А.Н.ГОРБАНЬ
Д.А.РОССИЕВ

Нейронные сети на персональном компьютере

РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР
(г. Красноярск)

А.Н. ГОРБАНЬ Д.А. РОССИЕВ

Нейронные сети на персональном компьютере

Ответственный редактор
доктор физико-математических наук *В.И. Быков*

НОВОСИБИРСК
"НАУКА"
СИБИРСКАЯ ИЗДАТЕЛЬСКАЯ ФИРМА РАН
1996

УДК 519.7
ББК 32.971
Г67

Нейронные сети на персональном компьютере / А.Н. Горбань, Д.А. Россиев. — Новосибирск: Наука. Сибирская издательская фирма РАН, 1996. — 276 с.
ISBN 5—02—031196—0.

Изложены методы решения задач с помощью нейронных сетей, реализованных на персональных компьютерах. Приведены примеры решенных задач из различных предметных областей: психологии, социально-политических прогнозов, медицины. Дано описание программного нейроимитатора MultiNeuron.

Книга предназначена для специалистов по информатике, программированию, созданию и применению экспертных систем, а также для широкого круга пользователей, интересующихся нейросетевыми технологиями обработки данных.

Табл. 30. Ил. 39. Библиогр.: 276 назв.

Рецензенты

академик РАН *И.И. Гительзон*,
доктор физико-математических наук *Е.А. Новиков*

Утверждено к печати Вычислительным центром
СО РАН (г. Красноярск)

Книга издана при финансовой поддержке
Сибирского отделения РАН

Г $\frac{1402050000-050}{042(02)-96}$ 64—96 I полугодие

ISBN 5—02—031196—0

© А.Н. Горбань, Д.А. Россиев, 1996
© Российская Академия наук, 1996

ПРЕДИСЛОВИЕ

Нейрокомпьютеры еще не вошли в нашу жизнь. Они находятся где-то на полпути. В мире растет число фирм, уже продающих изделия, называемые нейрокомпьютерами. Сотни лабораторий создают новые варианты машин шестого поколения. Однако весь этот вал работ еще далек от "нейрокомпьютера мечты" — машины, которая подобна мозгу, только лучше него — надежнее и быстрее.

К мечтам можно относиться по-разному, но очевидно, что без них невозможно появление чего-то нового. Сначала — смелая фантазия и прожекты, потом — строгий отбор идей, пробные варианты и, наконец, Проект. До Проекта (с большой буквы) нейрокомпьютера человечество еще не дошло. Разрабатываемые варианты изделий — пока только пробы. Их роль — освоение различных граней идеи нейрокомпьютера, его потенциальных прикладных возможностей. А еще нейрокомпьютерная волна движется стремлением заработать на модной новинке, поэтому не праздным является вопрос "Что останется, когда она спадет и стадо разбежится?" Проблема "сухого остатка" от нынешней бурной деятельности волнует всех, кто всерьез занимается искусственными нейронными сетями. Мы попытаемся ответить на этот вопрос. Ответ неизбежно будет личным и субъективным.

Нейро..., нейро..., нейро... — казалось бы это означает, что компьютеры подобны мозгу и в чем-то его замещают. Но вот незадача — современное движение информатики ведет не к созданию мыслящих машин — партнеров человека, а скорее, к созданию особого, нового компьютерно-информационного мира, в котором люди — и творцы, и жители. Представляя себе этот "дивный новый компьютерный мир", понимаешь, что главное сегодня *не компьютер вместо мозга, а компьютер вместо мира.*

Среди нейрокомпьютерных проб важную роль играют модели нейронных сетей и программные имитаторы нейрокомпьютеров на PC —

нейроимитаторы. Они выполняют различные функции: учебные — позволяют познакомиться с нейронными сетями, исследовательские — дают возможность апробировать новые алгоритмы и архитектуры, развлекательные — представляют собой занимательную интеллектуальную игрушку, наконец, к настоящему времени нейроимитаторы на PC стали конкурентоспособным программным обеспечением для решения ряда прикладных задач.

Появление прикладного уклона неизбежно при имитации нейронных сетей на PC. Нейроимитаторы должны доказать свое право на жизнь и занять определенный сектор рынка прикладных программ. Им нужно выдержать жестокую конкуренцию со стороны более традиционных методов обработки данных, а реализация на PC лишает нейронные сети одного из основных преимуществ — высокой степени параллелизма.

Тем не менее рынок нейроимитаторов складывается успешно. Ежегодно объем продаж возрастает в несколько раз. Появляются и аппаратные средства поддержки — нейросетевые сопроцессоры к PC.

В предлагаемой книге суммирован опыт десятилетней работы с нейроимитаторами, накопленный группой НейроКомп. Мы построили изложение по принципу "от результатов к аппарату": в первой части после общедоступного введения описаны конкретные задачи, решенные на PC с помощью нейроимитаторов.

Вторая часть несколько сложнее и рассчитана на читателя, владеющего математикой хотя бы в объеме курса технического вуза. В ней последовательно описаны элементы нейронных сетей (сумматоры, нейроны, синапсы и т.д.), способы их соединения и настройки для решения различных задач, набор дополнительных элементов, необходимых для создания полноценного нейрокомпьютера, и различные алгоритмы обучения нейронных сетей. Освоив этот материал, грамотный программист может сам создавать различные нейроимитаторы. Несмотря на это, книга ни в коем случае не является сборником рецептов по написанию нейроимитаторов: мы старались сконцентрировать внимание на основных идеях, оставляя в стороне многие важные детали алгоритмической и программистской "кухни".

В приложении дано описание нейроимитатора MultiNeuron.

На отборе материала для книги неизбежно сказались личные пристрастия и опыт авторов. В особенности это касается прикладных задач. Их гуманитарный характер (социология, психология и др.) не случаен.

Благодарности. Большая часть собранных в книге результатов группы НейроКомп — плод коллективного труда. Мы глубоко признательны программистам, участвовавшим в создании программ-нейроимитаторов: С.Е. Гилеву, Е.А. Миркесу, Д.А. Коченову; особая признательность медикам, предоставившим для анализа результаты многолетних исследований и сотрудничавшим с нами в ходе всей работы

по созданию нейросетевых экспертных систем: И.Н. Ананьеву, А.Г. Борисову, Е.В. Бутаковой, С.Г. Вахрушеву, С.Е. Головенкину, С.А. Догадину, И.Г. Захматову, О.П. Зинченко Е.В. Масленниковой, Г.В. Матюшину, А.В. Мызникову, Б.В. Назарову, К.Г. Ноздрачеву, Т.В. Орловой, А.А. Савченко, Н.В. Сухановой, П.А. Чертыкову, А.Г. Швецкому, В.А. Шульману, спасибо также всем сотрудникам и соавторам работ: З.М. Горбачевой, М.Г. Дорреру, А.Ю. Новоходько, Г.М. Садовской, Н.Ю. Сиротининой и другим. Мы благодарны зав. кафедрой медицинского страхования, менеджмента и маркетинга КГМА А.А. Модестову, директору фирмы "АЗА" И.Г. Сулькису, директору Красноярского высшего колледжа информатики Г.М. Цибульскому и директору ВЦК СО РАН В.В. Шайдурову за неоценимую организационную поддержку.

Работа над книгой была частично поддержана РФФИ (грант 95—02—03836—а) и Красноярским краевым фондом "Образование".

Часть 1

НЕЙРОИМИТАТОРЫ И ИХ ПРИЛОЖЕНИЯ

ВВЕДЕНИЕ

1. Область приложения нейроимитаторов

Любая область знаний ставит перед теоретиками и практиками множество задач, часть из которых успешно решается с помощью ЭВМ. Привычный нам процесс решения задачи компьютером достаточно очевиден и состоит из трех основных этапов:

1) человек вводит в компьютер исходные данные (условия задачи);

2) ЭВМ, используя введенные данные, проводит вычисления по заранее установленному алгоритму;

3) ЭВМ выводит результаты вычислений.

Однако основная работа по решению задачи остается за пределами данной схемы. Это поиск алгоритма решения. Всегда ли возможно его построить? Попытка всерьез отвечать на этот вопрос заведет нас слишком далеко. Поэтому примем легкомысленную гипотезу: если задачу как-то может решать человек, то и формальный алгоритм для нее найдется. Вопрос лишь в том, сколько понадобится времени, сил и средств, чтобы это сделать.

Представьте, что вам необходимо алгоритмизировать решение задачи, стоящей перед оператором конвейера на автозаводе в момент аварийной ситуации: останавливать конвейер или нет. Решение задачи — простой выбор одного варианта всего из двух возможных. Однако для составления алгоритма понадобится учесть весь опыт оператора, его знание всевозможных аварийных ситуаций, устройства автомобиля и конвейера, причем готовых формул для этого нет.

Существует целый класс задач, для решения которых человек использует не четкие правила, а опыт. Наличие опыта предусматривает возможное правильное решение и в том случае, если данная ситуация прежде не встречалась. Опытный врач верно поставит диагноз, даже если искажена симптоматика *заболевания и ранее он такую* никогда не наблюдал.

Понятно, что построение алгоритмов для решения подобных задач упирается в сложность или даже невозможность учета всех мыслимых сочетаний факторов (входных данных) и поиска закономерностей, связывающих условия задачи с результатом. Несмотря на быстродействие современных компьютеров, они существенно уступают человеку и животным в таких, казалось бы, простых задачах, как узнавание лиц, предметов, звуковых образов, ориентация в пространстве.

Многие науки и отрасли знания оперируют такими задачами. Это биология, медицина, социология, юриспруденция, криминалистика; некоторые области математики и физики, связанные с распознаванием образов, сигналов и выбором оптимальных решений. Не случайно, что именно в этих областях больше всего ограничено применение ЭВМ, а квалифицированные представители соответствующих профессий за рубежом ценятся и оплачиваются очень высоко.

Поиски алгоритмов, позволяющих автоматически накапливать и затем использовать опыт, продолжаются уже более 100 лет. Естественно, что с появлением дешевых ЭВМ произошел резкий скачок в этой отрасли. Возникла даже новая область деятельности — нейроинформатика (а также нейрокибернетика, нейроматематика и другие "нейронауки"). Приставка "нейро" в названии вовсе не случайна. Во-первых, нейрокомпьютеры и нейропрограммы являются самообучающимися, что отличает их от обычных ЭВМ (программ); во-вторых, принципы их работы напоминают (хотя и в примитивном виде) взаимодействие клеток нервной системы — нейронов — через специальные связи — синапсы.

Основой работы самообучающихся нейропрограмм является нейронная сеть, представляющая собой совокупность нейронов — элементов, связанных между собой определенным образом. Нейроны и межнейронные связи задаются программно на обычном компьютере или могут иметь "материальную" основу — особые микросхемы, какие применяются в специально созданных нейрокомпьютерах. Функционирование нейрона в нейрокомпьютере или нейропрограмме отдаленно напоминает работу биологического нейрона.

Бионейрон — клетка, имеющая длинные отростки, связанные с другими нейронами с помощью синапсов, передающих электрический импульс от одного нейрона к другому. Отростки подразделяются на дендриты, передающие сигналы к данному нейрону, и аксон, передающий сигнал от данного нейрона.

Функционирование же формального нейрона происходит примерно так.

В текущий момент времени нейрон получает по дендритам сигналы от других нейронов. Такие сигналы называются входными. Сигнал с каждого входа умножается на весовой коэффициент этого входа (весовые коэффициенты входов могут быть различными) и складывается с другими сигналами (также умноженными на весовые коэффициенты соответствующих входов). В зависимости от полученного зна-

чения формируется выходной сигнал, передающийся по аксону другим нейронам.

Таким образом, нейронная сеть, получающая на входе некоторый сигнал, способна после прохода его по нейронам выдавать на выходе определенный ответ, который зависит от весовых коэффициентов всех нейронов. Самообучение нейронной сети есть не что иное, как "настраивание" весовых коэффициентов для того, чтобы определенному входному сигналу соответствовал определенный выходной сигнал.

2. Задачи классификации

Из всех типов задач, решаемых на основе предыдущего опыта, довольно значительную группу составляют задачи классификации. Ответом для них является указание класса — выбор одного из нескольких возможных вариантов решения. Приведенная выше задача с оператором конвейера — пример бинарной классификации, когда в качестве решения выбирается один вариант из двух возможных.

Прежде чем перейти к рассмотрению задач, решаемых нейросетями, необходимо кратко остановиться на принципах, по которым создается обучающая выборка — набор данных, служащих для приобретения нейросетью опыта.

Обучающая выборка представляет собой обыкновенную базу данных — электронную таблицу (табл. 0.1). Она состоит из строк (записей), в которых содержатся обучающие примеры, и столбцов (полей), представляющих собой структуру входных параметров.

Здесь показан простейший демонстрационный пример выборки для обучения сети распознавать по ответам на пять вопросов пол респондента. Выборка состоит из шести примеров и пяти входных параметров. Входные параметры каждого примера — это условия задачи (ответы на вопросы). Они представлены в численном виде, причем условия в форме ДА — НЕТ также должны кодироваться числами (1 — 0). Каждый пример имеет параметр "класс", представляющий ЗАРАНЕЕ ИЗВЕСТНОЕ решение данного примера (кто респондент — мужчина

Таблица 0.1

Пример простейшей обучающей выборки

Заранее известный ответ (класс)	Набор входных параметров				
	A. Длинные волосы	B. Рост	C. Масса	D. Любит цветы	E. Сколько раз дрался
1	0	175	72	0	20
1	1	186	78	0	17
1	0	168	72	*1*	*26*
2	1	158	56	1	4
2	1	172	67	1	6
2	0	154	52	0	17

или женщина; мужчина кодируется цифрой 1, женщина — 2). Количества примеров каждого класса не обязательно должны быть одинаковыми, но желательно, чтобы они различались не слишком сильно. Можно заметить, что по какому-нибудь одному параметру судить о поле респондента не представляется возможным (мужчина может иметь длинные волосы, а женщина драться такое же количество раз, как и один из мужчин). Вывод можно сделать только при оценке нескольких параметров сразу, и этот вывод делается на основании опыта.

Цель обучения нейросети — приобрести этот опыт, поработав с данной обучающей выборкой. По ответам на пять вопросов обученная сеть должна делать вывод о том, какой пол имеет респондент.

3. Как обучается нейросеть

Обучение представляет собой автоматический поиск закономерности между совокупностью обучающих данных и заранее известным результатом. Очень упрощенно его можно представить в виде некоторых процессов, происходящих в пространстве выходных сигналов. Размерность этого пространства равна числу классов в задаче. В приведенном примере с бинарной классификацией рабочее пространство есть квадратный участок плоскости (двумерного пространства), минимальные и максимальные координаты которого ограничены значениями -1 и 1 (рис. 0.1).

Пространство разделяется на две части диагональной линией. В текущий момент времени параметры очередного примера подаются на входные нейроны нейросети. После преобразований весь пример в целом проецируется в определенную точку пространства и в зависимости от значений параметров попадает в одну из областей, разделенных диагональю. Область каждого класса в пространстве соответствует как можно большим значениям $(+1)$ "своей" оси координат и как можно меньшим значениям (-1) другой оси координат. В процессе обучения примеры, подаваемые сети многократно, постепенно "растаскиваются" как можно дальше от диагонали и как можно ближе к

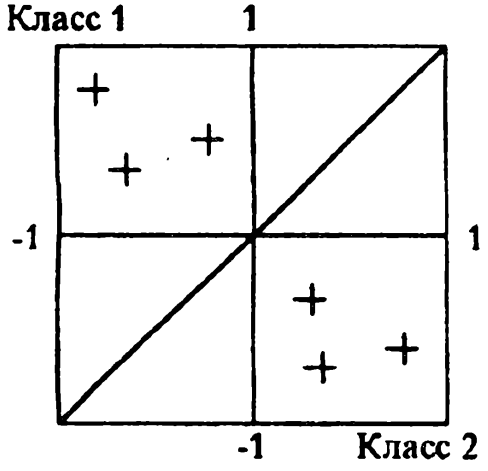


Рис. 0.1. Схема представления примеров в двумерном пространстве при бинарной классификации.

Таблица 0.2

Фрагмент обучающей выборки с одинаковыми примерами, принадлежащими разным классам

Класс	A. Длинные волосы	B. Рост	C. Масса	D. Любит цветы	E. Сколько было драк
1	0	175	72	0	20
2	0	175	72	0	20

углам, каждый из которых соответствует определенному классу. Чем ближе точки примеров к углам, тем лучше обучена сеть. По крайней мере, необходимо добиться такого положения, чтобы все примеры одного класса попадали по одну сторону диагонали, а примеры другого класса — по другую. В пакете программ бинарной классификации CLAB такая схема прямо выводится на дисплей и дает возможность наблюдать за обучением. Если число классов в задаче равно трем, обучение можно описать как такое же растаскивание точек по углам в трехмерном пространстве, хотя принцип останется тем же.

Бывают ли ситуации, когда сеть не может обучаться? Да. Это происходит в том случае, когда на определенном этапе обучения исчерпываются дальнейшие возможности поиска закономерностей между обучающими параметрами и результатами. Простейшая ситуация — когда два примера с совершенно одинаковыми наборами параметров подаются сети как принадлежащие различным классам (табл. 0.2).

Естественно, оба этих примера всегда попадут в одну и ту же точку в пространстве, их невозможно будет отделить друг от друга, и обучение остановится. Программа, управляющая нейросетями, сигнализирует об окончании процесса, причем указывает, полностью ли обучилась сеть.

Обученная нейросеть автоматически записывается на диск компьютера как обыкновенный файл и может храниться там сколько необходимо. В любой момент времени можно считать сеть с диска и продолжить обучение решению данной задачи со старой или новой обучающей выборкой. Одна нейросеть обучается решать только одну задачу классификации, однако может использовать разные обучающие выборки. Они могут различаться по количеству примеров, но должны соответствовать друг другу по числу обучающих параметров и их порядку, числу классов, а главное, по смыслу.

4. Тестирование (работа обученной нейросети)

Тестирование примеров нейросетью может иметь две цели:

1) проверка того, как обучилась нейросеть;
2) решение конкретных задач.

В первом случае осуществляется тестирование выборки с заранее известными ответами примеров. Таким образом можно проверить, правильно ли сеть определяет классы всех примеров и насколько уверенно она это делает. Определенный сетью класс примера сравнивается с заранее известным. Как правило, сначала тестирование проводят на той выборке, на которой сеть обучалась. Если она обучилась полностью, то при тестировании той же самой обучающей выборки классы всех примеров будут определяться правильно. Гораздо больший интерес представляет тестирование аналогичной выборки с заранее известными ответами, но примеры которой не участвовали в обучении сети. Неправильное определение классов некоторых примеров может быть вызвано следующими причинами.

1. Выборка, по которой обучалась нейросеть, недостаточно полно отражает картину соответствия классов обучающим параметрам, иначе говоря, в обучающей выборке слишком мало примеров.

2. Выборка, по которой обучалась нейросеть, составлена тенденциозно. Это означает, что для обучения подбирались примеры, которые, по мнению исследователя, "являются самыми яркими представителями своего класса". Это серьезная ошибка. При такой выборке, конечно, сеть будет обучаться намного лучше, но способность ее к тестированию других примеров существенно снижается. В действительности необходимо обучать сеть на реальных данных, какими бы противоречивыми они не были. Если сеть не сможет обучиться полностью, можно применить некоторые меры, о которых рассказано ниже.

3. Параметры, с помощью которых описываются данные, недостаточно полно отражают реальные ситуации. Приводимый выше пример с определением пола респондента имеет явно недостаточное количество параметров для хорошего обучения. Несмотря на то что сеть вполне обучится на представленных шести примерах, реальные ситуации могут совершенно не вписаться в данную схему. Придется увеличивать число примеров в выборке, которые иначе неизбежно начнут противоречить друг другу.

4. При создании сети не оптимально были выбраны некоторые сетевые параметры, в частности характеристика нейронов сети. Каждый нейрон включает нелинейный преобразователь сигналов вход — выход. В наших программах он имеет вид

$$f(A) = \frac{A}{h + |A|},$$

где h — характеристика, определяющая крутизну нелинейного преобразования (чем меньше h, тем круче график преобразования).

Параметр h задается при инициализации нейросети и его рекомендуемые для нейросетевых классификаторов значения лежат в интервале 0,1—0,8. Чем меньше характеристика, тем лучше обучается сеть, *но тем хуже она распознает* примеры, не включенные в обучающую

Таблица 0.3

Тестируемые примеры, класс которых неизвестен

Класс	A. Длинные волосы	B. Рост	C. Масса	D. Любит цветы	E. Сколько было драк
0	1	164	70	1	6
0	0	175	72	0	20

выборку, и наоборот, чем характеристика больше, тем труднее идет процесс обучения, но выше интер- и экстраполяционные способности сети. При тестировании примеров с известными решениями нейросеть имеет возможность проверить правильность своего решения, о чем сообщает, одновременно выводя заданный класс примера и тот, который она вычислила.

5. Задана неверная классификационная модель. Возможно, на самом деле примеры группируются в три класса, а вы задаете только два. Ниже рассказано о методе коррекции классификационной модели.

При решении конкретных задач сети подаются примеры, класс которых неизвестен (принимается равным нулю) (табл. 0.3.). В этой ситуации сеть не может проверить правильность решения.

При ответе нейросеть не только выдает результат — класс тестируемого примера. В отличие от экспертных систем, работающих по четким правилам, решение задачи на основе опыта всегда имеет нечеткий характер, т.е. включает элементы неопределенности, сомнения. Это очень похоже на принятие решения человеком, при котором всегда существует возможность ошибки, пусть даже очень малая. Поэтому кроме класса тестируемого примера сеть вычисляет процент уверенности в данном решении. Это выглядит так: в случае бинарной классификации сеть сообщает класс тестируемого примера (например, 1-й) и две процентные величины для каждого класса (например, за то, что это 1-й класс, — 87 %, 2-й класс — 13 %). Если задача допускает три класса, решение может выглядеть таким образом: класс 2, за 1-й класс — 5 %, за 2-й — 91 %, за 3-й — 4 %. Сумма процентных показателей всегда равна 100. Удобство такой схемы в том, что всегда видно, какова уверенность сети в данном ею ответе, а также к какому еще классу (классам) близок пример. Естественно, результат {1-й класс, 51 — 49 %} означает, что сеть не может дать вразумительного ответа, так как реальное превышение одной величины над другой составляет всего 2 %.

Из этого вытекает очень полезный для практики вывод. Изменяя в различных направлениях значения параметров примера и повторяя его тестирование, можно видеть, что и насколько нужно изменить, чтобы пример стал принадлежать к требуемому классу. Это может быть незаменимым, например, для медицинской диагностики и прогнозирования. Предположим, что сеть обучена дифференцировать больных и здоровых людей по набору клинических параметров. Изменяя на

компьютере эти параметры, можно добиться, чтобы пример, определяемый как "больной", стал принадлежать классу "здоровый". Таким образом, станет ясно, какие клинические параметры подлежат изменению для улучшения состояния больного.

5. Прогнозирование результата выборов президента США

Данная задача стала "классической" для демонстрации работы нейросетевого классификатора. Она компактна, значения всех обучающих параметров представляются в форме ДА — НЕТ, основана на реальных данных и дает хороший результат [53, 70].

Цель обучения сети — научиться предсказывать, какая партия победит на выборах президента США — правящая или оппозиционная (бинарная классификация).

В качестве обучающих параметров взяты ответы всего на 12 вопросов социологической анкеты. Обучающая выборка состояла из 31 примера, каждый из которых представляет ситуацию выборов начиная с 1864 г.

Класс 1 означает, что в данной ситуации был избран кандидат правящей партии, класс 2 — кандидат оппозиционной партии. Нейронная сеть обучается принимать решение о том, какой кандидат победит, за 2—3 мин работы (на простейшей IBM PC/AT-286). После обучения сеть способна предсказать исход реальной ситуации, которая не входила в обучающую выборку (когда производились эксперименты, результат выборов еще не был известен!): сеть однозначно предсказывает победу Клинтона (94 % — Клинтон, 6 % — Буш). Подробно все эксперименты с данной обучающей выборкой [231—233, 273] описаны в следующей главе.

В процессе обучения нейросеть способна оценить влияние каждого из обучающих параметров на принятие решения. Таким образом, можно выявить информативность каждого параметра по сравнению с остальными. Изучение информативности параметров по выборам президента США выделило самые значимые вопросы. Тестирование, в частности, показало, что шансы Д. Буша были бы существенно выше, если бы не известные события в Лос-Анджелесе. А если бы к тому же президент Буш был в глазах общественного мнения новатором, существенно изменившим политику по сравнению с предшествовавшим президентом Рейганом, то его шансы на повторное избрание стали бы вполне реальными.

6. Распознавание направления сдвига

Для испытаний алгоритмов и программ существует набор стандартных задач. Для нейронных сетей одной из таких стандартных задач стало распознавание направления сдвига последовательности из нулей и единиц.

Таблица 0.4

Распознавание направления сдвига

Число	Двоичный вид	Сдвиг влево	Сдвиг вправо
1	00000001	00000010	10000000
2	00000010	00000100	00000001
3	00000011	00000110	10000001
4	00000100	00001000	00000010
5	00000101	00001010	10000010
6	00000110	00001100	00000011
7	00000111	00001110	10000011
8	00001000	00010000	00000100
9	00001001	00010010	10000100
10	00001010	00010100	00000101

Давайте представим числа от 0 до 255 в двоичном виде с восемью знаками. Затем сделаем с каждым двоичным числом операции циклического сдвига влево и вправо. Если происходит сдвиг влево, у числа самая первая (левая) цифра переставляется в конец, если вправо — последняя (правая) цифра переставляется в начало (табл. 0.4).

Теперь соединим восемь цифр двоичного числа и восемь цифр соответствующего числа, полученного путем сдвига влево. В результате имеем последовательность из 16 цифр (нулей и единиц). То же самое проделаем с исходным числом и числом, полученным путем сдвига вправо. Задача для нейросети — по последовательности из 16 цифр научиться определять, в какую сторону был сделан сдвиг — влево (1-й класс) или вправо (2-й класс). Если взять для обучающей выборки первые 254 целых числа, получим 504 примера (по 252 каждого класса). Числа 0 (00000000) и 255 (11111111) брать нельзя, так же как и 10101010 и 01010101, поскольку при сдвиге в любую сторону для них будет получен один и тот же результат.

7. Коррекция классификационной модели

Достаточно большая нейронная сеть может освоить любую непротиворечивую обучающую выборку, однако, как показывает наш опыт, если достаточно малая нейронная сеть не может обучиться, то из этого можно извлечь полезную информацию. В частности, сеть всего из двух нейронов в тестовом обучении не смогла освоить задачу о распознавании направления сдвига восьмибитовой последовательности — два примера никак "не хотели" правильно распознаваться. Обратившись, наконец, к анализу задачника, мы обнаружили в нем две ошибки — именно те самые два примера. Сети же из 20 и более нейронов успешно осваивали задачник с ошибками — *учились давать неправильные ответы*. Этот казус нас многому научил. Теперь мы всегда выбираем параметры сети так, чтобы она осваивала задачу на пределе своих возможностей.

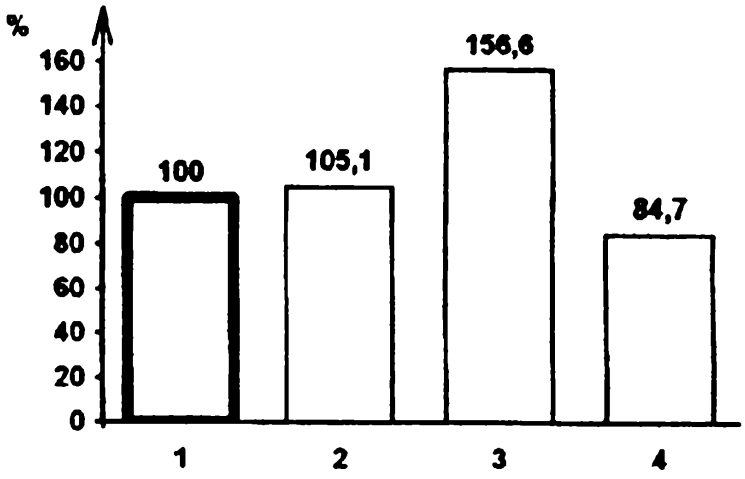


Рис. 0.2. Уровень внутриклеточного метаболизма лимфоцитов.

1 — у здоровых людей, принятый за 100 %, 2 — у больных с иммунодефицитными состояниями, рассчитанный на всю группу; 3 — в подгруппе больных, которые *остались* в процессе обучения; 4 — в подгруппе больных, которые были *исключены* в процессе обучения.

В одном из проводимых исследований [153, 263] мы обучали нейросеть ставить диагноз вторичного иммунодефицита по иммунологическим и метаболическим параметрам лимфоцитов (рис. 0.2). В реальной ситуации по сдвигам этих параметров иногда трудно сделать верное заключение. Досконально обследованы здоровые и больные люди, параметры которых использовались для обучения. Однако нейросеть упорно не обучалась, причем наблюдалась любопытная закономерность: нейросеть хорошо распознавала все до единого примеры здоровых людей, тогда как часть примеров больных путала со здоровыми. В итоге обучение прекращалось. Тогда был сделан следующий шаг: каждый раз, когда сеть останавливала работу, из обучающей выборки автоматически убирали пример, на данный момент самый трудный для распознавания, и после этого вновь запускался процесс обучения. Постепенно из обучающей выборки исключалась примерно треть больных (при этом ни одного здорового!), и только тогда сеть обучилась полностью.

Так как ни один пример из группы здоровых не был исключен из обучения, эта группа не изменилась, тогда как все больные оказались разделены на две подгруппы — оставшиеся и исключенные примеры. После проведения обычного статистического анализа выяснилось, что группы здоровых и группа больных практически не отличаются друг от друга по показателям метаболизма лимфоцитов. Однако получившиеся в результате автоматического разделения сетью две подгруппы больных статистически достоверно отличаются от здоровых и друг от друга по нескольким показателям внутриклеточного метаболизма лимфоцитов. Причем в одной подгруппе наблюдалось увеличение активности большинства лимфоцитарных ферментов по сравнению с таковой здоровых, а в другой подгруппе — депрессия (снижение активности по сравнению со здоровыми). На рис. 2 даны усредненные показатели активности лимфоцитарных ферментов во всех группах. Уровень здоровых для удобства принят за 100 %. Как видно, 1-й и 2-й столбцы диаграммы практически не различаются, тогда как 3-й и 4-й значительно отличаются от первого. Это демонстрирует исходную неоднородность больных по данным показателям, которая обнаружена именно

нейросетью: больные как с активацией, так и с депрессией клеточного метаболизма (относительно здоровых) были заданы как один класс. На самом деле они представляли две различные группы, а выбранная классификационная модель не учитывала неоднородности всей группы больных.

Необходимо было изменить классификационную модель.

После выделения подгруппы исключенных примеров больных как отдельного класса нейросеть очень быстро обучилась решать задачу. В конечном итоге созданная нейросетевая экспертная система не только определяет наличие иммунодефицитного состояния у обследуемого, но и устанавливает его метаболический вариант (активация или депрессия), связанный с фазой иммунодефицита (наличие или отсутствие компенсаторных реакций). В данной ситуации очень удобно использовать повторное тестирование примера после целенаправленного изменения его параметров, что позволяет, например, смоделировать воздействие различных препаратов на те или иные звенья иммунитета.

8. Автоматическая минимизация числа параметров

При изучении экологической ситуации в различных регионах важным показателем является накопленная доза радиоактивного облучения у проживающих в них людей. В последние годы активно ведется поиск методов, позволяющих проводить скрининг населения в зоне действия радиационно опасных производств, а также работающих на них людей. Существующие методы, как правило, сложны, дорогостоящи, требуют привлечения соответствующих специалистов.

Между тем, даже небольшая накопленная доза облучения заметно влияет на многие органы и системы организма, прежде всего иммунную и эндокринную, которые наиболее чутко реагируют на воздействие внешних факторов. Существующие методы позволяют достаточно тонко и полно оценивать состояние этих систем, однако часто бывает трудно установить, чем вызвано изменение их параметров: наличием повышенной дозы облучения или самостоятельным заболеванием.

В одном из наших исследований была поставлена цель обучить нейросеть по комплексу иммунологических, гормональных и биохимических параметров крови людей устанавливать степень радиоактивного облучения (накопленной дозы).

В качестве обучающей выборки использованы иммунологические, гормональные и биохимические параметры крови (всего 57) у 141 человека, занятых в производстве на многопрофильном предприятии атомной промышленности. У всех обследуемых проводился ежегодный дозиметрический контроль накопленной дозы. По величине этой дозы обследуемые были разделены на три класса. Четвертый класс составили люди, накопленная доза у которых находилась в пределах естественных фоновых значений.

При статистическом анализе обучающей выборки по большинству показателей не выявлялось достоверных различий между классами, имеющиеся же по некоторым параметрам различия не прослеживались во всех классах. Таким образом, сравнивая отдельные показатели, а также используя другие виды статистического анализа, мы не имели возможности адекватно определить заранее неизвестный класс примера.

Однако нейросетевой классификатор, используя весь комплекс параметров, полностью обучился распознавать класс каждого примера (группу, к которой относится обследуемый).

Несмотря на факт обучения сети, на практике представляется затруднительным обследовать человека по 57 параметрам крови для определения накопленной дозы. Поэтому была проведена автоматическая минимизация числа обучающих параметров с помощью нейросети. После полного обучения вычислялся наименее информативный из них. Затем создавалась новая нейросеть, наследующая все нейроны и синаптические связи только что обученной сети, за исключением тех, которые касались малозначимого параметра. Процесс обучения запускался вновь. Так повторялось несколько раз, до тех пор, пока сеть не перестала обучаться. Последняя полностью обучившаяся сеть работала уже только с 35 входными параметрами.

Несмотря на значительное уменьшение количества параметров, степень уверенности в ответах снизилась всего на 1,8 %. Это свидетельствует о том, что исключенные параметры малозначимы для принятия окончательного решения, а компенсировать снижение уверенности в ответах можно, увеличив число примеров в обучающей выборке. Таким образом, выявляется еще одно полезное свойство нейросетей — выделение оптимальных наборов обучающих данных, что на практике может серьезно сократить затраты в различных областях, исключив сбор фактически ненужной информации.

С помощью обученной таким образом нейросети была протестирована группа лиц (140 человек), проживающих в зоне влияния предприятия атомной промышленности, но не занятых на производстве. По данным нейросети, только у 3 человек накопленная доза была в пределах естественного фона; у 51 — определялась слабая доза, у 82 — средняя, и у 4 человек — сильная.

Таким образом, представляется возможным создание экспертных систем, позволяющих проводить скрининговые обследования населения для оценки влияния на него радиоактивного облучения в зонах действия соответствующих производств.

9. Нейросетевые предикторы

Мы рассмотрели свойства и примеры применения нейросетевых классификаторов — нейросетей, дающих в качестве ответа класс — выбор одного из нескольких вариантов. Однако задачи, решаемые на основе предыдущего опыта, могут требовать представления ответа в

виде действительного числа. Например, было бы неплохо, если бы в предыдущем примере нейросеть выдавала в качестве ответа не степень накопленной дозы (1, 2 или 3), а непосредственное значение дозы. Недавно закончены "испытания" только что созданного нейросетевого предиктора — именно такой программы.

По принципу работы предиктор мало отличается от классификатора (по крайней мере внешне). Различие состоит в том, что при обучении вместо заранее известного класса примера подставляется заранее известное действительное число, соответствующее данному примеру. Таким образом, предиктор работает как система, самостоятельно находящая функцию зависимости ответа от нескольких обучающих параметров: $Y = f(x_1, x_2, x_3, ..., x_n)$. После обучения, при тестировании конкретного примера, предиктор вычисляет Y в зависимости от подставленных значений $x_1, x_2, x_3, ..., x_n$.

Одной из тестовых задач для обучения предиктора была аппроксимация полиномов Чебышева. Нейросеть обучалась по точкам, лежащим через определенные интервалы на оси X, находить соответствующие значения Y полиномов. После обучения проводилось тестирование, причем использовались промежуточные точки, не участвующие в обучении. Вычисленный на основании опыта (!) ответ сравнивался с реальным значением функции в данной точке. Абсолютная ошибка лежала в пределах 0,0001—0,001, а относительная составляла в большинстве случаев сотые и десятые доли процента [246].

Следующим логическим этапом разработки нейросетевых программ стало создание "векторного предиктора". В качестве ответа такая нейросеть выдает вектор в многомерном пространстве. На практике это означает, что ответ выводится в виде нескольких действительных чисел. Это позволяет охватить целый класс задач, решением которых является *набор* некоторых действий. Ответ будет содержать указания, какие действия необходимо совершить и с какой "силой".

10. Создание нейросетевых экспертных систем

Кратко резюмируем преимущества возможных нейросетевых экспертных систем перед обычными, а также рассмотрим трудности и проблемы, возникающие на пути их создания.

1. Нейросети принимают решения на основе опыта, приобретаемого ими самостоятельно. "Самостоятельно" в данном случае означает то, что создателю экспертной системы не требуется устанавливать взаимосвязи между входными данными и необходимым решением, тратя время на разнообразную статобработку, подбор математического аппарата, создание и длительную проверку математических моделей.

2. Решение, принимаемое нейросетью, не является категоричным. Сеть выдает его вместе со степенью уверенности в нем, что составляет пользователю возможность критически оценивать ответ.

3. Нейросеть позволяет моделировать ситуацию принятия решения, и это дает возможность посмотреть, "а что будет, если...".

4. Нейросети дают ответ очень быстро (доли секунды), что позволяет использовать их в различных динамических системах, требующих незамедлительного принятия решения.

5. Возможности нейросетей (коррекция классификационной модели, минимизация параметров и др.) позволяют упрощать процесс создания экспертных систем.

Однако на пути создания нейросетевых систем есть и немало трудностей. Самый главный вопрос, который должен возникнуть у читателя, — насколько можно доверять решению нейросети. Ясно, что степень уверенности нейросети в своем решении не является ответом на этот вопрос. На самом деле, в данной ситуации остаются те же проблемы, какие возникают при создании обычных экспертных систем. Создание обучающей выборки, ее корректность, подбор обучающих параметров остаются на совести человека. Немаловажно четкое понимание того, чему же все-таки необходимо обучить нейросеть. Правильная постановка задачи — достаточно сложное дело. Поэтому не следует впадать и в другую крайность — нейросеть не панацея для решения всех проблем, а лишь гибкий инструмент в руках пользователя.

Главным критерием работы нейросетевых экспертных систем должна быть практика — многократные испытания и проверки в самых различных условиях.

Имеется еще одна трудность, выходящая за рамки математических и информационных проблем, — преодоление консерватизма людей, скептически относящихся к "умным" программам. Им можно ответить, что ни одна самая "интеллектуальная" программа, ни ее создатели не пытаются поставить под сомнение компетентность специалистов, а нейросетевое программное обеспечение войдет в повседневную жизнь так же, как привычные теперь всем микрокалькуляторы и текстовые редакторы на персональных компьютерах. Мы надеемся, что данная работа внесет в это свой скромный вклад.

Глава 1

КАК МЫ "ВЫБИРАЛИ" КЛИНТОНА

1. Данные для обучения

Нейроимитатор можно использовать как политического оракула — гадалку, предсказывающую исход выборов.

Какая партия победит на очередных выборах в США — правящая или оппозиционная? На первый взгляд кажется, что это зависит от

Таблица 1.

История президентских выборов в США

Выборы		Ответы на вопросы (у – да, п – нет)											
№	Год	1	2	3	4	5	6	7	8	9	10	11	12
						Победа правящей партии							
1	1864	п	п	п	п	у	п	п	у	у	п	п	п
2	1868	у	у	п	п	п	п	у	у	у	п	у	п
3	1872	у	у	п	п	у	п	у	п	п	п	у	п
4	1880	у	п	п	у	п	п	у	у	п	п	п	п
5	1888	п	п	п	п	у	п	п	п	п	п	п	п
6	1990	п	у	п	п	у	п	у	п	п	п	п	у
7	1904	у	у	п	п	у	п	п	п	п	п	у	п
8	1908	у	у	п	п	п	п	п	у	п	п	п	у
9	1916	п	п	п	п	у	п	п	у	п	п	п	п
10	1924	п	у	у	п	у	п	у	у	п	у	п	п
11	1928	у	у	п	п	п	п	у	п	п	п	п	п
12	1936	п	у	п	п	у	у	у	у	п	п	у	п
13	1940	у	у	п	п	у	у	у	у	п	п	у	п
14	1944	у	у	п	п	у	п	у	у	п	п	у	п
15	1948	у	у	у	п	у	п	п	у	п	п	п	п
16	1956	п	у	п	п	у	п	п	п	п	п	у	п
17	1964	п	п	п	п	у	п	у	п	п	п	п	п
18	1972	п	п	п	п	у	п	у	у	у	п	п	п
						Победа оппозиционной партии							
1	1860	у	п	у	у	п	п	у	п	у	п	п	п
2	1876	у	у	п	у	п	у	п	п	п	у	п	п
3	1884	у	п	п	у	п	п	у	п	у	п	у	п
4	1892	п	п	у	п	у	п	п	у	у	п	п	у
5	1896	п	п	п	у	п	у	п	у	у	п	у	п
6	1912	у	у	у	у	у	п	у	п	п	п	п	п
7	1920	у	п	п	у	п	п	п	у	у	п	п	п
8	1932	у	у	п	п	у	у	п	п	у	п	п	у
9	1952	у	п	п	у	п	п	у	п	п	у	п	у
10	1960	у	у	п	п	п	у	п	п	п	п	п	у
11	1968	у	у	у	у	п	п	у	у	у	п	п	п
12	1976	у	у	п	у	у	п	п	п	п	у	п	п
13	1980	п	п	у	у	у	у	п	п	п	у	п	у

личностей кандидатов и от их программ. Но это иллюзия. И программы, и образы кандидатов в президенты создаются профессионалами.

Оказывается, что если предвыборные кампании всех кандидатов отработаны добросовестно и все участники сделали все возможное, то выбор практически предопределяется объективными признаками ситуации и не зависит ни от программ, ни от личностей, ни от названия партий, но только от того, к какой партии принадлежит правящий президент. В одних ситуациях побеждает правящая партия, в других — оппозиция. А кто победит — можно решать на основании 12 вопросов:

1. Правящая партия была у власти более одного года?

2. Правящая партия получила больше 50 % на прошлых выборах?

3. В год выборов была активна третья партия?

4. Была серьезная конкуренция при выдвижении от правящей партии?

5. Кандидат от правящей партии был президентом в год выборов?

6. Был ли год выборов временем спада или депрессии?

7. Был ли рост среднего национального валового продукта на душу населения >2,1 %?

8. Произвел ли правящий президент существенные изменения в политике?

9. Во время правления были существенные социальные волнения?

10. Администрация правящей партии виновна в серьезной ошибке или скандале?

11. Кандидат правящей партии — национальный герой?

12. Кандидат оппозиционной партии — национальный герой?

Нейроимитатору группы "НейроКомп" для обучения предлагалась история выборов в США за 100 лет — для всех выборов были известны ответы на 12 вопросов и, конечно, какая партия когда победила (табл. 1.1) [249].

После обучения нейронные сети уверенно предсказывали результаты выборов: победу Рейгана, победу Буша над Дукакисом и, наконец, результат последних выборов — победу Клинтона.

2. Результаты тестирования

Тестирование нейроимитатора мы начали с того, что обучали нейронные сети на одной части выборки и проверяли на оставшейся. Поскольку выборы упорядочены по времени проведения, проводилось два эксперимента: предсказание будущего (обучение по первым выборам, предсказание результатов последующих) и восстановление прошлого (обучение по последним выборам, предсказание результатов предшествующих) (рис. 1.1). В обоих экспериментах примеры для обучения включались парами — победа правящей партии + победа оппозиционной партии (для равновесия между классами). По оси абсцисс на рисунках указано количество таких пар.

Скользящий контроль (выбрасывание одной ситуации выборов, обучение по оставшимся и проверка правильности предсказания на исключенном примере) дает 100 % правильных ответов при использовании консилиумов — подготовке 10 нейронных сетей и их "голосовании".

Сколько вопросов необходимо для правильной оценки ситуации? Сколько ответов надо знать, чтобы правильно предсказать результаты выборов? Мы выбрали четыре наиболее важных в среднем вопроса (4, 9, 7 и 8) и повторили для них первый тест — предсказание будущего (рис. 1.2). Как видим, результаты вполне удовлетворительны.

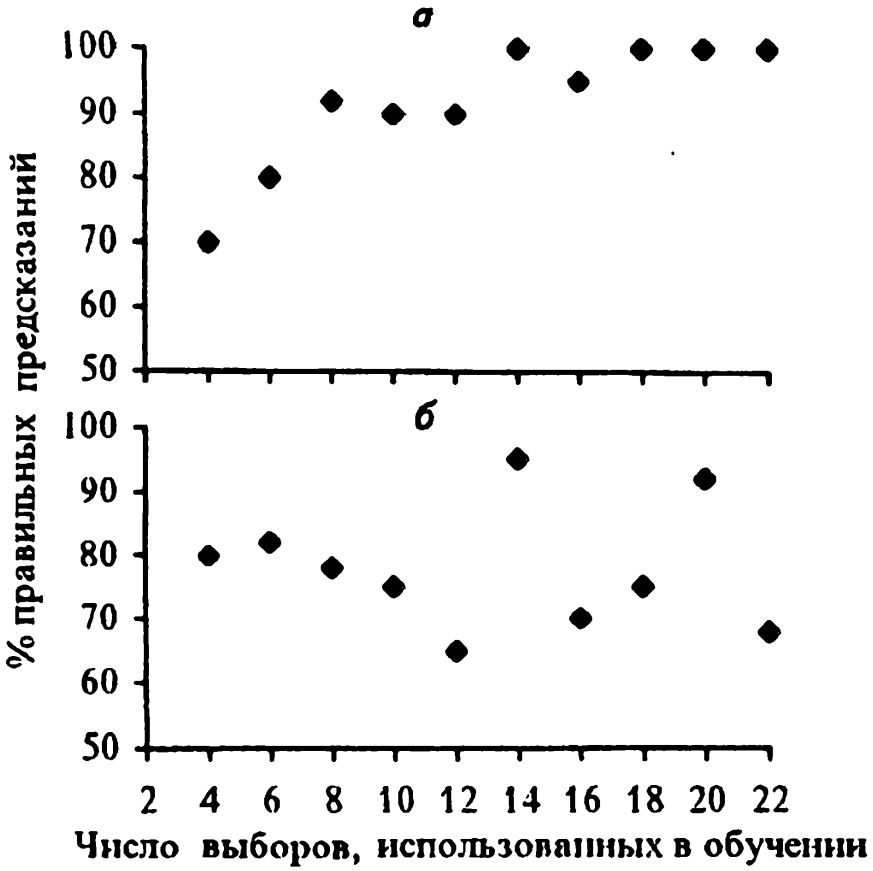


Рис. 1.1. Точность предсказаний обучающей выборки в зависимости от числа пар начальных (*а*) и последних (*б*) выборов.

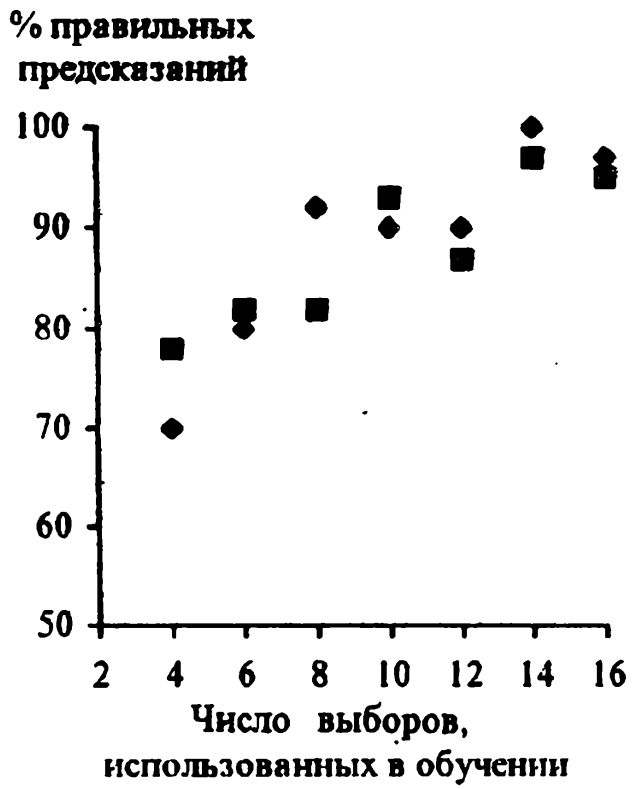


Рис. 1.2. Обучение сети с использованием всех (◆) и четырех самых важных (■) вопросов.

Мы предлагаем наши нейроимитаторы для медицинских и других не менее значимых приложений. Пользователи консультируются с нейроимитаторами по важным вопросам, от решения которых могут зависеть судьбы людей. Предсказание еще неизвестного будущего — лучший тест, поэтому в ходе тестирования нейроимитаторов мы решили доверить им свою репутацию и опубликовать предсказания итогов президентских выборов (Буш — Клинтон, 1992 г.) до их проведения [128, 232].

3. Голосование "консилиума""

Чтобы прогнозировать исход выборов, был собран специальный "консилиум " из 20 нейронных сетей. Дело в том, что нейронные сети в ходе обучения пользуются датчиком случайных чисел и приобретают индивидуальность — "мнения" сетей могут иногда различаться. Со счетом 19:1 консилиум решил, что изберут Клинтона. Необходимо отметить, что по большинству прошлых выборов все консилиумы дают однозначный ответ. Наличие одного нейроэксперта, "проголосовавшего" против Клинтона, означает, что результат выборов 1992 г. был

Таблица 1.2

Результаты голосования 20 нейронных экспертов

Были социальные волнения?	Был активный третий кандидат?		
	Да (реально)	Не знаю	Нет
Да (реально)	1 / 19	4 / 16	12 / 8
Не знаю	9 / 11	19 / 1	20 / 0
Нет	20 / 0	20 / 0	20 / 0

Примечание. В табл. 1.2 – 1.4 числитель – число экспертов, выбравших Буша, знаменатель – число экспертов, выбравших Клинтона.

Таблица 1.3

Результаты голосования 20 нейронных экспертов

Президент Буш произвел существенные изменения в политике	Президент Буш – национальный герой?		
	Да	Не знаю	Нет (реально)
Да	16 / 4	6 / 12	1 / 19
Не знаю	3 / 17	2 / 18	1 / 19
Нет (реально)	1 / 19	0 / 20	0 / 20

Таблица 1.4

Результаты голосования 10 нейронных суперэкспертов и 20 обычных нейронных экспертов

Эксперты	Был ли рост среднего национального валового продукта на душу населения >2,1 %?		
	Да	Не знаю	Нет (реально)
"Суперэксперты"	1/9	0/10	0/10
Обычные эксперты	17/3(!)	5/15	1/19

трудно предсказуем, это подтверждается и большим разнообразием мнений людей-экспертов, высказывавшихся в ходе выборов.

Нам было интересно выяснить, может ли президент Буш переломить ситуацию. Для этого мы исследовали, какими будут результаты выборов при изменении ответов на некоторые вопросы (табл. 1.2, 1.3).

На некоторые вопросы в Красноярске трудно получить ответ, в частности на 7-й. Наш нейроимитатор может готовить "суперэкспертов" для работы в условиях неопределенности — при неизвестных ответах на ряд вопросов. Мы подготовили 10 таких "суперэкспертов". В табл. 1.4 указаны результаты их голосования при различных ответах на 7-й вопрос. Можно видеть, что использование "суперэкспертов" делает прогноз более устойчивым к искажению информации в отдельных вопросах.

Итак, мнения обычных экспертов и "суперэкспертов" в одном месте (отмечено восклицательным знаком) существенно разошлись.

4. Анализ значимости признаков

Нейроимитатор не только делает предсказания, но еще и оценивает важность отдельных признаков. Так, самым важным признаком по мнению всех нейронных сетей оказывается единство правящей партии (4-й вопрос). Если при выдвижении от правящей партии была серьезная конкуренция, то почти наверняка победит оппозиция. Для оппозиции полезно, чтобы оппозиционных партий было несколько — активность третьей партии заметно снижает шансы правящей (3-й вопрос).

Второй по значимости признак — наличие существенных социальных волнений (9-й вопрос), затем идут экономические критерии (6-й и 7-й вопросы). Интересно, что нейронные сети почти всегда считают скандал выгодным для правящей партии.

В табл. 1.5 представлена важность всех 12 вопросов по мнению всех 20 нейронных экспертов. Каждый эксперт оценивал значимость каждого вопроса. В клетках таблицы указано место соответствующего вопроса по мнению соответствующего эксперта. Приведены шесть первых мест. Символы + и − означают, какой ответ повышает шансы

Таблица 1.5

Важность 12 вопросов по мнению различных экспертов

Номер эксперта	Номер вопроса											
	1	2	3	4	5	6	7	8	9	10	11	12
1			5−	2−			6+	1+	3−		4+	
2				2−		6−	4+	1+	3−			5−
3			6−	1−			2+	4+	3−		5+	
4				1−			4+	2+	3−		5+	6−
5			3−		5−			2+	4−		1+	6−
6			3−	1−			6+	2+	4−			5−
7	4−			1−	5−		3+		2+		6−	
8				3−	6+		5+	2+	1−		4+	
9			6−	1−			5+	2+	3−		4+	
10	6−			2−				1+	3−		5+	4−
11	3−		5−	1−	6+	4−			2−			
12	5−		4−	1−			2+	3+				6−
13			6−	1−			4+	2+	5−			3−
14				1−			4+	3+	2−		6+	5−
15			5−	1−			4+	2+	3−		6+	
16	6−			1−			3+	4+	2−		5+	
17				1−		6−	5+	4+	2−			3−
18				1−			5+	3+	2−		4+	6−
19			4−	1−			5+	2+	3−		6+	
20			6−	2−			4+	3+	1−		5+	

Буша (плюс — для правящей партии полезен ответ "да", минус — "нет").

Используя нейроимитатор, составить вопросник для прогнозирования и диагностики не очень сложно — можно сначала взять побольше вопросов, нейрокомпьютер (нейроимитатор) сам расклассифицирует их по важности и сократит список.

Сложнее с набором примеров для обучения — нужны факты. Существует соблазн использовать такой предсказатель для русской политики — по крайней мере, так поступают все студенты, которые получают доступ к нейроимитатору. Однако фактов для прогноза здесь маловато, несмотря на то что многие аналогии напрашиваются.

Проще с набором фактов в задачах медицинской диагностики и прогноза. Здесь традиционно собирают и берегут материал. Сейчас возникают научные коллективы и фирмы, активно берущиеся за создание нейроимитаторов — советчиков врача. Самый крупный известный нам проект — нейрокомпьютерный "двойник", поведение которого предсказывает, что может случиться с человеком при смене обстановки, при том или ином заболевании, стрессе и т.п.

Может быть, со временем такие двойники успешно дополнят обычные медицинские карты.

Глава 2

ПСИХОЛОГИЧЕСКАЯ ИНТУИЦИЯ НЕЙРОННЫХ СЕТЕЙ

1. Проблемы: адаптация тестов и измеренная индивидуальность

С помощью обучаемых нейронных сетей мы приступили к решению двух проблем, возникающих у разработчиков и пользователей компьютерных психологических тестов [80, 208, 209].

1. Культурная, национальная и социальная адаптация тестов. Математическое построение современных объективных диагностических тестов основано на сравнении, сопоставлении выявленного состояния с нормой, эталоном [3]. Однако понятно, что нормы, выработанные для одной социокультурной группы, вовсе не обязательно подходят для другой. В качестве примера можно привести сложности, которые приходится преодолевать при адаптации зарубежных методик.

2. Получение непосредственных рекомендаций и прогнозов, минуя стадию выставления явного диагноза (построения "измеренной индивидуальности" [22]). Явный диагноз — измеренная индивидуальность — желателен для понимания и вербального объяснения ситуации, но при его формулировке теряется огромная часть информации, содержащейся в исходном сыром материале. В ряде психологических работ обсуждается возможность отказа от таких диагнозов.

Если явные диагнозы (любые формы вербального типологического описания) не используются, то рекомендации и прогноз воспринимаются как интуитивные. В таком смысле мы и будем говорить о психологической интуиции нейронных сетей.

Исследование выполнено с применением нейросетевых программных имитаторов группы "НейроКомп" на психологическом материале, собранном в Красноярском гарнизонном военном госпитале и среди студентов Сибирской технологической академии.

Ниже описаны результаты двух типовых экспериментов.

2. Обработка результатов классического тестирования *(эксперимент 1)*

Задача эксперимента 1 — определить, насколько адекватно нейросетевой имитатор сможет воспроизвести результаты типовой психологической методики в постановке диагноза пациенту и научиться ставить классический явный диагноз на основании сырого материала. В качестве такого материала рассматриваются ответы на вопросы стандартного теста.

Для эксперимента выбрана достаточно проверенная клинической практикой психологическая методика ЛОБИ (личностный опросник Бехтеревского института).

Использовался нейроимитатор CLANAss. Пакет программ CLANAss представляет собой программный имитатор нейрокомпьютера, реализованный на IBM PC/AT и предназначенный для решения задач классификации. Данный пакет программ позволяет создавать и обучать нейросеть для того, чтобы по набору входных сигналов (например, по ответам на заданные вопросы) определить принадлежность объекта к одному из n ($n < 10$) классов, которые далее будем нумеровать целыми числами от 1 до n.

Необходимая для обучения выборка составлена из результатов обследования по методике ЛОБИ 35 призывников и военнослужащих, проходящих лечение в Красноярском гарнизонном военном госпитале.

Каждый обследуемый задавался вектором из 162 ответов на вопросы ЛОБИ. Координаты этого вектора равны +1 или −1 (−1 — вопрос не выбран, +1 — вопрос выбран). Для каждого психологического типа строилась сеть-классификатор. Она должна разделять обследуемых людей на два класса: 1-й — тип отсутствует, 2-й — тип диагностируется.

Было создано девять файлов задачника для CLANAss — по одному для каждого из следующих типов: ипохондрического, апатического, эгоцентрического, анозогнозического, параноидального, тревожного, меланхолического, неврастенического, сензитивного, эйфорического (вопросник ЛОБИ позволяет выделять еще несколько других типов, но для них данных в обучающих выборках было недостаточно).

Первоначально по составленным задачникам проведено обучение девяти нейросетей. В качестве базовой модели выбрана сеть из 16 нейронов, а время реакции (количество тактов сети между получением входных сигналов и выдачей результатов) принято равным 10.

При тестировании обученных сетей верное определение новых примеров (отсутствовавших в обучающей выборке) составило всего порядка 60 % от числа заданных.

Для улучшения качества распознавания мы избрали, на первый взгляд, парадоксальный путь — использование нейронной сети с прежними характеристиками, но с количеством нейронов, равным двум (рис. 2.1).

Еще более неожиданным оказался полученный результат. Надежность работы (процент правильно распознанных новых тестовых примеров) превысила 95 %. Не был решен только один тестовый пример. Ошиблась сеть, которая должна определять отсутствие или наличие апатического типа. В примере апатичность отсутствовала, а сеть поставила диагноз "апатический тип присутствует". Единственное внятное объяснение этой ситуации состоит в том, что большинство исследуемых (солдаты срочной службы) как раз и имеют психологическую склонность к наличию данного типа. По всей видимости, условия

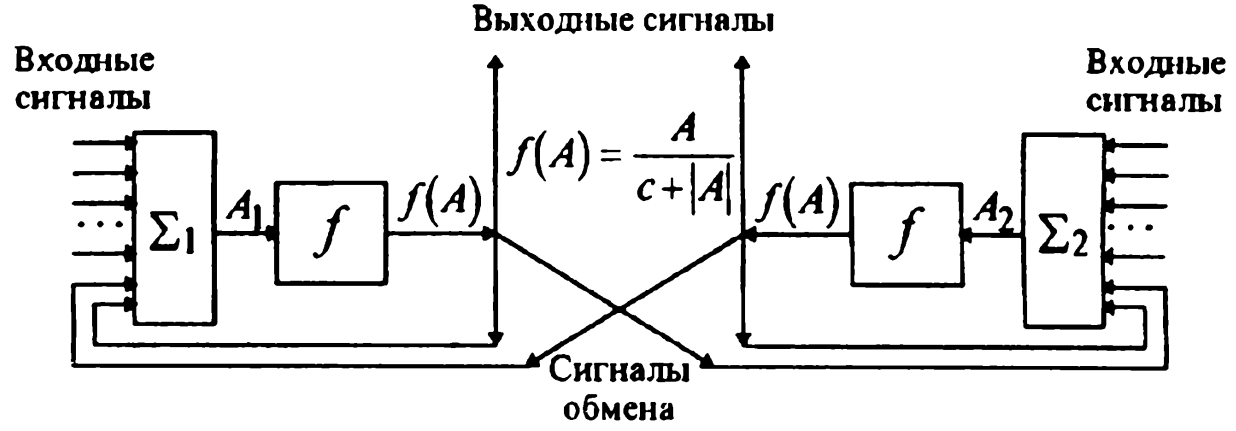


Рис. 2.1. Простейший двухнейронный эксперт.

службы не способствуют ослаблению этой тенденции. Поэтому примеров с *отсутствием* апатического типа оказалось недостаточно для формирования весов синапсов, которые могли бы с достаточной степенью надежности распознать данную ситуацию.

Малые сети лучше обобщают (если их вообще удается обучить). Сеть большой размерности имеет возможность "запомнить" большое количество примеров. При этом она сможет воспроизводить данные примеры, но будет плохо обобщать. По этой же причине меньшая сеть, не обладающая ресурсами для "запоминания" примеров, будет вынуждена учиться обобщению.

Таким образом, нейросетевые имитаторы (а следовательно, и нейронные сети как таковые) позволяют ставить психологические диагнозы. Следует уделять особое внимание подготовке обучающей выборки Ее состав оказывает решающее влияние на качество обученной сети.

Положительное решение данного вопроса открывает широкую перспективу по разработке адаптивных психологических методик. Возможность дообучения сети сводит к минимуму затраты на адаптацию тестов к местным условиям, т.е. для нейросетевых психологических тестов достаточно отобрать примеры диагнозов, не совпадающих с ее прежним опытом, и провести дообучение.

3. Предсказание отношений в группе минуя явные описания *(эксперимент 2)*

После подтверждения гипотезы о применимости нейросетей при оценке отдельной личности был поставлен вопрос о возможности их применения для определения взаимоотношений в коллективе, а также о реальности непосредственной выдачи рекомендаций по коррекции состояния объекта минуя построение "измеренной индивидуальности".

В процессе эксперимента предполагалось подтвердить (или опровергнуть) гипотезу о том, что нейросеть позволяет моделировать взаи-

моотношения в группе людей на основе их психологических особенностей и выдавать прогноз при вхождении в группу нового члена.

В качестве исходных данных взяты результаты двух опросов, параллельно проведенных в одной из студенческих учебных групп.

Входные данные формировались из результатов опроса исследуемых по психологическому тесту из 91 вопроса, объединенных в три группы, назначением которого является определение константных психологических качеств человека. При этом кодировка информации проводилась по принципу "да" = +1, "не знаю" = 0, "нет" = −1. Опросник составлен на основе стандартных методик (MMPI и др.).

Следует обратить особое внимание на количество вопросов — 91. Стандартные методики типа MMPI включают сотни вопросов. Поэтому в ответах человека на них содержатся сотни бит информации. Для надежной постановки явного диагноза ("измерения индивидуальности") эти сотни бит сжимаются до десятка бит (реже — нескольких десятков). Полученный десяток бит и составляет "результат теста" при классическом подходе. Далее на его основе делаются прогнозы и рекомендации. Мы предлагаем прямо использовать весь исходный массив для обучения нейронных сетей. Оказывается, что при таком прямом использовании можно сократить вопросники, оставляя лишь несколько десятков вопросов (в данном эксперименте — 91, возможно и дальнейшее сокращение).

Для формирования выходных данных брались результаты социометрического исследования студенческой учебной группы по вопросу "Насколько бы вы хотели работать по своей будущей специальности с каждым из членов группы?" Ответ предлагалось дать в виде 10-балльной оценки (0 — самое отрицательное отношение к человеку как к будущему сотруднику, 10 — максимально положительное). После сбора анкет был проведен расчет статистических характеристик социометрической матрицы — стеновая оценка статуса и экспансивности каждого из членов группы. Полученные характеристики и использовались в качестве элементов выходного вектора нейросети.

Эксперимент проводили на нейросетевом имитаторе "MultiNeuron" в режиме сети, имеющей на выходе вектор вещественных чисел (подробное описание см. в [19, 44]).

Для решения задачи применялся метод консилиума сетей с различными характеристиками (при этом предполагалось также получить сравнительные характеристики сетей с различным количеством нейронов). Членами такого консилиума стали шесть нейросетей с различными характеристиками (1-, 2- и 5-я — с 64 нейронами; 3-, 4- и 6-я — с 32).

Максимальная погрешность в ответе составляет менее 23 %, что, учитывая специфику предмета исследования, является отличным результатом. Оценка среднеквадратичной погрешности дает еще более впечатляющие результаты — она не превышает 9,4 % (при социо-

метрических исследованиях 10 % считается хорошим результатом и нормой воспроизводимости исследования).

Погрешность в решении тестовых примеров не подчиняется (по крайней мере внешне) какой-либо системе от примера к примеру и от сети к сети, причем сети с 32 нейронами показывают результаты не хуже, чем с 64, так что 32 можно признать достаточным числом нейронов для решения данной задачи.

Таким образом, при помощи нейронных сетей возможно моделировать взаимоотношения в группе людей и прогнозировать взаимоотношения с группой ее нового члена. Это удается сделать без формулировки явного диагноза и построения измеренной индивидуальности. Более того, отказ от явного диагноза приводит к экономии при выполнении самого трудоемкого этапа — получения первичной информации (91 вопрос вместо нескольких сотен!).

Результаты обнадеживают: психологическая интуиция нейронных сетей может служить основой для построения нового поколения методик психологического прогноза и консультирования (выработки рекомендаций). За счет отказа от промежуточной фазы явного диагноза удается сократить количество необходимой исходной информации при сохранении надежности. Нейронная сеть более эффективно использует исходную информацию, чем классические процедуры диагностики. При этом приходится отказаться от понимания ее функционирования на языке привычных и понятных нам психологических типов и других вербальных конструкций, что является неизбежной платой за эффективность.

Приложение

ВОПРОСНИКИ ДЛЯ ПРЯМОГО ПРЕДСКАЗАНИЯ ОТНОШЕНИЙ В ГРУППЕ

ТЕСТ 1

Предлагаемый опросник содержит утверждения или вопросы, касающиеся Вашего характера. С каждым из них Вы можете согласиться или не согласиться. Давайте ответы (“да“ или “нет“) в зависимости от того, подходит ли оно Вам. Заранее благодарны и надеемся, что Вам будет интересно!

1. Легко ли Вы генерируете новые идеи, связанные с работой?
2. Легко ли Вам выполнять работу, требующую длительного внимания и большой сосредоточенности?
3. Испытываете ли Вы чувство беспокойства, что Вас неправильно поняли в разговоре?
4. Любите ли Вы игры в быстром темпе?
5. Быстры ли у Вас движения рук?

6. Вы обычно говорите свободно, без запинок?

7. Легко ли Вы можете найти другие варианты решения известной задачи?

8. Вы когда-нибудь опаздываете на свидание или работу?

9. Часто ли Вам не спится из-за того, что Вы поспорили с друзьями?

10. Нравится ли Вам быстро бегать?

11. Испытываете ли Вы постоянную жажду деятельности?

12. Быстро ли Вы читаете вслух?

13. Если Вы обещали что-то сделать, всегда ли Вы выполняете свое решение независимо от того, удобно это Вам или нет?

14. Все ли Ваши привычки хороши и желательны?

15. Всегда ли Вы платили бы за провоз багажа на транспорте, если бы не опасались проверки?

16. Обычно Вам трудно переключать внимание с одного дела на другое?

17. Трудно ли Вам говорить очень быстро?

18. Дрожат ли у Вас иногда руки во время ссоры?

19. Обычно Вы предпочитаете легкую работу?

20. Медленны ли Ваши движения, когда Вы что-то мастерите?

21. Вы обычно предпочитаете выполнять только одну операцию?

22. Бывает ли так, что Вы говорите о вещах, в которых не разбираетесь?

23. Вы обычно предпочитаете делать несложные дела, не требующие от Вас большой энергии?

24. Сосет ли у Вас под ложечкой перед ответственным разговором?

25. Считаете ли Вы свои движения медленными и неторопливыми?

26. Ваша речь обычно медленна и неторопливa?

27. Склонны ли Вы иногда преувеличивать в своем воображении негативное отношение близких Вам людей?

28. Испытываете ли Вы тягу к напряженной ответственной деятельности?

29. Нравится ли Вам быстро говорить?

30. Легко ли Вы переключаетесь с одной работы на другую?

ТЕСТ 2

Предлагаемый опросник содержит утверждения или вопросы, касающиеся Вашего характера. С каждым из них Вы можете согласиться или не согласиться — в зависимости от того, подходит ли оно Вам. Выбирайте наиболее типичное для Вас и указывайте соответствующую букву. Заранее благодарны и надеемся, что Вам будет интересно!

1. Я предпочитаю несложную классическую музыку современным популярным мелодиям:

а) да; б) не уверен; в) нет.

2. У меня бывают такие волнующие сны, что я просыпаюсь:

а) часто; б) изредка; в) практически никогда.

3. Окружающим известно, что у меня много разных идей, и я почти всегда могу предложить какое-то решение проблемы:

а) да; б) нечто среднее; в) нет.

4. Если бы я сделал полезное изобретение, я предпочел бы:

а) работать над ним в лаборатории дальше; б) трудно выбрать; в) позаботиться о его практическом использовании.

5. Ко дню рождения, к праздникам:

а) я люблю делать подарки; б) затрудняюсь ответить; в) считаю, что покупка подарков — несколько неприятная обязанность.

6. Иногда у меня бывали огорчения из-за того, что люди говорили обо мне дурно за глаза без всяких на то оснований:

а) часто; б) затрудняюсь ответить; в) нет.

7. Мне нравится работа разнообразная, связанная с частыми переменами и поездками, даже если она немного опасна:

а) да; б) верно нечто среднее; в) нет.

8. Я предпочел бы иметь дачу:

а) в оживленном дачном поселке; б) нечто среднее; в) уединенную, в лесу.

9. Я провожу много свободного времени, беседуя с друзьями о тех прежних событиях, которые мы вместе пережили когда-то:

а) да; б) верно нечто среднее; в) нет.

10. Думаю, обо мне правильнее сказать, что я:

а) вежливый и спокойный; б) верно нечто среднее; в) энергичный и напористый.

11. Устаревший закон должен быть изменен:

а) только после основательного обсуждения; б) верно нечто среднее; в) немедленно.

12. Если бы кто-то разозлился на меня:

а) я постарался бы его успокоить; б) не знаю, что бы я предпринял; в) это вызвало бы у меня раздражение.

13. Иногда какая-нибудь навязчивая мысль не дает мне заснуть:

а) да; б) не уверен; в) нет.

14. Мне доставляет удовольствие совершать рискованные поступки только ради забавы:

а) да; б) верно нечто среднее; в) нет.

15. Я делаю людям резкие критические замечания, если мне кажется, что они этого заслуживают:

а) обычно; б) иногда; в) никогда.

16. Если бы я работал в хозяйственной сфере, мне было бы интереснее:

а) беседовать с заказчиками, клиентами; б) выбрать нечто среднее (между а) и в); в) вести счета и другую документацию.

17. Вечер, проведенный за любимым занятием, привлекает меня больше, чем оживленная вечеринка:

а) да; б) не уверен; в) нет.

18. Обычно я могу сосредоточенно работать, не обращая внимания на то, что люди вокруг очень шумят:

а) да; б) не всегда; в) нет.

19. Если начальство или члены семьи в чем-то меня упрекают, то, как правило, только за дело:

а) да; б) не всегда; в) нет.

20. Бывает, что я говорю незнакомым людям о вещах, которые кажутся мне важными, независимо от того, спрашивают меня или нет:

а) да; б) верно нечто среднее; в) нет.

21. При равной продолжительности дня и одинаковой зарплате мне было бы интереснее работать:

а) столяром или поваром; б) не знаю, что выбрать; в) официантом в хорошем ресторане.

22. Когда мною пытаются командовать, я нарочно делаю все наоборот:

а) да; б) нечто среднее; в) нет.

23. Люди относятся ко мне менее доброжелательно, чем я того заслуживаю своим добрым к ним отношением:

а) очень часто; б) иногда; в) никогда.

24. Обычно я спокойно переношу самодовольных людей, даже когда они хвастаются или другим образом показывают, что они высокого мнения о себе:

а) да; б) не всегда; в) нет.

25. При виде диких животных мне становится несколько не по себе, даже если они надежно заперты в клетках:

а) да; б) не уверен; в) нет.

ТЕСТ 3

Предлагаемый опросник содержит утверждения или вопросы, касающиеся Вашего характера. С каждым из них Вы можете согласиться или не согласиться, в зависимости от этого отвечайте "да" или "нет".

1. Часто Вы переходите на другую сторону улицы, чтобы не встретиться с кем-нибудь из знакомых?

2. Иногда Вам очень хотелось навсегда уйти из дома.

3. Дурные предчувствия всегда оправдываются.

4. Вы очень часто не в курсе дел и интересов тех людей, которые Вас окружают.

5. Иногда Вы так настаиваете на чем-нибудь, что люди начинают терять терпение.

6. Временами Вам приходят в голову такие нехорошие мысли, что о них лучше не рассказывать.

7. Нравятся ли Вам "первоапрельские" шутки?

8. Иногда у Вас пропадает или изменяется голос, даже если Вы в это время не простужены.

9. Вам неловко входить в комнату, где уже собрались и разговаривают люди.

10. Вы не раз замечали, что незнакомые люди смотрят на Вас критически.

11. Вы знаете, кто виноват в большинстве Ваших неприятностей.

12. Держитесь ли Вы обычно в тени на вечеринках или в компании?

13. Иногда Вы не уступаете людям не потому, что дело действительно важное, а просто из принципа.

14. Не все Ваши знакомые Вам нравятся.

15. Предпочитаете ли Вы иметь поменьше приятелей, но зато особенно близких Вам?

16. Иногда по нескольку дней Вы не можете отделаться от какой-нибудь пустяковой мысли.

17. Вы часто беспокоитесь о чем-нибудь.

18. Вы совершаете много поступков, о которых потом жалеете (больше и чаще, чем другие).

19. В гостях Вы держитесь за столом лучше, чем дома.

20. Некоторые так любят командовать, что Вам все хочется делать наперекор, хотя Вы знаете, что они правы.

21. Вы предпочитаете не заговаривать с людьми, пока они сами к Вам не обратятся.

22. Вам нетрудно внести оживление в довольно скучную компанию.

23. Когда Вы узнаете об успехах близкого знакомого, у Вас появляется чувство, что Вы неудачник.

24. Самое трудное для Вас — это справиться с собой.

25. Временами Вам так нравилась ловкость какого-нибудь преступника, что Вы надеялись, что его не поймают.

26. Если Вам не грозит штраф, то Вы переходите улицу там, где Вам удобно, а не там, где положено.

27. Вы часто испытываете тягу к новым впечатлениям, к тому, чтобы встряхнуться, испытать возбуждение.

28. Вы склонны принимать все слишком близко к сердцу.

29. Вы любите готовить (пищу).

30. Вы не осуждаете того, кто стремится взять от жизни все, что может.

31. Вы стараетесь избегать конфликтов и затруднительных положений.

32. Вы справляетесь с делом лучше, обдумывая его самостоятельно, а не обсуждая с другими?

33. Ваши родители и другие члены семьи часто придираются к Вам.

34. Временами, когда Вы себя плохо чувствуете, Вы бываете раздражительны.

35. Вы вели дневник.

36. Вы легко смущаетесь.

Глава 3

ОБЗОР МЕДИЦИНСКИХ И ФИЗИОЛОГИЧЕСКИХ ПРИМЕНЕНИЙ НЕЙРОИМИТАТОРА "MultiNeuron"

1. Применение нейросетей для изучения и диагностики иммунодефицитных состояний

Обследованы люди в возрасте 20—45 лет, здоровые (51 человек) и с клиническим диагнозом иммунодефицитного состояния (42 человека). Изучалась активность основных внутриклеточных ферментов лимфоцитов, проводился также анализ состояния клеточного иммунитета. Диагноз иммунодефицита больным ставился на основании клинического обследования. Входными данными при обучении нейросети была обучающая выборка из примеров, каждый из которых представлял собой набор лабораторных параметров одного человека. Диагноз являлся ответом примера (в нашем случае — больной или здоровый). Обучающая выборка включала примеры с заранее известными ответами.

Нейросеть обучалась различать здоровых и больных по 17 метаболическим параметрам лимфоцитов. В процессе обучения она использовала все примеры здоровых лиц, однако из группы больных исключила 30 примеров как не вписавшихся в заданную классификационную модель. Оставшиеся 12 человек были использованы сетью для обучения. После анализа полученных подгрупп обнаружено, что нейросеть на основе показателей активности ферментов лимфоцитов разделила общую выборку лиц с иммунодефицитами на две группы, различающиеся по уровню обмена веществ в лимфоцитах и по типу иммунного ответа.

По всей видимости, оставшиеся и исключенные лица имеют различные фазы иммунодефицитного состояния. В оставшейся группе при депрессии иммунного ответа наблюдается первая стадия компенсаторной реакции иммунной системы, проявляющаяся прежде всего в увеличении активности внутриклеточных ферментов. У исключенных лиц большинство метаболических показателей приближается к аналогичным параметрам здоровых индивидуумов и, вероятно, наблюдается вторая стадия компенсаторной реакции, выражающаяся в активации иммунореактивности [263].

После добавления к обучающей выборке параметров иммунного статуса и задания классификационной модели из трех групп нейросеть полностью обучилась отличать здоровых от больных и определять у больных фазу иммунодефицитного состояния, подтвердив оптимальность данной классификационной модели.

2. Информативность иммунологических, ферментативных и гормональных показателей в модели нейросетевого классификатора "здоровые—лица с иммунодефицитом"

Использование системного анализа с элементами нейросетевой классификации может способствовать разработке принципиально нового подхода в диагностике иммунной патологии. С этой целью обследованы здоровые люди и лица с иммунодефицитами различной этиологии. Диагноз иммунодефицита ставился только на основании клинического обследования, данные иммунного статуса при этом не учитывались. Нейросетевая классификация больных осуществлялась по трем группам параметров: данным иммунологического анализа, уровню НАД(Ф)-зависимых дегидрогеназ лимфоцитов и сывороточной концентрации основных гормонов.

Обнаружено, что по всем трем группам параметров нейросеть обучилась, не исключив из базы данных ни одного человека. Наиболее информативными иммунологическими показателями были процентное содержание супрессорной фракции Т-лимфоцитов, абсолютные концентрации хелперных фракций и сывороточная концентрация IgA. В данном случае нейросеть решила задачу классификации, используя преимущественно значения иммунорегуляторных субпопуляций Т-лимфоцитов. Причем необходимо отметить, что уровни Т-супрессоров и IgA в обследуемых группах людей не различаются. Наиболее информативными параметрами метаболизма лимфоцитов в модели нейросетевого классификатора явились ферменты, отражающие аэробные и пластические процессы, а также вспомогательные и шунтирующие реакции. Высокой значимостью среди гормональных параметров обладают тиреоглобулин, триодтиронин, кортизол и соматотропин.

Таким образом, с помощью нейросетевого классификатора показана возможность осуществления диагностики иммунодефицитного состояния не только по иммунологическим, но и по метаболическим показателям лимфоцитов и уровню гормонов крови.

3. Диагностика и изучение аллергических и псевдоаллергических реакций

Диагностика аллергических реакций не всегда проста. Дополнительная сложность состоит в существовании псевдоаллергических реакций, которые клинически схожи с аллергическими, но имеют

иммунологические отличия. Псевдоаллергические реакции часто встречаются у больных с лекарственной и пищевой непереносимостью и имеют свои особенности патогенеза, течения и лечения.

Целью исследования являлось применение нейросетевого классификатора для создания автоматизированного диагностикума аллергических и псевдоаллергических реакций, а также для изучения патофизиологических особенностей данных реакций. В качестве входных параметров использовали уровни активности НАД(Ф)-зависимых дегидрогеназ и концентрацию ряда метаболитов в лимфоцитах крови.

Исходная классификационная модель содержала три класса: 1) здоровые люди; 2) больные с истинной аллергией; 3) больные с псевдоаллергией.

При обучении нейронной сети с малым числом нейронов выяснилось, что для успешного обучения классификацию следует дополнить — выделить 4-й класс. В него входят лица из всех трех классов, диагноз которым трудно установить на основании данного набора параметров (всего около 17 % обследованных). Лица, принадлежащие к 4-му классу, нуждаются в дальнейшем уточненном обследовании.

