронные сети могут иметь меньшие размерности матриц синаптических весов нейронов одного слоя, чем двухслойные сети, реализующие то же самое отображение. К сожалению, строгая методика построения данных сетей пока отсутствует.

Отметим, что отечественному читателю приведенные результаты обычно известны 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.*

Таким образом, нейронные сети являются универсальными аппроксимирующими системами.

Очевидно, что процесс функционирования ИНС, т.е. сущность действий, которые она способна выполнять, зависит от величин

синаптических связей, поэтому, задавшись определенной структурой ИНС, отвечающей какой-либо задаче, разработчик сети должен найти оптимальные значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными).

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

2.4. Обучение нейронных сетей

Обучить нейросеть — значит, сообщить ей, чего мы от нее добиваемся. Этот процесс очень похож на обучение ребенка алфавиту. Показав ребенку изображение буквы «А», мы спрашиваем его: «Какая это буква?» Если ответ неверен, мы сообщаем ребенку тот ответ, который мы хотели бы от него получить: «Это буква А». Ребенок запоминает этот пример вместе с верным ответом, т. е. в его памяти происходят некоторые изменения в нужном направлении. Мы будем повторять процесс предъявления букв снова и снова до тех пор, когда все буквы будут твердо запомнены. Такой процесс называют «обучение с учителем» (рис. 2.7).

При обучении сети мы действуем совершенно аналогично. У нас имеется некоторая база данных, содержащая примеры (набор ру-

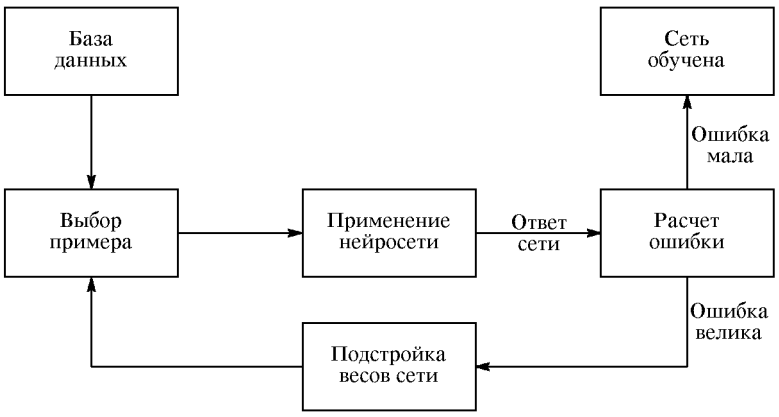

Рис. 2.7. Иллюстрация процесса обучения НС

кописных изображений букв). Предъявляя изображение буквы «А» на вход сети, мы получаем от нее некоторый ответ, не обя-

зательно верный. Нам известен и верный (желаемый) ответ — в данном случае нам хотелось бы, чтобы на выходе с меткой «А» уровень сигнала был максимален. Обычно в качестве желаемого выхода в задаче классификации берут набор (1, 0, 0, ...), где 1 стоит на выходе с меткой «А», а 0 — на всех остальных выходах. Вычисляя разность между желаемым ответом и реальным ответом сети, мы получаем (для букв русского алфавита) 33 числа — вектор ошибки. Алгоритм обучения — это набор формул, который позволяет по вектору ошибки вычислить требуемые поправки для весов сети. Одну и ту же букву (а также различные изображения одной и той же буквы) мы можем предъявлять сети много раз. В этом смысле обучение скорее напоминает повторение упражнений в спорте — тренировку.

Оказывается, что 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 выучила все примеры», «сеть обучена», или «сеть натренирована». В программных реализациях можно видеть, что в процессе обучения функция ошибки (например, сумма квадратов ошибок по всем выходам) постепенно уменьшается. Когда функция ошибки достигает нуля или приемлемого малого уровня, тренировку останавливают, а полученную сеть считают натренированной и готовой к применению на новых данных.

Важно отметить, что вся информация, которую сеть имеет о задаче, содержится в наборе примеров. Поэтому качество обучения сети напрямую зависит от количества примеров в обучающей выборке, а также от того, насколько полно эти примеры описывают данную задачу. Так, например, бессмысленно использовать сеть для предсказания финансового кризиса, если в обучающей выборке кризисов не представлено. Считается, что для полноценной тренировки требуется хотя бы несколько десятков (а лучше сотен) примеров.

Математически процесс обучения можно описать следующим образом.

В процессе функционирования нейронная сеть формирует выходной сигнал $\mathbf{Y}$ в соответствии с входным сигналом $\mathbf{X}$, реализуя некоторую функцию $\mathbf{Y} = \mathrm{G}(\mathbf{X})$. Если архитектура сети задана, то вид функции G определяется значениями синаптических весов и смещений сети.

Пусть решением некоторой задачи является функция $\mathbf{Y} = \mathrm{F}(\mathbf{X})$, заданная парами входных-выходных данных $(\mathbf{X}^1, \mathbf{Y}^1)$, $(\mathbf{X}^2, \mathbf{Y}^2)$, ..., $(\mathbf{X}^N, \mathbf{Y}^N)$, для которых $\mathbf{Y}^k = \mathrm{F}(\mathbf{X}^k)$ $(k = 1, 2, \ldots, N)$.

Обучение состоит в поиске (синтезе) функции G, близкой к F в смысле некоторой функции ошибки E (см. рис. 2.7).

Если выбраны множество обучающих примеров — пар $(\mathbf{X}^k, \mathbf{Y}^k)$ (где $k = 1, 2, \ldots, N$) и способ вычисления функции ошибки E, то обучение нейронной сети превращается в задачу многомерной оптимизации, имеющую очень большую размерность, при этом, поскольку функция E может иметь произвольный вид, обучение в общем случае — многоэкстремальная невыпуклая задача оптимизации.

Для решения этой задачи могут быть использованы следующие (итерационные) алгоритмы:

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

К первой группе относятся: градиентный алгоритм (метод скорейшего 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–Марквардта и др.

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 (функция ошибки E).

2.4.1. Алгоритм обратного распространения. Рассмотрим идею одного из самых распространенных алгоритмов обучения — алгоритм обратного распространения ошибки (back propagation). Это итеративный градиентный алгоритм обучения, который используется с целью минимизации среднеквадратичного отклонения текущего выхода и желаемого выхода многослойных нейронных сетей.

Алгоритм обратного распространения используется для обучения многослойных нейронных сетей с последовательными связями вида рис. 2.5. Как отмечено выше, нейроны в таких сетях делятся на группы с общим входным сигналом — слои, при этом на каждый нейрон первого слоя подаются все элементы внешнего

входного сигнала, а все выходы нейронов m-го слоя подаются на каждый нейрон слоя $(q+1)$. Нейроны выполняют взвешенное (с синаптическими весами) суммирование элементов входных сигналов; к данной сумме прибавляется смещение нейрона. Над полученным результатом выполняется активационной функцией затем нелинейное преобразование. Значение функции активации есть выход нейрона.

В многослойных сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, неизвестны, и трех- или более слойный персептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС. Наиболее приемлемым вариантом обучения в таких условиях оказался градиентный метод поиска минимума функции ошибки с рассмотрением сигналов ошибки от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название процедуры обратного распространения.

В данном алгоритме функция ошибки представляет собой сумму квадратов рассогласования (ошибки) желаемого выхода сети и реального. При вычислении элементов вектора-градиента использован своеобразный вид производных функций активации сигмоидального типа. Алгоритм действует циклически (итеративно), и его циклы принято называть эпохами. На каждой эпохе на вход сети поочередно подаются все обучающие наблюдения, выходные значения сети сравниваются с целевыми значениями и вычисляется ошибка. Значение ошибки, а также градиента поверхности ошибок используется для корректировки весов, после чего все действия повторяются. Начальная конфигурация сети выбирается случайным образом, и процесс обучения прекращается, либо когда пройдено определенное количество эпох, либо когда ошибка достигнет некоторого определенного уровня малости, либо когда ошибка перестанет уменьшаться (пользователь может сам выбрать нужное условие остановки).

Приведем словесное описание алгоритма.

Шаг 1. Весам сети присваиваются небольшие начальные значения.

Шаг 2. Выбирается очередная обучающая пара $(\mathbf{X}, \mathbf{Y})$ из обучающего множества; вектор $\mathbf{X}$ подается на вход сети.

Шаг 3. Вычисляется выход сети.

Шаг 4. Вычисляется разность между требуемым (целевым, $\mathbf{Y}$) и реальным (вычисленным) выходом сети.

Шаг 5. Веса сети корректируются так, чтобы минимизировать ошибку.

Шаг 6. Шаги со 2-го по 5-й повторяются для каждой пары обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемой величины.

Шаги 2 и 3 подобны тем, которые выполняются в уже обученной сети.

Вычисления в сети выполняются послойно. На шаге 3 каждый из выходов сети вычитается из соответствующей компоненты целевого вектора с целью получения ошибки. Эта ошибка используется на шаге 5 для коррекции весов сети.

Шаги 2 и 3 можно рассматривать как «проход вперед», так как сигнал распространяется по сети от входа к выходу. Шаги 4 и 5 составляют «обратный проход», поскольку здесь вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов.

Математическое представление алгоритма рассмотрим вначале на примере простейшей нейронной сети, содержащей только один нейрон (условно показанный кружком, см. рис. 2.8).

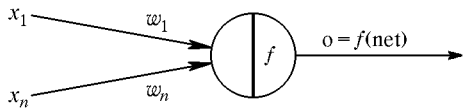

Рис. 2.8. НС из одного нейрона

Будем полагать, что выход (output) сети $\mathrm{o} = f(\mathrm{net})$ определяется функцией активации сигмоидного (см. табл. 2.1) типа

$$\mathrm{o} = \mathrm{o}(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}},$$

где $\mathbf{x}^\top = (x_1, \ldots, x_n)$ — вектор входных сигналов, $\mathbf{w}^\top = (w_1, \ldots, w_n)$ — вектор весов сети, «$\top$» — символ транспонирования.

Предположим, далее, что для обучения сети используется выборка

$$\begin{aligned}
&\mathbf{x}^1 = (x_1^1, x_2^1, \ldots, x_n^1)^\top, \quad y^1,\\
&\mathbf{x}^2 = (x_1^2, x_2^2, \ldots, x_n^2)^\top, \quad y^2,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\mathbf{x}^N = (x_1^N, x_2^N, \ldots, x_n^N)^\top, \quad y^N,
\end{aligned}$$

где y^k — значения желаемого (целевого) выхода.