Наиболее значимыми из входных параметров оказались уровни внутриклеточных субстратов и кофакторов. Анализ значимости входных параметров позволяет по-новому взглянуть на различия в патогенезе между аллергической и псевдоаллергической реакциями [161].

4. Классификация лиц с аллергиями по уровню метаболических процессов в лимфоцитах

Широкое распространение аллергических заболеваний связано с концентрацией населения в крупных промышленных центрах, загрязнением атмосферы и водоемов продуктами и отходами промышленных производств. Однако лечение данных патологий осуществляется преимущественно симптоматически, так как значительная вариабельность иммунологических показателей при практически полном отсутствии исследований по молекулярным механизмам аллергий не позволяет использовать патогенетические методы терапии.

Гиперактивное состояние лимфоцитов может наблюдаться только при повышении интенсивности их метаболических процессов. Однако при сравнении исходных выборок здоровых людей и с аллергиями у последних установлено снижение субстратного потока по циклу Кребса, частично компенсируемое за счет повышенного окисления продуктов катаболизма липидов и аминокислот и активации вспомогательной дегидрогеназной реакции. Это не позволяет характеризовать обмен веществ в лимфоцитах как активированный, обусловливающий гиперактивное состояние иммунитета. С помощью нейроклассификатора при обучении по модели "здоровые/с аллергиями" из общей группы лиц с

патологией выделены две подгруппы, одна из которых имеет сниженные показатели активности оксидоредуктаз лимфоцитов (исключенные), тогда как в другой подгруппе ("истинно" страдающие аллергиями) установлены нормализация уровней митохондриальных ферментов относительно параметров здоровых людей и повышение активности анаэробных и пластических процессов, что позволяет предположить в ней высокую функциональную активность иммунокомпетентных клеток.

Продемонстрированный метод может не только способствовать изучению метаболических механизмов патогенеза данной патологии, но и являться основой для усовершенствования принципа назначения гипосенсибилизирующей терапии.

5. Прогнозирование возникновения или усугубления застойной сердечной недостаточности у больных с нарушениями ритма сердца

На кафедре терапии № 1 КГМА обследованы 70 больных с различными нарушениями ритма сердца. Всем больным проводили клинический осмотр, электрокардиографию, холтеровское мониторирование, велоэргометрию, эхокардиографию, чреспищеводную кардиостимуляцию. Впоследствии у двадцати из них по поводу частых обморочных состояний, связанных с основной патологией, проведена имплантация электрокардиостимулятора (ЭКС).

Все больные были повторно обследованы через 1—10 (в среднем 7) лет и по результатам обследования разделены на четыре группы:

1) без ЭКС, стадия сердечной недостаточности (СН) не изменилась;

2) без ЭКС, возникли или усугубились симптомы СН;

3) с имплантированным ЭКС, стадия СН не изменилась;

4) с имплантированным ЭКС, возникли или усугубились симптомы СН.

Нейронные сети обучались прогнозированию возникновения или усугубления симптомов СН в зависимости от предполагаемой имплантации ЭКС. Для этого использовались две нейросети, первая из которых обучилась различать группы 1 и 2, вторая — группы 3 и 4. Таким образом, с помощью обученных сетей представляется возможным прогнозировать усугубление или появление симптомов СН в зависимости от данных первичного осмотра больных с различными нарушениями ритма сердца. Для подтверждения такой возможности все больные были протестированы обеими нейросетями. В результате показано, что моделирование имплантации ЭКС у больных 2-й группы значительно снижает прогнозируемую вероятность развития или усугубления СН. Наоборот, моделирование отсутствия ЭКС у больных 3-й группы либо

не изменяет прогноза, либо значительно ухудшает его в зависимости от вида патологии.

Проведенные исследования позволяют моделировать наличие ЭКС у больных с нарушениями ритма сердца для решения вопроса о его имплантации.

6. Прогнозирование сердечной недостаточности у больных с различными поражениями проводящей системы сердца

Обследованы 57 пациентов с различными поражениями проводящей системы сердца (синдром слабости синусового узла (СССУ), блокады левой и правой ножек пучка Гиса и сочетания этих патологий). Всем больным проводили клинический осмотр, электрокардиографию, холтеровское мониторирование, велоэргометрию, эхокардиографию, чреспищеводную кардиостимуляцию. Повторно их обследовали через 1—10 (в среднем 5) лет, в результате чего они разделены на две группы. В первую вошли пациенты, у которых не возникла/усугубилась сердечная недостаточность (СН), во вторую — больные, у которых СН возникла или усугубилась [261].

Целью исследования стал прогноз возникновения/усугубления СН на основе результатов первого обследования. Для создания системы прогнозирования были использованы самообучающиеся нейронные сети-классификаторы. Классификатор обучается решать задачи классификации (в нашем случае бинарной) на основе обучающего массива, состоящего из примеров. В свою очередь, каждый пример (в нашем случае — обследованный пациент) имеет входные параметры (результаты первого обследования) и заранее известный ответ (возникнет/усугубится СН или нет). Цель обучения нейросети — добиться 100%-й правильности распознавания ответа на обучающей выборке. Критерием способности нейросети к экстраполяции опыта является результат теста на примерах с известным ответом, но не входящих в обучающую выборку.

Созданная нейросеть с 20 нейронами полностью обучилась на сформированном массиве за 9940 тактов функционирования (69 циклов обращения к обучающему массиву).

При проверке прогностических возможностей нейросети на тестируемой выборке сеть выдавала правильный прогноз в 73 % случаев. Для улучшения результата теста необходимо увеличение объема обучающей выборки и доучивание нейросети.

7. Прогнозирование осложнений инфаркта миокарда

Прогнозирование осложнений острого и подострого периодов инфаркта миокарда весьма актуально, так как позволяет более целенаправленно проводить профилактические мероприятия [155]. На первом

этапе создания нейросетевой экспертной системы мы выбрали четыре вида осложнений, наиболее опасных, а также наиболее часто встречающихся у больных инфарктом миокарда: фибрилляция предсердий, тромбоэмболия, перикардит и возникновение/усугубление сердечной недостаточности. Задачи прогнозирования каждого из осложнений, за исключением первого, представляют собой бинарную классификацию (возникнет — не возникнет). При прогнозировании фибрилляции предсердий имеется три варианта ответа: осложнение не возникнет, возникнет пароксизмальная форма, возникнет постоянная форма. Для обучения нейронных сетей сформирована выборка, состоящая из данных о 300 больных инфарктом миокарда (ИМ). Использованы следующие обучающие параметры: пол, возраст, сведения о заболеваниях в анамнезе, локализация инфаркта, параметры электрокардиограммы и эхокардиографии, концентрация калия и натрия в крови при поступлении больного в клинику. Результаты наблюдения за возникновением вышеуказанных осложнений стали заранее известными ответами. Были обучены 13 нейросетей-классификаторов с адаптивной матрицей входов. За прогнозирование фибрилляции предсердий (ФП) отвечали 4 сети и по три сети — за прогнозирование остальных осложнений.

Одним из самых опасных осложнений инфаркта миокарда является мерцательная аритмия, чаще всего возникающая в острый и подострый периоды заболевания. Проводимые профилактические мероприятия не всегда оказываются эффективными и своевременными. Для прогнозирования этого осложнения мы применили четыре нейронных сети-классификатора, работающих совместно и образующих "консилиум". Обучающими параметрами стали 36 клинических и функциональных показателей у больных, только что перенесших инфаркт. В дальнейшем у части из них развилась мерцательная аритмия (пароксизмальная или постоянная форма), что стало заранее известным ответом при обучении классификаторов (1-й класс — осложнения нет, 2-й — пароксизмальная форма, 3-й — постоянная форма). Были обучены четыре нейросети: 1-я (главная) обучалась различать все три класса, 2-я — первый и второй классы, 3-я — первый и третий классы, 4-я — второй и третий классы. Все нейросети обучились полностью с достаточным уровнем надежности и за короткое время:

Номер нейроэксперта	Различаемые классы	Общее количество тактов обучения
1	1, 2, 3	93 553
2	1, 2	60 916
3	1, 3	37 110
4	2, 3	15 707

Таким образом, обученные нейроэксперты работают совместно. Они могут с большой точностью предсказывать развитие мерцательной аритмии и ее форму на основании данных, получаемых сразу же после

поступления пациента в клинику с диагнозом инфаркт миокарда. При обследовании больного он сначала тестируется первым нейроэкспертом, а затем, в зависимости от определенного им класса, двумя другими, умеющими различать этот класс.

Аналогичным образом были обучены нейронные сети для прогнозирования остальных выбранных осложнений [262].

При тестировании экспертной системы на 50 примерах, не входящих в обучающую выборку, прогностическая способность колебалась от 74 до 88 %.

8. Определение прогностически значимых медицинских параметров

При решении задач медицинской диагностики и прогнозирования часто бывает необходимо оценить информативность параметров, на основании которых ставится диагноз или делается прогноз. Для решения этой проблемы мы используем нейронные сети, которые с помощью специального алгоритма проводят анализ значимости сигналов, подаваемых на входные синапсы сети [144]. Нейроклассификаторы, с помощью которых была создана экспертная система прогнозирования одного из осложнений инфаркта миокарда (мерцательной аритмии), провели оценку значимости входных параметров. На вход каждого из четырех нейросетевых экспертов для решения данной задачи подается 36 параметров о больном, перенесшем инфаркт: пол, возраст больного, локализация поражения сердечной мышцы, некоторые параметры электрокардиограммы и эхокардиографии, клинические симптомы при поступлении в клинику, препараты, назначенные больному.

По мнению нейросетевых экспертов наиболее значимыми параметрами, влияющими на прогноз осложнения, являются частота сердечных сокращений, регистрируемая по электрокардиограмме, наличие у больного гипертонической болезни, содержание натрия в крови, нарушение сердечного ритма в анамнезе. При моделировании значений этих параметров в прогнозировании с помощью нейросети выявлено, что некоторые из них неоднозначно влияют на риск развития мерцательной аритмии. Так, увеличение частоты сердечных сокращений повышает риск развития постоянной формы мерцательной аритмии и снижает риск развития пароксизмальной (преходящей) формы. Наличие и-стадия гипертонической болезни оказывают противоположное воздействие. Таким образом, кроме практического применения в клинике, данная нейросетевая экспертная система может стать инструментом научных исследований при изучении патофизиологических механизмов заболевания.

9. Прогнозирование непосредственных результатов лечения облитерирующего тромбангиита

Облитерирующий тромбангиит — воспалительное заболевание артерий конечностей, сопровождающееся тромбозом и нарушением их проходимости. Консервативное лечение назначается практически всем больным вне зависимости от стадии заболевания. Оперативные вмешательства проводят при неэффективности такого лечения и/или при исходно значительных нарушениях кровотока. Однако до сих пор практически не разработаны четкие критерии определения необходимости операции и выбора ее метода. Эффективность большинства операций составляет всего около 50—60 %. При этом трудно добиться стойкого эффекта, что чаще всего приводит к ампутации конечности. Перед хирургом стоит задача определить в каждом случае необходимость операции и выбрать оптимальный метод с учетом возможных осложнений.

Непосредственные результаты лечения 140 больных тромбангиитом послужили заранее известными ответами при обучении нейросетей. Обучающими параметрами были пол, возраст, масса тела и рост, данные анамнеза заболевания, клинического, биохимического и иммунологического анализов крови, инструментальных методов исследования (ангиография, ультразвуковые методы) и проведенные методы лечения. Обучено пять нейронных сетей для прогнозирования следующих непосредственных результатов: будут ли какие-либо изменения в состоянии больного после проводимого лечения; уменьшится или усилится ишемия пораженной конечности; будет ли наблюдаться положительная динамика трофических расстройств пораженной конечности; купируются ли боли в покое.

Созданная система позволяет моделировать применение различных методов консервативного и оперативного лечения для конкретного больного и прогнозировать, как будут изменяться результаты в зависимости от того или иного метода [151, 152].

Для теста системы использовались больные с известными исходами заболевания. Каждый пример вначале тестировался нейросетями, прогнозирующими исходы, причем в качестве параметров лечения, проведенного в клинике, брались данные о реально назначенном хирургами лечении. Затем пример тестировался сетями, назначающими оптимальный набор методов лечения. После этого проводился повторный тест нейросетями, прогнозирующими исходы, но теперь в пример подставлялись предполагаемые методы лечения, назначенные нейросетями.

Рассмотрим результаты первого теста.

Критерий	Процент правильных прогнозов
Динамика ишемии	82,9
Регресс трофических расстройств	82,4
Динамика болевого синдрома	88,0
Предполагаемая ампутация	94,3

Результаты теста нейросетей, назначающих лечение. Из 35 больных система признала "трудными" 10. У 12 из 25 (35 примеров минус "трудные") лечение, назначенное нейросетями, полностью совпало с реально назначенным хирургами, и только в одном случае прогноз исходов оказался ошибочным. В остальных случаях наблюдались некоторые расхождения: 10 больным нейросети назначили другое лечение, которое (по результатам анализа хирургов) можно было действительно назначить данным больным. Исходы, прогнозируемые после такого лечения, были такими же, как и в случае реально назначенных лечебных мероприятий. В трех примерах лечение, назначенное сетями, было похоже на реально назначенное, однако превосходило его по мощности. В этих случаях прогнозировались исходы лучшие, чем наблюдавшиеся в реальности.

10. Обучение нейросетей выявлению накопленной дозы радиоактивного облучения

Даже небольшая накопленная доза радиоактивного облучения заметно влияет на состояние многих органов и систем организма, прежде всего иммунной и эндокринной. Существующие методы позволяют достаточно точно оценить состояние этих систем, однако часто бывает трудно установить, чем вызвано изменение их параметров: повышенной дозой облучения или самостоятельным заболеванием [147].

Нами ставились цели найти взаимосвязи между накопленной дозой радиоактивного облучения и комплексом иммунологических, гормональных и биохимических параметров крови людей и создать метод, позволяющий по этим параметрам судить о величине дозы. Для этого применен нейросетевой классификатор. Обучающими примерами являлись вышеуказанные параметры (всего 35) у 141 человека, занятого в производстве на предприятии атомной промышленности. Обследуемым проведен дозиметрический контроль, и по величине накопленной дозы они были разделены на три класса, 4-й класс составили люди, величина дозы у которых находилась в пределах естественных фоновых значений.

При статистическом анализе обучающей выборки по большинству параметров не выявлялось достоверных различий между классами, а имеющиеся различия не прослеживались по всем классам. Однако

нейросеть, используя весь комплекс параметров, полностью обучилась распознавать класс каждого примера. При тестировании на выборке с заранее известными ответами класс определялся правильно в 100 % случаев. Обученной сетью были протестированы 140 человек, проживающих в зоне влияния предприятия атомной промышленности, но не занятых на производстве. По данным нейросети только у трех человек накопленная доза была в пределах естественного фона; у 51 — определялась слабая доза, у 82 — средняя и у 4 человек — сильная.

11. Нейротехнология для ранней диагностики злокачественных опухолей сосудистой оболочки глаза

Меланомы сосудистой оболочки глаза составляют 88 % всех внутриглазных опухолей, но могут легко диагностироваться лишь на поздней стадии развития. Существующие методы выявления заболевания на ранней стадии не вполне надежны, довольно сложны и дорогостоящи.

Предлагаемый способ ранней диагностики меланом хориоидеи включает применение нового лабораторного метода, разработанного в Красноярском межобластном офтальмологическом центре им. проф. Макарова, и интерпретацию получаемых данных нейросетевым классификатором [140].

Лабораторный метод основан на косвенном измерении содержания пигмента (меланина) в ресницах. Полученные данные спектрофотометрии, проведенной на нескольких частотах для каждого глаза, а также некоторые общие характеристики обследуемого (пол, возраст и др.) подаются на входные синапсы 43-нейронного классификатора с ассоциативной матрицей входов. Нейросеть решает, имеется ли у пациента опухоль, и если да, то определяет ее стадию, выдавая, кроме того, процентную вероятность своей уверенности. Даже при подозрении на наличие опухоли больной может быть направлен на дальнейшее углубленное обследование. Таким образом, данная технология может использоваться для скрининговых профилактических обследований населения.

Стартовое обучение нейросетевого классификатора проведено на параметрах 195 обследованных людей с ранними и поздними стадиями опухолей, а также не имеющих опухоли. Нейроклассификатор обладает способностью дальнейшего накопления опыта по мере использования.

12. Нейронные сети для определения вида лазеротерапии после тонзиллэктомии

После удаления миндалин (тонзиллэктомии) многие пациенты нуждаются в продолжении терапии для достижения более быстрых и эффективных результатов. Один из наиболее действенных методов —

лазеротерапия — безболезненный, неинвазивный, легко дозируемый для каждого пациента. Однако не существует универсальных алгоритмов для определения оптимальных параметров лазеротерапии в каждом конкретном случае. Врач назначает эти параметры исходя из собственного опыта, что не всегда может быть оптимальным.

Мы попытались использовать для этого нейронные сети, обучив их на реальных данных. Обследовано 196 пациентов с декомпенсированными формами хронического тонзиллита, которым была проведена тонзиллэктомия. Задачу формулировали следующим образом: на основании 13 параметров пациента выбрать один из четырех видов облучения (гелий-неоновый лазер, инфракрасное некогерентное облучение, инфракрасное лазерное облучение, пульс-инфракрасное облучение). В качестве обучающих параметров использовались следующие данные: 1) день обследования после тонзиллэктомии; 2) форма декомпенсированного тонзиллита; 3) данные фарингоскопии; 4) индекс биофотометрии; 5) индекс баралгезиметрии; 6) индекс физиологического "теста глотка"; 7) индекс термометрии; 8) индекс реографии; 9) индекс биомикроскопии; 10) индекс pH-метрии; 11) гематологический индекс интоксикации; 12) показатель цитологического исследования; 13) комплексный показатель течения раневого процесса.

Созданные с помощью программы MultiNeuron 2.0 13-нейронные нейросети обучались на 84 примерах (каждый класс был представлен 21 примером). Для тестирования обученных сетей использовались 49 примеров, у которых назначенный вид облучения дал эффективный результат. В 47 случаях (95,9 %) ответ нейросетей совпал с заранее известным.

Анализ значимости обучающих параметров выявил, что наиболее важным показателем, влияющим на принятие решения, является индекс баралгезиметрии [272].

13. Нейросетевая дифференциальная диагностика на основе субъективных данных

Как правило, в компьютерных системах медицинской диагностики используются данные объективных методов обследования. Задачей данной работы стала проверка наличия взаимосвязей между субъективной психологической оценкой пациента и медицинским диагнозом с помощью нейронной сети. Ответом служил один из трех диагнозов, связанных с органическими поражениями головного мозга: 1) пост-травматическими, 2) сосудистыми поражениями и 3) эпилептической болезнью. Входными параметрами для диагностики служили данные теста, основанного на использовании ароматов как стимульного материала, которые оценивались обследуемыми по разработанному опроснику. Мы брали методику семантического дифференциала, разработанную Ч. Осгудом (1955 г.), позволяющую охарактеризовать и оценить ароматы. С помощью метода удается объективизировать и ко-

личественно оценить результаты исследования. В основу оригинального теста были положены шкалы, составленные рядом авторов и собственный опыт исследований. Опросник индивидуальной оценки запаха (аромата) состоял из 16 шкал (например, "интенсивный — неинтенсивный", "приятный — неприятный", "грубый — нежный" т.д.). Использовались три аромата — анисового, гвоздичного и розового масла. Обследованы 155 человек (70 — здоровые, 30 — с последствиями травмы головного мозга, 20 — с эпилептической болезнью, 35 — с сосудистой патологией головного мозга).

С помощью программы MultiNeuron 2.0 созданы три нейросети-классификатора, каждый из которых обучался разделять здоровых и больных с соответствующим диагнозом. Для обучения использовались 130 примеров.

При тестировании сетей на 25 респондентах, не участвовавших в обучении, валидность составила: сосудистая патология головного мозга — 88 %, травматическое поражение головного мозга — 56, эпилепсия — 92 %.

14. Нейронная сеть определяет пол человека по метаболическим и гормональным показателям

Определение пола человека по косвенным признакам ("не заглядывая в паспорт и хромосомный набор") является старой и хорошо известной проблемой классификации. В научном фольклоре сложность этой задачи давно осознана. Состояния нормы очень разнообразны, различные нарушения компенсируются и определить пол по физиологическим признакам непросто [160].

С помощью нейросетевого классификатора построена классификационная модель пола человека. Входные параметры модели: метаболические показатели лимфоцитов и уровни основных гормонов крови. Брали также ряд физиологических показателей, которые дают дополнительную информацию о половых особенностях гормональной регуляции метаболизма иммунокомпетентных клеток и формирования физиологического фона организма. В качестве метаболических параметров лимфоцитов изучались уровни НАД(Ф)-зависимых дегидрогеназ и внутриклеточное содержание катехоламинов и серотонина. Единственный входной параметр, имеющий четкие половые особенности, — это тестостерон (мужской половой гормон, вырабатываемый и у женщин, но обычно в более низких концентрациях).

Классификационная модель была построена и успешно протестирована. Анализ значимости параметров показал, что тестостерон занимает только третье место по значимости. На первом месте — отношение тироксина к тироксинсвязывающему глобулину, на втором — содержание катехоламинов в лимфоцитах крови.

15. Нейросетевая система дифференциальной диагностики заболеваний, проявляющихся синдромом "острого живота"

Задача дифференциальной диагностики заболеваний, проявляющихся синдромом "острого живота", классическая в хирургии и требует большого опыта врача, особенно в сложных ситуациях, которые встречаются довольно часто. Неверная диагностика приводит к значительному числу серьезных осложнений, напрасных оперативных вмешательств, неверному выбору методов операции. Неудивительно, что за последние годы создано большое количество компьютерных диагностических систем для решения этой задачи. Главными общими недостатками их являются необходимость фиксированного набора параметров для диагностики и невозможность адаптации к локальным социальным условиям, от которых часто зависит картина симптоматики заболеваний. Поэтому мы поставили задачу создания экспертной системы, способной к самообучению и накоплению диагностического опыта в процессе работы. Для этого применили нейросетевые классификаторы [154].

В качестве параметров, по которым нейросети выставляют диагноз, использованы 138 показателей, учитывающих особенности клинической картины больного, данные лабораторных тестов и анамнеза заболевания. Стартовая обучающая выборка состояла из 150 пациентов, параметры которых определялись в момент поступления в стационар, а диагноз был затем точно установлен. Обучено 10 нейронных сетей возможности дифференцировать пять заболеваний: острый аппендицит, перфоративную язву желудка или двенадцатиперстной кишки, острый панкреатит, острый холецистит и кишечную непроходимость. Особенностью системы является возможность выставления нескольких диагнозов одновременно, так как на практике такие случаи встречаются не так уж редко. В настоящее время планируется расширение возможностей системы для диагностики 11 заболеваний.

16. Нейросетевая экспертная система прогнозирования исходов и осложнений у больных с циррозами печени

Для циррозов печени характерна высокая летальность, связанная с развивающимися в процессе заболевания осложнениями. Центральным вопросом при лечении циррозов является решение об оперативном вмешательстве, поэтому прогноз исходов заболевания и развития осложнений остается актуальным.

Нами обследовано 373 пациента с циррозами печени различной этиологии и степени тяжести. Исследовалось 244 параметра, включающих 12 характеристик заболевания при поступлении больного в стационар, примененный метод лечения больного и 33 лабораторных показателя (клинический и биохимический анализы крови), которые

наблюдались в динамике 7-кратно в течение госпитального периода. Все обследованные были разделены на три группы по исходу заболевания (улучшение состояния, ухудшение состояния, летальный исход) и на две группы по наличию или отсутствию осложнений.

При создании экспертной системы обучены нейросетевые классификаторы в семи наборах. Нейросети одного набора обучались прогнозировать исход заболевания на основании 12 характеристик заболевания, предполагаемого метода лечения и 33 лабораторных параметров при поступлении. Остальные нейросети обучались прогнозу возникновения осложнений на основании вышеперечисленных параметров плюс лабораторные параметры в динамике.

Все нейросети (число нейронов от 10 до 30, число тактов функционирования — 2, характеристика — 0,8) успешно обучены на 273 примерах обучающей выборки, взятых случайным образом. Результаты теста на примерах, не вошедших в задачник, составили от 87 до 94 % правильно распознанных ответов. Проведенный анализ значимости входных параметров показал наибольшую информативность степени тяжести цирроза, динамики содержания альбуминов, билирубина и щелочной фосфатазы в крови. Система позволяет моделировать исход заболевания и возникающие осложнения в зависимости от различных предполагаемых методов лечения.

17. Сравнительное изучение метаболических показателей лимфоцитов у больных гепатитами А и В с помощью нейросетевого классификатора

Ключевая роль нарушения иммунореактивности в развитии многих заболеваний уже не вызывает сомнений. Именно с увеличением группы людей с функциональной недостаточностью иммунитета связывают тенденцию роста такого инфекционного заболевания, как вирусный гепатит. Однако даже при высоком уровне диагностики лечение его затруднено, так как до сих пор остаются неизученными молекулярные механизмы патогенеза. Целью исследования стало изучение информативности уровней метаболических ферментов в лимфоцитах крови у больных гепатитами А и В с помощью нейроклассификатора.

Выявлено, что для осуществления задачи классификации по уровню НАД(Ф)-зависимых дегидрогеназ лимфоцитов в нейросетевой модели "здоровые — больные гепатитом А" из обоих классов были исключены примеры: из группы здоровых — 4 человека (3,5 %), из группы больных гепатитом А — 3 (16,7 %). Обучение по классификационной модели "здоровые — больные гепатитом В" было завершено только после исключения 7 % из группы с патологией. Установлены различия в информативности метаболических ферментов лимфоцитов в данных нейросетевых моделях. Если в первой модели наиболее значимыми параметрами являлись ЛДГ, МДГ, НАДГДГ и НАДФГДГ,

то во второй — соотношение Обр. ЛДГ/ЛДГ, Обр. ЛДГ, ГЗФДГ и НАДФМДГ. Распределение информативности отражает принципиальные различия в обменных процессах иммунокомпетентных клеток у больных вирусными гепатитами. Так, у лиц с гепатитом А высокая информативность ферментов, связанных с аэробными процессами, отражает их относительную недостаточность, в то время как в более аэробных клетках больных гепатитом В нейросетью выделены оксидоредуктазы с анаэробной и пластической функциями.

Таким образом, применение классификатора при изучении ферментов лимфоцитов у больных гепатитами А и В позволило охарактеризовать реакции с недостаточной функциональной активностью, что позволяет исследовать патогенез данных заболеваний.

18. Оценка компенсаторных возможностей головного мозга при его органических поражениях с помощью искусственных нейросетей

Современные методы исследования в нейрохирургии и неврологии позволяют достаточно точно локализовать очаг поражения мозга, но, к сожалению, не могут оценить степень компенсации данного очага у конкретного больного. Нередко случается, что пациенты со значительным объемом поражения, при идентичной локализации, оказываются в большей степени сохранными, нежели аналогичные больные с малым очагом. С учетом такой ситуации уже давно ведутся поиски метода, который позволил бы судить объективно о реактивности мозга у конкретного пациента. В частности, профессор Красноярской медицинской академии В.А. Руднев для этой цели предложил исследовать сенсомоторику методом анализа миограммы репродуктивного тэппинга. Базируясь на данной методике, мы провели углубленное обследование 25 больных с органической патологией (ЦНС), в результате чего появилось много новых параметров, получаемых при чтении миограммы.

Субъективная интерпретация данных миограммы недостаточно надежна, поиск объективных критериев затруднен вследствие достаточно большого количества имеющихся параметров (до 1500 с одного больного) и их значительной вариабельности. Поэтому для оценки данных применены нейросетевые программы. Нейросеть научилась определять восемь показателей, каждый из которых характеризовал степень нарушения воспроизведения ритма в определенном направлении в процентах наличия патологии, что показывало уровень компенсации объемного процесса головного мозга у данного пациента. Нейросетевая интерпретация показателей в динамике позволяет судить о возможности реабилитации больных, определять эффект производимых оперативных вмешательств [149].

* * *

Здесь описана только часть медицинских и физиологических применений нейроимитаторов серии MultiNeuron. Ряд диагностических и

предсказывающих систем на этой основе успешно проходит опытную эксплуатацию в клиниках.

Исследовательские применения нейроимитаторов в медицине не менее важны, чем клинические. Медики-исследователи находят в нейроимитаторах интересного собеседника, и новые вопросы, которые возникнут в результате общения с нейронными сетями, представляются наиболее важными приобретениями.

Глава 4

ПРОГНОЗИРОВАНИЕ ОСЛОЖНЕНИЙ ИНФАРКТА МИОКАРДА НЕЙРОННЫМИ СЕТЯМИ

1. Задача прогнозирования осложнений

Глава содержит подробное описание работы с нейроимитатором на примере важнейшей задачи прогнозирования осложнений инфаркта миокарда.

Нейронные сети (классификаторы и предикторы) применяются для прогнозирования возможности и времени возникновения четырех видов осложнений инфаркта миокарда. Прогноз делается на основании данных о больном, поступившем в клинику с этим диагнозом. В качестве входных параметров нейросети используют данные анамнеза больного, характеристику ИМ, данные электрокардиографического и ультразвукового исследования сердца, показатели содержания калия и натрия в крови.

Инфаркт миокарда — грозное заболевание. Бурное его распространение за последние полвека стало одной из наиболее острых проблем современной медицины. Заболеваемость ИМ остается высокой во всех странах. Особенно это касается городского населения высокоразвитых стран, испытывающего стремительный ритм современной жизни и подвергающегося хроническому воздействию стрессовых факторов, нерегулярного и не всегда сбалансированного питания. Наиболее часто ИМ встречается у мужчин старше 50 лет, особенно работающих в областях с высоким уровнем эмоциональной нагрузки (управленческие должности, связанные с принятием решений, журналистика, вождение автомобиля и др.).

Несмотря на внедрение современных лечебно-профилактических мероприятий, несколько снизивших смертность от инфарктов, она продолжает оставаться довольно высокой. Особенно настораживает высокая смертность в госпитальной стадии (во время нахождения больного в клинике), которая по данным различных авторов составляет от 4 до 20 %. Смертность больных ИМ в Красноярске по данным, приводимым в [97], последние годы держится на уровне 12—15 %,

однако сам автор считает эти цифры заниженными вследствие ненадежной диагностики некоторых видов инфаркта.

Высокая смертность в госпитальный период в значительной мере вызвана развитием различных осложнений.

Самыми частыми осложнениями ИМ являются различные нарушения ритма сердца, возникающие в острый и подострый периоды заболевания. По данным мониторных исследований, различные аритмии наблюдаются у 90 % больных, перенесших ИМ [155]. Именно аритмии бывают чаще всего причиной летальных исходов у больных ИМ.

Примерно у 10 % больных в остром периоде наблюдаются признаки воспаления наружной оболочки сердца — перикардита, который может продолжаться в течение нескольких дней. Сам по себе перикардит редко бывает причиной летальности, однако на его фоне легче развиваются аритмии и нарушения проводимости.

Тромбоэмболические осложнения встречаются примерно у 5—6 % больных ИМ. Чаще всего возникают тромбоэмболии легочной артерии и ее ветвей, сосудов головного мозга, ведущие либо к смерти (30 % больных с тромбоэмболией легочной артерии), либо к серьезным осложнениям (нарушения мозгового кровообращения, инфаркт-пневмонии). Реже наблюдаются тромбоэмболии нижнего отдела брюшной аорты, артерий нижних конечностей, почечных и мезентериальных артерий, артерий селезенки. Довольно часто (по данным Bedell et al., Бостон, 1986; цит. по [97], в 39 % случаев) тромбоэмболические осложнения вообще прижизненно не диагностируются.

Нередко инфаркт развивается на фоне уже имеющейся застойной сердечной недостаточности (ЗСН). В острый и подострый периоды может наблюдаться возникновение или усугубление уже имеющейся ЗСН. Как правило, она возникает у больных с обширным трансмуральным ИМ, повторным ИМ, у лиц пожилого возраста. Развитию сердечной недостаточности способствует артериальная гипертензия. У больных ИМ возникновение/усугубление ЗСН чаще всего происходит на 3—4-й неделе заболевания.

Разрывы сердца являются нечастым, но крайне тяжелым осложнением инфаркта, в большинстве случаев ведущим к смерти больного.

Из других, относительно редко встречающихся осложнений, можно назвать эрозии и язвы желудочно-кишечного тракта, парез кишечника и др.

Насколько велика роль профилактики осложнений, можно судить по данным [97] о причинах смертности больных ИМ в госпитальном периоде. Смертность в первые 24 ч в 37 % случаях была от острой сердечной недостаточности, в 26 % — от разрыва миокарда, в 21 % — от нарушений сердечного ритма. Те же осложнения определяют смертность в период 24 ч — 3 нед: сердечная недостаточность — 44 %, разрыв сердца — 27, аритмии — 16 %.

В настоящее время разработаны схемы лечения больных инфарктом миокарда, включающие профилактику вышеуказанных осложнений. Особое внимание уделяется профилактике нарушений ритма и тромбоэмболий. Однако порой эти профилактические мероприятия оказываются недостаточно эффективными, о чем свидетельствует пока довольно большое количество осложнений.

В связи с этим поиски возможностей прогнозирования осложнений, которые могут возникнуть в госпитальный период, представляются очень актуальными. Прогнозирование необходимо осуществлять при поступлении больного в стационар, сразу же после стандартных методов обследования. Оно должно быть быстрым, проводиться неоднократно в процессе наблюдения за больным по мере поступления новых данных о его состоянии.

Кроме того, как уже говорилось, любое прогнозирование в медицине должно учитывать географические и социальные особенности региона, где проживают люди, поступающие на обследование и лечение. Невозможность полного учета всех возможных сочетаний факторов ставит преграду для разработки универсальных математических алгоритмов прогнозирования осложнений ИМ. Учет же в каждом случае локальных факторов потребует разработки математических моделей для каждого региона или контингента. Использование нейроинформационных технологий частично решает эту проблему [19, 222, 260], обеспечивая динамическое подучивание нейронных сетей по локальным данным, не прибегая к статистическим методам и математическому моделированию. Такое подучивание экспертной системы может проводиться прямо в клинике по мере на:опления данных о больных.

Получение вероятностного прогноза осложнений позволит врачу более целенаправленно проводить профилактику, усилить наблюдение за больным, скорректировать режим физической активности пациента (особенно в предполагаемые сроки возникновения осложнения). Применение нейросетей поможет также более детально изучить роль различных факторов и их сочетаний в развитии осложнений, получить ценные данные для разработки новых схем профилактики и лечения.

При создании первой версии экспертной системы прогнозирования осложнений ИМ мы выбрали четыре вида осложнений, наиболее частых и/или довольно опасных. Это фибрилляция предсердий, перикардит, тромбоэмболические осложнения и возникновение/усугубление хронической СН. К сожалению, такое серьезное осложнение, как разрыв сердечной мышцы, пока еще не вошло в число прогнозируемых по простой причине малого количества данных о таких случаях.

При постановке задачи мы исходили из необходимости прогнозировать возможность возникновения каждого из четырех выбранных осложнений в отдельности и возможного срока его появления, считая от момента поступления больного в стационар. Очевидно, что для прогнозирования каждого осложнения необходима своя нейронная сеть

Рис. 4.1. Основные осложнения ИМ и постановка задач для экспертной системы.

(или группа нейроэкспертов). Одна сеть может давать только один ответ (итог консилиума нейроэкспертов — тоже один ответ), тогда как у одного больного может наблюдаться несколько осложнений одновременно и в разные сроки (варианты ответа не являются взаимоисключающими).

Таким образом, задача разбивается на восемь подзадач, четыре из которых решаются нейросетями-классификаторами (возникновение осложнений), и четыре — нейросетями-предикторами (сроки возникновения осложнений) (рис. 4.1).

2. Обучающая выборка и входные параметры

Рассмотрим обучающие параметры. Для удобства дальнейшего использования каждому из них присвоено мнемоническое название из латинских букв, иногда представляющее перевод названия на английский язык, а иногда — условное сокращение. Если значение параметра представляет собой выбор из нескольких вариантов, после его названия даны числовые значения и соответствующая им расшифровка каждого варианта.

1. AGE	возраст, лет
2. SEX	пол: 0 — женский 1 — мужской
3. ANT-IM	наличие инфаркта передней стенки: 0 — нет 1 — есть
4. LAT-IM	наличие инфаркта боковой стенки: 0 — нет 1 — есть
5. INF-IM	наличие инфаркта нижней стенки: 0 — нет 1 — есть
6. POS-IM	наличие заднебазального инфаркта: 0 — нет 1 — есть
7. PGE-IM	наличие инфаркта правого желудочка: 0 — нет 1 — есть
8. QRS	вид комплекса QRS на ЭКГ: 1 — не изменен (мелкоочаговый, интрамуральный, нетрансмуральный инфаркт) 2 — типа QR (крупноочаговый инфаркт) 3 — типа QS (трансмуральный инфаркт)
9. AN-IM	инфаркты миокарда в анамнезе — количество (0 если не было)
10. AN-STEN	стенокардия в анамнезе давностью... (лет) (0 если стенокардия не наблюдалась)
11. FC-STEN	функциональный класс стенокардии: 0 — нет стенокардии 1 — I функциональный класс 2 — II функциональный класс 3 — III функциональный класс 4 — IV функциональный класс
12. GB	стадия гипертонической болезни: 0 — нет гипертонической болезни 1 — I стадия 2 — II стадия 3 — III стадия
13. ZSN	наличие застойной сердечной недостаточности: 0 — нет 1 — I стадия 2 — IIа стадия 3 — IIб стадия 4 — III стадия
14. AN-ARIT	нарушение ритма в анамнезе:

0 — нет
1 — предсердная экстрасистолия
2 — желудочковая экстрасистолия
3 — пароксизмы фибрилляции предсердий
4 — пароксизмы трепетания предсердий
5 — суправентрикулярная пароксизмальная тахикардия
6 — желудочковая пароксизмальная тахикардия
7 — фибрилляция желудочков
8 — синдром слабости синусового узла (СССУ), проявляющийся синусовой брадикардией
9 — СССУ, проявляющийся синоатриальной блокадой II степени II типа
10 — СССУ, проявляющийся тахи-брадикардией
11 — указание на неизвестный тип нарушения ритма

15. DIS-PROV нарушение внутрисердечной проводимости в анамнезе:
0 — нет
1 — атриовентрикулярная блокада I степени
2 — атриовентрикулярная блокада II степени I типа
3 — атриовентрикулярная блокада II степени II типа
4 — атриовентрикулярная блокада III степени
5 — блокада передней ветви левой ножки пучка Ниса (БПВЛНПГ)
6 — блокада задней ветви левой ножки пучка Гиса (БЗВЛНПГ)
7 — неполная блокада левой ножки пучка Гиса (НБЛНПГ)
8 — полная блокада левой ножки пучка Гиса (ПБЛНПГ)
9 — неполная блокада правой ножки пучка Гиса (НБПНПГ)
10 — полная блокада правой ножки пучка Гиса (ПБПНПГ)
11 — нарушение синоатриальной проводимости
12 — нарушение внутрипредсердной проводимости

16. AN-END наличие эндокринных заболеваний в анамнезе:
0 — нет
1 — есть

17. AN-TROMB наличие тромбоэмболии в анамнезе:
0 — нет
1 — есть

18. AN-BRONC наличие хронического бронхита в анамнезе:
0 — нет
1 — есть

19. AN-OBM наличие обмороков в анамнезе:

0 — нет
1 — есть

20. KALIUM — концентрация калия в крови, ммоль/л
21. HYPO-K — наличие гипокалиемии:
0 — нет
1 — есть
22. NATRIUM — концентрация натрия в крови, ммоль/л
23. HYPER-NA — наличие гипернатриемии:
0 — нет
1 — есть
24. PULSE — частота сердечных сокращений в минуту, регистрируемая по ЭКГ
25. P — длина зубца P на ЭКГ, сек
26. BIG-P — наличие аномально удлиненного зубца P на ЭКГ:
0 — нет
1 — есть
27. QT — длина интервала QT, сек
28. LP — размер левого предсердия по ЭХО-КГ, мм
29. BIG-LP — увеличение левого предсердия:
0 — нет
1 — есть
30. LG — размер левого желудочка по ЭХО-КГ, мм
31. BIG-LG — увеличение левого желудочка:
0 — нет
1 — есть
32. PG — размер правого желудочка по ЭХО-КГ, мм

Параметры с ANT-IM по PGE-IM охватывают зоны возможной локализации инфаркта. Разделение параметра локализации поражения сердечной мышцы на пять параметров объясняется возможностью наличия у больного поражения сразу нескольких зон. Параметр QRS выражает тип зубца Q на ЭКГ, который может оставаться не измененным (при мелкоочаговом, интрамуральном, нетрансмуральном инфаркте); принимать форму QS, характерную для трансмурального инфаркта, или QR (крупноочаговый инфаркт). Параметры AN-ARIT и DIS-PROV выражают основные нарушения ритма и проводимости в анамнезе. Несмотря на то что у больного может быть одновременно несколько нарушений ритма (или проводимости), решено выделять только основное нарушение.

Как видно, большинство параметров могут принимать всего два значения — 0 (нет признака) и 1 (есть признак). Следует обратить внимание на пары параметров KALIUM — HYPO-K, NATRIUM — HYPER-NA, P — BIG-P-ECG, LP — BIG-LP, LG — BIG-LG. Учитывая, что первые параметры в парах изменяются плавно, а наличие сдвигов существенно для прогноза, было решено ввести дополнительные качественные параметры, фиксирующие наличие выраженных

Таблица 4.1

Количество больных ИМ по классам в каждой из четырех выделенных по осложнениям подгрупп

Осложнение	Количество
Фибрилляция предсердий:	
нет	217
пароксизмальная форма	60
постоянная форма	23
Перикардит:	
нет	172
есть	118
Тромбоэмболические осложнения:	
нет	242
есть	58
Возникновение или усугубление сердечной недостаточности:	
нет	144
есть	156

отклонений в данных по принципу "Нет отклонения (0) или есть отклонение (1)". Проведенное впоследствии обучение нейросетей с такими дополнительными параметрами и без них подтвердило обоснованность такого подхода.

Для параметра концентрации калия (KALIUM) значения 4 ммоль/л и более соответствуют значению "0" параметра HYPO-K, если же калий снижается менее этого числа, параметр HYPO-K принимает значение "1" (есть отклонение). Для натрия это число составляет 150 (при увеличении концентрации натрия HYPER-NA принимает значение "1"); для параметра P-ECG — 0,1 (при удлинении зубца параметр BIG-P-ECG принимает значение "1"). Параметры BIG-LP и BIG-LG принимают значение "1" при увеличении соответственно размера левого предсердия более 36 мм и левого желудочка более 55 мм. Для правого желудочка мы не использовали аналогичный дополнительный параметр, так как при инфаркте миокарда (при отсутствии патологии, вызывающей перегрузку правых отделов сердца) размер правого желудочка, как правило, не выходит за пределы нормы, однако сам его размер может иметь значение для прогноза.

Стартовая обучающая выборка для создания экспертной системы составила 300 человек, поступивших в отделение реанимации и интенсивной терапии Красноярской городской больницы № 20 с диагнозом "инфаркт миокарда". Все вышеперечисленные обучающие параметры при поступлении каждого больного фиксировались в отдельной анкете.

Если в течение срока госпитализации у больного развивалось какое-либо из описанных осложнений, эта информация также вносилась в анкету с указанием дня его возникновения. Вся информация из анкет переносилась в базу данных. Распределение больных по осложнениям представлено в табл. 4.1.

3. Прогнозирование факта и времени возникновения фибрилляции предсердий

По данным разных авторов фибрилляция предсердий (ФП) наблюдается у 17—28 % больных [155, 188, 186], причем несколько чаще у женщин, чем у мужчин (24,5 и 10,6 % соответственно) [126]; при повторном инфаркте миокарда, чем без него (25,9 и 14,1 %). ФП чаще развивается при наличии у больных сердечной недостаточности, чем без нее (24,8 и 7,3 % соответственно). С возрастом вероятность осложнения увеличивается. Четкой связи с локализацией поражения сердечной мышцы не выявлено, однако замечено, что частота возникновения ФП больше при обширных (переднезадних) и глубоких инфарктах зависит от расширения левого предсердия, степени гипертрофии миокарда левого желудочка, выраженности обструкции пути оттока [107].

Одними из частых предвестников ФП являются предсердная экстрасистолия и синусовая тахикардия. По данным [126], синусовая тахикардия предшествовала развитию ФП в 96,3 % случаев.

Наиболее опасное время развития этого осложнения — первые часы после начала заболевания. Ранние пароксизмы ФП связаны с закупоркой левой огибающей коронарной артерии проксимальнее места отхождения от нее левопредсердной ветви и с нарушением кровотока в артерии атриовентрикулярного узла [7]. Причины пароксизмальной ФП в более позднее время могут быть различными: повышение конечно-диастолического давления в левом желудочке и митральная регургитация, психогенные, нервно-рефлекторные воздействия, расстройства электролитного (гипокалиемия) и кислотно-основного равновесия, воздействие лекарств [6].

У пациентов с ФП летальный исход наблюдается достоверно чаще, чем у больных без ФП (36,7 и 17,1 % соответственно) [126]. Отмечается, что при появлении ФП в первые сутки заболевания исход был лучше, чем при более позднем ее возникновении. Летальность в этих группах составила соответственно в 11,8 и 65 % случаев [126].

Возникающая аритмия подразделяется на две формы — пароксизмальную и постоянную. Фибрилляция предсердий считается пароксизмальной, если держится не более 14 дней. Наличие ФП дольше этого периода считается постоянной формой [169]. Понятно, что такое разделение достаточно условно, однако мы решили не пренебрегать возможностью дифференцировать патологический процесс и использовать нейронные сети в качестве инструмента для исследования сходства и различия между этими формами.

В табл. 4.2 приводятся результаты статистической обработки показателей больных, у которых не возникла или возникла ФП (пароксизмальная или постоянная форма).

Наибольшее количество различий наблюдается между группами 1 и 3, а также 1 и 2 (9 и 8 соответственно). Между группами 2 и 3 имеются различия только по трем параметрам. Как и следовало ожи-

Таблица 4.2

Средние значения и их ошибки ($M \pm m$) или частоты (%) параметров у больных ИМ

№	Параметр	Группа 1, n = 217	Группа 2, n = 60	Группа 3, n = 23	Достоверность различия
1	2	3	4	5	6
1	AGE	59,6 ± 1,8	63,9 ± 1,8	67,1 ± 2,3	—
2	SEX				$P_{1,3} < 0{,}01$
	0	32,3 %	45,0 %	60,9 %	
	1	67,7 %	55,0 %	39,1 %	
3	ANT-IM	67,3 %	60,0 %	56,5 %	
4	LAT-IM	53,0 %	28,3 %	47,8 %	$P_{1,2} < 0{,}00$
5	INF-IM	35,0 %	40,0 %	47,8 %	—
6	POS-IM	14,8 %	16,7 %	8,7 %	—
7	PGE-IM	2,3 %	1,7 %	—	—
8	QRS				
	1	11,6 %	5,0 %	4,4 %	—
	2	31,3 %	50,0 %	56,5 %	$P_{1,2} < 0{,}01$ $P_{1,3} < 0{,}02$
	3	57,1 %	45,0 %	39,1 %	—
9	AN-IM	0,39 ± 0,05	0,38 ± 0,12	0,35 ± 0,14	—
10	AN-STEN	2,14 ± 0,17	2,22 ± 0,35	2,26 ± 0,60	—
11	FC-STEN				
	0	43,3 %	43,3 %	39,1 %	—
	1	51,6 %	48,3 %	60,9 %	—
	2	5,1 %	8,4 %	—	—
12	GB				
	0	41,0 %	30,0 %	47,8 %	—
	1	1,4 %	—	—	—
	2	46,5 %	51,7 %	43,5 %	—
	3	11,1 %	18,3 %	8,7 %	—
13	ZSN				
	0	41,9 %	25,0 %	4,4 %	$P_{1,3} < 0{,}001$ $P_{2,3} < 0{,}05$
	1	5,1 %	8,3 %	17,4 %	$P_{1,3} < 0{,}05$
	2	29,9 %	36,7 %	34,8 %	—
	3	6,5 %	5,0 %	4,4 %	—
	4	16,6 %	25,0 %	39,0 %	$P_{1,3} < 0{,}01$
14	AN-ARIT				
	0	98,2 %	71,7 %	30,4 %	$P_{1,2} < 0{,}001$ $P_{1,3} < 0{,}001$ $P_{2,3} < 0{,}001$
	2	—	1,7 %	—	—
	3	—	13,3 %	60,9 %	$P_{2,3} < 0{,}001$
	6	0,5 %	—	—	—
	9	—	1,7 %	—	—
	11	1,3 %	11,6 %	8,7 %	$P_{1,2} < 0{,}001$ $P_{1,3} < 0{,}02$

Окончание табл. 4.2

1	2	3	4	5	6
15	DIS-PROV				
	0	66,8 %	53,3 %	65,2 %	—
	1	4,6 %	6,7 %	—	—
	2	0,5 %	—	—	—
	3	0,5 %	—	—	—
	4	1,4 %	1,7 %	—	—5
	5	12,9 %	18,3 %	8,6 %	—
	6	1,8 %	—	—	—
	7	0,5 %	—	—	—
	8	0,5 %	8,3 %	17,4 %	$P_{1,2} < 0{,}001$ $P_{1,3} < 0{,}001$
	9	8,3 %	6,7 %	4,4 %	—
	10	2,3 %	5,0 %	4,4 %	—
16	AN-END	11,5 %	6,7 %	—	—
17	AN-TROMB	13,8 %	11,7 %	26,1 %	—
18	AN-BRONC	15,2 %	16,7 %	17,4 %	—
19	AN-OBM	4,6 %	15,0 %	4,4 %	$P_{1,2} < 0{,}01$
20	KALUM	4,27 ± 0,04	4,23 ± 0,07	4,26 ± 0,13	—
21	HYPO-K	28,1 %	25,0 %	34,8 %	—
22	NATRIUM	138,1 ± 0,5	137,9 ± 0,7	139,0 ± 1,1	—
23	HYPER-NA	3,7 %	—	—	—
24	PULSE	81,5 ± 4,5	91,3 ± 3,2	97,0 ± 3,7	—
25	P	0,08 ± 0,002	0,10 ± 0,005	0,09 ± 0,005	$P_{1,2} < 0{,}05$
26	BIG-P	3,2 %	11,7 %	—	$P_{1,2} < 0{,}01$
27	QT	0,36 ± 0,004	0,37 ± 0,007	0,37 ± 0,020	—
28	LP	34,82 ± 1,79	36,82 ± 1,44	42,67 ± 3,66	—
29	BIG-LP	3,7 %	6,7 %	21,7 %	$P_{1,3} < 0{,}001$ $P_{2,3} < 0{,}05$
30	LG	59,29 ± 1,30	62,09 ± 3,58	58,33 ± 3,66	
31	BIG-LG	16,6 %	11,7 %	17,4 %	—
32	PG	22,35 ± 0,48	23,70 ± 1,99	26,83 ± 1,71	$P_{1,3} < 0{,}01$

Примечание. Группы: 1 — ИМ, не осложнившийся ФП, 2 — осложнившийся пароксизмальной формой ФП и 3 — постоянная форма.

дать, это указывает на то, что больные с пароксизмальной формой ФП гораздо ближе к больным с постоянной формой, чем к больным, не имеющим ФП вообще.

Как видно из табл. 4.2, пароксизмальная форма относительно реже возникает у больных с поражением боковой стенки и чаще у больных, не имевших в анамнезе нарушений ритма. Наличие зубца QS, ПБЛНПГ и пароксизмов ФП в анамнезе увеличивает риск развития пароксизмальной, и в еще большей степени постоянной форм ФП. Наличие обмороков в анамнезе у больных с развившейся пароксизмальной формой ФП хорошо согласуется с клиническими данными.

Таблица 4.3

Достоверные коэффициенты корреляции между возникновением ФП и обучающими параметрами

Параметр	Коэффициент корреляции	Достоверность (P)
SEX	−0,18	<0,01
AGE	0 23	<0,001
LAT-IM	−0,12	<0,05
AN-END	−0,11	<0,05
AN-ARIT	0,38	<0,001
ZSN	0,21	<0,001
PULSE	0,30	<0,001
P	0,20	<0,01
BIG-P	0,15	<0,05
LP	0,37	<0,001
BIG-LP	0,44	<0,001
PG	0,31	<0,001

У больных с ФП также чаще встречается некоторое увеличение размеров левого предсердия и правого желудочка.

Корреляционный анализ между обучающими параметрами и ответом (возникновение и форма ФП) показал наличие только очень слабых взаимосвязей.

В табл. 4.3 сведены только достоверные коэффициенты корреляции (достоверность более 95 %). Как видно, самый высокий коэффициент (ранговый) составляет всего 0,44 (между ФП и увеличением левого предсердия), средний (среднее арифметическое значение абсолютных величин коэффициентов корреляции каждого обучающего параметра с ответом) — всего 0,17.

При анализе корреляций между самими обучающими параметрами внутри каждого из классов выявлена некоторая тенденция к увеличению среднего коэффициента корреляции при возникновении ФП, особенно ее постоянной формы. Так, в группе без ФП средний коэффициент составил 0,255; в группе с пароксизмальной формой ФП — 0,458; с постоянной формой — 0,619 (разница между первым и третьим коэффициентами достоверна, $P < 0,05$). В принципе, это соответствует работам [89, 162], где показано повышение скоррелированности параметров при возникновении заболевания и особенно при увеличении его тяжести. Однако в данной ситуации нельзя считать такой вывод обоснованным, так как в выделенные группы попадает некоторое количество больных с другими осложнениями, что может в значительной мере изменять картину. Не имеет особого смысла выделять и рассматривать здесь "чистые" в этом отношении подгруппы больных, у которых не наблюдалось никаких других осложнений, так как подзадача состоит в необходимости прогнозирования возникно-

Таблица 4.4

Комплекс нейросетей для прогнозирования возникновения ФП и определения ее формы

Номер нейросети	Различаемые классы
ФП-к123-1	1, 2, 3
ФП-к12-2	1, 2
ФП-к13-3	1, 3
ФП-к23-4	2, 3

вения и формы ФП у любого больного инфарктом независимо от появления других осложнений.

При решении подзадачи о прогнозировании возникновения и формы ФП выделено три класса: 1 — отсутствие осложнения, 2 — возникновение пароксизмальной формы ФП, 3 — возникновение постоянной формы ФП.

Следующим этапом работы был подбор оптимальных параметров нейросетей. Для проведения экспериментов каждый раз инициализировался комплекс из четырех нейросетей-классификаторов — одного трехклассового и трех бинарных (табл. 4.4).

Применяемая система маркировки нейросетей состоит из кода задачи (ФП — фибрилляция предсердий, ПК — перикардит, ТЭ — тромбоэмболия, СН — сердечная недостаточность), вида нейросети (к — классификатор, п — предиктор) и порядкового номера нейроэксперта. Если сеть является классификатором, после буквы "к" перечисляются классы, различать которые обучался классификатор.

Одноименные нейросети каждого комплекса имели идентичную стартовую карту, что достигалось копированием файлов сетей. Далее нейросети каждого комплекса загружались в программу MultiNeuron, где перед началом обучения у них изменялись некоторые стартовые параметры (характеристики и число тактов функционирования).

Для экспериментов из обучающей выборки (300 человек) выделена контрольная (тестирующая) группа (50 человек). Все нейросети вначале обучались на 250 оставшихся пациентах, а затем тестировались на контрольной группе. Было создано и обучено несколько нейронных сетей с идентичными стартовыми картами, но с разными параметрами характеристики, числа тактов функционирования и плотности подачи сигналов на нейроны.

Для числа тактов функционирования и плотности подачи входных сигналов прослеживаются достаточно четкие закономерности изменения качества обучения. Параметр характеристики, соответствующий наилучшему качеству обучения, для каждой нейросети имеет свой оптимум. Эти и последующие эксперименты, в том числе на других медицинских задачах, подтвердили, что в подавляющем большинстве случаев следует инициализировать сети описанной архитектуры с не-

большим числом тактов функционирования (обычно 2 или 3), плотность же подачи сигналов на нейроны увеличивать как можно больше, приближая к числу нейронов. Параметр характеристики желательно оптимизировать путем проведения экспериментов для каждой задачи. Диапазон рекомендуемых значений 0,1 ÷ 0,7.

Классификаторы в окончательном наборе имели по 32 входных синапса (соответственно числу обучающих параметров) и 32 нейрона. Все сети были обучены уже на полной обучающей выборке с одинаковым уровнем надежности 0,8, однако имели различный параметр характеристики (соответственно порядку перечисления нейросетей в табл. 4.4: 0,1; 0,3; 0,5; 0,3).

Таким образом, был сформирован нейросетевой консилиум, работающий следующим образом. Пример тестируется всеми четырьмя сетями, каждая из которых выдает свой ответ. Всего классов три. Если один из классов выбран большинством сетей, то он и принимается в качестве окончательного ответа. Если же для двух классов мнения разделились поровну, сравниваются коэффициенты уверенности нейросетей и побеждает ответ пары, коэффициент уверенности у которой выше. Если и эти коэффициенты оказываются равными, что в общем маловероятно, пользователь экспертной системы специально уведомляется о невозможности принятия ею окончательного решения. В любом случае выводятся коэффициенты уверенности каждой нейросети.

Проведен анализ значимости обучающих параметров для принятия сетями решения об ответе.

Наличие нарушений ритма в анамнезе — самый важный параметр для прогнозирования ФП, что вполне согласуется с клиническими данными: возникновение ФП чаще наблюдается у больных, уже имевших нарушения ритма. Следующими по значимости идут наличие гипокалиемии и частота пульса. Сильное влияние этих параметров на возникновение ФП также общеизвестно. Заметим, что непосредственная концентрация калия является слабым по значимости параметром, так же как и концентрация натрия. Это подтверждает установленные границы нормы для калия и натрия: выявив существенные закономерности между ответом и наличием гипокалиемии и гипернатриемии, сеть слабо нуждается в информации о точных концентрациях, что проявляется в малой значимости этих параметров. Далее следует величина зубца Р, регистрируемая по ЭКГ и указывающая на время прохождения электрического импульса по предсердиям, и параметр, показывающий увеличение зубца Р. Наличие эндокринных заболеваний в анамнезе — шестой по значимости параметр. Рассмотрение механизма его влияния требует специального исследования с изучением различных эндокринных патологий у больных инфарктом.

Наименее значимые параметры — нижняя и задняя локализация инфаркта миокарда. При анализе малозначимых параметров необходима их корректная интерпретация. В данном случае малая значимость показателей свидетельствует не о том, что при этих ло-

кализациях редко возникает ФП (такая закономерность, в особенности сильная, наоборот, привела бы к увеличению значимости), а о том, что при наличии такой локализации возникновение ФП наблюдается не чаще и не реже, чем без нее (отсутствие взаимосвязи).

Малозначимыми параметрами также являются наличие стенокардии в анамнезе и увеличение левого желудочка. Увеличение же левого предсердия достаточно важно для прогноза.

Далеко не всегда самые значимые параметры имеют самые выраженные различия в подгруппах. Например, если по самому значимому показателю — аритмиям в анамнезе — много достоверных различий, то гипокалиемия и частота пульса не различаются достоверно. Это связано с тем, что нейросеть "рассматривает" параметры не в отдельности, а во взаимосвязи со всеми остальными.

Если нейросетевой консилиум делает вывод о возможности возникновения фибрилляции предсердий пароксизмальной или постоянной формы, возникает потребность в прогнозировании времени появления осложнения. Поэтому при сборе обучающих данных у больных с осложнениями фиксировалось, на какой день после поступления в стационар возникло каждое из них.

В полной обучающей выборке из 300 больных у 60 была пароксизмальная форма ФП, и у 23 — постоянная. Осложнение возникало в сроки от 1 до 27 дней. В первой подгруппе средний срок до осложнения составил $3{,}59 \pm 1{,}14$ дней, во второй — $1{,}3 \pm 0{,}37$ дней. Эти показатели достоверно не различаются. Достоверных коэффициентов корреляции срока появления осложнения с обучающими параметрами не обнаружено.

Группа из 83 человек составила обучающую выборку для нейросетей-предикторов. Кроме исходных 32 входных параметров на дополнительный входной синапс подавалась возникшая форма ФП. Предикторы имели по 33 входных синапса, 33 нейрона. При обучении предикторов применен метод повышения уровня допустимого отклонения при невозможности дальнейшего обучения. Обучение было начато с уровня отклонения 0,1. Это означает, что при вычислении срока возникновения аритмии предиктор может ошибаться не более чем на 0,1 дня. В действительности, конечно, такая точность не требуется хотя бы потому, что срок возникновения осложнений определялся с точностью до 1 дня. Предиктор одиннадцать раз прекращал обучение, однако после каждой остановки требования к точности ответа смягчались (уровень допустимого отклонения повышался на 0,1). Полное обучение стало возможным при уровне отклонения 1,1. На обучающей выборке нейросеть определяет момент возникновения фибрилляции предсердий с точностью почти ± 1 день. При проверке предикторов на тестирующей выборке максимальная ошибка, допущенная нейросетью, составила 3 дня.

Исследование значимости обучающих параметров для прогнозирования срока ФП показало, что наиболее информативными для ней-

росети являются локализация инфаркта (в особенности, поражение передних и нижних отделов), вид комплекса QRS, наличие бронхитов в анамнезе, длина зубца Р на ЭКГ. Сама форма ФП (дополнительный параметр) слабо определяла срок возникновения осложнения, так как была восемнадцатой по значимости. Наименьшую значимость имели показатели ультразвукового исследования сердца.

Завершая рассмотрение прогнозирования ФП, напомним, что для предсказания времени возникновения аритмии подача предиктору данных о больном должна осуществляться *только* в том случае, если консилиум классификаторов принял решение о возможности возникновения ФП. Предикторы обучались *только* на больных с возникшим осложнением и ничего "не знают" о больных без осложнения, поэтому подача данных о больном без прогнозируемого осложнения приведет к заранее некорректному ответу предиктора.

4. Прогнозирование возникновения перикардита

Как видно из табл. 4.5, перикардит чаще наблюдается при поражении переднебоковых отделов сердца, что подтверждает данные [155]. Для больных с перикардитом намного более характерен комплекс QRS вида QS (т.е. трансмуральный ИМ). Интересен факт, что судя по результатам статистической обработки перикардит после инфаркта несколько чаще возникает у пациентов с отсутствием приступов стенокардии в анамнезе и отсутствием гипертонической болезни. У больных с удлиненным зубцом Р перикардит бывает реже, а у больных с увеличенным левым желудочком — чаще.

Корреляционный анализ (табл. 4.6) выявил лишь слабые взаимосвязи между ответом и обучающими параметрами, которые подтверждают различия в средних значениях и частотах.

Инициализированные нейронные сети — классификаторы — имели по 32 входных синапса и 32 нейрона, как и в предыдущей подзадаче. Сети обучались лишь за 184 400—243 522 такта, что довольно много по сравнению с предыдущей задачей. Это говорит о том, что прогнозирование перикардита, видимо, несколько сложнее. Нейросети, инициализированные с характеристикой более 0,4 и числом тактов функционирования 3 и более вообще не смогли обучиться на данном задачнике. Поэтому окончательные сети имели характеристику 0,1 и число тактов функционирования 2. Количество правильно распознанных примеров в тестирующей выборке составило всего 75,3 %, что также подтверждает трудность задачи.

Тремя наиболее значимыми показателями для прогноза перикардита оказались возраст, наличие аритмий в анамнезе и увеличение левого предсердия. Одними из самых значимых параметров были также гипернатриемия, длина интервала QT на ЭКГ, концентрация калия и гипокалиемия. Существенная значимость обоих последних параметров указывает на то, что в отличие от ФП здесь имеют

Таблица 4.5

Средние значения и их ошибки ($M \pm m$) или частоты (%) параметров у больных ИМ

№	Параметр	Группа 1, n = 172	Группа 2, n = 118	Достоверность различия (P)
1	2	3	4	5
1	AGE	62,2 ± 1,89	58,42 ± 0,89	—
2	SEX			
	0	38,8 %	33,9 %	—
	1	61,2 %	66,1 %	—
3	ANT-IM	58,8 %	74,6 %	<0,01
4	LAT-IM	42,5 %	57,6 %	<0,02
5	INF-IM	42,5 %	28,8 %	<0,02
6	POS-IM	11,9 %	16,9 %	—
7	PGE-IM	1,9 %	2,5 %	—
8	QRS			
	1	13,8 %	3,4 %	<0,01
	2	44,4 %	24,6 %	<0,001
	3	41,8 %	72,0 %	<0,001
9	AN-IM	0,456 ± 0,072	0,305 ± 0,057	—
10	AN-STEN	2,031 ± 0,196	2,212 ± 0,236	—
11	FC-STEN			
	0	43,1 %	43,2 %	—
	1	51,9 %	50,9 %	—
	2	5,0 %	5,9 %	—
12	GB			
	0	34,4 %	47,5 %	<0,05
	1	0,6 %	1,7 %	—
	2	53,1 %	38,1 %	<0,02
	3	11,9 %	12,7 %	—
13	ZSN			
	0	33,8 %	39,8 %	—
	1	6,2 %	5,9 %	—
	2	33,8 %	27,9 %	—
	3	6,2 %	5,9 %	—
	4	20,0 %	20,5 %	—
14	AN-ARIT			
	0	83,1 %	97,5 %	<0,001
	3	11,3 %	0,9 %	<0,001
	6	0,6 %	—	—
	9	0,6 %	—	—
	11	4,4 %	1,6 %	<0,001

Окончание табл. 4.5

1	2	3	4	5
15	DIS-PROV			
	0	69,4 %	60,2 %	—
	1	3,1 %	7,6 %	—
	2	0,6 %	—	—
	3	—	0,9 %	—
	4	—	2,5 %	<0,05
	5	11,9 %	13,6 %	—
	6	1,9 %	3,4 %	—
	7	0,6 %	—	—
	8	2,5 %	0,8 %	—
	9	6,9 %	9,3 %	—
	10	3,1 %	1,7 %	—
16	AN-END	10,6 %	8,5 %	—
17	AN-TROMB	15,6 %	12,7 %	—
18	AN-BRONC	16,9 %	16,9 %	—
19	AN-OBM	4,4 %	9,3 %	—
20	KALIUM	4,27 ± 0,05	4,26 ± 0,06	—
21	HYPO-K	27,5 %	29,7 %	—
22	NATRIUM	138,1 ± 0,57	138,2 ± 0,59	—
23	HYPER-NA	2,5 %	2,5 %	—
24	PULSE	84,33 ± 1,37	82,77 ± 1,42	—
25	P	0,088 ± 0,002	0,083 ± 0,002	—
26	BIG-P	6,9 %	1,7 %	<0,05
27	QT	0,367 ± 0,006	0,356 ± 0,005	—
28	LP	37,50 ± 1,133	34,40 ± 0,897	<0,05
29	BIG-LP	6,9 %	4,2 %	—
30	LG	58,33 ± 2,130	60,51 ± 1,289	—
31	BIG-LG	9,4 %	24,6 %	<0,001
32	PG	23,62 ± 0,940	22,23 ± 0,522	—

Примечание. Группы: 1 — ИМ, не осложнившийся и 2 — осложнившийся перикардитом.

значение не только резкое снижение калия, за которое отвечает параметр HYPO-K, но и тонкие изменения концентрации элемента, учитываемые в показателе KALIUM.

При сборе данных для обучающей выборки нами брались только случаи реактивного воспаления сердечной сорочки (эпистенокардический перикардит). Перикардиты как проявление синдрома Дресслера в обучающую выборку пока не включались, поэтому сроки возникновения его у наших пациентов были в пределах первых четырех дней заболевания.

Нейросети-предикторы, созданные для предсказания времени возникновения перикардита, смогли полностью обучиться лишь с уровнем

Таблица 4.6

Достоверные коэффициенты корреляции между возникновением перикардита и обучающими параметрами

Параметр	Коэффициент корреляции	Достоверность (*P*)
AGE	−0,17	<0,010
ANT-IM	0,16	<0,05
LAT-IM	0,15	<0,05
INF-IM	−0,14	<0,05
QRS	0,30	<0,001
AN-ARIT	−0,16	<0,05
BIG-P	−0,15	<0,05
LP	−0,26	<0,001
BIG-LP	−0,26	<0,001
BIG-LG	0,35	<0,001

допустимого отклонения, равным 2. Это указывает на практическое отсутствие закономерностей между используемыми обучающими параметрами и ответом. Применение такой нейросети бессмысленно, так как разброс в ошибке вычисления полностью накрывает диапазон значений ответа. Поэтому в экспертную систему эти сети не были включены.

5. Прогнозирование факта и времени возникновения тромбоэмболических осложнений

Тромбоэмболические осложнения обычно возникают в первые 2 нед после инфаркта миокарда. Чаще всего источником тромбоэмболий становятся пристеночные тромбы, которые образуются в полости сердца, обычно в месте локализации некротизированного участка сердечной мышцы при трансмуральных и субэндокардиальных инфарктах. Факторами, способствующими образованию тромбов, являются обширные инфаркты, аневризмы, мерцательная аритмия, застойная сердечная недостаточность [188, 17]. Кроме того, возникновению тромбоэмболий способствуют ожирение, варикозное расширение вен [168]. Локализация инфаркта играет некоторое значение в развитии тромбоэмболий. Так, реже это осложнение бывает при нижней локализации инфаркта [186].

У 58 из 300 больных обучающей выборки отмечены те или иные тромбоэмболические осложнения различной степени тяжести и локализации (легочная артерия и ее ветви, сосуды головного мозга).

В табл. 4.7 приведены результаты сравнения групп с возникшей тромбоэмболией и без нее по исследуемым показателям. Тромбоэмболии несколько чаще бывают у больных с поражением боковых отделов сердца, при трансмуральном поражении сердечной мышцы

Таблица 4.7

Средние арифметические и их ошибки ($M \pm m$) или частоты (%) параметров у больных инфарктом миокарда

№	Параметр	Группа 1, n = 242	Группа 2, n = 58	Достоверность различия (P)
1	2	3	4	5
1	AGE	62,68 ± 1,55	57,86 ± 1,43	—
2	SEX			
	0	39,2 %	29,3 %	—
	1	60,8 %	70,7 %	—
3	ANT-IM	60,1 %	72,4 %	—
4	LAT-IM	41,8 %	58,6 %	<0,05
5	INF-IM	41,2 %	37,9 %	—
6	POS-IM	9,8 %	17,2 %	—
7	PGE-IM	2,6 %	—	—
8	QRS			
	1	12,4 %	5,2 %	—
	2	44,4 %	27,6 %	<0,02
	3	43,2 %	63,8 %	<0,01
9	AN-IM	0,510 ± 0,078	0,224 ± 0,065	—
10	AN-STEN	2,065 ± 0,195	2,017 ± 0,352	—
11	FC-STEN			
	0	39,2 %	55,2 %	<0,05
	1	54,9 %	41,4 %	—
	2	5,9 %	3,4 %	—
12	GB			
	0	39,2 %	39,7 %	—
	1	0,7 %	—	—
	2	48,4 %	48,3 %	—
	3	11,7 %	12,0 %	—
13	ZSN			
	0	32,0 %	20,7 %	—
	1	7,2 %	3,5 %	—
	2	33,3 %	36,2 %	—
	3	5,2 %	12,1 %	—
	4	22,3 %	27,5 %	—
14	AN-ARIT			
	0	83,0 %	98,3 %	<0,01
	3	11,1 %	1,7 %	<0,05
	6	0,7 %	—	—
	9	0,6 %	—	—
	11	4,6 %	—	—

Окончание табл. 4.7

1	2	3	4	5
15	DIS-PROV			
	0	70,6 %	62,1 %	—
	1	4,6 %	1,7 %	—
	2	0,7 %	—	—
	4	0,6 %	—	—
	5	11,1 %	12,1 %	—
	6	1,9 %	3,4 %	—
	7	—	1,7 %	—
	8	3,3 %	—	—
	9	3,9 %	17,2 %	<0,001
	10	3,3 %	1,7 %	—
16	AN-END	12,4 %	1,7 %	<0,02
17	AN-TROMB	13,7 %	18,9 %	—
18	AN-BRONC	15,7 %	24,1 %	—
19	AN-OBM	3,9 %	10,3 %	<0,05
20	KALIUM	4,25 ± 0,05	4,23 ± 0,07	—
21	HYPO-K	29,4 %	29,3 %	—
22	NATRIUM	138,0 ± 0,58	137,1 ± 0,83	—
23	HYPER-NA	2,6 %	—	—
24	PULSE	84,46 ± 1,47	82,36 ± 1,99	—
25	P	0,089 ± 0,003	0,077 ± 0,003	—
26	BIG-P	7,8 %	—	—
27	QT	0,371 ± 0,007	0,353 ± 0,007	—
28	LP	38,14 ± 1,113	35,00 ± 1,64	—
29	BIG-LP	7,8 %	5,2 %	—
30	LG	60,55 ± 1,95	60,28 ± 2,69	—
31	BIG-LG	12,4 %	20,7 %	—
32	PG	23,68 ± 0,97	22,11 ± 0,81	—

Примечание. Группы: 1 — ИМ, не осложнившийся и 2 — осложнившийся тромбоэмболией.

(QS-тип комплекса QRS), имеющих в анамнезе обмороки и полную блокаду правой ножки пучка Гиса.

Корреляционный анализ взаимосвязей между обучающими параметрами и ответами выявил всего шесть достоверных коэффициентов корреляции, которые очень малы (табл. 4.8).

Несмотря на это, инициализированный комплекс нейронных сетей показал достаточно хорошую обучаемость: 32-нейронные сети обучались за 94 282—112 116 тактов с характеристикой 0,6 и числом тактов функционирования 2.

Основными значимыми показателями являются наличие обмороков, аритмий, застойной сердечной недостаточности и инфарктов миокарда в анамнезе. Последние два параметра связаны с изменениями в

Таблица 4.8

Достоверные коэффициенты корреляции между возникновением тромбоэмболических осложнений и обучающими параметрами

Параметр	Коэффициент корреляции	Достоверность (*P*)
AGE	−0,20	<0,001
LAT-IM	0,15	<0,05
QRS	0,20	<0,001
FC-STEN	−0,14	<0,05
ANAM-IM	−0,15	<0,05
P	−0,24	<0,001

свертывающей системе крови, которые, в свою очередь, увеличивают риск возникновения тромбоэмболических осложнений.

Проверка нейросетей на тестирующей выборке показала правильное распознавание в 82 % случаев. Интересно, что у всех неверно распознанных больных время появления осложнения было 20 и более дней с момента поступления в стационар.

Эксперименты по обучению предикторов, прогнозирующих время развития тромбоэмболий, подтвердили найденную закономерность.

Предикторы, которые мы пытались обучить на всех 58 больных с развившимися тромбоэмболиями, не смогли обучиться. Была применена методика последовательного исключения из обучающей выборки примеров с максимальными оценками. Только после того как предиктор исключил 24 больных, он смог обучиться с уровнем отклонения 1,3.

Для всех исключенных больных время возникновения осложнения составляло от 18 до 28 дней, для всех оставшихся — от 1 до 17 дней. Можно предположить, что настолько четкое разделение обучающей выборки происходит вследствие сильно различающихся закономерностей, связывающих обучающие параметры и срок возникновения тромбоэмболии в двух образованных подгруппах. При сравнении этих подгрупп на достоверность различия показателей выявлено, что у больных подгруппы с поздними тромбоэмболиями чаще наблюдается третья стадия гипертонической болезни (4,17 и 27,2 %), реже — гипокалиемия (41,67 и 20,59 %). По остальным параметрам обучающей выборки достоверных различий не обнаружено.

На следующем этапе исследования были созданы сети-классификаторы, которые обучались различать три класса: больных без тромбоэмболии, с "ранней" тромбоэмболией и с "поздней". Эти сети смогли обучиться решению задачи лишь после исключения из обучающей выборки нескольких (от 4 до 6) примеров 3-го класса.

Полностью обучиться смогли лишь бинарные классификаторы, различающие больных с "ранней" и "поздней" тромбоэмболией. При

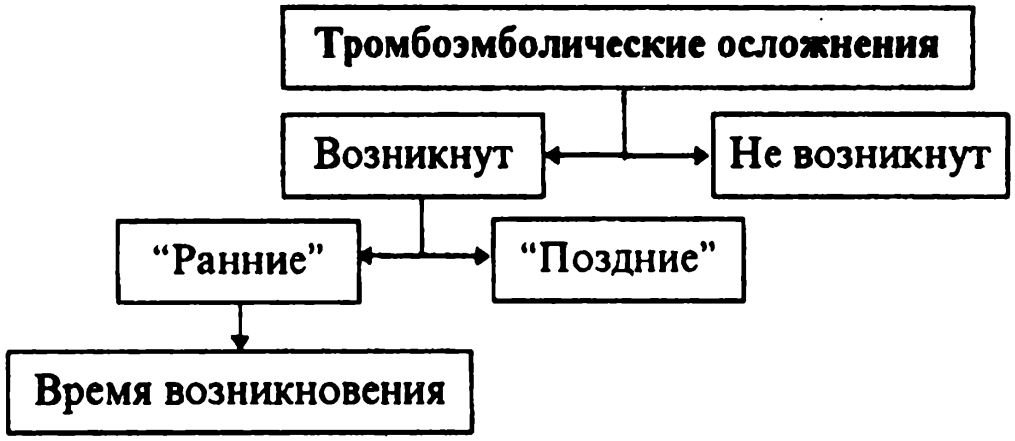


Рис. 4.2. Схема прогнозирования тромбоэмболических осложнений.

тестировании эти сети показывали хорошую распознаваемость примеров 1-го класса (88,4 %) и довольно низкую — 2-го класса (67,4 %).

Однако мы не оставляли надежду обучить сети прогнозированию срока возникновения тромбоэмболий. После разделения больных на две подгруппы и обучения классификаторов, разделяющих эти подгруппы, сделана попытка обучить предикторы прогнозированию времени возникновения осложнения в каждой подгруппе отдельно. Как и следовало ожидать, на "ранней" группе предиктор обучился полностью, на "поздней" не смог обучиться вообще, несмотря на все ухищрения по настройке параметров сетей.

Итак, обучение нейронных сетей показало возможность прогнозирования возникновения тромбоэмболических осложнений после инфаркта миокарда. Тромбоэмболии, возникающие в первые 2 нед после инфаркта, прогнозируются хорошо. Время их появления также достаточно успешно прогнозируется. Тромбоэмболии, которые возникают после 2 нед, прогнозируются гораздо хуже, что подтверждено комплексом из 12 нейросетей, классификаторов и предикторов с различной постановкой задач. Прогнозировать время появления "поздних" тромбоэмболий по исследуемым параметрам, по-видимому, не представляется возможным. Единственными показателями, достоверно различающимися у больных обеих подгрупп, являются частоты третьей стадии гипертонической болезни и гипокалиемии, что, естественно, не может приниматься в качестве критерия прогноза.

Схема нейросетевого прогноза тромбоэмболических осложнений с учетом результатов экспериментов несколько изменяется (рис. 4.2). Так, можно предположить, что появление "ранних" тромбоэмболий во многом зависит от состояния больного до и во время инфаркта миокарда, т.е. в какой-то степени предопределено.

Возникновение же "поздних" тромбоэмболий, по-видимому, обусловлено факторами, выходящими на первый план во время подострого периода инфаркта миокарда. Единственной информацией о состоянии

больных в этот период в наших данных были факты и время возникновения других осложнений. Однако дополнительное статистическое исследование взаимосвязей между возникновением "поздних" тромбоэмболий и других осложнений также не выявило значимых закономерностей.

В заключение можно сделать два вывода. Во-первых, нейронные сети в данном случае служили инструментом проведения научных исследований, позволяющим ставить содержательные вопросы исследователю. Мы попытались изложить последовательность и логику экспериментов для иллюстрации такого подхода.

Во-вторых, сама проблема требует дальнейшего углубленного изучения после сбора дополнительных данных о протекании госпитального периода у больных ИМ, что поможет углубить знания о причинах и механизмах развития тромбоэмболических осложнений. Вполне вероятно, такое исследование приведет к необходимости дифференцированной профилактики "ранних" и "поздних" тромбоэмболических осложнений.

6. Прогнозирование факта и времени возникновения или усугубления сердечной недостаточности

У значительной части больных с инфарктом миокарда СН наблюдается уже при поступлении в клинику. Это связано с нарушением моторной функции сердца непосредственно после ИМ. Возникновение при этом СН носит острый характер и нередко ведет к развитию кардиогенного шока и смерти больного. У многих больных ИМ развивается на фоне уже имеющейся хронической СН и значительно усугубляет ее [155]. Развитию СН способствует артериальная гипертензия.

Довольно большую опасность представляет возникновение или резкое усугубление уже имеющейся СН в подостром периоде. Состояние больного после поступления в клинику сначала постепенно улучшается, он увеличивает физическую активность, однако в какой-то момент неожиданно появляются или усиливаются симптомы СН. Это наблюдается у значительной части больных, однако иногда ведет к резкому ухудшению состояния больного, новым осложнениям, а порой и к летальному исходу. Как уже говорилось, подобные ситуации могут возникнуть даже на 3—4-й неделе после начала заболевания.

Диагностика развившейся СН не представляет особого труда. Схемы лечения ее также достаточно хорошо отработаны.

При сборе обучающих данных больные, у которых в течение госпитального периода возникла (на фоне ее первоначального отсутствия) или усугубилась уже имеющаяся СН, были объединены в специальную группу из 156 человек.

Таблица 4.9

Средние значения и их ошибки ($M \pm m$) или частоты (%) параметров у больных ИМ

№	Параметр	Группа 1, n = 144	Группа 2, n = 156	Достоверность различия (P)
1	2	3	4	5
1	AGE	56,71 ± 3,20	62,91 ± 2,88	—
2	SEX			
	0	31,0 %	38,5 %	—
	1	69,0 %	61,5 %	—
3	ANT-IM	53,5 %	67,3 %	<0,02
4	LAT-IM	53,5 %	44,9 %	—
5	INF-IM	46,6 %	37,2 %	—
6	POS-IM	10,3 %	12,2 %	—
7	PGE-IM	5,2 %	0,6 %	<0,02
8	QRS			
	1	10,3 %	11,5 %	—
	2	46,6 %	37,2 %	—
	3	43,1 %	51,3 %	—
9	AN-IM	0,190 ± 0,072	0,519 ± 0,076	<0,01
10	AN-STEN	1,672 ± 0,285	2,192 ± 0,207	—
11	FC-STEN			
	0	44,8 %	43,6 %	—
	1	50,0 %	51,3 %	—
	2	5,2 %	5,1 %	—
12	GB			
	0	41,4 %	39,1 %	—
	1	1,7 %	—	
	2	51,7 %	46,8 %	—
	3	5,2 %	14,1 %	<0,01
13	ZSN			
	0	93,1 %	4,5 %	<0,001
	1	1,7 %	8,3 %	<0,01
	2	3,5 %	45,5 %	<0,001
	3	—	9,6 %	<0,001
	4	1,7 %	32,1 %	<0,001
14	AN-ARIT			
	0	94,8 %	84,6 %	<0,01
	3	1,7 %	10,9 %	<0,001
	6	—	0,6 %	—
	9	—	0,6 %	—
	11	3,5 %	3,3 %	—

Окончание табл. 4.9

1	2	3	4	5
15	DIS-PROV			
	0	81,0 %	63,5 %	<0,001
	1	3,5 %	4,5 %	—
	2	—	0,6 %	—
	4	—	0,6 %	—
	5	5,2 %	13,5 %	<0,02
	6	—	3,2 %	—
	7	—	0,6 %	—
	8	—	3,2 %	—
	9	8,6 %	7,1 %	—
	10	1,7 %	3,2 %	—
16	AN-END	8,6 %	9,6 %	—
17	AN-TROMB	15,5 %	14,7 %	—
18	AN-BRONC	3,5 %	23,1 %	<0,001
19	AN-OBM	6,9 %	5,1 %	—
20	KALIUM	4,25 ± 0,09	4,25 ± 0,05	—
21	HYPO-K	27,6 %	30,1 %	—
22	NATRIUM	138,5 ± 1,10	137,4 ± 0,52	—
23	HYPER-NA	3,5 %	1,3 %	—
24	PULSE	77,72 ± 2,81	86,29 ± 2,45	<0,05
25	P	0,090 ± 0,002	0,085 ± 0,003	—
26	BIG-P	5,2 %	5,8 %	—
27	QT	0,357 ± 0,010	0,364 ± 0,006	—
28	LP	36,80 ± 1,25	36,91 ± 1,01	—
29	BIG-LP	1,7 %	8,9 %	<0,01
30	LG	60,60 ± 4,98	60,42 ± 1,61	—
31	BIG-LG	5,2 %	18,6 %	<0,001
32	PG	25,60 ± 3,49	22,79 ± 0,62	—

Примечание. Группы: 1 — ИМ, не осложненный и 2 — осложненный возникновением или усугублением сердечной недостаточности.

Результаты сравнения этой группы с остальными больными приведены в табл. 4.9. Наиболее яркими различиями являются повышенная частота гипертонической болезни третьей стадии и исходной ЗСН в исследуемой группе, а также более частое увеличение левых отделов сердца по результатам эхокардиографии. Для нее также характерно некоторое увеличение частоты сердечных сокращений на ЭКГ при поступлении в клинику.

Эти же закономерности прослеживаются при корреляционном анализе (табл. 4.10). Большинство коэффициентов корреляции весьма низки, однако взаимосвязь с предшествующей СН довольно значительна (коэффициент корреляции 0,72).

Таблица 4.10

Достоверные коэффициенты корреляции между возникновением (усугублением) сердечной недостаточности и обучающими параметрами

Параметр	Коэффициент корреляции	Достоверность (*P*)
AGE	0,25	<0,001
PGE-IM	−0,15	<0,05
ANAM-IM	0,17	<0,05
ZSN	0,72	<0,001
PULSE	0,23	<0,001

Нейросетевые классификаторы из 32 нейронов обучились решению задачи за 51 744—78 828 тактов. Число тактов функционирования — 2, характеристика — 0,4.

Наличие ЗСН в анамнезе является основным параметром для принятия решения. Он существенно превосходит остальные по значимости. Исключительная важность его подтверждается и результатами корреляционного анализа. Вполне естественно, что при столь высокой корреляции с ответом (0,72) нейросеть выделила этот показатель в качестве главного.

При проверке обученных сетей на тестирующей выборке они правильно распознавали 78 % примеров. Проанализировав неправильно распознанные случаи, мы обнаружили, что для них время возникновения или усугубления ЗСН было 18 и более дней. Проведение и результаты последующих экспериментов практически повторяли описанное в предыдущей главе для тромбоэмболий. Предикторы смогли обучиться прогнозированию времени развития ЗСН, только исключив 31 пример со сроком развития этого осложнения 18 дней и более. Отметим, что в отличие от ситуации с тромбоэмболиями, в число примеров, по которым предикторы смогли обучиться, попали 5 больных со сроком возникновения ЗСН от 21 до 24 дней. Однако в число исключенных не попал ни один, имеющий время возникновения или усугубления ЗСН менее 18 дней. Раздельное обучение предикторов на полученных подгруппах привело к такому же результату, как и при тромбоэмболиях: на подгруппе "ранней" СН предиктор обучился прогнозировать срок ее возникновения с уровнем допустимого отклонения 1,2. Попытка обучить предикторы на подгруппе "поздней" ЗСН не удалась.

В данном случае также проведен статистический анализ полученных подгрупп, который не выявил никаких достоверных различий между параметрами.

Таким образом, закономерность, выявленная при исследовании тромбоэмболических осложнений, прослеживается и в ситуации с сердечной недостаточностью. Нейросети-предикторы достаточно хорошо обучаются прогнозировать время возникновения этих осложнений

только на больных, у которых они появились не позднее 17-го дня. Четкое совпадение этого значения в двух ситуациях с различными осложнениями, по которым обучались совершенно независимо друг от друга различные нейросети, указывает на то, что данный момент является своеобразной "точкой перехода" к иным причинно-следственным связям в протекании периода реконвалесценции. За этой точкой, по-видимому, либо значительно ослабевает действие стартовых факторов развития инфаркта миокарда, либо возникают новые факторы, определяющие дальнейшее состояние больного.

7. Изучение влияния исследуемых параметров на прогноз осложнений инфаркта миокарда

Проведение подобных исследований с помощью нейронных сетей имеет две основные цели. Во-первых, это изучение взаимосвязей между параметрами и ответами, которое позволяет глубже понять зависимость осложнений инфаркта от комплексного воздействия факторов. Во-вторых, моделирование ответа нейросетей на конкретном больном иногда позволяет выяснить оптимальные способы профилактики патологических состояний.

Объем работы не позволяет дать детальный обзор всех взаимосвязей между параметрами даже в отдельно рассматриваемой задаче, так как число всевозможных комбинаций параметров очень велико. Поэтому наметим лишь способы проведения таких работ и проиллюстрируем их примерами.

Первый метод заключается в последовательном изменении значений одного или двух параметров и проверке, как на это реагирует ответ данной нейросети. В случае одного параметра ответ нейросети можно представлять как в виде класса, так и в виде сигналов, снимаемых с каждого выходного нейрона (в случае классификатора). В случае двух параметров ответ обычно можно представлять только в виде класса. Ограничение не принципиально, а связано в основном с возможностями графического представления результатов (рис. 4.3).

Располагая возможностями лишь двумерного изображения, трудно показать изменения ответов каждого нейрона (они представляли бы три пересекающиеся плоскости в трехмерной системе координат), поэтому каждой координате на рисунке соответствует класс, который выдает сеть при соответствующих значениях параметров PULSE (частота пульса по ЭКГ) и P (длина зубца P и по ЭКГ).

Построение подобных диаграмм с помощью нейропрограмм несложно и может использоваться для определения индивидуальных особенностей больных и групповых закономерностей.

При большом числе параметров интерпретация соответствующего количества диаграмм становится трудоемкой. Кроме того, во многих

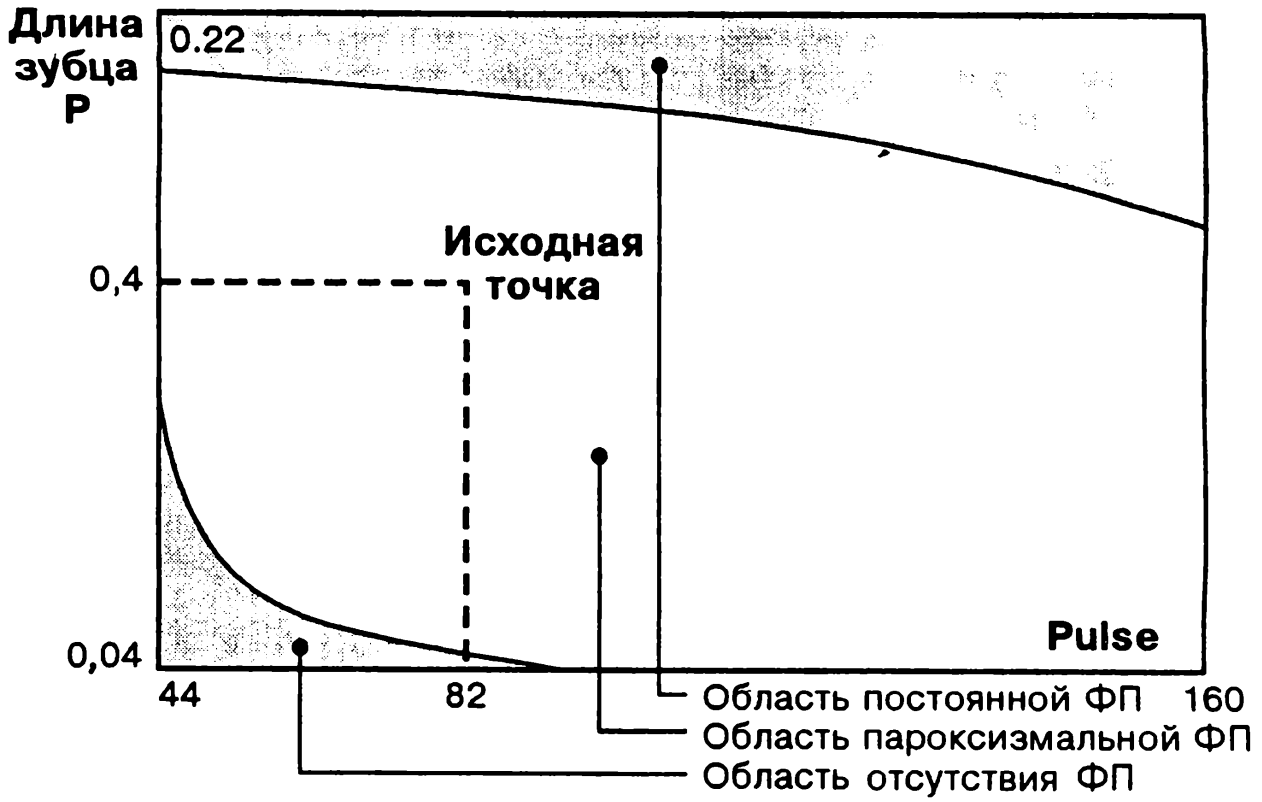


Рис. 4.3. Зависимость ответа нейронной сети от изменения двух параметров (PULSE и длина зубца P) у больного с ИМ, осложнившимся пароксизмальной формой ФП (исходные значения: PULSE — 82, P — 0,14).

случаях желательно подстраивать многие параметры одновременно, добиваясь нужного ответа.

В табл. 4.11 приведено описание больного, для которого нейросеть ФП-к123-1 прогнозирует возникновение постоянной формы ФП.

Во второй колонке дается список параметров, причем те из них, которые возможно подстраивать, подчеркнуты. В данном случае их всего 5. В третьей колонке находятся исходные значения каждого параметра, в четвертой — новые значения подчеркнутых параметров, вычисленные нейросетью так, чтобы ответ изменился на "возникновение пароксизмальной формы ФП". Нейросеть старается подстраивать параметры оптимальным способом, чтобы все сдвиги были минимальными. В пятой колонке нейросеть продолжила подстройку параметров, чтобы "перевести" больного в класс 1 — "отсутствие фибрилляции предсердий".

Исходные сигналы выходных нейронов в этом примере составляли 31,5 (1-й класс), 9,3 (2-й класс) и 87,1 (3-й класс), что и определяло прогноз постоянной формы ФП.

Для того чтобы изменить ответ с 3-го класса на 2-й, нейросеть "предложила" снизить концентрацию натрия со 146 до 120, уменьшить частоту пульса со 120 до 90, уменьшить длину зубца P на ЭКГ с 0,11 до 0,04 (что автоматически изменяет параметр BIG-P с 1 на 0) и увеличить длину зубца QT с 0,32 до 0,38. Как видно, основные изменения затронули концентрацию натрия и частоту пульса. Нейро-

Таблица 4.11

Подстройка нейросетью ФП-к123-1 разрешенных параметров у больного с прогнозируемой постоянной формой фибрилляции предсердий (класс 3) для получения требуемого ответа (сначала класс 2, затем 1)

№	Параметр	Исходно	Для класса 2	Для класса 1
1	AGE	55	—	—
2	SEX	1	—	—
3	ANT-IM	1	—	—
4	LAT-IM	1	—	—
5	INF-IM	0	—	—
6	POS-IM	0	—	—
7	PGE-IM	0	—	—
8	QRS	3	—	—
9	AN-IM	0	—	—
10	AN-STEN	6	—	—
11	FC-STEN	2	—	—
12	GB	0	—	—
13	ZSN	1	—	—
14	AN-ARIT	3	—	—
15	DIS-PROV	0	—	—
16	AN-END	0	—	—
17	AN-TROMB	0	—	—
18	AN-BRONC	0	—	—
19	AN-OBM	0	—	—
20	KALIUM	3,5	3,5	3,8
21	HYPO-K	1	1	1
22	NATRIUM	146	120	120
23	HYPER-NA	0	0	0
24	PULSE	120	90	74
25	P	0,11	0,04	0,04
26	BIG-P	1	0	0
27	QT	0,32	0,38	0,32
28	LP	40	—	—
29	BIG-LP	1	—	—
30	LG	76	—	—
31	BIG-LG	1	—	—
32	PG	22	—	—

сеть не изменила концентрацию калия, который, по-видимому, не играет существенной роли в определении прогноза при данном уникальном наборе параметров. Сигналы выходных нейронов теперь составляют 8,9, 13,9 и 10,9.

Теперь потребуем от нейросети продолжить подстройку параметров, чтобы "перевести" больного в класс 1. Концентрация натрия и

длина зубца Р оставлены прежними, незначительно возросла концентрация калия, существенно снижена частота пульса, а длина зубца QT возвращена на прежний уровень. Теперь при таком наборе значений параметров нейросеть выдает ответ "1-й класс" (сигналы выходных нейронов — 16,7, 13,3 и 8,7). Несмотря на достижение требуемого ответа, сигнал с 1-го выходного нейрона ненамного превышает остальные, что определяет достаточно высокий риск развития осложнений.

Приведенный пример удобен с точки зрения возможности подстройки параметров. В действительности, многие примеры не удается "перевести" в требуемый класс вследствие того, что прогнозируемый неблагоприятный ответ во многом определяется неподстраиваемыми параметрами, например данными анамнеза или показателями эхокардиографии. В таких случаях нейросеть добивается лишь "минимального риска" возникновения неблагоприятного ответа, определяемого соотношением сигналов на выходных нейронах.

Не следует забывать и о возникновении других осложнений. Поэтому подстройка параметров должна осуществляться одновременно всем нейросетям, отвечающим за прогноз нескольких осложнений.

В заключение подчеркнем, что проведенное обучение нейросетей являлось стартовым. При использовании в клинике экспертной системы необходим некоторый период тестирования и доучивания нейросетей, особенно если система используется в другом регионе. Более того, сама идеология нейросетевых экспертных систем направлена на постоянное обучение нейросетей во время использования. Помощник врача обучается и набирается опыта вместе с врачом.

Часть 2

НЕЙРОННЫЕ СЕТИ

ВВЕДЕНИЕ. ИДЕЯ НЕЙРОКОМПЬЮТЕРА

В этом разделе описаны основные идеи, вокруг которых поднялся нейрокомпьютерный "бум". Один из источников идей — попытка подражать природе при создании компьютера, в чем-то подобного мозгу. Однако мозг — это естественный объект и, чтобы имитировать его, надо иметь "план Создателя". Такого плана у нас, увы, нет, поэтому правильнее сказать, что в идею нейрокомпьютера закладываются более-менее формализованные *представления* о структуре и функционировании мозга, которые и составляют то *что-то*, в чем достигается подобие [258].

Ядром используемых представлений является мысль о том, что нейроны можно моделировать довольно простыми автоматами [111], а вся сложность мозга, гибкость его функционирования и другие важнейшие качества определяются связями между нейронами. Каждая связь рассматривается как совсем простой элемент, служащий для передачи сигнала. Предельным выражением этой точки зрения может служить лозунг *"структура связей — все, свойства элементов — ничто"*.

Совокупность идей и научно-техническое направление, определяемое описанным представлением о мозге, называется *коннекционизмом* [204] (по-английски connection — связь). Как все это соотносится с реальным мозгом [203]? Так же, как карикатура или шарж со своим прототипом-человеком [163]. Это нормально: в бионике важно не соответствие живому прототипу, а продуктивность технической идеи.

С коннекционизмом тесно связан следующий блок идей:

1) однородность системы (элементы одинаковы и чрезвычайно просты, все определяется структурой связей);

2) надежные системы из ненадежных элементов и "аналоговый ренессанс" — использование простых аналоговых элементов;

3) "голографические" системы — при разрушении случайно выбранной части система сохраняет свои полезные свойства.

Предполагается, что система связей достаточно богата по своим возможностям и избыточна, чтобы скомпенсировать бедность выбора элементов, их ненадежность, возможные разрушения части связей.

Коннекционизм и связанные с ним идеи однородности, избыточности и голографичности еще ничего не говорят о том, как же такую систему научить решать реальные задачи. Хотелось бы, чтобы это обучение обходилось не слишком дорого.

На первый взгляд кажется, что коннекционистские системы не допускают прямого программирования, т.е. формирования связей по явным правилам. Это, однако, не совсем так. Существует большой класс задач: нейронные системы ассоциативной памяти, статистической обработки, фильтрации и др., для которых связи формируются по явным формулам [172, 241, 248]. Но еще больше (по объему существующих приложений) задач требуют *неявного* процесса [54, 98, 99, 131, 137, 180, 237, 239]. По аналогии с обучением животных или человека этот процесс мы также называем обучением.

Обучение обычно строится так [2, 7, 10, 54, 171, 183, 264, 265]: существует задачник — набор примеров с заданными ответами. Эти примеры предъявляются системе. Нейроны получают по входным связям сигналы — "условия примера", преобразуют их, несколько раз обмениваются преобразованными сигналами и, наконец, выдают ответ — также набор сигналов. Отклонение от правильного ответа штрафуется. Обучение состоит в *минимизации* штрафа как (неявной) функции связей. Примерно четверть нашей книги состоит в описании техники такой оптимизации и возникающих при этом дополнительных задач.

Неявное обучение приводит к тому, что структура связей становится "непонятной" — не существует иного способа ее прочитать, кроме как запустить функционирование сети. Сложно ответить на вопрос: как нейронная сеть получает результат?, т.е. построить понятную человеку логическую конструкцию, воспроизводящую действия сети.

Это явление можно назвать "логической непрозрачностью" нейронных сетей, обученных по неявным правилам. В работе с такими сетями иногда оказываются полезными представления, разработанные в психологии и педагогике, и обращение с обучаемой сетью как с дрессируемой зверушкой или с обучаемым младенцем — это еще один источник идей. Возможно, со временем возникнет такая область деятельности — "нейропедагогика" — обучение искусственных нейронных сетей.

Кроме того, при использовании нейронных сетей в экспертных системах на PC возникает потребность прочитать и логически проинтерпретировать навыки, выработанные сетью. В книге описаны созданные для этого алгоритмы контрастирования [41, 54, 234—236] —

получения неявными методами "логически прозрачных" нейронных сетей [63, 64]. Однако за логическую прозрачность приходится платить снижением избыточности, так как при контрастировании удаляются все связи кроме самых важных, без которых задача не может быть решена.

Итак, очевидно наличие двух источников идеологии нейроинформатики. Это представления о строении мозга и о процессах обучения. Существуют группы исследователей и научные школы, для которых эти источники идей имеют символическое, а иногда даже мистическое или тотемическое значение.

Не будем останавливаться на аналогиях из физики (спиновые стекла), общей биологии (теория отбора) и т.п. На наш взгляд, эти источники вдохновения не общезначимы, а связаны с биографией отдельных исследователей.

Есть, однако, еще один источник, связанный с особым представлением алгоритмов параллельно-последовательной обработки данных. Оно ориентировано на создание специализированных сопроцессоров — устройств, параллельно выполняющих большое число однотипных операций с данными.

Хотя общепризнанной формализации еще нет, существуют и идея, и различные варианты устройств, предназначенных для реализации этой идеи. Чтобы дать достаточно общее ее изложение, не сводящееся к описанию частного варианта устройства, нам придется пожертвовать формальной определенностью.

Практически все формализации, а тем более реализации, предполагают существование особого устройства (формального или реального), которое получает вектор данных и вычисляет некоторые его функции, которые могут зависеть от набора параметров (настраиваемых). При описании функционирования устройства предполагается, что заранее определены рабочие варианты значений параметров (проведена настройка). Эти значения хранятся в особом устройстве памяти с быстрым доступом.

Особенно важна организация алгоритмов настройки — поиска значений параметров, необходимых для обработки данных. Требуется, чтобы алгоритмы настройки эффективно использовали то же самое устройство (сопроцессор) и передавали ему для исполнения те однотипные вычисления, объем которых растет пропорционально размерности вектора данных или быстрее.

Представим работу сопроцессора более формально. Пусть дано некоторое конечное множество F функций от двух векторных переменных x, a. (Далее x будет вектором данных, a — вектором параметров.) Пусть, кроме того, определено конечное множество Ф функций одной переменной. Тогда сопроцессор выполняет следующие операции:
1) вычислить (для вектора данных x, набора функций $f_1 \in F$ и соответствующего набора векторов параметров a_i) вектор y с координатами

$$y_i = f_i(x, a_i);$$

2) для данного $\varphi \in \Phi$ по вектору данных x вычислить вектор y с координатами

$$y_i = \varphi(x_i).$$

При некоторых ограничениях на размерность векторов предполагается, что эти операции выполняются параллельно.

Обычный "джентльменский набор" в нейрокомпьютерных построениях: функции $f(x, a)$ представляют собой скалярные произведения

$$f(x, a) = (x,a) \left(= \sum_j x_j a_j\right);$$

множество Ф состоит из небольшого числа функций (двух-трех), например,

$$\varphi_1(x_i) = \frac{x_i}{c + |x_i|};\ \varphi_2(x_i) = |x_i| \text{ или}$$

$$\varphi_1(x_i) = \text{th}(x_i);\ \varphi_2(x_i) = |x_i|;\ \varphi_3(x_i) = x_i^2.$$

Очевидны возможные обобщения, такие как наличие настраиваемого параметра у некоторых функций из Ф.

Представление о специализированном сопроцессоре и специальном устройстве памяти для настраиваемых параметров так же, как и нейрофизиология или педагогическая психология, служит источником идей и постановки задач в нейроинформатике.

Для нас в этой книге нейронные сети, в первую очередь, являются средством решения задач на PC. Они обладают рядом дополнительных преимуществ, связанных с высоким параллелизмом и возможностью применять специализированные сопроцессоры, но наш опыт показывает, что нейросетевая технология обработки данных хороша даже без использования параллелизма.

Глава 1

ЭЛЕМЕНТЫ НЕЙРОННЫХ СЕТЕЙ

1. Перечень элементов

Для описания алгоритмов и устройств в нейроинформатике выработана специальная "схемотехника", в которой элементарные устройства — сумматоры, синапсы, нейроны и т.п. — объединяются в сети, предназначенные для решения задач. Интересен статус схемотехники — для многих начинающих кажется неожиданным, что ни в аппа-

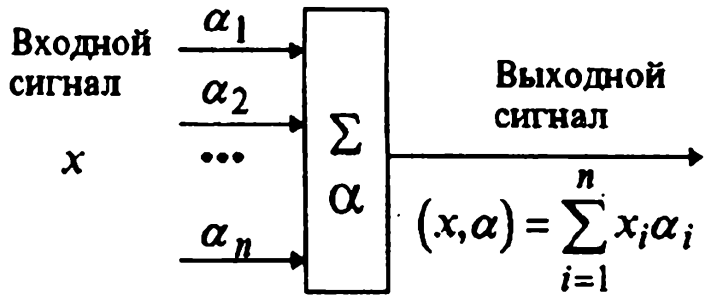


Рис. 1.1. Адаптивный сумматор.

Рис. 1.2. Нелинейный преобразователь.

ратной реализации нейронных сетей, ни в профессиональном программном обеспечении все эти элементы вовсе не обязательно реализуются как отдельные части или блоки. Используемая в нейроинформатике идеальная схемотехника представляет собой особый язык для представления нейронных сетей и их обсуждения. При программной и аппаратной реализации выполненные на этом языке описания переводятся на языки другого уровня, более для этого пригодные.

Самый заслуженный и, вероятно, наиболее важный элемент нейросистем — это *адаптивный сумматор*. Он вычисляет скалярное произведение вектора входного сигнала x на вектор параметров a. На схемах будем обозначать его так, как показано на рис. 1.1. Адаптивным называем его из-за наличия вектора настраиваемых параметров a.

Нелинейный преобразователь сигнала (рис. 1.2) получает скалярный входной сигнал x и переводит его в $\varphi(x)$.

Последовательным соединением адаптивного сумматора с нелинейным преобразователем получаем *нейрон* (рис. 1.3).

В ряде задач оказывается полезным разложить адаптивный сумматор на набор из n параллельных линейных связей и следующий за ними простой сумматор.

Линейная связь (синапс — рис. 1.4) имеет один настраиваемый параметр a, получает на входе скалярный сигнал x и выдает на выходе произведение ax.

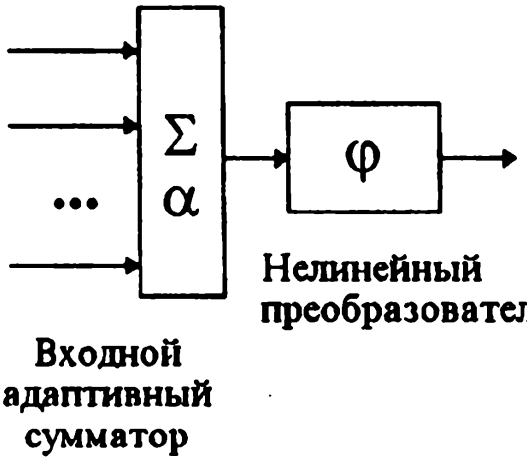
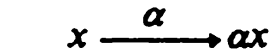


Рис. 1.3. Нейрон.

Рис. 1.4. Линейная связь.



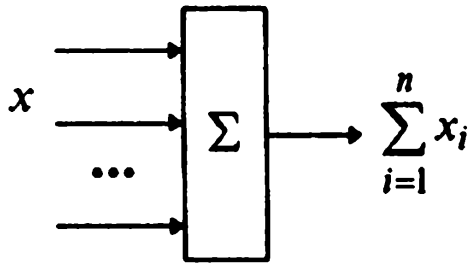
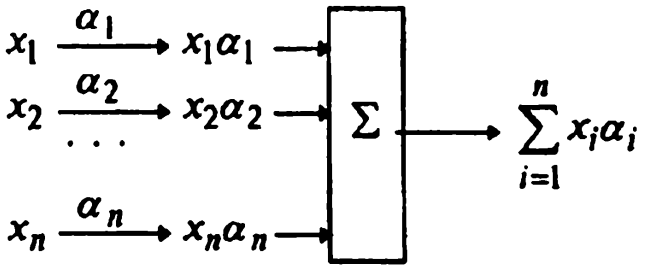


Рис. 1.5. Простой сумматор.

Рис. 1.6. Адаптивный сумматор, представленный как композиция n линейных связей и простого сумматора.



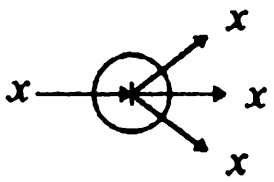


Рис. 1.7. Точка ветвления.

У простого сумматора (рис. 1.5) нет настраиваемых параметров. Он имеет векторный вход и выдает на выходе сумму координат входного сигнала.

Адаптивный сумматор может быть представлен с использованием n линейных связей и простого сумматора (рис. 1.6).

Точка ветвления служит для рассылки одного сигнала по нескольким адресам (рис. 1.7). Она получает скалярный входной сигнал x и передает его всем своим выходам.

В большинстве случаев на выходе каждого нейрона стоит точка ветвления, служащая для рассылки выходного сигнала нейрона по входам других элементов (рис. 1.8).

Выходная звезда. Иногда полезно объединять синапсы и точки ветвления, образуя элемент, называемый выходной звездой. На входе он получает один сигнал (A), на k своих выходах выдает сигналы a_iA $(i = 1,...,k)$. Выходная звезда — элемент, двойственный адаптивному сумматору, — переходит в него, если обратить распространение сигналов. Аналогично этому точка ветвления двойственна простому сумматору.

Элементы, представленные на рис. 1.1—1.8, являются основными и стандартными в нейроинформатике. Существует, однако, множество дополнений и вариаций. Наметим некоторые из них, не претендуя на полноту.

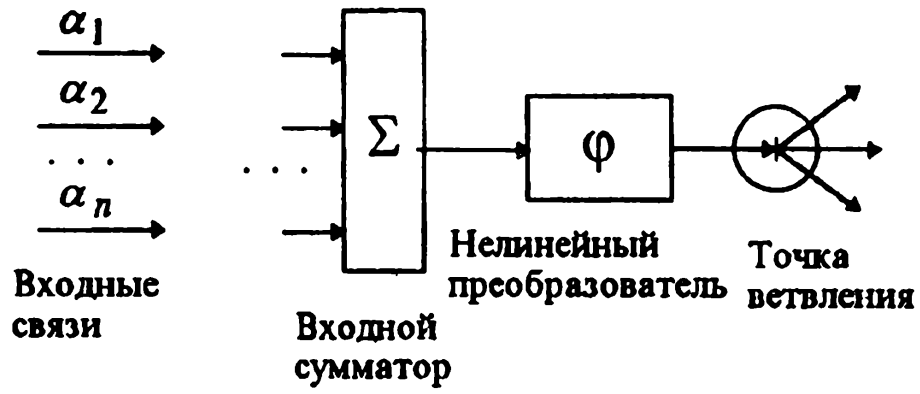


Рис. 1.8. Нейрон с точкой ветвления на выходе.

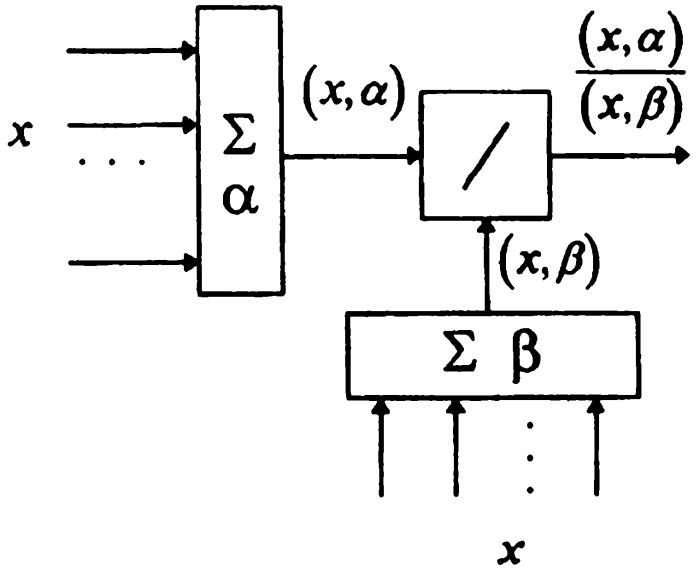


Рис. 1.9. Паде-нейрон.

Паде-нейрон [214, 223, 227] (рис. 1.9) состоит из двух адаптивных сумматоров и элемента с двумя выходами, вычисляющего частное от своих входных сигналов. Получая на входе n-мерный вектор x, Паде-нейрон выдает на выходе отношение $(x, a) / (x, \beta)$, имеет $2n$ настраиваемых параметров — координат векторов α, β. Название происходит от аппроксимации Паде — метода рациональной аппроксимации функций.

Квадратичный адаптивный сумматор (рис. 1.10) — элемент, вычисляющий квадратичную форму от входного сигнала. При размерности входного сигнала n квадратичный сумматор имеет $n(n + 1)/2$ настраиваемых параметров — коэффициентов квадратичной формы $Q_{ij}(q_{ij} = q_{ji})$.

Счетчик совпадений — элемент, получающий на входе n-мерный двоичный вектор x (с координатами 0, 1) и имеющий n-мерный двоичный вектор настраиваемых параметров a. На выход выдает целое число — число совпадений значений x_i с a_i, т.е. число тех i, для которых $x_i = a_i$.

В определении адаптивного сумматора (рис. 1.1) фигурирует однородная линейная функция (x, a). Небольшое изменение способа по-

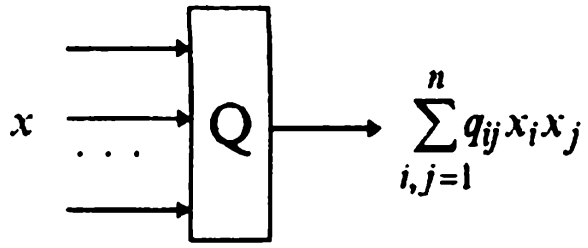
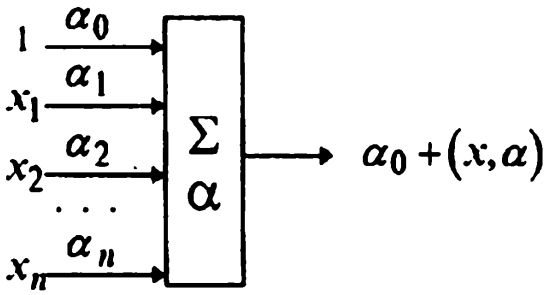


Рис. 1.10. Квадратичный адаптивный сумматор.

Рис. 1.11. Адаптивный сумматор, вычисляющий неоднородную линейную функцию вектора входных сигналов.



дачи сигналов (рис. 1.11) позволяет вычислять произвольную линейную неоднородную функцию. Для этого нужно добавить еще один вход, на который подается стандартный единичный сигнал, и увеличить на единицу размерность вектора настраиваемых параметров a. Получаем важнейший новый элемент — *неоднородный адаптивный сумматор.*

2. Задачи для одноэлементных систем

В нейроинформатике задачи решаются с помощью нейронных сетей — систем из многих взаимосвязанных нейронов. Однако и одноэлементные системы имеют свою достаточно широкую область применения. В этом разделе дан краткий обзор задач, решаемых с помощью одноэлементных систем, а в следующем описаны алгоритмы настройки параметров элементов для решения задач.

Есть необходимость вычислять линейные функции вектора данных в огромном числе приложений. Вот лишь некоторые из них: кодирование, декодирование, фильтрация (выделение сигнала из шума), статистическая обработка. Линейные функции используются также в простейших задачах классификации для двух классов: в них x — вектор признаков объекта, а принадлежность объекта к первому или второму классу определяется по знаку линейной функции:

если $a_0 + (x, a) > 0$, то объект принадлежит к первому классу;

если $a_0 + (x, a) \leqslant 0$, то объект принадлежит ко второму классу.

В этом примере мы впервые сталкиваемся с интерпретатором ответов — устройством или программой, которые должны обработать выходной сигнал и сформулировать решение. Описанный интерпретатор очень прост — он проверяет знак выходного сигнала. Это действие можно представить пороговым элементом. Пусть характеристика нелинейного преобразователя $\varphi(x)$ — функция Хевисайда (пороговый элемент)

$$\varphi(x) = \begin{cases} 1, \text{ если } x > 0; \\ 0, \text{ если } x \leqslant 0. \end{cases} \tag{1.1}$$

Тогда задача линейной классификации на два класса решается нейроном (рис. 1.3), у которого $\varphi(x)$ — функция Хевисайда. Интерпретация ответа совсем проста: выходной сигнал 1 означает принадлежность объекта к первому классу, а 0 — ко второму. Такой нейрон с пороговым нелинейным элементом называется *перцептроном.*

Паде-нейрон вычисляет произвольную дробно-линейную функцию вектора x. Так же как и для адаптивного сумматора, числитель и знаменатель можно сделать неоднородными линейными функциями x: $(a_0 + (x, a))/(\beta_0 + (x, \beta))$. Являясь элементом, более гибким, чем адаптивный сумматор, Паде-нейрон может использоваться как его обобщение в тех случаях, когда линейных функций становится недостаточно, в частности в задачах регрессии, для построения и интерполяции эмпирических зависимостей.

Квадратичный адаптивный сумматор так же, как и линейный сумматор, с помощью дополнительного стандартного единичного сигнала может вычислять произвольный полином второго порядка от вектора входных сигналов:

$$Q(x) = \sum_{ij} q_{ij} x_i x_j + \sum_i p_i x_i + r. \tag{1.2}$$

Задачи все те же: регрессия, классификация и др. Особо следует подчеркнуть, что для многомерных нормальных распределений квадратичный классификатор является наилучшим среди всех других мыслимых элементов. Минимум вероятности ошибки дает квадратичная разделяющая поверхность:

если $Q(x) > 0$, то объект принадлежит к первому классу;

если $Q(x) < 0$, то объект принадлежит ко второму классу

(при условии правильного выбора коэффициентов $Q(x)$).

Поэтому аналог перцептрона с квадратичным сумматором и пороговым элементом на выходе очень привлекателен для решения задач бинарной классификации. Его недостаток — большое число настраиваемых параметров — может быть преодолен с помощью специальной предварительной обработки данных (см. следующий раздел).

3. Настройка параметров одноэлементных систем для решения задач

Линейная локальная фильтрация. Если имеется последовательность сигналов с шумом, то можно попытаться уменьшить его влияние, предполагая, что соседние сигналы более скоррелированы, чем соответствующие возмущения. В таком случае, сглаживая сигнал, уменьшаем относительную долю шума. В этом состоит идея сглаживающих фильт-

ров. Простейший пример — для линейной последовательности сигналов x_i строим преобразование (среднее арифметическое)

$$x_i' = \frac{1}{2k+1} \sum_{j=-k}^{j=k} x_{i+j}. \tag{1.3}$$

При необходимости учесть ослабление связи с ростом $|j|$ можно положить

$$x_i' = \frac{1-r}{1+r-2r^{k+1}} \sum_{j=-k}^{j=k} r^{|j|} x_{i+j}, \tag{1.4}$$

где $|r| < 1$. Удобно выбирать в качестве r оценку коэффициента корреляции соседних значений сигнала x_i.

Аналогичные формулы применяются для фильтрации изображений — двумерных массивов сигналов x_{ij}. Общая форма локального сглаживающего фильтра для изображений

$$x_{ij}' = \sum_{l,q=-k}^{k} a_{lq} x_{i+1,j+q} \tag{1.5}$$

при условии $\sum_{l,q=-k}^{k} a_{lq} = 1$.

Так же как и для линейных последовательностей, удобно полагать

$$a_{lq} = r^{|l|+|q|} / \sum_{l,q=-k}^{k} r^{|l|+|q|}, \tag{1.6}$$

где r — оценка коэффициента корреляции соседних пикселей изображения.

Часто используются фильтры с априорными значениями коэффициентов. Например, для изображений

$$(\alpha_{ql}) = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ или } \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \tag{1.7}$$

$q, l = -1,0,1$, центральный элемент матрицы соответствует значениям $l = q = 0$.

Кроме подавления шумов, линейные преобразования сигналов могут использоваться для выделения характерных точек (линий), в которых сигнал (изображение) резко меняется. Фильтры, подчеркивающие перепады значений, для одномерных массивов имеют вид

$$x_i' = -x_{i-1} + 3x_i - x_{i+1} \text{ или } x_i' = -x_{i-1} + 2x_i - x_{i+1}; \tag{1.8}$$

для изображений

$$x_{ij}' = \sum_{l,q=-k}^{k} a_{lq} x_{i+l,j+q} \qquad (1.9)$$

со следующими вариантами a_{lq}:

$$(a_{lq}) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \text{ или } \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \text{ или } \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix},$$

или (операторы Лапласа):

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 1 & -1 & 0 \end{bmatrix} \text{ или } \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \text{ или } \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}.$$

Первые три фильтра сохраняют области постоянной яркости, последние три обращают для них сигнал в нуль.

Линейный локальный фильтр — это один сумматор с априори фиксированными настраиваемыми параметрами. Предполагается, что он сканирует последовательность сигналов. Существует два принципиально разных порядка сканирования: рекурсивный и нерекурсивный. При нерекурсивном сканировании преобразованные сигналы получаются из исходных постоянным линейным преобразованием — на вход сумматору каждый раз подаются исходные значения x_i (или x_{ij} для изображений). Рекурсивное сканирование отличается тем, что после каждого вычисления нового (отфильтрованного) значения сигнала x_i' (или x_{ij}') оно записывается на место исходного, и на следующем шаге фильтрации сумматору подается уже новый вектор сигнала, часть координат которого преобразована, а часть — нет. Для рекурсивных фильтров приобретает значение порядок фильтрации. При обработке линейных массивов таких порядков по существу два (справа налево или слева направо), если в фильтрации участвует только один сумматор. Для двумерных изображений вариантов существенно больше. Обычно используется развертка по строкам или столбцам, хотя в некоторых системах лучше применять такие развертки, как спиральная, идущая от центра к границе по спирали, или развертка Гильберта (см. разд. 1.6).

Вычисление выборочного коэффициента ковариации вектора данных с эталоном. Эта задача возникает очень часто. Пусть заданы n-мерные вектор данных $x = (x_i)$ и вектор эталона $(y = y_i)$. При определении коэффициента ковариации числа x_i, y_i рассматриваются как реализации некоторых случайных величин. Напомним определения выборочного среднего $\mathrm{M}(x)$, выборочного коэффициента ковариации $\mathrm{cov}(x, y)$ и коэффициента корреляции $r(x, y)$:

$$\mathrm{M}(x) = \frac{1}{n} \sum_i x_i;$$

$$\operatorname{cov}(x, y) = \mathbf{M}\left((x - \mathbf{M}(x))(y - \mathbf{M}(y))\right) = \frac{1}{n}\sum_i \left(x_i - \frac{1}{n}\sum_j x_j\right) \times$$

$$\times \left(y_i - \frac{1}{n}\sum_j y_j\right) = \frac{1}{n}\sum_i x_i\left(y_i - \frac{1}{n}\sum_j y_j\right). \qquad (1.10)$$

Эта задача решается сумматором с настраиваемыми параметрами

$$a_i = \frac{1}{n}\left(y_i - \frac{1}{n}\sum_j y_j\right). \qquad (1.11)$$

Коэффициент корреляции векторов x и y определяется как

$$r(x, y) = \operatorname{cov}(x, y)/\sigma_x\sigma_y, \qquad (1.12)$$

где выборочное среднеквадратичное уклонение σ_x — также "нейросетевая" функция:

$$\sigma_x = \left(\sum_i \left(x_i - \frac{1}{n}\sum_j x_j\right)^2\right)^{1/2} \qquad (1.13)$$

($\mathbf{M}(x)$ находится сумматором, далее для каждой координаты вектора $x_i - \mathbf{M}(x)$ вычисляется функция одной переменной $\varphi(x) = x^2$, далее — снова сумматор и, наконец, функция одной переменной $x^{1/2}$).

Линейная аппроксимация функции, заданной в конечном числе точек (задача линейной регрессии). Пусть для некоторого набора векторов $x^1,\ldots,x^n$ определены значения функции $f(x^1) = f_1,\ldots, f(x^n) = f_n$. Требуется построить наилучшую линейную аппроксимацию f, т.е. найти такой вектор a, чтобы приближение $f(x) \approx (x, a)$ давало наименьшую ошибку.

В н и м а н и е! Здесь в очень простой ситуации впервые появляется важный объект — оценка качества работы устройства. Мы будем всегда определять оценку так, чтобы ее уменьшение соответствовало улучшению работы.

Наиболее распространенный вариант определения оценки — метод наименьших квадратов

$$H(a) = \sum_j [f(x^j) - (x^j, a)]^2. \qquad (1.14)$$

Когда оценка определена, задача сводится к минимизации функции $H(a)$. Для квадратичной функции оценки H минимизацию можно проводить различными способами.

Приравнивая к нулю все частные производные $\partial H(a)/\partial a_i$, получаем систему линейных уравнений

$$\frac{\partial H(a)}{\partial a_i} = -2\sum_j (f(x^j) - (x^j, a))x_i^j = 0 \qquad (1.15)$$

из n уравнений (по одному для каждого i). Решая эту систему любым способом, находим вектор a.

Градиентная минимизация $H(a)$. Вычисляем вектор градиента $\mathrm{grad}H$ с координатами $\partial H/\partial a_i$. Для достаточно малых шагов h верно неравенство

$$H(a - h\mathrm{grad}H) < H(a). \tag{1.16}$$

Это обстоятельство формулируют так: антиградиент (минус градиент, т.е. $-\mathrm{grad}H$) есть направление спуска для $H(a)$[1]. Заменяем a на $a - h \times \mathrm{grad}H$, вновь вычисляем градиент H, снова делаем шаг по антиградиенту и т.д., пока с достаточной точностью не достигнем минимума.

В этом алгоритме наискорейшего спуска два момента требуют уточнения: выбор шага h и выбор момента остановки. Выбор шага может быть сделан следующим образом. При $h = 0$ значение $H(a) = H_0$ известно. Вычисляем для некоторого h_1 величину $H_1 = H(a - h_1\mathrm{grad}H)$. Если она меньше $H(a)$, то увеличиваем шаг вдвое, $h_2 = 2h_1$, если больше, то уменьшаем его, $h_2 = h_1/2$. В новой точке вычисляем $H_2 = H(a - h_2\mathrm{grad}H)$ (вектор $\mathrm{grad}H$ — все тот же, найденный в точке a). Для трех значений $h = 0, h_1, h_2$ оценка имеет три значения: H_0, H_1, H_2. Замечаем, что $H(a - h\mathrm{grad}H)$ — квадратичная функция h (как ограничение квадратичной формы на прямую). По трем значениям строим квадратный трехчлен $ah^2 + bh + c$,

$$c = H_0,\ ah_1^2 + bh_1 + c = H_1,\ ah_2^2 + bh_2 + c = H_2. \tag{1.17}$$

Находим точку минимума этого трехчлена: $h_{\min} = -b/2a$. Выбираем шаг равным $h_{\min}$. Новое значение a равно $a - h_{\min} \times \mathrm{grad}H$, снова вычисляем для него градиент и т.д.

Вопрос о выборе момента остановки пока обсуждать не будем.

Формула Уидроу. Обращаемся к выражению для $\partial H/\partial a_i$. Оно имеет интересную структуру. Если обозначить ошибку в j-й строке (x^j) через $\Delta_j = f(x^j) - (x^j, a)$, то эта структура станет особенно прозрачной:

$$\frac{\partial H}{\partial a_i} = -2\sum_j x_i^j \Delta_j. \tag{1.18}$$

Под знаком суммы стоит произведение ошибки в j-й точке на входной сигнал, поступающий на i-й вход сумматора (так же в j-й точке). В том случае, когда рассматривается ошибка H для одного вектора сигнала x, получаем [171]

---

[1]Направлением спуска для функции H в точке x называется любой вектор y, для которого $H(x + Ey) \leqslant H(x)$ при достаточно малых $E > 0$.

$$\frac{\partial H}{\partial a_i} = -2x_i\Delta. \qquad (1.19)$$

Нетрудно аппаратно организовать вычисление этих величин. Существует способ адаптации (настройки параметров a), работающий с поточечными оценками $H_j = [f(x^j) - (x^j, a)]^2$. В этом случае для каждого x^j делается один шаг коррекции a по антиградиенту H_j, после чего обрабатывается другое (следующее) x^j. При работе с поточечными оценками H_j выбор шага $h = h_{\min}$ может оказаться слишком радикальным и привести к потере устойчивости — параметр a при переходе от точки к точке начнет совершать колебания увеличивающейся амплитуды. Поэтому шаг выбирают достаточно маленьким исходя из соображений устойчивости.

Дробно-линейная аппроксимация. Вновь имеем *"обучающую выборку"* — набор значений функции $f_1,\ldots,f_n$ в некотором наборе точек $x^1,\ldots,x^n$:

$$f(x^i) = f_i. \qquad (1.20)$$

Требуется построить наилучшую дробно-линейную аппроксимацию, т.е. такие векторы a, β, чтобы приближение

$$f(x) = (x, a)/(x, \beta) \qquad (1.21)$$

давало наименьшую ошибку.

Так же как и для линейной аппроксимации, использование неоднородных линейных функций в числителе и знаменателе не вносит в задачу ничего принципиально нового — достаточно увеличить размерность векторов x^i на единицу, добавив к ним координату $x_0^i = 1$ и соответственно дополнив векторы a, β координатами a_0, β_0 (рис. 1.11).

Условия точного равенства

$$(x^i, a)/(x^i, \beta) = f_i \qquad (1.22)$$

преобразуются в систему линейных уравнений относительно a, β:

$$(x^i, a) = (x^i, \beta)f_i \text{ или } (x^i, a - f_i\beta) = 0. \qquad (1.23)$$

Эти уравнения однородны и ненулевое решение a, β, если оно существует, всегда неединственно — одновременно с ним решением является пара λa, $\lambda\beta$ для любого λ.

Для подбора параметров a, β минимизацией ошибок лучше выбрать исходное дробно-линейное выражение. Для него квадраты ошибки во всех точках x^j входят в общую оценку с равными (единичными) коэффициентами:

$$H \cong \sum_j \left(f_j - \frac{(x^j, a)}{(x^j, \beta)}\right)^2. \qquad (1.24)$$

Градиент H (частные производные H по a_i, β_i) легко вычисляется. Удобно представить это вычисление по частям.
Обозначим Δ_j ошибку аппроксимации в j-й точке:

$$\Delta_j = f_j - A_j/B_j;$$

$$A_j = (x^j, a), \quad B_j = (x^j, \beta);$$

$$H = \sum_j \Delta_j^2; \quad \Delta_{A_j} = \frac{\partial(\Delta_j^2)}{\partial A} = -2\Delta_j/B_j; \quad \Delta_{B_j} = \frac{\partial(\Delta_j^2)}{\partial B} = 2\Delta_j A_j/B_j^2; \quad (1.25)$$

$$\frac{\partial(\Delta_j^2)}{\partial a_i} = x_i^j \Delta_{A_j};$$

$$\frac{\partial(\Delta_j^2)}{\partial \beta_i} = x_i^j \Delta_{B_j}.$$

Это аналог формулы Уидроу для Паде-нейрона. Градиентная оптимизация ошибки H проводится так же, как и для обычных адаптивных сумматоров с единственным отличием: если для квадратичных по параметрам a оценок H оптимальное значение шага находится явно после вычисления H_0, H_1, H_2, то в общем случае одномерная оптимизация в направлении $-\mathrm{grad}H$ может потребовать несколько шагов (обычно $3 \div 5$).

Квадратичная аппроксимация даже проще, чем дробно-линейная, поскольку функция $\sum_j q_{ij} x_i x_j$ линейно зависит от искомых коэффициентов q_{ij} и функция оценки H квадратично зависит от этих параметров. Решение этой задачи такое же, как и для линейной аппроксимации. Аналог формулы Уидроу для квадратичных сумматоров:

$$\frac{\partial(\Delta_j^2)}{\partial q_{ik}} = -2x_i^j x_k^j \Delta_j, \quad (1.26)$$

где x^j — точки из обучающей выборки; Δ_j — ошибки аппроксимации в этих точках.

Задача четкого разделения двух классов по обучающей выборке ставится так: имеется два набора векторов $x^1,\ldots,x^n$ и $y^1,\ldots,y^m$. Заранее известно, что x^i относится к первому классу, а y^i — ко второму. Требуется построить *решающее правило*, т.е. определить такую функцию $f(x)$, что при $f(x) > 0$ вектор x относится к первому классу, а при $f(x) < 0$ — ко второму.

Координаты классифицируемых векторов представляют собой значения некоторых признаков (свойств) исследуемых *объектов*. Строго говоря, классифицируются не векторы свойств, а объекты, которые обладают этими свойствами. Это замечание становится важным в тех

случаях, когда возникают затруднения с построением решающего правила, например тогда, когда встречаются принадлежащие к разным классам объекты, имеющие одинаковые признаки. В этих случаях возможно несколько решений:

1) искать дополнительные признаки, позволяющие разделить классы;

2) примириться с неизбежностью ошибок, назначить за каждый тип ошибок свой штраф (c_{12} — за то, что объект первого класса отнесен ко второму; c_{21} — за то, что объект второго класса отнесен к первому) и строить разделяющее правило так, чтобы минимизировать риск;

3) перейти к нечеткому разделению классов — строить так называемые "функции принадлежности" $f_1(x)$ и $f_2(x)$; $f_i(x)$ оценивает степень уверенности при отнесении объекта к i-му классу ($i = 1, 2$), для одного и того же x может быть так, что и $f_1(x) > 0$, и $f_2(x) > 0$.

Линейное разделение классов состоит в построении линейного решающего правила, т.е. таких вектора a и числа a_0 (называемого также порогом), что при $a_0 + (x, a) > 0$ x относится к первому классу, а при $a_0 + (x, a) < 0$ — ко второму. Один из способов найти такие a и a_0 — потребовать, чтобы правило было безошибочным на обучающей выборке, и решить систему линейных неравенств

$$\begin{cases} a_0 + (x^i, a) > 0 \quad (i = 1, ..., n); \\ a_0 + (y^i, a) < 0 \quad (j = 1, ..., m). \end{cases} \tag{1.27}$$

Здесь x^i $(i = 1,...,n)$ — векторы из обучающей выборки, относящиеся к первому классу, а y^j $(j = 1,...,n)$ — ко второму. Удобно переформулировать задачу. Увеличим размерности всех векторов на единицу, добавив еще одну координату a_0 к a, $x_0 = 1$ — ко всем x^i и $y_0 = 1$ — ко всем y^j. Сохраним для новых векторов прежние обозначения — это не приведет к путанице. Наконец, положим $z^i = x^i$ $(i = 1,...,n)$, $z^{n+j} = -y^j$ $(j = 1,...,m)$. Тогда получим систему $n + m$ неравенств

$$(z^i, a) > 0 \quad (i = 1,...,n + m). \tag{1.28}$$

Если множество ее решений непусто, то любой его элемент a порождает решающее правило, безошибочное на обучающей выборке.

Разделение центров масс — простейший способ построения решающего правила. Выберем в качестве a центр масс векторов z^i:

$$a = \bar{z} = \frac{1}{n + m} \sum_{i=1}^{n+m} z^i. \tag{1.29}$$

Если n и m существенно различаются, либо ошибки отнесения объекта существенно неравнозначны, то имеет смысл приписывать векторам из различных классов разную массу:

$$a = \frac{1}{nP_1 + mP_2}\left(P_1\sum_{i=1}^{n} z^i + P_2\sum_{i=n+1}^{n+m} z^i\right). \qquad (1.30)$$

Этот же прием можно использовать и в том случае, когда значимость (цена) ошибки классификации сильно зависит от класса объекта.

Вес P_j растет с увеличением значимости неправильного отнесения объекта j-го класса и с уменьшением числа объектов j-го класса в обучающей выборке.

Большие уклонения — векторы x^i (или y^i), значительно отклоняющиеся от соответствующего центра масс $\bar{x}$ (или $\bar{y}$), — могут вносить существенный вклад в a. Если большие уклонения считаются случайными ошибками, то их вклад можно уменьшить следующим образом. Положим

$$M_1 = \frac{1}{n}\sum_{i=1}^{n} z^i, \quad M_2 = \frac{1}{m}\sum_{i=n+1}^{n+m} z^i;$$
$$D_1 = \frac{1}{n}\sum_{i=1}^{n} (z^i - M_1)^2, \quad D_2 = \frac{1}{m}\sum_{i=n+1}^{n+m} (z^i - M_2)^2. \qquad (1.31)$$

Выберем некоторую монотонную функцию $w(p)$, обладающую свойствами $w(0) = 1$, $w(p) \to 0$ при $p \to \infty$, например,

$$w(p) = e^{-p}, \quad w(p) = e^{-p^2} \text{ или } w(p) = \frac{1}{1 + p^2}. \qquad (1.32)$$

Желательно, чтобы на отрезке [0, 1] функция $w(p)$ принимала значения порядка 1, а далее быстро убывала. Положим

$$w_i = w\left(\frac{\| z^i - M_1 \|^2}{D_1}\right) \text{ при } i = 1,\ldots,n$$
и
$$w_i = w\left(\frac{\| z^i - M_2 \|^2}{D_2}\right) \text{ при } i = n+1,\ldots,m. \qquad (1.33)$$

Определим центр масс с учетом этих весов:

$$a = \frac{1}{nP_1\sum_{i=1}^{n} w_i + mP_2\sum_{i=n+1}^{n+m} w_i}\left(P_1\sum_{i=1}^{n} w_i z^i + P_2\sum_{i=n+1}^{n+m} w_i z^i\right). \qquad (1.34)$$

При определении a по этой формуле снижается вклад больших уклонений.

Обучение персептрона. Линейные решающие правила, построенные на основании разделения центров масс, могут ошибаться на примерах из обучающей выборки даже в тех случаях, когда возможно и безошибочное линейное разделение. Пусть такие ошибки обнаружены.

Будем их исправлять, увеличивая на единицу массу каждого z^i, для которого $(z^i, a) < 0$. Эта коррекция ошибок в конце концов приводит к безошибочному разделению обучающей выборки, если такое разделение вообще возможно. Однако заранее неизвестно, сколько раз нужно пройти обучающую выборку, чтобы добиться желаемого результата.

Алгоритм обучения персептрона сыграл большую роль в истории нейроинформатики, поэтому опишем его детально, начиная с постановки задачи.

Итак, в k-мерном пространстве задана обучающая выборка $x^1,...,x^n$ (первый класс) и $y^1,...,y^m$ (второй класс). Требуется построить линейную функцию, отделяющую все x^i от y^i, т.е. найти такие k-мерный вектор a и число a_0, что для всех $i = 1,...,n$ и $j = 1,...,m$ $a_0 + (x^i, a) > 0$ и $a_0 + (y^j, a) < 0$.

Переформулируем задачу, сведя ее к отделению нуля от конечного множества векторов в $k + 1$-мерном пространстве. Для этого рассмотрим $k + 1$-мерные векторы $\tilde{a}$ и z^i $(i = 1,...,n + m)$:

$$\tilde{a}_0 = a_0,\ \tilde{a}_l = a_l \text{ при } l = 1,...,k;$$

$$z_0^i \equiv 1;\ z_l^i = x_l^i \text{ при } i = 1,...,n,\ l = 1,...,k; \tag{1.35}$$

$$z_0^{n+i} \equiv -1,\ z_l^{n+i} = -y_l^i \text{ при } i = 1,...,m,\ l = 1,...,k.$$

В этих обозначениях условия разделения превращаются в систему $n + m$ однотипных неравенств:

$$(z^i, \tilde{a}) > 0 \text{ при } i = 1,...,n + m. \tag{1.36}$$

В качестве нулевого приближения можно выбрать любой вектор, например $\tilde{a} = 0$. Цикл алгоритма состоит в том, что для всех $i = 1,...,n + m$ проверяется неравенство $(z^i, \tilde{a}) > 0$. Если для данного $i \leqslant m + n$ оно выполнено, переходим к следующему i (либо при $i = m + n$ заканчиваем цикл), если же не выполнено, то модифицируем $\tilde{a}$:

$$\tilde{a} := \tilde{a} + z^i. \tag{1.37}$$

Когда за весь цикл нет ни одной ошибки (т.е. модификации $\tilde{a}$), то решение $\tilde{a}$ найдено, в случае же ошибок полагаем $i = 1$ и снова проходим цикл.

Замечания. 1. Поскольку вектор $\tilde{a}$ определен с точностью до положительного множителя, то нормировку (деление на сумму масс) не проводим.

2. Хотя в качестве начального приближения можно взять любой вектор a, разумно выбрать какой-нибудь из тех, что получены с помощью разделения центров масс.

3. Шаг модификации может быть выбран в виде $\tilde{a} := \tilde{a} + h_p z^i$, где p — порядковый номер модификации, а $\{h_p\}$ — произвольная убывающая последовательность положительных чисел с расходящейся суммой $\sum_p h_p$, например $h_p = \frac{1}{p}$.

Обучение по примерам, по страницам, по всему задачнику. В традиционном алгоритме обучения персептрона модификация вектора весов $\tilde{a}$ проводится сразу после обнаружения очередной ошибки. Можно поступить по-другому: найдем в обучающей выборке *все* те z^i, для которых не выполняется требуемое неравенство $(z^i, \tilde{a}) > 0$. Обозначим это множество Err. После проверки *всей* обучающей выборки положим

$$\tilde{a} := \tilde{a} + \sum_{z^i \in \text{Err}} z^i. \tag{1.38}$$

Естественно, не требуется хранить все множество Err — достаточно накапливать сумму тех z^i, на которых персептрон ошибается.

Назовем первый способ (с коррекцией $\tilde{a}$ после каждого обнаружения ошибки) методом обучения по примерам, а второй (с коррекцией $\tilde{a}$ только после прохождения всей обучающей выборки) — методом обучения по всему задачнику.

Как показывают тестовые испытания, обучение по всему задачнику, как правило, сходится быстрее, чем обучение по отдельным примерам, однако за это улучшение приходится платить дополнительным расходом памяти, так как отдельно от $\tilde{a}$ накапливается вектор поправки

$$\Delta\tilde{a} = \sum_{z^i \in \text{Err}} z^i. \tag{1.39}$$

Существует промежуточный вариант обучения — по страницам задачника: обучающее множество разбивается на подмножества (*страницы*) и задается последовательность прохождения страниц, столько-то циклов по первой странице, потом столько-то по второй и т.д. Коррекция $\tilde{a}$ проводится после прохождения страницы. Задачник разбивается на страницы по различным эвристическим правилам, часто исходящим из психолого-педагогических аналогий, например по правилу "*от простого к сложному*".

При аппаратной реализации нейронных сетей дополнительные затраты "быстрой" памяти на накопление Δa могут быть довольно большими, поэтому разработчики стремятся к тому, чтобы алгоритмы обучения работали с отдельными примерами.

Кроме того, обучение по всему задачнику предполагает, что он хранится и может быть вызван снова. Это требует больших объемов памяти. Возможна такая ситуация, когда извне поступает поток задач,

тогда по каждой возникшей ошибке проводится коррекция, после чего система ждет новой задачи, а старые не сохраняются. Кроме того, существует промежуточный вариант — хранится и время от времени обновляется сравнительно небольшая часть поступивших задач — страница, по которой система проходит несколько циклов обучения. Хорошим правилом формирования такой страницы является принцип *"стандартные образцы + k последних задач"*.

При реализации на РС, как показывает наш опыт, чаще всего наилучшим является обучение по всему задачнику, иногда (при большом задачнике) — обучение по страницам, размеры которых определяются объемом доступной оперативной памяти. Мы никогда не сталкивались с ситуациями, в которых обучение на отдельных примерах давало бы на РС хоть какое-нибудь преимущество. Это, однако, опыт, а не теорема.

Метод максимума правдоподобия. Продолжаем изучать задачу разделения двух классов. Предположим, что с каждым классом связано вероятностное распределение в пространстве векторов x (векторов значений признаков). Будем обозначать плотности этих распределений $p(x|C_i)$ ($i = 1,2$; C_i — событие, состоящее в том, что объект принадлежит к i-му классу). Нас интересует апостериорная вероятность: $P(C_i|x)$ — вероятность принадлежности объекта к i-му классу при условии, что он характеризуется вектором признаков x. Известная из элементарной теории вероятностей формула Байеса дает

$$P(C_i|x) = \frac{P(C_i)p(x|C_i)}{\sum_j P(C_j)p(x|C_j)}. \tag{1.40}$$

Важно заметить, что в эту формулу входят $P(C_i)$ — вероятности появления объектов i-го класса.

Выпишем эту формулу для многомерных нормальных распределений:

$$p(x|C_i) = \frac{1}{(2\pi)^{k/2}(\det\Sigma^i)^{1/2}\exp\left[-\frac{1}{2}(x - \mathrm{M}^i, (\Sigma^i)^{-1}(x - \mathrm{M}^i))\right]}, \tag{1.41}$$

где $\mathrm{M}^i = \int x p(x|C_i)\, d^k x$ — математическое ожидание x в i-м классе; ковариационная матрица для i-го класса

$$(\Sigma^i)_{pq} = \int (x_p - \mathrm{M}^i_p)\,(x_q - \mathrm{M}^i_q) p(x|C_i)\, d^k x. \tag{1.42}$$

В результате обработки данных находят статистические оценки M^i и Σ^i: пусть для i-го класса имеются векторы $x^1,\ldots,x^r$. Тогда полагаем

$$\mathbf{M}^i = \frac{1}{r}\sum_{j=1}^{r} x^j;$$

$$(\Sigma^i)_{pq} = \frac{1}{r}\sum_{j=1}^{r}(x^j_p - \mathbf{M}^i_p)(x^j_q - \mathbf{M}^i_q). \qquad (1.43)$$

Для многомерных нормальных распределений формула Байеса имеет вид

$$P(C_i|x) = \frac{P(C)(\det\Sigma^i)^{-1/2}\exp\left[-\frac{1}{2}(x - \mathbf{M}^i, (\Sigma^i)^{-1}(x - \mathbf{M}^i))\right]}{\sum_j P(C_j)(\det\Sigma^j)^{-1/2}\exp\left[-\frac{1}{2}(x - \mathbf{M}^j, (\Sigma^j)^{-1}(x - \mathbf{M}^j))\right]}. \qquad (1.44)$$

Минимизация в формуле Байеса вероятности ошибки дает очень простое *решающее правило*: x принадлежит i-му классу, если $P(C_i|x) > P(C_j|x)$ при всех $j \neq i$, т.е. выбирается тот класс, для которого $P(C_i|x)$ максимально. Поскольку в формуле Байеса для всех C_i знаменатель общий, решающее правило выглядит так: выбираем то i, для которого $P(C_i)\rho(x|C_i)$ максимально. Для нормальных распределений удобно еще и прологарифмировать эту величину. Окончательно получаем: x принадлежит i-му классу, если среди величин

$$P_j = \ln P(C_j) - \frac{1}{2}\ln|\det\Sigma^j| - \frac{1}{2}(x - \mathbf{M}^j, (\Sigma^j)^{-1}(x - \mathbf{M}^j)) \qquad (1.45)$$

P_i максимальная.

Итак, разделяющей является поверхность второго порядка — квадрика, а операцию разделения на два класса выполняет квадратичный адаптивный сумматор в комбинации с пороговым нелинейным элементом. Квадратичный сумматор вычисляет выражение

$$P_1 - P_2 = \ln(P(C_1)/P(C_2)) - \frac{1}{2}\ln|(\det\Sigma^1)/(\det\Sigma^2)| -$$

$$-\frac{1}{2}(x, [(\Sigma^1)^{-1} - (\Sigma^2)^{-1}]x) + (x, (\Sigma^1)^{-1}\mathbf{M}^1 - (\Sigma^2)^{-1}\mathbf{M}^2) -$$

$$-\frac{1}{2}(\mathbf{M}^1, (\Sigma^1)^{-1}\mathbf{M}^1) + \frac{1}{2}(\mathbf{M}^2, (\Sigma^2)^{-1}\mathbf{M}^2). \qquad (1.46)$$

Пороговый элемент вычисляет

$$h(P_1 - P_2) = \begin{cases} 1, & \text{если } P_1 - P_2 > 0; \\ 0, & \text{если } P_1 - P_2 \leqslant 0. \end{cases} \qquad (1.47)$$

В результате для первого класса получаем ответ 1, для второго — 0.

Существенным элементом описанного метода является оценивание по экспериментальным данным матриц $(\Sigma^j)^{-1}$. Здесь могут встретиться затруднения. Во-первых, некоторые собственные числа матрицы Σ^i могут быть весьма малы. Во-вторых, достоверное оценивание $(\Sigma^i)^{-1}$ может потребовать слишком большого числа данных. Наконец, само вычисление обратной матрицы большой размерности — достаточно трудоемкая операция. Поэтому вместо $(\Sigma^i)^{-1}$ часто используются ее различные аппроксимации. Простейшая из них — замена Σ^i на $\delta_i^2 \times 1$. Величина δ_i^2 оценивается как средний квадрат расстояния от x^j до M^i для примеров i-го класса:

$$\delta_i^2 = \frac{1}{r} \sum_{j=1}^{r} (x^j - \mathrm{M}^i)^2. \tag{1.48}$$

Если $\delta_1 \neq \delta_2$, то при такой оценке разделяющая два класса квадрика становится сферой радиуса

$$R = \left| \frac{1}{\delta_1^2} - \frac{1}{\delta_2^2} \right|^{1/2}. \tag{1.49}$$

Для $\delta_1 = \delta_2$ она превращается в гиперплоскость (разделение центров масс).

Пусть установлены различные цены ошибок c_{12} (цена отношения объекта первого класса ко второму) и c_{21} (цена отнесения объекта второго класса к первому) и минимизируется средний штраф

$$c_{12}P_{12} + c_{21}P_{21}, \tag{1.50}$$

где P_{ij} — вероятность отнесения объекта i-го класса к j-му. Тогда разделяющая поверхность задается уравнением

$$c_{12}P(C_1|x) = c_{21}P(C_2|x). \tag{1.51}$$

С использованием формулы Байеса это уравнение приводится к виду

$$c_{12}P(C_1)\rho(x|C_1) = c_{21}P(C_2)\rho(x|C_2). \tag{1.52}$$

Для квадратичных распознающих устройств неравнозначность ошибок приводит к сдвигу порога на величину $\ln(c_{12}/c_{21})$.

Расстояние Махаланобиса. С ковариационной матрицей Σ связано скалярное произведение

$$(x, y)_{\mathrm{M}} = (x, \Sigma^{-1}y) \tag{1.53}$$

и соответствующая норма

$$\|x\|_{\mathrm{M}} = (x, \Sigma^{-1}x)^{1/2}. \tag{1.54}$$

Расстоянием Махаланобиса между векторами x, y называется $\|x - y\|_M$. В этом расстоянии выборка максимально приближена к "шаровому скоплению" единичного радиуса [81]. В задачах классификации можно вводить несколько расстояний Махаланобиса, например для каждого класса свое. Часто пользуются ковариационной матрицей объединенной выборки, включающей все классы.

Объединенная ковариационная матрица Σ для q классов имеет вид

$$\Sigma = \sum_{i=1}^{q} \frac{n_i}{n} \Sigma^i, \tag{1.55}$$

где Σ^i — ковариационная матрица i-го класса; n_i — число примеров i-го класса; $n = \sum_{i=1}^{q} n_i$.

Расстояние Махаланобиса можно трактовать как "*истинное*" расстояние между векторами, учитывающее характер статистического разброса.

Квадрупольное разделение классов. Решающее правило для задачи классификации совершенно не обязательно должно использовать только линейные функции или квадратичные формы. Существует огромный произвол в выборе разделяющих функций, ограниченный лишь фантазией и здравым смыслом исследователей [2], а также (в меньшей степени) техническими возможностями. Обширна и литература, посвященная этому вопросу. Мы ограничимся только одним примером.

Пусть дана обучающая выборка: x^i $(i = 1,...,n)$ принадлежит первому классу, y^j $(j = 1,...,m)$ — второму. Положим

$$F(x) = \sum_{i=1}^{n} \frac{q_1}{\|x - x^i\|^d} - \sum_{j=1}^{m} \frac{q_2}{\|x - y^j\|^d}, \tag{1.56}$$

где $q_{1,2} > 0$ — "заряды" классов; $\|x - y\| = \left(\sum_i (x_i - y_i)^2\right)^{1/2}$ — обычное эвклидово расстояние.

Показатель степени d может принимать любое положительное значение. Существует эвристическое правило, согласно которому выбирается $d = N - 2$, где N — размерность вектора данных x (число признаков). Основанием для такого выбора служит то, что функция $\varphi(x) = \|x\|^{2-N}$ есть решение уравнения Лапласа с точечным источником: $\Delta\varphi = \delta(x)$ (здесь Δ — оператор Лапласа, $\delta(x)$ — δ-функция). Используется аналогия с теорией поля в N-мерном пространстве. Естественно, этот выбор d не является единственно возможным.

Единственное предположение об обучающей выборке: для всех i, j $x^i \neq y^j$. В этом случае в некоторой окрестности любого x^i выпол-

нено неравенство $F(x) > 0$, а в некоторой окрестности любого y^i справедливо $F(x) < 0$. В качестве решающего правила можно было бы взять следующее: x принадлежит первому классу, если $F(x) > 0$, и второму, если $F(x) < 0$. Это, однако, не очень удобно, так как в выражение для $F(x)$ в явном виде входит вся обучающая выборка. Можно воспользоваться каким-нибудь разложением. Например, по системе ортогональных полиномов или каких-либо собственных функций оператора Лапласа. Мы же воспользуемся простым разложением Тейлора.

Обозначим центры масс

$$x^0 = \frac{1}{n}\sum_{i=1}^{n} x^i, \quad \Delta x^i = x^i - x^0;$$
$$y^0 = \frac{1}{m}\sum_{j=1}^{m} y^j, \quad \Delta y^j = y^j - y^0. \tag{1.57}$$

Заменим в выражении для F величины $\frac{1}{\|x - x^i\|^d}$, $\frac{1}{\|x - y^j\|^d}$ членами до второй степени разложения Тейлора по x^i, y^j вблизи точек x^0, y^0. Из-за выбора x^0, y^0 сумма членов первой степени обратится в нуль и мы получим:

$$F(x) \cong F_0(x) + F_2(x);$$

$$F_0(x) = \frac{nq_1}{\|x - x^0\|^d} - \frac{mq_2}{\|x - y^0\|^d}; \tag{1.58}$$

$$2F_2(x) = \frac{dq_1}{\|x - x^0\|^{d+4}}\sum_{i=1}^{n}[(d+2)(x - x^0, \Delta x^i)^2 - \|x -$$

$$- x^0\|^2 \|\Delta x^i\|^2] - \frac{dq_2}{\|x - y^0\|^{d+4}}\sum_{j=1}^{m}[(d+2)(x - y^0, \Delta y^j)^2 -$$

$$- \|x - y^0\|^2 \|\Delta y^j\|^2].$$

Выражения в квадратных скобках — квадратичные формы от $x - x^0$, $x - y^0$. Их коэффициенты суть квадратичные формы от Δx^i, Δy^j. Независимо от числа слагаемых (элементов обучающей выборки) каждая квадратичная форма определяется $N(N-1)/2$ коэффициентами (N — размерность вектора данных).

Полученная функция $F_0(x) + F_2(x)$ может эффективно использоваться для разделения классов. "Заряды" q_1, q_2 настраиваются исходя из числа примеров каждого класса и оценки риска ошибок. Чем выше штраф за неправильное определение объекта i-го класса (т.е. за ошибочное отнесение его к j-му классу), тем больше q_1. Чем меньше в

обучающей выборке примеров, относящихся к i-му классу, тем больше q_i (например, $q_1 \div 1/n$, $q_2 \div 1/m$).

Простейшее (и важнейшее) обобщение состоит в использовании для каждого класса вместо эвклидового расстояния своей метрики Махаланобиса.

$$F(x) = \sum_{i=1}^{n} \frac{q_1}{\| x - x^i \|_{M1}^{d}} - \sum_{j=1}^{m} \frac{q_2}{\| x - y^j \|_{M2}^{d}};$$

$$F(x) \cong F_0(x) + F_2(x);$$

$$F_0(x) = \frac{nq_1}{\| x - x^0 \|_{M1}^{d}} - \frac{mq_2}{\| x - y^0 \|_{M2}^{d}}; \tag{1.59}$$

$$2F_2(x) = \frac{dq_1}{\| x - x^0 \|_{M1}^{d+4}} \sum_{i=1}^{n} [(d+2)(x - x^0, \Delta x^i)_{M1}^2 - \| x -$$

$$- x^0 \|_{M1}^2 \| \Delta x^i \|_{M1}^2] - \frac{dq_2}{\| x - y^0 \|_{M2}^{d+4}} \sum_{j=1}^{m} [(d+2)(x - y^0, \Delta y^j)_{M2}^2 -$$

$$- \| x - y^0 \|_{M2}^2 \| \Delta y^j \|_{M2}^2].$$

Можно также воспользоваться объединенной метрикой Махаланобиса двух классов.

Построенные разделяющие функции по аналогии с электростатикой можно назвать *квадрупольными*. Разделение классов с помощью таких функций является обобщением метода максимума правдоподобия и позволяет строить невыпуклые разделяющие поверхности. Использование квадрупольных разделяющих функций привлекательно еще и потому, что из массивных вычислений, объем которых растет с увеличением размерности вектора данных, требуется только найти значения нескольких (не более четырех) квадратичных форм. Настройка параметров требует накопления коэффициентов квадратичных форм типа

$$\sum_{i=1}^{n} [(d+2)(\cdot, \Delta x^i)^2 - \| \cdot \|^2 \| \Delta x^i \|^2] \tag{1.60}$$

(точкой обозначен переменный вектор, от которого будет вычисляться форма), а также определения положений центров масс x^0, y^0. Все это можно сделать последовательно за два прохода обучающей выборки.

Наиболее трудоемкая операция возникает при использовании метрики Махаланобиса и состоит в вычислении матрицы Σ^{-1}. Однако и эта задача не сложнее, чем метод максимального правдоподобия, — ее решение есть часть этого метода.

Рекуррентная оценка метрики Махаланобиса. Расстояние Махаланобиса входит в большинство статистически наилучших правил разделения классов, однако вычисление матрицы Σ^{-1} может быть весьма сложной и плохо обусловленной операцией. Поэтому для вычисления метрики Махаланобиса может оказаться полезной следующая приближенная процедура. Опишем ее для больших выборок: предполагается, что число примеров n много больше размерности пространства данных N.

Напомним, что обучающая выборка — $\{x^p\}$ $(p = 1,...,n)$, $\Delta x^p = x^p - \mathrm{M}(x)$,

$$\delta^2 = \frac{1}{n}\sum_p (\Delta x^p, \Delta x^p), \quad \Sigma = \frac{1}{n}\sum_p \Delta x^p \otimes \Delta x^{p\mathrm{T}}, \tag{1.61}$$

где верхний индекс T означает транспонирование, а $\otimes$ — произведение вектора-столбца на вектор-строку (тензорное произведение).

В качестве начального приближения примем

$$S_0 = \frac{1}{\delta^2} \times 1. \tag{1.62}$$

Далее на каждом (p-м) шаге будем отнимать от S_{p-1}^{-1} матрицу $1 \times \frac{1}{n}(\Delta x^p, \Delta x^p)$ (вклад Δx^p в $1 \times \delta^2$), прибавлять матрицу $\frac{1}{n}\Delta x^p \otimes \Delta x^{p\mathrm{T}}$ (вклад Δx^p в Σ) и вычислять обратную матрицу в первом приближении по $1/n$. В результате получим

$$S_p = S_{p-1}\left(1 + \frac{1}{n}(\Delta x^p, \Delta x^p)\right) - \frac{1}{n}((S_{p-1}\Delta x^p) \top (S_{p-1}\Delta x^p)^{\mathrm{T}}). \tag{1.63}$$

Приближенным значением для Σ^{-1} является S_n. Вычисление S_n не сложнее, чем накопление Σ по исходной выборке. Использованный нами прием аналогичен построению одношагового квазиньютоновского метода (BFGS-формула — [45]; см. также гл. 3) в теории оптимизации.