В качестве функции ошибки для k-го образца (k-го элемента обучающей выборки) примем величину, пропорциональную ква-

драту разности желаемого выхода и выхода сети:

$$E_k = \frac{1}{2}(y^k - \mathrm{o}^k)^2 = \frac{1}{2}\left(y^k - \mathrm{o}^k(\mathbf{w}^\top \mathbf{x}^k)\right)^2 = \frac{1}{2}\left(y^k - \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x}^k}}\right)^2.$$

Соответственно, суммарная функция ошибки по всем элементам выборки:

$$E = \sum_{k=1}^{N} E_k.$$

Очевидно, как E_k, так и E являются функциями вектора весов сети $\mathbf{w}$. Задача обучения сети сводится в данном случае к подбору такого вектора $\mathbf{w}$, при котором достигается минимум E. Данную задачу (оптимизации) будем решать градиентным методом, используя соотношение

$$\mathbf{w} := \mathbf{w} - \eta E'_k(\mathbf{w}),$$

где «:=» — оператор присвоения, $E'_k(\mathbf{w})$ — обозначение вектора-градиента, η — некоторая константа.

Представляя данный вектор в развернутом виде и учитывая приведенное выше (в § 2.2) выражение для производной сигмоидной функции, получим:

$$E'_k(\mathbf{w}) = \frac{d}{d\mathbf{w}}\left(\frac{1}{2}\left(y^k - \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x}^k}}\right)^2\right) = -(y^k - \mathrm{o}^k)\mathrm{o}^k(1-\mathrm{o}^k)\mathbf{x}^k.$$

Это дает возможность записать алгоритм коррекции (подстройки) вектора весовых коэффициентов сети в форме

$$\mathbf{w} := \mathbf{w} + \eta(y^k - \mathrm{o}^k)\mathrm{o}^k(1-\mathrm{o}^k)\mathbf{x}^k = \mathbf{w} + \eta\delta_k\mathbf{x}^k,$$

где

$$\delta_k = (y^k - \mathrm{o}^k)\mathrm{o}^k(1-\mathrm{o}^k).$$

Полученные математические выражения полностью определяют алгоритм обучения рассматриваемой НС, который может быть представлен теперь в следующем виде.

1. Задаются некоторые η $(0 < \eta < 1)$, $E_{\max}$ и некоторые малые случайные веса w_i сети.

2. Задаются $k = 1$ и $E = 0$.

3. Вводится очередная обучающая пара $(\mathbf{x}^k, y^k)$. Производятся обозначения

$$\mathbf{x} := \mathbf{x}^k, \quad y := y^k$$

и вычисляется величина выхода сети:

$$\mathrm{o} = \mathrm{o}(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}.$$

4. Обновляются (корректируются) веса:

$$\mathbf{w} := \mathbf{w} + \eta(y - \mathrm{o})\mathrm{o}(1 - \mathrm{o})\mathbf{x}.$$

5. Корректируется (наращивается) значение функции ошибки:

$$E := E + \frac{1}{2}(y - \mathrm{o})^2.$$

6. Если $k < N$, тогда $k := k + 1$ и переход к шагу 3, в противоположном случае — переход на шаг 7.

7. Завершение цикла обучения. Если $E < E_{\max}$, то окончание всей процедуры обучения. Если $E \geqslant E_{\max}$, тогда начинается новый цикл обучения переходом к шагу 2.

Рассмотрим теперь более общий случай, полагая, что двухслойная НС содержит несколько (L) скрытых нейронов и один выходной (рис. 2.9).

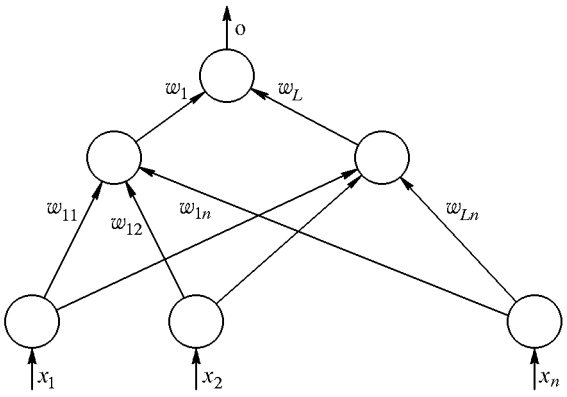

Рис. 2.9. Двухслойная нейронная сеть

В данном случае функция ошибки зависит от векторов весов скрытого слоя и вектора весов, связанных с выходным нейроном.

Выход сети описывается выражением

$$\mathrm{O}^k = \frac{1}{1 + e^{-\mathbf{W}^\top \mathrm{o}^k}},$$

где $\mathbf{W}$ — вектор весов выходного нейрона, $\mathbf{o}^k$ — вектор выходов нейронов скрытого слоя с элементами

$$\mathrm{o}_i^k = \frac{1}{1 + e^{-\mathbf{w}_i^\top \mathbf{x}^k}},$$

$\mathbf{w}_i$ обозначает вектор весов, связанных с i-м скрытым нейроном, $i = 1, 2, \ldots, L$.

Правило корректировки весов в рассматриваемой НС также основано на минимизации квадратичной функции ошибки градиентным методом на основе выражений:

$$\mathbf{W} := \mathbf{W} - \eta \frac{\partial E_k(\mathbf{W}, \mathbf{w})}{\partial \mathbf{W}},$$

$$\mathbf{w}_i := \mathbf{w}_i - \eta \frac{\partial E_k(\mathbf{W}, \mathbf{w})}{\partial \mathbf{w}_i},$$

где $\eta = \mathrm{const}$ — коэффициент скорости обучения $(0 < \eta < 1)$, $i = 1, 2, \ldots, L$.

Используя правило дифференцирования сложной функции и выражение для производной сигмоидной функции активации, получим:

$$\frac{\partial E_k(\mathbf{W}, \mathbf{w})}{\partial \mathbf{W}} = \frac{1}{2} \frac{\partial}{\partial \mathbf{W}} \left(y^k - \frac{1}{1 + e^{-\mathbf{W}^\top \mathrm{o}^k}} \right)^2 = \\ = -(y^k - \mathrm{O}^k)\mathrm{O}^k(1 - \mathrm{O}^k)\mathbf{o}^k,$$

откуда следует

$$\mathbf{W} := \mathbf{W} + \eta(y^k - \mathrm{O}^k)\mathrm{O}^k(1 - \mathrm{O}^k)\mathbf{o}^k = \mathbf{W} + \eta\delta_k\mathbf{o}^k,$$

или в скалярной форме

$$W_i := W_i + \eta\delta_k \mathrm{o}_i^k, \quad i = 1, 2, \ldots, L,$$

где

$$\delta_k = (y^k - \mathrm{O}^k)\mathrm{O}^k(1 - \mathrm{O}^k).$$

Поступая аналогично, найдем

$$\frac{\partial E_k(\mathbf{W}, \mathbf{w})}{\partial \mathbf{w}_i} = -(y^k - \mathrm{O}^k)\mathrm{O}^k(1 - \mathrm{O}^k)W_i\mathrm{o}_i^k(1 - \mathrm{o}_i^k)\mathbf{x}^k,$$

откуда получаем

$$\mathbf{w}_i := \mathbf{w}_i + \eta\delta_k W_i \mathrm{o}_i^k(1 - \mathrm{o}_i^k)\mathbf{x}^k,$$

или (в скалярной форме)

$$w_{ij} := w_{ij} + \eta\delta_k W_i \mathrm{o}_i^k (1 - \mathrm{o}_i^k) x_j^k, \quad i = 1, 2, \ldots, L, \quad j = 1, 2, \ldots, n.$$

Алгоритм обучения может быть теперь представлен в виде следующих шагов.

1. Задаются некоторые η $(0 < \eta < 1)$, $E_{\max}$ и некоторые малые случайные веса w_i сети.

2. Задаются $k = 1$ и $E = 0$.

3. Вводится очередная обучающая пара $(\mathbf{x}^k, y^k)$. Производятся обозначения

$$\mathbf{x} := \mathbf{x}^k, \quad y := y^k$$

и вычисляется величина выхода сети:

$$\mathrm{O} = \frac{1}{1 + e^{-\mathbf{W}^\top \mathrm{o}}},$$

где $\mathbf{W}$ — вектор весов выходного нейрона, $\mathbf{o}^k$ — вектор выходов нейронов скрытого слоя с элементами

$$\mathrm{o}_i = \frac{1}{1 + e^{-\mathbf{w}_i^\top \mathbf{x}}},$$

$\mathbf{w}_i$ обозначает вектор весов, связанных с i-м скрытым нейроном, $i = 1, 2, \ldots, L$.

4. Производится корректировка весов выходного нейрона:

$$\mathbf{W} := \mathbf{W} + \eta\delta\mathbf{o},$$

где $\delta = (y - \mathrm{O})\mathrm{O}(1 - \mathrm{O})$.

5. Корректируются веса нейронов скрытого слоя:

$$\mathbf{w}_i := \mathbf{w}_i + \eta\delta W_i \mathrm{o}_i (1 - \mathrm{o}_i)\mathbf{x}, \quad i = 1, 2, \ldots, L.$$

6. Корректируется (наращивается) значение функции ошибки:

$$E := E + \frac{1}{2}(y - \mathrm{o})^2.$$

Если $k < N$, тогда $k := k + 1$ и переход к шагу 3, в противоположном случае переход на шаг 8.

7. Завершение цикла обучения. Если $E < E_{\max}$, то окончание всей процедуры обучения. Если $E \geqslant E_{\max}$, тогда начинается новый цикл обучения переходом к шагу 2.

Рассмотренная процедура может быть легко обобщена на случай сети с произвольным количеством слоев и нейронов в каждом

слое. Обратим внимание, что в данной процедуре сначала происходит коррекция весов для выходного нейрона, а затем — для нейронов скрытого слоя, т.е. от конца сети к ее началу. Отсюда и название — обратное распространение ошибки. Ввиду использования для обозначений греческой буквы δ, эту процедуру обучения называют еще иногда *обобщенным дельта-правилом*.

Дадим изложенному геометрическую интерпретацию.

В алгоритме *обратного распространения* вычисляется вектор градиента поверхности ошибок. Этот вектор указывает направление кратчайшего спуска по поверхности из данной точки, поэтому, если мы «немного» продвинемся по нему, ошибка уменьшится. Последовательность таких шагов (замедляющаяся по мере приближения к дну) в конце концов приведет к минимуму того или иного типа. Определенную трудность здесь представляет вопрос о том, какую нужно брать длину шагов (что определяется величиной коэффициента скорости обучения η).