Матрица S_n должна быть симметрична и положительно определена. Симметричность S_n очевидна по построению, а положительная определенность при достаточно больших n и невырожденности Σ связана с тем, что на каждом шаге S_p получается из S_{p-1} умножением на "почти единичную" матрицу:

$$S_p = S_{p-1}\left(1 + \frac{1}{n}(\Delta x^p, \Delta x^p) - \frac{1}{n}((\Delta x^p) \otimes (S_{p-1}\Delta x^p)^{\mathrm{T}})\right).$$

Поэтому определитель S_p равен произведению положительных чисел и сам больше нуля.

Если неверно, что $n \gg N$, то можно добиться положительного результата, формально удваивая (утраивая и т.п.) выборку — так, чтобы выражение в скобках (множитель при S_{p-1}) в последней формуле было достаточно близко к единичной матрице.

4. Значимость параметров и сигналов. Сокращение описания

Сокращение множества параметров и входных сигналов может преследовать несколько целей:

1) упрощение специализированных технических устройств;

2) сокращение объема используемой памяти и увеличение быстродействия программ;

3) удешевление сбора данных за счет сокращения их объема;

4) обеспечение (или облегчение) явной вербальной интерпретации процесса и результатов обработки данных.

С точки зрения пользователя особенно важны две последние цели: сокращение объема данных уменьшает затраты на самый трудоемкий процесс — сбор данных, возможность явной вербальной интерпретации — новое потребительское свойство системы, в результате программное обеспечение предоставляет новые возможности, с которыми пользователь может обращаться по своему усмотрению без дополнительных затрат.

Существует, как минимум, два пути сокращения описания: *снизу вверх* — постепенное удаление параметров от наименее значимых к более значимым и *сверху вниз* — выделение наиболее значимых параметров и постепенное дополнение их менее значимыми, пока не станет достаточно.

Последовательность действий при сокращении описания "снизу вверх" очень проста:

1) определяются наименее значимые параметры;

2) они устраняются вместе с соответствующими элементами системы;

3) оставшиеся параметры системы модифицируются так, чтобы наилучшим образом решать задачу.

Далее вновь определяются наименее значимые параметры и т.д., пока задача решается с удовлетворительной точностью. Проверка точности проводится после выполнения п. 3.

Если сокращенная система не может решить задачу с требуемой точностью (т.е. упрощение было слишком радикальным), то можно поступить двумя способами. Первый состоит в том, что просто возвращаемся на цикл назад и проводим менее радикальное сокращение, например отбрасываем вдвое меньше параметров. Второй способ более тонкий. Заметим, что система модифицировалась (часть параметров обратилась в нуль, а оставшиеся были изменены для лучшего решения задачи). В соответствии с этим изменилась и относительная значимость

отброшенных параметров. Определим эту новую относительную значимость и вернем в систему элементы, соответствующие наиболее значимым параметрам (из ранее отброшенных).

Определение значимости параметров на основании функции оценки опишем сначала в наиболее общем виде. Итак, имеется набор N-мерных векторов данных x^i $(i = 1,...,n)$ с координатами x_j^i $(j = 1,...,N)$, M — мерный вектор параметров a с координатами a_k $(k = 1,...,M)$ и некоторая функция (оценка) $H(x, a)$, оценивающая работу системы с параметрами a на векторе данных x. Оценка H может быть расстоянием от вектора выходных сигналов системы до нужного ответа или до множества правильно интерпретируемых ответов и т.д. Требуется выделить наименее значимые параметры a_k и компоненты данных x_j и модифицировать систему, отбрасывая эти параметры и компоненты данных.

Интересно, что сама процедура отбрасывания вовсе не однозначна. Простейший вариант — обращение в нуль — не всегда наилучший. Он не учитывает, например, корреляции между данными. Учет корреляций приводит к тому, что отбрасываемые компоненты данных лучше всего заменять некоторыми функциями остающихся компонент.

Пусть для каждого a_k определено некоторое значение a_k^0 и сокращение a_k означает приравнивание $a_k := a_k^0$. Аналогично, пусть для каждой компоненты данных x_j и любого примера x^i определено x_j^{i0} и отбрасывание j-й компоненты данных для i-го примера означает приравнивание $x_i = x_j^{i0}$.

Такая постановка задачи допускает очень большую свободу в доопределении процедуры сокращения описания. В качестве простейшего базового варианта будем иметь в виду $a_k^0 = 0$ и для всех i $x_j^{i0} = \frac{1}{n}\sum_{p=1}^{n} x_j^p$ (параметры обращаются в нуль, данные заменяются средними значениями по выборке). Более тонкие методы предполагают замену отбрасываемых параметров и сигналов на некоторые функции оставшихся.

Показатели значимости вычисляются в два этапа: сначала они оцениваются для одного вектора данных (примера), потом — по всей выборке. Для данного x^p значимости a_k и x_j оцениваются как

$$\chi(a_k|x^p) = \left|\frac{\partial H(x, a)}{\partial a_k}\right|_{x=x^p} \times |a_k - a_k^0|;$$
$$\chi(x_j|x^p) = \left|\frac{\partial H(x, a)}{\partial x_j}\right|_{x=x^p} \times |x_j^p - x_j^{p0}|, \qquad (1.64)$$

где χ суть просто вычисленные в линейном приближении абсолютные величины изменения H при сокращении описания.

Оценка по всей выборке x^i может проводиться по-разному. В нашем программном обеспечении используются две нормы:

1) сумма модулей —

$$\chi(a_k) = \sum_p \chi(a_k|x^p);\ \chi(x_j) = \sum_p \chi(x_j|x^p); \tag{1.65}$$

2) максимум модуля —

$$\chi(a_k) = \max_p \chi(a_k|x^p);\ \chi(x_j) = \max_p \chi(x_j|x^p). \tag{1.66}$$

Как всегда, пользователю должна предоставляться возможность оценить важность отдельного примера (или класса примеров). Соответствующая модификация оценок значимости имеет вид:

1) взвешенная сумма модулей —

$$\chi(a_k) = \sum_p w_p\chi(a_k|x^p);\ \chi(x_j) = \sum_p w_p\chi(x_j|x^p); \tag{1.65'}$$

2) максимум модуля взвешенных показателей —

$$\chi(a_k) = \max_p w_p\chi(a_k|x^p);\ \chi(x_j) = \max_p w_p\chi(x_j|x^p). \tag{1.66'}$$

Веса w_p определяются пользователем с учетом риска при неправильной классификации, числа примеров данного класса в выборке и др.

Часто приходится иметь дело с системой, которая меняет свои параметры, например, в ходе обучения. Тогда к моменту принятия решения о значимости может быть накоплена информация о частных производных H в разных точках $a = a^1,...,a^q$. Ее можно использовать следующим образом. Обозначим угловыми скобками процедуру усреднения по множеству параметров $a^1,...,a^q$:

$$\langle f(a,...)\rangle = \frac{1}{q}\sum_{S=1}^{q} f(a^S,...). \tag{1.67}$$

Положим

$$\begin{aligned}
\chi(a_k|x^p) &= \left\langle \left| \frac{\partial H(x,a)}{\partial a_k}\right|_{x=x^p} \right| > x|a_k - a_k^0|; \\
\chi(x_j|x^p) &= \left\langle \left| \frac{\partial H(x,a)}{\partial x_j}\right|_{x=x^p} \right| > x|x_j^p - x_j^{p0}|.
\end{aligned} \tag{1.68}$$

Усредняются абсолютные значения производных, а приращения берутся в тех точках, в которых будет проводиться процедура сокращения описания. Использование усреднения по нескольким значениям векто-

ра параметров a важно для нелинейных систем, в которых производные H могут сильно меняться от точки к точке.

Так же как и при переходе от $\chi(\alpha_k|x^p)$ к $\chi(\alpha_k)$, можно брать не усреднение модулей производных, а их максимумы. Такие показатели χ оценивают не среднюю, а максимальную возможную значимость параметров и данных. В программном обеспечении мы обычно реализуем обе возможности, предоставляя пользователю право выбора. По умолчанию принимается усреднение, а при желании можно перейти к максимуму модуля.

Сокращение описания на основании функции оценки H предполагает, что главная задача — сохранить качество работы системы, оцениваемое с помощью H. Для этого требуется знать назначение системы и иметь способ оценки ее соответствия своему назначению. Возможен, однако, и другой подход, не предполагающий никакого знания о способах оценки. Ставится задача *сократить описание, минимально изменяя функционирование системы.* В этом случае роль оценки играет изменение выходного сигнала системы после сокращения.

Значимость параметров по изменению выходных сигналов системы определяется практически так же, как и с помощью функции оценки. Пусть вектор выходных сигналов системы есть $F(x, \alpha)$, где x — вектор данных, α — вектор параметров. Так же как и ранее, пусть задан набор векторов данных x^i, на которых будет оцениваться функционирование системы, и определены значения x_j^{i0}, α_k^0. Вычислим в линейном приближении изменение вектора F при обращении α_k в α_k^0 и x_j^p в x_j^{p0}:

$$\begin{aligned} \Delta F &= \frac{\partial F(x, \alpha)}{\partial \alpha_k}\Big|_{x=x^p} \times (\alpha_k^0 - \alpha_k), \\ \Delta F &= \frac{\partial F(x, \alpha)}{\partial x_j}\Big|_{x=x^p} \times (x_j^{p0} - x_j^p). \end{aligned} \tag{1.69}$$

Пусть в пространстве выходных сигналов системы задана некоторая норма $\|\cdot\|$ (например, обычная эвклидова). Тогда положим

$$\begin{aligned} \chi(\alpha_k|x^p) &= \Big\|\frac{\partial F(x, \alpha)}{\partial \alpha_k}\cdot\Big|_{x=x^p}\Big\| \times |\alpha_k - \alpha_k^0, \\ \chi(x_j|x^p) &= \Big\|\frac{\partial F(x, \alpha)}{\partial x_j}\Big|_{x=x^p}\Big\| \times |x_j^p - x_j^{p0}|. \end{aligned} \tag{1.70}$$

Таким образом, для каждого α_k и любого x_j определен вектор показателей значимости. Координаты вектора соответствуют точкам x^p. Теперь нужно вычислить какую-либо норму этого вектора и объявить ее показателем значимости. Как и выше, можно в качестве нормы взять максимум модуля или сумму модулей. При использовании эвклидовой

нормы в пространстве выходных сигналов бывает удобно и далее выбирать такую же норму, полагая

$$\chi(\alpha_k) = \left(\sum_p \chi(\alpha_k|x^p)^2\right)^{1/2},$$
$$\chi(x_j) = \left(\sum_p \chi(x_j|x^p)^2\right)^{1/2}. \tag{1.71}$$

Подход к определению значимости через изменение выходного сигнала не имеет альтернатив в том случае, когда рассматриваемая система лишь подсистема в некоторой сети (например, сумматор или нейрон в нейронной сети). Тогда при изменении элемента в наиболее общем случае приходится ограничиваться требованием: выходной сигнал этого элемента должен измениться как можно меньше, чтобы не нарушилось функционирование целого.

Сокращение числа входов в адаптивном линейном сумматоре (путь "снизу вверх"). Рассмотрим адаптивный линейный сумматор, вычисляющий линейную неоднородную функцию $F(x, \alpha) = \alpha_0 + (x, \alpha)$. Будем решать задачу о сокращении числа входных сигналов. Начнем с определения значимости по изменению выходного сигнала. Заметим, что

$$\frac{\partial F}{\partial \alpha_0} \equiv 1, \quad \frac{\partial F}{\partial \alpha_i}\Big|_{x=x^p} = x_1^p, \quad \frac{\partial F}{\partial x_j}\Big|_{x=x^p} = \alpha_i \ (i = 1, ..., N). \tag{1.72}$$

Уничтожить i-й входной сигнал можно двумя способами: заменой параметра α_i на 0 или заменой x_i на постоянную величину (не зависящую от p). В последнем случае получаем новую функцию

$$F_{1i}(x, \alpha) = \alpha_0 + x_i^0\alpha_i + \sum_{\substack{j=1 \\ j\neq i}}^{N} x_j\alpha_j. \tag{1.73}$$

Такое преобразование означает, что одновременно с уничтожением i-й входной связи α_0 приобретает новое значение: $\alpha_0 := \alpha_0 + x_i^0\alpha_i$. Естественно, при этом можно добиться меньшего изменения $F(x, \alpha)$, чем просто при приравнивании α_i нулю. Остановимся поэтому на замене i-го входного сигнала на постоянную величину x_i^0. Значение этой постоянной определим исходя из минимизации изменения $F(x^p, \alpha)$. Оценим его в эвклидовой норме:

$$\chi(x_i) = \left[\sum_p \left(F(x^p, \alpha) - F_{1i}(x^p, \alpha)\right)^2\right]^{1/2} =$$
$$= \left[\sum_p (x_i^p\alpha_i - x_i^0\alpha_i)^2\right]^{1/2} = |\alpha_i|\left[\sum_p (x_i^p - x_i^0)^2\right]^{1/2}. \tag{1.74}$$

Минимизация этой величины дает

$$x_i^0 = \frac{1}{n}\sum_{p=1}^{n} x_i^p. \qquad (1.75)$$

Таким образом, оптимальной является замена x_i на среднее значение x_i по исходной выборке. В обозначениях теории вероятностей

$$x_i^0 = M(x_i),\ \chi(x_i) = n^{1/2}\alpha_i\sigma(x_i), \qquad (1.76)$$

где $\sigma(x_i)$ — среднеквадратичное уклонение x_i на выборке $\{x^p\}$, а множитель $n^{1/2}$ (n — объем выборки) — общий для всех x_i, поэтому здесь он не играет роли.

Если оценивать значимость по изменению выходного сигнала, то при исключении i-го входного сигнала следует заменить его средним значением. Значимость такого изменения оценивается как $\chi_i = |\alpha_i|\sigma(x_i)$. При исключении сигналов по одному они сортируются в соответствии со значениями χ_i и отбрасываются (заменяются средними) сначала те, что соответствуют меньшим χ_i. Еще раз заметим, что этот путь "снизу вверх" универсален, но не оптимален. В частности, для сумматоров и других элементов, линейных по параметрам (например, квадратичных сумматоров), существует учитывающий все корреляции путь исключения "сверху вниз" с ортогонализацией (см. далее).

Определим теперь для сумматора показатель значимости параметров в соответствии с функцией оценки. Пусть для каждого вектора данных из выборки $\{x^p\}$ известно требуемое значение выходного сигнала f_p. Оценку будем определять согласно методу наименьших квадратов: $H_p = (F(x^p, \alpha) - f_p)^2$. Тогда

$$\frac{\partial H_p}{\partial x_i}\Big|_{x=x^p} = \Delta_p\alpha_i, \qquad (1.77)$$

где $\Delta_p = F(x^p, \alpha) - f_p$ — ошибка на p-м примере (x^p).

Если сокращение проводить заменой x_i на $\mathrm{M}(x_i)$ (точнее, на среднее по выборке), то в качестве показателей значимости получим

$$\chi(x_i|x^p) = |\Delta_p| \times |\alpha_i| \times |x_i^p - \mathrm{M}(x_i)|, \qquad (1.78)$$

если для получения $\chi(x_i)$ воспользоваться эвклидовой нормой, то

$$\chi(x_i) = |\alpha_i| \times \left[\sum_p \Delta_p^2 (x_i^p - \mathrm{M}(x_i))^2\right]^{1/2}. \qquad (1.79)$$

Мы получили почти то же самое, что и без функции оценки, только вместо среднеквадратичного уклонения по выборке $\sigma(x_i)$ стоит среднеквадратичное уклонение "с весами" Δ_p^2 (чем выше ошибка в точке x^p, тем больше вклад этой точки в оценку значимости $\chi(x_i)$).

При дальнейшем рассмотрении примеров различных элементов в этой главе, избегая лишнего дублирования, ограничимся оценкой значимости по изменению выходного сигнала. Переход к использованию функции оценки, как правило, очевиден.

Показатели значимости для нейрона с дифференцируемым нелинейным элементом φ ищутся почти так же, как и для сумматора. Пусть $F(x, \alpha) = \varphi(\alpha_0 + (x, \alpha))$. Тогда

$$\frac{\partial F}{\partial s_i}\Big|_{x=x^p} = \varphi'(\alpha_0 + (x^p, \alpha)) \times \alpha_i. \tag{1.80}$$

В эвклидовой норме (что соответствует методу наименьших квадратов) получаем

$$\chi(x_i) = |\alpha_i| \left[\sum_p (\varphi'(\alpha_0 + (x^p, \alpha)))^2 (x_i^p - \mathrm{M}(x_i))^2\right]^{1/2}. \tag{1.81}$$

Вновь имеем произведение модуля параметра α_i на среднеквадратичное уклонение "с весами". Роль веса играет квадрат производной функции φ в точке $\alpha_0 + (x^p, \alpha)$.

Показатели значимости для нейрона с пороговым нелинейным элементом (персептрона) требуют для своего вычисления еще одного эвристического хода, так как прямо воспользоваться предыдущими формулами невозможно.

Пусть функция, вычисляемая нейроном, имеет вид $F(x, \alpha) = h(\alpha_0 + (x, \alpha))$, где h — функция Хевисайда ($h(x) = 1$ при $x > 0$, $h(x) = 0$ при $x \leqslant 0$). Для каждого вектора данных x^p необходимо оценить значимость изменения аргумента h при замене x_i^p на x_i^0. Значение $h(a)$ меняется только тогда, когда a переходит через нуль, поэтому естественно оценивать значимость изменения Δa переменной a в масштабе, определяемом текущим значением переменной, т.е. значимость Δa оценивается как $|\Delta a/a|$. Из этого эвристического рассуждения получаем

$$\chi(x_i|x^p) = \frac{|\alpha_i| \times |x_i^p - x_i^0|}{|\alpha_0 + (x^p, \alpha)|}. \tag{1.82}$$

Если положить $x_i^0 = \mathrm{M}(x_i)$ и воспользоваться эвклидовой метрикой, то вновь получим произведение модуля параметра α_i на среднеквадратичное уклонение с весом. В качестве весов выступают квадраты величин, обратных выходным сигналам сумматоров:

$$\chi(x_i) = |\alpha_i| \times \left[\sum_p \frac{(x_i^p - \mathrm{M}(x_i))^2}{p(\alpha_0 + (x^p, \alpha))^2}\right]^{1/2}. \tag{1.83}$$

Полученные выражения для показателей значимости позволяют упрощать основные элементы нейронных сетей "снизу вверх", начиная с исключения самых малозначимых параметров.

5. Сокращение описания "сверху вниз" — подбор достаточного семейства наиболее значимых параметров

Метод исключения параметров "сверху вниз" с ортогонализацией является более мощным, требует несколько большего объема вычислений, но применим не ко всяким функциям $F(x, \alpha)$, а только к таким, которые имеют вид

$$F(x, \alpha) = \varphi\left(\sum_i \alpha_i f_i(x)\right). \tag{1.84}$$

Важное достоинство метода — автоматический учет корреляций между $f_i(x)$.

Начнем с устройств, вычисляющих функции вида

$$F(x, \alpha) = \sum_i \alpha_i f_i(x). \tag{1.85}$$

К ним относятся линейные адаптивные сумматоры ($f_0(x) \equiv 1$, $f_i(x) = x_i$ при $i > 0$), квадратичные сумматоры и др.

Пусть, как и прежде, заданы векторы данных x^p ($p = 1, ..., n$). Координаты этих векторов обозначаем x_i^p ($i = 1, ..., N$). Поставим задачу сокращения описания следующим образом: так определить некоторое множество индексов J (по возможности наименьшее) и набор чисел β_0 ($j \in J$), чтобы норма отклонения

$$\| \Delta F \| = \| F - F' \|, \tag{1.86}$$

где $F' = \sum_{j \in J} \beta_j f_j(x)$, не превышала некоторой наперед заданной величины ε. Все функции рассматриваются на конечном множестве $\{x^p\}$, и $\| \cdot \|$ есть некоторая норма в пространстве функций $\{x^p\}$. Проще всего иметь дело с эвклидовой нормой: для любой функции $\psi(x)$

$$\| \psi \| = \left[\sum_p \psi^2(x^p)\right]^{1/2}. \tag{1.87}$$

С каждой функцией $f(x)$ связан n-мерный вектор f с координатами $f^p = f(x^p)$. Скалярное произведение таких векторов определяется стандартным образом:

$$(f, g) = \sum_p f^p g^p. \tag{1.88}$$

Вектор F с координатами $F^p = \sum_i \alpha_i f_i(x^p)$ является линейной комбинацией векторов f_i с координатами $f_i^p = f_i(x^p)$. Линейную оболочку семейства векторов f_i обозначим $L = L(\{f_i\})$. Построим в пространстве L ортонормированный базис с помощью последовательной ортогонализации векторов f_i. Каждый следующий шаг ортогонализации выполняем так, чтобы величина проекции F на новый вектор базиса была максимальной из возможных. Процесс ортогонализации продолжаем, пока

$$\| F \|^2 - \| F_\perp \|^2 > \varepsilon^2, \tag{1.89}$$

где $F_\perp$ — проекция F на построенную ортонормированную систему. После окончания процесса полагаем $F' = F_\perp$.

Опишем вычисления более детально:

1) проводим нормировку: полагаем для всех i

$$g_i^0 = f_i / \| f_i \|;$$

2) вычисляем $|(F, g_i^0)|$, находим среди этих чисел максимальное (пусть его номер $i_{0\max}$), полагаем $e_1 = q^0_{i_{0\max}}$, исключаем $g^0_{i_{0\max}}$ из множества $\{g_i^0\}$, полагаем $g_i^1 = g_i^0 - (g_i^0, e_1)e_1$, исключаем из этого множества нулевые векторы, если таковые существуют, проводим нормировку $g_i^1 := g_i^1 / \| g_i^1 \|$;

3) пусть определены $e_1, \ldots, e_l$ и не более чем $n - 1$ нормированных векторов g_i^l. Среди векторов g_i^l ищем такой $g^l_{i_{1\max}}$, для которого $|(F, g_i^l)|$ принимает максимальное значение, полагаем $e_{l+1} = g^l_{i_{1\max}}$, исключаем $g^l_{i_{1\max}}$ из множества векторов $\{g_i^l\}$; полагаем $g_i^{l+1} = g_i^l - (g_i^l, e_{l+1})e_{l+1}$, исключаем из этого множества нулевые векторы, если таковые существуют, нормируем $g_i^{l+1} := g_i^{l+1} / \| g_i^{l+1} \|$.

Вычисления проводим, пока

$$\| F \|^2 - \sum_{i=1}^{l} (F, e_l)^2 > \varepsilon^2. \tag{1.90}$$

После завершения вычислений имеем набор ортонормированных векторов $e_1, \ldots, e_l$ $(1 \leqslant n)$. Они являются линейными комбинациями векторов $f_{i_{1\max}}, \ldots, f_{i_{l\max}}$. Коэффициенты разложения e_j по набору $f_{i_{1\max}}, \ldots, f_{i_{l\max}}$ могут быть вычислены и сохранены в ходе ортогонализации. Полагаем $J = \{i_{1\max}, \ldots, i_{l\max}\}$,

$$F' = \sum_{j=1}^{l} (F, e_j)e_j = \sum_{j \in J} \beta_j f_j. \tag{1.91}$$

Это и есть решение задачи. Числа β_j выражаются через коэффициенты разложения e_1 $(l = 1, ..., l)$ по f_j $(j \in J)$ и скалярные произведения (F, e_j): если $e_i = \sum_{j \in J} q_{ij} f_j$, то $\beta_j = \sum_{i=1}^{l} (F, e_i) q_{ij}$.

Разложение e_i по f_j $(j \in J)$ имеет треугольную структуру и прозрачную рекурсивную форму:

$$e_1 = \frac{1}{\| f_{i_{1\max}} \|} f_{i_{1\max}},$$

$$e_2 = \frac{f_{i_{2\max}} - (f_{i_{2\max}}, e_1)e_1}{\| f_{i_{2\max}} - (f_{i_{2\max}}, e_1)e_1 \|},$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \qquad (1.92)$$

$$e_j = \frac{f_{i_{j\max}} - \sum_{r=1}^{j-1} (f_{i_{j\max}}, e_r)e_r}{\| f_{i_{j\max}} - \sum_{r=1}^{j-1} (f_{i_{j\max}}, e_r)e_r \|},$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

Замечание. В некоторых случаях часть функций f_i предполагается сохранить в F', например, для адаптивного сумматора естественно всегда сохранять $f_0 \equiv 1$, так как эта функция не использует данных. Тогда следует начинать ортогонализацию с сохраняемых функций f_i, после чего переходить к описанной процедуре с максимизацией $|(F, g_i^l)|$.

Для функций вида

$$F(x, \alpha) = \varphi\left(\sum_i \alpha_i f_i(x)\right) \qquad (1.93)$$

с дифференцируемой φ процедура аналогична с точностью до замены скалярного произведения в L: вместо

$$(f, g) = \sum_p f^p g^p \qquad (1.94)$$

используется скалярное произведение с весами

$$(f, g) = \sum_p w_p f^p g^p, \qquad (1.95)$$

где $w_p = \left(\varphi'\left(\sum_i \alpha_i f_i(x^p)\right)\right)^2$. В этом скалярном произведении вычисляются все нормы $\| f \| = (f, f)^{1/2}$ и проводится ортогонализация.

Для функций с пороговой нелинейностью на выходе

$$F(x, \alpha) = h\left(\sum \alpha_i f_i\right) \tag{1.96}$$

(h — функция Хевисайда) вновь используем скалярное произведение с весами

$$w_p = \left|\sum_i \alpha_i f_i(x^p)\right|^{-2}. \tag{1.97}$$

Описанная процедура сокращения "сверху вниз" с ортогонализацией особенно важна для упрощения элементов сложных сетей, в структуре которых и вектор входных сигналов элемента может быть далек от исходных данных, и его выходной сигнал далек от оцениваемого выхода всей сложной системы.

Проблема обусловленности. При проведении ортогонализации системы функций f_i на некоторой выборке $\{x^p\}$ часто возникают сложности, связанные с тем, что векторы значений этих функций на $\{x^p\}$ почти линейно зависимы. Приведем простейший пример. Пусть на плоскости задана выборка из двух векторов: $x^1 = (1, 2)$, $x^2 = (2{,}4 + 0{,}0001)$, а исходные функции $f_{1,2}$ — просто значения координат: $f_1 = x_1$, $f_2 = x_2$. Построим из них ортонормированный базис:

$$e_1 = f_1 / \|f_1\|, \; e_2 = (f_2 - (f_2, e_1)e_1) / \| f_2 - (f_2, e_1)e_1 \|,$$

$$\|f_1\| = \sqrt{5}, \; e_1 = (1/\sqrt{5}, \; 2/\sqrt{5}),$$

$$(f_2, e_1) = 10/\sqrt{5} + 0{,}0002/\sqrt{5},$$

тогда функция $g = f_2 - (f_2, e_1)e_1$ принимает на точках выборки значения

$$g(x^1) = -0{,}00004, \; g(x^2) = 0{,}00002;$$

соответственно

$$\| f_2 - (f_2, e_1)e_1 \| = 10^{-5}\sqrt{20}.$$

Небольшие ошибки в исходных данных могут катастрофически сильно изменить вектор e_2 и результат разложения.

Для контроля обусловленности можно поступать по-разному. Простейший способ состоит в том, что в ходе ортогонализации не проводится нормировка векторов g_i^l, но норма вычисляется, и как только обнаруживается $\| g_i^l \| < \delta$, соответствующий вектор g_1^l исключается из дальнейшего рассмотрения (здесь $\delta > 0$ — некоторый заранее задаваемый порог). Опишем подробнее алгоритм с контролем обусловленности.

1. Вычисляем $\| f_i \|$, полагаем $\{g_i^0\} = \{f_1 | \; \|f_i\| > \delta\}$.

2. Вычисляем $|(F, g_i^0)| / \| g_i^0 \|$,

находим среди этих чисел максимальное (пусть его номер $i_{0\max}$),

полагаем $e_1 = g^0_{i_{0\max}} / \| g^0_{i_{0\max}} \|$,

исключаем $g^0_{i_{0\max}}$ из множества $\{g^0_i\}$,

полагаем $g^1_i = g^0_i - (g^0_i, e_1)e_1$,

вычисляем $\| g^1_i \|$,

исключаем из множества $\{g^1_i\}$ те векторы, для которых $\| g^1_i \| < \delta$.

3. Пусть определены $e_1, \ldots, e_l$ и не более чем $n - 1$ векторов g^l_i, для которых $\|g^l_i\| > \delta$,

ищем среди векторов g^l_i такой $g^l_{i_{1\max}}$, для которого $|(F, g^l_i)| / \| g^l_i \|$ принимает максимальное значение,

полагаем $e_{l+1} = g^l_{i_{1\max}}$,

исключаем $g^l_{i_{1\max}}$ из множества векторов $\{g^l_i\}$,

полагаем $g^{l+1}_i = g^l_i - (g^l_i, e_{l+1})e_{l+1}$, вычисляем $\| g^{l+1}_i \|$, исключаем из множества $\{g^{l+1}_i\}$ те векторы, для которых $\| g^{l+1}_i \| < \delta$.

Вычисления проводим, пока выполнено неравенство (1.90), либо до того момента, когда $\{g^l_i\} = \varnothing$.

Для контрастирования с контролем обусловленности важно выполнение принятого ранее соглашения: искомая функция *реализована* и происходит только сокращение уже существующего элемента — этого достаточно для того, чтобы требования достаточно высокой точности (ε) и достаточно хорошей обусловленности не вступали в противоречие при правильном выборе ε, δ.

Процедуры анализа значимости и сокращения описания выделяют наиболее важные параметры и связи в нейронных сетях. По аналогии с обработкой изображений их называют процедурами *контрастирования.*

Контрастирование — чрезвычайно важная операция в проектировании нейронных сетей. Его роль не сводится только к сокращению описания. Более общая, например, такая задача: привести параметры системы к выделенному набору значений, в частности, уменьшить разрядность, что важно для удешевления специализированных устройств, экономии памяти и др.

Методов контрастирования существует много, и со временем появятся подробные технические руководства "Контрастирование нейронных сетей". Здесь ограничимся несколькими примерами, демонстрирующими основные блоки идей.

6. Рекурсивное контрастирование и бинаризация

Рекурсивное контрастирование состоит в последовательной модификации параметров системы — одного за другим. Для этого параметры должны быть как-то линейно упорядочены $(\alpha_1, \alpha_2, \ldots, \alpha_N)$. При

модификации α_i используются модифицированные значения $\alpha_1, ..., \alpha_{i-1}$ и исходные $\alpha_{i+1}, ..., \alpha_N$. Приведем пример такой процедуры.

Пусть для адаптивного сумматора задана обучающая выборка входных векторов $x^1, ..., x^n$ и соответствующих выходных сигналов $f_1, ..., f_n$, а также известны значения параметров, которые реализуют эту функцию:

$$f_i = \alpha_0 + (x^i, \alpha). \tag{1.98}$$

Требуется произвести *бинаризацию* сумматора, т.е. найти такие числа a, b и вектор β с координатами 0 или 1, чтобы значения функции

$$\varphi(x) = a + b(x, \beta) \tag{1.99}$$

на выборке $\{x^i\}$ по возможности меньше отличались от f_i. Критерием такого отличия будем считать

$$H = \sum_{i=1}^{n} (f_i - \varphi(x^i))^2. \tag{1.100}$$

Построим координаты вектора β по порядку: $\beta_1, \beta_2, \ldots$. Если на основании каких-либо дополнительных соображений предлагается какой-либо специальный порядок, то полагаем, что β_i выстроены именно в нем и координаты соответствующим образом перенумерованы.

Пусть построены $\beta_1, ..., \beta_{i-1}$. Обозначим:

β^{0i} — вектор с координатами $(\beta_1, ..., \beta_{i-1}, 0, 0, ...)$ (последние $N - i + 1$ координат — нули);

β^{1i} — вектор с координатами $(\beta_1, ..., \beta_{i-1}, 1, 0, ...)$ (последние $N - i$ координат — нули);

α^i — вектор с координатами $(0, ..., 0, \alpha_{i+1}, ...)$ (первые i координат — нули).

Введем функции:

$$\varphi_0^i(x) = a_{0i} + b_{0i}(x, \beta^{0i}) + (x, \alpha^i);$$

$$\varphi_1^i(x) = a_{1i} + b_{1i}(x, \beta^{1i}) + (x, \alpha^i); \tag{1.101}$$

$$H_{0i} = \sum_{j=1}^{n} (f_j - \varphi_0^i(x^j))^2; \; H_{1i} = \sum_{j=1}^{n} (f_j - \varphi_1^i(x^j))^2.$$

Определим параметры a_{0i}, b_{0i}, a_{1i}, b_{1i} из условий $H_{0i} \to \min$, $H_{1i} \to \min$. Без труда выписываются явные формулы для этих параметров (обычные формулы для коэффициентов линейной регрессии, получаемые методом наименьших квадратов). Обозначим соответствующие минимальные значения H_{0i}, H_{1i} через h_{0i}, h_{1i}.

Если $h_{0i} \leqslant h_{1i}$, то полагаем $\beta_i = 0$, в противном случае $\beta_i = 1$.

После того как построены все $\beta_1, \ldots, \beta_N$ (N — размерность вектора данных), автоматически определяются a и b: если $\beta_N = 0$, то $a = a_{0N}$, $b = b_{0N}$, в противном случае $a = a_{1N}$, $b = b_{1N}$.

Если же бинаризация проведена, а необходимая точность не достигнута, то можно построить второй бинаризованный сумматор, корректирующий ошибку первого — так, чтобы в сумме они хорошо аппроксимировали работу исходного сумматора на элементах обучающей выборки. В описанной процедуре заменяем

$$f_i := f_i - \varphi(x^i), \ \alpha := 0, \tag{1.102}$$

и для этих исходных данных вновь строим бинаризованный сумматор по алгоритму рекурсивной бинаризации. Повторяем такое построение, пока не будет достигнута удовлетворительная точность. В результате получим набор бинаризованных сумматоров, которые в совокупности (т.е. в результате суммирования выходных сигналов) достаточно точно аппроксимируют исходный.

Предположение о возможности бинаризации содержательно опирается на два других:

1) избыточность (размерность вектора данных существенно больше числа значимых факторов);

2) невырожденность главных факторов (достаточно много координат вектора данных не являются приблизительно постоянными на обучающей выборке).

При появлении весов, определяющих значимость отдельных примеров из обучающей выборки, рекурсивная бинаризация проводится точно так же, только в функциях H появляются веса. Способы назначения весов при контрастировании для различных элементов подробно описаны ранее.

Если требуется тем же способом упростить квадратичный сумматор или любой другой элемент, линейный по параметрам

$$F(x, \alpha) = \sum_i \alpha_i f_i(x), \tag{1.103}$$

то вместо обучающей выборки x^j берем семейство векторов y^j с координатами

$$y_i^j = f_i(x^j). \tag{1.104}$$

После такого преобразования рассматриваемый элемент превращается в обычный сумматор. Последовательность действий для него уже описана.

Мы построили процесс рекурсивной бинаризации так, чтобы элемент сохранял качество своей работы на заданной конечной выборке входных векторов. Другая постановка задачи возникает, если потребовать сохранения этого качества на бесконечном множестве. Например, при обработке сигналов и изображений можно попытаться провести

бинаризацию сумматоров, используя скоррелированность значений сигнала в близких точках. Требуется, чтобы качество работы сохранялось на всех таких сигналах. Для этого примем гипотезу, что множество сигналов, обладающих ожидаемыми корреляционными свойствами, может быть получено из множества всех сигналов, ограниченных по амплитуде некоторым числом, с помощью *сглаживающей фильтрации.* Пусть U — множество сигналов, ограниченных по амплитуде некоторым числом, M — операция сглаживающей фильтрации. Тогда требуется оценить в конечном виде качество работы элемента на множестве

$$V = \{M(u) \mid u \in U\}. \tag{1.105}$$

Для однородной решетки отсчетов (моментов снятия сигнала, пикселей изображения и др.) процедура сглаживающей фильтрации имеет вид

$$(M(u))_r = \sum_{r'} m_{r-r'} u_{r'}, \tag{1.106}$$

где m_r — ядро фильтра; r, r' — точки решетки отсчетов (векторы); а суммирование проводится по всей решетке.

Действие сумматора на сглаженном векторе из V представляется как

$$a_0 + \sum_{r,r'} \alpha_r m_{r-r'} u_{r'} = \alpha + \sum_{r'} \left(\sum_{r} \alpha_r m_{r-r'} \right) u_{r'}. \tag{1.107}$$

Для того чтобы в результате бинаризации слабо изменилось действие сумматора на сглаженных сигналах из V, достаточно, чтобы малым было изменение его действия на ядро фильтра. Можно ввести среднеквадратичное расстояние между сумматорами $\alpha_0 + (x, \alpha)$ и $\beta_0 + (x, \beta)$:

$$H = (\alpha_0 - \beta_0)^2 + \sum_{r'} \left(\sum_{r} (\alpha_r - \beta_r) m_{r-r'} \right)^2. \tag{1.108}$$

Рекурсивную бинаризацию проводим, минимизируя H на каждом шаге. Эта процедура даже проще той, которая была построена для конечных выборок, так как на каждом шаге строится однородное уравнение линейной регрессии — бинаризованный сумматор вычисляет функцию $\alpha_0 + b(x, \beta)$ с исходным α_0.

Выбор ядра для сглаживающего фильтра m_r может проводиться по разным критериям. Удобным и в некоторых отношениях оптимальным является такой фильтр:

$$m_r = \begin{cases} Z\left(1 - \dfrac{\| r \|}{r_0}\right), & \text{если } 1 - \dfrac{\| r \|}{r_0} > 0; \\ 0, & \text{если } 1 - \dfrac{\| r \|}{r_0} \leqslant 0, \end{cases} \tag{1.109}$$

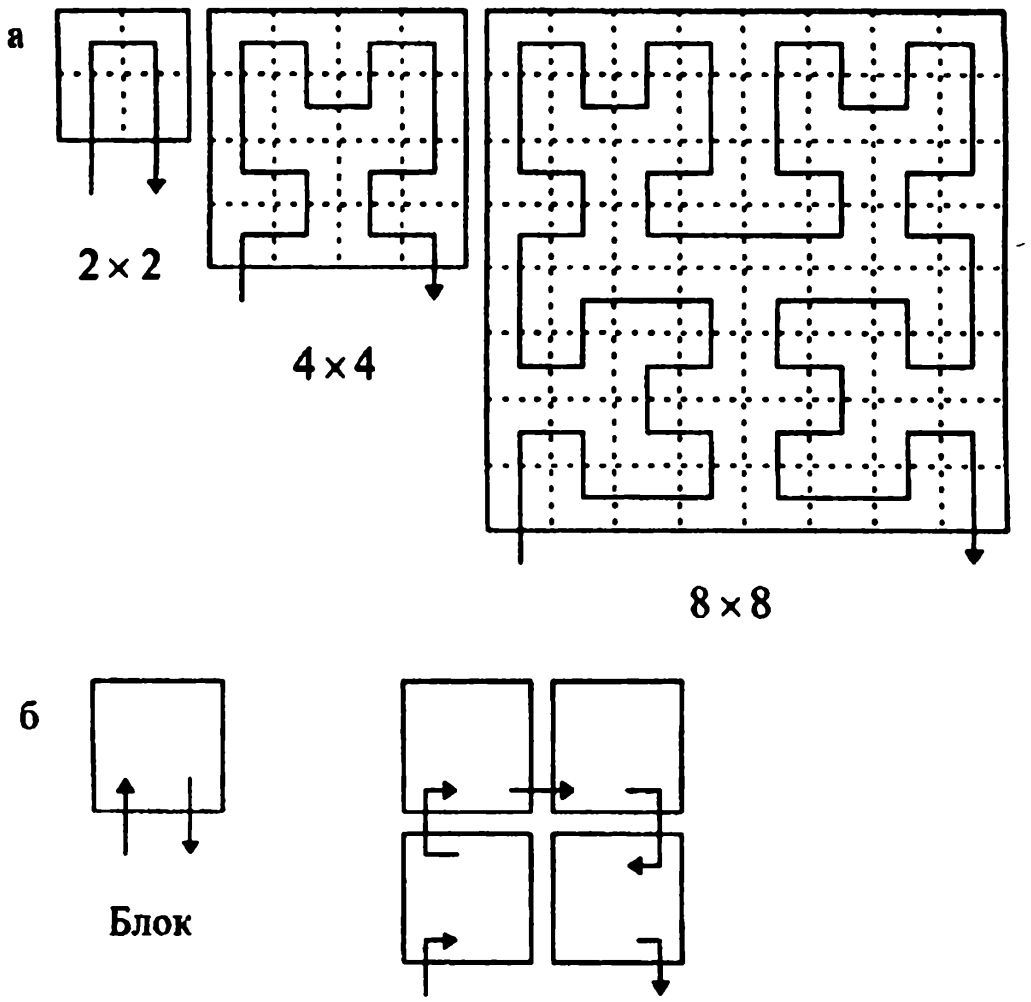


Рис. 1.12. Развертка Гильберта.
а — примеры для $k = 1, 2, 3$; *б* — схема удвоения стороны.

где $\| r \|$ — некоторая норма r; r_0 — радиус корреляции (единственный параметр фильтра); Z — константа нормировки, обеспечивающая условие $\sum_r m_r = 1$. При необходимости краевые эффекты (ограниченность множества отсчетов на решетке) легко учитываются — все компоненты, выходящие за границу, полагаются равными нулю.

Осуществление рекурсивных алгоритмов контрастирования предполагает выбор процедуры линейного упорядочения координат. В некоторых случаях (временной ряд) существует естественный линейный порядок. Если же такового нет (например, для двумерных массивов, возникающих при обработке изображений), то необходимо выбрать процедуру *развертки.* Чтобы показать нетривиальность этого выбора и возможные здесь идеи, приведем один пример.

Развертка Гильберта строится рекурсивным способом для двумерных квадратных решеток со стороной 2^k (рис. 1.12). Основное достоинство развертки Гильберта состоит в том, что на прямой максимально близко располагаются точки, близкие на плоскости.

В следующей главе приступим к работе с сетями. Предпошлем этому переходу два замечания:

1) существует большое число просто настраиваемых и реализуемых элементарных "устройств", с помощью которых уже можно решать различные задачи; прежде чем начинать конструирование сетей, полезно проверить, нельзя ли решить нужную задачу с помощью одного элемента;

2) существует много способов работы (например, упрощения) с отдельными элементами, в том числе включенными в систему; переходя к сетям, полезно об этом помнить и возвращаться на "элементарный" уровень при первой возможности.

Глава 2

НЕЙРОННЫЕ СЕТИ

1. Основные архитектуры

В большинстве обзоров, книг и статей мир нейронных сетей представлен как своеобразный "зоопарк". Каждая нейронная сеть — обитатель зоопарка — имеет свою архитектуру, систему предобработки и подачи входных сигналов, способ интерпретации ответов, свои правила настройки параметров и тип решаемых задач. Так же как и в настоящем зоопарке, различные элементы и подсистемы могут быть весьма похожи, но отдельно от целых обитателей они не живут. Такой подход для технических систем долго существовать не может. Необходимо стандартизировать элементы и подсистемы и создать своего рода конструктор, позволяющий собирать разнообразные нейронные системы обработки информации. Набор стандартных блоков тоже может рассматриваться как зоопарк, но обитают в нем уже детали, а не целые устройства (в этом заключается принципиальное отличие технологического зоопарка от биологического).

Мы не претендуем здесь на решение задачи стандартизации, но все же делаем шаг в эту сторону, по возможности разделяя элементы, архитектуры, предобработчики, правила обучения, способы интерпретации ответов и т.п. Раздельное рассмотрение элементов и блоков и сборка из них систем для решения различных задач обработки данных не содержит в себе существенного новаторства, однако систематическое следование этому подходу особенно полезно при проектировании и создании программного обеспечения. К этому вопросу мы еще вернемся.

В этой главе и далее рассматриваются нейронные сети, *синхронно функционирующие в дискретные моменты времени.* При программной реализации на PC асинхронные сети и непрерывное время не представ-

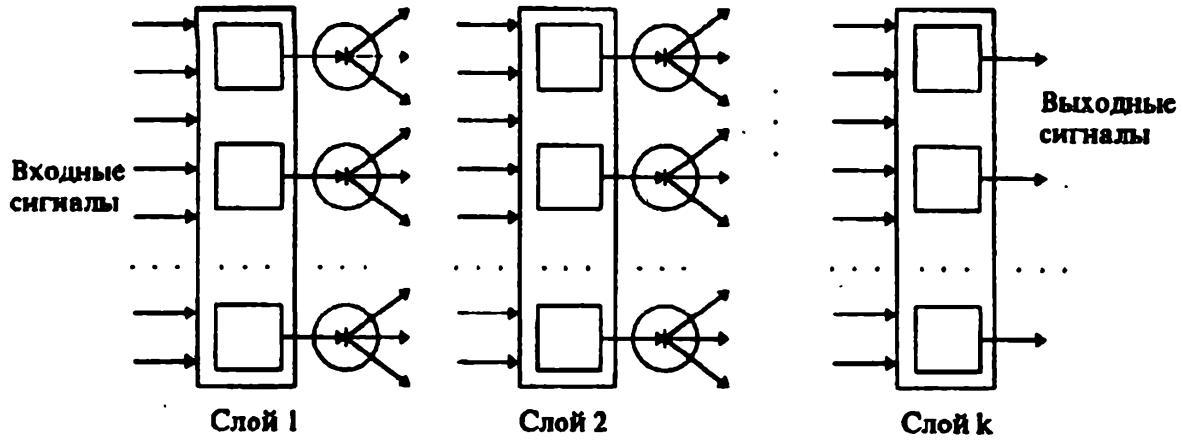


Рис. 2.1. Слоистая сеть.

ляют интереса. Другое дело — при проектировании аналоговых устройств, однако этот сюжет далек от обсуждаемых вопросов.

В зоопарке нейронных сетей можно выделить две базовые архитектуры — слоистые и полносвязные сети — и еще несколько особенно важных дополнительных.

Слоистые сети: нейроны расположены в несколько слоев (рис. 2.1). Нейроны первого слоя получают входные сигналы, преобразуют их и через точки ветвления передают нейронам второго слоя. Далее срабатывает второй слой и т.д. до k-го, который выдает выходные сигналы для интерпретатора и пользователя. Если не оговорено противное, то *каждый* выходной сигнал i-го слоя подается на вход *всех* нейронов $i + 1$-го. Число нейронов в каждом слое может быть любым и никак заранее не связано с количеством нейронов в других слоях. Стандартный способ подачи входных сигналов: все нейроны первого слоя получают каждый входной сигнал. Особое распространение получили *трехслойные* сети, в которых каждый слой имеет свое наименование: первый — входной, второй — скрытый, третий — выходной.

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

Элементы слоистых и полносвязных сетей могут выбираться поразному. Существует, впрочем, стандартный выбор — нейрон с адаптивным линейным сумматором на входе (рис. 1.8).

Для полносвязной сети входной сумматор нейрона фактически распадается на два: первый вычисляет линейную функцию от входных сигналов сети, второй — линейную функцию от выходных сигналов других нейронов, полученных на предыдущем шаге.

Функция активации нейронов (характеристическая функция) φ — нелинейный преобразователь, преобразующий выходной сигнал сумматора (см. рис. 1.8), — может быть одной и той же для всех нейронов сети. В этом случае сеть называют *однородной (гомогенной)*.

Если же φ зависит еще от одного или нескольких параметров, значения которых меняются от нейрона к нейрону, то сеть называют *неоднородной (гетерогенной).*

Выбор в качестве элементов сети нейронов стандартного вида (см. рис. 1.8) не является обязательным. Слоистая или полносвязная архитектура не налагает существенных ограничений на участвующие в ней элементы. Единственное жесткое требование, предъявляемое архитектурой к элементам сети, — это соответствие размерности вектора входных сигналов элемента (она определяется архитектурой) числу его входов.

Если полносвязная сеть функционирует до получения ответа заданное число тактов k, то ее можно представить как частный случай k-слойной сети, все слои которой одинаковы и каждый из них соответствует такту функционирования полносвязной сети.

Существенное различие между полносвязной и слоистой сетями возникает тогда, когда число тактов функционирования заранее не ограничено — слоистая сеть так работать не может.

Монотонные слоистые сети (рис. 2.2) — специальный частный случай слоистых сетей с дополнительными условиями на связи и элементы. Каждый слой, кроме последнего (выходного), разбит на два блока: возбуждающий (B) и тормозящий (T). Связи между блоками разделяются на два типа: возбуждающие (+) и тормозящие (-). Если от блока A к блоку C ведут только возбуждающие связи, то это означает, что любой выходной сигнал блока C является *монотонной неубывающей* функцией любого выходного сигнала блока A. Если же эти связи только тормозящие, то любой выходной сигнал блока C является *монотонной невозрастающей* функцией любого выходного сигнала блока A.

Связи в монотонной сети: к последнему слою от тормозящего блока предпоследнего слоя идут тормозящие связи, а от возбуждающего блока — возбуждающие связи. Между собой возбуждающие и тормозящие блоки взаимодействуют так:

— от возбуждающего блока $i-1$-го слоя идут возбуждающие связи к возбуждающему блоку i-го слоя и тормозящие к тормозящему блоку i-го слоя;

— от тормозящего блока $i-1$-го слоя идут возбуждающие связи к тормозящему блоку i-го слоя и тормозящие к возбуждающему блоку i-го слоя (см. рис. 2.2).

Для элементов монотонных сетей необходима монотонная зависимость выходного сигнала элемента от параметров и входных сигналов.

И первый возбуждающий, и первый тормозящий блоки могут получать все входные сигналы. Для обеспечения *логической прозрачности* при настройке параметров монотонных сетей накладывается дополнительное требование минимизации числа общих входных сигналов этих слоев. В идеале такое требование приводит к разбиению

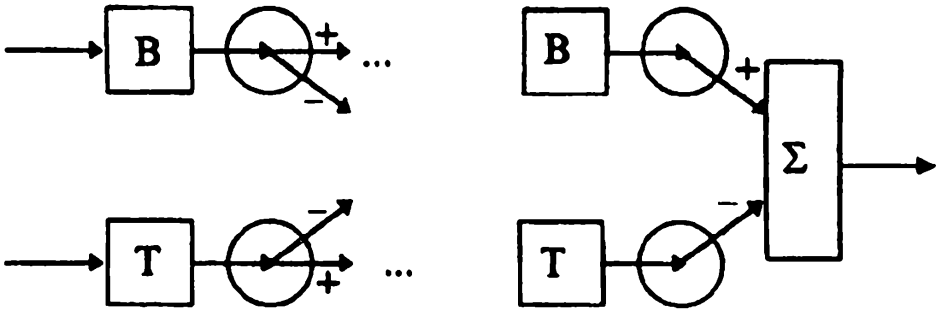


Рис. 2.2. Монотонная слоистая сеть.

множества входных сигналов на два непересекающихся подмножества, одно из которых подается первому возбуждающему слою, второе — первому тормозящему.

Основное достоинство монотонных сетей — возможность для каждого параметра и промежуточного сигнала определить характер его влияния на каждый выходной и промежуточный сигнал, основываясь только на месте элемента в структуре сети.

Легко видеть, что требования монотонности можно распространить не только на слоистые сети, сохраняя главное условие: для каждого промежуточного сигнала в сети y, каждого ее параметра α и любого выходного сигнала z определены знаки зависимостей z от y и α — это либо монотонное возрастание, либо убывание. Мы, однако, склонны использовать исключительно слоистые монотонные сети.

Слоисто-циклические сети отличаются тем, что слои замкнуты в кольцо — последний (k-й) передает свои выходные сигналы первому (рис. 2.3). Все слои равноправны и могут как получать входные сигналы, так и выдавать выходные. Такие сети могут до получения ответа функционировать неограниченно долго — так же, как и полносвязные.

Слоисто-полносвязные сети состоят из слоев, каждый из которых, в свою очередь, представляет собой полносвязную сеть. При функционировании сигналы передаются от слоя к слою и происходит обмен сигналами внутри слоя. В каждом слое этот процесс протекает следующим образом: прием сигналов с предыдущего слоя (и/или

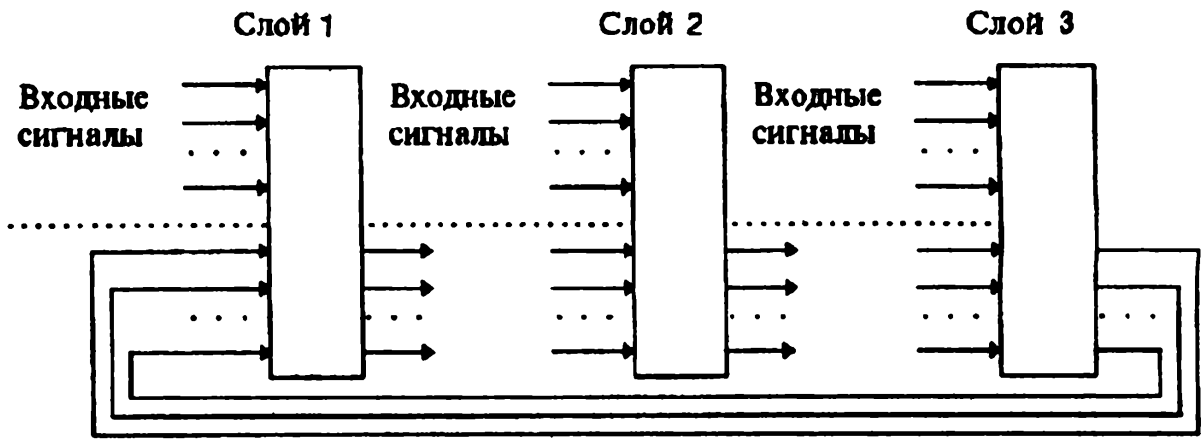


Рис. 2.3. Слоисто-циклическая сеть.

входных) → обмен сигналами внутри слоя → передача последующему слою (или на выход).

Подобные сети до получения ответа функционируют определенное число тактов, соответствующее количеству слоев — так же, как и слоистые сети.

Полносвязно-слоистые сети по своей структуре такие же, но функционируют по-другому. В них не разделяются фазы обмена внутри слоя и передачи следующему: на каждом такте нейроны всех слоев принимают сигналы от нейронов как своего слоя, так и предыдущего, после чего передают сигналы как внутри слоя, так и последующему (или на выход). До получения ответа такие сети могут функционировать неограниченно долго — так же, как и полносвязные.

2. Входы, выходы и функционирование

Стандартный способ подачи входных сигналов состоит в том, что они подаются на специальные входы элементов (для стандартных нейронов — на специальные входные сумматоры). Существует, однако, и несколько других вариантов подачи входных сигналов сети:

1) *один сигнал — один нейрон* — входные сигналы подаются сразу на нелинейные преобразователи, минуя сумматоры, которые используются только для обмена сигналами между нейронами[1];

2) *подача на те же сумматоры, которые служат для обмена сигналами;*

3) *параметрическая подача* — выделенному массиву параметров (обычно синаптическим весам — некоторым параметрам выделенных адаптивных сумматоров) присваиваются значения, равные входным сигналам.

Стандартный способ снятия выходных сигналов состоит в том, что некоторые нейроны объявляются выходными и их выходные сигналы в определенные моменты времени считаются выходными сигналами сети. Только это и выделяет их из всего набора нейронов.

Существует также несколько дополнительных вариантов снятия сигналов:

1) *аддитивное снятие с синапсов* — на выход подаются сигналы, проходящие по выделенным связям, уже умноженные на соответствующие веса;

2) *параметрическое снятие выходов* — выходными сигналами считаются установившиеся значения выделенных параметров (например, веса выделенных связей).

---

[1]Несмотря на то что этот способ значительно хуже стандартного — требует большего числа нейронов для решения тех же задач, фактически вынуждает увеличивать число тактов функционирования или слоев и т.п., он до сих пор часто используется. Это означает, что многие пользователи и разработчики программного обеспечения заботятся больше о моде (о возможности заявить: "У нас тоже есть нейронные сети"), чем об эффективности.

В последнем случае процесс функционирования должен включать изменение выходных синапсов. В обычной постановке задачи это соответствует, скорее, обучению. Функционирование с изменением параметров можно рассматривать как обучение со вспомогательной оценкой, параметры которой являются подстраиваемыми в ходе "настоящего" обучения, устанавливающего правильное соответствие вход — выход при данном типе выходных сигналов.

Далее, если специально не оговорено противное, мы ограничимся рассмотрением стандартных способов организации входов и выходов сети.

Сети периодического функционирования. Простейшие представления об этих сетях таковы. В начальный момент состояния всех нейронов одинаковы, выходных сигналов нет. Предполагается, что характеристики нейронов это позволяют. Подаются входные сигналы, инициирующие активность сети (нулевой такт). Далее входные сигналы могут подаваться на каждом такте функционирования. На каждом такте могут сниматься выходные сигналы. После k тактов цикл функционирования заканчивается и сеть возвращается в исходное состояние, готовая к новому циклу (акту). Между актами функционирования могут вставляться акты обучения.

В общем случае в результате цикла из k тактов нейронная сеть выдает в ответ на последовательность из k наборов входных сигналов последовательность из k наборов выходных сигналов.

Чаще используется упрощенный вариант — входные сигналы подаются только в самом начале, выходные снимаются в самом конце.

Для слоистых и слоисто-полносвязных сетей начальные слои по мере срабатывания освобождаются и могут заниматься новой задачей, пока последние слои заканчивают работу над предыдущей.

Сети периодического функционирования по характеру использования напоминают обычные ЭВМ: на вопрос следует ответ, причем воспроизводимый. Иначе обстоит дело с сетями непрерывного функционирования.

Можно организовать функционирование нейронной сети, не приводя ее каждый раз перед подачей задания в фиксированное состояние. При этом перед получением задания сеть находится в некотором состоянии, определяемом предысторией, нейроны выдают выходные сигналы. В течение k тактов подаются входные и снимаются выходные сигналы. После такого функционирования можно проводить цикл обучения, оценивая успешность работы, а можно продолжать функционирование.

Непрерывное функционирование нейронной сети (с подучиванием или без него) более соответствует имеющимся представлениям о поведении живых существ, чем периодическое. Опыт показывает, что, чередуя циклы функционирования и обучения, для таких сетей можно получить хорошие результаты адаптации.

Важная особенность сетей непрерывного функционирования: для них естественным образом можно ввести *спонтанную активность нейронов.* Не требуется отсутствия выходных сигналов при нулевых входных. Простейший пример характеристической функции нейрона со спонтанной активностью:

$$\varphi(A) = \frac{A}{\alpha_1 + |A|} + \alpha_2,\ \alpha_2 > 0. \qquad (2.1)$$

Параметры спонтанной активности (здесь α_2) для таких нейронных сетей должны подбираться в ходе обучения.

Конечно, нейроны со спонтанной активностью можно применять и в сетях периодического функционирования, однако там это менее естественно из-за необходимости стандартного начального состояния и жесткой связи входных и выходных сигналов.

Из описанных основных архитектур любая может использоваться для периодического функционирования, но для непрерывного необходимы сети с циклами: полносвязные, слоисто-циклические или полносвязно-слоистые.

3. Сети с включением макронейронов — новое средство моделирования

Многие исследователи не доверяют адаптивному моделированию с нейронными сетями или без них, из-за его "чистого эмпиризма" — всегда вызывает сомнение возможность адаптивных моделей к обобщению, к работе с теми данными, которые не входили в обучающую выборку.

Теоретическое мышление — мощный инструмент, специально придуманный для выхода в мышлении за пределы непосредственного опыта. Хорошо продуманная модель с мощной теоретической базой предпочтительнее регрессионного уравнения, полученного из обработки данных.

Даже в том случае, когда имеющиеся данные лучше согласуются с эмпирическим уравнением, всегда есть подозрение, что новые разрушат это согласование. Разрушение же модели, получившей теоретическое обоснование и как-то согласующейся с экспериментом, представляется нам существенно менее вероятным. Более того, даже если обнаружено значительное расхождение такой модели с эмпирическими данными, то это, в свою очередь, представляет важную информацию — каждое такое нарушение может быть осмыслено и содержательно истолковано. Напротив, ошибки эмпирической модели обычно не несут большого смысла, так как сама модель не имеет глубоких теоретических оснований и контекст для обсуждения этих ошибок слишком беден.

Противоречие между эмпиризмом и теоретическим мышлением Нового времени имеет давнюю историю, богатую великими событиями. Еще Аристотель строил математические модели падения тел, подбирая формулы, описывающие зависимость времени падения от веса тела. Виртуозно продолжал эту линию Леонардо да Винчи. Задачей эмпиризма всегда было установление прямых связей между наблюдаемыми величинами. Метод — обработка результатов наблюдений и опытов.

Фундаментальное теоретическое мышление отчетливо проявилось, пожалуй, в работах Галилея. Истоки же можно найти в представлениях Платона о мире идей и в теологическом учении о душе. Модель как теоретически предполагаемая "душа" объекта апробируется потом в эксперименте. И эксперимент этот принципиально отличается от опытов, существующих в рамках эмпиризма: он требует специально очищенной ситуации, позволяющей сделать выбор одного из теоретически проработанных вариантов. Для эмпиризма опыт — источник, для теоретического мышления эксперимент — судья.

В первой половине XX в. шло интенсивное обсуждение вопроса о выводимости теоретических принципов и конструкции из опытов. Итог (на сегодняшний день): вывести их нельзя.

Строго говоря, ни чистый эмпиризм, ни чисто теоретическое моделирование никогда не реализуются. В исследовании всегда присутствуют априорные элементы и рациональные конструкции — чтобы избавиться от них, пришлось бы отказаться от использования языка, поэтому чистый эмпиризм не достижим. Но и теоретические модели всегда оставляют место для эмпирического подбора параметров. Еще ни разу не удалось построить такой теории, в которой было бы выводимо все и не оставалось бы ни одного свободного параметра.

Будучи идеологически диаметрально противоположными, эмпиризм и теоретическое мышление успешно сосуществуют в любой практике моделирования. Способ их сосуществования можно описать так. Во-первых, они существуют на фоне друг друга и каждая из указанных тенденций придает смысл противоположной, позволяет выделить ее из практики моделирования, где они слиты.

Во-вторых, теоретическое мышление задает способы представления объекта и его выделения ("очищения") и этим определяет некоторые рамки для моделирования, а эмпиризм адаптирует модели, оставаясь в пределах этих рамок.

Наконец, эмпиризм оценивает фундаментальные конструкции с точки зрения соответствия объекту "как он есть".

В результате естественным образом складывается процесс последовательного исчерпывания объекта: модели создаются, "подгоняются" под опытные данные, критикуются, совершенствуются вместе со своими теоретическими основаниями и т.п.

Нейросетевые модели представляют собой, как правило, крайне эмпирическое направление. Как крупное достижение порой рассмат-

ривается освобождение от теоретических (онтологических) рамок — нейросетевые системы пригодны для моделирования любых объектов, независимо от их природы. Единая технология для работы в различных областях — это, конечно, замечательно, но хотелось бы не терять при этом преимуществ фундаментального теоретического подхода, позволяющего заглядывать в суть вещей и экономить усилия.

Для объединения теоретических моделей и универсального эмпирического нейросетевого подхода предлагается по-новому взглянуть на проблему соответствия модели объекту.

Традиционно различия между теоретической моделью и реальным поведением объекта представляются как "потенциально рациональные", и только временно не схваченные из-за дефектов модели и/или теории. Взглянем на это по-другому. Модель никогда не будет полностью соответствовать объекту. Для любой теории различия всегда остаются. Обозначим их как "зону хаоса" (в противовес рациональной модели) и отдадим универсальным эмпирическим нейросетевым моделям — пусть нейросети, адаптируясь, научатся сводить непредсказуемость ошибки к минимуму. Предсказанная ошибка — уже не совсем ошибка.

Итак, предлагается вводить затравочные рациональные теоретические обоснованные модели и в качестве поправок к ним использовать нейросетевые адаптивные эмпирические модели, которые учатся на ошибках затравочных. Это предложение находит свое архитектурное воплощение в использовании комбинированных устройств — нейронных сетей, объединенных с автоматами, реализующими рациональные модели.

Назовем эти автоматы *макронейронами.*

Взаимодействие нейронной сети с макронейронами может происходить следующим образом:

1) *коррекция выходов* — выходом объединенной системы считается результат преобразования выходного вектора макронейрона с помощью нейронной сети;

2) *коррекция параметров* — параметры макронейрона являются выходными сигналами нейронной сети;

3) *коррекция входов* — макронейрону подаются не истинные входные сигналы, а результат их преобразования нейронной сетью.

Все эти три варианта могут реализовываться одновременно. Со временем появятся детально разработанные инструментальные средства для моделирования на основе нейронных сетей с макронейронами. Естественная область приложения таких моделей очень широка: моделирование сложных экологических, экономических, биологических и технологических систем. Возможно применение и в более традиционных областях: газо- и гидродинамика, управление летательными аппаратами и т.п. В лингвистике, в машинном переводе тоже найдется немало задач для таких систем.

4. Задачи для нейронных сетей

Множество задач, решаемых с помощью нейронных сетей, постоянно пополняется, поэтому сложно дать их завершенную классификацию. Принимаясь за разработку программного инструментария, необходимо иметь обзор приложений, хотя бы и неполный. Этим определяется и форма предлагаемого перечня задач, и наше отношение к нему как к незавершенному и открытому для дополнений.

Многие задачи, для решения которых используются нейронные сети, могут рассматриваться как частные случаи следующих основных проблем.

1. Построение функции по конечному набору значений.
2. Оптимизация.
3. Построение отношений на множестве объектов.
4. Распределенный поиск информации и ассоциативная память.
5. Фильтрация.
6. Сжатие информации.
7. Идентификация динамических систем и управление ими.
8. Нейросетевая реализация классических задач и алгоритмов вычислительной математики: решение систем линейных уравнений, решение задач математической физики сеточными методами и др.

Однозначно построить функцию (обычно многих действительных переменных) по конечному набору значений невозможно без специальных дополнительных условий. В качестве таких условий могут использоваться требования минимизации различных функционалов, например интеграла суммы квадратов вторых производных — требование максимальной гладкости. При этом известные в конечном множестве точек значения функции превращаются в набор ограничений, при которых ищется минимум функционала.

С помощью нейронных сетей строится, естественно, нейросетевая реализация функции: создается нейронная сеть, которая, получая на входе вектор аргументов, выдает на выходе значение функции. Обычно предполагается, что типичные нейросетевые реализации достаточно "хорошие" и любая из них подойдет в качестве решения задачи. При необходимости вместо наложения условий типа максимальной гладкости минимизируют число слоев, количество нейронов и/или число связей, а также требуют, чтобы функция активации нейронов была "максимально пологой". Для обычно используемых нами

$$\varphi(A) = \frac{A}{h + |A|} \qquad (2.2)$$

последнее означает максимизацию h. Все это — при условии, что функция принимает известные значения в заданных точках.

К построению функции по конечному набору значений приводит одна из самых нужных для пользователей задач: *заполнение пропусков в таблицах.* Пусть, как обычно, каждая строка таблицы данных

соответствует какому-либо объекту, а в строках указаны значения признаков (свойства) соответствующих объектов.

В подавляющем большинстве случаев данные неполны: по крайней мере, для части объектов неизвестны значения некоторых признаков. Необходимо как-то их восстановить. Достоверное статистическое оценивание должно давать для отсутствующих данных их условное математическое ожидание (условия — известные значения других признаков) и характеристику разброса — доверительный интервал. Это, однако, требует либо непомерно большого объема известных данных, либо очень сильных предположений о виде функций распределения. Приходится вместо *статистически достоверных* уравнений регрессии использовать *правдоподобные* нейросетевые реализации[2].

Анализ временных рядов содержит комплекс задач, которые сводятся к построению функций по конечным наборам значений и заполнению пробелов в таблицах. Временные ряды представляют собой весьма специальный вид таблиц и заслуживают отдельного рассмотрения. Для каждого типа объектов выделяется набор постоянных признаков (констант) и множество свойств, меняющихся со временем (переменных признаков). Предполагается, что в любой момент времени для каждого объекта существуют свои значения переменных признаков. Вот, к примеру, три задачи, специфичные для обработки временных рядов:

а) определение констант (всех или части) по известным значениям переменных в разные моменты времени;

б) предсказание значений части переменных в некоторый момент времени по известным значениям констант, переменных в нескольких предшествовавших моментах времени и части переменных в текущий момент;

в) определение объема данных о прошлом, достаточных для предсказания будущего на конкретное время и с заданной точностью.

Кроме обычных проблем нехватки данных, в решении этих задач может встретиться еще специфическое осложнение — неравномерность отсчетов времени.

Классификация с учителем также может рассматриваться как задача заполнения пропусков в таблицах: для каждого класса в таблицах есть поле, в котором указывается, принадлежит объект данному классу или нет. В эти поля можно ставить численные значения,

---

[2]Термин *правдоподобные* мы понимаем в смысле, предложенном Дж. Пойя в его знаменитой книге "Математика и правдоподобные рассуждения". Любая, даже самая строгая, математическая конструкция сначала создается всего лишь как правдоподобная гипотеза. *Правдоподобные* выводы — это те, которые еще не прошли испытания на достоверность и строгость, однако именно в таком виде рождается все сколько-нибудь интересное. Кроме того, уместно напомнить, что утверждения о статистической достоверности базируются на весьма ограничительных гипотезах о статистической природе эмпирического материала (см. обсуждение этих вопросов, например, в [20]).

например 1, если объект принадлежит классу, и 0 (или −1) — в противном случае. Возможна и иная кодировка.

Для части объектов, составляющих обучающую выборку, известно, каким классам они принадлежат. Требуется построить нейронную сеть, которая по признакам объекта (записанным в других полях таблицы) определяла бы, к какому классу он принадлежит, т.е заполняла бы соответствующие поля.

Представляет интерес и задача *нечеткой классификации с учителем.* Если 1 на соответствующем месте в таблице означает принадлежность объекта к классу, а 0 — точное отрицание этого, то действительное число между 0 и 1 можно интерпретировать как "степень уверенности" — качественную характеристику того, насколько уверенно можно отнести объект к данному классу.

Пусть построена нейронная сеть, дающая на обучающей выборке требуемые ответы 0 и 1. Обычно выходные сигналы нейронной сети — действительные числа. При обработке примера, чей класс неизвестен, нейросеть для каждого класса выдаст число. Этот набор чисел можно проинтерпретировать, считая, например, что максимальное из них указывает на класс, к которому действительно принадлежит объект. Такое правило интерпретации называется "победитель забирает все". Если выходные сигналы интерпретируются по этому правилу, то мы получаем обычную четкую классификацию. Можно поступить и по-другому. Рассмотрим выходные сигналы сети как меры уверенности при отнесении объекта к различным классам. В таком случае получим нечеткую классификацию. Роль интерпретатора при этом сводится только к нормировке выходных сигналов сети — приведению их к единичной сумме, если предполагается, что любой объект принадлежит какому-либо известному классу. В противном случае сумма выходных сигналов может быть и меньше 1. Дефект (единица минус сумма выходных сигналов) интерпретируется как степень уверенности в принадлежности объекта к неизвестному классу, отсутствующему в первоначальной классификационной модели.

Оценку производных функций, заданных только конечными наборами значений, удобно проводить с помощью их нейросетевых реализаций. Если для некоторой функции f построена вычисляющая ее нейронная сеть, то трудоемкость вычисления градиента f лишь в несколько раз (обычно не более трех) превосходит трудоемкость вычисления одного значения f. Это не зависит от размерности пространства входов, что может показаться странным, так как обычная интуитивно понятная оценка дает не менее одного вычисления f на одну частную производную $\partial f/\partial x_i$:

$$T_{\mathrm{grad}f} \geqslant nT_f, \tag{2.3}$$

где $T_{\mathrm{grad}f}$ и nT_f — количества машинных операций, необходимых для вычисления $\mathrm{grad}f$ и f соответственно.

Тем не менее, если аккуратно организовать вычисление градиента сложной функции многих переменных, не вычисляя ничего лишнего и не дублируя уже проведенных расчетов, то для функций, реализованных с помощью сетей из обычных нейронов с линейными сумматорами, получим

$$T_{\text{grad}\, f} \leqslant CT_f, \tag{2.4}$$

где C — не зависящая от размерности константа ($C \approx 3$). (Подробнее об этом в следующей главе.)

Простота вычисления градиента позволяет использовать нейросетевые реализации таблично заданных функций для решения задачи *оптимизации градиентными методами.* Сочетая эту оптимизацию с дополнительными измерениями, получаем удобную *технологию планирования оптимизирующего эксперимента:*

1) по заданным начальным экспериментальным данным строим нейросетевую реализацию целевой функции;

2) градиентными или комбинированными методами (о методах оптимизации в следующей главе) находим экстремум нейросетевой реализации целевой функции;

3) проводим дополнительный эксперимент в найденной точке экстремума, результат добавляем к обучающей выборке и вновь возвращаемся в п. 1.

Весь цикл повторяется, пока не будет достигнут удовлетворительный результат.

До сих пор речь шла о функциях, определяемых конечным набором значений, однако задача *оптимизации аналитически заданных функций* может быть не менее важной и сложной. Для ряда таких функций удается построить минимизирующие их нейронные сети. Самая простая из таких задач — минимизация квадратичной формы. Из более сложных укажем (а в следующем разделе подробно опишем) приближенный поиск экстремума в знаменитой задаче коммивояжера: дано n городов и указаны все попарные расстояния между ними. Требуется объехать все города, побывав в каждом по одному разу, и вернуться в исходную точку так, чтобы пройденный путь был минимален.

Построение отношений на множестве объектов — одна из самых загадочных и открытых для творчества областей применения искусственного интеллекта. Первым и наиболее распространенным примером этой задачи является *классификация без учителя.* Задан набор объектов, каждому объекту поставлен в соответствие вектор значений признаков (строка таблицы). Требуется разбить эти объекты на классы эквивалентности. Естественно, прежде чем приступить к решению этой задачи, нужно ответить на вопросы: зачем проводится это разбиение и что мы будем делать с его результатом. Ответы позволят приступить к формальной постановке задачи, которая всегда требует компромисса между сложностью решения и точностью фор-

мализации: буквальное следование содержанию задачи нередко порождает сложную вычислительную проблему, а следование за простыми и элегантными алгоритмами может привести к противоречию со здравым смыслом.

Итак, зачем нужно строить отношения эквивалентности между объектами? В первую очередь, для фиксации знаний. Люди накапливают знания о классах объектов — это практика многих тысячелетий, зафиксированная в языке: знание относится к имени класса (пример стандартной древней формы: "люди смертны", "люди" — имя класса). В результате классификации как бы появляются *новые имена и правила их присвоения.* Для каждого нового объекта[3] мы должны сделать две вещи: 1) найти класс, к которому он принадлежит; 2) использовать новую информацию, полученную об этом объекте, для исправления (коррекции) правил классификации.

Какую форму могут иметь правила отнесения к классу? Веками освящена традиция представлять класс его "типичным", "средним", "идеальным" и т.п. элементом. Этот типичный объект является идеальной конструкцией, олицетворяющей класс. Отнесение объекта к классу проводится путем его сравнения с типичными элементами разных классов и выбора ближайшего. Правила, использующие типичные объекты и меры близости для их сравнения с другими, очень популярны и сейчас.

Простейшая мера близости объектов — квадрат эвклидового расстояния между векторами значений их признаков (чем меньше расстояние, тем ближе объекты). Соответствующее определение признаков типичного объекта — среднее арифметическое значение признаков по выборке, представляющей класс.

Ближайшее обобщение — квадрат расстояния Махаланобиса. Тут даже три варианта: свое расстояние Махаланобиса в каждом классе, общее расстояние, порожденное всей совокупностью объектов, и общее, соответствующее средней ковариационной матрице классов. Для этих мер близости вектор значений признаков типичного объекта вновь получается усреднением по классу (т.е. взятием центра масс выборки, представляющей класс).

Другая мера близости, естественно возникающая при обработке сигналов, изображений и т.п., — квадрат коэффициента корреляции (чем он больше, тем ближе объекты). Возможны и иные варианты — все зависит от задачи.

Если число классов m заранее определено, то задачу классификации без учителя можно поставить следующим образом. Пусть $\{x^p\}$ — векторы значений признаков для рассматриваемых объектов и в пространстве таких векторов определена мера их близости $\rho\{x, y\}$

---

[3]Здесь и далее мы не оговариваем специально существование априорных ограничений налагаемых на новые объекты. Естественно, что "вселенная" задачи много уже и гораздо определеннее Вселенной.

Для определенности примем, что чем ближе объекты, тем меньше ρ. С каждым классом будем связывать его типичный объект. Далее называем его *ядром* класса. Требуется определить набор из m ядер $y^1, y^2, \ldots, y^m$ и разбиение $\{x^p\}$ на классы:

$$\{x^p\} = Y_1 \cup Y_2 \cup \ldots \cup Y_m, \tag{2.5}$$

минимизирующее следующий критерий:

$$Q = \sum_{i=1}^{m} \left(\sum_{x^p \in Y_i} \rho(x^p, y^i) \right) \to \min. \tag{2.6}$$

Минимум Q берется по всем возможным положениям ядер y^i и всем разбиениям $\{x^p\}$ на m классов Y_i.

Если число классов заранее не определено, то полезен *критерий слияния классов*: классы Y_i и Y_j сливаются, если их ядра ближе, чем среднее расстояние от элемента класса до ядра в одном из них:

$$\rho(y^i, y^j) < \max \left\{ \frac{1}{|Y_i|} \sum_{x^p \in Y_i} \rho(x^p, y^i), \frac{1}{|Y_j|} \sum_{x^p \in Y_j} \rho(x^p, y^j) \right\}, \tag{2.7}$$

где $|Y|$ — число элементов в Y. Возможны варианты: введение коэффициента перед max, замена максимума по классам Y_i, Y_j средним значением и т.д.

Использовать критерий слияния классов можно так: сначала принимаем гипотезу о достаточном числе классов, строим их, минимизируя Q, затем некоторые Y_i объединяем, повторяем минимизацию Q с новым числом классов и т.д.

Существует много эвристических алгоритмов классификации без учителя, основанных на использовании мер близости между объектами. Каждый из них имеет свою область применения, а наиболее распространенным недостатком является отсутствие четкой формализации задачи: совершается переход от идеи кластеризации прямо к алгоритму, в результате неизвестно, что ищется (но что-то в любом случае находится, иногда — неплохо).

Приведем пример иного рода. Поставим проблему так: разбить объекты на классы, в которых значения признаков связаны близкими соотношениями. Эта *задача мозаичной регрессии* также требует построения правил классификации, но ни мер близости между объектами, ни типичных представителей классов в ней не возникает.

Простейший и самый важный для приложений случай — это *линейные соотношения* между данными. Пусть, как и выше, $\{x^p\}$ — исходная выборка, а Y — некоторое ее подмножество. Скажем, что в Y имеется k линейных соотношений между данными, если Y целиком лежит в некотором линейном многообразии (аффинном подпространстве) L коразмерности k.

Нас интересуют приближенные соотношения, поэтому введем следующую меру близости множества векторов данных Y до произвольного линейного многообразия L:

$$\rho(Y, L) = \sum_{x^p \in Y} \rho(x^p, L), \tag{2.8}$$

где $\rho(x^p, L)$ — квадрат эвклидового расстояния от точки x^p до L, т.е. квадрат длины перпендикуляра, опущенного из x^p на L.

Обобщения на другие варианты определения мер близости между точками (расстояние Махаланобиса и др.) очевидны.

Выделим три вопроса:

1) определение коразмерности k;
2) определение числа классов m;
3) поиск классов при заданных k, m.

Последний вопрос формализуется легче всего (по крайней мере однозначнее). Пусть заданы k, m. Для m линейных многообразий L_i $(i = 1, 2, \ldots, m)$ коразмерности k и разбиения

$$\{x^p\} = Y_1 \cup Y_2 \cup \ldots \cup Y_m.$$

Поставим задачу:

$$Q = \sum_{i=1}^{m} \rho(Y_i, L_i) \to \min \tag{2.9}$$

— построение классов определяется как поиск набора $\{L_i\}$ и разбиения $\{Y_i\}$, минимизирующих Q.

Определение числа классов m можно проводить с помощью эвристических правил слияния и разделения уже построенных классов. Что касается k, то представляют интерес классификации при различных значениях коразмерности. Если данных достаточно, то имеет смысл проводить классификацию последовательно, начиная с $k = 1$, далее разбивать полученные классы для $k = 2, 3$ и т.д. Получим разбиение последовательно измельчаемое с ростом k.

Необходимо помнить, что k, m, общее число объектов в выборке N и размерность векторов данных n должны быть связаны неравенствами

$$m(n - k + 1) \ll N, \tag{2.10}$$

поскольку множество из $n - k + 1$ точек в n-мерном пространстве всегда лежит на линейном многообразии коразмерности k и в ситуации общего положения однозначно определяет его.