При большой длине шага сходимость будет более быстрой, но имеется опасность перепрыгнуть через решение или (если поверхность ошибок имеет особо вычурную форму) уйти в неправильном направлении. Классическим примером такого явления при обучении нейронной сети является ситуация, когда алгоритм очень медленно продвигается по узкому оврагу с крутыми склонами, прыгая с одной его стороны на другую. Напротив, при маленьком шаге, вероятно, будет схвачено верное направление, однако при этом потребуется очень много итераций. На практике величина шага берется пропорциональной крутизне склона (так что алгоритм замедляет ход вблизи минимума) с некоторой константой (η), которая, как отмечалось, называется коэффициентом скорости обучения. Правильный выбор скорости обучения зависит от конкретной задачи и обычно осуществляется 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). Этот член 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.

Таким образом, алгоритм действует итеративно, и его шаги принято называть эпохами. На каждой эпохе на вход сети поочередно подаются все обучающие наблюдения, выходные значения сети сравниваются с целевыми значениями и вычисляется

ошибка. Значение ошибки, а также градиента поверхности ошибок используется для корректировки весов, после чего все действия повторяются. Начальная конфигурация сети выбирается случайным образом, и процесс обучения прекращается, либо когда пройдено определенное количество эпох, либо когда ошибка достигнет некоторого определенного уровня малости, либо когда ошибка перестанет уменьшаться (пользователь может сам выбрать нужное условие остановки).

Классический метод обратного распространения относится к алгоритмам с линейной сходимостью и известными недостатками его являются: невысокая скорость сходимости (большое число требуемых итераций для достижения минимума функции ошибки), возможность сходиться не к глобальному, а к локальным решениям (локальным минимумам отмеченной функции), возможность паралича сети (при котором большинство нейронов функционирует при очень больших значениях аргумента функций активации, т. е. на ее пологом участке; поскольку ошибка пропорциональна производной, которая на данных участках мала, то процесс обучения практически замирает).

Для устранения этих недостатков были предложены многочисленные модификации алгоритма обратного распространения, которые связаны с использованием различных функций ошибки, различных процедур определения направления и величины шага и т. п.

Переобучение и обобщение. Одна из наиболее серьезных трудностей изложенного подхода заключается в том, что таким образом мы минимизируем не ту ошибку, которую на самом деле нужно минимизировать, — ошибку, которую можно ожидать от сети, когда ей будут подаваться совершенно новые наблюдения.

Иначе говоря, мы хотели бы, чтобы нейронная сеть обладала способностью *обобщать* результат на новые наблюдения. В действительности сеть обучается минимизировать ошибку на обучающем множестве, и в отсутствие идеального и бесконечно большого обучающего множества это совсем не то же самое, что минимизировать «настоящую» ошибку на поверхности ошибок в заранее неизвестной модели явления.

Сильнее всего это различие проявляется в проблеме переобучения, или слишком близкой подгонки. Это явление проще будет продемонстрировать не для нейронной сети, а на примере аппроксимации посредством полиномов, — при этом суть явления абсолютно та же.

Полином (или многочлен) — это выражение, содержащее только константы и целые степени независимой переменной. Примеры:

$$y = 2x + 3, \quad y = 3x^2 + 4x + 1.$$

Графики полиномов могут иметь различную форму, причем чем выше степень многочлена (и, тем самым, чем больше членов в него входит), тем более сложной может быть эта форма. Если у нас есть некоторые данные, мы можем поставить цель подогнать к ним полиномиальную кривую (модель) и получить таким образом объяснение для имеющейся зависимости. Наши данные могут быть зашумлены, поэтому нельзя считать, что самая лучшая модель задается кривой, которая в точности проходит через все имеющиеся точки. Полином низкого порядка может быть недостаточно гибким средством для аппроксимации данных, в то время как полином высокого порядка может оказаться чересчур гибким, и будет точно следовать данным, принимая при этом замысловатую форму, не имеющую никакого отношения к форме настоящей зависимости.

Нейронная сеть сталкивается с точно такой же трудностью. Сети с большим числом весов моделируют более сложные функции и, следовательно, склонны к переобучению. Сеть же с небольшим числом весов может оказаться недостаточно гибкой, чтобы смоделировать имеющуюся зависимость.

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

Ответ состоит в том, чтобы использовать механизм контрольной кросс-проверки, при котором часть обучающих наблюдений резервируется и в обучении по алгоритму обратного распространения не используется. Вместо этого, по мере работы алгоритма, она используется для независимого контроля результата. В самом начале работы ошибки сети на обучающем и контрольном множествах будут одинаковыми (если они существенно отличаются, то, вероятно, разбиение всех наблюдений на два множества было неоднородно). По мере того как сеть обучается, ошибка обучения, естественно, убывает, и, пока обучение уменьшает действительную функцию ошибок, ошибка на контрольном множестве также будет убывать. Если же контрольная ошибка перестала убывать или даже стала расти, это указывает на то, что сеть начала слишком близко аппроксимировать данные и обучение следует остановить. Это явление чересчур точной аппроксимации в процессе обучения и называется переобучением. Если такое случилось, то обычно советуют уменьшить число скрытых элементов и/или слоев, ибо

сеть является слишком мощной для данной задачи. Если же сеть, наоборот, была взята недостаточно богатой для того, чтобы моделировать имеющуюся зависимость, то переобучения, скорее всего, не произойдет, и обе ошибки — обучения и проверки — не достигнут достаточного уровня малости.

Описанные проблемы с локальными минимумами и выбором размера сети приводят к тому, что при практической работе с нейронными сетями, как правило, приходится экспериментировать с большим числом различных сетей, порой обучая каждую из них по несколько раз (чтобы не быть введенным в заблуждение локальными минимумами) и сравнивая полученные результаты. Главным показателем качества результата является здесь контрольная ошибка. При этом в соответствии с общенаучным принципом, согласно которому при прочих равных следует предпочесть более простую модель, из двух сетей с приблизительно равными ошибками контроля имеет смысл выбрать ту, которая меньше.

Классический метод обратного распространения относится к алгоритмам с линейной сходимостью. Для увеличения скорости сходимости необходимо использовать матрицы вторых производных функции ошибки.

Были предложены многочисленные модификации алгоритма обратного распространения, которые связаны с использованием различных функций ошибки, различных процедур определения направления и величины шага.

1. Функции ошибки:

- интегральные функций ошибки по всей совокупности обучающих примеров;
- функции ошибки целых и дробных степеней.

2. Процедуры определения величины шага на каждой итерации:

- дихотомия;
- инерционные соотношения;
- отжиг.

3. Процедуры определения направления шага:

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

2.4.2. Обучение без учителя. Рассмотренный алгоритм обучения нейронной сети с помощью процедуры обратного распространения подразумевает наличие некоего внешнего звена, предоставляющего сети кроме входных также и целевые выходные образы.

Алгоритмы, пользующиеся подобной концепцией, называются алгоритмами обучения с учителем. Для их успешного функционирования необходимо наличие экспертов, создающих на предварительном этапе для каждого входного образа эталонный выходной. Так как создание искусственного интеллекта движется по пути копирования природных прообразов, ученые не прекращают спор на тему, 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 не мало учителей — и в буквальном, и в переносном смысле, — которые координируют внешние воздействия. Вместе с тем, чем бы ни закончился спор приверженцев этих двух концепций обучения — с учителем и без учителя, 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 сети. Некоторые алгоритмы, правда, изменяют и структуру сети, т.е. количество нейронов и их взаимосвязи, но такие преобразования правильнее назвать более широким термином — самоорганизацией, и здесь они рассматриваться не будут. Очевидно, что подстройка весов может проводиться только на основании информации, доступной в нейроне, т.е. его состояния и уже имеющихся весовых коэффициентов. Исходя из этого соображения и, что более важно, по аналогии с известными принципами самоорганизации нервных клеток, построены алгоритмы обучения Хебба.

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

$$w_{ij} := w_{ij} + \eta \mathrm{o}_j^{(q-1)} \mathrm{o}_i^{(q)},$$

где $\mathrm{o}_j^{(q-1)}$ — выходное значение j-го нейрона слоя $(q-1)$, $\mathrm{o}_i^{(q)}$ — выходное значение i-го нейрона слоя q; w_{ij} — весовой коэффициент синапса, соединяющего эти нейроны, η — коэффициент скорости обучения. Здесь и далее, для общности, под q подразумевается произвольный слой сети. При обучении по данному методу усиливаются связи между возбужденными нейронами.

Полный алгоритм обучения с применением вышеприведенной формулы будет выглядеть так:

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

2. На входы сети подается входной образ, и сигналы возбуждения распространяются по всем слоям согласно принципам классических сетей прямого распространения (feedforward), т. е. для каждого нейрона рассчитывается взвешенная сумма его входов, к которой затем применяется активационная (передаточная) функция нейрона, в результате чего получается его выходное значение.

3. На основании полученных выходных значений нейронов по приведенной формуле производится изменение весовых коэффициентов.

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

На втором шаге цикла попеременно предъявляются все образы из входного набора.

Следует отметить, что вид откликов на каждый класс входных образов неизвестен заранее и будет представлять 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

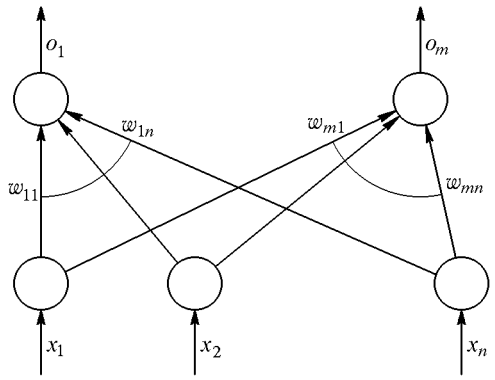

Рис. 2.10. Структура сети Кохонена

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 (Kohonen) — предусматривает самообучение по правилу «победитель забирает все». Структура сети, реализующей данное правило, представлена на рис. 2.10.

Критерий подстройки весов сети (все векторы весов должны быть нормализованы, т. е. иметь единичную длину: $\|\mathbf{w}_i\| = 1$, $i = 1, 2, \ldots, m$) выглядит следующим образом:

$$\|\mathbf{x} - \mathbf{w}_r\| = \min_{i=1,2,\ldots,m} \|\mathbf{x} - \mathbf{w}_i\|,$$

где индекс r обозначает нейрон-победитель, соответствующий вектору весов $\mathbf{w}_r$, который ближе всех расположен к (текущему) входному вектору $\mathbf{x}$.

Поскольку (с учетом того, что $\mathbf{w}_i^\top \mathbf{w}_i = \|\mathbf{w}_i\|^2 = 1$)

$$\|\mathbf{x} - \mathbf{w}_i\|^2 = (\mathbf{x} - \mathbf{w}_i)^\top (\mathbf{x} - \mathbf{w}_i) =$$
$$= \mathbf{x}^\top \mathbf{x} - 2\mathbf{w}_i^\top \mathbf{x} + \mathbf{w}_i^\top \mathbf{w}_i = \mathbf{x}^\top \mathbf{x} - 2\mathbf{w}_i^\top \mathbf{x} + 1,$$

процедура нахождения $\mathbf{w}_r$ эквивалентна решению оптимизационной задачи

$$\mathbf{w}_r^\top \mathbf{x} = \max_{i=1,2,\ldots,m} \mathbf{w}_i^\top \mathbf{x},$$

чему можно дать следующую геометрическую интерпретацию (см. рис. 2.11).

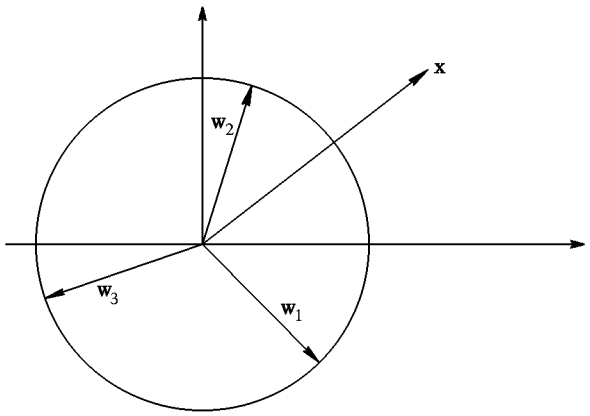

Рис. 2.11. Иллюстрация к алгоритму самообучения Кохонена

Так как скалярное произведение $\mathbf{w}_i^\top \mathbf{x}$ с учетом $\|\mathbf{w}_i\| = 1$ представляет собой просто проекцию вектора $\mathbf{x}$ на направление вектора $\mathbf{w}_i$, то нейрон-победитель определяется по тому вектору весов, чье

направление ближе всего к направлению $\mathbf{x}$ (на рис. 2.11 таким является вектор $\mathbf{w}_2$).

После выявления нейрона-победителя его выход устанавливается равным единице (у остальных нейронов устанавливаются нулевые выходы), а веса корректируются так, чтобы уменьшить квадрат величины рассогласования $\|\mathbf{x} - \mathbf{w}_r\|^2$. При использовании градиентного подхода это приводит к следующей математической формулировке:

$$\mathbf{w}_r := \mathbf{w}_r - \eta \frac{d\|\mathbf{x} - \mathbf{w}_r\|^2}{d\mathbf{w}_r},$$

где, как и раньше, η — константа, определяющая скорость обучения.

Найдем производную в правой части последнего выражения:

$$\frac{d\|\mathbf{x} - \mathbf{w}_r\|^2}{d\mathbf{w}_r} = \frac{d\big((\mathbf{x} - \mathbf{w}_r)^\top (\mathbf{x} - \mathbf{w}_r)\big)}{d\mathbf{w}_r} = \\ = \frac{d(\mathbf{x}^\top \mathbf{x} - 2\mathbf{w}_r^\top \mathbf{x} + \mathbf{w}_r^\top \mathbf{w}_r)}{d\mathbf{w}_r} = -2(\mathbf{x} - \mathbf{w}_r).$$

При этом

$$\mathbf{w}_r := \mathbf{w}_r + \eta(\mathbf{x} - \mathbf{w}_r).$$

Заметим, что коррекции весов у других нейронов не производится.

Алгоритм обучения (без учителя) Кохонена может быть теперь описан следующим образом.

1. Задаются случайные нормализованные по длине векторы $\mathbf{w}_i$.
2. Начало цикла обучения; ввод очередного вектора входов $\mathbf{x}$.
3. Определение нейрона-победителя, корректировка вектора его весов и задание единичного выхода:

$$\mathbf{w}_r := \mathbf{w}_r + \eta(\mathbf{x} - \mathbf{w}_r), \quad \mathrm{o}_r = 1.$$

4. Нормализация найденного вектора:

$$\mathbf{w}_r := \frac{\mathbf{w}_r}{\|\mathbf{w}_r\|}.$$

5. Задание значений для остальных нейронов:

$$\mathbf{w}_i := \mathbf{w}_i, \quad \mathrm{o}_i = 0, \quad i \neq r.$$

6. Проверка выполнения правила останова (в качестве такого правила можно принять, например, стабилизацию векторов весов

на каких-то значениях); если оно не выполнено — продолжение цикла обучения (переходом к шагу 2), в противоположном случае — переход к шагу 7.

7. Конец процедуры обучения.

Заметим, что выражение для коррекции вектора весовых коэффициентов нейрона-победителя может быть представлено в форме

$$\mathbf{w}_r := \mathbf{w}_r + \eta(\mathbf{x} - \mathbf{w}_r) = (1 - \eta)\mathbf{w}_r + \eta\mathbf{x},$$

т.е. новое (скорректированное) значение данного вектора является взвешенной суммой старого значения (до коррекции) и предъ-

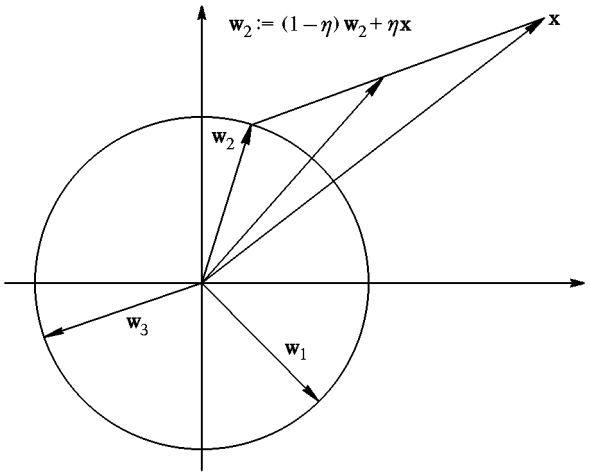

Рис. 2.12. Иллюстрация к процедуре коррекции вектора весов нейрона-победителя

явленного вектора входов, чему соответствует геометрическая иллюстрация рис. 2.12.

Нетрудно показать, что итоговым результатом подобных коррекций (в условиях рассматриваемого примера для двумерного случая) являются векторы весов, показывающие на центры кластеров (центры группирования) входных образов (рис. 2.13).

Иначе говоря, алгоритм обучения Кохонена обеспечивает решение задачи автоматической классификации, т.е. отнесения предъявленного вектора входов к одному из классов (на рис. 2.13 таких классов 3). Правда, такая классификация возможна только в случае, когда кластеры являются линейно разделимыми (гиперплоскостями) относительно начала координат в пространстве входов НС.

Отметим, что число нейронов НС для успешного решения указанной задачи должно быть не меньше, чем число кластеров; поскольку точное число кластеров может быть заранее неизвестно, количество нейронов задают с определенным запасом. «Лишние»

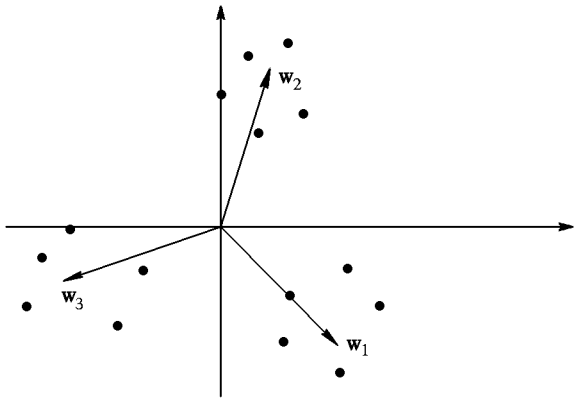

Рис. 2.13. Векторы весов НС после окончания процесса обучения

нейроны, у которых в процессе обучения сети веса изменяются хаотически по завершении данного процесса могут быть удалены.

2.5. Применение нейросети

После того как сеть обучена, мы можем применять ее для решения различных задач (рис. 2.14).

Важнейшая особенность человеческого мозга состоит в том, что, однажды обучившись определенному процессу, он может верно

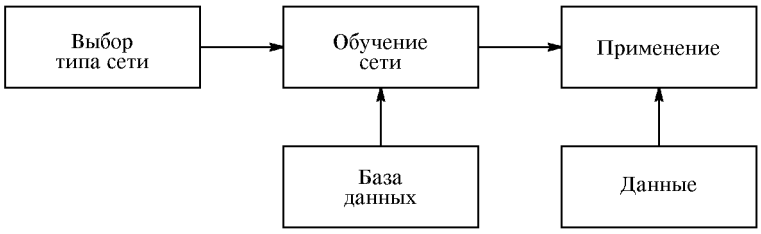

Рис. 2.14. Этапы нейросетевого проекта

действовать и в тех ситуациях, в которых он не бывал в процессе обучения. Например, мы можем читать почти любой почерк, даже

если видим его первый раз в жизни. Так же и нейросеть, грамотным образом обученная, может с большой вероятностью правильно реагировать на новые, не предъявленные ей ранее данные. Например, мы можем нарисовать букву «А» другим почерком, а затем предложить нашей 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.

Области применения нейросетей: классификация. Отметим, что задачи классификации (типа распознавания букв) очень плохо алгоритмизуются. Если в случае распознавания букв верный ответ очевиден для нас заранее, то в более сложных практических задачах обученная нейросеть 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 и т. д.). Конечно, «мнение» сети в этом случае нельзя считать окончательным. Классификация предприятий по степени их перспективности — это уже привычный способ использования нейросетей в практике западных компаний (деление компаний на перспективные и убыточные). 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» и т. д. Во втором же случае учитывается только один фактор, показанный на графике, — курс за определенный период времени. Нейросеть позволяет обрабатывать огромное количество факторов (до нескольких тысяч), независимо от их наглядности, — это универсальный «хороший врач», который может поставить свой диагноз в любой области.

Кластеризация и поиск зависимостей. Помимо задач классификации, нейросети широко используются для поиска зависимостей в данных и кластеризации.

Например, нейросеть на основе методики МГУА (метод группового учета аргументов) позволяет на основе обучающей выборки 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.

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, таким образом можно быстро выявить фальсифицированные страховые случаи или недобросовестные предприятия.

П р о г н о з и р о в а н и е. Задачи прогнозирования особенно важны для практики, в частности, для финансовых приложений, поэтому поясним способы применения нейросетей в этой области более подробно.

Рассмотрим практическую задачу, ответ в которой неочевиден, — задачу прогнозирования курса акций на 1 день вперед.