При $k = n$ получаем нульмерные L_i-точки и возвращаемся к рассмотренной ранее классификации, т.е. группировке вокруг типичных представителей. Последнее неравенство превращается при этом в $m \ll N$ — число классов много меньше, чем число точек.

Если для некоторого k решена задача мозаичной регрессии, найдены наилучшие L_i и Y_i $(i = 1, 2, ..., m)$, то это означает, что k координат вектора данных могут быть восстановлены с помощью линейных соотношений. Эти соотношения определяются линейными многообразиями L_i и одинаковы для всех объектов из Y_i.

Если о новом объекте получена неполная информация с l пропущенными данными $(l < k)$, то вместо вектора (точки) x мы можем поставить в соответствие этому объекту l-мерное линейное многообразие S. Недостающие данные попытаемся восстановить следующим образом. Среди линейных многообразий L_i ищем ближайшее к S. Расстояние между L_i и S определяется так:

$$\rho(S, L_i) = \min_{x \in S, y \in L_i} \rho(x, y). \qquad (2.11)$$

Пусть L — ближайшее к S многообразие L_i и $x^0 \in S$ — та точка, в которой достигается минимум. Тогда можно рассматривать x^0 как оценку вектора значений признаков нового объекта. Для $l = k$ (и тем более при $l > k$) подобное восстановление уже невозможно: в ситуации общего положения k-мерное многообразие имеет общую точку с $n - k$-мерным и $\rho(S, L_i) = 0$ для всех i.

Таким образом, решив задачу мозаичной регрессии, получаем возможность определять класс объекта при неполной информации о нем и восстанавливать недостающие сведения.

Не вызывает особых проблем замена линейных многообразий алгебраическими (с ограничениями на степень), однако из-за возрастания числа подгоночных параметров требуется соответствующее увеличение количества экспериментальных точек, что не всегда достижимо.

Следующий пример показывает, что практический интерес представляют не только отношения эквивалентности. Постановка обычной задачи регрессии (или мозаичной регрессии) исходит из гипотезы о том, что одни характеристики объектов могут быть функциями других и эти функции одни и те же для всех объектов (или соответственно классов объектов). Транспонируем таблицу данных (поменяем местами слова "объект" и "признак"). Рассмотрим гипотезу о том, что значения признака одного объекта могут быть функциями значений того же признака других объектов и эти функции одни и те же для всех признаков (или классов признаков). Получаем формально те же задачи регрессии или мозаичной регрессии (будем называть их *транспонированными задачами регрессии*). Есть, однако, два содержательных отличия транспонированных задач от исходных:

1) *инвариантность к смене шкал измерения* — кажется маловероятным, чтобы существенные связи между признаками различных объектов зависели от шкалы измерения, поэтому необходимо, чтобы уравнения транспонированной регрессии были инвариантны от-

носительно смены шкалы измерения любого признака (обычно — линейного неоднородного преобразования $x_i' = ax_i + b$, однородная часть которого описывает смену единицы измерения, а свободный член — сдвиг начала отсчета);

2) в традиционных задачах регрессии предполагается, что объектов достаточно много, по сравнению с числом признаков, иначе (при $N < n$) точные линейные соотношения возникнут просто из-за малого числа объектов, так как через N точек всегда можно провести линейное многообразие размерности $N - 1$. В противовес этому "транспонированное" предложение о достаточно большом числе признаков ($n \gg N$) кажется нереалистичным.

Требование инвариантности к смене шкал приводит к специальным ограничениям на вид функций регрессии, а недостаточность количества признаков (в сравнении с числом объектов) для построения транспонированной регрессии вынуждает нас по каждому объекту искать небольшую группу объектов, по свойствам которых можно восстановить характеристики данного.

Задача построения таких групп объектов была чрезвычайно популярна в химии перед открытием Менделеевым периодического закона (1871 г.). С 1817 г. (Деберейнер) опубликованы десятки работ на эту тему [164]. Именно они поставили исходный материал для систематизации элементов. Деберейнер обнаружил триады, в которых свойства среднего элемента могут быть оценены как средние значения этих свойств для крайних членов триады. Его труды продолжили Гмелин, Гладстон, Дюма и др. Вот некоторые из таких триад:

K—Na—Li, Ba—Sr—Ca, Cl—Br—J, S—Se—Te, P—As—Sb,

W—V—Mo, ...

Один из наиболее полных списков триад был опубликован Ленсеном (1857 г.)[4]. Он же заметил, что для большей точности иногда полезно брать "эннеады" — девятки, составленные из трех триад.

Более общим образом задача ставится так: найти для каждого объекта наилучшую линейную формулу, выражающую его вектор признаков через векторы признаков других объектов (которых должно быть по возможности меньше). Эта формула должна быть инвариантна относительно смены шкал.

Итак, требуется построить отношение, связывающее объекты с группами объектов, по которым для него строятся интерполяционные формулы. Проделав такую работу "в лоб" (по базам данных и без обращения к интуиции химиков) для большого числа элементов (объ-

---

[4]Д.И. Менделеев писал: "...между всеми... учеными, которые раньше меня занимались сравнением величин атомных весов элементов, я считаю, что обязан преимущественно двум: Ленсену и Дюма. Я изучил их исследования и они меня побудили искать действительный закон" (цит. по [164, с. 220—222]).

ектов) и потенциалов ионизации (признаков), мы получили хорошее согласие с экспериментом и предсказали ряд неизвестных ранее высших потенциалов ионизации [66, 67].

Заметим, что в своем обсуждении задач, приводящих к построению отношений между объектами, мы не отошли далеко от проблемы заполнения пробелов в таблицах — все остальные задачи пока возникали как надстройки над этой проблемой и средства для ее решения. Есть все же круг вопросов, которые требуют построения отношений между объектами и не связаны прямо с восстановлением неизвестных данных. Это, например, построение отношений сходства для быстрого поиска объектов, сходных с данным, — задача, возникающая при конструировании *ассоциативной памяти* [98, 99, 175, 276]. Такие отношения сходства могут не быть транзитивными (т.е., если объект A сходен с объектом B, а B сходен с C, то A уже не обязательно сходен с C), поэтому задача их построения отличается от задач классификации.

Для конструирования ассоциативной памяти необходимо:

1) построить отношения сходства между объектами;

2) так организовать хранение и обработку данных, чтобы по вновь поступившей информации о каком-либо объекте легко осуществлялся поиск наиболее близких к нему;

3) при наличии в поступающей информации ошибок и/или ее неполноте ассоциативная память также должна удовлетворительно функционировать (по крайней мере, при не слишком больших искажениях);

4) пополнение данных и функционирование системы ассоциативной памяти должны в максимальной степени допускать параллельную реализацию — в идеале поиск объектов, наиболее сходных с данным, должен происходить за несколько тактов сети (один такт — срабатывание всех нейронов) — так же, как и внесение в память сведений о новом объекте.

Соответствие последнему требованию — одно из основных достоинств нейронных систем ассоциативной памяти.

Мы потребовали от ассоциативной пямяти успешной работы при наличии шума и/или при неполной информации об объекте. Это сближает поставленную задачу с проблемой *фильтрации* [266, 270] — отделения сигнала от шума. Есть и существенные отличия: ассоциативная память работает на основе базы данных об известных объектах, а фильтрация, кроме того, может основываться всего лишь на общих предположениях о качественных отличиях сигнала от шума. От этих предположений существенно зависит построение фильтров. Так, предполагая, что сигнал более гладкий, чем шум (соседние значения более скоррелированны), получим в простейшем случае сглаживающий фильтр. Если же, напротив, предположить, что основную значимую информацию несут резкие перепады значений сигнала, то придем к контрастирующим фильтрам. Примеры и тех, и других приводились в гл. 1.

Если же существует гипотеза о том, что сигнал, очищенный от шума, близок к одному из некоторого стандартного набора (например, при обработке речи, печатного или рукописного текста и т.п.), то задача фильтрации по существу погружается в проблему конструирования ассоциативной памяти.

Каждый пользователь персонального компьютера знаком с проблемой *сжатия информации.* Популярны программы-архиваторы для хранения и передачи данных. Эти программы могут сжать произвольный файл, с тем чтобы при необходимости развернуть его в исходное состояние. Классические архиваторы отличаются двумя свойствами: во-первых, они используют при архивировании только ту информацию, которая хранится в сжимаемом файле (и соответственно при разворачивании — только информацию из архивированного файла) и, во-вторых, они осуществляют полностью обратимое сжатие, позволяющее точно восстановить исходную информацию.

В своей практике все мы применяем и другие типы сжатия информации. Так, посылая кому-нибудь библиографическое описание статьи вместо ее текста, мы рассчитываем на то, что получатель сможет полностью восстановить содержащуюся в статье информацию, обратившись, например, в библиотеку. Это дает пример использования дополнительной информации. С другой стороны, при подготовке реферата мы тоже сжимаем информацию, но необратимо — невозможно точно восстановить книгу (статью, диссертацию) по реферату. Обычный в статистической обработке факторный анализ [109] позволяет нам заменять вектор данных значениями главных факторов. Этот вариант сжатия информации и необратим, и использует дополнительные сведения о выборке, к которой принадлежит вектор данных.

По перечисленным признакам можно составить таблицу из восьми типов сжатия информации.

Обычные архиваторы (arj и т.п.) принадлежат 5-му типу, замена статьи библиографическим описанием — 6-му, реферирование может соответствовать любому типу с 1-го по 4-й (в зависимости от способа составления реферата и его использования). Наконец, сжатие при факторном анализе относится к 4-му типу.

При попытке дать более формальные определения неизбежно возникнут вопросы такого типа: что следует считать "дополнительной информацией"?

Чаще всего нейронные сети используются для различных вариантов *нелинейного факторного анализа, позволяющего понизить размерность данных,* но с потерей части информации при обработке новых объектов, отсутствующих в обучающей выборке. Возможно создание других, в частности гибридных, архиваторов, в которых за счет линейного или нелинейного факторного анализа происходит сжатие 4-го типа, а *различия* между исходной и восстановленной информацией сжимаются обычным архиватором 5-го типа. В результате

Таблица 2.1

Типы сжатия информации

№	Обратимость	Использование дополнительных данных	
		при сжатии	при разворачивании
1	Нет	Нет	Нет
2	Нет	Нет	Да
3	Нет	Да	Нет
4	Нет	Да	Да
5	Да	Нет	Нет
6	Да	Нет	Да
7	Да	Да	Нет
8	Да	Да	Да

получим эффективное сжатие 8-го типа — обратимое и использующее дополнительную информацию.

Нейросетевое моделирование динамических систем осуществляется так же, как и обычный подбор параметров в моделях [91, 194, 256]. Из априорных соображений выбирается архитектура нейросетевой модели, строится функционал ошибок, оценивающий отклонение выходного сигнала сети от истинного поведения наблюдаемых величин. Далее этот функционал минимизируется (чаще всего, градиентными методами). Весь процесс может происходить в реальном времени при постоянном поступлении новых данных о моделируемой системе.

Возможно одновременно с моделированием решать и задачу *оптимального адаптивного управления.* Для этого в сети выделяют две группы выходных сигналов: одна соответствует наблюдаемым величинам, другая — управляющим воздействиям на систему. Минимизируется взвешенная сумма функционала ошибок модели с целевым функционалом задачи управления. Обучение должно приводить к их совместной минимизации.

Задача адаптивного управления с помощью нейронных сетей принципиально не отличается от хорошо разработанных общих задач этого типа [183, 253] — специфика состоит только в особой нейросетевой конструкции управляющего устройства. Поэтому основные достижения теории адаптивного управления без особых затруднений переносятся и на управление с помощью нейронных сетей.

Решение классических задач вычислительной математики притягательно хотя бы тем, что каждая из них возникает во множестве предметных областей и ее реализация существенно повышает потребительские свойства нейрокомпьютера или программы-нейроимитатора. Первая такая задача — *поиск экстремума гладкой функции.* Если вычисление значений функции реализовано с помощью нейронной сети, то соответствующая задача оптимизации с помощью градиентных методов решается также с использованием этой сети: она вычисляет градиент функции, остальное выполняется дополнительным

устройством (подпрограммой) — учителем (см. в следующих главах). Если функция задана каким-либо иным образом, то можно поступать двояко: либо строить ее нейросетевую реализацию, либо прямо конструировать оптимизирующую сеть. Последний прием чаще всего применяется для аналитически заданных функций, например квадратичных форм.

Некоторые классические задачи могут быть поставлены как задачи минимизации гладкой функции, хотя исходно они имеют иной вид. Наиболее важный пример [166, 213]: решение системы линейных уравнений $Ax = b$ сводится к минимизации квадратичной формы $(Ax - b)^2 \to \min$.

Сеточные (в особенности современные многосеточные [184]) методы решения задач для уравнений в частных производных как будто специально придуманы для нейросетевой реализации, причем не надо ничего изобретать — они практически сразу имеют нужную форму, достаточно перевести формулы на язык нейросетевых схем.

Сложнее обстоит дело с дискретными задачами — такими как сортировка. Для них изобретаются параллельные (в частности, нейросетевые) алгоритмы решения, но успехи здесь пока ограниченны.

Уже каждая из упомянутых задач отличается своей спецификой и характерными сложностями реализации. Поскольку далее они имеют для нас факультативное значение, ограничимся кратким перечислением.

Еще раз отметим, что приведенный в этом разделе перечень задач для нейронных сетей неполон и открыт для дополнений.

5. Настройка нейронных сетей для решения задач

Основной сюжет этого раздела — формирование нейронных сетей для решения задач. Прежде чем приступить к поиску параметров сети (настройке адаптивных сумматоров и т.п.), нужно поставить задачу, т.е. ответить на вопросы: какие сигналы сеть будет получать, как мы будем интерпретировать сигналы, поступающие от сети, и, если сеть обучается путем минимизации ошибки, как мы будем оценивать работу сети (т.е., что такое вектор ошибок и как вычисляется целевая функция — оценка функционирования сети). Ответы на поставленные вопросы воплощаются в специальных устройствах или программах: *предобработчике, интерпретаторе ответов* и *оценке.* Итак, прежде чем формировать сеть, необходимо создать ее окружение.

В процессе обучения используются еще три "устройства": *обучающая выборка* (система, работающая с исходными данными), *учитель,* модифицирующий параметры, и *контрастер* (система, упрощающая нейронную сеть). О них речь пойдет в следующих разделах.

Нормировка и центрирование данных используются почти всегда (кроме тех случаев, когда данные представляют собой бинарные векторы с координатами 0,1 или ±1, либо символьные последовательности). Цель этих преобразований — сделать так, чтобы каждая компонента вектора данных лежала в отрезке [−1, 1] (либо [0, 1] в тех случаях, когда существенна положительность) или, по крайней мере, не слишком далеко выходила из этого отрезка, и ее характерный разброс тоже был бы примером единичным.

Вот два стандартных преобразования исходной выборки $\{x^p\}$ $(p = 1, ..., M)$:

1) $$x_i := \frac{x_i - \mathrm{M}(x_i)}{\sigma(x_i)};$$

2) $$x_i := \frac{x_i - \mathrm{M}(x_i)}{\max\limits_p |x_i^p - \mathrm{M}(x_i)|},$$

где x_i — i-я координата вектора; $\mathrm{M}(x_i) = \frac{1}{n}\sum\limits_p x_i^p$ — выборочная оценка математического ожидания x_i (среднее значение); $\sigma(x_i) = \left(\frac{1}{n}\sum\limits_p \left(x_i^p - \mathrm{M}(x_i)\right)^2\right)^{1/2}$ — выборочная оценка среднего квадратичного уклонения. Координаты, не имеющие разброса, либо исключаются из рассмотрения, либо исходя из априорных предположений о возможном появлении разброса в новых данных для них оценивают ожидаемую величину σ или максимум уклонения от среднего.

В формулах нормировки и центрирования нет ничего сложного, однако разработчика нейронных приложений подстерегает одна неприятность: любое изменение выборки $\{x^p\}$ должно согласно этим формулам менять и нормировку. Это порождает сложности с совместимостью различных нейросетевых приложений — по крайней мере, требуется все время следить за аккуратным пересчетом нормировки. Чтобы избежать таких малосодержательных затруднений, обычно вместо $\mathrm{M}(x_i)$ и $\sigma(x_i)$ берут некоторые фиксированные (не зависящие от выборки) величины, подходящие на данную роль по смыслу. Так, если x — температура больного, то вместо среднего по выборке можно взять 36,6 °С, а в качестве σ — 1 °С. Получим правило нормировки и центрирования $x := x - 36{,}6$, которое вполне выполняет свою функцию — сделать разброс примерно единичным, а среднее — примерно нулевым.

Иногда встречается мнение, что входной адаптивный сумматор в ходе обучения сам выстроит требуемую нормировку и центрирование и

нет необходимости выносить эти операции за пределы сети. В известных пределах так оно и есть: адаптивный сумматор может вычислить любую линейную функцию. Однако при обучении нейронных сетей на параметры налагаются ограничения[1] — обычно в виде их принадлежности некоторому отрезку. Из-за этого класс функций, вычисляемых нейронной сетью, теряет инвариантность по отношению к линейным преобразованиям координат входного вектора. Следовательно, возникает выбор: либо проводить нормировку и центрирование отдельно, перед подачей сигналов сети, либо для входных сумматоров выбирать пределы изменения параметров в зависимости от данных.

Мы предпочитаем, чтобы сеть была универсальной и ограничения на ее параметры не зависели от данных. Поэтому выбираем первый путь: нормировка и центрирование проводятся отдельно, а параметры лежат в заранее определенном интервале (в большинстве случаев — в отрезке $[-1, +1]$).

Нормировка и центрирование вписывают ("примерно вписывают") исходную выборку в куб со стороной 2, вершинами которого являются векторы с координатами ± 1. Чаще всего выборка при этом остается в каких-то направлениях вытянутой, а в каких-то — сжатой. Направления определяются собственными векторами и числами ковариационной матрицы Σ (см. гл. 1): выборку можно смоделировать эллипсоидом, главные оси которого расположены вдоль собственных векторов Σ, а квадраты их длин равны соответствующим собственным числам. Это обстоятельство используется в двух сравнительно редко применяемых процедурах предобработки. Обе они связаны с переходом в ортонормированный базис из собственных векторов Σ. Предположим пока, что такой базис найден: $e^1, e^2, ..., e^N$, а соответствующие собственные числа упорядочены от большего к меньшему: $\lambda_1 \geqslant \lambda_2 \geqslant ... \geqslant \lambda_N$. Первый способ предобработки состоит в том, что нейронной сети подается вектор сигналов y с координатами $y_i = (x, e^i)$. Другими словами, строится предобрабатывающий слой сумматоров с фиксированными весами, равными координатам векторов e^i $(i = 1, ..., N)$. Вектор данных x (нормированный и центрированный) сначала подается этому слою, а потом — обучаемой сети. Смысл этого преобразования состоит в том, что сразу выделяются и доминируют *главные факторы* — первые координаты вектора y в среднем больше остальных $(\sigma^2(y_i) \cong \lambda_i)$. Напомним, что параметры обучаемой сети выдерживаются в фиксированных пределах, поэтому такое доминирование главных факторов неустранимо преобразованием параметров.

---

[1]Иногда используются схемы без априорных ограничений на параметры. Они порождают семейство фатальных трудностей — таких как "убегание" параметров в ходе обучения на $\pm\infty$. В результате появляется ряд псевдопроблем и значительные усилия направляются на их решение. В прикладном программном обеспечении группы НейроКомп всегда применяются априорные ограничения на параметры сети.

Обычно не стоит подавать все y_i, поскольку большинство из них мало отличаются от нуля, поэтому ограничиваются несколькими первыми $y_1, \ldots, y_l$. Их число определяют из последовательности собственных чисел: l — наименьший из номеров, для которых $\lambda_{l+1}/\lambda_1 < \varepsilon^2$ (ε — малое число, выбираемое из соображений точности, обычно $\varepsilon = 0,1 \div 0,001$).

Можно использовать собственный базис ковариационной матрицы Σ прямо противоположным образом — не для того чтобы сократить описание и подавать только главные факторы, а для превращения выборки в "почти шарообразную" и метрики Махаланобиса — в эвклидово расстояние. Для этого достаточно всего-навсего подавать сети не $y_i = (x, e^i)$, а центрированные значения y_i:

$$z_i = \frac{1}{(\lambda_i)^{1/2}} (x, e^i). \qquad (2.12)$$

После такой предобработки эвклидово расстояние в пространстве векторов z равно расстоянию Махаланобиса в исходном пространстве.

Во многих реальных задачах часть проекций (x, e^i) настолько незначима, что их все равно приходится отбрасывать по критерию $\lambda_i/\lambda_1 < \varepsilon^2$ при достаточно малых ε. В сочетании с центрированием оставшихся координат получим "почти шарообразную" выборку в пространстве меньшей размерности.

Самое неприятное в описанной процедуре — вычисление собственных векторов и чисел матрицы Σ. Вспомним, однако, что нужна всего лишь оценка. Поэтому поступим следующим образом. Выберем некоторое начальное приближение $e^{1(0)}, e^{2(0)}, \ldots, e^{N(0)}$, $\lambda_1^{(0)} \geqslant \lambda_2^{(0)} \geqslant \ldots \geqslant \lambda_N^{(0)}$. С ним связана матрица Σ_0, для которой $e^{i(0)}$ — собственные векторы, а $\lambda_i^{(0)}$ — собственные числа. Построим рекуррентную процедуру поиска e^i, λ_i. Для этого на каждом (p-м) шаге будем добавлять к матрице Σ_{p-1} матрицу $\frac{1}{n} x^p \otimes x^{pT}$[2], вычитать $\frac{1}{n}\Sigma_0$ и вычислять поправки к собственным векторам и собственным числам в первом приближении по $\frac{1}{n}$. Пусть на $(p-1)$-м шаге получены приближенные собственные векторы $e^{i(p-1)}$ и собственные числа $\lambda_i^{(p-1)}$. Обозначим $\Delta e^i = e^{i(p)} - e^{i(p-1)}$, $\Delta\lambda_i = \lambda_i^{(p)} - \lambda_i^{(p-1)}$. В первом порядке теории возмущений получим

---

[2] Напомним, что $x \otimes y^T$ — произведение вектора-столбца x на вектор-строку y^T (тензорное произведение); выборка предполагается центрированной, поэтому берется произведение $x^p \otimes x^{pT}$.

$$\Delta\lambda_i = \frac{1}{n}\,(e^{i(p-1)},\ A_p e^{i(p-1)}),$$

$$\Delta e^i = \sum_{j\neq i} a_j e^{j(p-1)},\ a_j = \frac{1}{n}\,\frac{(e^{j(p-1)},\ A_p e^{i(p-1)})}{\lambda_i^{(p-1)} - \lambda_j^{(p-1)}}, \qquad (2.13)$$

где A_p — матрица возмущений (малый параметр $\frac{1}{n}$ для наглядности всюду вынесен за скобки):

$$A_p = x^p \otimes x^{p\mathrm{T}} - \Sigma_0. \qquad (2.14)$$

Как видим, последовательность операций достаточно проста и имеет типично нейросетевой характер: вычисляются скалярные произведения x^p на векторы $e^{j(p-1)}$ и матричные элементы Σ_0 в базисе $\{e^{j(p-1)}\}$. Можно даже нарисовать нейронную сеть, проводящую эти вычисления. Решением задачи считаем $e^i = e^{i(n)}$, $\lambda_i = \lambda_i^{(n)}$.

Выбор начального приближения Σ_0 может проводиться по-разному. Предложим здесь два варианта. Пусть $x^1, \ldots, x^N$ — случайно выбранные различные элементы исходной выборки, N — размерность пространства данных. Положим

$$e^{1(0)} = x^1 / \| x^1 \|,\ \lambda_1^{(0)} = \| x^1 \|^2; \qquad (2.15)$$

$$e^{2(0)} = \frac{x^2 - (x^2, e^{1(0)})e^{1(0)}}{\| x^2 - (x^2, e^{1(0)})e^{1(0)} \|},\ \lambda_2^{(0)} = \| x^2 - (x^2, e^{1(0)})e^{1(0)} \|^2;$$

.....

$$e^{k(0)} = \frac{x^k - \sum_{i=1}^{k-1}(x^k, e^{i(0)})e^{i(0)}}{\| x^k - \sum_{i=1}^{k-1}(x^k, e^{i(0)})e^{i(0)} \|},\ \lambda_k^{(0)} = \| x^k - \sum_{i=1}^{k-1}(x^k, e^{i(0)})e^{i(0)} \|^2,$$

.....

Другой вариант — поиск максимума $\lambda_k^{(0)}$ на каждом шаге по исходной выборке векторов $\{x^p\}$.

Без особых затруднений строится модификация метода на тот случай, когда векторы исходной выборки не являются полной системой в пространстве векторов данных. Нужно начинать с совокупности векторов максимального ранга k. Далее формулы теории возмущений не будут выводить из порожденного ими подпространства.

Какие *функции может реализовать нейронная сеть?* Многие задачи обучения нейронных сетей сводятся к следующей проблеме: на конечном множестве точек в n-мерном эвклидовом пространстве зада-

ны значения функции. Требуется построить нейтронную сеть, продолжающую эту функцию на область пространства.

Многими авторами доказывались *теоремы полноты для* функций, вычисляемых нейронными сетями [205, 242, 246, 267]. В частности, множество функций, допускающих нейросетевую реализацию, плотно в пространстве непрерывных функций на n-мерном кубе в топологии равномерной сходимости. Во всех этих теоремах, естественно, рассматриваются нейронные сети без ограничений на число элементов и значения параметров. Более того, как уже говорилось в первой части книги, избыточные способности сети — возможность аппроксимировать любую функцию — негативно сказываются на возможности ее использования для интерполяции и экстраполяции, так как в классе непрерывных функций существует слишком много различных вариантов продолжения функции с конечного множества на область.

Если задано число элементов сети и значения ее параметров заключены в определенные пределы, то возникают ограничения и на вычисляемые функции. Проще всего сформулировать существенную часть таких ограничений, используя константу Липшица. Сделаем это. Напомним определение константы Липшица вектор-функции $f(x)$ в области D:

$$L_f = \sup_{x,y \in D,\, x \neq y} \frac{\| f(x) - f(y) \|}{\| x - y \|}. \tag{2.16}$$

Верхняя грань берется по области определения D. Выбор нормы в этом определении не особенно важен, однако далее для оценки L_f мы пользуемся эвклидовой нормой, так как в ней особенно просто оценивается константа Липшица для важнейшего элемента нейронных сетей — адаптивного сумматора. Константа L_f легко оценивается сверху для суперпозиции вектор-функций $(\varphi \circ f)(x) = \varphi(f(x))$;

$$L_{\varphi \circ f} = L_\varphi L_f. \tag{2.17}$$

Аналогично, для линейной комбинации функций $f = \sum_i \alpha_i f_i$ получаем

$$L_f \leqslant \sum_i |\alpha_i| L_{f_i}. \tag{2.18}$$

Константа Липшица для адаптивного сумматора, вычисляющего функцию $\alpha_0 + (x, \alpha)$, есть

$$L_\Sigma = \| \alpha \|. \tag{2.19}$$

Для стандартного нейрона с гладкой функцией активации $\varphi(y)$ и адаптивным сумматором $\alpha_0 + (x, \alpha)$ на входе

$$L_N \leqslant \| \alpha \| \max_y |\varphi'(y)|. \tag{2.20}$$

Для прямой суммы вектор-функций

$$f(x) = f_1(x) \oplus f_2(x) \oplus \ldots \oplus f_N(x) \tag{2.21}$$

имеем

$$\mathrm{d}L_f \leqslant \left[\sum_i L_{f_i}^2\right]^{1/2}. \tag{2.22}$$

Тем самым константа Липшица слоя из стандартных нейронов оценивается как

$$L_s \leqslant \left[\sum_i \| \alpha^{(i)} \|^2 \max |\varphi'(y)|^2\right]^{1/2}, \tag{2.23}$$

где $\alpha^{(i)}$ — вектор параметров входного сумматора i-го нейрона (сумматор вычисляет линейную неоднородную функцию от вектора своих входных сигналов x: $\alpha_0^i + (x, \alpha^{(i)})$).

При последовательном соединении слоев или последовательных тактах функционирования одного слоя оценки для констант Липшица слоев перемножаются:

$$L_{\text{net}} \cong \prod_s L_s. \tag{2.24}$$

Таким образом, для всех архитектур нейронной сети легко исходя из ограничений на ее параметры получить оценку константы Липшица любой функции, которую эта сеть может вычислить. С другой стороны, если функция задана на конечной выборке, то можно получить выборочную оценку константы Липшица L_{sam}, заменяя в определении L_f всю область D на конечное множество:

$$L_{\text{sam}} \geqslant \sup_{x,y \in \{x^p\}, x \neq y} \frac{\| f(x) - f(y) \|}{\| x - y \|}. \tag{2.25}$$

Это — оценка L_{sam} снизу. Функцию можно реализовать с помощью данной сети только тогда, когда выборочная оценка L_{sam} снизу меньше, чем оценка L_{net} сверху по архитектуре сети.

При преобразовании нормировки $x_i := (x_1 - \mathrm{M}(x_i))/\sigma_i$ изменение выборочной оценки константы Липшица L_{sam} оценим как

$$\max_i \sigma_i L_{\text{sam}} \geqslant L'_{\text{sam}} \geqslant \min_i \sigma_i L_{\text{sam}}. \tag{2.26}$$

Получение выборочной оценки константы Липшица и проверка ее соответствия возможностям сети — важный этап предварительной обработки данных.

Не всегда, однако, удается заранее провести выборочную оценку константы Липшица — например, это вызывает трудности тогда, когда выборка значений функции в начальный момент не задана. Такие задачи встречаются часто: при анализе временных рядов, моде-

лировании динамических систем, планировании оптимизирующего эксперимента и др. В этих случаях значения функций, для которых строится нейросетевая реализация, поступают в ходе работы и заранее неизвестны. Тогда можно поступить по-другому.

Для каждой координаты вектора данных x_i обычно примерно известен отрезок $[a_i, b_i]$, в котором она должна лежать (оценка отрезка не должна быть слишком широкой — значения a_i и b_i должны быть достижимы).

Введем новую величину $\varepsilon_i > 0$ — *порог различимости.* Смысл ее состоит в следующем: при замене x_i на ε_i наблюдаемые величины могут значимо измениться. Естественно предполагать, что ε_i превосходит ошибку измерения x_i. Чтобы формализовать значимость изменения; поступим просто: за счет нормировок и безразмерных ограничений на параметры сеть работает с безразмерными величинами. Для них значимое изменение будем трактовать как изменение на величину порядка единицы. С учетом этого после нормировки

$$x_i := \frac{2x_i - (a_i + b_i)}{2\varepsilon_i}. \tag{2.27}$$

Такое преобразование может, однако, войти в противоречие с требованием нормировки входных сигналов. Здесь необходимо менять архитектуру или использовать специальную предобработку. Существует несколько приемов предобработки, позволяющих обойти описанную трудность. Все они заключаются в том, чтобы превратить одну координату x_i в несколько — *заменить число вектором.*

Пусть требуется уменьшить L_i в k раз (округленно возьмем k целым). Идея предобработки состоит в замене x_i некоторым числом функций (от k до $2k + 1$) этой переменной. С аналитической точки зрения простейшим является набор тригонометрических функций

$$\left\{\sin\frac{\pi l x_i}{2}, \cos\frac{\pi l x_i}{2}, l = 1, ..., k\right\}$$

(x_i предполагается уже центрированным и нормированным).

Два других варианта используют сигмоидные функции $\varphi(y)$ с единичной производной в нуле (например, $\varphi(y) = y/(1 + |y|)$ или $\varphi(y) = \text{th}\, y$) либо "шапочки" — производные сигмоидных функций (примеры шапочек: $\varphi(x) = 1/(1 + y^2)$ или $1/\text{ch}^2 y$). В обоих случаях число x_i преобразуется в вектор значений $2k + 1$ функций:

$$\left\{\varphi\left(\frac{1}{k}\left(x_i - \frac{l}{k}\right)\right)\right\}, l = 0, \pm 1, ..., \pm k. \tag{2.28}$$

(Как и выше предполагается, что нормировка и центрирование x_i уже проведены.)

Предостережение. Может возникнуть соблазн увеличить константу Липшица для функций, вычисляемых нейронной сетью, за счет увеличения крутизны функции активации нейрона. Конечно, при приближении $\varphi(y)$ к функции Хевисайда константа Липшица неограниченно возрастает, появляются скачки вычисляемой функции, но, увы, не там, где надо. Кроме того, ухудшаются интерполяционные и экстраполяционные возможности. Такое "улучшение" работы сети — к сожалению, самообман. Наша практика показывает, что для оценки характерного значения константы Липшица функций, реально вычисляемых сетью (а не гарантированной оценки сверху), целесообразно вместо максимума $|\varphi'(y)|$ подставлять в выражение для L_N среднее значение этой величины по интервалу изменения входных сигналов. Такое среднее значение для стандартных нейронов — величина порядка единицы, поэтому при оценке характерного значения L (не сверху, а именно характерного) для функций, вычисляемых сетью, рационально заменять константу Липшица нейрона L_N на константу Липшица сумматора L_Σ:

$$L_N \cong L_\Sigma = \| \alpha \|. \qquad (2.29)$$

Анализ выборочных констант Липшица и порогов разрешимости позволяет строить для решения задач именно такие сети, какие нужны. Они обладают достаточными, но не слишком избыточными возможностями.

Одна из важных функций предобработчика — обеспечение *инвариантности ответа сети относительно групп преобразования входных сигналов.* Конечно, можно учить сеть выдавать ответ, не меняющийся при заданных преобразованиях входов, но зачем усложнять задачу, когда наборы инвариантов для классических групп давно известны и достаточно организовать их вычисление перед подачей сигнала сети — с тем чтобы инвариантными были уже входные сигналы.

Задача инвариантной предобработки чаще всего возникает при анализе изображений. Продемонстрируем на примере этой области некоторые способы построения инвариантов.

Черно-белое или одноцветное полутоновое изображение представим вектором x, координаты которого — значения яркости в точках некоторой решетки.

Цветное изображение будем представлять набором векторов $x^1, \ldots, x^k$ для k базовых цветовых плоскостей.

Последовательно рассмотрим несколько групп преобразований, действующих на пространстве изображений.

Группа сдвигов: в отличие от поворотов, растяжений и проективных преобразований имеет достаточно богатую подгруппу, переводящую решетку в себя. Ограничимся поэтому такими сдвигами на векторы решетки. Формально будем считать решетку бесконечной,

изображение — бесконечномерным вектором с координатами x_r (r — вектор, принадлежащий решетке, только конечное число координат x_r отлично от нуля) или, для цветных изображений, конечным множеством таких векторов. Сдвиг на вектор d преобразует x_r в x_{r-d}. Обозначим это преобразование T_d. Обычное скалярное произведение в пространстве изображений инвариантно относительно сдвига:

$$(x, y) = \sum_r x_r y_r = \sum_r x_{r-d} y_{r-d} = (T_d x, T_d y). \qquad (2.30)$$

Вторая сумма получается из первой простым изменением нумерации.

Полиномиальные сдвиговые инварианты рассмотрим в порядке возрастания их степени.

Первая степень, одноцветное изображение x:

$$I^1(x) = \sum_r x_r \text{ (всего одно число)}.$$

Первая степень, цветное изображение, задаваемое векторами $x^1, \ldots, x^k$ — получаем k инвариантов первого порядка:

$$I^1(x^l) = \sum_r x^l_r \quad (l = 1, \ldots, k). \qquad (2.31)$$

Вторая степень, одноцветное изображение x, получаем вектор инвариантов второго порядка:

$$I^2_d(x, x) = (x, T_d x) = \sum_r x_r x_{r-d}. \qquad (2.32)$$

Вторая степень, цветное изображение $(x^1, \ldots, x^k)$, получаем набор векторных инвариантов второго порядка — по одному для каждой пары x^i, x^j:

$$I^2_d(x^i, x^j) = (x^i, T_d x^j) = \sum_r x^i_r x^j_{r-d}. \qquad (2.33)$$

Инварианты высших степеней уже не нужны.

Векторные инварианты второго порядка обладают следующим свойством симметрии:

$$I^2_d(x^i, x^j) = I^2_{-d}(x^j, x^i), \qquad (2.34)$$

поэтому таких различных инвариантов для k векторов $x^1, \ldots, x^k$ не более чем $k + C^2_k = k(k+1)/2$. За счет той же симметрии в инварианте для одного вектора не все координаты различны:

$$I^2_d(x, x) = I^2_{-d}(x, x), \qquad (2.35)$$

что тоже позволяет сократить описание.

Для использования более удобны другие инварианты (простые функции приведенных). Они соответствуют статистическому смыслу,

представляя собой математические ожидания, дисперсии и коэффициенты ковариации. Пусть N — число точек решетки в матрице, на которой рассматривается изображение. Положим:

$$\mathrm{M}(x) = \frac{1}{N}\sum_r x_r, \tag{2.36}$$

$$C_d(x, y) = \frac{1}{N}(x - \mathrm{M}(x), T_d y - \mathrm{M}(y)) =$$

$$= \frac{1}{N}\sum_r (x_r - \mathrm{M}(x))\ (y_{r-d} - \mathrm{M}(y)).$$

Набор инвариантов для одноцветного изображения x —

$$\mathrm{M}(x) \text{ и вектор } C_d(x, x);$$

для цветного изображения $(x^1, ..., x^k)$ —

$$\mathrm{M}(x^i) \text{ и набор векторов } C_d(x^i, x^j)\ (i, j = 1, ..., k;\ i \geqslant j).$$

Для других групп (общих преобразований движения, подобия и т.п.) при построении инвариантов также играет важную роль инвариантное скалярное произведение. Другая полезная идея — использование преобразования Фурье, связанного с группой. Для него инварианты второго порядка имеют особо простой вид. Например, для группы сдвигов преобразование Фурье инварианта $I_d^2(x, x)$ равно квадрату модуля преобразования Фурье самого изображения:

$$\mathrm{F}(I_d^2(x, x))\ (k) = |F(x)\ (k)|^2, \tag{2.37}$$

где F — преобразование Фурье.

При работе с любыми группами, кроме сдвигов, для заданных на решетке изображений возникает следующая сложность: решетка не является инвариантной ни относительно группы, ни относительно ее достаточно богатых подгрупп. Например, квадратная решетка выдерживает только 3 поворота (на $\pi/2$, π, $3\pi/2$). Поэтому приходится как-то доопределять действие преобразований на изображение. Простейший способ (по смыслу, но необязательно по реализации) состоит в том, что изображение преобразуется в некоторую функцию яркости уже на плоскости, а после преобразований происходит обратный переход — вычисляется след преобразованной функции на решетке.

Инварианты для группы движений могут вычисляться почти так же, как и для группы сдвигов. Заметим, что сдвиги плоскости образуют коммутативную нормальную подгруппу группы движений, а соответствующая фактор-группа — вращения вокруг одного центра. При движении x сдвиговые инварианты первого порядка $I^1(x)$ вновь сохраняются, а инварианты второго порядка $I_d^2(x, y)$ преобразуются как функ-

ции на плоскости (d — точка плоскости) соответствующим вращением из фактор-группы вокруг точки $d = 0$.

Построим инварианты второго порядка относительно вращений для произвольной интегрируемой с квадратом функции на плоскости $f(d)$. Для их описания перейдем к полярным координатам — опишем d парой (ρ, φ). Инвариант — снова функция ρ и угла φ:

$$J^2_{\rho,\varphi}(f, f) = \int_{R^2} f(\rho, \varphi) f(\rho, \varphi - \psi)\, d\psi. \tag{2.38}$$

Для набора функций $f^1, \ldots, f^q$ получаем систему инвариантов

$$J^2_{\rho,\psi}(f^i, f^j) = \int_{R^2} f^i(\rho, \varphi) f^j(\rho, \varphi - \psi)\, d\varphi. \tag{2.39}$$

— все почти точно так же, как и для сдвигов (за исключением сложностей, связанных с переходом от изображений на решетке к функциям на плоскости).

В группе преобразований подобия сдвиги также оставляют нормальную подгруппу, соответствующая фактор-группа образована суперпозициями вращений относительно нуля и гомотетий с центром в нуле. Общий вид такого преобразования в полярных координатах:

$$(\rho, \varphi) \to (\alpha\rho, \varphi + \psi). \tag{2.40}$$

В отличие от сдвигов и поворотов, сохраняющих обычную площадь, инвариантное интегрирование для гомотетий проводится с весом $1/\rho$:

$$\int_{R^2} \frac{f(\rho, \varphi)}{\rho}\, d\varphi\, d\rho. \tag{2.41}$$

Ввиду особенности в нуле, инварианты для любогй гладкой функции $f(\rho, \varphi)$ вычисляются так: инвариантно $f_0 = f(0, \varphi)$ (это число не зависит от φ), далее

$$K^1(f) = \int_{R^2} \frac{f(\rho, \varphi) - f_0}{\rho}\, d\varphi\, d\rho \tag{2.42}$$

и

$$K^2_{\alpha,\psi}(f, f) = \int_{R^2} \frac{1}{\rho}(f(\rho, \varphi) - f_0)\,(f(\alpha^{-1}\rho, \varphi - \psi) - f_0)\, d\varphi\, d\rho. \tag{2.43}$$

Аналогично для системы функций $f^1, \ldots, f^q$ получаем набор инвариантов:

$$f^i_0 = f^i(0, \varphi),$$

$$K^1(f^i) = \int_{R^2} \frac{f^i(\rho, \varphi) - f_0^i}{\rho} d\varphi\, d\rho, \qquad (2.44)$$

$$K^2_{\alpha,\psi}(f^i, f^j) = \int_{R^2} \frac{1}{\rho}(f^i(\rho, \varphi) - f_0^i)\ (f^j(\alpha^{-1}\rho, \varphi - \psi) - f_0^i)\, d\varphi\, d\rho.$$

Вновь подчеркнем, что наибольшую вычислительную сложность (по сравнению с группой сдвигов) вызывает необходимость переходить от функции на решетке к гладкой функции на плоскости и обратно.

Для вычисления инвариантов относительно общих преобразований подобия можно взять векторные инварианты второго порядка относительно сдвигов, преобразовать их в гладкие функции на плоскости, вычислить для них инварианты f_0, K^1, K^2 относительно поворотов гомотетий с центром в нуле. Полученные функциональные инварианты $K^2_{\alpha,\psi}$ можно вновь вернуть на решетку — для единообразного представления.

Использование связанного с группами гармонического анализа переводит свертки в произведения и может сократить вычисления. (Пример для сдвигов приведен выше.) Для более общих групп это также не вызывает особых затруднений.

Вычисление сдвиговых инвариантов второго порядка проводится по схеме: изображение сдвигается, накладывается на исходное, поточечно вычисляется произведение яркостей и результаты складываются (интегрируются). Можно предположить, что постоянные мелкие движения глаз осуществляют примерно такую предобработку изображений. Дополнительный аргумент в пользу предлагаемой интерпретации — животные, у которых нет таких движений, во многих случаях не различают неподвижных объектов, а движущийся объект сам по себе обеспечивает "вычисления" глазом некоторых сдвиговых инвариантов.

В а ж н ы й в о п р о с: инварианты второго порядка суть функции на плоскости, например, на той же решетке, что и изображения. Нужно ли вычислять и использовать все значения этих функций, или можно обойтись какой-то их частью? Очевидное сокращение описания (в силу симметрии) предполагается выполненным.

Ответ на этот вопрос дает эвристическое правило, также допускающее биологическое толкование и проверку: достаточно ограничиться значениями инвариантов в окрестности нуля, диаметр которой соответствует размерам значимых деталей изображения. Иными словами, надо использовать сдвиги, величина которых не превосходит линейных размеров характерных деталей.

Стохастическая инвариантная предобработка заключается в вычислении скалярных произведений изображения или его цветовых плоскостей на реализации инвариантных относительно нужной группы случайных полей. Важный класс таких полей реализуется с помощью

различных распределений случайных отрезков [173] — для каждого отрезка вычисляется суммарная яркость изображения на нем и эти значения складываются по всем отрезкам. Простота технической реализации и математическое изящество делают такой способ предобработки весьма привлекательным.

Частичная инвариантность используется тогда, когда группа преобразований слишком обширна и требуется, чтобы предобработанный сигнал не слишком сильно изменялся под действием преобразований из некоторой окрестности единицы этой группы. Стандартный пример — изменение изображений под действием группы диффеоморфизмов плоскости. Понятно, что диффеоморфизмы, близкие к тождественному преобразованию, не меняют смысла ни всего изображения, ни его деталей. Однако, если потребовать точной инвариантности предобработки относительно таких преобразований, то в результате получим инвариантность относительно всех диффеоморфизмов с положительным якобианом (т.е. относительно включающей тождественное преобразование связной компоненты всей группы). Это слишком сильное требование — заметим только, что любая перестановка конечного множества точек может быть получена в результате диффеоморфизма плоскости. Поэтому требование "не слишком сильных" изменений под действием преобразований из некоторой окрестности единицы нельзя заменить условием отсутствия изменений (т.е. точной инвариантности).

Пусть G — группа преобразований в пространстве векторов данных, $U \subset G$ — заданная окрестность единицы в G, P — преобразование предобработки, x — вектор данных, $\varepsilon > 0$. Тогда требование частичной инвариантности P относительно U на векторе x с точностью ε означает, что

$$\| P(x) - P(gx) \| < \varepsilon \tag{2.45}$$

для любого преобразования $g \in U$.

Искать частично инвариантные преобразования предобработки можно в следующем классе:

$$P(x) = \int_G \frac{gx}{\rho(g) + 1}\, dg. \tag{2.46}$$

В этой формуле $\rho(g)$ — мера близости ("расстояние", не обязательно удовлетворяющее аксиомам метрического пространства) элемента g к единичному, равная нулю в единице и большая за пределами U; dg — некоторая мера на группе. Произвол в выборе $\rho(g)$ и dg заставляет считать приведенное выражение для $P(x)$ всего лишь наводящим соображением.

Приведем пример частично инвариантной предобработки, построенный для важного случая, включающего обработку сигналов, изобра-

жений и др. и очень обширную группу G перестановок координат сигнала (точек изображения и т.п.).

Окрестность единицы в G определим с помощью отношения близости между координатами сигнала (точками изображения и т.п.). Пусть обрабатываемый вектор x имеет координаты x_i. Множество V номеров i конечно — иногда его удобно представлять как набор натуральных чисел, иногда — как некоторое множество точек решетки и т.п. Способ представления V далее неважен. Определим в V некоторое отношение соседства. Обозначим его $\sim$: $i \sim j$ будем читать как "i соседствует с j". Отношение $\sim$ симметрично, рефлексивно, но не транзитивно. Определение соседства зависит от смысла задачи. Для любых двух $i, j \in V$, $i \neq j$ построим расстояние между ними, порождаемое отношением соседства, $\rho(i, j)$ равно минимальному числу k, таких $i_1, \ldots, i_k \in V$, что

$$i = i_0 \sim i_1 \sim i_2 \sim \ldots \sim i_k = j;\ \rho(i, j) = k. \tag{2.47}$$

Зададим "окрестность" единицы U в группе G перестановок множества V следующим образом: $g \in U$ тогда и только тогда, когда $g(i) \sim i$ для всех $i \in V$.

Задачу частичной инвариантной относительно U предобработки решим с помощью специальных сглаживающих фильтров. Выберем некоторое натуральное число r — радиус фильтра в метрике ρ. Одновременно $\varepsilon = 1/r$ будет оценкой точности для частичной инвариантности результата предобработки. Для каждого $i \in V$ определим вектор $\Delta^i(r)$ с координатами

$$\Delta^i_j(r) = \begin{cases} 1 - \dfrac{\rho(i, j)}{r}, & \text{если } \rho(i, j) < r; \\ 0, & \text{если } \rho(i, j) \geqslant r. \end{cases} \tag{2.48}$$

Для любого вектора данных x зададим действие на нем линейного оператора предобработки P_r как

$$P_r(x)_j = \sum_{\substack{i \in V \\ \rho(i,j) < r}} \Delta^i_j(r)\, x_i. \tag{2.49}$$

За этой операцией следует нормировка (если необходимо).

Ядро фильтра $\Delta^i(r)$ есть результат применения P_r к единичному вектору δ^i ($\delta^i_j = 1$, если $i = j$, и 0, если $i \neq j$) и является ближайшим к δ^i и минимальным по норме вектором, для которого выполнено следующее требование полуинвариантности:

$$\left| (g\Delta^i)_j - \Delta^i_j \right| < 1/r \text{ для всех } j \text{ и } g \in U. \tag{2.50}$$

Выбор радиуса фильтра r может определяться из априорных соображений, связанных с задачей. Они здесь обсуждаться не могут, но есть и универсальный способ, основанный на заданных эталонных сигналах и предназначенный в первую очередь для решения задач классификации. Пусть задан набор эталонов $\{x^p\}$ (для задачи классификации предполагается, что они принадлежат разным классам). Предположим, что известна некоторая мера близости $D(x, y)$ в пространстве векторов данных и на основании ее значений может проводиться оценка близости (сходства) различных объектов. Такой мерой могут быть: эвклидово расстояние, коэффициент корреляции, расстояние Махаланобиса и т.п. Для определенности примем, что чем меньше D, тем ближе объекты, и $D(x, x) = 0$. Способ выбора r основывается на том, чтобы увеличивать его, пока для пар эталонных объектов (x^i, x^j) не начнут заметно, но не слишком сильно изменяться величины $D(x^i, x^j)$.

Для формализации этого "заметно, но не слишком сильно" вводится некоторая граница $\beta \in (0, 1)$ (например, $\beta = 0{,}5;\ 0{,}75;\ \ldots$). Напомним, что если необходимо, после сглаживания P_r проводится нормировка. Будем тогда обозначать $P_r(x)$ сглаженный и нормированный вектор x. В этом случае исходные эталоны предполагаются нормированными. Условие выбора r:

$$r = \max\left\{ r \,\middle|\, \min_{i,j,i\neq j} \frac{D(P_r(x^i), P_r(x^j))}{D(x^i, x^j)} > \beta \right\}. \qquad (2.51)$$

Естественно, описанный способ частично инвариантной предобработки — только один из возможных. В отличие от точной инвариантности, существует много способов формально поставить задачу о частичной инвариантности.

В заключение вновь обратим внимание на разнообразие функций и структур у предобработчиков. Многие из этих устройств могут быть описаны как специальные нейронные сети, структура и параметры которых конструируются еще до решения задачи. Мы не смогли описать все операции, выполняемые в ходе предварительной обработки — в рамках одного раздела или главы сделать это невозможно. Надеемся, однако, что удалось продемонстрировать разнообразие задач и методов, а также показать роль эвристических и правдоподобных рассуждений.

5.2. Интерпретация ответов сети

Казалось бы, нет ничего проще: получили нужный выходной сигнал, а проинтерпретировать его пользователь всегда сумеет. В каком-то смысле с этим можно согласиться — семейство интерпретаторов устроено гораздо проще и однообразнее, чем множество предобработчиков. Однако и при истолковании выходных сигналов требуется и аккуратность, и порой изобретательность — ведь от этого истолко-

вания зависят требования, которые мы предъявляем к работе нейронной сети. Удачная их формулировка может упростить обучение и повысить точность работы, неудачная — свести на нет предыдущие усилия.

Масштабирование — естественная операция при обработке выходных сигналов. Стандартные ("обезразмеренные") нейронные сети формируются так, чтобы их выходные сигналы лежали в интервалах $[-1, 1]$ (или $[0, 1]$). Если нам нужно получить сигнал в интервале $[a, b]$, то нужно преобразовать выходной сигнал y:

$$y \to \frac{a+b}{2} + \frac{b-a}{2} y. \tag{2.52}$$

Это простейшее преобразование неприменимо в том случае, когда отношение требуемой точности ε к разбросу $b - a$ много меньше характерной точности ответов нейронной сети. В таких случаях следует прибегать к преобразованию *"вектор-число"*. Напомним, что обратная задача встречалась нам при конструировании предобработчиков. Простейший способ улучшить незлонамеренную (случайную) относительную ошибку в k раз — сложить k^2 величин, а потом отмасштабировать сумму в нужный интервал $[a, b]$. Это фактически означает установку на выходе специального сумматора, который складывает и масштабирует k^2 выходных сигналов сети.

Другой вариант заменяет по существу вывод числа решением задачи классификации: весь отрезок значений $[a, b]$, в котором должен лежать обработанный ответ сети, разбивается на m частей (m выбирается достаточно большим, так, чтобы $(b - a)/m$ было величиной порядка требуемой точности). Этим m частям ставятся в соответствие, например, m нейронов. Выходные сигналы нейронов обрабатываются по правилу "победитель забирает все": если сигнал, принятый с i-го нейрона, больше, чем другие, то делаем вывод, что искомая величина лежит в i-й части отрезка $[a, b]$ (например, для разбиения на равные части она приближенно равна $a + (b - a)\frac{2i-1}{2m}$). Возможно использование других интерпретаторов классификации с меньшим числом нейронов, но существо дела сохраняется — переход от вопроса "чему равно число?" — к другому "к какому отрезку принадлежит точка?".

Большинство других известных вариантов носит гибридный характер: за каждым нейроном закрепляется его область на отрезке $[a, b]$, а выходные сигналы нейронов позволяют более точно локализовать точку в этой области (аналог предобработки с помощью "шапочек" или сигмоидных функций). Интересна чисто алгебраическая реализация идеи. Пусть функция одного переменного $f(x)$ обладает следующими свойствами: она определена на $[-1, 1]$, монотонно возрастает, выпукла вниз (т.е. $f''(x) > 0$), $f(-1) = 0$, $f(1) = 1$. Например, $f(x) =$

$= \left(\frac{x+1}{2}\right)^q$ $(q > 1)$ или $f(x) = 1 - \cos\frac{\pi(x+1)}{4}$ и т.п. Положим, что искомое число y получается из выходных сигналов нейронов $y_1, \ldots, y_k$ следующим образом:

$$y = y_1 + 1 + f(y_1)\,(y_2 + 1) + f(y_1)f(y_2)\,(y_3 + 1) + \ldots + \left(\prod_{i=1}^{k-1} f(y_i)\right)(y_k + 1), \qquad (2.53)$$

минимальное значение y равно нулю, максимальное — $2k$ и выходной сигнал i-го нейрона "работает" тогда, когда близки к единице выходные сигналы всех j-х при $j < i$.

В задачах классификации наиболее распространено уже многократно упоминавшееся правило интерпретации "победитель забирает все": число нейронов равно числу классов, номер нейрона с максимальным сигналом интерпретируется как номер класса. К сожалению, если классов много, то этот наглядный метод является слишком расточительным — потребляет слишком много выходных нейронов.

Знаковая интерпретация требует только $\log_2 m$ нейронов (m — число классов), строится она так. Пусть $y_1, \ldots, y_k$ — совокупность выходных сигналов нейронов. Заменим в этой последовательности положительные числа единицами, а отрицательные — нулями. Полученную последовательность нулей и единиц рассматриваем как номер класса в двоичной записи.

Порядковая интерпретация является еще более емкой, чем знаковая. В ней с помощью k нейронов можно описать принадлежность к $k!$ классам (а не 2^k, как для знаковой). Пусть $y_1, \ldots, y_k$ — выходные сигналы. Проведем их сортировку и обозначим n_i — номер i-го сигнала после сортировки (1 соответствует самому маленькому сигналу, k — самому большому). Перестановку

$$\sigma = \begin{pmatrix} 1 & 2 & \ldots & k \\ b_1 & n_2 & \ldots & n_k \end{pmatrix}$$

рассмотрим как слово, кодирующее номер класса. Всего возможно $k!$ перестановок — потенциальные возможности очень велики. Для того чтобы этим интерпретатором можно было пользоваться, необходимо, чтобы характерная ошибка выходного сигнала была меньше $1/k$. Даже при $k - 10$ получаем реализуемые требования к точности ($< 1/10$) и богатые возможности (10! классов).

Нечеткая интерпретация для классификаторов основывается на оценке возмущения выходных сигналов, меняющего принадлежность к классу. Она обычно настраивается над четкой интерпретацией. Для правила "победитель забирает все" выходные сигналы нейронов могут (после масштабирования — приведения значений в отрезок [0, 1])

рассматриваться как функции принадлежности к соответствующим классам. "Запас уверенности" определяется *разностью* Δ *максимального сигнала и следующего за ним по величине*. Пусть в ходе обучения к сети предъявлялось требование, чтобы на обучающей выборке Δ превосходило некоторое ε ($0 < \varepsilon < 2$, эта величина называется уровнем надежности, подробнее о ней и о требованиях к сети сказано ниже). Тогда степень уверенности в ответе можно оценить как Δ/ε (или 1, если $\Delta/\varepsilon > 1$). При $\Delta/\varepsilon \geqslant 1$ предъявленный объект отнесен к своему классу столь же уверенно, как и объекты из обучающей выборки.

Для знаковой интерпретации вектора выходных сигналов мерой уверенности в ответе является *наименьший* модуль сигнала. Вновь используем для нее обозначения Δ. Если при формировании сети к ней предъявлялось требование, что на обучающей выборке $\Delta \geqslant \varepsilon$, то вновь можно вводить степень уверенности в ответе: Δ/ε или 1 (если $\Delta/\varepsilon \geqslant 1$). Аналогично, для порядковой интерпретации в качестве Δ нужно рассматривать модуль разности *ближайших* по величине сигналов. Так же, исходя из требований, предъявляемых к этой величине на обучающей выборке ($\Delta \geqslant \varepsilon$), строится степень уверенности в отнесении объекта к данному классу.

Во всех случаях при неуверенном ответе (степень уверенности меньше 1) можно выделить множество классов, к которому объект уверенно отнесен. Пусть ε описывает уровень требований и $\Delta < \varepsilon$. Рассмотрим совокупность классов, которые могут быть получены при интерпретации сигналов, принадлежащих отрезкам $[y_i - \varepsilon, y_1 + \varepsilon]$ для знакового кодирования и $\left[y_i - \frac{\varepsilon}{2}, y_i + \frac{\varepsilon}{2}\right]$ для порядковой интерпретации и правила "победитель забирает все" (различие объясняется тем, что в первом случае интерпретируются значения сигналов, а в других — их разности). Положим, что сеть уверенно относит объект к этому множеству классов. Такая интерпретация дополняет четкое определение класса объектов. Назовем совокупность классов, построенную для данного набора выходных сигналов $y_1, \ldots, y_k$, *множеством уверенной классификации*. Обозначим его $K(y, \varepsilon)$ (y — вектор выходных сигналов, ε — уровень надежности, предъявляемый в виде требований при обучении сети). Для каждого $q \in K(y, \varepsilon)$ определим

$$\Delta(q) = \inf_{x \in q} \max_i |x_i - y_i|, \qquad (2.54)$$

где x — вектор с координатами $x_i \in [y_i - \varepsilon, y_i + \varepsilon]$ (для знаковой интерпретации) или $x_i \in \left[y_i - \frac{\varepsilon}{2}, y_i + \frac{\varepsilon}{2}\right]$ (для порядковой интерпретации и правила "победитель забирает все"); отношение $x \in q$ читается "вектор x интерпретируется (по соответствующему четкому правилу) как принадлежащий классу q"; $\Delta(q)$ оценивает, насколько надо

изменить выходные сигналы, чтобы попасть в класс q. Естественно, $\Delta(q) < \varepsilon$. Величина $\Delta(q)/\varepsilon$ оценивает степень уверенности в *непринадлежности* объекта классу q.

Итак, для каждого вектора выходных сигналов можно определить:

1) класс, к которому он может быть отнесен с наибольшей уверенностью;

2) степень уверенности в отнесении к этому классу;

3) множество уверенной классификации K — совокупность классов, к которой можно уверенно отнести объект;

4) степени уверенности в непринадлежности к каждому классу из K, кроме указанного в первом пункте.

Следующий пример интерпретатора касается только задачи коммивояжера, однако он типичен для широкого круга задач оптимизации. Пусть задано n городов, $n(n-1)/2$ расстояний между ними. Требуется объехать все города, побывать в каждом по одному разу и вернуться в исходный так, чтобы пройденный путь был минимальным. Если мы хотим решать эту задачу с помощью нейронной сети, то надо выбрать способ интерпретации выходных сигналов. Перенумеруем города. Каждому поставим в соответствие n нейронов (строку). Найдем наибольший из их выходных сигналов. Номер соответствующего нейрона интерпретируем как номер города, куда нужно ехать из данного. Понятно, что диагональные нейроны (i-й нейрон в i-й строке) не играют никакой роли — их можно не рассматривать. Итак, имеем $n(n-1)$ нейронов для n городов. Выходной сигнал каждого нейрона должен принадлежать отрезку $[0, 1]$.

5.3. Как оценивается работа сети?

Оценить работу сети необходимо в трех ситуациях.

1. Если мы собираемся для решения прикладной задачи сконструировать явными методами нейронную сеть, то удобно свести эту задачу к поиску минимума некоторой функции выходных сигналов (оценки). Тогда процесс функционирования сети конструируется как реализация какого-нибудь алгоритма оптимизации.

2. Если собираемся обучать сеть неявными оптимизационными методами, требуя, чтобы на обучающей выборке выходные сигналы сети соответствовали известным фактам.

3. Если требуется сравнить различные архитектуры, алгоритмы обучения и др. (т.е. для тестовых испытаний).

Для того чтобы построить функцию оценки, необходимо уже иметь интерпретатор выходных сигналов: если неизвестно, как интерпретируются результаты работы сети, то практически невозможно их оценить.

Примерами первого типа (оценка для явного контрастирования сети) будут служить задача коммивояжера и решение систем *линейных* уравнений.

Для задачи коммивояжера примем интерпретацию сигналов, описанную в конце предыдущего раздела. Пусть $y_{i1}, \ldots, y_{in}$ — сигналы, выдаваемые строкой нейронов, соответствующей i-му городу, а Y — вся матрица сигналов y_{ij}. Положим, что "идеальный" ответ имеет вид

$$Y_{ij} = \begin{cases} 1, & \text{если в оптимальном маршруте коммивояжер} \\ & \text{отправляется из } i\text{-го города в } j\text{-й,} \\ 0 - & \text{в остальных случаях.} \end{cases} \qquad (2.55)$$

Функцию оценки будем строить, набирая суммированием частные оценки, совместная минимизация которых обеспечит решение задачи. Первое слагаемое — H_l — оценивает длину маршрута:

$$H_l = \sum_{i,j} l_{ij}\, y_{ij}, \qquad (2.56)$$

где l_{ij} — расстояние между i-м и j-м городами.

Функция H_l дает точную длину маршрута, когда в любой строке матрицы y_{ij} один и только один элемент равен единице, остальные — нулю. Чтобы обеспечить это условие, вводится еще два слагаемых: $H_{0\vee 1}$ и H_1. Минимизация $H_{0\vee 1}$ приведет к тому, что каждый сигнал y_{ij} станет либо нулем, либо единицей:

$$H_{0\vee 1} = \sum_{i,j} y_{ij}^2 (1 - y_{ij})^2. \qquad (2.57)$$

Минимизация H_1 обращает в единицу сумму элементов каждой строки:

$$H_1 = \sum_i (1 - \sum_j y_{ij})^2. \qquad (2.58)$$

Множество общих точек минимума $H_{0\vee 1}$ и H_1 (и соответственно минимума их суммы) есть совокупность (y_{ij}), в которых все элементы равны нулю или единице и в каждой строке существует одна и только одна единица.

Минимизация взвешенной суммы $\alpha_l H_l + \alpha_{0\vee 1} H_{0\vee 1} + \alpha_1 H_1$ (все $\alpha \ldots > 0$) еще не дает требуемого результата — точкой минимума будет единичная матрица (один коммивояжер "расщепился" на n, каждый из которых сидит в своем городе и никуда не едет). От этого легко избавиться двумя способами: либо введя в оценку сумму диагональных элементов $\mathrm{sp}Y = \sum_i y_{ii}$, либо просто выбросив из рассмотрения диагональные нейроны. Выбираем первый путь, и вот почему: устранив циклы длины 1 ("новых коммивояжеров"), нам нужно будет устранить циклы длины 2, 3 и т.д. — чтобы остался только один цикл

длины n. При наличии циклов длины 2 появляются положительные (для 0—1 матриц единичные) элементы на диагонали матрицы Y^2, а циклы длины k приводят к появлению положительных элементов на диагонали y^k. Поэтому нужна еще одна компонента оценки, убивающая при оптимизации циклы длины, меньшей n, — H_c:

$$H_c = \sum_{k=1}^{n-1} \alpha_k \mathrm{sp} Y^k \quad (\alpha_k > 0). \tag{2.59}$$

Два выбора α_k являются "естественными":

1) $\alpha_k = 1$ при всех $k = 1, \ldots, n-1$;

2) $\alpha_k = C_{n-1}^k$, $H_c = \mathrm{sp}(1 + (y_{ij}))^{n-1} - n$.

Минимизация полной функции оценки

$$H = \alpha_l H_l + \alpha_{0 \vee 1} H_{0 \vee 1} + \alpha_1 H_1 + \alpha_c H_c, \tag{2.60}$$

если и не дает оптимального цикла, то, по крайней мере, часто приводит к хорошим приближениям. Наличие дополнительных подгоночных параметров (на самом деле трех, так как один из них можно приравнять к 1, например α_l), с одной стороны, повышает гибкость, предоставляя пользователю дополнительные рукоятки настройки, с другой — усложняя работу, надо думать, как эти рукоятки крутить.

Интерпретация выходных сигналов с помощью максимального элемента позволяет модифицировать H_l так, чтобы она давала лучшее приближение для длины маршрута и тогда, когда выходная матрица еще не вполне соответствует контрольным условиям (элементы — нули или единицы, в каждой строке — только одна единица). Эту модификацию можно провести с сохранением гладкости, для чего положим

$$y'_{ij} = \frac{y_{ij}^q}{\left(\sum_j y_{ij}^p\right)^{q/p}}. \tag{2.61}$$

С ростом степеней p, q матрица (y'_{ij}) приближается к 0—1 матрице перестановок, соответствующей интерпретации выходных сигналов y_{ij}. Оценку длины маршрута H_l определим теперь как $\sum_{i,j} l_{ij} y'_{ij}$.

Сложность вычисления оценки H (порядка n^4 операций при стандартном подходе) не должна нас смущать, так как для решения задачи коммивояжера (точно и "в лоб") требуется перебор $n!$ вариантов.

Для решения системы линейных уравнений

$$Ax = b \tag{2.62}$$

простейшую оценку дает метод наименьших квадратов (МНК):

$$H = \| Ax - b \|^2, \tag{2.63}$$

где $\|\cdot\|$ — эвклидова норма. Использование других норм возможно, например, в l_p норме:

$$H = \|Ax - b\|^p = \sum_i (Ax - b)_i^p. \tag{2.64}$$

Также представляют интерес энтропийная оценка

$$H = \sum_i (Ax - b)_i \ln|(Ax)_i / b_i| \tag{2.65}$$

и оценка МНК "с люфтом" ε:

$$H = \sum_i P\left(\frac{(Ax - b)_i}{\varepsilon}\right), \tag{2.66}$$

где

$$P(\Delta) = \begin{cases} (|\Delta| - 1)^2, & \text{если } |\Delta| \geqslant 1, \\ 0, & \text{если } |\Delta| < 1. \end{cases} \tag{2.67}$$

Оценки с люфтом ε позволяют прекращать оптимизацию, если достигнута заданная величина невязок.

Для обучения сети на известных примерах необходимо, чтобы требования к ней (оценка) соответствовали тому результату, который мы хотим получить (интерпретатор). Несмотря на это, казалось бы, очевидное соображение, наиболее популярна оценка МНК как суммы квадратов ошибок. Ошибки определяются как отклонения полученных сигналов от идеальных требуемых. Оценка МНК имеет свою естественную область применимости: задачи, в которых выходные сигналы сети должны с заданной и одинаковой для всех сигналов точностью ε совпадать с известными векторами. Однако и в этом случае лучше использовать другую оценку — МНК с люфтом ε. Если требуемая точность ε_i для i-й компоненты вектора выходных сигналов зависит от i, то используется МНК с весами

$$H = \sum_i \left(\frac{\Delta_i}{\varepsilon_i}\right)^2 \tag{2.68}$$

или МНК с люфтом, зависящим от номера сигнала,

$$H = \sum_i P\left(\frac{\Delta_i}{\varepsilon_i}\right), \tag{2.69}$$

где $\Delta_i = y_i - \tilde{y}_i$ — отклонение i-й компоненты ответа сети y_i от требуемого значения $\tilde{y}_i$, а

$$P(\Delta) = \begin{cases} (|\Delta| - 1)^2, & \text{если } |\Delta| \geqslant 1, \\ 0, & \text{если } |\Delta| < 1. \end{cases} \tag{2.70}$$

Для задач классификации правильный ответ задается не единственным образом, а как множество, определяемое интерпретатором. Так, если выбран интерпретатор "победитель забирает все", то для примера i-го класса множество правильных ответов образует многогранный конус в пространстве выходных векторов. Этот конус определяется системой неравенств:

$$y_i > y_j \text{ для всех } j \neq i \ (y_i \text{ — сигнал-"победитель"}). \qquad (2.71)$$

Для знаковой интерпретации множество правильных ответов — тоже многогранный конус. Описывается он так. Пусть $\sigma_i = \pm 1$ определяет требуемый знак y_i. Тогда правильные ответы задаются неравенствами

$$\sigma_i y_i > 0. \qquad (2.72)$$

Наконец, пусть для порядковой интерпретации требуемый порядок следования выходных сигналов на прямой задается подстановкой $\pi(i)$:

$$\text{в}y_{\pi(1)} > y_{\pi(2)} > \ldots > y_{\pi(k)}. \qquad (2.73)$$

Эта система неравенств и определяет множество правильных ответов.

Точным аналогом оценки МНК для задач классификации служит такая оценка: квадрат расстояния от вектора выходных сигналов до множества правильных ответов —

$$H(y) = [\text{dist}(y, M)]^2, \qquad (2.74)$$

где y — вектор выходных сигналов; M — множество правильных (т.е. правильно интерпретируемых) ответов; $\text{dist}(y, M) = \inf_{z \in M} \|y - z\|^2$; $\|\cdot\|$ — эвклидова норма.

Использование расстояния до M неудобно в одном отношении: для любого $\varepsilon > 0$ в M присутствуют векторы, интерпретация которых меняется при ε-малом возмущении — изменении координат на величину, меньшую ε. Чтобы избежать этой неустойчивости интерпретации, вводится M_ε — множество правильных ответов, интерпретация которых не меняется при изменении координат на величину, меньшую ε:

$$M_\varepsilon = \{y | y + \Delta \in M, \text{ если } |\Delta_i| < \varepsilon \text{ при всех } i\}. \qquad (2.75)$$

Аналогом МНК с люфтом служит оценка

$$H_\varepsilon(y) = [\text{dist}(y, M_\varepsilon)]^2. \qquad (2.76)$$

Величина ε называется *уровнем надежности*. Для знаковой интерпретации $0 \leqslant \varepsilon \leqslant 1$, для правила "победитель забирает все" $0 \leqslant \varepsilon \leqslant 2$, для порядковой интерпретации $0 \leqslant \varepsilon \leqslant 2/(k-1)$, где k — размерность вектора выходных сигналов.

Вычисление расстояния от точки y до многогранных конусов M_ε не вызывает принципиальных затруднений, а некоторое усложнение (по сравнению с расстоянием от точки до точки) является небольшой платой за следующие приобретения:

1) ускорение обучения (на стандартных тестовых примерах — в 10 раз и более);

2) дополнительные "педагогические" возможности, связанные с изменением уровня надежности ε в ходе обучения;

3) разнообразные варианты нечеткой интерпретации ответов: если при обучении использовалась оценка H_ε, то принадлежность вектора выходных сигналов y множеству M_ε интерпретируется как уверенный ответ (степень уверенности 1), если же $y \in M_\delta$ только при некоторых $\delta < \varepsilon$, то вводится "степень уверенности", равная Δ/ε, где

$$\Delta = \sup\{\delta > 0 \,|\, y \in M_\delta\}. \tag{2.77}$$

(Подробнее об этом см. в предыдущем разделе.)

До сих пор мы рассматривали только оценки вектора выходных сигналов для одного примера. При обучении же обычно требуется оценить работу сети на некотором множестве примеров. Для этого из оценок отдельных из них $H(y^p)$ (p — номер примера) формируется оценка H всего множества. Простейший вариант — суммирование:

$$H = \sum_p H(y^p). \tag{2.78}$$

Ближайшее обобщение — суммирование с весами:

$$H = \sum_p w_p H(y^p), \tag{2.79}$$

где $w_p \geqslant 0$ — вес p-го примера.

Суммирование с весами используется в следующих случаях:

1) для выделения наиболее важных примеров (их вес увеличивают);

2) для уравновешивания различных групп примеров (в задачах классификации полезно сразу назначать веса таким образом, чтобы суммарный вес обучающих примеров в каждом классе не зависел от класса, скажем, $w_p = 1/n_i$, для любого примера i-го класса, где n_i — число таких примеров);

3) для выделения наиболее трудных примеров (их вес увеличивают, чтобы повысить их вклад в процесс обучения сети).

Существуют случаи, когда невозможна оценка одного вектора выходных сигналов, но только некоторой их последовательности. Так, при подготовке нейросетевого датчика случайных чисел (векторов) можно оценить только достаточно длинные последовательности выходных сигналов, сравнивая их моменты, корреляционные характеристики и т.п. с теми, которые нужно получить.

Универсальный путь обучения. Существует универсальный способ обучения нейронных сетей — минимизация оценки H как неявно (т.е. через функционирование) заданной функции параметров сети. В подавляющем большинстве случаев при реализации этого подхода предполагается, что задана обучающая выборка, состоящая из векторов входных сигналов x^p, известные требования к соответствующим выходным сигналам y^p зафиксированы в функции оценки $H(y^p)$, а оценка H по всей выборке или какой-либо ее части строится известным способом по значениям $H(y^p)$. Возможны варианты: задается не обучающая выборка, а способ ее получения, нет оценки отдельного примера, а только их совокупности (см. конец предыдущего раздела).

В любом случае после окончания предварительной работы, предшествующей обучению, имеем способ вычисления некоторой функции H, минимизация которой как функции параметров настроит сеть для правильной работы. При минимизации, использующей производные H, существуют две задачи: вычисление производных и собственно минимизация. Им посвящены две следующие главы.

За универсальность и экономичность обученных сетей приходится, однако, платить временем обучения. Поэтому, когда возможно, прибегают к более простым и явным способам формирования сети. Иногда такие явные способы вполне соответствуют сути дела.

Здесь, следуя за списком задач из разд. 2.4, опишем настройку сетей для их решения. В тех случаях, когда существуют удачные специальные методы настройки, изложим их подробно. Для универсального оптимизационного способа будут строиться соответствующие функции оценки и архитектура сетей.

Метод динамических ядер в классификации без учителя. Сетевые алгоритмы классификации без учителя строятся на основе *метода динамических ядер* [174, 206]. Опишем его сначала в наиболее общей абстрактной форме. Пусть задана выборка предобработанных векторов данных $\{x^p\}$. Пространство векторов данных обозначим E. Каждому классу будет соответствовать некоторое ядро a. Пространство ядер обозначим A. Для каждых $x \in E$ и $a \in A$ определяем меру близости $d(x, a)$, для каждого набора из k ядер $a_1, \ldots, a_k$ и любого разбиения $\{x^p\}$ на k классов $\{x^p\} = P_1 \cup P_2 \cup \ldots \cup P_k$ — *критерий качества*:

$$D = D(a_1, \ldots, a_k, P_1, \ldots, P_k) = \sum_{i=1}^{k} \left(\sum_{x \in P_i} d(x, a_i) \right). \qquad (2.80)$$

Требуется найти набор $a_1, \ldots, a_k$ и разбиение $P_1, \ldots, P_k$, минимизирующие D.

Шаг алгоритма разбивается на два этапа:

1-й этап — для фиксированного набора ядер $a_1, \ldots, a_k$ ищем минимизирующее критерий качества разбиение $P_1, \ldots, P_k$; оно дается

решающим правилом: $x^p \in P_i$, если $d(x^p, a_i) < d(x^p, a_j)$ при $j \neq i$, когда для x^p минимум $d(x^p, a_i)$ достигается при нескольких значениях i, выбор между ними может быть сделан произвольно;

2-й этап — для каждого P_i $(i = 1, ..., k)$, полученного на 1-м этапе, ищется $a_i \in A$, минимизирующее критерий качества (т.е. слагаемое в D для данного i – $D_i = \sum_{x \in P_i} d(x, a_i)$).

Начальные значения $a_1, ..., a_k$, $P_1, ..., P_k$ выбираются произвольно, либо по какому-нибудь эвристическому правилу. На каждом шаге и этапе алгоритма уменьшается критерий качества D, отсюда для большинства конкретизаций задачи следует сходимость алгоритма — после конечного числа шагов разбиение $P_1, ..., P_k$ уже не меняется.

Если ядру a_i ставится в соответствие элемент сети, вычисляющий по входному сигналу x функцию $d(x, a_i)$, то решающее правило для классификации дается интерпретатором "проигравший забирает все": элемент x принадлежит классу ε_i, если выходной сигнал i-го элемента $d(x, a_i)$ меньше всех остальных (практически все тот же интерпретатор "победитель забирает все", только со сменой знака).

Единственная вычислительная сложность в алгоритме может состоять в поиске ядра по классу на 2-м этапе алгоритма, т.е. в поиске $a \in A$, минимизирующего

$$D_i = \sum_{x \in P_i} d(x, a). \tag{2.81}$$

В связи с этим в большинстве конкретных реализаций метода мера близости d выбирается такой, чтобы легко можно было найти a, минимизирующее D_i для данного P_i.

В простейшем случае пространство ядер A совпадает с E, а мера близости $d(x, a)$ — положительно определенная квадратичная форма от $x - a$, например, квадрат эвклидового расстояния или метрики Махаланобиса. Тогда ядро a_i, минимизирующее D_i, есть центр тяжести класса P_i:

$$a_i = \frac{1}{|P_i|} \sum_{x \in P_i} x \quad , \tag{2.82}$$

где $|P_i|$ — число элементов в P_i.

В этом случае также упрощается решающее правило, разделяющее классы. Обозначим $d(x, a) = (x - a, x - a)$, где $(\cdot\ , \cdot)$ — билинейная форма (если d — квадрат эвклидового расстояния между x и a, то $(\cdot\ , \cdot)$ — обычное скалярное произведение). В силу билинейности

$$d(x, a) = (x - a, x - a) = (x, x) - 2(x, a) + (a, a). \tag{2.83}$$

Чтобы сравнить $d(x, a_i)$ для разных i и найти среди них минимальное, достаточно вычислить линейную неоднородную функцию от x:

$$d_1(x, a_i) = (a_i, a_i) - 2(x, a_i). \tag{2.84}$$

Минимальное значение $d(x, a_i)$ достигается при том же i, что и минимум $d_1(x, a_i)$, поэтому решающее правило реализуется с помощью k сумматоров, вычисляющих $d_1(x, a_i)$ и интерпретатор, выбирающий сумматор с минимальным выходным сигналом. Номер этого сумматора и есть номер класса, к которому относится x.

Пусть теперь мера близости — коэффициент корреляции между вектором данных и ядром класса:

$$r(x, a) = \sum_j \frac{x_j - \mathrm{M}(x)}{\sigma_x} \times \frac{a_j - \mathrm{M}(a)}{\sigma_a}, \tag{2.85}$$

где x_j, a_j — координаты векторов; $\mathrm{M}(x) = \frac{1}{n}\sum_j x_j$; n — размерность пространства данных; $\sigma_x = \left(\frac{1}{n}\sum_j (x_j - \mathrm{M}(x))^2\right)^{1/2}$. Предполагается, что предобработка состоит в преобразовании

$$x_j \to \frac{x_j - \mathrm{M}(x)}{\sigma_x}, \tag{2.86}$$

поэтому для всех x, поступающих сети, $\mathrm{M}(x) = 0$, $\sigma_x = 1$; аналогично $\mathrm{M}(a) = 0$, $\sigma_a = 1$ и $r(x, a) = (x, a)$, где $(\cdot, \cdot)$ — обычное скалярное произведение.

Заметим также, что вместо минимизации во всех операциях метода динамических ядер для этой меры близости надо брать максимум, так как, чем ближе x и a, тем больше $r(x, a)$.

Задача поиска ядра a_i для заданного класса P_i превращается в поиск вектора a, максимизирующего

$$D_i = \sum_{x \in P_i} (x, a) \tag{2.87}$$

при ограничениях $\sum_j a_j = 0$, $\sum_j a_j^2 = 1$. Этот максимум достигается в точке

$$a = \frac{\sum_{x \in P_i} x}{\left\| \sum_{x \in P_i} x \right\|}, \tag{2.88}$$

где $\| \cdot \|$ — эвклидова норма, почти то же самое, что и для эвклидовой нормы, отличие по существу только в предобработке.