Пусть у нас имеется база данных, содержащая значения курса за последние 300 дней. Простейший вариант в данном случае — попытаться построить прогноз завтрашней цены на основе курсов за последние несколько дней. Понятно, что прогнозирующая сеть должна иметь всего один выход и столько входов, сколько предыдущих значений мы хотим использовать для прогноза — например, 4 последних значения. Составить обучающий пример очень просто: входными значениями будут курсы за 4 последовательных дня, а желаемым выходом — известный нам курс в следующий день за этими четырьмя, т.е. каждая строка таблицы с обучающей последовательностью (выборкой) представляет собой обучающий пример, где первые 4 числа — входные значения сети, а пятое число — желаемое значение выхода.

Заметим, что 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 мы имеем дело с огромным массивом данных, что делает обучение и использование сети практически невозможным. При выборе числа входов следует учитывать это, выбирая разумный компромисс между глубиной предсказания (число входов) и качеством обучения (объем тренировочного набора).

Вообще говоря, в зависимости от типа решаемой задачи, целесообразно применять нейронную сеть наиболее подходящей для

такой задачи структуры. Рассмотрим некоторые, наиболее употребительные виды НС.

2.5.1. Персептроны. В качестве научного предмета искусственные нейронные сети впервые заявили о себе в 40-е годы. Стремясь воспроизвести функции человеческого мозга, исследователи создали простые аппаратные (а позже программные) модели биологического нейрона и системы его соединений. Когда нейрофизиологи достигли более глубокого понимания нервной системы человека, эти ранние попытки стали восприниматься как весьма грубые аппроксимации. Тем не менее на этом пути были достигнуты впечатляющие результаты, стимулировавшие дальнейшие исследования, приведшие к созданию более изощренных сетей.

Первое систематическое изучение искусственных нейронных сетей было предпринято Маккалохом и Питтсом в 1943 г. Позднее они исследовали сетевые парадигмы для распознавания изображений, подвергаемых сдвигам и поворотам, используя при этом простую нейронную модель, показанную на рис. 2.15. Элемент Σ умножает каждый вход x_i на вес w_i и суммирует взвешенные

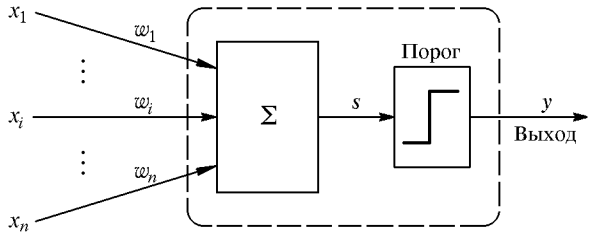

Рис. 2.15. Персептронный нейрон

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 (см. рис. 2.16), хотя в принципе описываются и более сложные системы.

В 60-е годы персептроны вызвали большой интерес и оптимизм. Розенблатт доказал замечательные теоремы об обучении персептронов, приводимые ниже. Уидроу дал ряд убедительных демонстраций систем персептронного типа, и исследователи во всем мире стремились изучить возможности этих систем. Первоначальная эйфория сменилась разочарованием, когда оказалось,

что персептроны не способны обучиться решению ряда простых задач. Минский строго проанализировал эту проблему и показал, что имеются 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, исследователи перешли в более многообещающие области, и исследования в области нейронных сетей

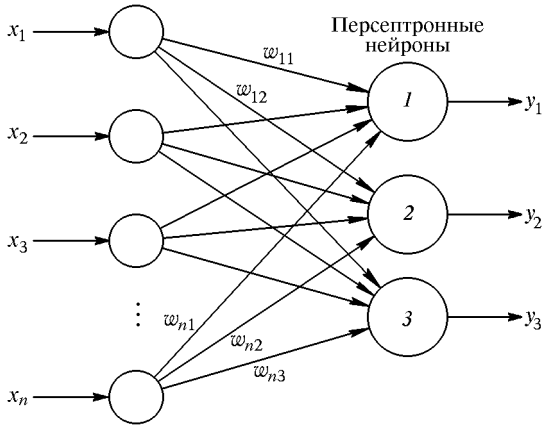

Рис. 2.16. Персептрон со многими выходами

пришли в упадок. Недавнее открытие методов обучения многослойных сетей в большей степени, чем какой-либо иной фактор, повлияло на возрождение интереса и исследовательских усилий.

Работа Минского возможно и охладила пыл энтузиастов персептрона, но обеспечила время для необходимой консолидации и развития лежащей в основе теории. Важно отметить, что анализ Минского не был опровергнут. Он остается важным исследованием и должен изучаться, чтобы ошибки 60-х годов не повторились.

Несмотря на свои ограничения, персептроны широко изучались (хотя не слишком широко использовались). Теория персептронов является основой для многих других типов искусственных нейронных сетей, в силу чего они являются логической исходной точкой для изучения искусственных нейронных сетей.

Рассмотрим в качестве примера трехнейронный персептрон (рис. 2.16), нейроны которого имеют активационную функцию в виде единичного скачка.

На n входов поступают входные сигналы, проходящие по синапсам, на три нейрона, образующие единственный слой этой сети

и выдающие три выходных сигнала:

$$y_j = f\left(\sum_{i=1}^{n} x_i w_{ij}\right), \quad j = 1, \ldots, 3.$$

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

$$\mathbf{Y} = \mathbf{f}(\mathbf{XW}),$$

где $\mathbf{X}$ и $\mathbf{Y}$ — соответственно входной и выходной сигнальные векторы (здесь и далее под вектором понимается вектор-строка), $\mathbf{f}(\mathbf{S})$ — активационная функция, применяемая поэлементно к компонентам вектора $\mathbf{S}$.

На рис. 2.17 представлен двухслойный персептрон, полученный из персептрона с рис. 2.16 путем добавления второго слоя, состоящего из двух нейронов.

Здесь уместно отметить важную роль нелинейности активационной функции, так как, если бы она не обладала данным свойст-

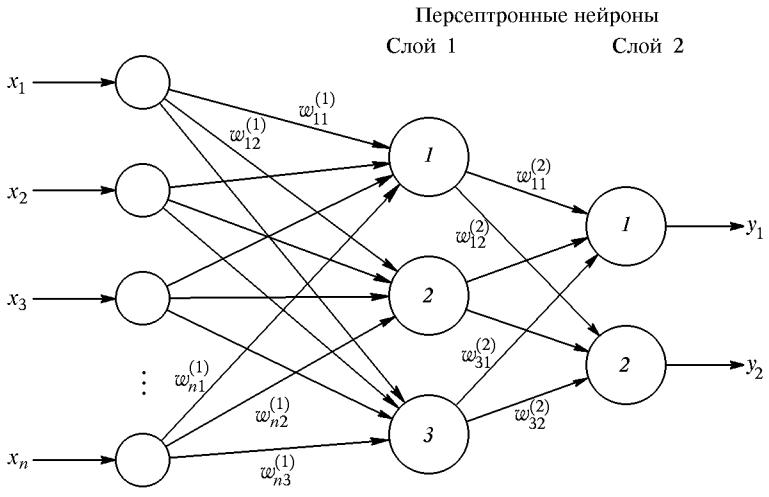

Рис. 2.17. Двухслойный персептрон

вом или не входила в алгоритм работы каждого нейрона, результат функционирования любой Q-слойной нейронной сети с весовыми

матрицами $\mathbf{W}^{(q)}$ для каждого слоя $q = 1, \ldots, Q$ сводился бы к перемножению входного вектора сигналов $\mathbf{X}$ на матрицу:

$$\mathbf{W}_{(\Sigma)} = \mathbf{W}^{(1)} \ldots \mathbf{W}^{(q)} \ldots \mathbf{W}^{(Q)}.$$

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

$$\mathbf{Y} = \mathbf{X}\mathbf{W}_{(\Sigma)}.$$

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 случая однослойного персептрона

$$\sum_{i=1}^{n} x_i w_{ij} = \theta_j, \quad j = 1, 2, \ldots, m.$$

Каждая полученная область является областью определения отдельного класса. Число таких классов для персептрона не превышает 2^n, где n — число его входов. Однако не все из классов могут быть разделимы данной нейронной сетью. 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. 2.3).

Уравнение сети для этого случая

$$x_1 w_1 + x_2 w_2 = \theta$$

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. 2.18). Невозможность 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 двух- и более слойных сетей или сетей с нелинейными синапсами, однако и тогда существует вероятность, что корректное разделение некоторых входных сигналов на классы невозможно.

Обучение персептрона сводится к формированию весов связей между первым и вторым (см. рис. 2.17) слоями в соответствии со следующим алгоритмом.

Таблица 2.3. **Логическая функция ИСКЛЮЧАЮЩЕЕ ИЛИ**

x_1 \ x_2	0	1
0	B	A
1	A	B

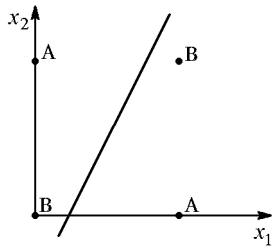

Рис. 2.18. Линейная неразделимость функции ИСКЛЮЧАЮЩЕЕ ИЛИ

Шаг 1. Проинициализировать элементы весовой матрицы (обычно 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 идеальным d и полученным Y 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 соответственно 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 втором шаге на разных итерациях поочередно в случайном порядке предъявляются все возможные входные векторы. К сожалению, нельзя заранее определить число итераций, которые потребуется выполнить, а в некоторых случаях и гарантировать полный успех.

Сходимость рассмотренной процедуры устанавливается теоремами, утверждающими, что для любой классификации обучающей последовательности можно подобрать такой набор (из бесконечного набора) элементарных нейронов, в котором будет осуществлено разделение обучающей последовательности при помощи линейного

решающего правила, и что, если относительно задуманной классификации можно найти набор элементов, в котором существует решение, то в рамках этого набора оно будет достигнуто в конечный промежуток времени.

2.5.2. 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 так называемых 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. Это может быть полезно в тех приложениях, где долгая обучающая процедура невозможна

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 позволяет эффективно использовать данную сеть для распознавания и восстановления образов, а также для усиления сигналов.

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.

Сеть встречного распространения имеет два слоя с последовательными связями. Первый слой — слой Кохонена, второй — слой Гроссберга. Каждый элемент входного сигнала подается на все нейроны слоя Кохонена. Каждый нейрон слоя Кохонена соединен со всеми нейронами слоя Гроссберга. Отличие сети встречного распространения от других многослойных сетей с последовательными связями состоит в операциях, выполняемых нейронами Кохонена и Гроссберга.

В режиме функционирования сети предъявляется входной сигнал X и формируется выходной сигнал Y. В режиме обучения на вход сети подается входной сигнал и веса корректируются, чтобы сеть выдавала требуемый выходной сигнал.

Функционирование сети

Слой Кохонена. В своей простейшей форме слой Кохонена функционирует по правилу «победитель получает все». Для данного входного вектора один и только один нейрон Кохонена выдает логическую единицу, все остальные выдают ноль. Выход каждого нейрона Кохонена является просто суммой взвешенных элементов входных сигналов:

$$s_j = w_{1j}x_1 + w_{2j}x_2 + \ldots + w_{nj}x_n = \sum_i x_i w_{ij},$$

где s_j — выход j-го нейрона Кохонена, $\mathbf{W}_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$ — вектор весов j-го нейрона Кохонена, $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ — вектор входного сигнала, или в векторно-матричной форме:

$$\mathbf{S} = \mathbf{XW},$$

где $\mathbf{S}$ — вектор выходов слоя Кохонена.

Нейрон Кохонена с максимальным значением s_j является «победителем». Его выход равен единице, у остальных он равен нулю.

Слой Гроссберга. Слой Гроссберга функционирует в сходной манере. Его выход является взвешенной суммой выходов слоя Кохонена (т.е. он является слоем нейронов с линейными активационными функциями).

Если слой Кохонена функционирует таким образом, что лишь один выход равен единице, а остальные равны нулю, то каждый нейрон слоя Гроссберга выдает величину веса, который связывает этот нейрон с единственным нейроном Кохонена, чей выход отличен от нуля.

Предварительная обработка входных сигналов. Рассматриваемая НС требует предварительной обработки входных векторов путем их нормализации. Такая нормализация выполняется путем деления каждой компоненты входного вектора на длину вектора (квадратный корень из суммы квадратов компонент вектора). Это превращает входной вектор в единичный вектор с тем же направлением, т.е. в вектор единичной длины в n-мерном пространстве.

Обучение слоя Кохонена. 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 (затем задачей слоя Гроссберга является получение требуемых выходов).

Слой Кохонена обучается без учителя (самообучается). В результате обучения слой приобретает способность разделять несхожие входные векторы. Какой именно нейрон будет активироваться при предъявлении конкретного входного сигнала, заранее трудно предсказать.

При обучении слоя Кохонена на вход подается входной вектор и вычисляются его скалярные произведения с векторами весов всех нейронов. Скалярное произведение является мерой сходства между входным вектором и вектором весов. Нейрон с максимальным значением скалярного произведения объявляется «победителем» и его веса подстраиваются (весовой вектор приближается к входному).

Уравнение, описывающее процесс обучения, имеет вид

$$w_{\text{н}} = w_{\text{с}} + \eta(x - w_{\text{с}}),$$

где $w_{\text{н}}$ — новое значение веса, соединяющего входную компоненту x с выигравшим нейроном, $w_{\text{с}}$ — предыдущее значение этого веса, η — коэффициент скорости обучения.

Каждый вес, связанный с выигравшим нейроном Кохонена, изменяется пропорционально разности между его величиной и величиной входа, к которому он присоединен. Направление изменения минимизирует разность между весом и соответствующим элементом входного сигнала.

Коэффициент скорости обучения η вначале обычно полагается равным 0,7 и может затем постепенно уменьшаться в процессе обучения. Это позволяет делать большие начальные шаги для быстрого грубого обучения и меньшие шаги при подходе к окончательной величине.

Если бы с каждым нейроном Кохонена ассоциировался один входной вектор, то слой Кохонена мог бы быть обучен с помощью одного вычисления на вес ($\eta = 1$). Как правило, обучающее множество включает много сходных между собой входных векторов, и сеть должна быть обучена активировать один и тот же нейрон Кохонена для каждого из них. Веса этого нейрона должны получаться усреднением входных векторов, которые должны его активировать.

Обучение слоя Гроссберга. Выходы слоя Кохонена подаются на входы нейронов слоя Гроссберга. Выходы нейронов вычисляются, как при обычном функционировании. Далее каждый вес корректируется лишь в том случае, если он соединен с нейроном Кохонена, имеющим ненулевой выход. Величина коррекции веса пропорциональна разности между весом и требуемым выходом нейрона Гроссберга.

Обучение слоя Гроссберга — это обучение с учителем, алгоритм использует заданные желаемые выходы.

В полной модели сети встречного распространения имеется возможность получать выходные сигналы по входным и наоборот. Этим двум действиям соответствуют прямое и обратное распространение сигналов.

Области применения: распознавание образов, восстановление образов (ассоциативная память), сжатие данных (с потерями).

Недостатки. Сеть не дает возможности строить точные аппроксимации (точные отображения). В этом сеть значительно уступает сетям с обратным распространением ошибки.

К недостаткам модели также следует отнести слабую теоретическую проработку модификаций сети встречного распространения.

Преимущества. Сеть встречного распространения проста. Она дает возможность извлекать статистические свойства 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 $1/p$, p — число нейронов Кохонена.

Сеть быстро обучается. Время 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.

Модификации. Сети встречного распространения могут различаться способами определения начальных значений синаптических весов. Так, кроме традиционных 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 $1/p$ (p — число нейронов Кохонена), то ему временно увеличивают порог, давая тем самым 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 целая группа нейронов Кохонена, имеющих наибольшие выходы, может передавать свои выходные сигналы в слой Гроссберга. Этот метод повышает точность отображений, реализуемых сетью, и реализован в так называемых *самоорганизующихся картах.*

2.5.3. Нейронные сети Хопфилда и Хэмминга. Среди различных конфигураций искусственных нейронных сетей (НС) встречаются такие, при классификации которых по принципу обучения, строго говоря, не подходят ни обучение с учителем, ни обучение без учителя. В таких сетях весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных, и все обучение сети сводится именно к этому расчету. С одной стороны, предъявление априорной информации можно расценивать как помощь учителя, но с другой, — сеть фактически просто запоминает образцы до того, как на ее вход поступают реальные данные, и не может изменять свое поведение, поэтому говорить о звене обратной связи с «миром» (учителем) не приходится. Из сетей с подобной логикой работы наиболее известны *сеть Хопфилда* и *сеть Хэмминга* (представляющие собой разновидности сетей с обратными связями), которые обычно используются для организации ассоциативной памяти. Далее речь пойдет именно о них. Структурная схема сети Хопфилда приведена на рис. 2.19. Она состоит

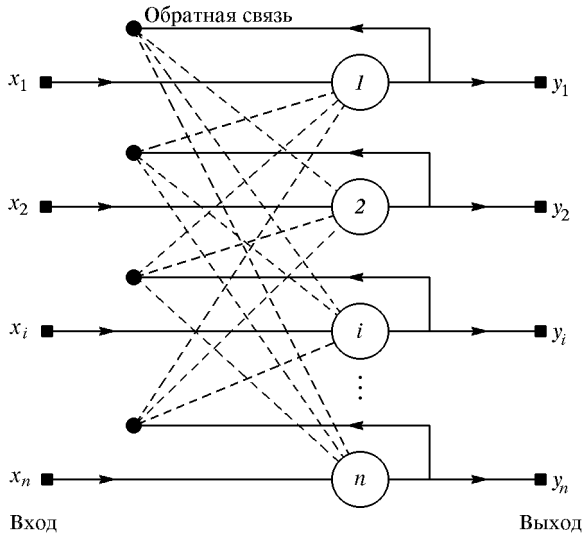

Рис. 2.19. Структурная схема сети Хопфилда

из единственного слоя нейронов, число которых является одновременно числом входов и выходов сети. Каждый нейрон связан синапсами со всеми остальными нейронами, а также имеет один

входной синапс, через который осуществляется ввод сигнала. Выходные сигналы, как обычно, образуются на аксонах.

Задача, решаемая данной сетью в качестве ассоциативной памяти, как правило, формулируется следующим образом. Известен некоторый набор двоичных сигналов (изображений, звуковых оцифровок, прочих данных, описывающих некие объекты или характеристики процессов), которые считаются образцовыми. Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить («вспомнить» по частичной информации) соответствующий образец (если такой есть) или «дать заключение» о том, что входные данные не соответствуют ни одному из образцов. В общем случае, любой сигнал может быть описан вектором $\mathbf{X} = \{x_i : i = 1, 2, \ldots, n\}$, n — число нейронов в сети и размерность входных и выходных векторов. Каждый элемент x_i равен либо $+1$, либо -1. Обозначим вектор, описывающий k-й образец, через $\mathbf{X}^k$, а его компоненты, соответственно, — x_i^k, $k = 1, 2, \ldots, m$, где m — в данном случае число образцов. Когда сеть распознает (или «вспомнит») какой-либо образец на основе предъявленных ей данных, ее выходы будут содержать именно его, т.е. $\mathbf{Y} = \mathbf{X}^k$, где $\mathbf{Y}$ — вектор выходных значений сети: $\mathbf{Y} = \{y_i : i = 1, 2, \ldots, n\}$. В противном случае, выходной вектор не совпадет ни с одним образцовым.

Если, например, сигналы представляют собой некие изображения, то, отобразив в графическом виде данные с выхода сети, можно будет увидеть картинку, полностью совпадающую с одной из образцовых (в случае успеха), или же «вольную импровизацию» сети (в случае неудачи).

На стадии инициализации сети весовые коэффициенты синапсов устанавливаются следующим образом:

$$w_{ij} = \begin{cases} \sum\limits_{k=1}^{m} x_i^k x_j^k, & i \neq j, \\ 0, & i = j. \end{cases}$$

Здесь i и j — индексы, соответственно, предсинаптического и постсинаптического нейронов; x_i^k, x_j^k — i-й и j-й элементы вектора k-го образца.

Алгоритм функционирования сети следующий (t — номер итерации):

1. На входы сети подается неизвестный сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов:

$$y_i(0) = x_i, \quad i = 1, 2, \ldots, n,$$

поэтому обозначение на схеме сети входных синапсов в явном виде носит чисто условный характер. Ноль в скобке справа от y_i означает нулевую итерацию в цикле работы сети.

2. Рассчитывается новое состояние нейронов

$$s_j(t+1) = \sum_{i=1}^{n} w_{ij} y_j(t), \quad j = 1, 2, \ldots, n,$$

и новые значения аксонов

$$y_j(t+1) = f\lfloor s_j(t+1) \rfloor,$$

где f — пороговая активационная функция с порогом $\theta = 0$ (см. табл. 2.1).

3. Проверка, изменились ли выходные значения аксонов за последнюю итерацию. Если да — переход к пункту 2, иначе (если выходы застабилизировались) — конец. При этом выходной вектор представляет собой образец, наилучшим образом сочетающийся с входными данными.

Таким образом, когда подается новый вектор, сеть переходит из вершины в вершину, пока не стабилизируется. Устойчивая вершина определяется сетевыми весами и текущими входами. Если входной вектор частично неправилен или неполон, сеть стабилизируется в вершине, ближайшей к желаемой.