Если мера близости определяется как квадрат коэффициента корреляции (т.е. важным считается наличие линейной корреляционной связи, а не ее знак), то

$$D_i(a) = \sum_{x \in P_i} (x, a)^2, \tag{2.89}$$

$D_i(a)$ — квадратичная форма от a. Она вырождена, так как для вектора e, у которого все координаты $e_i = 1$, имеем $D_i(e) = 0$. Чтобы найти максимум $D_i(a)$ при указанных ограничениях ($M(a) = 0$, $\sigma_a = 1$), достаточно отыскать наибольшее собственное число λ и соответствующий собственный вектор a_i матрицы этой квадратичной формы с точностью до множителя $|P_i|$ (ковариационной матрицы класса):

$$|P_i| \Sigma_i = \sum_{x \in P_i} x \otimes x^{\mathrm{T}}. \tag{2.90}$$

Этот вектор a_1 и будет искомым ядром класса P_1.

В очередной раз возникает задача спектрального анализа ковариационной матрицы. Всегда существует вопрос: включать ли в арсенал основных средств работы решение этой задачи, либо различных ее вариантов и подзадач (как здесь — определение собственного вектора, соответствующего наибольшему собственному числу). Существуют итерационные методы, позволяющие работать с ковариационной матрицей, перебирая примеры "по одному" (см. гл. 1) и реализуемые на нейронных сетях.

В тех простейших случаях, когда ядро класса точно определяется как *среднее арифметическое* (или нормированное среднее арифметическое) элементов класса, а решающее правило основано на сравнении выходных сигналов линейных адаптивных сумматоров, нейронную сеть, реализующую метод динамических ядер, называют *сетью Кохонена*. В определение ядра a_i для сетей Кохонена входят суммы $\sum_{x \in P_i} x$. Это позволяет накапливать новые динамические ядра, обрабатывая по одному примеру и пересчитывая a_i после появления в P_i нового примера. Сходимость при такой модификации, однако, ухудшается.

Прежде чем перейти к рассмотрению тех случаев, когда $E \neq A$ (ядра классов лежат в другом пространстве), остановимся на различных способах использования полученных классификаторов.

1. Базовый способ. Для вектора данных x и каждого ядра a_i вычисляется $y_i = d(x, a_i)$ (или для квадратичных форм d — линейная функция $y_i = -d_1(x, a_i)$); по правилу "победитель забирает все" строка ответов y_i преобразуется в строку, где только один элемент, соответствующий максимальному y_i, равен 1, остальные — нули. Эта

строка и является результатом функционирования сети. По ней могут быть определены номер класса (номер места, на котором стоит *1*) и другие показатели.

2. Метод аккредитации. За слоем элементов базового метода, выдающих сигналы 0 и 1 по правилу "победитель забирает все" (далее называем его слоем базового интерпретатора), надстраивается еще один слой выходных сумматоров. С каждым (i-м) классом ассоциируется q-мерный выходной вектор z^i с координатами z^i_j. Он может формироваться по-разному: от двоичного представления номера класса до вектора ядра класса. Вес связи, ведущей от i-го элемента слоя базового интерпретатора к j-му выходному сумматору, определяется точно так же, как z^i_j. Если на этом i-м элементе базового интерпретатора получен сигнал 1, а на остальных 0, то на выходных сумматорах будут получены числа z^i_j.

3. Нечеткая классификация. Пусть вектор данных x обработан слоем элементов, вычисляющих $y_i = d(x, a_i)$ (для квадратичных форм $y_i = -d_1(x, a_i)$). Идея дальнейшей обработки состоит в том, чтобы выбрать из этого набора $\{y_i\}$ несколько самых больших чисел и после нормировки объявить их значениями функций принадлежности к соответствующим классам. Предполагается, что к остальным классам объект наверняка не принадлежит. Для выбора семейства G наибольших y_i определим следующие числа:

$$y_{\max} = \max\{y_i\},\ \mathrm{M}(y) = \frac{1}{k}\sum_i y_i,\ s = (1-\alpha)\mathrm{M}(y) + \alpha y_{\max}, \quad (2.91)$$

где число α характеризует отклонение "уровня среза" s от среднего значения $\mathrm{M}(y)$, $\alpha \in [-1, 1]$, по умолчанию обычно принимается $\alpha = 0$. Полагаем $G = \{y_i | y_i > s\}$.

Множество $J = \{i | y_i \in G\}$ трактуется как совокупность номеров тех классов, к которым может принадлежать объект, а нормированные на единичную сумму неотрицательные величины

$$f_i = \frac{y_i - s}{\sum\limits_{j \in J}(y_j - s)} \text{ (при } i \in J \text{ и } f_i = 0 \text{ в противном случае)} \quad (2.92)$$

интерпретируются как значения функций принадлежности этим классам.

4. Метод интерполяции. Надстраивается над нечеткой классификацией аналогично тому, как метод аккредитации связан с базовым способом. С каждым классом связывается q-мерный выходной вектор z_i. Строится слой из q выходных сумматоров, каждый из которых должен выдавать свою компоненту выходного вектора. Весовые коэффициенты связей, ведущих от того элемента нечеткого классифика-

тора, который вычисляет f_i, к j-му выходному сумматору, определяются как z_j^i. В итоге вектор выходных сигналов сети есть

$$z = \sum_i f_i z^i. \tag{2.93}$$

В отдельных случаях по смыслу задачи требуется нормировка f_i на единичную сумму квадратов:

$$f_i = \frac{y_i - s}{\left(\sum_{j \in J} (y_j - s)^2\right)^{1/2}} \text{ (при } i \in J \text{ и } f_i = 0 \text{ в противном случае)}. \tag{2.94}$$

Один из описанных четырех вариантов использования сети (или какой-нибудь другой) выбирается в соответствии с нуждами пользователя. Предлагаемые четыре способа покрывают большую часть потребностей.

Задача мозаичной регрессии является по существу обобщением одного из вариантов только что рассмотренной задачи классификации методом динамических ядер, в котором мера близости — квадрат коэффициента корреляции.

Итак, пусть пространство ядер A_r состоит из всех r-мерных линейных многообразий в n-мерном пространстве E векторов данных, а мера близости $d(x, a)$ определяется как квадрат эвклидового расстояния от точки x до многообразия a. По данному множеству примеров P_i построим a_i, минимизирующее

$$D_i = \sum_{x \in P_i} d(x, a). \tag{2.95}$$

Решение строится так. Определим

$$M_i = \frac{1}{|P_i|} \sum_{x \in P_i} x. \tag{2.96}$$

Построим ковариационную матрицу множества данных P_i:

$$\Sigma_i = \frac{1}{|P_i|} \sum_{x \in P_i} (x - M_i) \otimes (x - M_i)^{\mathrm{T}}. \tag{2.97}$$

Пусть λ_j^i ($\lambda_1^i \geqslant \lambda_2^1 \geqslant \ldots \geqslant \lambda_n^i$) — собственные числа Σ_i, взятые в порядке убывания, а e^{ij} — соответствующие λ_j^i ортонормированные собственные векторы. Если первые r собственных чисел — ненулевые и $\lambda_r^i \neq \lambda_{r+1}^i$, то искомое a_i — единственно и оно задается с помощью набора из r параметров (внутренних координат) $\gamma_1, \ldots, \gamma_y$:

$$a_i = \{M_i + \sum_{j=1}^{r} \gamma_j e^{ij}\}. \tag{2.98}$$

Для такого a_i и произвольного $x \in E$

$$d(x, a_i) = \|x - M_i\|^2 - \sum_{j=1}^{r} (x - M_i, e^{ij})^2 = \sum_{j=r+1}^{n} (x - M_i, e^{ij})^2. \tag{2.99}$$

В тех случаях, когда решение неединственно, обозначим A_i множество тех $a \in A$, для которых D_i минимально. Ядром класса P_i будет уже не одно линейное многообразие a_i, а все множество A_i, а для произвольного $x \in E$ положим

$$\begin{aligned} d_{\min}(x, A_i) &= \min_{a \in A_i} d(x, a), \\ d_{\max}(x, A_i) &= \max_{a \in A_i} d(x, a). \end{aligned} \tag{2.100}$$

Объединение всех a из A_i снова представляет собой линейное многообразие вида $M_i + L_i$, где L_i — линейное пространство с базисом $\{e^{ij} \mid \lambda_j^i \geqslant \lambda_r^i\}$, поэтому для $d_{\min}(x, A_i)$ получаем практически прежнюю формулу:

$$d_{\min}(x, A_i) = \sum_{\lambda_j^i < \lambda_r^i} (x - M_i, e^{ij})^2. \tag{2.101}$$

Аналогично, $d_{\max}$ находится как расстояние от x до пересечения всех $a \in A_i$:

$$d_{\max}(x, A_i) = \sum_{\lambda_j^i \leqslant \lambda_r^i} (x - M_i, e^{ij})^2 \tag{2.102}$$

(отличие от предыдущей формулы только в перемене знака $<$ на $\leqslant$).

Итерационный процесс построения классов P_i и их ядер a_i требует доопределения $d(x, a_i)$ на случай неединственного решения. Предлагается использовать $d_{\min}(x, A_i)$. Выбор мотивируется так: в результате присоединения x к P_i величина D_i возрастает более чем на $d_{\min}(x, A_i)$. Если для нескольких P_i значения $d_{\min}(x, A_i)$ совпадают, то выбор класса, к которому относится x, при обучении происходит произвольно.

Процесс сходится за конечное число шагов — получаем классы P_i и их ядра a_i (или для вырожденного случая множества ядер A_i).

Полученный классификатор в невырожденном случае используется следующим образом. Пусть x — вектор данных. Для каждого (i-го) класса вычисляются:

1) вектор $x - M_i$, квадрат его эвклидовой нормы $\|x - M_i\|^2$;

2) набор скалярных произведений $(x - M_i, e^{ij})$ $(j = 1, ..., r)$;

3) вектор $x'^i = M_i + \sum_{j=1}^{r} (x - M_i, e^{ij}) e^{ij}$;

4) величина $d(x, a_i) = \|x - M_i\|^2 - \sum_{j=1}^{r} (x - M_i, e^{ij})^2$.

Пусть среди чисел $d(x, a_i)$ минимальное $d(x, a_{i_0})$. Вектором выходных сигналов сети является x'^{i_0}.

Пусть x — вектор неполных данных, в котором неизвестно l координат (не более чем $n - r - 1$, т.е. $l + r < n$). Обозначим множество номеров неизвестных координат I. С данным I связана операция, переводящая любой n-мерный вектор u в $n - l$-мерный, получаемый из u вычеркиванием l координат u_i $(i \in I)$. Обозначим эту операцию P_I. Для векторов неполных данных x с множеством номеров неизвестных координат I определим

$$d(x, a_i) = d(P_I x, P_I a_i). \tag{2.103}$$

Геометрический смысл определения: с вектором неполных данных x связано линейное многообразие a_x, у элементов которого $y \in a_x$ координаты $y_i = x_i$ при $i \notin I$ и y_i — произвольные числа при $i \in I$ (многообразие a_x задается в E уравнением $P_I y = P_I x$);

$$d(x, a_i) = \operatorname{dist}(a_x, a_i)^2 = \inf_{y \in a_x, z \in a_i} \|y - z\|^2 \tag{2.104}$$

— квадрат минимального расстояния между точками $y \in a_x$, $z \in a_i$. Точки $y_{\min}$, $z_{\min}$, в которых достигается нижняя грань, существуют и играют особую роль в обработке неполных данных. Для каждого класса определены операции *восстановления* данных: $x \to x' = y_{\min}$ и исправления данных: $x \to x'' = z_{\min}$. При восстановлении неизвестные данные заменяются координатами ближайшей к a_x точки из a_i, при исправлении весь вектор x заменяется на ближайший к a_x вектор из a_i. Если многообразия a_x, a_i имеют параллельные прямые, то векторы $y_{\min}$, $z_{\min}$ неединственны и задачи восстановления и исправления данных x не имеют однозначного решения.

Линейное многообразие $P_I a_i$ имеет вид

$$P_I M_i + P_I L_i, \tag{2.105}$$

где L_i — линейная оболочка векторов $e^{i1}, ..., e^{ir}$. Проекции этих векторов $P_I e^{ij}$ $(j = 1, ..., r)$ порождают линейное пространство $P_I L_i$. В ситуации общего положения семейство векторов $\{P_I e^{ij}\}$ $(j = 1, ..., r)$ линейно независимо и образует базис в $P_I L_i$. Для определения

$d(x, a_i)$ при неполных данных достаточно найти ортогональную проекцию $P_I x - P_I M_i$ на $P_I L_i$. Обозначим ее $Pr_{P_I L_i}(P_I x - P_I M_i)$. Тогда

$$d(x, a_i) = d(P_I x, P_I a_i) = \|P_I x - P_I M_i\|^2 - \\ - \|Pr_{P_I L_i}(P_I x - P_I M_i)\|^2. \qquad (2.106)$$

Если векторы $P_i e^{ij}$ $(j = 1, ..., r)$ линейно независимы (ситуация общего положения при $r + l < n$), то любой вектор $y \in P_I L_i$ имеет при отображении P_I единственный прообраз в L_i. Этот прообраз легко определить, зная разложение y по $P_I e^{ij}$ $(j = 1, ..., r)$:

если $$y = \sum_{j=1}^{r} y_j P_I e^{ij}, \text{ то } P_{Ii}^{-1} y = \sum_{j=1}^{r} y_j e^{ij}. \qquad (2.107)$$

Таким образом, для действия P_I на L_i определено обратное отображение, которое будем обозначать P_{Ii}^{-1} (ссылка на номер i обязательна: для разных i P_{Ii}^{-1} — различные отображения пространства $P_I L_i$ в E с образом L_i).

Цель наших построений — вектор

$$x''^i = M_i + P_{Ii}^{-1}(Pr_{P_I L_i}(P_I x - P_I M_i)) \qquad (2.108)$$

— представляет собой результат исправления и дополнения вектора данных x с помощью регрессионного закона i-го класса. Геометрически x''^i является точкой многообразия a_i, ближайшей к линейному многообразию — прообразу $P_I x$ при отображении P_I. Если же нас интересует только наилучшее восстановление неизвестных данных, а не исправление известных, то вместо x''^i следует рассматривать x'''^i:

$$x'''^i_j = \begin{cases} x'_j, & \text{если } j\text{-я координата данных } x_j \text{ известна;} \\ x'_j i, & \text{если } j\text{-я координата данных } x_j \text{ неизвестна.} \end{cases}$$

Итак, чтобы восстановить неизвестные данные и/или исправить известные при неполной информации, надо вычислить для каждого i величину $d(x, a_i)$, выбрать из них наименьшую и для соответствующего i найти x''^i. Достаточно отыскать векторы

$$v_i(x) = Pr_{P_I L_i}(P_I x - P_I M_i) \qquad (2.109)$$

и для i, при котором минимально $d(x, a_i)$, найти прообразы $v_i(x)$ в L_i:

$$w_i(x) = P_{Ii}^{-1}(Pr_{P_I L_i}(P_I x - P_I M_i)) \qquad (2.110)$$

(тривиальные операции — прибавление известных векторов и вычисление эвклидовой нормы — намного проще).

Для поиска $v_i(x)$ и $w_i(x)$ достаточно владеть такой операцией: строить разложение любого вектора $y \in P_I L_i$ по векторам $P_I e^{ij}$ ($j = 1, ..., r$). Напомним, что коэффициенты y_j разложения $y = \sum_{j=1}^{r} y_j P_I e^{ij}$ удовлетворяют системе r линейных уравнений

$$\sum_{j=1}^{r} y_j (P_I e^{ij}, P_I e^{ik}) = (y, P_I e^{ik}) \quad (k = 1, ..., r). \tag{2.111}$$

Обозначим G_i матрицу Грама $Gr\{P_I e^{i1}, ..., P_I e^{ik}\}$:

$$G_i = (g^i_{jk}), \; g^i_{ij} = (P_I e^{ij}, P_I e^{ik}). \tag{2.112}$$

Система уравнений на y_j имеет вид

$$\sum_k g^i_{jk} y_k = (y, P_I e^{ij}). \tag{2.113}$$

Для ее решения при произвольном y нужно иметь обратную матрицу Грама G_i^{-1} (использование ортогонализации $P_I e^{ij}$ есть по существу обращение G_i методом Гаусса).

Заранее, т.е. для всех возможных расположений пробелов данных (произвольных I), вычислить все G_i^{-1} невозможно: для каждого l их число равно kC_n^l (k — число классов, n — размерность данных, l — число неизвестных данных). Для каждого нового I надо заново настраивать систему, формируя k матриц Грама $G_i = Gr\{P_I e^{i1}, ..., P_I e^{ir}\}$ и вычисляя их обратные.

Задача упрощается в трех случаях.

1. Малые r. Уже при $r = 1, 2, 3$ задача мозаичной регрессии является вполне содержательной, а задача построения и обращения соответствующих матриц Грама проста и может заново решаться для каждого вектора неполных данных x.

2. Малые l. Если число неизвестных данных l невелико ($l = 1, 2, 3$), то вычисление G_i^{-1} также может существенно упрощаться (об этом чуть позже).

3. Заранее определенные I. В некоторых случаях заранее известны совокупности номеров недостающих данных. Для этих нескольких множеств I можно вычислить матрицы G_i^{-1} и сохранять их.

Вариант малых l разберем начиная с самого простого случая $l = 1$ (один пробел в данных). Для него I состоит из одного элемента: $I = \{i_0\}$. Ввиду ортонормированности векторов e^{ij} ($j = 1, ..., r$) матрица Грама G_i имеет вид

$$g^i_{jk} = (P_I e^{ij}, P_I e^{ik}) = \delta_{jk} - e^{ij}_{i_0} e^{ik}_{i_0}, \tag{2.114}$$

где $\delta_{jk} = \begin{cases} 1, \text{ если } j = k, \\ 0, \text{ если } j \neq k \end{cases}$; $e^{ij}_{i_0}$ — координата e^{ij}, соответствующая пропущенным данным.

Обозначим b_i r-мерный вектор с координатами $b^i_j = e^{ij}_{i_0}$ (если векторы-столбцы e^{ij} расположить один за другим в одну таблицу, то координаты b^j соответствуют строке этой таблицы за номером i_0). Тогда

$$G_i = 1 - b^i \otimes b^{i\,\mathrm{T}} \tag{2.115}$$

и, как легко проверить,

$$G_i^{-1} = 1 + \frac{b^i \otimes b^{i\,\mathrm{T}}}{1 - \|b^i\|^2} \tag{2.116}$$

(вообще, для любого проектора выполнено равенство

$$P(1 - \alpha P)^{-1} = 1 + \alpha P/(1 - \alpha)). \tag{2.117}$$

Пусть теперь I состоит из нескольких элементов: $I = \{i_1, i_2, \ldots, i_l\}$. Тогда

$$g^i_{jk} = \delta_{jk} - \sum_{p=1}^{l} b^{pi}_j b^{pi}_k, \tag{2.118}$$

где b^{pi} — l-мерный вектор с координатами $b^{pi}_j = e^{ij}_{i_p}$, образующими в таблице со строками e^{ij} строку с номером $i_p \in I$. Пусть $l < r$ (l мало). Обозначим B_{il} подпространство r-мерного пространства R^r, порожденное векторами $\{b^{pi}\}$, $p = 1, \ldots, l$, P_{Bil} — ортогональный проектор на это подпространство, а K_{il} — ограничение

$$\sum_{p=1}^{l} b^{pi} \otimes b^{pi\,\mathrm{T}}\Big|_{B_{il}}. \tag{2.119}$$

Тогда G_i можно переписать в виде

$$G_i = 1 - K_{il}P_{Bil} \text{ и } G_i^{-1} = 1 + K_{il}(1 - K_{il})^{-1}P_{Bil}. \tag{2.120}$$

Обратная матрица вычисляется только для оператора $1 - K_{il}$, действующего в l-мерном пространстве. Для перевода всех операций в координатную форму достаточно построить в B_{il} ортонормированный базис, проведя ортогонализацию b^{pi}, и представить с его помощью операторы K_{il} и P_{Bil}. Явные формулы для малых l ($l = 1, 2$) и простые алгоритмы для несколько больших l легко выписываются.

Задача восстановления l неизвестных данных сводится по существу к решению системы l линейных уравнений. Выпишем эту систему. Пусть x — вектор неполных данных. Это означает, что данные лежат в линейном многообразии L_x:

$$L_x = \{x^0 + \sum_{i=1}^{l} \beta_i g^i\}, \qquad (2.121)$$

где β_i — произвольные числа (всевозможные значения пропущенных данных); g^i — соответствующие пропущенным данным базисные векторы; x^0 — вектор данных, в котором недостающие значения заменены нулями, $(x^0, g^i) = 0$, $(g^i, g^j) = \delta_{ij}$.

Пусть ядро класса a — линейное многообразие вида $a = M + L$, где M — вектор; L — линейное подпространство с ортонормированным базисом $\{e^j\}$ $(i = 1, ..., r)$.

Расстояние d от произвольного элемента $y \in L_x$ до a есть квадратичный многочлен от β_i:

$$d = \|x^0 + \sum_{i=1}^{l} \beta_i g^i - M\|^2 - \sum_{j=1}^{r} (x^0 + \sum_{i=1}^{l} \beta_i g^i - M, e^j)^2. \qquad (2.122)$$

Точка минимума d является решением системы линейных уравнений

$$\beta_i - \sum_{k,j} (g^i, e^j)(e^j, g^k)\beta_k = (g^i, M) - \sum_j (g^i, e^j)(e^j, x^0 - M)$$

$$(i, k = 1, ..., l;\ j = 1, ..., r). \qquad (2.123)$$

Решив эту систему, найдем числа β_i — наилучшую оценку недостающих данных.

Пусть для данного I в нашем распоряжении имеется набор матриц G_i^{-1}, вычисленный заранее, либо специально для этого случая. Опишем процесс восстановления неизвестных данных с номерами из I и/или исправления известных с помощью набора $\{a_i\}$.

Пусть x — вектор неполных данных.

1-й шаг. Для каждого i вычисляем $P_I x - P_I M_i$.

2-й шаг. Вычисляем скалярные произведения

$$S_j^i(x) = (P_I x - P_I M_i, P_I e^{ij}) \quad (j = 1, ..., r;\ i = 1, ..., k).$$

3-й шаг. Для каждого i действуем матрицей G_i^{-1} на r-мерный вектор $s^i(x)$ с координатами $s_j^i(x)$, вычисленными на предыдущем шаге:

$$u^i(x) = G_i^{-1} s^i(x)$$

(координаты этого вектора $u_j^i(x)$ суть коэффициенты разложения $Pr_{P_I L_i}(P_I x - P_I M_i) = \sum_{j=1}^{r} u_j^i(x) P_I e^{ij}$).

4-й шаг. Находим значения квадратичных форм, у которых матрицы коэффициентов совпадают с G_i:

$$Q_i = \| Pr_{P_I L_i}(P_I x - P_I M_i) \|^2 =$$

$$= \sum_{j,k=1}^{r} (P_I e^{ij}, P_I e^{ik}) u_j^i(x) u_k^i(x) = \sum_{j,k=1}^{r} g_{jk}^i u_j^i(x) u_k^i(x).$$

5-й шаг. Вычисляем для всех i

$$d(x, a_i) = \|P_I x - P_I M_i\|^2 - Q_i.$$

6-й шаг. Находим наименьшее из чисел $d(x, a_i)$ $(i = 1, ..., k)$. Соответствующее i обозначим $i_{\min}$.

7-й шаг. Для $i = i_{\min}$ вычисляем

$$w_i(x) = \sum_{j=1}^{r} u_j^i(x) e^{ij}.$$

8-й шаг. Вычисляем исправленный и дополненный x:

$$x' = M_i + w_i(x) \quad (i = i_{\min}).$$

Либо *шаг 8′*. Вычисляем дополненный x без исправлений:

$$x_j'' = \begin{cases} x_j', & \text{если } x_j \text{ неизвестно;} \\ x_j, & \text{если } x_j \text{ известно.} \end{cases}$$

Результат шага 8 (или *8′*) и является итогом обработки. Большинство операций выполняется слоями простых элементов (сумматоры, квадратичные сумматоры). В целом нетрудно нарисовать нейронную сеть (со слоем интерпретатора "победитель забирает все"), которая выполняет описанную обработку.

До сих пор рассматривался невырожденный случай (общего положения). Возможно несколько вариантов вырождения.

1. Неединственность ядра класса — вместо одного a_i имеем семейство линейных многообразий A_i. В таком случае для классификации используется мера близости

$$d_{\max}(x, A_i) = \max_{a \in A_i} d(x, a) = d(x, \bigcap_{a \in A_i} a) \qquad (2.124)$$

(напомним, что при обучении, напротив, выбиралось $d_{\min}(x, A_i)$). Применение $d_{\max}$ означает по существу уменьшение размерности a_i по сравнению с невырожденным случаем — других отличий нет.

2. Линейная зависимость векторов $P_I e^{ij}$ $(j = 1, ..., r)$ при обработке неполных данных. В этом случае пробелы в данных не могут быть полностью восстановлены.

3. Попадание примера на границу между классами.

Неполные данные используются также при обучении классификатора (при $1 + r < n$). Для этого применяется мера близости $d(x, a) = d(P_I x, P_I a)$. Подробнее ее вычисление см. выше.

Описанные алгоритмы мозаичной регрессии используют в качестве "элементарной" процедуры поиск собственных подпространств ковариационной матрицы. Слегка модифицируя задачу, можно построить алгоритм, обращающийся к более простой основной операции — минимизации квадратичных многочленов от многих переменных. За такое упрощение придется платить увеличением размерности.

Пусть известна некоторая информация о возможном расположении пробелов в данных. Оформим их в виде гипотез о наборах одновременно неизвестных значений: заданы множества индексов $J_1, ..., J_s \subset \subset \{1, ..., n\}$ и предполагается, что номера неизвестных данных для каждого объекта лежат хотя бы в одном из J_i ($i = 1, ..., s$). В противном случае объект не рассматривается (например, отправляется запрос о дополнительных данных).

При определении J_i важен компромисс между количеством этих множеств s (оно не должно быть слишком большим), полнотой охвата различных случаев (отбрасываемых объектов должно быть как можно меньше) и точностью оценки для каждого объекта (различие между хотя бы одним из множеств J_i и множеством $J_x \subset J_i$ номеров недостающих данных для рассматриваемого объекта должно быть не слишком велико).

Наибольшая полнота охвата одновременно с наиболее точной оценкой достигается таким выбором: $J_1, ..., J_s$ — все нетривиальные подмножества $\{1, ..., n\}$. При этом, однако, $s = 2^n - 2$ — слишком большое число уже для весьма умеренных значений n.

Минимальное s при максимальной полноте (не будет отброшен ни один объект, про который хоть что-нибудь известно) достигается при выборе в качестве $J_1, ..., J_s$ совокупности всех $n - 1$-элементных подмножеств $\{1, ..., n\}$ ($s = n$). Недостаток такого выбора — даже отсутствие только одной компоненты данных будет моделироваться как нехватка $n - 1$-го значения.

Выбор компромиссного решения определяется конкретной задачей.

Схема дальнейших действий такова. Для каждого J_i задается пространство возможных регрессионных зависимостей неизвестных данных (с номерами из J_i) от известных. Эти регрессионные зависимости служат ядрами классов при классификации без учителя методом динамических ядер. В результате для каждого J_i получаем классификатор, а для каждого класса — регрессионную зависимость. При обработке вектора неполных данных x с номерами неизвестных данных J_x поступаем так:

1) ищем те J_i, для которых $J_x \subset J_i$, $J_x \neq J_i$;

2) для каждого их этих J_i ищем класс, к которому относится x и определяем соответствующую регрессионную зависимость;

3) по найденным регрессионным зависимостям проводим пробное восстановление известных данных — ищем те x_q, для которых $q \in J_i \setminus J_x$;

4) сравниваем средние квадратичные ошибки пробного восстановления (в расчете на одну координату) для различных J_i; выбираем те из них, для которых эта ошибка меньше;

5) для найденного на предыдущем этапе J_i соответствующего класса и регрессионной зависимости проводим восстановление неизвестных данных x_q при $q \in J_x$.

Для того чтобы можно было сравнивать ошибки при восстановлении различных координат вектора данных, необходима предварительная их нормировка. Предполагается, что такая предобработка проведена.

Для завершения описания достаточно определить меру близости вектора неполных данных к ядру класса и процедуру построения ядра по классу (в который входят векторы неполных данных). Можно ограничиться рассмотрением одного множества индексов J, так как классификаторы строятся и работают для различных J_i независимо.

С каждым множеством индексов $J \in \{1, ..., n\}$ связано разложение пространства векторов данных E в прямую сумму:

$$E = E_j^0 \oplus E_J^1,\ x = x^{0J} \oplus x^{1J}, \text{ для любого } x \in E,$$

где x^{0J} — вектор из тех координат x, номера которых не принадлежат J; x^{1J} — вектор из тех координат x, номера которых принадлежат J.

Это разложение достигается перегруппировкой координат.

Для каждого J задается линейное пространство Rg_j допустимых регрессионных уравнений, выражающих x^{1J} через x^{0J}. Выбор Rg_j практически не важен для нашего изложения. Чаще всего используется линейная регрессия. Для нее Rg_j состоит из всех неоднородных линейных зависимостей вида

$$x^{1J} = Ax^{0J} + b,$$

где A — линейный оператор из E_j^0 в E_j^1 (задается матрицей размерности $|J| \times (n - |J|)$); b — произвольный элемент E_j^1 (вектор размерности $|J|$).

Пространство Rg_j может также состоять из квадратичных зависимостей и т.д. Для нас здесь важно лишь то, что оно линейное, и минимизация квадратичного многочлена в нем сводится к решению системы линейных уравнений.

Пусть x — вектор неполных данных, для которого множество номеров неизвестных данных J_x лежит в J, но не совпадает с ним:

$J_x \subset J$, $J_x \neq J$. Для каждой регрессионной зависимости $F \in Rg_J$ определим меру близости

$$d(x, F) = \sum_{i \in J \setminus J_x} (x_i - F(x^{0J}))^2. \qquad (2.125)$$

Для сравнения различных J потребуется мера близости d, нормированная как среднеквадратичная ошибка на одну координату:

$$\tilde{d}(x, F) = \frac{1}{|J \setminus J_x|} d(x, F). \qquad (2.126)$$

Если P — множество векторов данных (полных или неполных) и для каждого $x \in P$ выполнено $J_x \subset J$, $J_x \neq J$, то положим

$$D(P, F) = \sum_{x \in P} d(x, F). \qquad (2.127)$$

Величина $D(P, F)$ является квадратичным полиномом от F. Поиск регрессионной зависимости F, минимизирующей $D(P, F)$, состоит в минимизации этого полинома. Тем самым определено все необходимое для решения задачи классификации без учителя методом динамических ядер.

Транспонированная регрессия предполагает построение для каждого объекта A некоторого (небольшого) множества $M = M(A) = \{B_1, ..., B_g\}$ других объектов и функции $f(y_1, ..., y_g)$ от q переменных, позволяющей при всех i вычислить значение i-го признака объекта $A(x_i)$ по значениям этого признака для q объектов из M $(x_i^1, ..., x_i^q)$:

$$x_i \cong f(x_i^1, ..., x_i^q). \qquad (2.128)$$

Дополнительное ограничение на f — инвариантность к смене шкал:

$$f(ay_1 + b, ..., ay_q + b) = af(y_1, ..., y_q) + b. \qquad (2.129)$$

Способ реализации этого ограничения — представление f в виде

$$f(y_1, ..., y_q) = M_y + \sum_{j=1}^{q} \varphi_j(y_1 - M_y, ..., y_q - M_y)(y_j - M_y), \qquad (2.130)$$

где $M_y = \frac{1}{q} \sum_{j=1}^{q} y_j$; φ_j — однородные функции нулевой степени: для любого a:

$$\varphi_j(az_1, ..., az_q) = \varphi_j(z_1, ..., z_q).$$

Простейшие примеры таких φ:

$$1)\ \varphi = \text{const};\ 2)\ \varphi = \frac{a_1 z_1 + ... + a_q z_q}{b_1 z_1 + ... + b_q z_q},$$

3) $\varphi = \dfrac{a_1\varphi_1 + \ldots + a_q\varphi_q}{b_1\varphi_1 + \ldots + b_p\varphi_p}$, где все функции φ_i — однородные нулевой степени.

Из приведенных выражений видна особая роль однородных Паде-нейронов в задачах транспонированной регрессии.

Выделим сначала более простую задачу: пусть множество M задано. Будем последовательно строить функцию регрессии f, минимизируя оценку H (сумму квадратов ошибок):

$$H = \sum_i (x_i - f(x_i^1, \ldots, x_i^q))^2 \qquad (2.131)$$

(суммирование ведется по всем известным признакам x_i объекта A). Будем последовательно усложнять семейство допустимых φ, пока не достигнет требуемой точности. Начнем с линейных f:

$$\varphi_j^i = \text{const}_j;\ f^0(y_1, \ldots, y_q) = M_y + \sum_{j=1}^{q} \varphi_j^0 (y_j - M_y). \qquad (2.132)$$

(Заметим, что химики в XIX в. работали с еще более простой функцией $f \equiv M_y$.) Подберем коэффициенты φ_j^0, минимизируя H. Этот процесс можно представить как обучение нейронной сети (рис. 2.4, *а*). Веса выходного сумматора этой сети равны φ_j^0 (для приема сигналов $y_j - M_y$) и 1 (для приема сигнала M_y). Если точность полученной аппроксимации нас не удовлетворяет, переходим к $\varphi = \varphi^0 + \varphi^1$, где ρ^1 получается в результате дробно-линейного преобразования:

$$\varphi_j^1 = \frac{\sum_{r=1}^{q} a_{jr}(y_r - M_y)}{\sum_{r=1}^{q} b_{jr}(y_r - M_y)}. \qquad (2.133)$$

Такое преобразование можно представить с помощью слоя из q Паде-нейронов. Выходные сигналы этого слоя прибавляются к весам φ_j^0 выходного сумматора (рис. 2.4, *б*).

В случае необходимости можно перейти к построению добавки φ_j^2, получаемой в результате последовательного действия двух дробно-линейных преобразований на вектор с координатами $y_r - M_y$, и т.д.

Подбор коэффициентов может проводиться с помощью любого стандартного алгоритма оптимизации (см. в следующих главах).

Выбор множества M осуществляется по-разному. Если заранее ограничиться определенным малым числом элементов (например, $|M| = 2$ или 3), то можно просто перебрать все такие M, построить для каждого из них функцию регрессии f заданной (ограниченной по

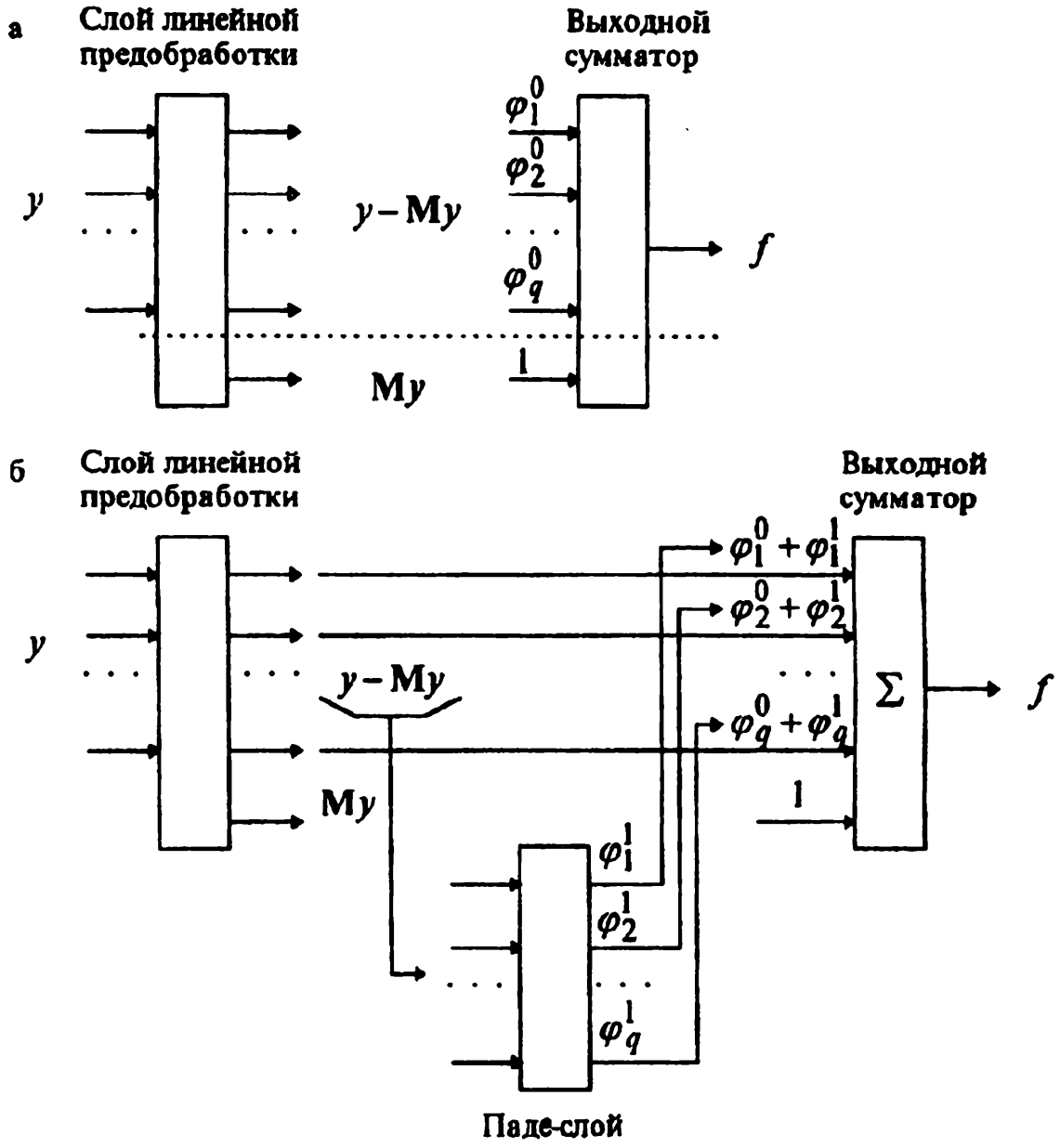


Рис. 2.4. Сеть для решения задачи транспонированной регрессии.
а — линейное приближение (веса выходного сумматора равны φ_i^0); *б* — приближение с одним дробно-линейным (Паде-) слоем (веса выходного сумматора равны $\varphi_j^1 + \varphi_j^0$).

числу Паде-слоев) сложности и выбрать наилучшее M по минимуму ошибки. Возможны различные вариации и усложнения этого подхода.

Другой путь состоит в том, что мы начинаем с избыточно большого M (например, M — множество всех объектов), строим ограниченную по сложности функцию f, а потом контрастируем полученную нейронную сеть, отбрасывая максимально возможное число входных сигналов (напомним, что при транспонированной регрессии каждому входному сигналу сети соответствует объект из M). Выбор метода контрастирования — по вкладу в оценку или по изменению функционирования — не принципиален. Чаще используются методы, основанные на использовании оценки.

Важно поддерживать разумное соотношение между числом примеров в обучающей выборке и количеством параметров. В рассматривае-

ном случае примеров столько же, сколько известных значений признаков у данного объекта. Число входных сигналов равно $|M|$ — количеству элементов в множестве объектов, а подстраиваемых параметров у системы (см. рис. 2.4) $q + lq(2q - l)$, где $q = |M|$, а l — число Паде-слоев у системы (каждый однородный Паде-нейрон имеет $2q - 1$ независимых настраиваемых параметров). Контрастирование связей сократит множество параметров до минимально возможного, но даже при этом желательно при больших $|M|$ задавать меньше Паде-слоев, увеличивая их число по мере уменьшения $|M|$. В принципе, можно начинать с линейных функций f (рис. 2.4, *а*) или даже просто со средних значений, как это делали химики прошлого века, а потом, по мере уменьшения количества объектов, наращивать слои и повышать точность.

Приведем пример множеств $M(A)$, построенных с помощью линейной транспонированной регрессии при автоматической обработке свойств (первых четырех потенциалов ионизации) химических элементов [66, 67]. Для того чтобы получать линейные прогнозы потенциалов ионизации железа (Fe), можно использовать следующие "триады" и "тетрады" (в отличие от триад Деберейнера прогноз получается с помощью линейной регрессии, а не просто усреднением): Na—V—Fe, K—Fe—Co, Ca—Sc—Fe, Cr—Mn—Fe—Co, Mn—Fe—Ni.

Для кобальта (Co) получаем следующие множества: K—Fe—Co, Cr—Mn—Fe—Co, Cr—Mn—Fe—Co, Fe—Co—Ni, Co—Mo—In, Ca—Sc—Co.

Для никеля (Ni): Fe—Co—Ni, Cr—Mn—Ni, Ca—Sc—Ni, B—Ni—Fr.

Наряду с наборами, более или менее очевидными из таблицы Менделеева (Fe—Co—Ni, Cr—Mn—Fe—Co, Mn—Fe—Ni, Ca—Sc—Fe и т.п.), в список включены и более неожиданные группы (Co—Mo—In, B—Ni—Fr). Детальнее с этой работой можно ознакомиться по публикациям.

Минимизация функции, заданной аналитически, не вызывает затруднений, если удается построить сеть, вычисляющую ее градиент. Если требуется минимизировать $H(x)$ и есть черный ящик, вычисляющий $\operatorname{grad} H(x)$, то построим итерационный процесс (рис. 2.5):

$$x \to x - h \operatorname{grad} H(x), \qquad (2.134)$$

где параметр h (шаг) задается извне как параметр настройки. Процесс заканчивается при отсутствии или малости изменений (зацикливании). По крайней мере, будет найдена точка x с малым значением $\| \operatorname{grad} H(x) \|$. В большинстве случаев таким образом попадаем в окрестность локального минимума H, хотя нетрудно привести примеры ложных остановок вдали от точек минимума.

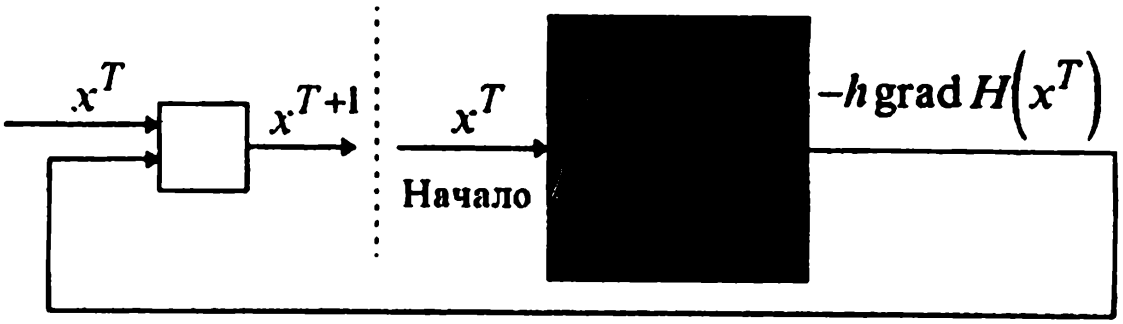


Рис. 2.5. Цикл функционирования системы, минимизирующей функцию H. Верхние индексы — номера циклов, пунктирной линией разделены начало T-го цикла и его окончание.

Пусть H — квадратичный полином от n переменных $x_1, \ldots, x_n$:

$$H(x) = \frac{1}{2}(x, QX) + (b, x), \tag{2.135}$$

где x — n-мерный вектор с координатами x_i; Q — симметричная $n \times n$-матрица, $Q = (q_{ij})$; b — n-мерный вектор. Тогда

$$\mathrm{grad}\, H = Qx + b. \tag{2.136}$$

Этот вектор является выходным для слоя сумматоров, каждый из которых вычисляет линейную неоднородную функцию

$$b_i + \sum_j q_{ij} x_j. \tag{2.137}$$

Таким образом, сеть, минимизирующая квадратичную форму, строится очень просто — черный ящик (см. рис. 2.5) заменяется слоем сумматоров.

Решение системы линейных уравнений

$$Ax = b,\ A = (a_{ij}) \tag{2.138}$$

сводится к минимизации квадратичной формы:

$$H(x) = \frac{1}{2}\|Ax - b\|^2 = \frac{1}{2}(Ax, Ax) - (Ax, b) + \frac{1}{2}(b, b);$$

$$\mathrm{grad}\, H = A^{\mathrm{T}}Ab - A^{\mathrm{T}}b = A^{\mathrm{T}}(Ax - b), \tag{2.140}$$

где A^{T} — транспонированная матрица A.

Вычисление grad H удобно представить в виде работы двух слоев сумматоров: первый вычисляет по x и b вектор $Ax - b$, действие второго состоит в умножении вектора его входных сигналов на A^{T}. На i-й сумматор первого слоя подаются вектор x и координата b_i вектора b. Входные веса этого i-го сумматора равны элементам i-й строки матрицы A — a_{ij} (для приема x_j) и -1 (для приема b_i). Коэффициенты k-го сумматора второго слоя состоят из элементов k-го столбца матрицы A.

Простейшее обобщение квадратичных форм — функции H следующего вида:

$$H(x) = \frac{1}{2}(x, Qx) + (b, x) - \sum_i f(x_i), \qquad (2.141)$$

где f — некоторая дифференцируемая функция одного переменного.

Вычисление grad H тоже может проводиться простой сетью:

$$\frac{\partial H}{\partial x_i} = b_i + \sum_j q_{ij} x_j - f'(x_i). \qquad (2.142)$$

Дальнейшие усложнения и вариации бесконечно разнообразны. Большинство из них основываются на следующем соображении: если построена сеть, вычисляющая функцию $H(x)$, то после небольшого усложнения и обращения стрелок легко получить сеть, вычисляющую grad $H(x)$ (принцип двойственности — подробнее об этом в следующей главе).

Выбор шага h для устройств, изображенных на рис. 2.5, без вычисления и сравнения значений функции H — довольно сложный вопрос. Существует, однако, простой универсальный выход: выбирается шаг h_T, зависящий от момента дискретного времени (номера итерации) T. Если $h_T > 0$, $h_T \to 0$ и $\sum_T h_T \to \infty$ (например, $h = h_0/T$ или $h_T = h_0/T^{1/2}$, или $h_T = h_0/\ln|T + 1|$), то сходимость метода обеспечена.

Этот способ не очень эффективен, так как сначала шаги слишком велики (по сравнению с оптимальными), а со временем становятся слишком малыми, но зато он универсален и не предполагает дополнительного управляющего вмешательства в вычисления.

Некоторые модели ассоциативной памяти. *Построение ассоциативной памяти* ориентируется на представление об идеальном устройстве, которое должно вести себя примерно так, как описано ниже.

А. Векторы сигналов принадлежат n-мерному эвклидову пространству E. Задана некоторая мера близости d в E: для любых $x, y \in E$ определена процедура вычисления $d(x, y)$. Примем для определенности, что $d(x,x) = 0$, $d(x, y) \geqslant 0$, $d(x, y) = d(y, x)$ и уменьшение $d(x, y)$ содержательно означает сближение x и y.

Б. Ассоциативная память должна запомнить множество из N эталонов $\{x^p\}$ и при получении вектора сигналов x выдать ближайший к нему эталон x^p (с минимальным $d(x^p, x)$). Желательно также иметь дополнительную возможность: если эталон x^p по какой-либо причине не устраивает пользователя, то память выдает x^q со следующим за $d(x^p, x)$ значением $d(x^q, x)$, и т.д. в порядке возрастания $d(x^q, x)$.

В. Наконец, три требования эффективности:

1) число запоминаемых эталонов N может существенно превосходить размерность n пространства сигналов (идеал — $N \simeq 2^n$, желательно $N \simeq n^k$, $k > 1$);

2) сложность устройства и объем вычислений при вспоминании должны с увеличением n расти как const $\times$ n (не быстрее);

3) запоминание дополнительного эталона должно быть не сложнее (или ненамного сложнее), чем один акт вспоминания (const $\times$ n).

Есть подозрение, что эти три требования противоречивы. Однако доказать противоречивость можно только при более строгой формализации, и всегда останется место для сомнения: а вдруг при другой формализации требования эффективности станут совместимыми?

Простейший способ реализации ассоциативной памяти состоит в том, что каждому эталону x^p ставится в соответствие свой "нейрон" первого слоя, вычисляющий для произвольного x меру близости $d(x^p, x)$. Далее сигналы $A_p = d(x^p, x)$ могут обрабатываться по-разному. Первый вариант — они поступают на интерпретирующее устройство, которое преобразует вектор A_p в

$$B_p = \begin{cases} 1, \text{ если } A_p = \min\limits_q \{A_q\}, \\ 0, \text{ в противном случае.} \end{cases}$$

С каждым p связан требуемый вектор выходных сигналов $y = y^p$. Он может быть, например, двоичным кодом (номером) эталона, его именем или, наконец, самим эталоном x^p. Каждой координате y_i соответствует выходной элемент — сумматор, вычисляющий сумму $y_i = \sum\limits_p y_i^p B_p$. Таким образом, веса связей выходных сумматоров системы равны y_i^p.

Вычисление B_p не вполне соответствует представлению о нейросистемах. Можно пойти и по другому пути. Пусть от ассоциативной памяти требуется выдавать сами векторы эталонных сигналов x^p. Получив в результате работы первого слоя вектор сигналов $A_p = d(x^p, x)$, положим

$$x' = \sum_p x^p f(A_p),$$

где f — монотонно (и достаточно быстро) убывающая функция, например, $f(A) = \dfrac{1}{A + \varepsilon}$ ($\varepsilon > 0$ и достаточно мало) или $f(A) = e^{-A/\varepsilon}$. После соответствующей нормировки подадим x' вместо x на вход системы и будем повторять эти преобразования, пока не придем к неподвижной точке. Варианты нормировки следующие:

1) $x' \to x'/\|x'\|$, где $\|\cdot\|$ — некоторая (не обязательно эвклидова) норма;

2) $x' \to \operatorname{sign} x'_1$;

3) $x'_i = \begin{cases} 1, \text{ если } x'_1 > M(x'), \\ 0, \text{ если } x'_i \leqslant M(x'), \end{cases}$

где $M(x) = \frac{1}{n} \sum_i x_i$.

Первая нормировка не нуждается в комментариях, второй и третий варианты представляют собой частные случаи *бинаризации* и используются тогда, когда координаты входного вектора могут принимать только два значения (бинарные изображения, 0—1 последовательности и т.п.).

Выбор мер близости очень богат. Простейший случай — $d(x, y)$ — квадрат эвклидового расстояния между x, y. При сравнении между собой $d(x^p, x)$ и $d(x^r, x)$ можно ограничиться вычислением линейных функций от x:

$$l(x^p, x) = d(x^p, x) - \|x\|^2 = \|x^p\|^2 - 2(x^p, x).$$

Наиболее важны другие варианты выбора $d(x, y)$:

1) $d(x, y) = 1 - r(x, y)$, где $r(x, y)$ — коэффициент корреляции —

$$r(x, y) = \sum_i \frac{x_i - M_x}{\sigma_x} \times \frac{y_i - M_y}{\sigma_y};$$

2) $d(x, y)$ — квадрат расстояния между образами x и y в пространстве инвариантов некоторой группы.

В целом подход, основанный на вычислении мер близости $d(x^p, x)$, удовлетворяет первому и последнему требованиям эффективности, но находится в противоречии со вторым: приходится использовать столько элементов первого слоя, сколько задано эталонов — сложность растет, как число эталонов N.

Хорошим учебным примером для реализации на PC является обработка бинарных изображений ($x_i = 0; 1$) по следующей схеме:

$$x \to Tx \to \lceil Tx \rceil, \tag{2.143}$$

где $Tx = \sum_p r(x, x^p) x^p$; $\lceil y \rceil_i = \begin{cases} 1, \text{ если } y_i > \varepsilon M_y, \\ 0, \text{ если } y_i \leqslant \varepsilon M_y; \end{cases}$

$M_y = \frac{1}{n} \sum_{i=1}^{n} y_i$; $\varepsilon > 0$ — уровень среза (по умолчанию полагаем $\varepsilon = 1$).

Другой пример, несколько отступающий от предложенной схемы, дает проекционная сетчатка:

$$x \to Px \to \lceil Px \rceil, \tag{2.144}$$

где P — ортогональный проектор на линейную оболочку $\{x^p\}$.

В определении проекционной сетчатки можно перейти к проектированию на аффинную оболочку $\{x^p\}$ — на линейное многообразие:

$$\mathrm{Aff}(\{x^p\}) = \{x \mid x = \sum_p \lambda^p x^p, \sum_p \lambda^p = 1\} \qquad (2.145)$$

(отличается от линейной оболочки дополнительным условием $\sum_p \lambda^p = 1$). В таком случае первое проектирование будет устранять компоненту сигнала, ортогональную разностям $x^p - x^q$ ("незлонамеренный" шум, т.е. шум, не переводящий один эталон в другой).

Оба примера (2.143), (2.144) не удовлетворяют первому требованию эффективности — при числе эталонов, приближающемся к размерности n, предлагаемые алгоритмы перестают работать. В связи с этим для них не имеет смысла второе требование, а вот третье, очевидно, выполняется: добавление нового эталона x^{N+1} к $\{x^p\}$ приводит всего-навсего к появлению одного слагаемого в Tx, а для проекционной сетчатки такое добавление ведет к замене P на P':

$$P'x = Px + (x^{N+1} - Px^{N+1}) \frac{(x^{N+1} - Px^{N+1}, x)}{\|x^{N+1} - Px^{N+1}\|^2}, \qquad (2.146)$$

если $x^{N+1} \neq Px^{N+1}$ и $P' = P$ в противном случае.

Неоднократно предпринимались попытки построить системы ассоциативной памяти, сложность которой определялась бы размерностью пространства сигналов n, а объем памяти был бы намного больше. Как правило, эти попытки не достигали цели и приводили к уменьшению числа запоминаемых эталонов до const $\times n$. Наиболее знамениты сетчатки Хопфилда [241]. Они минимизируют функцию

$$H(x) = -\frac{1}{2}\sum_p (x, x^p)^2 + \lambda \sum_i (1 - x_i)^2 (1 + x_i)^2, \qquad (2.147)$$

где $\lambda > 0$, $x_i^p = \pm 1$.

Большое число публикаций посвящено исследованию таких сетчаток и их модификаций. Открыто много интересных свойств, но общий вывод таков — сетчатки Хопфилда и им подобные могут устойчиво запоминать не более n эталонов (обычно менее — порядка $0{,}2n$), да и то при условии, что эталоны "почти ортогональны":

$$(x^p, x^q) \ll \|x^p\| \times \|x^q\| \text{ при } p \neq q. \qquad (2.148)$$

Рабочие свойства у этих сетчаток не лучше (хуже), чем у системы (2.144). С другой стороны, их привлекательность для исследователей определяется простым и содержательно осмысленным правилом формирования матрицы связей: линейная по x часть $y = \mathrm{grad}\, H$ имеет вид

$$y_i = \sum_j k_{ij} x_j,\ k_{ij} = -\sum_p x_i^p x_j^p; \qquad (2.149)$$

оно связано с известным предположением — при подкреплении успешного действия ассоциативные связи устанавливаются между теми нейронами (i-м и j-м), которые возбуждаются одновременно.

Квадратичную часть функции H в сетчатках Хопфилда трактуют как аналог потенциальной энергии некоторого двухчастичного взаимодействия. Переход к трех-, четырехчастичному и более высоким степеням взаимодействия порождает ассоциативную память, которая работает намного лучше, чем сетчатки Хопфилда.

Аналогично, и в проекционной сетчатке (2.144) можно перейти к пространству двух-, трехчастичных и более высоких k-частичных функций. Делается это, например, так. С каждым вектором данных x связывается k-индексная величина $F_k(x)$:

$$(F_k(x))_{i_1 i_2 \ldots i_k} = x_1 \times x_{i_1} \times x_{i_2} \times \ldots \times x_{i_k}. \qquad (2.250)$$

Множество всех k-индексных величин над n-мерным пространством данных E образует n^k-мерное линейное пространство — k-ю тензорную степень E. Обозначим его $E^{\otimes k}$. Для $k = 2$ его можно представить как пространство матриц $k \times k$. В пространстве $E^{\otimes k}$ определено скалярное произведение

$$(F, G) = \sum_{i_1 i_2 \ldots i_k} F_{i_1 i_2 \ldots i_k} G_{i_1 i_2 \ldots i_k}, \qquad (2.151)$$

поэтому можно строить ортогональные проекторы. Для каждого эталона x^q определим соответствующий вектор $F_k(x^q)$ и построим систему векторов $V^q \in E^{\otimes k}$, обладающую следующими свойствами:

1) V^q принадлежит линейной оболочке векторов $F_k(x^p)$ — $V^q \in \in \mathrm{Lin}\{E_k(x^p)\}$;

2) $(V^q, F_k(x^p)) = \delta_{pq}$ (1, если $p = q$, и 0, если $p \neq q$).

Определим в E *нелинейный* оператор проецирования

$$Px = \sum_p x^p (V^p, F_k(x)) \qquad (2.152)$$

и построим соответствующую сетчатку

$$x \to Px \to \lceil Px \rceil. \qquad (2.153)$$

Польза перехода от проецирования в E к проецированию в $E^{\otimes k}$ в нескольких словах поясняется так: преобразование F_k может переводить линейно зависимые векторы эталонов в независимые и тем самым повышать информационную емкость сетчатки.

Система векторов $\{V^p\}$ строится последовательно, аналогично тому, как строился проектор в E. Если эталон один — x^1, то $V^1 =$

$= x^1/(F_k x^1, F_k x^1)$; пусть построены векторы $V^1, \ldots, V^N$; при добавлении нового эталона x^{N+1} полагаем

$$W^{N+1} = F_k(x^{N+1}) - \sum_{p=1}^{N} F_k(x^p)\,(V^p, F_k(x^{N+1})),$$
$$V^{N+1} = W^{N+1}/(W^{N+1}, F_k(x^{N+1})), \tag{2.154}$$

если $F_k(x^{N+1})$ линейно зависит от векторов $\{F_k(x^p)\}$, то $W^{N+1} = 0$ и эталон x^{N+1} пропускаем (не добавляем к системе векторов $\{x^p\}$; если вычислен вектор V^{N+1}, то переопределяем векторы V^p $(p = 1, \ldots, N)$ так, чтобы они стали ортогональны x^{N+1}:

$$W^p = V^p - V^{N+1}(V^p, F_k(x^{N+1})),$$
$$V^p := W^p/(W^p, F_k(x^p)). \tag{2.155}$$

Большая размерность пространства тензоров $E^{\otimes k}$ (n^k) не означает пропорционального n^k роста сложности вычислений, так как реально не производится операций с тензорами, а формулы (2.154), (2.155) используются для построения отображения $E \to E$ (2.152). Возможно и сокращение размерности пространства тензоров, например за счет ограничения радиуса учитываемых корреляций.

Пусть у векторов данных номера координат принадлежат некоторой решетке в векторном пространстве (моменты отсчета сигнала, точки изображения и т.п.). Перейдем от пространства $E^{\otimes k}$ к пространству k-индексных величин, у которых разности между индексами ограничены по норме (т.е. у $F_{i_1 i_2 \ldots i_k}$ оставляются только те координаты, для которых $\|i_s - i_t\| \leqslant r$, где $s, t = 1, \ldots, k$, а $r > 0$ — радиус корреляции). Это преобразование переводит каждый вектор $F \in E^{\otimes k}$ в вектор меньшей размерности F^*.

Можно обогатить возможные варианты построения $F_k^*(x^p)$, практически не усложняя задачу построения сетчатки. Пусть, как и выше, номера координат вектора данных принадлежат некоторой решетке (моменты отсчета сигнала, точки изображения и т.п.) и составляют ее конечное подмножество U. Выберем на этой решетке некоторое множество векторов I, включающее начало координат. Назовем его корреляционной маской. Положим

$$F^* = F_{i+j_1\, i+j_2 \ldots i+j_k}, \tag{2.156}$$

где индексы складываются как векторы решетки, $i \in U$, номера j_s принадлежат корреляционной маске — $j_s \in I$ $(s = 1, \ldots, k)$, а суммы $i + j_s$ являются номерами координат *вектора данных* — $i + j_s \in U$ $(s = 1, \ldots, k)$. Работа с такими F^* не сложнее, чем с величинами, у

которых разности между индексами ограничены по норме, а свободный выбор I дает пользователю много новых возможностей.

Проблема построения эффективной ассоциативной памяти остается открытой. По своему действию ("убрать из сигнала все лишнее и добавить недостающее до ближайшего эталона") многие системы ассоциативной памяти похожи на специализированные *нелинейные фильтры*. В особенности это относится к проекционным сетчаткам (2.144), (2.146), которые уже первым шагом (проецирование P) отбрасывают компоненту сигнала, ортогональную всем эталонам, и тем самым гасят шум, нескоррелированный с эталонными сигналами.

Можно ослабить третье требование эффективности, ограничивая сложность одного поиска аналогов (вспоминания), но не всей системы в целом. Такую систему построить нетрудно: пусть все множество $\{x^p\}$ из N эталонов разбито на k классов примерно равного объема, каждый из этих классов — еще на k и т.д., пока не дойдем до классов, состоящих из одного эталона. Получаем классификационное дерево (рис. 2.6), каждой вершине которого соответствует класс. Если класс состоит более чем из одной точки, то ему соответствует классификатор — устройство, разбивающее его на подклассы (не более k). Любой вектор входных сигналов этот классификатор относит к одному из подклассов.

Глубина классификации, т.е. число ветвлений на пути от корня к вершинам, оценивается как $\log_k N$. Сложность стоящего в вершине дерева классификатора на k классов можно оценить как k (например, сеть Кохонена, построенная для этой задачи, будет состоять из k нейронов). Таким образом, если подать на вход первого классификатора вектор сигналов, потом передать его по дереву следующему и т.д., то сложность сработавшей подсистемы можно оценить как $k\log_k N$. Эта величина достигает минимума при $k = e$ ($= 2{,}7 \ldots$). Среди натуральных k наилучшим является выбор $k = 3$. Необходимые классификаторы, делящие поступающие сигналы на три класса, можно по-

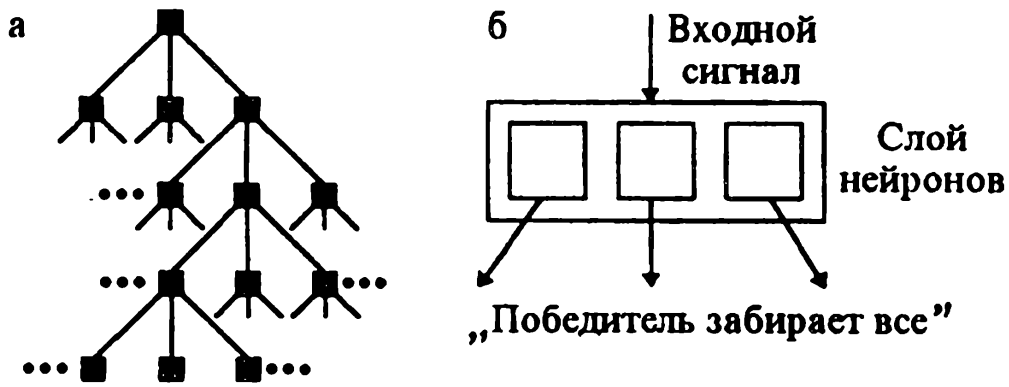


Рис. 2.6. Иерархический поиск.
а — дерево классификации; *б* — схема работы классификатора, соответствующего вершине дерева.

строить методом динамических ядер (сети Кохонена и т.п.) с использованием мер близости d. Если при работе памяти необходимо получить не один, а несколько ближайших аналогов, то достаточно вернуться назад — вверх по классификационному дереву на нужное количество шагов. Каждый такой шаг в k раз увеличивает число вспоминаемых эталонов.

* *
*

На этом мы обрываем изложение методов построения нейронных сетей для решения задач. Это именно обрыв, а не завершение: за рамками изложения остались сети для компрессии данных и для нелинейного факторного анализа, а также многое другое.

За пестротой изложенного материала и тех задач, решение которых не обсуждалось, можно разглядеть два важных обстоятельства.

1. Нейросетевая технология обработки данных захватывает практически все известные задачи, методы и алгоритмы обработки, иногда просто перекладывая их на свой язык, иногда трансформируя и адаптируя известное, а изредка — изобретая новое.

2. За счет универсальности подхода приобретается новый обобщенный взгляд на проблему обработки и выделяется основной круг математических задач, встречающихся повсеместно. Все они связаны с одной проблемой, которая очень просто формулируется, но сложно решается: *заполнение пропусков в таблицах*. Чуть утрируя, можно сказать, что других проблем нет — все (и компрессия, и классификация, и факторный анализ и т.д.) сводится к этой базовой проблеме и/или надстраивается над ней в виде дополнительных вопросов и конструкций.

Глава 3

ОБУЧЕНИЕ КАК ОПТИМИЗАЦИЯ

1. Универсальный метод построения нейронных сетей для решения задач

Если сформулировать требования к нейронной сети как задачу минимизации некоторой функции — *оценки*, зависящей от части сигналов (входных, выходных, ...) и от параметров сети, то можно рассматривать обучение как оптимизацию и строить соответствующие алгоритмы, программное обеспечение и, наконец, устройства (hardware). Функция оценки обычно довольно просто (явно) зависит от части сигналов — входных и выходных. Ее зависимость от параметров

сети может быть сложнее и включать как явные компоненты (слагаемые, сомножители, ...), так и неявные — через сигналы (сигналы, очевидно, зависят от параметров, а функция оценки — от сигналов).

За пределами задач, в которых нейронные сети формируются по явным правилам (сетчатки Хопфилда, проективные сетчатки, минимизация аналитически заданных функций и т.п.), нам неизвестны случаи, когда требования к нейронной сети нельзя было бы представить в форме минимизации функции оценки. Не следует путать такую постановку задачи и ее весьма частный случай — "обучение с учителем". Уже метод динамических ядер, описанный в предыдущей главе, показывает, каким образом обучение без учителя в задачах классификации может быть описано как минимизация некоторой целевой функции, оценивающей качество разбиения на классы.

Если для решения задачи не удается явным образом сформировать сеть, то проблему обучения можно, как правило, сформулировать как задачу минимизации оценки. Осторожность предыдущей фразы ("как правило") связана с тем, что на самом деле нам неизвестны и никогда не будут известны все возможные задачи для нейронных сетей, и быть может, где-то в неизвестности есть задачи, которые несводимы к минимизации оценки.

Минимизация оценки — сложная проблема: параметров астрономически много (для стандартных примеров, реализуемых на PC — от 100 до 1 000 000), адаптивный рельеф (график функции оценки) сложен, может содержать много локальных минимумов, извилистых оврагов и т.п. Наконец, даже для того чтобы воспользоваться простейшими методами гладкой оптимизации, нужно вычислять градиент функции оценки. Здесь мы сталкиваемся с одной "очевидной" догмой, без разрушения которой было бы невозможно эффективное обучение нейронных сетей. Пусть затраты на вычисление одного значения функции n переменных $H(x_1, ..., x_n)$ (оцениваемые временем, работы некоторого универсального вычислительного устройства) примерно равны T_H. Сколько времени потребуется тому же устройству на вычисление grad H (при разумном составлении программы)? Большинство математиков с университетским дипломом ответят:

$$T_{\text{grad}H} \approx nT_H.$$

Это неверно! Правильный ответ:

$$T_{\text{grad}H} \approx CT_H,$$

где C — константа, не зависящая от размерности n (в большинстве случаев $C \approx 3$). Результат выглядит парадоксально, так как grad H — это n функций $\partial H/\partial x_i$. Каждая из них, вычисляемая отдельно, требует затрат времени того же порядка, что и H. Почему же вычисление всех их вместе дешевле, чем по отдельности? Оказывается, если представить H как сложную функцию, являющуюся суперпозицией функ-

ций малого числа переменных (а по-другому вычислять функции многих переменных мы не умеем), и аккуратно воспользоваться правилом дифференцирования сложной функции, не производя лишних промежуточных вычислений и сохраняя полезные промежуточные результаты, то вычисление всей совокупности функций $\partial H/\partial x_i$ ($i = 1, \ldots, n$) ненамного сложнее, чем одной из них — они все собраны из одинаковых блоков. С этим хорошо знакомы физики-теоретики, которым приходится проделывать сложные преобразования формул: выбор удачных промежуточных обозначений намного сокращает вычисления.

Поиск grad H удобно представить как некоторый двойственный процесс над структурой вычисления H. В данной главе описан принцип двойственности для систем автоматов. Проблема обучения нейронных сетей поставлена как специфическая задача многомерной оптимизации. Приведены основные идеи построения алгоритмов обучения нейронных сетей, используемые группой НейроКомп.

2. Двойственное функционирование сетей автоматов при вычислении градиентов сложных функций

Основную идею двойственного функционирования можно понять уже на простейшем примере. Рассмотрим вычисление производной сложной функции. Пусть заданы функции одной переменной f_1, f_2, ..., f_n. Образуем из них сложную функцию

$$F(x) = f_n(f_{n-1}(\ldots(f_1(x))\ldots)). \tag{3.1}$$

Можно представить вычисление $F(x)$ как результат работы n автоматов, каждый из которых имеет один вход и выдает на выходе значение $f_1(A)$, где A — входной сигнал (рис. 3.1, *а*). Чтобы построить систему автоматов, вычисляющую $F'(x)$, надо дополнить исходные автоматы такими, которые вычисляют функции $f'_i(A)$, где A — входной сигнал, и еще цепочкой из $n - 1$ одинаковых автоматов, имеющих по два входа, по одному выходу и подающих на выход произведение входов. Тогда формулу производной сложной функции

$$\frac{dF}{dx} = f'_n(f_{n-1}(\ldots(x)\ldots))f'_{n-1}(\ldots(x)\ldots) \times \ldots \tag{3.2.}$$

можно реализовать с помощью сети автоматов (рис. 3.1, *б*). Сначала по такой схеме вычисления идут слева направо: на входы f_1 и f'_1 подаются значения x, после вычислений f_1 этот входной сигнал подается на входы f_2 и f'_2 и т.д. В конце цепочки оказываются вычисленными все $f_i(f_{i-1}(\ldots))$ и $f'_i(f_{i-1}(\ldots))$.

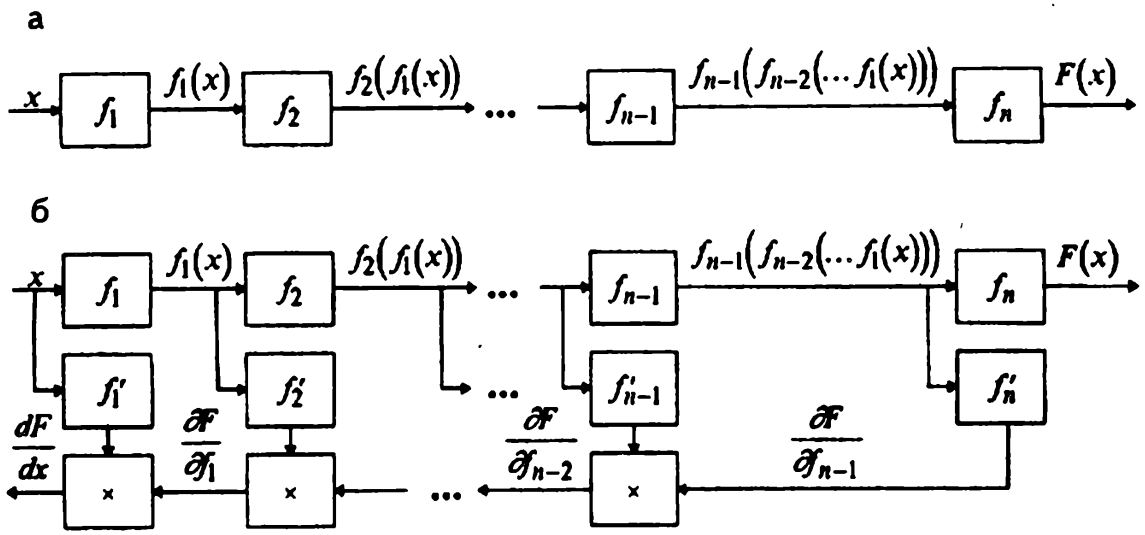


Рис. 3.1. Последовательность автоматов для вычисления сложной функции (*а*) и ее производной (*б*); ⊠ — автомат, подающий на выход произведение входов.

Тем самым для каждого автомата нижней строки, кроме самого правого, оказывается сформированным по одному значению входа, а у самого правого — оба, поэтому он может сработать и начать передачу сигнала справа налево. Это обратное движение есть последовательное вычисление попарных произведений и передача их налево. В конце получаем dF/dx. Дальнейшее содержание настоящего раздела — обобщение схем прямого и обратного функционирования (см. рис. 3.1) на сложные функции многих переменных.

Обратимся сначала к простым (не сложным) функциям многих переменных. Переменные, от которых зависит функция f, разобьем на две группы: параметры $\alpha = (\alpha_1, \ldots, \alpha_k)$ и собственно переменные $x = (x_1, \ldots, x_n)$. Смысл этого разбиения таков: если речь идет об автомате, вычисляющем f, то значения параметров хранятся при автомате (образно говоря, описывают положения "рукояток настройки"), а значения переменных подаются на входы и каждый раз новые. Для нейронной сети это разбиение соответствует выделению параметров нейронов, весов синапсов, с одной стороны, и входных сигналов этих элементов — с другой. Для функций общего вида принципиального различия между параметрами и переменными, конечно, нет.

Будем изображать автомат, вычисляющий простую функцию, как показано на рис. 3.2.

Для вычисления сложной функции параметров $\alpha = (\alpha_1, \ldots, \alpha_k)$ и переменных $x = (x_1, \ldots, x_n)$ задается некоторое количество N простых функций f_i ($i = 1, \ldots, N$) и правило подстановки аргументов. Среди функций f_i возможны совпадающие. Обозначим n_i — число переменных, а k_i — число параметров, от которых зависит f_i. Подчеркнем, что на входы f_i в качестве переменных могут подаваться не только x_i, но также и результаты вычисления других функций f_j.

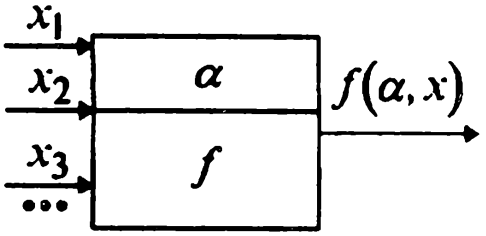


Рис. 3.2. Автомат, вычисляющий функцию нескольких переменных $f(\alpha, x)$.

Чтобы построить процесс вычисления для сложной функции, надо указать порядок, в котором вычисляются f_i, и способ формирования входов каждого автомата. Для единообразия обозначений входов удобно в список функций f_i включить независимые переменные x: $f_i = x_i$ $(i = 1, ..., n)$. Эти первые n "функций" не имеют параметров и вообще не вычисляются. Их объединяет с остальными f_i то, что они в равной степени могут подаваться на вход других автоматов. Далее используем такую нумерацию, изменив соответственно N.

Предполагаем, что порядок вычисления f_i согласован с выбранной нумерацией: если вычислены $f_1, ..., f_{i-1}$, то определены все входы f_i.

Каждому входу f_i соответствует "адрес входа" — номер j того f_j, выход которого подается на данный вход. Адрес q-го входа f_i обозначим $\mathrm{in}(i, q)$. Функция $\mathrm{in}(i, q)$ определена при $n < i \leqslant N$ и $1 \leqslant q \leqslant n_i$. Ее значения должны удовлетворять неравенству

$$\mathrm{in}(i, q) < i, \tag{3.3}$$

которое означает, что все входы f_i сформированы, если известны значения $f_1, ..., f_{i-1}$.

Каждому f_j можно поставить в соответствие совокупность "адресов выхода" — $\mathrm{out}(j)$. Она представляет собой совокупность таких пар (i, q), что $\mathrm{in}(i, q) = j$: i — это номер f_i, на q-й вход которого подается результат вычисления f_j. Множество $\mathrm{out}(j)$ можно обозначить как $\mathrm{in}^{-1}(j)$.

Вычисление f_i с распределением входных и выходных сигналов показано на рис. 3.3. Важный структурный элемент этой схемы — точка ветвления. Она возникает, если выход f_i подается на несколько различных входов, т.е. число элементов в $\mathrm{out}(i)$ больше единицы. Подчеркнем, что все сигналы, выходящие из точки ветвления, равны входящему в нее сигналу.

Работу описанной цепочки автоматов можно представить себе так: на ее вход подаются значения переменных x_i $(i = 1, ..., n)$. На каждом шаге вычисляется очередная функция f_j от своих аргументов $(j = n + 1, ..., n)$. На выходе при данных значениях параметров получаем сложную функцию $F(\alpha, x)$. Она представляет собой результат

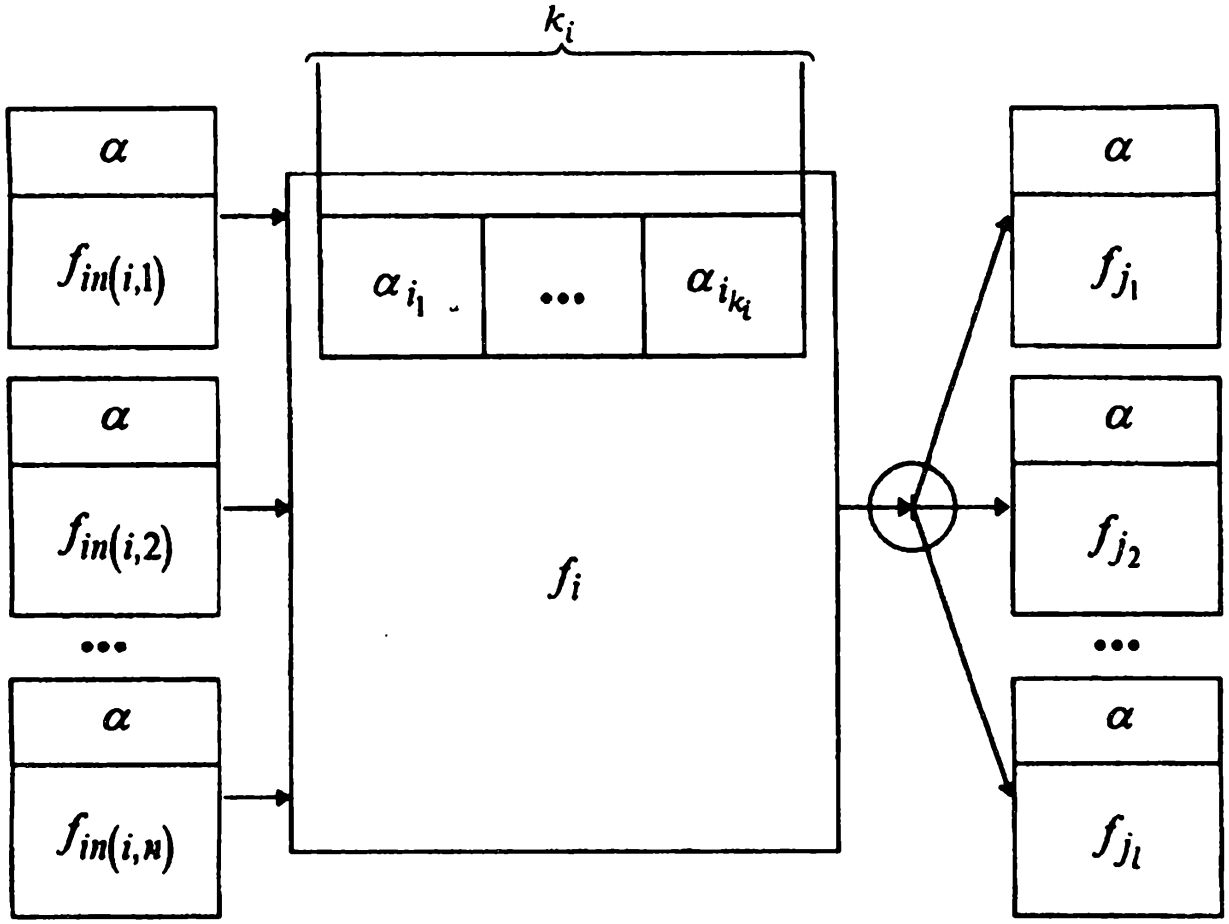


Рис. 3.3. Автомат, вычисляющий f_i, с указанием адресов входов, выхода и точки ветвления; q_p $(p = 1, \ldots, l)$ — номера тех входов f_{j_l}, на которые подается выходной сигнал f_i; при этом $\{(j_p, q_p) \mid p = 1, \ldots, l\} = \text{out}(i)$.

срабатывания последнего устройства цепочки, вычисляющего f_N (рис. 3.4).

Перейдем теперь к схеме, вычисляющей grad F — производные $\partial F/\partial x_1$, $\partial F/\partial \alpha_i$. Она состоит из двух цепочек — прямой и обратной. Цепочка прямого функционирования кроме вычисления f_i включает еще вычисление частных производных этих функций по их аргументам при тех же значениях аргументов.