Показано, что достаточным условием устойчивой работы такой сети является выполнение условий

$$w_{ij} = w_{ji}, \quad w_{ii} = 0.$$

Как говорилось выше, иногда сеть не может провести распознавание и выдает на выходе несуществующий образ. Это связано с проблемой ограниченности возможностей сети. Для сети Хопфилда число запоминаемых образов N не должно превышать величины, примерно равной $0{,}15n$. Кроме того, если два образа A и B сильно похожи, они, возможно, будут вызывать у сети перекрестные ассоциации, т. е. предъявление на входы сети вектора **A** приведет к появлению на ее выходах вектора **B** и наоборот.

Когда нет необходимости, чтобы сеть в явном виде выдавала образец, т. е. достаточно, скажем, получать номер образца, ассоциативную память успешно реализует сеть Хэмминга. Данная сеть характеризуется, по сравнению с сетью Хопфилда, меньшими затратами на память и объемом вычислений, что становится очевидным из ее структуры (рис. 2.20).

Сеть состоит из двух слоев. Первый и второй слои имеют по m нейронов, где m — число образцов. Нейроны первого слоя

имеют по n синапсов, соединенных с входами сети (образующими фиктивный нулевой слой). Нейроны второго слоя связаны между собой ингибиторными (отрицательными обратными) синаптическими связями. Единственный синапс с положительной обратной связью для каждого нейрона соединен с его же аксоном.

Идея работы сети состоит в нахождении расстояния Хэмминга от тестируемого образа до всех образцов (расстоянием Хэмминга называется число отличающихся битов в двух бинарных векторах). Сеть должна выбрать образец с минимальным расстоянием

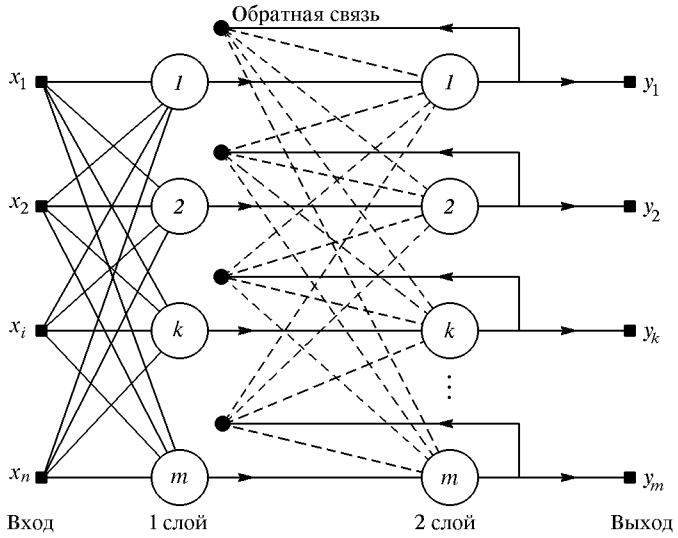

Рис. 2.20. Структурная схема сети Хэмминга

Хэмминга до неизвестного входного сигнала, в результате чего будет активизирован только один выход сети, соответствующий этому образцу.

На стадии инициализации весовым коэффициентам первого слоя и порогу активационной функции присваиваются следующие значения:

$$w_{ik} = \frac{x_i^k}{2}, \quad \theta_k = \frac{n}{2}, \quad i = 1, 2, \ldots, n, \;\; k = 1, 2, \ldots, m.$$

Здесь x_i^k — i-й элемент k-го образца.

Весовые коэффициенты тормозящих синапсов во втором слое берут равными некоторой величине $0 < \varepsilon < 1/m$. Синапс нейрона, связанный с его же аксоном, имеет вес $+1$.

Алгоритм функционирования сети Хэмминга следующий:

1. На входы сети подается неизвестный вектор $\mathbf{X} = \{x_i : i = 1, \ldots, n\}$, исходя из которого рассчитываются состояния нейронов первого слоя (верхний индекс в скобках указывает номер слоя):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=1}^{n} w_{ij} x_i + \theta_j, \quad j = 1, 2, \ldots, m.$$

После этого полученными значениями инициализируются значения аксонов второго слоя:

$$y_j^{(2)} = y_j^{(1)}, \quad j = 1, 2, \ldots, m.$$

2. Вычисляются новые состояния нейронов второго слоя:

$$s_j^{(2)}(t+1) = y_j(t) - \varepsilon \sum_{k=1}^{m} y_k^{(2)}(t), \quad k \neq j, \;\; j = 1, 2, \ldots, m,$$

и значения их аксонов:

$$y_j^{(2)}(t+1) = f\lfloor s_j^{(2)}(t+1) \rfloor, \quad j = 1, 2, \ldots, m.$$

3. Проверяется, изменились ли выходы нейронов второго слоя за последнюю итерацию. Если да — перейди к шагу 2. Иначе — конец.

Из оценки алгоритма видно, что роль первого слоя весьма условна: воспользовавшись один раз на шаге 1 значениями его весовых коэффициентов, сеть больше не обращается к нему, поэтому первый слой может быть вообще исключен из сети (заменен на матрицу весовых коэффициентов).

В заключение можно сделать следующее обобщение. Сети Хопфилда и Хэмминга позволяют просто и эффективно разрешить задачу 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 применений.

2.5.4. Сеть с радиальными базисными элементами (RBF). В общем случае под термином Radial Basis Function Network (сеть

RBF) понимается двухслойная сеть без обратных связей, которая содержит скрытый слой радиально симметричных скрытых нейронов (шаблонный слой). Для того чтобы шаблонный слой был радиально-симметричным, необходимо выполнение следующих условий:

- наличие центра, представленного в виде вектора во входном пространстве; обычно этот вектор сохраняется в пространстве весов от входного слоя к слою шаблонов;
- наличие способа измерения расстояния входного вектора от центра; обычно это стандартное евклидово расстояние;
- наличие специальной функции прохождения от одного аргумента, которая определяет выходной сигнал нейрона путем отображения функции расстояния; обычно используется функция Гаусса

$$\varphi(s) = e^{-s^2}.$$

Другими словами, выходной сигнал шаблонного нейрона — это функция только от расстояния между входным вектором $\mathbf{X}$ и сохраненным центром $\mathbf{C}$:

$$f(\mathbf{X}) = \varphi\left(\frac{\|\mathbf{X} - \mathbf{C}\|}{\sigma}\right).$$

Выходной слой сети является линейным, так что выходы сети определяются выражением

$$y_j = \sum_{i=1}^{K} w_{ij}\varphi\left(\frac{\|\mathbf{X} - \mathbf{C}_i\|}{\sigma_i}\right), \quad j = 1, 2, \ldots, m,$$

где $\mathbf{C}_i$ — центры, σ_i — отклонения радиальных элементов.

Обучение RBF-сети происходит в несколько этапов. Сначала определяются центры и отклонения для радиальных элементов; после этого оптимизируются параметры w_{ij} линейного выходного слоя.

Расположение центров должно соответствовать кластерам, реально присутствующим в исходных данных. Рассмотрим два наиболее часто используемых метода.

Выборка из выборки. В качестве центров радиальных элементов берутся несколько случайно выбранных точек обучающего множества. В силу случайности выбора они «представляют» распределение обучающих данных в статистическом смысле. Однако, если число радиальных элементов невелико, такое представление может быть неудовлетворительным.

Алгоритм K-средних. Этот алгоритм стремится выбрать оптимальное множество точек, являющихся центроидами кластеров в обучающих данных. При K радиальных элементах их центры располагаются таким образом, чтобы:

- каждая обучающая точка «относилась» к одному центру кластера и лежала к нему ближе, чем к любому другому центру;
- каждый центр кластера был центроидом множества обучающих точек, относящихся к этому кластеру.

После того как определено расположение центров, нужно найти отклонения. Величина отклонения (ее также называют сглаживающим фактором) определяет, насколько «острой» будет гауссова функция. Если эти функции выбраны слишком острыми, сеть не будет интерполировать данные между известными точками и потеряет способность к обобщению. Если же гауссовы функции взяты чересчур широкими, сеть не будет воспринимать мелкие детали (на самом деле сказанное — еще одна форма проявления дилеммы пере/недообучения). Как правило, отклонения выбираются таким образом, чтобы «колпак» каждой гауссовой функций захватывал несколько соседних центров. Для этого имеется несколько методов:

Явный. Отклонения задаются пользователем.

Изотропный. Отклонение берется одинаковым для всех элементов и определяется эвристически с учетом количества радиальных элементов и объема покрываемого пространства.

K ближайших соседей. Отклонение каждого элемента устанавливается (индивидуально) равным среднему расстоянию до его K ближайших соседей. Тем самым отклонения будут меньше в тех частях пространства, где точки расположены густо, — здесь будут хорошо учитываться детали, — а там, где точек мало, отклонения будут большими (и будет производиться интерполяция).

После того как выбраны центры и отклонения, параметры выходного слоя оптимизируются с помощью стандартного метода линейной оптимизации — алгоритма псевдообратных матриц (сингулярного разложения).

Могут быть построены различные гибридные разновидности сетей с радиальными базисными функциями. Например, выходной слой может иметь нелинейные функции активации, и тогда для его обучения используется какой-либо из алгоритмов обучения многослойных сетей, например метод обратного распространения. Можно также обучать радиальный (скрытый) слой с помощью алгоритма обучения сети Кохонена — это еще один способ разместить центры так, чтобы они отражали расположение данных.

Сети RBF имеют ряд *преимуществ* перед рассмотренными многослойными сетями прямого распространения (хотя их структура и соответствует приведенной на рис. 2.5). Во-первых, они моделируют произвольную нелинейную функцию с помощью всего одного промежуточного слоя, тем самым избавляя нас от необходимости решать вопрос о числе слоев. Во-вторых, параметры линейной комбинации в выходном слое можно полностью оптимизировать с помощью хорошо известных методов линейной оптимизации, которые работают быстро и не испытывают трудностей с локальными минимумами, так мешающими при обучении с использованием алгоритма обратного распространения ошибки. Поэтому сеть RBF обучается очень быстро (на порядок быстрее, чем с использованием алгоритма обратного распространения).

Недостатки сетей RBF: данные сети обладают плохими экстраполирующими свойствами и получаются весьма громоздкими при большой размерности вектора входов.

2.5.5. Вероятностная нейронная сеть (PNN). Задача оценки плотности вероятности по имеющимся данным имеет давнюю историю в математической статистике.

Обычно при этом предполагается, что плотность имеет некоторый определенный вид (чаще всего, — что она имеет нормальное распределение). После этого оцениваются параметры модели. Нормальное распределение часто используется потому, что тогда параметры модели (среднее и стандартное отклонение) можно оценить аналитически.

Заметим, что предположение о нормальности далеко не всегда оправдано.