Примем обозначения: $f_{i,r}$ — производная f_i по r-му аргументу (входу); $f_{i,\alpha j}$ — производная f_i по α_j $(f_{i,\alpha j} = \partial f_i/\partial \alpha_j)$. При прямом функционировании значения этих производных вычисляются, запоми-

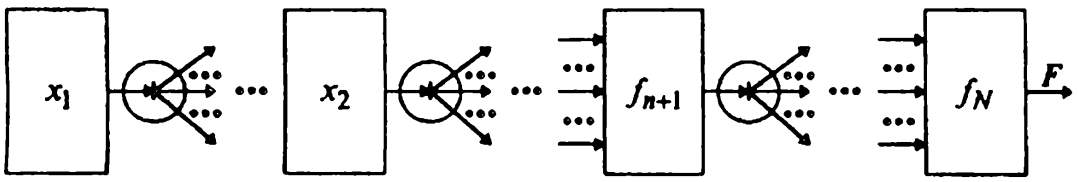


Рис. 3.4. Цепочка автоматов, вычисляющих сложную функцию $F(\alpha, x)$, входы f_i формируются из выходов предыдущих элементов цепочки.

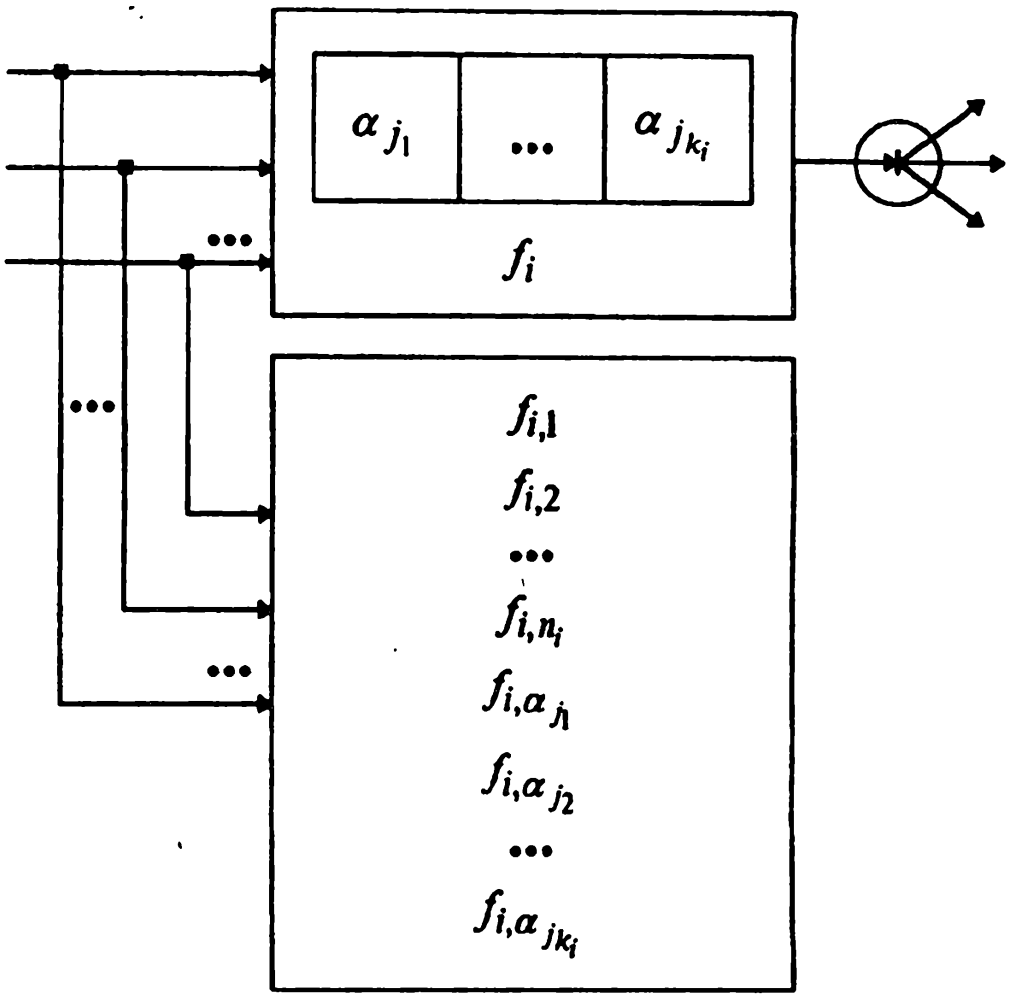


Рис. 3.5. Такт прямого функционирования с вычислением частных производных простых функций.

наются, но никуда не передаются — они будут использованы при обратном функционировании (рис. 3.5).

Обратное функционирование опишем сначала с помощью формул, а потом зададим схематически. Процесс этот заключается в последовательном вычислении цепочки двойственных f_i величин μ_i начиная с μ_N и кончая μ_1:

$$\mu_N = 1, \quad \mu_i = \sum_{(j,q)\in \mathrm{out}(i)} \mu_j f_{j,q}. \tag{3.4}$$

Здесь суммирование распространяется на все такие пары (j, q), что величина f_i в прямом функционировании посылается на q-й вход f_j — это и означает, что $(j, q) \in \mathrm{out}(i)$ или, что то же самое, $i = \mathrm{in}(j, q)$. Частные производные $f_{j,q}$ были вычислены в ходе прямого функционирования.

В ходе обратного функционирования вычисляются производные $\partial F/\partial x_i$ — это просто величины μ_i при $i = 1, \ldots, n$. Для вычисления $\partial F/\partial \alpha_j$ требуются дополнительные операции:

$$\frac{\partial F}{\partial \alpha_p} = \sum_j \mu_j f_{j,\alpha p}. \qquad (3.5)$$

Здесь суммирование ведется по всем тем j, для которых f_j зависит от α_p. Конечно, по существу эта формула совпадает с (3.4.) для μ_i.

При схемной реализации формул (3.4), (3.5) умножение проводят элементы, соответствующие f_j (будем обозначать их φ_j), а суммирование — сумматоры, соответствующие правым частям. Для формулы (3.4) сумматор обозначим Σ_i, на его выходе получаем μ_i, для формулы (3.5) — соответственно $\Sigma_{\alpha p}$ и $\partial F/\partial \alpha_p$. Такт обратного функционирования изображен на рис. 3.6 для $i < N$.

Случай $i = N$ отличается отсутствием сумматора Σ_N — на вход φ_N подается $\mu_N = 1$.

Элемент φ_i представляет набор из $n_i + k_i$ простых автоматов, каждый из которых имеет два входа, один выход и подает на выход произведение входов (см. рис. 3.6).

В качестве входов на каждый из таких автоматов подаются частная производная f_i, вычисленная в ходе прямого функционирования, и переменная обратного функционирования μ_i. Результат умножения подается на сумматор, определяемый аргументом, по которому взята частная производная f_i. Если это производная по j-му входу, то сигнал отправляется туда, откуда при прямом функционировании приходил вход, т.е. на $\Sigma_{\text{in}(i,j)}$. Если же это производная по параметру α_j, то сигнал посылается на сумматор $\Sigma_{\alpha j}$.

При обратном функционировании элементы φ_i, соответствующие независимым переменным ($i = 1, \ldots, n$), не нужны — для них остаются только сумматоры.

Заметим, что при обратном функционировании выходы φ_i соответствуют входам f_i, вход φ_i — выходу f_i, а входной сумматор Σ_i — точке ветвления на выходе f_i. Входные адреса соответствуют выходным, и обратно.

Пример. Пусть

$$F(\alpha, x_1, x_2) = f_4(\alpha, x_1, x_2, f_3(\alpha, x_1, x_2)). \qquad (3.6)$$

В определении F участвуют две функции: f_3 — от одного параметра и двух переменных и f_4 — от одного параметра и трех переменных. Напомним, что номера 1, 2 зарезервированы за независимыми переменными.

На схеме рис. 3.7 видно, что часть сумматоров Σ_i при обратном функционировании служит просто звеном, передающим единственный сигнал без всякого изменения. Это происходит тогда, когда выходной сигнал f_i передается при прямом функционировании по единственному

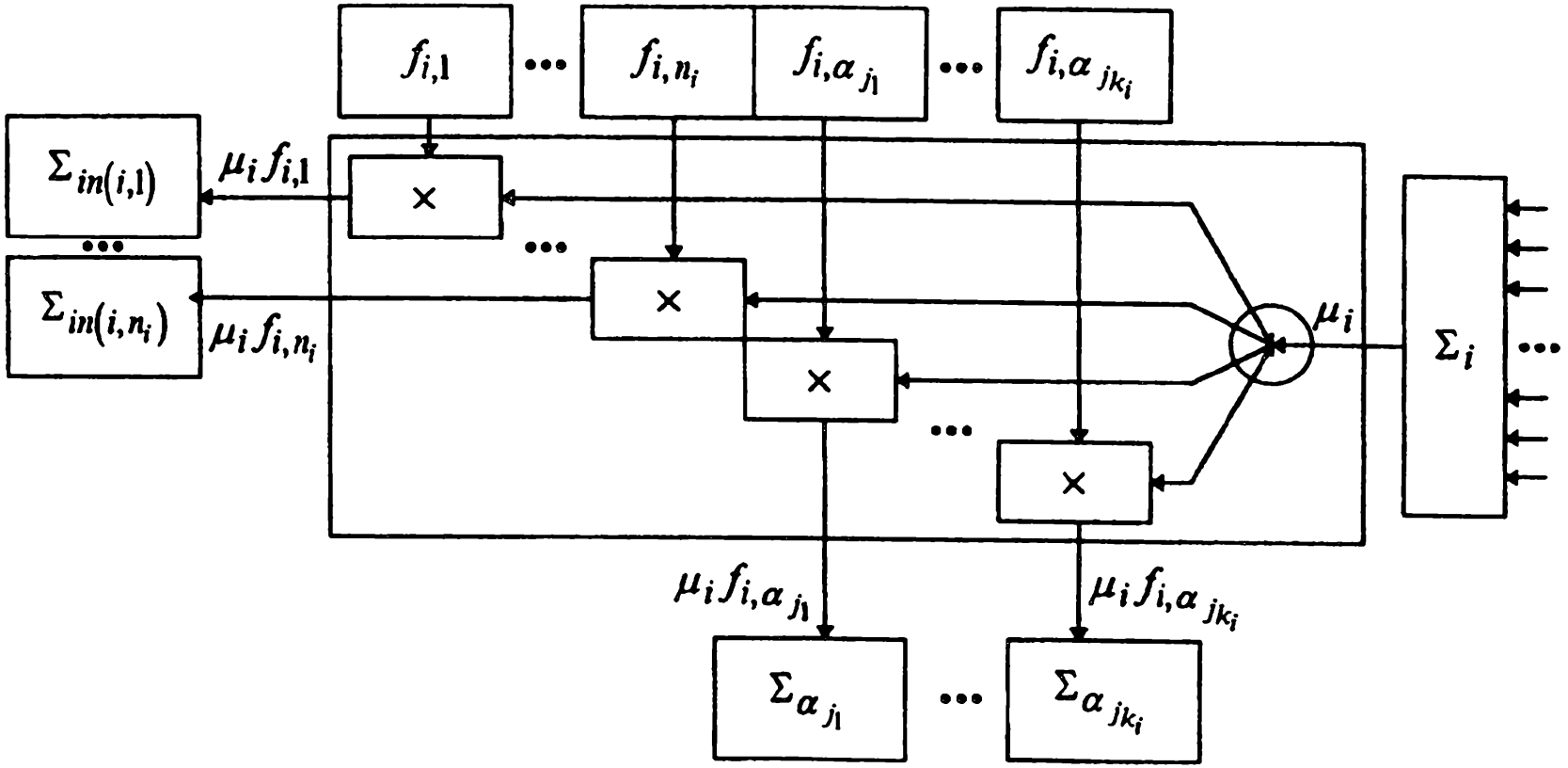


Рис. 3.6. Такт обратного функционирования: ⊠ — автомат, подающий на выход произведение входов; на входы Σ_i (справа) подаются произведения $\mu_j f_{j,q}$ от Φ_j при $(j, q) \in \mathrm{out}(i)$.

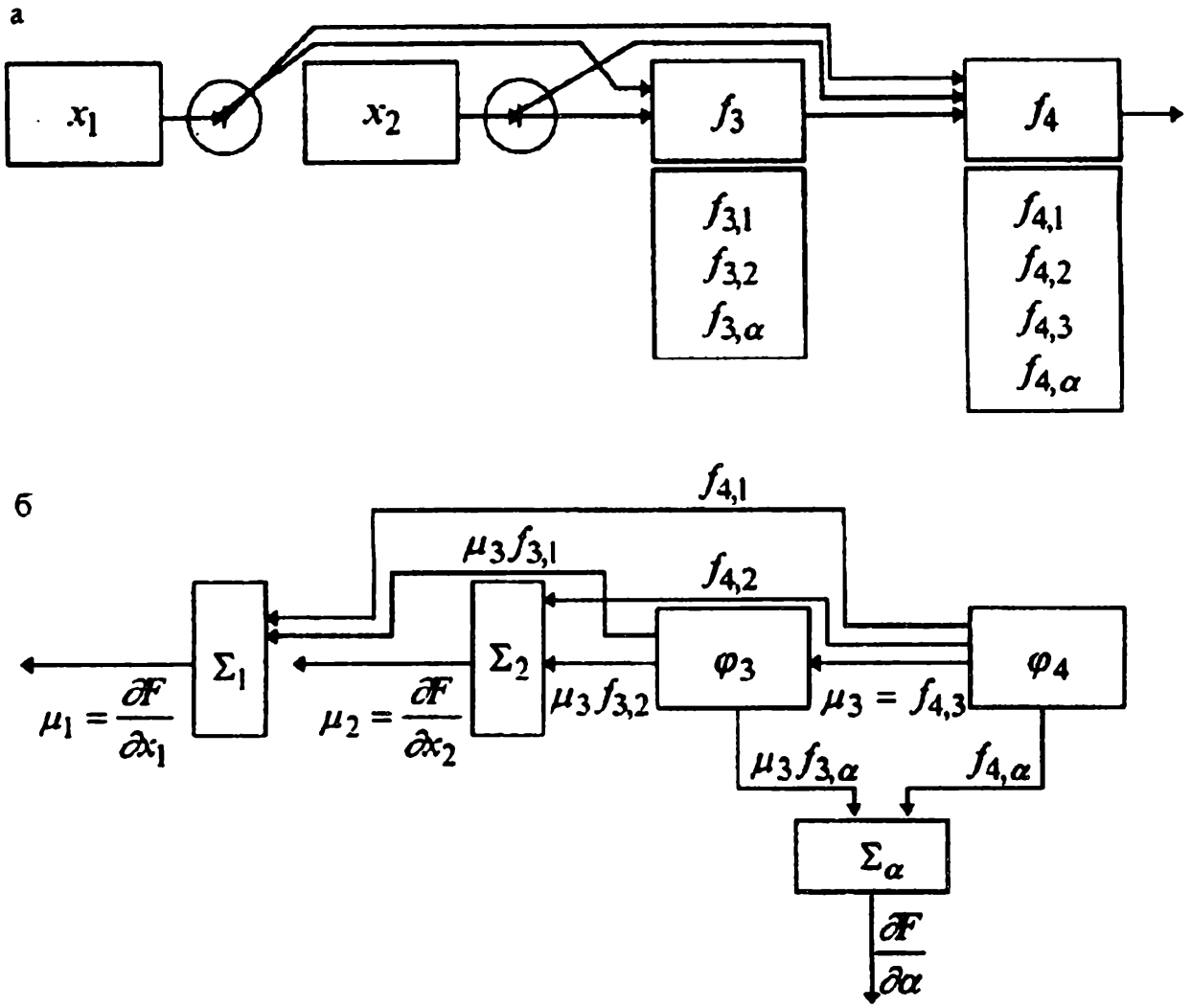


Рис. 3.7. Прямое (*а*) и обратное (*б*) функционирование при вычислении функции F (3.6) и ее производных.

адресу без ветвления. Тривиальное ветвление — тривиальный сумматор.

До сих пор речь шла о последовательном режиме вычисления сложной функции и соответственно последовательном обратном функционировании. В этом месте возможны усовершенствования, а именно — переход к параллельно-последовательным вычислениям. Если иметь в виду нашу основную задачу — обучение нейрокомпьютеров, то такой переход необходим.

Рассмотрим цепочку функций $f_1, \ldots, f_N$, участвующую в вычислении сложной функции F, и ее разбиения на подмножества. Подмножество цепочки будем задавать набором индексов J. Пусть задано разбиение цепочки на несколько (непересекающихся) множеств с наборами индексов $J_1, \ldots, J_p$ и для любых $i \in J_r$, $1 \leqslant j \leqslant n_i$ выполнено

$$\text{in}_i(j) \in J_q \text{ при некотором } q < r. \qquad (3.7)$$

Последнее означает, что для вычисления f_i ($i \in J_r$) требуются значения только тех функций, номера которых лежат в J_q ($q =$

$= 1, ..., r - 1$). Если такое разбиение возможно, то функции, номера которых лежат в одном J_r, можно вычислять независимо и, следовательно, одновременно. Схема вычисления F приобретает послойный вид — в одном слое лежат те f_i, номера которых принадлежат одному J_r. Вычисления в одном слое проводятся параллельно, между слоями есть последовательные связи. Аналогично, послойную структуру приобретает процесс обратного функционирования.

3. Двойственное функционирование сетей автоматов. Обобщения

Цель раздела — перейти от цепочек автоматов, описанных в предыдущем разделе, к более общим сетям. Первое обобщение — введение сумматоров перед каждым входом всех элементов. При обратном функционировании эти сумматоры перейдут в точки ветвления.

Сразу будем рассматривать послойно организованные сети. Каждый элемент вычисляет некоторую функцию входных сигналов и характеризуется парой номеров i, j (i — номер слоя, j — номер элемента в слое). Обозначим n_{ij} число входов i, j-го элемента, f_{ij} — саму функцию. Для каждой тройки i, j, q (i — номер слоя, j — номер элемента в слое, q — номер входа) определено множество адресов входа $\mathrm{in}(i, j, q)$. Множество $\mathrm{in}(i, j, q)$ — множество пар (p, r) (p — номер слоя, r — номер элемента в слое), причем при $p < i$ сигналы подаются с предыдущих слоев. На q-й вход i, j-го элемента подается сумма выходных значений f_{pr} p, r-х элементов

$$A_{ijq} = \sum_{(p,r)\in \mathrm{in}(i,j,q)} f_{pr}. \tag{3.8}$$

Суммирование проводится по всем (p, r) из $\mathrm{in}(i, j, q)$. На нулевом слое A не вычисляется, f_{0j} — значение j-го независимого переменного. На выходе i, j-го элемента получаем число

$$f_{ij} = f_{ij}(A_{ij1}, ..., A_{ijn_{ij}}). \tag{3.9}$$

Функционирование i, j-го элемента можно представить схемой (рис. 3.8).

Каждый элемент f_{ij} зависит от некоторого набора параметров. Эти параметры α_k могут быть общими для разных элементов, а могут быть и различными.

На последнем N-м слое стоит один элемент f_N. Позднее при рассмотрении нейронных сетей он превратится в вычислитель оценки.

Пусть на вход сети поданы переменные x_i ($i = 1, ..., n$) и определены параметры α. Тогда после срабатывания всех слоев сети получаем на выходе f_N. Это значение есть некоторая функция $F(x, \alpha)$. Ее частные производные можно найти, соответствующим образом органи-

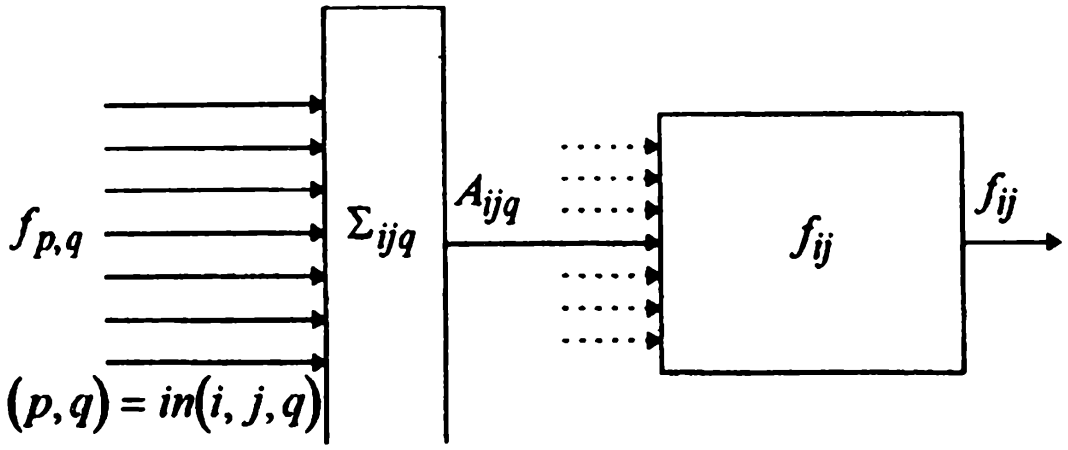


Рис. 3.8. Функционирование элемента с сумматорами перед входами.

зовав прямое и обратное функционирование. Для этого в ходе прямого функционирования кроме f_{ij} надо вычислять еще набор производных

$$f_{ij,q} = \frac{\partial f_{ij}(\alpha, A_{ij1}, \ldots)}{\partial A_{ijq}}, \; f_{ij,\alpha p} = \frac{\partial f_{ij}(\alpha, \ldots)}{\partial \alpha_p}.$$

При обратном функционировании сумматоры заменяются точками ветвления, точки ветвления — сумматорами. Вычисляются переменные обратного функционирования μ_{ij}, двойственные f_{ij}:

$$\mu_N = 1, \; \mu_{ij} = \sum_{\substack{r,s,l: \\ (i,j)\in \mathrm{in}(r,s,l)}} \mu_{rs} f_{rs,l} \; . \tag{3.10}$$

Суммирование распространяется здесь на все тройки (r, s, l) (r — номер слоя, s — номер элемента в слое, l — номер входа), для которых при прямом функционировании f_{ij} подается на входной сумматор l-го входа f_{rs}.

Множество $\mathrm{in}(i, j, q)$ обладает свойством

$$p < i \text{ для любых } (p, r) \in \mathrm{in}(i, j, q). \tag{3.11}$$

Поэтому для всех слагаемых в (3.10) $r > i$, вычисления суммы можно проводить внутри каждого слоя параллельно, а в последовательности слоев — от большего номера к меньшему. Каждое произведение, входящее в сумму (3.10), можно вычислять в соответствующем r, s-м элементе обратного функционирования. Схема такого элемента представлена на рис. 3.9. Он отличается от элемента, показанного на рис. 3.6, разветвлениями на выходах. Эти разветвления двойственны сумматорам при входах прямого функционирования. Обозначение сумматоров обратного функционирования отличается от сумматоров прямого штрихом (Σ' вместо Σ).

При вычислении μ_{ij}, спустившись до нулевого слоя, получаем

$$\mu_{0j} = \frac{\partial F}{\partial x_j}. \tag{3.12}$$

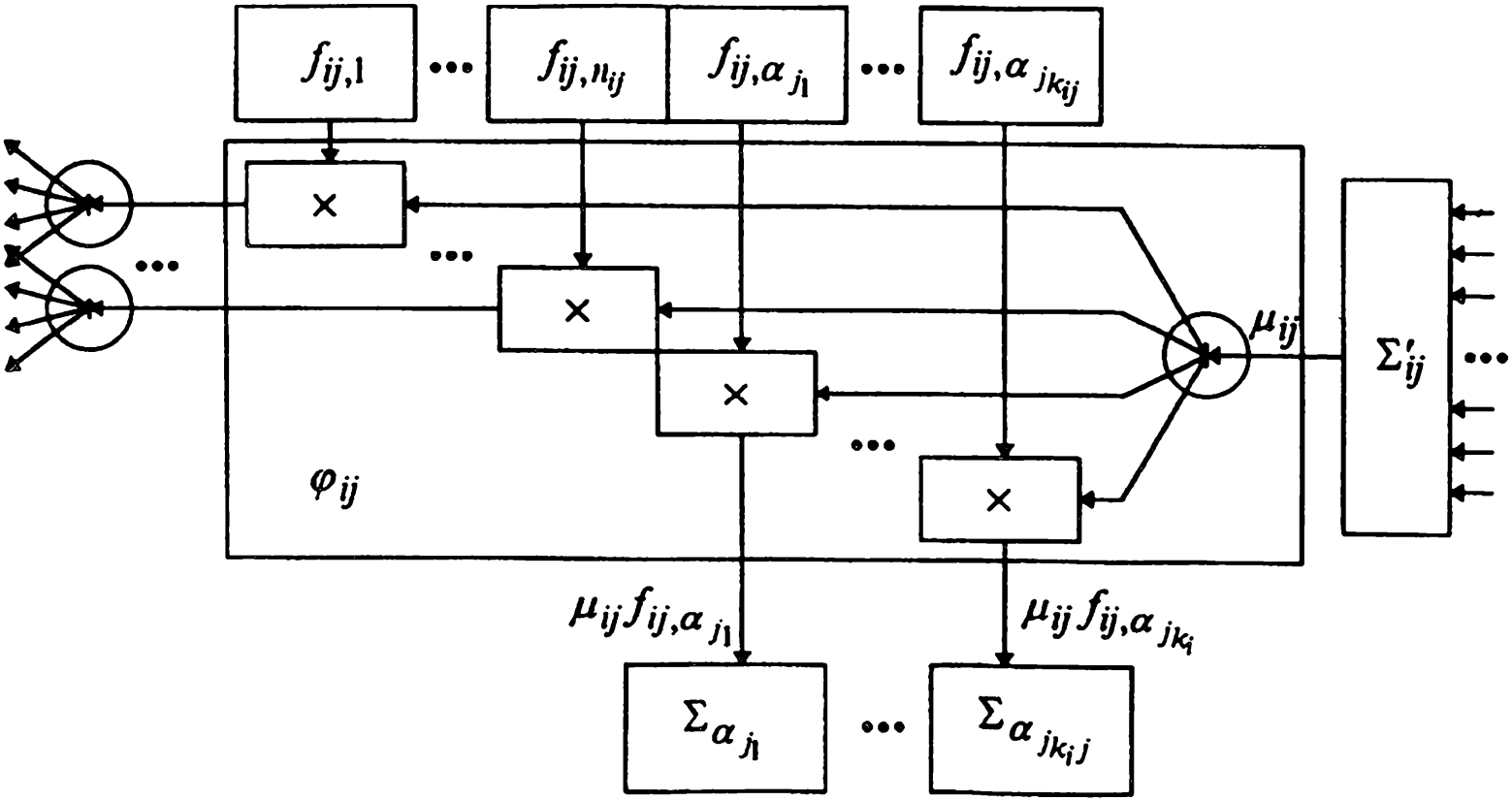


Рис. 3.9. Схема элемента обратного функционирования Φ_{ij} при наличии сумматоров при входах прямого функционирования: на сумматор Σ'_{ij} подаются $\mu_{rs} f_{rs,l}$ от таких Φ_{rs}, что $(i, j) \in \text{in}(r, s, l)$; получаемые на выходе (слева) сигналы $\mu_{ij} f_{ij,k}$ передаются через точки ветвления ко всем сумматорам Σ'_{pq}, для которых $(p, q) \in \text{in}(i, j, k)$.

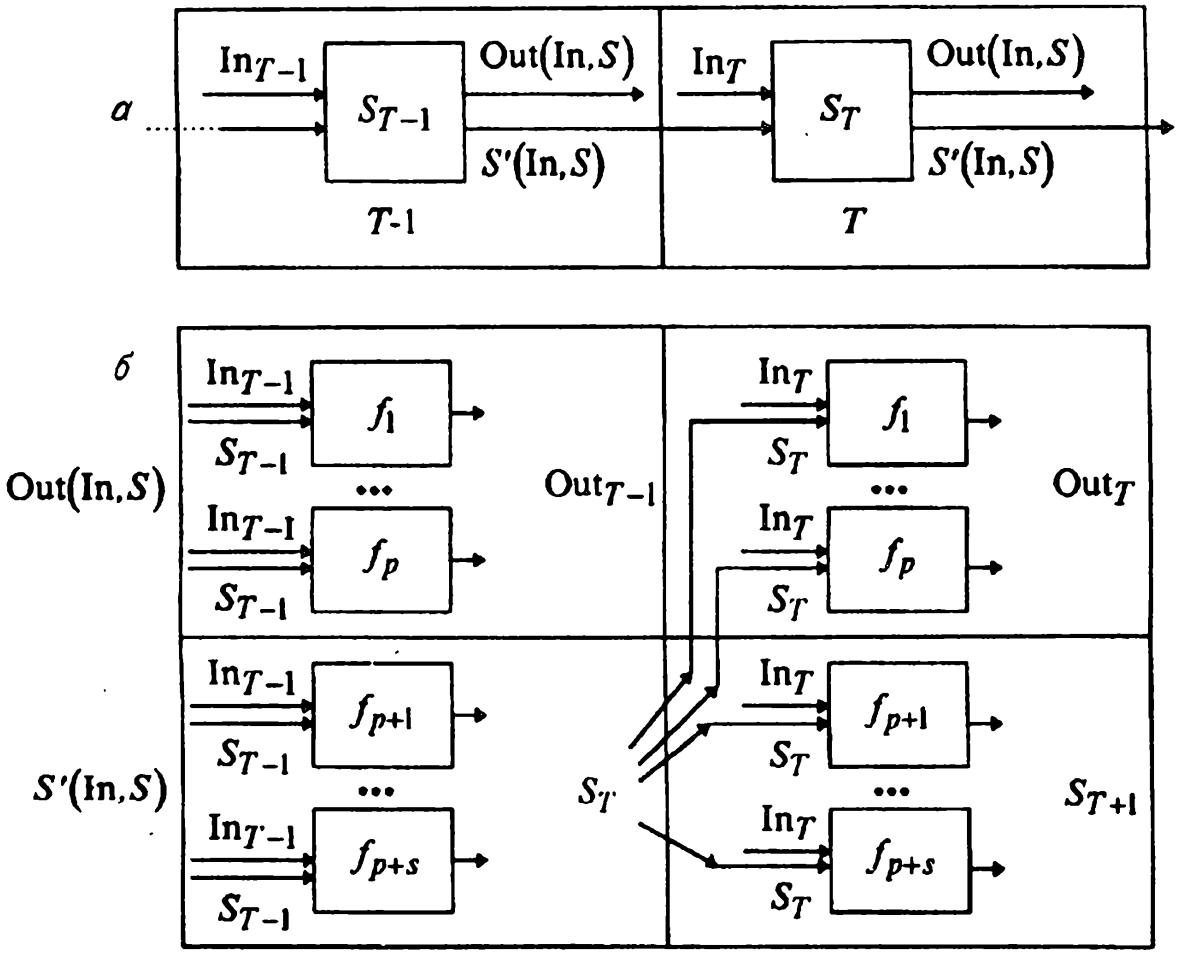


Рис. 3.10. Функционирование автомата в последовательные моменты времени. *а* — в общем виде; *б* — в разбиении на более простые автоматы; In — $(x_1, ..., x_n)$, S — $(x_{n+1}, ..., x_{n+s})$; Out — $f_i(\text{In}, S)$ $(i = 1, ..., p)$; S' — $f_i(\text{In}, S)$ $(i = p + 1, ..., p + s)$; T — дискретное время.

Произведение $\partial F/\partial \alpha_j$ накапливается в специальных сумматорах так же, как и в предыдущем разделе.

Обратимся теперь к проблеме обучения сети автоматов, синхронно функционирующих в дискретном времени. Для обучения желательно вычислять градиент ошибки — частные производные этой функции по параметрам. Одновременно могут быть вычислены частные производные этой функции по входным сигналам. Об использовании таких производных будет рассказано в следующем разделе. Опишем сначала функционирование сети, а потом построим двойственный процесс.

Каждый автомат имеет несколько входов (n), несколько выходов (p) и конечный набор (s) параметров состояния. Он вычисляет $s + p$ функций от $n + s$ переменных. Аргументы этих функций — входные сигналы (их n) и текущие параметры состояния (их s). Значение функций — выходные сигналы (их p) и параметры состояния на следующем шаге (их s). Каждый такой автомат можно представить как систему из $s + p$ более простых автоматов (рис. 3.10). Простые автоматы вычисляют по одной функции от $n + s$ переменных. Смена состояний достигается за счет того, что часть значений этих функций

на следующем шаге становится аргументами — так соединены автоматы (см. рис. 3.10).

Таким образом, без потери общности можно рассматривать сеть автоматов как набор устройств, каждое из которых вычисляет функцию нескольких переменных $f_i(\alpha, A_{i1}, ..., A_{in})$ $(i = 1, ..., N)$.

Каждая из этих функций может зависеть от параметров $\alpha = (\alpha_1, ..., \alpha_k)$. Конечно, в конкретных приложениях f_i зависит только от некоторых из α_j, и это может привести к существенным упрощениям, но пока ограничимся общим случаем.

На каждом шаге функционирования всем функциям f_i подаются входные сигналы A_{ij} $(j = 1, ..., n)$. Значение A_{ij} называем j-м входным сигналом f_i. Эти входные сигналы снимаются с входных сумматоров Σ_{ij}. На входные сумматоры могут поступать сигналы двух видов — значения некоторых f_k на предыдущих шагах функционирования и внешние сигналы.

Обозначим T текущее дискретное время. Оно может принимать целые значения. Величины, относящиеся ко времени T, будем отмечать индексом T.

В каждый момент времени системе подается входной сигнал $x_T = (x_{1T}, ..., x_{nT})$. Входным сумматорам Σ_{ij} соответствует набор адресов входа $\mathrm{in}(i, j)$. Он состоит из двух множеств: $\mathrm{in}_I(i, j)$ — номера тех x_{pT}, которые подаются на Σ_{ij}, $\mathrm{in}_S(i, j)$ — номера тех f_i, значения которых на предыдущем шаге f_{iT-1} подаются на Σ_{ij}. В соответствии с введенными обозначениями значение на T-м шаге A_{ijT} равно

$$A_{ijT} = \sum_{p \in \mathrm{in}_I(i,j)} x_{pT} + \sum_{q \in \mathrm{in}_S(i,j)} f_{qT-1}. \qquad (3.13)$$

Индексы I и S у in соответствуют внешнему входу (In) и состоянию (S).

Возможно *обобщение*, состоящее в том, что на входные сумматоры в момент T подаются сигналы не только из набора текущих внешних входных x_T и предыдущих выходных f_{T-1}, но и из x_{T-1}, ...; f_{T-2}, ... Формально оно строится так. Каждому входному сумматору Σ_{ij} ставится в соответствие множество адресов входа $\mathrm{in}(i, j)$, состоящее из множеств $\mathrm{in}_{I0}(i, j)$, $\mathrm{in}_{I1}(i, j)$, ..., $\mathrm{in}_{S1}(i, j)$, $\mathrm{in}_{S2}(i, j)$, ... Величина j-го входа f_i на шаге T формируется так:

$$A_{ijT} = \sum_{r=0}^{R_I} \sum_{p \in \mathrm{in}_{Ir}(i,j)} x_{pT-r} + \sum_{r=0}^{R_S} \sum_{q \in \mathrm{in}_{Sr}(i,j)} f_{qT-r}. \qquad (3.14)$$

Верхние пределы суммирования по r — времена запаздывания.

Еще одно обобщение. Сама система автоматов может иметь послойную структуру — состоять из поочередно срабатывающих подсистем. Вспомните функционирование нейронной сети:

нейроны → синапсы → сумматоры → нейроны...

Такт функционирования такой системы разбивается на несколько более мелких единиц — тактов подсистем (слоев). Сигнал на Σ_{ij} может подаваться не только с предыдущего такта системы, но и с предыдущих слоев того же такта. Формализуем и эту ситуацию. У f и A появится еще один индекс — номер слоя.

Итак, системы с дополнительной послойной структурой (последовательно-параллельные системы) также составлены из устройств, вычисляющих функции нескольких переменных. Эти устройства объединены в слои. Каждая функция задается двумя номерами: f_{ij} (i — номер слоя, j — номер функции в слое). Каждому входу соответствует сумматор Σ_{ijk} (k — номер входа). Переменная, выдаваемая сумматором Σ_{ijk} на k-й вход f_{ij}, обозначается A_{ijk}. Каждому сумматору Σ_{ijk} соответствует набор адресов $\mathrm{in}(i, j, k)$, составленный из множеств $\mathrm{in}_{I0}(i, j, k)$, $\mathrm{in}_{I1}(i, j, k)$, ..., $\mathrm{in}_{S0}(i, j, k)$, $\mathrm{in}_{S1}(i, j, k)$, ... Элементы $\mathrm{in}_{Ir}(i, j, k)$ — номера внешних входных сигналов с шага $T - r$, подаваемых на вход Σ_{ijk} в момент T. Элементы $\mathrm{in}_{Sr}(i, j, k)$ — пары индексов p, q, для которых выходной сигнал f_{pqT-r} (момент — $T - r$) подается на вход Σ_{ijk} в момент T. Основное отличие от предыдущего случая — наличие $\mathrm{in}_{S0}(i, j, k)$. Элементы этого множества должны удовлетворять неравенству:

$$\text{если } (p, r) \in \mathrm{in}_{S0}(i, j, k), \text{ то } p < i. \tag{3.15}$$

Оно означает, что сигналы, берущиеся в Σ_{ijk} на том же такте системы, должны приходить с предшествующих слоев.

Аналог (3.14) теперь принимает вид

$$A_{ijT} = \sum_{r=0}^{R_I} \sum_{p \in \mathrm{in}_{Ir}(i,j,k)} x_{pT-r} + \sum_{r=0}^{R_S} \sum_{q \in \mathrm{in}_{Sr}(i,j,k)} f_{pqT-r}. \tag{3.16}$$

Будем далее в этом разделе рассматривать сети и с запаздыванием, и с послойной организацией. При необходимости будем возвращаться к менее громоздким частным случаям (3.13), (3.14).

Рассмотрим общую сеть автоматов в режиме периодического функционирования. В начальный момент ($T = 1$) она должна обратиться за входными сигналами к предыдущим тактам функционирования ($T = 0, -1, \ldots$), а они отсутствуют. Поэтому работа сети в режиме периодического функционирования предполагает фазу инициации — нужно задать числа, которые будут в соответствии с (3.16) посылаться на входы Σ_{ijk}. Это — числа x_{pl}, f_{pqm} ($l = 0, -1, \ldots, 1 - R_I$; $m = 0$,

$-1, ..., 1-R_S$). Даже в отсутствие запаздывания нужно задать f_{pqm} $(m = 0)$.

Существуют два способа инициации.

1. *Стандартная инициация.* Неизвестные сигналы с отсутствующих предыдущих тактов определяются некоторым стандартным образом. Чаще всего они полагаются нулевыми.

2. *Обучаемые параметры инициации.* Сигналы, соответствующие недостающим предыдущим тактам, рассматриваются как дополнительные подстраиваемые параметры. Они участвуют в процессе обучения наравне с другими параметрами.

Итак, пусть сеть автоматов инициирована — заданы недостающие величины с предыдущих шагов. Всюду далее они называются параметрами инициации. Потом подаются входные сигналы x_1, после чего последовательно срабатывают все слои сети, затем подаются входные сигналы x_2 и т.д. до последнего срабатывания при $T = \Theta$. На этом цикл функционирования ("акт" в отличие от такта) кончается. Теперь нужно оценить функционирование сети и вычислить производные оценки по всем подстроечным параметрам.

Оценка цикла функционирования есть некоторый функционал от входных и выходных сигналов. В качестве выходных могут использоваться любые f_{ijT} $(T = 1, ..., \Theta)$. Рассмотрим все f_{ijT} $(T = 1, ..., \Theta)$ как выходные сигналы. Возможно упрощение из-за того, что не все f_{ijT} используются как выходные сигналы. Оно приведет к уменьшению слагаемых в дальнейших формулах.

Оценка — функция от всей совокупности переменных x_{iT}, f_{ijT}: $H = H(\{x_{iT}\}, \{f_{ijT}\})$.

Функционирование сети с вычислением оценки можно представлять себе как вычисление сложной функции, описанное в предыдущем разделе. Вычисление происходит послойно, входные независимые переменные — это параметры инициации и входные сигналы x_{iT}.

Автомат, вычисляющий f_{ij}, в разные моменты времени T можно представить как разные экземпляры одного автомата, существующие одновременно. При этом временное функционирование сети представляется как последовательное срабатывание нескольких экземпляров сети. Такое одновременное представление удобно. Номером слоя, к которому принадлежит f_{ijT}, становится пара (i, T) со следующим определением порядка:

$$(i_1, T_1) > (i_2, T_2), \text{ если } T_1 > T_2, \text{ или } (T_1 = T_2 \text{ и } i_1 > i_2). \quad (3.16)$$

Знак ... > ... означает "... следует за ... в функционировании".

Кроме функций f_{ijT} в новом слое, следующем за всеми этими функциями, появляется автомат f_H, вычисляющий функционал оценки. На его входы подаются x_{iT}, f_{ijT} $(T = 1, ..., \Theta)$, на выходе получаем оценку H.

Представив период функционирования сети таким образом, можно воспользоваться с соответствующими преобразованиями формулами (3.10) для описания обратного функционирования и вычисления производных H по параметрам α, параметрам инициации и входным сигналам x_{iT}. Каждая из производных будет накапливаться в соответствующем сумматоре.

Итак, цикл функционирования и его оценивание для сети автоматов представлены как вычисление сложной функции послойно организованной сетью автоматов. При этом временные такты располагаются друг за другом пространственно, аналогично последовательным слоям. Обратное функционирование проходит в обратном направлении — от f_H к выходам, в результате в специальных сумматорах накапливаются значения градиента оценки.

Во всех конкретных случаях необходима лишь конкретизация приведенных выше формул и схем.

Формула (3.10) является главной для организации двойственного функционирования. Она служит далеко идущим обобщением формулы Уидроу (см. гл. 1) для адаптивных сумматоров.

4. Смысл двойственных переменных

Переменные обратного функционирования μ получаются как вспомогательные при вычислении производных сложной функции не случайно. Они постоянно возникают в задачах оптимизации и являются множителями Лагранжа.

Мы вводим μ исходя из правил дифференцирования сложной функции. Возможен другой путь, связанный с переходом от функции Лагранжа к функции Гамильтона. Изложим его и параллельно получим ряд дальнейших обобщений.

Для всех сетей автоматов, встречавшихся нам в предыдущих разделах, можно выделить три группы переменных: внешние входные сигналы $x_{\dots}$, переменные функционирования — значения $f_{\dots}$ и переменные обучения $\alpha_{\dots}$ (многоточиями заменяются различные наборы индексов). Объединим их в две группы: вычисляемые величины $y_{\dots}$ — значения f и задаваемые — $\beta_{\dots}$. Упростим индексацию, перенумеровав $f_{\dots}$ и $\beta_{\dots}$ натуральными числами: $f_1, \dots, f_N$; $\beta_1, \dots, \beta_M$.

Пусть функционирование системы задается набором из N уравнений

$$\psi_i(y_i, \dots, y_N, \beta_1, \dots, \beta_M) = 0. \tag{3.17}$$

Простейший пример такого уравнения —

$$\psi_i = f_i(\beta, y_{i_1}, \dots, y_{i_{n_i}}) - y_i,\ i_j = \mathrm{in}(i, j) < i. \tag{3.18}$$

Предполагается, что система уравнений (3.17) задает способ вычисления y_i.

Пусть имеется целевая функция (Лагранжиан) $H(y_1, ..., y_N, \beta_1, ..., \beta_M)$. Эта функция зависит от β_i и явно, и неявно — через переменные функционирования y. Если представить, что уравнения (3.17) разрешены относительно всех $y_i (y_i = y_i(\beta))$, то H можно представить как функцию от β:

$$H = H_1(\beta) = H(y_1(\beta), ..., y_N(\beta), \beta), \tag{3.19}$$

где β — вектор с компонентами β_i.

Для задачи обучения требуется найти производные $\partial H_1(\beta)/\partial\beta_i$. Непосредственно и явно это сделать трудно. Поступим по-другому. Введем новые переменные $\mu_1, ..., \mu_N$ (множители Лагранжа) и производящую функцию W:

$$W(y, \beta, \mu) = H(y, \beta) + \sum_i \mu_i \psi_i(y, \beta). \tag{3.20}$$

В функции W аргументы y, β и μ — независимые переменные. Уравнения (3.17) можно записать как

$$\frac{\partial W}{\partial \mu_i} = 0. \tag{3.21}$$

Заметим, что для тех y, β, которые удовлетворяют уравнениям (3.17), при любых μ

$$W(y, \beta, \mu) \equiv H(y, \beta), \tag{3.22}$$

т.е. для истинных значений переменных функционирования y при данных β функция $W(y, \beta, \mu)$ совпадает с оценкой.

Попытаемся подобрать такую зависимость $\mu_i(\beta)$, чтобы, используя (3.22), получить для D_i наиболее простые выражения[1]. На многообразии решений (3.21)

$$D_i = \frac{\partial H}{\partial \beta_i} + \sum_f \frac{\partial H}{\partial y_j}\frac{\partial y_j}{\partial \beta_i} = \frac{\partial W}{\partial \beta_i} + \sum_f \frac{\partial W}{\partial y_j}\frac{\partial y_j}{\partial \beta_i} + \sum_j \frac{\partial W}{\partial \mu_j}\frac{\partial \mu_j}{\partial \beta_i} =$$

$$= \sum_j \left(\frac{\partial H}{\partial y_j} + \sum_k \mu_k \frac{\partial \psi_k}{\partial y} \right) \frac{\partial y_j}{\partial \beta_i} + \sum_j \mu_j \frac{\partial \psi_j}{\partial \beta_i} + \frac{\partial H}{\partial \beta_i}. \tag{3.23}$$

Если выбрать такие μ_k, что слагаемые в первой сумме выражения (3.23) обратятся в нуль, то формула для D_i резко упростится. Положим поэтому

---

[1]Произвол в определении $\mu_i(\beta)$ надо использовать наилучшим образом — все равно от него придется избавляться, доопределяя зависимости.

$$\frac{\partial H}{\partial y_j} + \sum_k \mu_k \frac{\partial \psi_k}{\partial y_j} = 0. \qquad (3.24)$$

Это — система уравнений для определения μ_k ($k = 1, \ldots, N$). Если μ_k определены согласно (3.24), то получим

$$D_i = \frac{\partial H}{\partial \beta_i} \sum_j \mu_j \frac{\partial \psi_i}{\partial \beta_i} \qquad (3.25)$$

— в точности те выражения (с точностью до переобозначения индексов), которыми мы пользовались при поиске производных сложных функций, вычисляемых сетью автоматов. В расчетах мы брали явное описание функционирования, что соответствует выбору ψ_i в виде (3.18). Метод множителей Лагранжа допускает и менее явные описания сетей.

5. Несколько заключительных замечаний о двойственности

В математическом фольклоре бытует такой тезис: сложная задача оптимизации может быть решена только при эффективном использовании двойственности. Мы определили обучение нейронных сетей как задачу оптимизации и потратили немало места на описание конструкций, двойственных нейронной сети. Что же позволяет сделать использование двойственности? Попытаемся в нескольких пунктах ответить на этот вопрос.

1. Двойственные сети позволяют быстро в последовательно-параллельном режиме вычислять градиент функции оценки как неявной функции параметров и входов.

2. Если требуется, двойственная сеть без изменений своей структуры и параметров может вычислять производные по параметрам сети и входам для любой другой функции выходных сигналов.

И это все? Да, пока все. Отметим еще только, что для чисто коннекционистских моделей цепи обратного функционирования используют ту же структуру связей между элементами и те же веса α_{ij}, что и при прямом функционировании. Это существенно упрощает реализацию таких сетей. Именно использование при обратном функционировании той же системы связей, что и при прямом, имеют в виду, когда говорят, что сеть сама вычисляет направление наискорейшего обучения.

Что же касается явных формул для конкретных сетей, то их очень легко получить в качестве упражнения (см. также [50, 54]).

6. Особенности задачи оптимизации, возникающей при обучении нейронных сетей

Предлагается рассматривать обучение нейронных сетей как задачу оптимизации. Это означает, что весь мощный арсенал методов оптимизации может быть испытан для обучения. Так и видится: нейрокомпьютеры наискорейшего спуска, нейрокомпьютеры Ньютона, Флетчера и т.п. — по названию метода нелинейной оптимизации.

Существует, однако, ряд специфических ограничений. Они связаны с огромной размерностью задачи обучения. Число параметров может достигать 10^8 — и даже более. Уже в простейших программных имитаторах на персональных компьютерах подбирается $10^3 - 10^4$ параметров.

Из-за высокой размерности возникает два требования к алгоритму:

1. *Ограничение по памяти.* Пусть n — число параметров. Если алгоритм требует затрат памяти порядка n^2, то он вряд ли применим для обучения. Вообще говоря, желательно иметь алгоритмы, которые требуют затрат памяти порядка Kn, $K = \text{const}$.

2. *Возможность параллельного выполнения* наиболее трудоемких этапов алгоритма, и желательно нейронной сетью. Если какой-либо особо привлекательный алгоритм требует памяти порядка n^2, то его все же можно использовать, если с помощью анализа чувствительности и, возможно, контрастирования сократить число обучаемых параметров до разумных пределов.

Еще два обстоятельства связаны с нейрокомпьютерной спецификой.

1. Обученный нейрокомпьютер должен с приемлемой точностью решать все тестовые задачи (или, быть может, почти все с очень малой частью исключений). Поэтому задача обучения становится по существу *многокритериальной* задачей оптимизации: надо найти точку общего минимума большого числа функций. Обучение нейрокомпьютера исходит из гипотезы о существовании такой точки. Основания гипотезы — очень большое число переменных и сходство между функциями. Само понятие "сходство" здесь трудно формализовать, но опыт показывает, что предположение о существовании общего минимума или, точнее, точек, где значения всех оценок мало отличаются от минимальных, часто оправдывается.

2. Обученный нейрокомпьютер должен иметь возможность приобретать *новые навыки без утраты старых.* Возможно более слабое требование: новые навыки могут сопровождаться потерей точности в старых, но потеря не должна быть особо существенной, а качественные изменения должны быть исключены. Это означает, что в достаточно большой окрестности найденной точки общего минимума оценок их значения незначительно отличаются от минимальных. Мало того, что должна быть найдена точка общего минимума, так она еще должна лежать в достаточно широкой низменности, где значения всех мини-

мизируемых функций близки к минимуму. Для решения этой задачи нужны специальные средства.

Итак, имеем четыре специфических ограничения, выделяющих обучение нейрокомпьютера из общих задач оптимизации: астрономическое число параметров, необходимость высокого параллелизма при обучении, многокритериальность решаемых задач, необходимость найти достаточно широкую область, в которой значения всех минимизируемых функций близки к минимальным. В остальном это просто задача оптимизации, и многие классические и современные методы достаточно естественно ложатся на структуру нейронной сети.

В следующих разделах описаны основные методы, реально используемые нами в программном обеспечении для PC.

7. Учет ограничений при обучении

Для параметров сети возможны ограничения простейшего вида

$$\alpha_{i\,\max} \geqslant \alpha_i \geqslant \alpha_{i\,\min}.$$

Они вводятся из разных соображений: чтобы избежать слишком крутых или, наоборот, слишком пологих характеристик нейронов, чтобы предотвратить появление слишком больших коэффициентов усиления сигнала на синапсах и т.п.

Учитывать ограничения можно, например, методом штрафных функций либо методом проекций.

Использование метода штрафных функций означает, что в оценку добавляются штрафы за выход параметров из области ограничений. Это не требует никаких дополнительных изменений сети — меняется только функция оценки, а в градиенты H вводятся производные штрафных функций.

Проективный метод означает, что если сети предлагается изменение параметров

$$\alpha_1 := W_i$$

и W_i для некоторых i выходит за ограничения, то следует положить

$$\alpha_i := \begin{cases} W_i, \text{ если } \alpha_{i\,\max} \geqslant W_i \geqslant \alpha_{i\,\min}; \\ \alpha_{i\,\max}, \text{ если } W_i > \alpha_{i\,\max}; \\ \alpha_{i\,\min}, \text{ если } W_i < \alpha_{i\,\min}. \end{cases}$$

После этого обучение (т.е. минимизация H_{pg}) продолжается.

В нашем опыте проективный метод не приводил к каким-либо затруднениям. Обращение со штрафными функциями было менее успешным.

Далее будем полагать, что ограничения учтены одним из указанных методов. Будем говорить об обучении как о безусловной минимизации, подразумевая либо наличие штрафных функций, либо опи-

санную проекционную поправку, вносимую во все изменения параметров.

8. Одномерная оптимизация

Все пошаговые методы оптимизации состоят из двух важнейших частей: выбора направления и выбора шага в данном направлении. Методы одномерной оптимизации дают эффективный способ для выбора шага.

Использование оптимального шага в задачах многокритериальной оптимизации невозможно, если критерии рассматриваются "по одному" — сначала улучшается значение одного, потом другого и т.д. Необходим синтез обобщенного (интегрального) критерия. Простейший вариант — суммирование всех. Чуть сложнее — суммирование с весами. При ограничениях по памяти, а также в тех случаях, когда получение значения каждого критерия требует заметных затрат, возможна *постраничная организация* [54] — выбирается достаточно богатый набор критериев (*страница* примеров), из них формируется интегральный (оценка страницы) и ведется его оптимизация.

Для обучения по отдельным примерам, т.е. без постраничной организации, методы одномерной оптимизации неприменимы — они слишком эффективны, поэтому давая заметное улучшение на одном обучающем примере, они часто разрушают навык решения предыдущих. При постраничном обучении такого обычно не происходит. Нам неоднократно встречались публикации, в которых утверждалось, что оптимизация шага ухудшает обучение. При ближайшем рассмотрении оказывалось, что авторы этих работ учили нейронные сети по отдельности на каждом примере...!

Кроме того, при использовании интегральных критериев (оценка страницы, сумма оценок по всему задачнику и т.п.) оптимизация шага необходима. Сказанное является хорошо известным общим местом в теории оптимизации, и мы уделили этому внимание только из-за распространенности ошибок в данных вопросах при обучении нейронных сетей (приступая к решению задачи оптимизации, полезно изучить опыт предшественников, делавших то же самое до нейросетевого "бума").

Из огромного количества алгоритмов одномерной оптимизации опишем один. Он хорошо показал себя в обучении нейронных сетей.

Алгоритм использует процедуру вычисления минимизируемой функции (функции ошибки на странице $H = H_{pg}$). Перед началом работы он получает начальную точку — вектор α^0, направление движения — вектор s, начальный шаг d_0 и информацию о том, *является ли s безусловным направлением спуска.* Последнее означает априорную информацию о том, существует ли такое $\varepsilon > 0$, что при $0 < h < \varepsilon$

$$H(\alpha^0) > H(\alpha^0 + hs).$$

Если известно, что существует, то s — безусловное направление спуска и при минимизации разумно ограничиваться случаем $h > 0$. Если неизвестно, то может оказаться полезным и выбор $h < 0$.

Безусловным направлением спуска является антиградиент H.

Кроме того, алгоритм может получать вычисленные ранее значения $H(\alpha^0)$, $H(\alpha^0 + h_0 s)$. Если какое-либо из них или оба неизвестны, то эти значения вычисляются и сравниваются.

Если $H(\alpha^0) \geqslant H(\alpha^0 + h_0 s)$, то выполняется процедура Increase (увеличение шага). В противном случае приходим к процедуре Decrease (уменьшение шага).

Обратите внимание на знак неравенства в условии: $\geqslant$, а не $>$. Это существенно при программной реализации — шаг может оказаться слишком малым, а улучшение — незаметным. Для проверки шаг надо увеличивать, а не уменьшать.

Increase. Процедура получает α^0, s, h_0 и значения минимизируемой функции $H(\alpha^0)$, $H(\alpha^0 + h_0 s)$. При этом $H(\alpha^0) \geqslant H(\alpha^0 + h_0 s)$. Используется процедура вычисления минимизируемой функции.

В процедуре Increase используются константа возрастания r, три переменных шага $h_1 < h_2 < h_3$ и три — для соответствующих значений функции H_1, H_2, H_3. Получив входные данные, полагаем

$$h_1 := 0,\ h_2 := h_0,\ h_3 := rh_0,\ H_1 := H(\alpha^0),\ H_2 := H(\alpha^0 + h_2 S).$$

Вычисляем $H_3 = H(\alpha^0 + h_3 s)$. Если $H_3 > H_2$, то переходим к параболическому поиску — процедура Parabola. В противном случае проверяем, образует ли тройка аргументов $h_{1,2,3}$ и значений $H_{1,2,3}$ выпуклую комбинацию:

$$H_2 < \frac{h_3 - h_2}{h_3 - h_1} H_1 + \frac{h_2 - h_1}{h_3 - h_1} H_3. \tag{3.26}$$

Если это верно, то следующий шаг делается в вершину параболы, график которой проходит через точки (h_i, H_i) $(i = 1, 2, 3)$; $H_1 = ah_i^2 + bh_i^2 + c$ $(i = 1, 2, 3)$. Вычисляем

$$W = -\frac{b}{2a} = \frac{h_3 + h_1}{2} - \frac{H_3 - H_1}{\dfrac{H_3 - H_2}{h_3 - h_2} - \dfrac{H_2 - H_1}{h_2 - h_1}}. \tag{3.27}$$

При $W > h_3$ полагаем $h_1 := h_2$, $h_2 := h_3$, $h_3 := W$, $H_1 := H_2$, $H_2 := H_3$, вычисляем $H_3 = H(\alpha^0 + h_3 s)$ и возвращаемся к проверке условия $H_3 > H_2$.

При $W < h_3$ (заметим, что при указанных условиях всегда $W > h_2$) полагаем: $h_1 := h_2$, $h_2 := W$, $H_1 := H_2$, вычисляем $H_2 = H(\alpha^0 + h_2 s)$ и возвращаемся к началу.

Если же условие выпуклости (3.26) не выполняется, то полагаем $h_1 := h_2$, $h_2 := h_3$, $h_3 := rh_3$ и тоже возвращаемся к началу.

Дополнительные проверки. Бесконечный шаг: $h_3 > A$; где A — заранее заданное большое число. Процедура одномерной оптимизации для безусловного направления спуска заканчивается сообщением "бесконечный шаг", для небезусловного возможен перезапуск с $h_0 := -h_0$ (идем в противоположную сторону). Остановка: если после вычисления вершины параболы W, присвоения новых значений h_i, H_i и вычисления недостающего H_i выполнены условия

$$\frac{|h_3 - h_2|}{|h_3|} < \varepsilon, \quad |H_3 - H_2| < \delta,$$

где ε и δ — заданные малые числа, то оптимизация заканчивается, в качестве результата процедуры выдаются наименьшее из H_2, H_3 и соответствующая этому наименьшему значению величина шага.

Decrease. Эта процедура существенно различна для безусловных и небезусловных направлений спуска. Для безусловных направлений спуска шаг дробится (умножается на константу p, $0 < p < 1$) до тех пор, пока $H(\alpha^0 + hs)$ не станет меньше, чем $H(\alpha^0)$, либо шаг не станет нулевым (меньше заранее заданного малого $\gamma > 0$). Если шаг стал нулевым, одномерная оптимизация заканчивается и выдается сообщение "нулевой шаг". Если после очередного уменьшения шага стало

$$H(\alpha^0) > H(\alpha^0 + phs) \ (< H(\alpha^0 + hs)),$$

то переходим к параболическому поиску — процедура Parabola.

Для небезусловных направлений оптимизации сразу делается шаг в противоположном направлении и вычисляется $H(\alpha^0 - h_0 s)$. Если

$$H(\alpha^0) > H(\alpha^0 + h_0 s),$$

то переходим к процедуре Increase, полагая $h_0 := -h_0$.

В противном случае переходим к параболическому поиску — процедура Parabola.

Parabola. Получаем α^0, s, три различных значения шага $h_1 < h_2 < h_3$ и соответствующие значения функции H_1, H_2, H_3, среднее из которых меньше одного из крайних и не превосходит другого:

$$H_1 \geqslant H_2 < H_3 \text{ или } H_1 > H_2 \leqslant H_3. \qquad (3.28)$$

На каждом шаге снова строится тройка значений h, для которых H удовлетворяют условию (3.28). Используется процедура вычисления оценки.

Вычисляется W (3.27) и $H_W = (\alpha^0 + Ws)$. Из величин H_2 и H_W выбирается меньшая. Если это H_W, то выясняется $W > h_2$ или нет. Если $W > h_2$, то полагаем $h_1 := h_2$, $h_2 := W$, если нет, то — $h_3 := h_2$, $h_2 := W$. Если же $H_2 < H_W$, также сравниваем W и h_2. Если $W > h_2$, то полагаем $h_3 := W$, если нет, то — $h_1 := W$. Соответственные значения присваиваются и H_i.

После этого возвращаемся к началу и вычисляем новое W.

Выход из параболического поиска может совершиться по нескольким основаниям.

1. Если в последовательно вычисляемых вершинах парабол значения H отличаются меньше, чем на заранее заданную малую величину.

2. Если $h_3 - h_2$ или $h_2 - h_1$ становятся меньше некоторой заранее заданной малой величины.

3. Если парабола строилась N раз, где N — заранее заданное натуральное число.

Условие выхода конструируется из 1, 2, 3 либо как дизъюнкция — 1 или 2 или 3, либо как (1&2) или 3.

На выходе из процедуры получаем наименьшее из построенных значений H и соответствующую величину шага.

Итак, в результате одномерной оптимизации находим величину шага и значение H, либо сообщение о том, что улучшить оценку в данном направлении не удается.

Кроме того, можно получить полезную дополнительную информацию — оценку вторых производных H в данном направлении в точке выбранного шага. Результативный выход из оптимизации возможен только после построения параболы. Пусть она строится по трем значениям шага $h_1 < h_2 < h_3$ и соответствующим трем значениям оценки H_1, H_2, H_3. Тогда в качестве приближения можно принять вторую производную этой квадратичной зависимости:

$$\frac{d^2}{dx^2}H(\alpha^0 + xs) \approx 2\left(\frac{H_3 - H_2}{h_3 - h_2} - \frac{H_2 - H_1}{h_2 - h_1}\right) : (h_3 - h_1). \qquad (3.29)$$

Ее можно использовать в процедурах многомерной оптимизации.

9. Наискорейший спуск, случайный поиск и партан-методы

Мы уже имеем в своем распоряжении процедуру одномерной оптимизации. В о п р о с: в каких направлениях проводить поиск?

Пусть задано начальное значение параметров α^0 и вычислена функция оценки $H = H(\alpha^0)$. Процедура одномерной оптимизации даст приближенное положение минимума $h(x) = H(\alpha^0 + xs)$ (вообще говоря, локального). Процедура двумерного поиска уменьшает значение

$h(x, y) = H(\alpha^0 + xs_1 + ys_2)$. Теперь нужно определить направления s или s_1 и s_2 и начальные шаги.

Наиболее очевидный выбор s для одномерной оптимизации — направление антиградиента H : $s = -\nabla H$. Выберем на каждом шаге это направление, потом проведем одномерную оптимизацию, потом — снова вычислим градиент H и т.д. Это — метод наискорейшего спуска, примитивнейший из градиентных методов. Иногда работает хорошо.

Другой способ — случайный выбор направления s для одномерной оптимизации. Получаемый метод требует большого числа шагов, но зато предъявляет минимальные требования к сети — ему необходимо только прямое функционирование с вычислением оценки. Кроме того, встречаются ситуации, в которых метод наискорейшего спуска не работает, а поиск в случайном направлении медленно, но верно вытаскивает в нужную область параметров.

Для того чтобы исправить недостатки наискорейшего спуска, существует огромное число методов. В этом разделе обсудим одно семейство таких методов — итерационный и модифицированный партан-методы.

Итерационный партан-метод (k-партан) строится так. В начальной точке α^0 вычисляется градиент H и делается шаг наискорейшего спуска — для этого используется одномерная оптимизация. Далее — снова наискорейший спуск и так k раз. Начальные параметры α^0 при этом хранятся в памяти, промежуточные — нет. После k шагов наискорейшего спуска получаем α^k и проводим одномерную оптимизацию из α^0 в направлении $s = \alpha^k - \alpha^0$ (от α^0 к α^k) с начальным шагом $h = 1$ (сразу Increase). После этого цикл повторяется.

Модифицированный партан-метод также требует запоминания дополнительной карты параметров. Он строится так. Из α^0 делается два шага наискорейшего спуска. Получаем α^1, α^2. Далее — одномерная оптимизация из α^0 в направлении $\alpha^2 - \alpha^0$. Получаем α^3. После этого α^0 уже не используется. Далее — наискорейший спуск из α^3. Получаем α^4. Потом одномерная оптимизация из α^2 в направлении $\alpha^4 - \alpha^2$, получаем α^5 и т.д. Четные α^{2k} получаем наискорейшим спуском из α^{2k-1}, нечетные α^{2k+1} — одномерной оптимизацией из α^{2k-2} в направлении $s = \alpha^{2k} - \alpha^{2k-2}$ (начальный шаг $h = 1$, сразу Increase).

По нашим наблюдениям, модифицированный партан-метод в задачах обучения работает лучше, чем k-партан.

Из наискорейшего спуска, одномерной оптимизации в случайном направлении и партан-методов можно построить небольшой конструктор алгоритмов. Обозначим **S** — шаг наискорейшего спуска, **R** — шаг оптимизации в случайном направлении. Запись будем читать слева направо. Например,

$$\alpha^0\mathbf{SRRSS} \qquad (3.30)$$

означает, что из начальной точки α^0 делается шаг наискорейшего спуска, получаем α^1, потом из α^1 — шаг оптимизации в случайном направлении, получаем α^2, из α^2 — шаг оптимизации в случайном направлении, получаем α^3, из α^3 — наискорейший спуск в α^4, из α^4 — наискорейший спуск в α^5.

Специфические шаги итерационного и модифицированного партан-методов будем обозначать квадратными скобками. Скобка означает, что параметры, полученные в процедурах, описанных до этого знака, запоминаются и будут служить начальной точкой для партан-шага. Скобка] означает, что из вектора параметров α^k, который был получен в процедурах, описанных до открытия скобки, делается шаг в направлении $\alpha^l - \alpha^k (1 < k)$. Вектор a^l определен в результате процедур, описанных до закрытия скобки. Величина шага подбирается методом одномерной оптимизации, его начальное значение равно 1 (сразу Increase). Например, если в предыдущей записи (3.30) поставить скобки:

$$\alpha^0 \mathbf{SR[RS]S}, \tag{3.31}$$

это будет означать, что результат последовательного применения к α^0 сначала шага наискорейшего спуска, потом — одномерной оптимизации в случайном направлении (т.е. α^2) запоминается, потом — оптимизация в случайном направлении, получаем α^3, из α^3 — наискорейший спуск в α^4. Затем скобка закрывается, значит предпринимается шаг из точки, полученной перед открытием скобки в направлении точки, полученной перед закрытием: из α^2 проводится одномерная оптимизация в направлении $s = \alpha^4 - \alpha^2$. Находим новое α^5 — не то, что прежде. И наконец, наискорейший спуск из α^5 в α^6. Результат — α^6.

Схема итерационного партан-метода (k-партан):

$$\alpha^0 [\underbrace{\mathbf{SS \ldots S}}_{k \text{ раз}}] \ldots [\underbrace{\mathbf{SS \ldots S}}_{k \text{ раз}}] \ldots .$$

Схема модифицированного партан-метода:

$$\alpha^0 \mathbf{[SS[\,]S[\,]S[\,]S[\,]} \ldots .$$

Математическое обеспечение нейрокомпьютеров должно предполагать конструктор алгоритмов. Уже три операции: **R**, **S**, [ ] дают возможность конструирования.

Укажем еще несколько привлекательных схем:

$$\alpha^0 [\underbrace{\mathbf{RR \ldots R}}_{k \text{ раз}}] [\underbrace{\mathbf{RR \ldots R}}_{k \text{ раз}}] \ldots$$

$$\alpha^0 [\underbrace{\ldots}_{k \text{ раз}} \mathbf{[S[S]\,[S]\,[S]} \ldots \underbrace{\mathbf{[S]]]}}_{k \text{ раз}} \ldots];$$

$$\alpha^0 \mathbf{[R[S]\,[R]\,[S]\,[R]S]} \ldots .$$

10. Одношаговый квазиньютоновский метод и сопряженные градиенты

Направление наискорейшего спуска — антиградиент функции оценки H — традиционно считается не самым удачным для оптимизации. Во всех учебниках по оптимизации приводятся примеры уже довольно простых функций, с которыми метод наискорейшего спуска не может справиться.

В тех случаях, когда матрица вторых производных оценки D_2 положительно определена, наилучшим считается ньютоновское направление

$$-D_2^{-1}\nabla H. \tag{3.32}$$

Квадратичные формы минимизируются с использованием (3.32) за один шаг. Однако применять эту формулу трудно по многим причинам.

1. *Время.* Поиск всех вторых производных и обращение D_2 требуют слишком много вычислительных затрат.

2. *Память.* Для решения задач большой размерности N приходится хранить N^2 элементов D_2^{-1} — слишком много.

3. А что, если D_2 *не является положительно определенной?*

Для того чтобы справиться с этими трудностями, придумана масса ухищрений. По существу, вся литература по методам гладкой оптимизации посвящена этому.

Идея первая. Можно не дифференцировать H и не обращать матрицу D_2, а использовать итерационную процедуру, позволяющую за много шагов накопить "что-то вроде D_2^{-1}".

Идея вторая. Можно хранить в памяти не матрицу, а только результаты ее действия на некоторые векторы.

Идея квазиньютоновских методов с ограниченной памятью состоит в том, что поправка к направлению наискорейшего спуска ищется как результат действия матрицы малого ранга. Сама матрица не хранится, а ее действие на векторы строится с помощью скалярных произведений на несколько специально подобранных векторов.

Простейший и весьма эффективный метод основан на BFGS формуле (Бройден — Флетчер — Гольдфарб — Шанно) и использует результаты предыдущего шага. Для его описания обозначим: s_k — направление спуска на k-м шаге; h_k — величина k-го шага (k-й шаг — сдвиг на $h_k s_k$); g_k — градиент функции оценки в начальной точке k-го шага; $y_k = g_{k+1} - g_k$ — изменение градиента в результате k-го шага. Естественно, конечная точка k-го шага является начальной для $k+1$-го.

BFGS — формула для направления спуска на $k+1$-м шаге [45]:

$$s_{k+1} = -g_{k+1} + \frac{1}{(y_k, s_k)}[(s_k, g_{k+1})y_k + (y_k, g_{k+1})s_k] -$$

$$- h_k \frac{(s_k, g_{k+1})}{(y_k, s_k)} s_k - \frac{(y_k, y_k)\,(s_k, g_{k+1})}{(y_k, s_k)^2} s_k, \qquad (3.33)$$

где (,), как всегда, обозначает обычное скалярное произведение — сумму произведений координат.

Если одномерную оптимизацию в поиске шага проводить достаточно точно, то новый градиент g_{k+1} будет практически ортогонален предыдущему направлению спуска: $(s_k, g_{k+1}) = 0$. Формула (3.33) при этом упростится — получим

$$s_{k+1} = - g_{k+1} + \frac{(y_k, g_{k+1})}{(y_k, s_k)} s_k. \qquad (3.34)$$

Это — формула весьма популярного метода сопряженных градиентов. Он требует достаточно аккуратную одномерную оптимизацию при выборе шага.

В описанных методах предполагается, что начальное направление спуска $p_0 = - g_0$. Время от времени после некоторой последовательности из k шагов целесообразно вновь возвращаться к наискорейшему спуску — проводить рестарт. Если рестарт проводится не с направления наискорейшего спуска, то формулу (3.34) надо изменить:

$$s_{k+1} = - g_{k+1} + \frac{(y_k, g_{k+1})}{(y_k, s_k)} s_k + \frac{(y_0, g_{k+1})}{(y_0, p_0)} s_0. \qquad (3.35)$$

Здесь s_0 — направление рестарта. В качестве s_0 можно выбирать, например, последнее направление спуска перед рестартом и сразу переходить к s_1. Рестарт используется в тех случаях, когда очередное s_{k+1} — плохое направление спуска, т.е. движение вдоль него приводит к слишком маленькому шагу, либо вообще не дает улучшения.

Итак, описано три способа для выбора поправок к направлению наискорейшего спуска: одношаговый квазиньютоновский метод (3.33), метод сопряженных градиентов (3.34) и метод сопряженных градиентов с рестартом (3.35).

11. Экспериментальное сравнение методов обучения нейронных сетей

Задача обучения нейросети рассматривается как задача минимизации средней оценки по обучающему множеству.

Сравнивались методы: А) поиска в случайных направлениях (RND); Б) наискорейшего спуска (SD); В) один из квазиньютоновских методов (BFGS).

Методы сопоставлялись по критерию "скорость обучения", а именно сравнивалось число тактов нейросети, потребовавшихся для обучения нейросети правильному распознаванию всех примеров в обучаю-

щей выборке. (Один такт нейросети — единица дискретного времени, за которое все нейроны обмениваются своими сигналами).

Сравнение проводилось для двух задач бинарной классификации:

1) определения направления циклического сдвига конечной двоичной последовательности (с элементами 0, 1);

2) определения симметричности или несимметричности конечной двоичной последовательности.

Направление сдвига определялось для последовательностей длины 8. На вход сети подавались исходная последовательность и циклически сдвинутая на одну позицию вправо или влево (всего 16 входных сигналов). Объем задачника при этом — 504 примера.

В задаче определения симметрии длина последовательности полагалась равной десяти, что дает объем задачника 1024 примера, из которых лишь 32 принадлежат классу симметричных последовательностей. В связи с этим в последней задаче оценка симметричных примеров учитывалась с весом 31.

Нулевой сигнал кодировался значением входного сигнала равным -1, а единичный сигнал $+1$. В задаче распознавания симметрии подавался дополнительный сигнал, равный 1 для всех примеров. Принятая схема подачи входных сигналов: один сигнал — один входной нейрон.

Использовались полносвязные нейросети с функцией активации нейрона $y = x/(c + |x|)$, где c — крутизна ("характеристика") сигмоидной функции. В экспериментах бралось значение $c = 0{,}1$.

Градиент оценки вычислялся по принципу двойственности.

В случае выполнения неравенства $y_{N-1} - y_N > \varepsilon$, где $\{y_i\}_{i=1}^N$ — вектор выходных сигналов нейросети, ответ нейросети интерпретировался как принадлежность к 1-му классу; в случае $y_N - y_{N-1} > \varepsilon$ — ко 2-му. Константа $\varepsilon \geqslant 0$ — "уровень надежности". Она может быть различной при обучении и распознавании. Для распознавания достаточно полагать $\varepsilon = 0$, а при обучении использовать большие значения для достижения большей устойчивости к возмущениям. В качестве функции оценки при обучении бралось расстояние до множества достоверно идентифицируемых ответов при уровне надежности $\varepsilon = 1$. Обучение прекращалось, когда все примеры из обучающей выборки правильно идентифицировались при $\varepsilon = 0$.

Для сравнения методов были сгенерированы по 100 нейросетей в каждой из тестовых задач (при тестировании обучения задаче распознавания симметрии методом RND использовались только первые 20 нейросетей). Подстроечные параметры нейросетей ("синаптические веса") выбирались случайно с равномерным распределением в интервале $(-0{,}01 \div 0{,}01)$ (табл. 3.1).

Полученные результаты позволяют сделать следующие выводы:

Таблица 3.1

Время обучения нейросети, тыс. тактов

Метод	AV	σ	min	max
Задача определения направления сдвига				
RND	7713	2499	3680	16 259
SD	2132	935	839	6199
BFGS	296	102	191	1028
Задача определения симметрии последовательности				
RND	33 114	12 310	18 190	65 150
SD	2820	1053	1180	7115
BFGS	731	259	388	1831

Примечание. AV — среднее время обучения; σ — среднеквадратичное отклонение.

1) метод RND может быть конкурентоспособным в том случае, если при аппаратной реализации нейросети затраты на запоминание сигналов превышают примерно в 50 раз затраты на обмен сигналами;

2) для методов, использующих производные целевой функции, целесообразно применять быстрые антиовражные алгоритмы (например, квазиньютоновские), так как эффект от увеличения быстродействия покрывает дополнительные затраты памяти, составляющие примерно 25 %.

Вычислительный эксперимент проводился С.Е. Гилевым [34], результаты публикуются с его любезного разрешения.

12. Использование случайных возмущений в обучении

Обучение можно уподобить спуску в долину по гористой местности. Все описанные методы спуска на каждом шаге ведут вниз. Это — достоинство, но оно может превращаться в недостаток. Если идти вниз и только вниз, то, быть может, придется долго следовать извилистыми ущельями и застревать даже в маленьких котловинах. Таких задержек можно избежать. Один из приемов: время от времени подниматься по ближайшему склону и искать оттуда направление наилучшего спуска.

Реализовать метаформу можно с помощью случайных возмущений. Между циклами спуска вставляется операция "Bump" (шлепок). Параметрам сети даются независимые случайные приращения. Bump вставляется между циклами алгоритмов спуска. После случайного сдвига следует рестарт — спуск продолжается.

Bump ставит перед учителем сети несколько вопросов.

1. Как выбрать *вид распределения* для случайных сдвигов? Обычно выбирается равнораспределение в некотором интервале.

2. Как выбрать *интервалы* случайного сдвига для различных параметров? В частности, можно ли выбрать их одинаковыми?

3. Как *изменять характеристики* случайных сдвигов в *ходе обучения?*

Равноправие всех параметров в структуре сети — ситуация весьма специальная. Пример — полносвязная чисто коннекционистская сеть с одинаковыми нейронами. Но и эта симметрия разрушается случайной генерацией начальной карты, разделением нейронов на входные и выходные, заданием функции оценки. Поэтому естественно выбирать интервалы случайного сдвига для различных параметров по-разному.

Если основная задача — ускорить обучение, то уровень шума естественнее снижать и устремлять к нулю по мере приближения оценки к минимуму. Для создания запаса устойчивости шум в ходе обучения полезно увеличивать.

Что может служить задаваемой извне характеристикой шума, по которой будут контролироваться его уровень и приниматься решение об увеличении или уменьшении амплитуды? Один из вариантов — скачок оценки ("подъем"). Этот скачок можно задать в абсолютных величинах, а можно и в относительных — в долях от текущей оценки. Если на данном шаге подъем превысил заданное значение, то нужно уменьшить уровень шума и повторить изменение (с той же начальной карты). Итак, пока сохраняется превышение. Если на данном шаге подъем меньше заданного предела, то можно принять это случайное изменение, но на следующем шаге повысить уровень шума.

При описании случайных возмущений сети удобно воспользоваться физическими аналогиями. Для этого вводятся новые величины: температура сети и энергия активации различных параметров.

Температура характеризует общий уровень шума, а *энергия активации* показывает, насколько данный параметр способен удерживать свое значение, т.е. какая температура необходима для того, чтобы он начал существенно флуктуировать.

Случайный сдвиг i-го параметра α_i определяется так:

$$r_i A_i \exp\left(-\frac{E_i}{T}\right), \tag{3.36}$$

где r_i — случайная величина, равномерно распределенная в отрезке $[-1, 1]$; A_i — полуширина интервала приемлемых изменений α_i; E_i — энергия активации α_i; T — температура сети.

Температура подбирается из ограничений на подъем оценки ΔH. Эти ограничения со временем стоит менять и лучше вообще задавать их в относительных единицах $(\Delta h/H)$.

Различные алгоритмы определения энергии активации параметров и задания температуры сети подробно описаны в [54].

Чтобы шум ускорял начальное обучение и не мешал на завершающих стадиях, его уровень надо снижать по мере уменьшения оценки.

Но одна из основных проблем в обучении нейронных сетей — создание навыков, которые не теряются при дальнейшем обучении. Как сделать так, чтобы старые навыки не вытеснялись новыми? Или, по крайней мере, вытеснялись не слишком быстро.

Средство, которое позволяет это сделать, если такая устойчивость вообще возможна, — использование шума, нарастающего в ходе обучения. Такой шум моделирует помехи от предстоящего обучения новым примерам, поэтому он должен в меньшей степени, чем ускоряющий шум, щадить параметры с высокими показателями чувствительности. Ведь дальнейшее обучение будет менять эти параметры наравне с остальными.

Предлагается устанавливать одинаковый интервал сдвига для параметров, симметрично участвующих в структуре сети. Так, для полносвязной сети с линейными синапсами и однопараметрическими нейронами есть три группы параметров: веса синапсов связи, веса синапсов памяти и параметры нейронов. Для слоистой сети параметры различных слоев структурно различаются и соответственно увеличивается количество групп.

Внутри группы структурно эквивалентных параметров выбирается один интервал случайного сдвига. Для каждой группы устанавливается также свой допустимый уровень подъема при сдвиге. Bump проводится по очереди в разных группах с контролем подъема и коррекцией уровня по группам. Допустимые сдвиги со временем увеличиваются до заданной границы. При этом либо достигается требуемый уровень устойчивости, либо, увы, не достигается приемлемо низкий уровень оценки.