Другой подход к оценке плотности вероятности основан на так называемых *ядерных оценках*. Можно рассуждать так: тот факт, что наблюдение расположено в данной точке пространства, свидетельствует о том, что в этой точке имеется некоторая плотность вероятности. Кластеры из близко лежащих точек указывают на то, что в этом месте плотность вероятности большая. Вблизи наблюдения имеется большее доверие к уровню плотности, а по мере отдаления от него доверие убывает и стремится к нулю. В методе ядерных оценок в точке, соответствующей каждому наблюдению, помещается некоторая простая функция, затем все они складываются и в результате получается оценка для общей плотности вероятности. Чаще всего в качестве ядерных функций берутся гауссовы функции (с формой колокола). Если обучающих примеров достаточное количество, то такой метод дает достаточно хорошее приближение к истинной плотности вероятности.

Метод аппроксимации плотности вероятности с помощью ядерных функций во многом похож на метод радиальных базисных функций, и таким образом мы естественно приходим к понятиям вероятностной нейронной сети (PNN) и обобщенно-регрессионной нейронной сети (GRNN). PNN-сети предназначены для задач классификации, а GRNN — для задач регрессии. Сети этих двух типов представляют собой реализацию методов ядерной аппроксимации, оформленных в виде нейронной сети.

Сеть PNN имеет по меньшей мере три слоя: входной, радиальный и выходной. Радиальные элементы берутся по одному на каждое обучающее наблюдение. Каждый из них представляет гауссову функцию с центром в этом наблюдении. Каждому классу соответствует один выходной элемент. Каждый такой элемент соединен со всеми радиальными элементами, относящимися к его классу, а со всеми остальными радиальными элементами он имеет нулевое соединение. Таким образом, выходной элемент просто складывает отклики всех элементов, принадлежащих к его классу. Значения выходных сигналов получаются пропорциональными ядерным оценкам вероятности принадлежности соответствующим классам, и, пронормировав их на единицу, мы получаем окончательные оценки вероятности принадлежности классам.

Выход рассматриваемой сети, соответствующий какому-либо классу, описывается выражением

$$y = \frac{1}{N\sigma^n} \sum_{k=1}^{N} \varphi\left(\frac{\|\mathbf{X} - \mathbf{X}^k\|}{\sigma}\right),$$

где n — размерность входного вектора, N — объем обучающей выборки, $\mathbf{X}^k$ — элемент (вектор) этой выборки, соответствующий отмеченному классу.

Базовая модель PNN-сети может иметь две модификации.

Вероятностная нейронная сеть имеет единственный управляющий параметр обучения, значение которого должно выбираться пользователем, — отклонение гауссовой функции σ (параметр сглаживания). Как и в случае RBF-сетей, этот параметр выбирается из тех соображений, чтобы «шапки» определенное число раз перекрывались: выбор слишком маленьких отклонений приведет к «острым» аппроксимирующим функциям и неспособности сети к обобщению, а при слишком больших отклонениях будут теряться детали. Требуемое значение несложно найти опытным путем, подбирая его так, чтобы контрольная ошибка была как можно меньше. К счастью, PNN-сети не очень чувствительны к выбору параметра сглаживания.

Наиболее важные *преимущества* PNN-сетей состоят в том, что выходное значение имеет вероятностный смысл (и поэтому его легче интерпретировать), и в том, что сеть быстро обучается. При обучении такой сети время тратится практически только на то, чтобы подавать ей на вход обучающие наблюдения, и сеть работает настолько быстро, насколько это вообще возможно.

Существенным *недостатком* таких сетей является их объем. PNN-сеть фактически вмещает в себя все обучающие данные, поэтому она требует много памяти и может медленно работать.

PNN-сети особенно полезны при пробных экспериментах (например, когда нужно решить, какие из входных переменных использовать), так как благодаря короткому времени обучения можно быстро проделать большое количество пробных тестов.

2.5.6. Обобщенно-регрессионная нейронная сеть (GRNN). Данная сеть устроена аналогично вероятностной нейронной сети (PNN), но она предназначена для решения задач регрессии, а не классификации. Как и в случае PNN-сети, в точку расположения каждого обучающего наблюдения помещается гауссова ядерная функция. Мы считаем, что каждое наблюдение свидетельствует о некоторой нашей уверенности в том, что поверхность отклика в данной точке имеет определенную высоту, и эта уверенность убывает при отходе в сторону от точки. GRNN-сеть копирует внутрь себя все обучающие наблюдения и использует их для оценки отклика в произвольной точке. Окончательная выходная оценка сети получается как взвешенное среднее выходов по всем обучающим наблюдениям:

$$y = \frac{\sum\limits_{k=1}^{N} y^k \varphi\left(\|\mathbf{X} - \mathbf{X}^k\|/\sigma\right)}{\sum\limits_{k=1}^{N} \varphi\left(\|\mathbf{X} - \mathbf{X}^k\|/\sigma\right)},$$

где $\mathbf{X}^k$, y^k — точки обучающей выборки.

Первый промежуточный слой сети GRNN состоит из радиальных элементов. Второй промежуточный слой содержит элементы, которые помогают оценить взвешенное среднее. Каждый выход имеет в этом слое свой элемент, формирующий для него взвешенную сумму. Чтобы получить из взвешенной суммы взвешенное среднее, эту сумму нужно поделить на сумму весовых коэффициентов. Последнюю сумму вычисляет специальный элемент второго слоя. После этого в выходном слое производится собственно деление (с помощью специальных элементов «деления»). Таким образом, число элементов во втором промежуточном слое на единицу больше, чем в выходном слое. Как правило, в задачах регрессии

требуется оценить одно выходное значение, и, соответственно, второй промежуточный слой содержит два элемента.

Можно модифицировать GRNN-сеть таким образом, чтобы радиальные элементы соответствовали не отдельным обучающим случаям, а их кластерам. Это уменьшает размеры сети и увеличивает скорость обучения. Центры для таких элементов можно выбирать с помощью любого предназначенного для этой цели алгоритма (выборки из выборки, K-средних или Кохонена).

Достоинства и недостатки у сетей GRNN в основном такие же, как и у сетей PNN, — единственное различие в том, что GRNN используются в задачах регрессии, а PNN — в задачах классификации. GRNN-сеть обучается почти мгновенно, но может получиться большой и медленной (хотя здесь, в отличие от PNN, не обязательно иметь по одному радиальному элементу на каждый обучающий пример, их число все равно будет большим). Как и сеть RBF, сеть GRNN не обладает способностью экстраполировать данные.

2.5.7. Линейные НС. Согласно общепринятому в науке принципу, если более сложная модель не дает лучших результатов, чем более простая, то из них следует предпочесть вторую. В терминах аппроксимации отображений самой простой моделью будет линейная, в которой аппроксимирующая (подгоночная) функция определяется гиперплоскостью. В задаче классификации гиперплоскость размещается таким образом, чтобы она разделяла собой два класса (линейная дискриминантная функция); в задаче регрессии гиперплоскость должна проходить через заданные точки. Линейная модель обычно задается уравнением

$$\mathbf{Y} = \mathbf{X}\mathbf{W} + \mathbf{B},$$

где $\mathbf{W}$ — матрица весов сети, $\mathbf{B}$ — вектор смещений.

На языке нейронных сетей линейная модель представляется сетью без промежуточных слоев, которая в выходном слое содержит только линейные элементы (т.е. элементы с линейной функцией активацией). Веса соответствуют элементам матрицы, а пороги — компонентам вектора смещения. Во время работы сеть фактически умножает вектор входов на матрицу весов, а затем к полученному вектору прибавляет вектор смещения.

2.6. Эффективность нейронных сетей

Эффективность нейронных сетей устанавливается рядом так называемых теорем о полноте. Ранее в нестрогой формулировке была приведена одна из них. Рассмотрим еще одну подобную теорему.

В 1989 г. Funahashi показал, что бесконечно большая нейронная сеть с единственным скрытым слоем способна аппроксимировать любую непрерывную функцию, сформулировав данное утверждение в форме следующей теоремы.

Т е о р е м а. *Пусть $\phi(x)$ — непостоянная, ограниченная и монотонно возрастающая непрерывная функция. Пусть, далее, $U \subset \mathbb{R}^n$ — ограниченное множество и*

$$f\colon U \to \mathbb{R}$$

— вещественная непрерывная функция, определенная на U.

Тогда для произвольного $\varepsilon > 0$ существует целое L и вещественные константы w_i, w_{ij} такие, что аппроксимация

$$\widetilde{f}(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{L} w_i \phi\left(\sum_{j=1}^{n} w_{ij} x_j\right)$$

удовлетворяет неравенству

$$\|f - \widetilde{f}\|_\infty = \sup_{\mathbf{x} \in U} |f(\mathbf{x}) - \widetilde{f}(\mathbf{x})| \leqslant \varepsilon.$$

Другими словами, любое непрерывное отображение может быть аппроксимировано в смысле однородной топологии на U двухслойной нейронной сетью с активационными функциями $\phi(x)$ для нейронов скрытого слоя и линейными активационными функциями

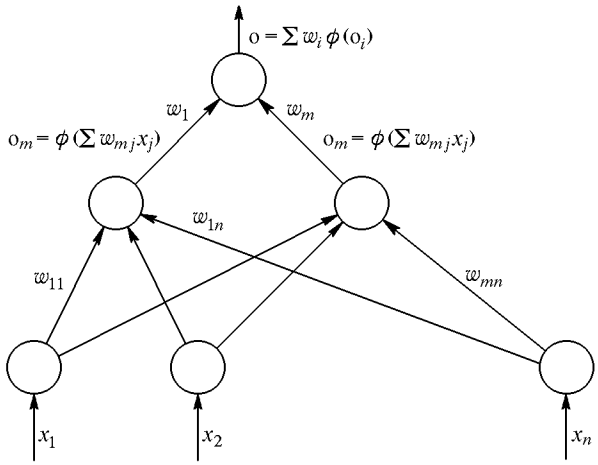

Рис. 2.21. Нейронная сеть Funahashi

для нейронов выходного слоя. На рис. 2.21 представлена НС Funahashi для аппроксимации скалярной функции векторного аргумента.

Отметим еще раз, что приведенная теорема о полноте является далеко не единственной из известных.

Основными недостатками аппарата нейронных сетей являются:

- отсутствие строгой теории по выбору структуры НС;
- практическая невозможность извлечения приобретенных знаний из обученной НС (нейронная сеть практически всегда — «вещь в себе», черный ящик для исследователя).

* * *

Приведенный краткий материал о теории искусственных нейронных сетей является достаточным для перехода к следующей главе. Более подробная информация по структурам, алгоритмам обучения и использованию НС приведена, например, в рекомендуемой литературе.