13. Несколько слов о философии оптимизации

Уже ясно, что для обучения нейронной сети можно брать любой алгоритм многомерной оптимизации, если он не требует слишком большой памяти. Для работы по этим алгоритмам учителю предоставляется несколько основных процедур. Среди них вычисление значения целевой функции и ее градиента.

При организации алгоритма желательно использовать естественную структуру сети: параметры, градиенты и другие характеристики разбиты на много локальных массивов, действия в которых можно проводить параллельно. Кроме того, существует возможность раскрашивания сети.

В такой ситуации естественно использовать различные ускоряющие преобразования: одни — внутри локальных массивов, другие — между ними, третьи — между цветами.

Остается, однако, неудовлетворенность подходом в целом. Грубо говоря, теория безусловной оптимизации посвящена минимизации квадратичных форм: $H(x) = (x, Qx) + (b, x) + c$. Постоянно дела-

ются попытки уйти как можно дальше от функций, задаваемых квадратичными формами, но все же и метод Ньютона, и метод сопряженных градиентов, и другие усовершенствованные методы спуска создавались применительно к квадратичным формам и тем функциям, которые с их помощью хорошо аппроксимируются.

Идеализированный модельный объект — путеводная звезда теории и ориентация на него неизбежна, какие бы слова при этом ни произносились. Теория выпуклой оптимизации потому и удалась, что выпуклая функция обладает многими важными свойствами положительно определенной квадратичной формы.

Адаптивный ландшафт нейронной сети — график зависимости H от параметров. Вряд ли он похож на чашу — график положительной квадратичной формы. Скорее всего, он напоминает ландшафт горного района без особо острых пиков и пропастей — нечто вроде Карпатских гор.

То же можно сказать и про многие другие функции. Вывод — нужен новый модельный объект, который бы дал другую идеологическую установку для теории оптимизации.

За общностью математических теорий обычно спрятаны лежащие в основе и подразумеваемые семейства модельных объектов.

В теории безусловной оптимизации неоднократно вводились дополнительные претенденты на роль модельных объектов. Именно поэтому большую популярность обрели стандартные тестовые неквадратичные задачи, такие как минимизация функции Розенброка:

$$F(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Однако для многомерной оптимизации вряд ли возможно строить дополнительный набор тестовых задач, собранный "поштучно", как это было для малых размерностей. Как же поступить в этом случае?

Сказать трудно, но можно предположить, что модельным должен быть какой-нибудь достаточно простой класс случайных функций. Для него нужно будет строить алгоритмы и оценивать скорость сходимости. Эти алгоритмы будут хорошо работать с теми функциями, которые близки к типичным реализациям случайных функций данного класса. Можно привлекать также аналогии из статистической физики. Теории такого типа пока нет, но уже предварительные соображения позволяют предложить новые алгоритмы.

14. Четыре типа устойчивости

Навыки обученного нейрокомпьютера должны быть устойчивы к возмущениям разных типов. Это очевидно, однако придать точный смысл понятию "устойчивость" непросто. В теории динамических систем и дифференциальных уравнений известны десятки ее определений.

Разработчики нейрокомпьютеров находятся в несколько лучшей ситуации — для практических целей не нужно учитывать нюансы,

возникающие из-за наличия в теории бесконечной оси времени. Достаточно описать различия между содержательно разными типами устойчивости и создать средства для выработки устойчивых навыков. Мы использовали представления о четырех типах устойчивости. По типу возмущающих воздействий определяется ее тип: 1 — к случайным возмущениям входных сигналов, 2 — к флуктуациям параметров сети, 3 — к разрушению части элементов сети, 4 — к обучению новым примерам.

В конкретных ситуациях необходимо доопределять возмущения, по отношению к которым нужно выработать устойчивость. Например, при распознавании визуальных образов можно выделить несколько разновидностей возмущений входного сигнала: прибавление случайного сигнала (шум фона), затенение части исходного изображения, искажение изображения некоторым преобразованием.

В существенной конкретизации нуждается также четвертый тип устойчивости: требуется указать, что это за новые примеры. Трудно даже представить себе устойчивость к обучению любому новому примеру. Если же совокупность новых примеров задана, то их можно включить в задачник. Тем самым выработка устойчивости к обучению этим примерам заменяется просто обучением.

Если принять гипотезу, что обучение новым примерам будет действовать на старые навыки так же, как случайный сдвиг параметров, то получим, что выработка устойчивости 2-го типа является средством для получения устойчивости 4-го типа.

Другое средство — выработка устойчивости к обучению "до предела" отдельным примерам, уже входящим в задачник. Это свойство устойчивости 1-го типа состоит в том, что обучение до минимума оценки по любому (одному) из обучающих примеров не разрушает навыка решения остальных. Возмущение здесь состоит в изменении процесса обучения.

Для выработки устойчивости 1-го типа примеры предъявляются сети не полностью, а по одному, и сеть учится каждому из них до предела. Это изменение, надо сказать, было бы неразумным, если бы речь шла о скорейшем приобретении навыка. Однако для выработки важнейшей устойчивости 4-го типа такая периодически производимая порча процесса обучения может быть полезной.

Проще интерпретировать возмущения, входящие в описания 2-го и 3-го типов устойчивости. Для 2-го типа это случайное изменение каждого параметра на величину, лежащую в заданных пределах. Для 3-го типа это удаление из сети заданной части элементов каждого типа: определяется число удаляемых нейронов, синапсов и т.п., после чего случайным образом выбираются удаляемые элементы.

Для выработки устойчивости первых трех типов полезны генераторы случайных искажений. Для устойчивости 1-го типа генератор искажений производит возмущение входных сигналов и тем самым преоб-

разует обучающий пример. Для устойчивости 2-го типа генератор искажений меняет случайным образом параметры сети в заданных пределах, а для устойчивости 3-го типа — удаляет случайно выбранную часть сети из заданного количества элементов.

Опыт показывает, что обучение может выработать устойчивость к весьма сильным возмущениям. Так, в решавшихся нами задачах распознавания визуальных образов уровень шума на входе мог в несколько раз превосходить общую интенсивность сигнала, случайный сдвиг параметров — достигать 0,5—0,7 их предельного значения, разрушение 30—50 % элементов. И все равно обученная сеть делает не более 10 % ошибок или не делает их вовсе!

15. Метод виртуальных частиц

Метод виртуальных частиц может надстраиваться практически над любым методом оптимизации. Он создан для повышения устойчивости обученных сетей и неслучайного вывода сетей из возникающих при обучении локальных минимумов оценки [54].

Метод состоит в том, что к точке, спускающейся по адаптивному ландшафту (частице), добавляются несколько других, траектории которых получаются из траектории частицы сдвигом на случайный вектор. Такие "виртуальные" частицы время от времени (через несколько циклов оптимизации) уничтожаются и порождаются новые. При этой регенерации частиц центром служит частица из предыдущей генерации, у которой индивидуальная оценка минимальна.

Спуск строится так, чтобы уменьшалась взвешенная сумма высот. Относительное положение частиц при спуске не меняется (разности между векторами их параметров сохраняются — фактически учится только одна, "средняя", сеть, а остальные служат для сглаживания оценки).

Для метода виртуальных частиц стандартным модельным примером являются уже не квадратичные формы, а функции вида

"квадратичная форма + случайная функция".

За счет сдвигов и последующего суммирования случайная компонента адаптивного рельефа стирается, а метод оптимизации, над которым надстраивается метод виртуальных частиц, успешно минимизирует квадратичную форму.

Метод виртуальных частиц имеет все достоинства методов глобальной оптимизации, использующих случайные возмущения, но лишен многих их недостатков. Остановимся на нем подробнее, вплоть до результатов его сравнительных испытаний. Эти испытания проводились Н. Сиротининой, результаты публикуются с ее любезного разрешения.

В разделе приняты следующие сокращения:

ВЧ — виртуальные частицы; НС — нейронные сети; НС ВЧ — нейронные сети, обучающиеся по методу виртуальных частиц.

Параметрами облака ВЧ являются: число частиц n; разброс частиц R; вес частиц w.

Веса ВЧ (из интервала [0, 1]) назначались одинаковыми. Основная частица имела вес 1.

Требование центрального положения основной частицы влияет не только на процедуру генерации облака ВЧ, но и на его поведение в ходе обучения. В рассматриваемой модели НС веса синаптических связей имеют ограниченную область значений (обычно [−1, 1]). В случае одной (главной) частицы это требование осуществляется напрямую запрещением выхода значений весов связей из области допустимых значений. В процессе обучения НС ВЧ неизбежно возникает ситуация, когда некоторые частицы оказываются вне этой ограниченной области. Возможны следующие варианты поведения облака ВЧ.

1. "Исчезновение" частиц, выходящих за границы. В этом случае необходимо иметь в виду, что, как правило, по окончании обучения часть весов принимает значения, близкие к граничным. По этой причине к окончанию обучения число частиц может существенно уменьшиться, что нежелательно. Кроме того, выполнение условия центрального положения главной частицы здесь затруднено.

2. Запрещение выхода частиц за границы, т.е. проецирование выходящих за ограничение частиц на грань гиперкуба, который описывается введенным ограничением. При этом также нарушается форма облака.

3. Разрешение выхода частиц за границы. Этот вариант наиболее просто реализуется и не требует дополнительных расходов на перегенерацию облака ВЧ. Неограниченный рост абсолютного значения параметров не происходит, поскольку ВЧ задаются смещением относительно главной частицы. Если выход за ограничения главной частицы запрещен, то любая ВЧ выходит за ограничения не более, чем на радиус разброса.

В программной модели НС ВЧ был реализован последний вариант поведения ВЧ при выходе за ограничения.

В ходе обучения НС облако ВЧ может уничтожаться и создаваться заново. Важным вопросом является выбор моментов перегенерации облака ВЧ. Здесь просматриваются следующие варианты.

1. Генерация стационарного облака. В этом случае облако генерируется однажды и сохраняется до окончания процесса обучения. Такой вариант требует минимальных затрат на реализацию. Но необходимо учесть, что при обучении НС минимизация идет в пространстве с высокой размерностью (в исследуемой модели число обучаемых параметров порядка 200). Для обеспечения сколько-нибудь полного

сканирования окрестностей главной частицы число частиц должно быть достаточно большим.

2. Перегенерация облака осуществляется, когда скорость обучения падает ниже некоторой критической. Такой способ менее консервативен, но при этом остается открытым вопрос о выборе критического значения скорости обучения.

3. Порождение нового облака частиц после каждого цикла базового алгоритма оптимизации (шага по градиенту, партан-шага и т.д.). Этот способ сохраняет базовый алгоритм и позволяет осуществить сканирование окрестностей главной точки малым количеством частиц за счет их регулярного обновления. Недостатком такого способа может оказаться низкая устойчивость процесса обучения при малом числе частиц, когда каждая перегенерация облака будет приводить к кардинальной смене направления спуска.

Мы апробировали способы 1 и 3 — стационарное облако и регенерацию облака после каждого шага.

В качестве базовой программы, модифицируемой для метода ВЧ, выбрана программа SIGMOID1 из пакета прикладных программ "учебник по нейронным сетям" (авторы — Е.М. Миркес, С.Е. Гилев, А.Н. Горбань, группа НейроКомп). Программа предназначена для научно-учебного использования и позволяет обучить НС классифицировать бинарные изображения, формируемые пользователем, на 5 классов. Нейроны сети реализуют сигмоидную функцию активации вида

$$\frac{A}{c+|A|}.$$

Сети полносвязные, каждый нейрон получает на начальном шаге строку изображения, после чего два раза происходит обмен сигналами между нейронами.

В программе SIGMOID1 реализована возможность обучения НС различными методами, в том числе случайного поиска, градиентного спуска и партан-методом. Предусмотрена возможность контрастирования карты весов синаптических связей. Программа позволяет проводить тестирование устойчивости работы обученной нейросети к внесению в обрабатываемые изображения различных видов шумов (добавляющего, гасящего, инвертирующего) и к удалению части изображения (тест с "тенью"), а также к случайному изменению обучаемых параметров.

В первом цикле экспериментов по исследованию метода ВЧ проводилось обучение решению сравнительно несложной задачи — распознаванию пяти первых букв латинского алфавита. Каждый класс в обучающей выборке был представлен только одним примером.

Время обучения НС зависит от начального набора параметров сети. Поэтому при сравнении нескольких методов тесты осуществ-

лялись с использованием набора пяти стартовых карт (за исключением особо оговоренных случаев). Для сравнения каждая карта обучалась и без ВЧ. Как показали эксперименты, время обучения и свойства НС со стационарным облаком частиц сильно зависят также от положения частиц. Разница во времени обучения сетей с одной начальной картой и одинаковыми параметрами облака ВЧ, но с разным положением частиц составляет до 5—20 раз.

Обучение проводилось с использованием партан-метода и оценки типа "расстояние до множества" с надежностью 1. При тестировании на устойчивость к шуму в изображении выбраны следующие параметры теста: число искажений на пример — 100, уровень шума (гасящего, добавляющего, инвертирующего) — 0,1.

Тестировались две модели: со стационарным облаком ВЧ (модель 1) и с облаком частиц, регенерируемым после каждого партан-шага (модель 2).

1. О б щ и е р е з у л ь т а т ы:

А. НС ВЧ обучаются. В некоторых случаях время обучения в пересчете на одну частицу оказывается меньше, чем для обычных способов обучения.

Б. Нейронная сеть, обученная методом виртуальных частиц, в 97 % случаев имеет большую устойчивость к шуму в изображении; повышение надежности — 3—12 %, в среднем 6 %. Это свойство является побочным эффектом применения виртуальных частиц.

2. Влияние разброса ВЧ на время обучения и надежность НС (табл. 3.2).

Условия эксперимента: стационарное облако ВЧ, число частиц $n = 15$, вес виртуальных частиц $w = 0{,}5$.

Среднее время обучения стартовых карт без ВЧ — 820 тактов (время измеряется в тактах сети, за один такт полносвязной сети все нейроны срабатывают по одному разу, время работы системы виртуальных сетей измеряется также в тактах на 1 частицу — периодах срабатывания всех нейронов всех сетей по одному разу).

В ы в о д ы. Разброс менее 0,01 неэффективен, поскольку в этом случае обучение ВЧ практически вырождается.

При разбросе более 0,3 время обучения резко возрастает (более 50000 тактов на частицу) и зачастую обучение оканчивается неудачей.

Предпочтительно выбирать разброс ВЧ 0,05—0,2. Выбор конкретного значения зависит от желаемого уровня устойчивости к случайному изменению обучаемых параметров.

Напомним, что пределы изменения параметров: от $+1$ до -1. Разброс R означает, что параметры виртуальных сетей генерируются со случайным сдвигом от параметров основной сети, равномерно распределенным в интервале $[-R, R]$.

2′. Влияние разброса ВЧ на время обучения НС ВЧ с перегенерацией облака после каждого партан-шага.

Таблица 3.2

Влияние разброса ВЧ на время обучения НС ВЧ

Разброс	Время обучения, такт/частица		
	min	max	среднее
	Модель 1		
0,01	965	1880	1105
0,05	445	2925	973
0,1	500	2540	903
0,2	2515	>50 000	4150
	Модель 2		
0,05	665	3845	1108
0,1	1155	6842	1823
0,2	4005	>100 000	11 497

Условия эксперимента: $n = 10$, $w = 0{,}5$.

В ы в о д ы. С ростом разброса ВЧ время обучения растет, причем более интенсивно, чем у НС со стационарным облаком.

3. Влияние разброса ВЧ на устойчивость к случайному изменению обучаемых параметров (модель 1) (табл. 3.3, 3.4).

Внесение случайных изменений в карту весов синаптических связей осуществляется процедурой Bump (шлепок с использованием генератора случайных чисел. Процедура имеет параметр Bl (Bump level — уровень Bump), который определяет максимально допустимое изменение параметров (случайные изменения каждого параметра при Bump равномерно распределены в интервале [−Bl, +Bl]).

В ы в о д ы. Разброс менее 0,01 не приводит к видимому увеличению устойчивости к Bump. С увеличением радиуса разброса ВЧ устойчивость растет за счет сканирования частицами большего пространства.

Эффект повышения устойчивости к случайному изменению обучаемых параметров наблюдается и когда уровень Bump больше радиуса разброса частиц. При большом радиусе разброса (0,2) иногда возникают карты-рекордсмены, в которых разрушение навыков происходит при уровне Bump 0,5 — 0,7 (!!!).

3′. Влияние разброса ВЧ НС с перегенерацией облака после каждого партан-шага на устойчивость к случайному изменению обучаемых параметров в основном такое же, как и для модели 1.

4. Влияние числа частиц на эффективность обучения (модель 1) (табл. 3.5).

Условия эксперимента: $R = 0{,}2$, $w = 0{,}5$.

Среднее время обучения НС без ВЧ — 820 тактов.

В ы в о д ы. С ростом числа частиц имеется тенденция к увеличению времени обучения. Это объясняется тем, что, формируя общую оценку для НС ВЧ как взвешенную сумму оценок по всем

Таблица 3.3

Влияние разброса ВЧ на устойчивость к Bump (модель 1)

Разброс	Устойчивость к Bump, % верных ответов		
	min	max	среднее
	Bl = 0,1. В среднем без ВЧ верно 65 %		
0,01	50,2	74,8	62,2
0,05	92,4	99,4	94,28
0,1	99,4	100	99,8
0,2	97,2	100	99,8
	Bl = 0,2. В среднем без ВЧ верно 40 %		
0,01	32	46	35,4
0 ,05	58,6	82,2	70,65
0,1	78,4	79,8	93,6
0,2	79,4	97,6	96,5
	Bl = 0,3. В среднем без ВЧ верно 33 %		
0,01	23	35	28,7
0,05	38,4	63,2	46,8
0,1	59,6	89	80,5
0,2	74,4	91,6	88,2

частицам, мы тем самым накладываем более жесткие требования на точку завершения обучения: требуем, чтобы оценка была нулевой не только в точке, соответствующей главной частице, но и во всех точках, соответствующих виртуальным частицам. Не каждый минимум, удовлетворяющий условию окончания обучения нейросети без виртуаль-

Таблица 3.4

Влияние разброса ВЧ на устойчивость к Bump (модель 2)

Разброс	Устойчивость к Bump, % верных ответов		
	min	max	среднее
	Bl = 0,1. В среднем без ВЧ верно 65 %		
0,05	91,6	98,6	92
0,1	98	100	99,5
0,2	—	—	100
	Bl = 0,2. В среднем без ВЧ верно 40 %		
0,05	50,4	79	63,7
0,1	81,4	96,4	90,3
0,2	97,2	100	99,8
	Bl = 0,3. В среднем без ВЧ верно 33 %		
0,05	30,2	59,2	50,4
0,1	59	89,3	74,8
0,2	88	96,4	92,1

Таблица 3.5

Влияние числа частиц на время обучения

Число ВЧ	Время обучения, такт/частица		
	min	max	среднее
	Модель 1		
3	2108	4561	3025
5	1531	7010	3897
8	905	15 645	4166
10	2656	27 070	3943
15	1055	5416	4150
20	960	15 605	4415
	Модель 2		
3	5100	65 140	8585
5	5485	32 010	10 190
8	6600	46 285	10 845
10	4005	>100 000	18 990

Таблица 3.6

Влияние числа ВЧ на устойчивость к Bump (модель 1)

Число ВЧ	Устойчивость к Bump, % верных ответов		
	min	max	среднее
	Bl = 0,1. В среднем без ВЧ верно 65 %		
3	74,2	97,8	86,4
5	80,2	92,6	88,5
8	86,6	100	96,8
10	91,6	100	96,6
15	99,4	100	99,95
20	99,8	100	100
	Bl = 0,2. В среднем без ВЧ верно 40 %		
3	45,2	75,6	58,4
5	51,2	74,4	61,6
8	66,8	94,4	79,5
10	68,6	95	80,6
15	86,4	99,6	96,45
20	91	99,4	97,04
	Bl = 0,3. В среднем без ВЧ верно 33 %		
3	34,2	54,2	43,8
5	42,4	62,2	45,6
8	49,4	82,4	63,1
10	55,2	81,2	65,2
15	70,6	95,4	88,2
20	74,6	92,2	88,2

ных частиц, удовлетворяет НС ВЧ, поэтому время обучения возрастает.

4′. Влияние числа частиц на время обучения НС ВЧ с перегенерацией облака после каждого партан-шага (модель 2).

Условия эксперимента: $R = 0{,}2$, $w = 0{,}5$.

В ы в о д ы. Время обучения НС ВЧ с перегенерацией облака после каждого партан-шага растет с ростом числа ВЧ. Время обучения (в пересчете на частицу) значительно больше, чем для НС ВЧ со стационарным облаком, и растет более интенсивно.

5. Влияние числа ВЧ на устойчивость к Bump (модель 1) (табл. 3.6).

В ы в о д ы. С ростом числа ВЧ растет устойчивость к случайному изменению весов синаптических связей за счет более полного сканирования окрестностей главной частицы.

При количестве ВЧ порядка 15—20 происходит насыщение облака и дальнейшее увеличение числа частиц стационарного облака представляется нецелесообразным, так как практически не улучшает свойств сети, но приводит к значительным дополнительным затратам на обучение.

5′. Влияние числа частиц на устойчивость к Bump НС ВЧ с перегенерацией облака после каждого партан-шага (модель 2) (табл. 3.7).

В ы в о д ы. Устойчивость к Bump НС ВЧ с перегенерацией облака после каждого партан-шага растет с увеличением числа ВЧ. Насыщение облака происходит при 5 ВЧ, дальнейшее увеличение числа частиц представляется нецелесообразным.

Достижение высокого уровня Bump-устойчивости обеспечивается в модели 2 значительно меньшим числом частиц. Так, показатели для этой модели НС ВЧ с 5 ВЧ соответствуют показателям НС ВЧ со стационарным облаком из 20 частиц. Но время обучения (как прямое, так и в пересчете на одну частицу) значительно больше.

6. Влияние предобучения на эффективность обучения и надежность НС (табл. 3.8).

Учитывая высокую затратность метода ВЧ при его моделировании на последовательном процессоре, естественным представляется желание улучшить начальное положение НС на гиперповерхности оценки для сокращения времени обучения.

Условия эксперимента: $n = 15$, $R = 0{,}2$, $w = 0{,}5$, модель 1.

Варианты предобучения НС: 1 — без предобучения; 2 — полностью обученная до оценки 0 при надежности 1,0 НС без ВЧ; 3 — обученная до оценки 0 при надежности 0,1 НС без ВЧ; 4 — обученная до оценки порядка 0,5 при надежности 0,1 НС без ВЧ.

В ы в о д ы. Некоторый эффект наблюдается при всех видах предобучения. Наибольшую эффективность показали полное обучение НС без ВЧ и обучение НС без ВЧ до оценки порядка 0,5 при уровне надежности 1,0 (оценка типа "расстояние до множества"). Эти же виды предобучения показали (незначительно) большую устойчивость к

Таблица 3.7

Влияние числа ВЧ на устойчивость к Bump (модель 2)

Число ВЧ	Устойчивость к Bump, % верных ответов		
	min	max	среднее
Bl = 0,1. В среднем без ВЧ верно 65 %			
3	99,8	100	100
5	—	—	100
8	—	—	100
10	—	—	100
Bl = 0,2. В среднем без ВЧ верно 40 %			
3	93,6	100	94,2
5	96,6	98,8	98
8	97,8	99,4	98,2
10	97,2	100	98,4
Bl = 0,3. В среднем без ВЧ верно 33 %			
3	79,8	97,6	82,2
5	85,6	88,4	88
8	88,4	90,6	89,4
10	88	96,4	89,3

Таблица 3.8

Влияние предобучения на время обучения НС ВЧ

Вариант предобучения	Время обучения, такт/частица		
	min	max	среднее
1	960	27 070	4851
2	1531	4655	2612
3	1774	6487	4077
4	905	15 065	3930

Bump — на 3—8 % выше двух других. При необходимости эти варианты предобучения можно использовать.

7. Влияние веса частиц на время обучения и надежность работы НС.

Устойчивой зависимости между весом ВЧ и характеристиками обученной НС не обнаружено. По результатам эксперимента рекомендуется вес частицы выбирать в пределах 0,25—0,5.

Уникальность метода виртуальных частиц состоит в том, что он позволяет отыскивать минимум функции оценки с гарантированной устойчивостью к изменениям подстраиваемых параметров — впадину гарантированной ширины на адаптивном рельефе.

17. Тестирование нейронных сетей с квадратичными сумматорами

При рассмотрении алгоритмов обучения неоднократно подчеркивался большой произвол в выборе характеристик элементов, составляющих обучаемую сеть. Не вызывает затруднений использование формул быстрого дифференцирования (принципа двойственности) и стандартных алгоритмов оптимизации для обучения сетей, составленных из произвольных гладких автоматов. Проблема, скорее, в другом — обосновать выбор определенной архитектуры из бесконечного потенциального разнообразия. Тут есть несколько путей — можно остановиться на сетях из стандартных нейронов с линейными сумматорами при входах — раз нет достаточных оснований для выбора определенной архитектуры, то выбираем простейшую и универсальную. Но есть серьезные причины для использования сетей с квадратичными сумматорами. Первое из них — для гауссовских распределений один квадратичный сумматор решает задачу разделения классов методом максимума правдоподобия. Поэтому квадратичные сумматоры выделены в классе автоматов — по крайней мере так же, как распределение Гаусса выделено среди всех вероятностных распределений. Есть и другие соображения, основанные на нейрофизиологических аналогиях. Ниже излагаем результаты, полученные С.Е. Гилевым [32], следуя тексту его отчета (с любезного разрешения автора).

Идея использовать нейросети с квадратичными сумматорами для улучшения способности сети к обобщению возникла на одном из семинаров А.Н. Горбаня с группой студентов. Она базируется на хорошо известном факте индукции в естественных нейросетях, когда возбуждение в одних областях мозга влияет на возбуждение в других. Простейшей формализацией этого является введение коэффициента, пропорционального сигналу от j-го нейрона, при вкладе от сигнала i-го нейрона в величину, формирующую сигнал заданного нейрона (без индекса). Снабдив такое произведение весом q_{ij} — "коэффициентом индукции", получим рассматриваемую архитектуру:

$$y = \Phi(A), \tag{3.27}$$

$$A = Q(x) + L(x) + p, \tag{3.28}$$

в отличие от традиционной, где вместо (3.28) использовалось

$$A = L(x) + p. \tag{3.28'}$$

Здесь y — выходной сигнал нейрона; x — вектор сигналов, поступающих на нейрон с его синапсов; Φ — характеристическая функция нейрона, вид которой достаточно произволен, однако в описываемых здесь экспериментах она была такой:

$$\varphi(A) = \frac{A}{c + |A|}; \tag{3.29}$$

Q — квадратичная, L — линейная формы; p — константа. Коэффициенты форм Q и L и константы p являются подстроечными параметрами, определяющимися в ходе обучения.

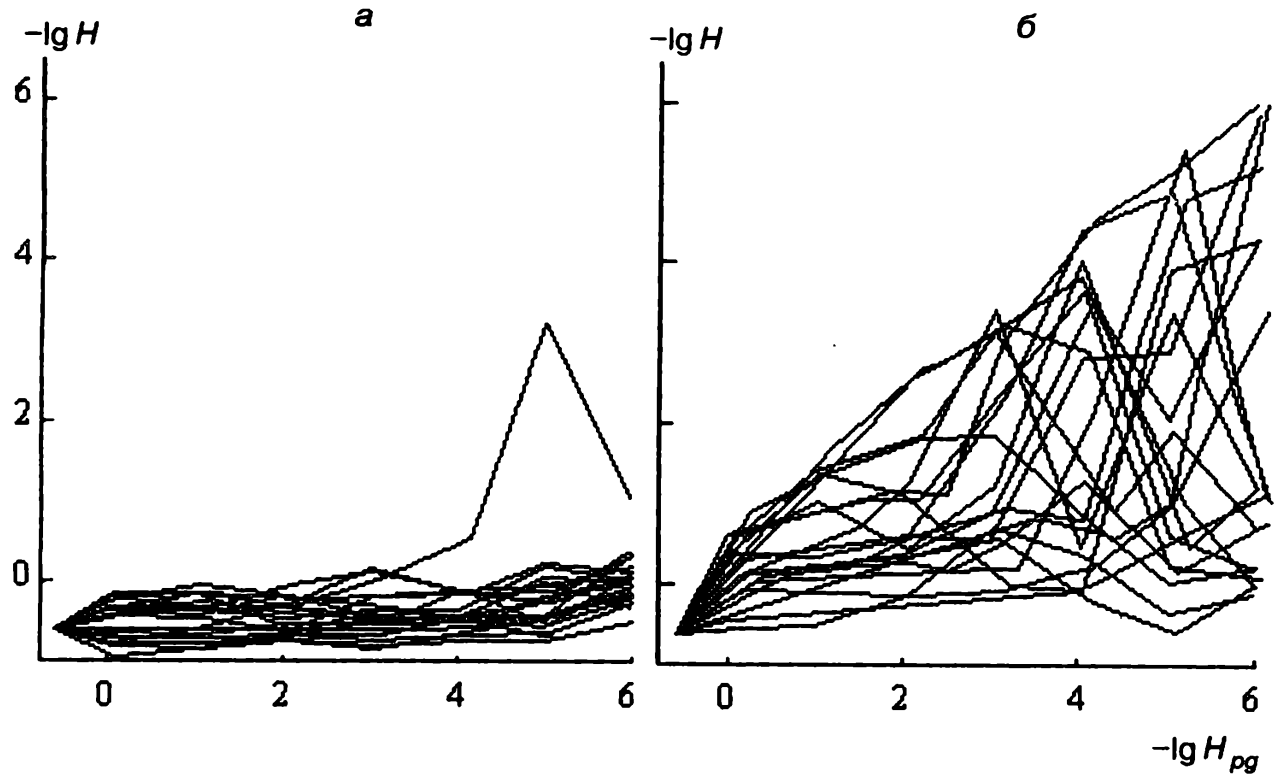


Рис. 3.11. Зависимости оценок сетей по всему задачнику (H) от их оценок по обучающей выборке из 32 примеров (H_{pg}) (масштаб логарифмический).

а — для сетей с линейными сумматорами; *б* — для сетей с квадратичными сумматорами.

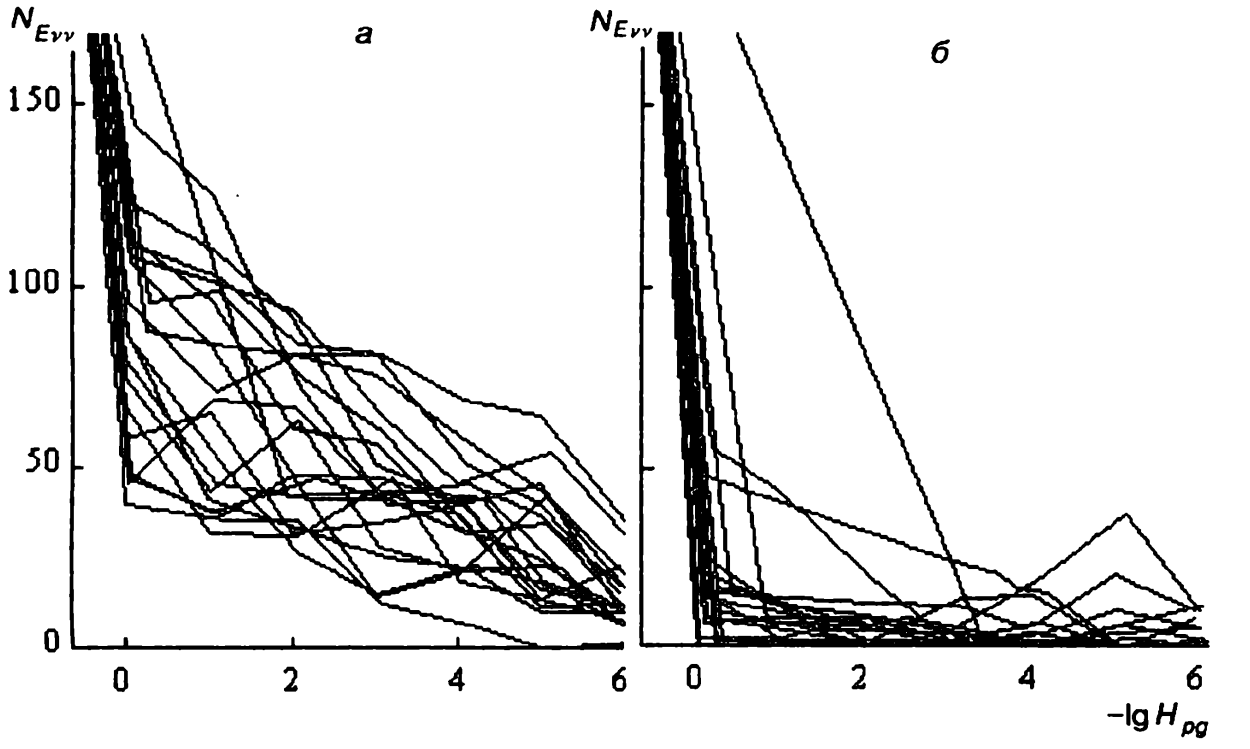


Рис. 3.12. Зависимости числа ошибок, допускаемых сетями на всем задачнике (N_{Err}) от их оценок по обучающей выборке из 32 примеров (H_{pg}) (масштаб полулогарифмический).

а — для сетей с линейными сумматорами; *б* — для сетей с квадратичными сумматорами.

Сети с линейными и квадратичными сумматорами сравнивались между собой. Для этого генерировалось по 20 нейросетей каждого вида с примерно одинаковым числом подстроечных параметров, по 20 шт.

каждого вида. Особенно важно подчеркнуть, что одинаковым было число параметров, а не число нейронов.

Нейросети с квадратичными сумматорами показали более высокую (примерно в 2 — 4 раза) скорость обучения (время измерялось в тактах сети), но главное, — способность по такой же или меньшей обучающей выборке выдавать правильные ответы для большего числа образцов.

Обучались и тестировались полносвязные сети, число тактов срабатывания сети перед получением окончательного ответа принималось равным 3. Для тестирования использовалась задача определения направления циклического сдвига 8-элементного вектора из нулей и единиц. Выходной сигнал снимался с двух нейронов. Сигналы (– 1, 1) трактовались как сдвиг вправо, (1, – 1) — как сдвиг влево. В качестве оценки брался квадрат эвклидова расстояния выходного сигнала от точек (– 1, 1) и (1, – 1) для сдвигов влево и вправо соответственно.

Весь задачник состоял из 504 примеров. Проводилось обучение сети на одной странице (32 примера), после чего сеть тестировалась на всем задачнике. Тестирование для каждой сети проводилось несколько раз — после каждого уменьшения оценки в 10 раз.

На рис. 3.11 и 3.12 приведены полученные результаты. Здесь по оси абсцисс отложены — $\lg_{10} H_{pg}$ для средней оценки H_{pg} по обучающей выборке (странице), содержавшей 32 из возможных 504 примеров. На фрагментах *а* представлены данные для нейросети с квадратичными сумматорами, а на фрагментах *б* — с линейными. На рис. 3.11 по оси ординат отложен — $\lg_{10} H$ средней оценки H по полной выборке — представлена зависимость — $\lg_{10} H$ полученной при тестировании сети, после того как в ходе обучения была достигнута соответствующая оценка H_{pg} по обучающей выборке.

На рис. 3.12 по оси абсцисс представлены зависимости количества ошибок, полученных при тестировании обученной сети на полной выборке, от оценки H_{pg}, достигнутой при обучении на странице из 32 примеров (по оси абсцисс отложены — $\lg_{10} H_{pg}$).

Характеристика нейронов (*c*) принималась равной 0,1. При увеличении характеристики до 1 способность к обобщению у сетей повышалась — сети с квадратичными сумматорами, обученные на одной странице (32 примера) до уровня оценки 0,001, вообще не совершали ошибок при тестировании на всем задачнике.

18. Характеристики нейрона и функции, вычисляемые сетью. Теорема о полноте

Мы неоднократно упоминали о том, что в наших программах характеристическая функция (функция активации) нейрона имеет вид

$$\varphi(A) = \frac{A}{c + |A|}.$$

Речь шла и о других возможностях — вплоть до Паде-нейронов, вычисляющих частное от выходных сигналов двух сумматоров.

Читатель вправе поинтересоваться, чем различаются классы функций, вычисляемые нейронными сетями с разными функциями активации нейронов. В данном разделе дается ответ на этот вопрос. Звучит он просто: для разных гладких нелинейных $\varphi(A)$ классы функций, вычисляемых сетями, не различаются.

Есть различия в скорости обучения, числе необходимых для аппроксимации нейронов и прочих количественных характеристиках, которые нелегко поддаются теоретическому оцениванию. Для их сравнения используются вычислительные эксперименты.

Прежде чем приступить к формальному рассмотрению, отметим, что сеть обычно определяет функцию от входных сигналов на $\mathbf{R}^n$, а реальное функционирование происходит на компактном множестве входных сигналов — кубе в $\mathbf{R}^n$. Функция активации нейрона $\varphi(A)$ задана на всей вещественной прямой $\mathbf{R}$.

Что такое класс функций, вычисляемых сетями с данной функцией активации $\varphi(A)$? Этот класс содержит функции любого числа вещественных аргументов, замкнут относительно сложения (обеспечивается с помощью сумматоров), умножения на число (его производят связи) и суперпозиции функций (обеспечивается возможностью передавать выходные сигналы сети другой сети в качестве входных).

Формализуем сказанное. Пусть для каждого натурального n задано $\mathbf{E}_n$ — некоторое линейное пространство определенных на всем $\mathbf{R}^n$ непрерывных функций от n вещественных переменных x_i $(i = 1, \ldots, n)$. Предположим, что эта последовательность $\mathbf{E}_n$ обладает следующими свойствами:

1) при всех n $1 \in \mathbf{E}_n$ (1 — функция, тождественно равная 1);

2) $\mathbf{E}_n$ включает все n координатных функций $x_i (i = 1, \ldots, n)$;

3) если $f_1, f_2, \ldots, f_k \in \mathbf{E}_n$ и $f \in \mathbf{E}_k$, то $f(f_1, f_2, \ldots, f_k) \in \mathbf{E}_n$;

4) в $\mathbf{E}_1$ существует дважды непрерывно дифференцируемая функция φ, вторая производная которой отлична от нуля в точке x_0.

Теорема о полноте. *При выполнении условий 1—4 для любого прямоугольного параллелепипеда* $W \subset \mathbf{R}^n$ $(W = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n])$, *любой непрерывной функции* f *на* W *и всякого* $\varepsilon > 0$ *существует такая функция* $f_\varepsilon \in \mathbf{E}_n$, *что*

$$\sup_{x \in W} \| f(x) - f_\varepsilon(x) \| < \varepsilon. \tag{3.30}$$

Доказательство. Будем называть функцию $f_\varepsilon(x)$, описанную в формулировке теоремы, нейросетевым ε-приближением f на W. Доказательство опирается на три леммы.

Лемма 1. *Для любого* $\varepsilon > 0$ *в* $\mathbf{E}_1$ *существует нейросетевое* ε-*приближение функции* x^2 *на произвольном отрезке* $[a, b]$.

Доказательство. По формуле Тейлора

$$\varphi(x_0 + \delta x) = \varphi(x_0) + \varphi'(x_0)\delta x + \frac{1}{2}\varphi''(x_0)\ (\delta x)^2 + o((x)^2).$$

По условиям 1 и 2 и в силу того, что $\mathbf{E}_n$ — линейные пространства, получаем, что при любом $\delta > 0$ следующие функции $\Phi_\delta(x)$ принадлежат $\mathbf{E}_1$:

$$\Phi_\delta(x) = 2[\varphi(x_0 + \delta x) - \varphi(x_0) - \varphi'(x_0)\delta x]/[\varphi''(x_0)\delta^2];$$

$$\Phi_\delta(x) = x^2 + o(1)$$

($o(1) \to 0$ при $\delta \to 0$).
Для любого $\varepsilon > 0$ и всякого отрезка $[a, b]$ найдется такое $\delta = \delta(\varepsilon, a, b) > 0$, что функция $\Phi_\delta(x)$ является ε-приближением x^2 на $[a, b]$.

Лемма 2. *Для любого $\varepsilon > 0$ в $\mathbf{E}_2$ существует нейросетевое ε-приближение функции $x_1 \times x_2$ (произведения координат) на произвольном прямоугольнике $[a_1, b_1] \times [a_2, b_2]$.*

Доказательство.

$$x_1 \times x_2 = \frac{1}{2}[(x_1 + x_2)^2 - x_1^2 - x_2^2]. \qquad (3.31)$$

По условию функции x_1, x_2, $(x_1 + x_2)$ принадлежат $\mathbf{E}_2$. По лемме 1 построим $\varepsilon/3$ — приближения функций $(x_1 + x_2)^2$, x_1^2, x_2^2 на отрезках $[a_1 + a_2, b_2 + b_1]$, $[a_1, b_1]$, $[a_2, b_2]$ соответственно. Их линейная комбинация, соответствующая $x_1 \times x_2$ (3.31), является ε-приближением этой функции на прямоугольнике $[a_1, b_1] \times [a_2, b_2]$.

Лемма 3. *Для любого многочлена P от n переменных, любого прямоугольного параллелепипеда $W \subset \mathbf{R}^n$ и всякого $\varepsilon > 0$ существует нейросетевое ε-приближение P на W.*

Доказательство. Согласно лемме 2 существует сколь угодно точное нейросетевое приближение функции двух переменных $x_1 \times x_2$ на любом прямоугольнике $[a_1, b_1] \times [a_2, b_2] \subset \mathbf{R}^2$. Пользуясь предположениями 2, 3 из условий теоремы, из δ-приближений этой функции (при достаточно малых $\delta > 0$) и координатных функций $x_i (i = 1, 2, \ldots, n)$ конструируем ε-приближение P: сначала строим приближения мономов — произведений координатных функций, а затем получаем приближение P как их линейную комбинацию.

Окончательно теорема о полноте следует из лемм 1—3 и классических теорем о равномерных приближениях непрерывных функций полиномами.

Другими словами, теорему о полноте можно выразить так: любая непрерывная функция на замкнутом ограниченном множестве может быть равномерно приближена функциями, вычисляемыми нейронными

сетями, если функция активации нейрона дважды непрерывно дифференцируема и нелинейна. Существуют и другие варианты теорем о полноте [205, 242, 246, 267].

Конечно, явный путь построения приближений, использованный в доказательстве теоремы о полноте, никогда не применяется на практике буквально. Существуют более практичные неявные способы, основанные на минимизации ошибки приближения.

19. Сравнение различных функций активации в вычислительном эксперименте

Задача обучения нейросети рассматривается как задача минимизации средней оценки по обучающему множеству.

Характеристические функции (функции активации) сопоставлялись по критерию "скорость обучения" — сравнивалось число тактов нейросети, потребовавшихся для обучения правильному распознаванию всех примеров в обучающей выборке. (Один такт нейросети — единица дискретного времени, за которое все нейроны обмениваются своими сигналами.)

Для сравнения были выбраны следующие характеристические функции:

а) $y = \mathrm{sign}(x) \,|\, \ln(1 + |x/c|) \,|\, / (1 + |\ln(1 + |x/c|)|)$ (LN);

б) $y = (x/c)\ (1 + |x/c|)$ (SM);

в) $y = \mathrm{th}(x/c)$ (TH).

Все три функции монотонны, симметричны относительно смены знака и непрерывно дифференцируемы. Их значения принадлежат интервалу $(-1, 1)$, а производные в нуле равны $1/c$. Сравниваемые функции отличаются скоростью выхода на насыщение; эта скорость наименьшая в случае (LN) и наибольшая для (TH). Асимптотическое поведение величины $1 - |y|$ при $x \to \pm\infty$ имеет порядок: $0(1/\ln(x))$ (LN), $0(1/x)$ (SM) и $0(\exp(-|x|))$ (TH).

Сравнение проводилось для двух задач бинарной классификации:

1) определения направления циклического сдвига конечной двоичной последовательности (с элементами 0, 1);

2) определения симметричности или несимметричности конечной двоичной последовательности.

Более детальные описания задач и оценок даны в разд. 3.10.

Для сравнения методов были сгенерированы по 100 нейросетей для каждой из тестовых задач. Перед обучением параметры нейросетей ("синаптические веса") выбирались случайно с равномерным распределением в интервале $(-0;\ 0{,}1 \div 0{,}01)$. Обучение велось квазиньютоновским методом, основанным на BFGS-формуле.

Вычислительный эксперимент проводился С.Е. Гилевым [35], результаты публикуются с его любезного разрешения (табл. 3.9).

Полученные результаты позволяют сделать следующие выводы:

Таблица 3.9

Время обучения нейросети, тыс. тактов

$f(c)$	AV	σ	min	max
Задача определения направления сдвига				
LN	296	76	190	635
SM	296	102	191	1028
TH	383	345	201	3349
Задача определения симметрии последовательности				
LN	864	621	382	6074
SM	731	259	388	1831
TH	699	135	494	1149

1) скорость обучения существенно зависит от выбора характеристической функции, причем для различных тестовых задач оптимальный выбор оказывается различным;

2) наилучшим выбором (по результатам тестовых испытаний) является SM — на обеих тестовых задачах эта функция дала результаты, близкие к наилучшим, в то время как обе другие функции на одной из задач показывают значительно худшие результаты.

Именно этого выбора

$$\varphi(A) = \frac{A}{c + |A|}$$

мы обычно придерживаемся.

20. Контрастирование и логически прозрачные нейронные сети

В гл. 1 описаны основные идеи и методы контрастирования нейронных сетей. Если мы вычисляем градиент функции оценки H по параметрам сети и входным сигналам, то этим легко воспользоваться для вычисления показателей значимости и контрастирования сети. Операция контрастирования будет надстраиваться над процессом обучения и потребует совсем немного дополнительных ресурсов. Показатели значимости могут быть усреднены по нескольким шагам обучения — это позволит избежать исключения параметров α, для которых производная $|\partial H/\partial\alpha|$ близка к нулю в данной точке, но резко возрастает в ее окрестности. Другой прием, служащий той же цели, — усреднение величин $|\partial H/\partial\alpha|$, вычисленных в конечном множестве точек, полученных случайным изменением параметров. Задается некоторый диапазон для случайного сдвига параметров, для каждого параметра заданное число раз генерируется случайный сдвиг, в новых точках вычисляются величины $|\partial H/\partial\alpha|$, и они усредняются. Именно этот прием применяется в серии программ MultiNeuron.

С помощью анализа показателей значимости решаются следующие задачи:

1) сокращение числа параметров и входных сигналов за счет отбрасывания наименее значимых;

2) уменьшение разрядности параметров сети (приведение их к небольшому числу выделенных значений — это особенно важно при проектировании нейросетевых устройств);

3) приведение структуры нейронной сети к специальному "логически прозрачному" (т.е. понятному и допускающему осмысленный словесный пересказ) виду.

В пояснениях нуждается лишь третья задача. Что такое логически прозрачные сети и как их строить с помощью контрастирования? Сразу следует сказать, что одного раз и навсегда данного определения логической прозрачности быть не может. В зависимости от контекста может меняться представление о простоте и понятности структуры сети. Все же несколько признаков и показателей прозрачности можно привести:

1) чем меньше нейронов, тем прозрачнее сеть;

2) желательно, чтобы связи между нейронами не содержали циклов или (хотя бы) чтобы различных простых циклов было как можно меньше;

3) чем меньше входов приходится на один нейрон, тем прозрачнее сеть (показателем прозрачности здесь может служить максимум числа входных сигналов у нейрона, максимум берется по всем нейронам);

4) чем меньше входных сигналов сеть использует для решения задачи, тем она прозрачнее;

5) монотонные сети прозрачнее немонотонных.

Список нетрудно продолжить. Общая схема достижения логической прозрачности такова: строится заведомо избыточная сеть одной из стандартных архитектур (слоистая, полносвязная, ...). Можно сразу заложить в архитектуру некоторые требования прозрачности, например монотонность. Сеть учится решать задачу, и после некоторого предварительного обучения мы начинаем уничтожать наименее значимые связи из наиболее мешающих прозрачности.

Поясним на примерах. Пусть важнейшим показателем прозрачности считается число входных сигналов на один нейрон. Тогда следует взять те нейроны, для которых количество входных сигналов максимально, и попытаться для каждого из них сократить это количество хотя бы на единицу. Если же, например, важнейшим показателем считается число нейронов, то следует попытаться уничтожить тот нейрон (т.е. все его связи), для которого сумма показателей значимости связей минимальна, и т.п.

Комбинируя различные показатели прозрачности и проводя контрастирование соответствующим образом, мы продвигаемся от менее прозрачных сетей к более прозрачным. Со временем здесь возникнет обширная область логики и, возможно, философии, а пока сказанного об этом достаточно.

Добавим лишь небольшой пример. При попытке построить логически прозрачную сеть для прогнозирования результатов президентских выборов в США были получены следующие содержательные результаты.

1. Из 12 признаков ситуации (см. гл. 1 первой части) можно ограничиться пятью ответами на вопросы:

(3) была ли во время выборов активна третья партия?

(4) была ли серьезная конкуренция при выдвижении кандидата от правящей партии?

(6) был ли год выборов годом экономического спада или депрессии?

(8) произвел ли правящий президент серьезные изменения в политике?

(9) имели ли место серьезные социальные волнения?

2. Построенная слоистая нейронная сеть имеет в первом слое всего два нейрона. Эти нейроны преобразуют "симптомы" — входные сигналы (ответы на вопросы, кодируемые так: + 1 — "да", – 1 — "нет") в "синдромы" первого уровня. Сумматор первого нейрона вычисляет сумму $x_3 + x_4 + x_9$, сумматор второго — разность $x_6 - x_8$. Дальнейшее зависит только от значения этих сумм. Первая из них может интерпретироваться как "синдром политической нестабильности", вторая имеет интересный смысл "депрессия минус новаторство" (новатору прощается экономический спад — может быть, так?). Два этих числа полностью характеризуют ситуацию; сеть, обученная на результатах выборов, происходивших до 1980 г., уверенно предсказала все дальнейшие события.

Приложение

MULTINEURON 2.0 — ПРОГРАММНЫЙ ИНСТРУМЕНТАРИЙ ДЛЯ СОЗДАНИЯ НЕЙРОСЕТЕВЫХ ЭКСПЕРТНЫХ СИСТЕМ

Разрабатывая программный инструментарий, позволяющий создавать нейросетевые экспертные системы, мы определили требования, которым он должен удовлетворять.

1. Дружественный интерфейс, соответствующий общепринятым стандартам, например стандарту System Application Architecture (SAA) фирмы IBM:

а) система меню и "горячих" клавиш;

б) оконная организация;

в) полная обработка ошибок — как системных, так и ошибок пользователя;

г) возможность отмены действий пользователя и предупреждения перед совершением необратимых действий (например, перезаписи файла) с возможностью отмены;

д) наличие контекстной подсказки (помощи) в любой точке программы в любой момент времени;

е) поддержка "мыши".

2. Возможность работы под управлением операционной системы MS-DOS, используемой на большинстве IBM — совместимых персональных компьютерах, импортируемых и выпускаемых в России.

3. Совместимость с каким-либо общепринятым и широкоиспользуемым стандартом компьютерных баз данных.

4. Возможность создания и обучения нейронных сетей без знания пользователем математики и нейроинформатики.

5. Возможность конструирования пользователем нейросетевых экспертных систем без знания им языков программирования.

Был также определен круг знаний и навыков, необходимых и достаточных для пользователя, конструирующего экспертную систему в своей профессиональной области с помощью пакета MultiNeuron 2.0.

1. Пользовательские навыки работы с компьютером:

а) файловая организация операционной системы MS-DOS (создание, чтение, запись, перезапись, стирание файлов и каталогов);

б) запуск программ.

2. Способы и принципы числового кодирования (представления) профессиональной информации.

3. Умение определить профессиональную задачу (ответ и набор входных параметров — условий).

Рассмотрим устройство, возможности и принцип работы программного комплекса MultiNeuron 2.0, а также последовательность действий пользователя при создании им нейросетевой экспертной системы.

Пакет программ написан на языке высокого уровня Borland Pascal 7.0 с использованием модифицированной автором объектной библиотеки Turbo Vision 2.0 и библиотеки поддержки баз данных Paradox-стандарта Paradox Engine 3.5.

Пакет MultiNeuron 2.0 состоит из трех программ.

1. Trainer (обучатель) — программа для создания, обучения и тестирования нейронных сетей, для проведения различных экспериментов с нейросетями, для реализации дополнительных возможностей нейронных сетей (вычисление значимости обучающих параметров, минимизация параметров, исключение примеров и др.).

2. Constructor (сборщик экспертной системы) — программа для конструирования элементов экспертных систем: структуры базы данных, окна редактора данных, окна вывода результатов работы системы, определение списка нейронных сетей, которые будут работать в создаваемой экспертной системе. Сконструированные элементы помещаются в специальный файл-ресурс, используемый в дальнейшем программой-экспертной системой.

3. NES (Neural Expert System — нейросетевая экспертная система) — программа, представляющая базовый каркас экспертной системы и потребляющая файл-ресурс, созданный предыдущей программой, и в зависимости от элементов, содержащихся в нем, приобретает свойства, заданные пользователем в программе Constructor.

Пакет MultiNeuron 2.0 работает со следующими типами файлов:

1. Файлы баз данных (расширение DB) стандарта Paradox — одной из самых широкоиспользуемых в России и за рубежом систем управления базами данных. Используются для хранения примеров для обучения и тестирования, а пользователь созданной экспертной системы хранит и накапливает в них текущую информацию, с которой она работает.

2. Файлы нейронных сетей (расширение NN). В каждом хранится одна нейронная сеть, решающая установленную подзадачу.

3. Файлы подзадач (проекты обучения, расширение PR), содержащие информацию, какую нейронную сеть, на каких данных и как обучать (стратегия, параметры, установки).

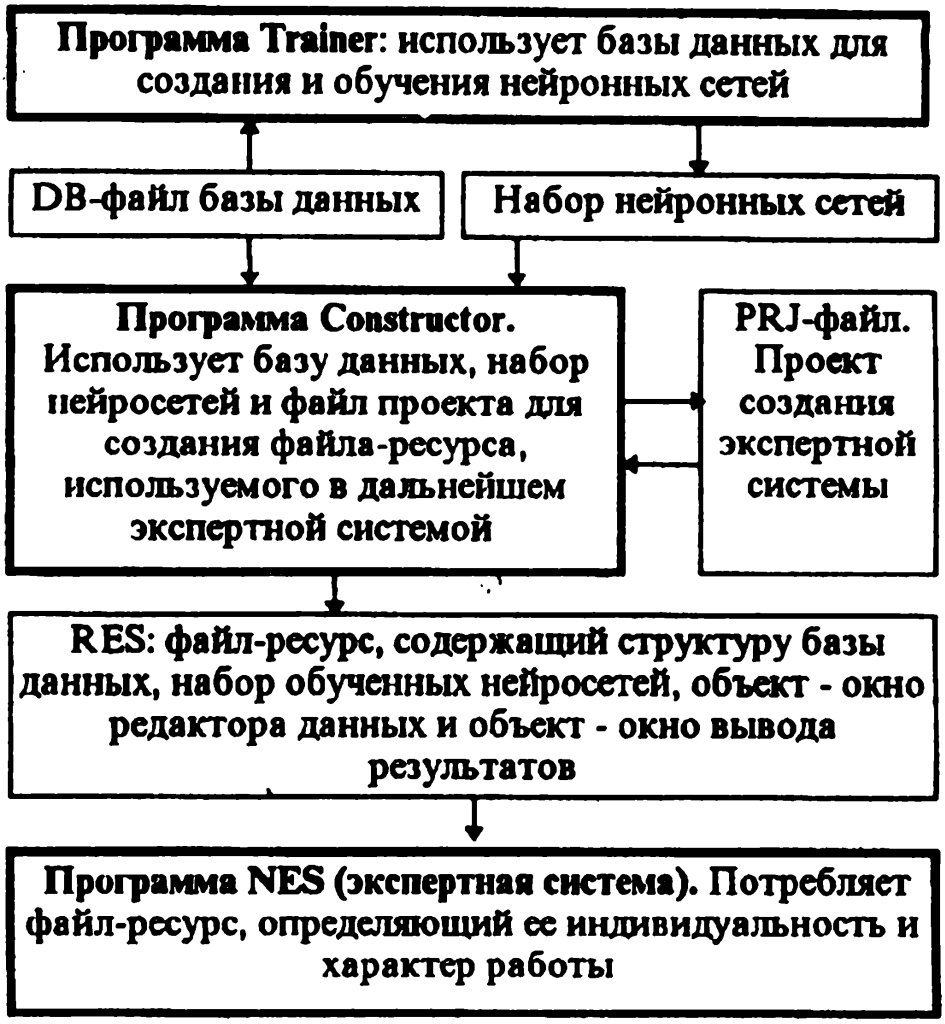


Рис. П.1. Схема взаимодействия программ в пакете Multi-Neuron 2.0.

4. Файлы задач (суперпроекты обучения, расширение SP), представляющие собой списки файлов подзадач, логически связанных одной темой.

5. Файлы проектов экспертных систем (расширение PRJ). Каждый из них хранит схему, определяющую элементы создаваемой экспертной системы: указатель на базу данных, структура которой будет использоваться в экспертной системе; указатели на один или несколько файлов нейронных сетей, обученных для экспертной системы; проект окна редактора и проект окна вывода результатов.

6. Файлы-ресурсы (расширение RES), потребляемые программой NES и содержащие соответственно файлам-проектам структуру базы данных, комплект нейронных сетей, объект — окно редактора данных и объект — окно вывода результатов (рис. П.1).

Пользователь начинает работу по созданию экспертной системы с использования программы Trainer. В этой программе может быть создана и заполнена база данных, содержащая примеры (каждая запись (строка) базы данных рассматривается как пример с известным или неизвестным ответом). База данных может быть создана в другой программе, поддерживающей формат баз данных Paradox. Затем эта

база данных может использоваться программой Trainer. Программа позволяет производить с базой данных большинство стандартных операций, осуществляемых системами управления базами данных (создание, чтение, запись, редактирование структуры и данных и др.).

Внутри программы Trainer пользователь создает нейронные сети, связанные с выбранными им параметрами (полями) любой базы данных. При создании сети программа автоматически связывает каждый входной и выходной синапс нейросети с полем базы данных, имеющим уникальное имя. Это обеспечивает корректное нахождение и использование данных нейросетью при обучении и тестировании независимо от структуры базы данных и порядка расположения в ней обучающих параметров. При этом пользователю нет необходимости следить за подачей сигналов на синапсы нейросети. Созданные им нейронные сети записываются на диск и могут храниться как угодно долго.

Объем, занимаемый файлом нейронной сети на магнитном носителе, очень мал и приблизительно вычисляется как 6 × квадрат числа нейронов (реальный объем на несколько сотен байт больше, так как в файл нейросети записываются еще несколько ее параметров и имена полей базы данных, с которыми связаны входные и выходные синапсы). Так, объем нейросети с 20 нейронами составляет чуть более 3 Кб. Поэтому на диске одного персонального компьютера можно разместить огромное количество нейронных сетей, используемых сотнями экспертных систем.

Для удобства и обеспечения четкой логики в работе пользователь имеет возможность определения подзадач и задач, сохраняя директивы стратегии обучения в специальных файлах.

Обучение нейросетей в программе Trainer осуществляется в одном из трех режимов по выбору пользователя.

1. *Рабочий режим.* Пользователь обучает одну выбранную нейросеть на выбранной базе данных. Как правило, этот режим используется для создания очень простых односетевых экспертных систем, либо для экспериментов, когда обучение проводится на глазах у пользователя.

2. *Режим обучения подзадаче* (обучение по проекту). Обучается также одна нейросеть, либо набор одинаковых нейросетей-экспертов, решающих одну и ту же подзадачу. Используется стратегия, установленная пользователем в файле проекта, к которому программа Trainer обращается для определения действий. Обучение по проекту ведется в автономном режиме и не требует присутствия пользователя. Если определено поведение программы в случае возникновения тех или иных ситуаций или установлена автоматическая стратегия, то программа предпримет оптимальные действия для достижения конечной цели.

При задании стратегии обучения в проекте пользователь определяет поведение программы в случае успешного обучения и в случае невозможности обучения на каком-то этапе. Предусмотрены режимы изменения уровня надежности/отклонения в любую сторону, режим

исключения трудных примеров и режим минимизации параметров. Возможно задание полностью автоматической стратегии, при этом программа ищет наиболее оптимальные режимы обучения для достижения определенной пользователем цели с максимальной эффективностью.

3. *Режим обучения задаче* (обучение по суперпроекту). Пользователь задает список подзадач, обычно связанных одной темой. Список может иметь неограниченное количество подзадач. Программа может автономно работать на протяжении многих дней и даже недель без вмешательства пользователя, последовательно обучая сотни и тысячи нейронных сетей. Каждый час она записывает на диск текущую обучаемую нейросеть (предотвращение потери результатов обучения при выключении электропитания). После обучения каждой подзадаче программа отмечает в специальном файле результат работы и время его окончания. Предусмотрена защита от влияния временного выключения электропитания на процесс обучения. Установка автоматического запуска программы Trainer при включении персонального компьютера в сеть обеспечивает возобновление работы программы после временного выключения электропитания, начиная с последней сохраненной точки.

Программа Trainer позволяет проверять обученность нейросетей путем тестирования примеров любой базы данных. Предусмотрено три режима тестирования.

1. Оперативный режим тестирования одного примера. Тестируется текущий (установленный в редакторе) пример текущей базы данных нейронной сетью, загруженной в программу.

2. Тестирование всей текущей базы данных. Тестируются все либо указанные пользователем примеры текущей базы данных нейросетью, загруженной в программу.

3. Тестирование согласно подзадаче (по проекту). Тестируется список примеров базы данных, определенной в файле проекта, нейросетью, также определенной в проекте.

Результаты любого теста выводятся в специальном окне с подробной информацией об известном и вычисленном нейросетью ответе, коэффициенте уверенности и ответах нейронов. Текст, выводимый в окне теста, может быть оперативно распечатан на принтере и/или помещен в текстовый файл с возможностью перезаписи или добавления.

Программа Trainer дает возможность подсчета значимости обучающих параметров. Операция может производиться как по прямому указанию пользователя, так и автоматически согласно заданной стратегии обучения, в этом случае результаты подсчета значимости параметров используются программой для определения дальнейших действий на каком-либо этапе решения подзадачи.

При создании пользователем экспертной системы работа с программой Trainer заканчивается после обучения всех созданных нейронных сетей, после чего начинается работа с программой Constructor.

В программе Constructor пользователь прежде всего определяет базу данных, структура (шаблон данных) которой будет использоваться создаваемой экспертной системой.

Важной частью работы пользователя с экспертной системой является ввод исходных данных. Для удобного ввода данных необходим индивидуальный в каждой экспертной системе редактор данных с пояснениями и расшифровкой информации. Программа Constructor обеспечивает пользователю возможность создания редактора со вставкой в его окно нескольких типов объектов ввода информации — строк ввода для текстовой и числовой информации, дат, а также пунктов одиночного и множественного выбора. Предусматривается возможность отсутствия вводимой информации и обработка таких ситуаций редактором. Объекты ввода информации связываются пользователем с полями установленной базы данных и обеспечивают размещение информации при редактировании в соответствующие ячейки базы данных.

На следующем этапе пользователь определяет список нейронных сетей, которые будут использоваться экспертной системой, и последовательность работы с ними.

Программа Constructor обеспечивает также конструирование окна вывода результатов экспертной системы. При этом пользователь определяет произвольные тексты, выводимые в окно, и текстовую интерпретацию ответов каждой нейронной сети, используемой экспертной системой и определенной на предыдущем этапе.

Созданные пользователем установки могут быть сохранены в файле-проекте экспертной системы и записаны на диск для обеспечения возможности последующего редактирования и конструирования похожих экспертных систем.

На основе созданных установок пользователь создает файл-ресурс, в который помещаются готовые объекты (структура базы данных, окно редактора с объектами ввода, нейросети, определенные в списке, и окно вывода результатов). Имя файла-ресурса определяет пользователь. По желанию пользователя одновременно с файлом-ресурсом программа может создать программу экспертной системы путем копирования файла NES.EXE в другой файл, имя которого совпадает с файлом создаваемого ресурса. На этом создание экспертной системы заканчивается и она готова к работе.

Основой экспертной системы служит программа NES.EXE. При ее запуске пользователем она ищет файл-ресурс со своим именем. Поэтому пользователь может создать любое количество экспертных систем, копируя файл программы NES в новые файлы с другими именами и присоединяя к ним файлы-ресурсы с соответствующими именами (как указывалось выше, эта операция может производиться автоматически при создании ресурса программой Constructor).

Программа NES обеспечивает работу пользователя с базой данных только такой структуры, которая задана в файле-ресурсе. При создании нового файла базы данных или загрузке его с диска программа

помещает на экран окно со списком записей (в списке выводятся данные первого поля). Пользователь имеет возможность вставлять записи в любое место базы, добавлять их в конец базы и удалять. При вызове операции редактирования программа выводит на экран окно редактора (заданного в файле-ресурсе) и помещает в него данные из текущей записи списка.

Тест записи нейронными сетями может проводиться как из окна редактора, так и непосредственно из списка, при этом выводится окно с результатами теста (по шаблону, заданному в файле-ресурсе), которые можно распечатать на принтере или поместить в текстовый файл с возможностью перезаписи или добавления информации.

Таким образом, программный комплекс MultiNeuron 2.0 обеспечивает выполнение пользователем функций и операций, необходимых как для создания прикладной нейросетевой экспертной системы, так и для проведения исследований медико-биологических данных.

СПИСОК ЛИТЕРАТУРЫ[1]

1. **Айвазян С.А., Бежаева З.И., Староверов О.В.** Классификация многомерных наблюдений. — М.: Статистика, 1974. — 240 с.
2. **Айзерман М.А., Браверман Э.М., Розоноэр Л.И.** Метод потенциальных функций в теории обучения машин. — М.: Наука, 1970. — 383 с.
3. **Анастази А.** Психологическое тестирование. — М.: Педагогика, 1982. — Кн. 1. — 320 с.; Кн. 2. — 360 с.
4. **Андерсон Т.** Введение в многомерный статистический анализ. — М.: Физматгиз, 1963. — 500 с.
5. **Андерсон Т.** Статистический анализ временных рядов. — М.: Мир, 1976. — 755 с.
6. **Ануфриев А.Ф.** Психодиагностика как деятельность и научная дисциплина // Вопросы психологии. — 1994. — № 2. — С. 123—130.
7. **Аркадьев А.Г., Браверман Э.М.** Обучение машины классификации объектов. — М.: Наука, 1971. — 172 с.
8. **Биотехника** — новое направление компьютеризации / Ю.К. Ахапкин, Н.И. Всеволдов, С.И. Барцев и др. — М.: изд. ВИНИТИ, 1990. — 144 с. — (Серия "Теоретическая и прикладная биофизика".)
9. **Барцев С.И.** Некоторые свойства адаптивных сетей. — Красноярск, 1987. — 17 с. — (Препр. / ИФ СО АН СССР; № 71Б.)
10. **Барцев С.И., Гилев С.Е., Охонин В.А.** Принцип двойственности в организации адаптивных систем обработки информации // Динамика химических и биологических систем. — Новосибирск: Наука. Сиб. отд-ние, 1989. — С. 6—55.
11. **Барцев С.И., Ланкин Ю.П.** Сравнительные свойства адаптивных сетей с полярными и неполярными синапсами. — Красноярск, 1993. — 26 с. — (Препр. / Ин-т биофизики СО РАН; № 196Б.)
12. **Барцев С.И., Ланкин Ю.П.** Моделирование аналоговых адаптивных сетей. — Красноярск, 1993. — 36 с. — (Препр. / Ин-т биофизики СО РАН; № 203Б.)

[1]Список литературы заведомо неполон. В него включены классические работы, монографии и статьи обзорного характера, а также публикации,непосредственно использованные в книге. Кроме того, в список включены все сколько-нибудь существенные публикации членов красноярской группы НейроКомп, вышедшие к моменту сдачи книги в печать.

13. **Барцев С.И., Машихина Н.Ю., Суров С.В.** Нейронные сети: подходы к аппаратной реализации. — Красноярск, 1990. — 14 с. — (Препр. / ИФ СО АН СССР; № 122Б.)
14. **Барцев С.И., Охонин В.А.** Адаптивные сети обработки информации. — Красноярск, 1986. — 20 с. — (Препр. / ИФ СО АН СССР; № 59Б.)
15. **Барцев С.И., Охонин В.А.** Адаптивные сети, функционирующие в непрерывном времени // Эволюционное моделирование и кинетика. — Новосибирск: Наука. Сиб. отд-ние, 1992. — С. 24—30.
16. **Беркинблит М.Б., Гельфанд И.М., Фельдман А.Г.** Двигательные задачи и работа параллельных программ // Интеллектуальные процессы и их моделирование. Организация движения. — М.: Наука, 1991. — С. 37—54.
17. Болезни сердца и сосудов / Под ред. Е.И. Чазова. — М.: Медицина, 1992. — Т. 2. — С. 90.
18. **Боннер Р.Е.** Некоторые методы классификации // Автоматический анализ изображений. — М.: Мир, 1969. — С. 205—234.
19. **Борисов А.В., Гилев С.Е., Головенкин С.Е., Горбань А.Н., Догадин С.А., Коченов Д.А., Масленникова Е.В., Матюшин Г.В., Миркес Е.М., Ноздрачев К.Г., Россиев Д.А., Савченко А.А., Шульман В.А.** Нейроимитатор "MultiNeuron" и его применения в медицине // Математическое обеспечение и архитектура ЭВМ: Материалы науч.-техн. конф. "Проблемы техники и технологий XXI века", 22—25 марта 1994 г. — Красноярск: КГТУ, 1994. — С. 14—18.
20. **Браверман Э.М., Мучник И.Б.** Структурные методы обработки эмпирических данных. — М.: Наука. Гл. ред. физ.-мат. лит., 1983. — 464 с.
21. **Букатова И.Л.** Эволюционное моделирование и его приложения. — М.: Наука, 1979. — 231 с.
22. **Бурлачук Л.Ф., Коржова Е.Ю.** К построению теории "измеренной индивидуальности" в психодиагностике // Вопр. психологии. — 1994. — № 5. — С. 5—11.
23. **Вавилов Е.И., Егоров Б.М., Ланцев В.С., Тоценко В.Г.** Синтез схем на пороговых элементах. — М.: Сов. радио, 1970.
24. **Вапник В.Н., Червоненкис А.Ф.** Теория распознавания образов. — М.: Наука, 1974.
25. **Веденов А.А.** Моделирование элементов мышления. — М.: Наука, 1988.
26. **Гаврилова Т.А., Червинская К.Р., Яшин А.М.** Формирование поля знаний на примере психодиагностики // Изв. АН СССР. Техн. кибернетика. — 1988. — № 5. — С. 72—85.
27. **Галушкин А.И.** Синтез многослойных схем распознавания образов. — М.: Энергия, 1974.
28. **Галушкин А.И., Фомин Ю.И.** Нейронные сети как линейные последовательные машины. — М.: Изд-во МАИ, 1991.
29. **Гельфанд И.М., Цетлин М.Л.** О математическом моделировании механизмов центральной нервной системы // Модели структурно-функциональной организации некоторых биологических систем. — М.: Наука, 1966. — С. 9—26.
30. **Гилев С.Е.** Гибрид сети двойственности и линейной сети // Нейроинформатика и нейрокомпьютеры: Тез. докл. рабочего семинара, 8—11 октября 1993 г. — Красноярск: Ин-т биофизики СО РАН, 1993. — С. 25.
31. **Гилев С.Е.** Метод получения градиентов оценки по подстроечным параметрам без использования back propagation // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 91—100.
32. **Гилев С.Е.** Нейросеть с квадратичными сумматорами // Нейроинформатика и нейрокомпьютеры: Тез. докл. рабочего семинара, 8—11 октября 1993 г. — Красноярск: Ин-т биофизики СО РАН, 1993. — С. 11—12.

33. **Гилев С.Е.** Forth-propagation — метод вычисления градиентов оценки // Тез. докл. II Всероссийского рабочего семинара "Нейроинформатика и ее приложения", Красноярск, 7—10 октября 1994 г. — Красноярск: изд. КГТУ, 1994. — С. 36—37.
34. **Гилев С.Е.** "Сравнение методов обучения нейронных сетей" // Тез. докл. III Всероссийского семинара "Нейроинформатика и ее приложения". — Красноярск: изд. КГТУ, 1995. — С. 80—81.
35. **Гилев С.Е.** "Сравнение характеристических функций нейронов" // Тез. докл. III Всероссийского семинара "Нейроинформатика и ее приложения". — Красноярск: изд. КГТУ, 1995. — С. 82.
36. **Гилев С.Е., Горбань А.Н., Миркес Е.М.** Малые эксперты и внутренние конфликты в обучаемых нейронных сетях // Докл. АН СССР. — 1991. — Т. 320, № 1. — С. 220—223.
37. **Гилев С.Е., Горбань А.Н., Миркес Е.М. и др.** Определение значимости обучающих параметров для принятия нейронной сетью решения об ответе // Нейроинформатика и нейрокомпьютеры: Тез. докл. рабочего семинара, 8—11 октября 1993 г. — Красноярск: Институт биофизики СО РАН, 1993. — С. 8.
38. **Гилев С.Е., Горбань А.Н., Миркес Е.М. и др.** Нейросетевая программа MultiNeuron // Там же. — С. 9.
39. **Гилев С.Е., Горбань А.Н., Миркес Е.М. и др.** Пакет программ имитации различных нейронных сетей // Там же. — С. 7.
40. **Гилев С.Е., Коченов Д.А., Миркес Е.М., Россиев Д.А.** Контрастирование, оценка значимости параметров, оптимизация их значений и их интерпретация в нейронных сетях // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 66—78.
41. **Гилев С.Е., Миркес Е.М., Новоходько А.Ю. и др.** Проект языка описания нейросетевых автоматов // Тез. докл. II Всероссийского рабочего семинара "Нейроинформатика и ее приложения", Красноярск, 7—10 октября, 1994 г. — Красноярск: изд. КГТУ, 1994. — С. 35.
42. **Гилев С.Е., Миркес Е.М.** Обучение нейронных сетей // Эволюционное моделирование и кинетика. — Новосибирск: Наука. Сиб. отд-ние, 1992. — С. 9—23.
43. **Гилева Л.В., Гилев С.Е., Горбань А.Н.** Нейросетевой бинарный классификатор "CLAB" (описание пакета программ). — Красноярск, 1992. — 25 с. — (Препр. / Ин-т биофизики СО РАН; № 194Б.)
44. **Гилева Л.В., Гилев С.Е., Горбань А.Н. и др.** Нейропрограммы: Учебное пособие: В 2 ч. // Красноярск: Краснояр. гос. техн. ун-т, 1994. — 260 с.
45. **Гилл Ф., Мюррей У., Райт М.** Практическая оптимизация. — М.: Мир, 1985. — 509 с.
46. **Головенкин С.Е., Назаров Б.В., Матюшин Г.В. и др.** Прогнозирование возникновения мерцательной аритмии в острый и подострый периоды инфаркта миокарда с помощью компьютерных нейронных сетей // Актуальные проблемы реабилитации больных с сердечно-сосудистыми заболеваниями: Тез. докл. симп., 18—20 мая 1994 г. — Красноярск, 1994. — С. 28.
47. **Головенкин С.Е., Россиев Д.А., Назаров Б.В. и др.** Прогнозирование возникновения фибрилляции предсердий как осложнения инфаркта миокарда с помощью нейронных сетей // Диагностика, информатика и метрология-94: Тез. докл. науч.-техн. конф., Санкт-Петербург, 28—30 июня 1994 г. — СПб., 1994. — С. 349.
48. **Головенкин С.Е., Россиев Д.А., Шульман В.А. и др.** Прогнозирование сердечной недостаточности у больных со сложными нарушениями сердечного ритма с помощью нейронных сетей // Там же. — С. 350—351.

49. **Горбань А.Н.** Алгоритмы и программы быстрого обучения нейронных сетей // Эволюционное моделирование и кинетика. — Новосибирск: Наука. Сиб. отд-ние, 1992. — С. 36—39.
50. **Горбань А.Н.** Двойственность в сетях автоматов // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 32—66.
51. **Горбань А.Н.** Мы предлагаем для контроля качества использовать нейрокомпьютеры // Стандарты и качество. — 1994. — № 10. — С. 52.
52. **Горбань А.Н.** Нейрокомп // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 3—31.
53. **Горбань А.Н.** Нейрокомпьютер, или аналоговый ренессанс // Мир ПК. — 1994. — № 10. — С. 126—130.
54. **Горбань А.Н.** Обучение нейронных сетей. — М.: изд. СССР — США СП "ParaGraph", 1990. — 160 с. (Пер. на англ.: AMSE Trans., Scientific Siberian, A. — 1993. — Vol. 6. Neurocomputing. — P. 1—134).
55. **Горбань А.Н.** Проекционные сетчатки для обработки бинарных изображений // Математическое обеспечение и архитектура ЭВМ: Материалы науч.-техн. конф. "Проблемы техники и технологий XXI века", 22—25 марта 1994 г. — Красноярск: изд. КГТУ, 1994. — С. 50—54.
56. **Горбань А.Н.** Размытые эталоны в обучении нейронных сетей // Тез. докл. II Всероссийского рабочего семинара "Нейроинформатика и ее приложения", Красноярск, 7—10 октября 1994 г. — Красноярск: изд. КГТУ, 1994. — С. 6—9.
57. **Горбань А.Н.** Этот дивный новый компьютерный мир. Заметки о нейрокомпьютерах и новой технической революции // Математическое обеспечение и архитектура ЭВМ: Материалы науч.-техн. конф. "Проблемы техники и технологий XXI века", 22—25 марта 1994 г. — Красноярск: изд. КГТУ, 1994. — С. 42—49.
58. **Горбань А.Н.** Системы с наследованием и эффекты отбора // Эволюционное моделирование и кинетика. — Новосибирск: Наука. Сиб. отд-ние, 1992. — С. 40—71.
59. **Горбань А.Н., Кошур В.Д.** Нейросетевые модели и методы решения задач динамики сплошных сред и физики взаимодействующих частиц // 10-я Зимняя школа по механике сплошных сред: Тез. докл. — Екатеринбург: УрО РАН, 1995. — С. 75—77.
60. **Горбань А.Н., Миркес Е.М.** "Компоненты нейропрограмм" // Тез. докл. III Всероссийского семинара "Нейроинформатика и ее приложения". — Красноярск: изд. КГТУ, 1995. — С. 17.
61. **Горбань А.Н., Миркес Е.М.** Нейросетевое распознавание визуальных образов "EYE": описание пакета программ. — Красноярск, 1992. — 36 с. — (Препр. / Ин-т биофизики СО РАН; № 193Б.)
62. **Горбань А.Н., Миркес Е.М.** Кодирование качественных признаков для нейросетей // Тез. докл. II Всероссийского рабочего семинара "Нейроинформатика и ее приложения", Красноярск, 7—10 октября 1994 г. — Красноярск: изд. КГТУ, 1994. — С. 29.
63. **Горбань А.Н., Миркес Е.М.** Контрастирование нейронных сетей // Тез. докл. III Всероссийского семинара "Нейроинформатика и ее приложения". — Красноярск: изд. КГТУ, 1995. — С. 78—79.
64. **Горбань А.Н., Миркес Е.М.** Логически прозрачные нейронные сети // Там же. — С. 32.
65. **Горбань А.Н., Миркес Е.М.** Функциональные компоненты нейрокомпьютера // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 79—90.

66. **Горбань А.Н., Миркес Е.М., Свитин А.П.** Полуэмпирический метод классификации атомов и интерполяции их свойств // Математическое моделирование в химии и биологии: Новые подходы. — Новосибирск: Наука. Сиб. отд-ние, 1992. — С. 204—220.
67. **Горбань А.Н., Миркес Е.М., Свитин А.П.** Метод мультиплетных покрытий и его использование для предсказания свойств атомов и молекул // Журн. физ. химии. — 1992. — Т. 66, № 6. — С. 1503—1510.
68. **Горбань А.Н., Россиев Д.А., Бутакова Е.В., Гилев С.Е., Головенкин С.Е., Догадин С.А., Коченов Д.А.. Масленникова Е.В., Матюшин Г.В., Миркес Е.М., Назаров Б.В., Ноздрачев К.Г., Савченко А.А., Смирнова С.В., Чертыков П.А., Шульман В.А.** Медицинские и физиологические применения нейроимитатора "Multi-Neuron" // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 101—113.
69. **Горбань А.Н., Россиев Д.А., Коченов Д.А.** Применение самообучающихся нейросетевых программ. Разд. 1. Введение в нейропрограммы: Учебно-методическое пособие для студентов специальностей 22.04 и 55.28.00 всех форм обучения. — Красноярск: СТИ, 1994. — 24 с.
70. **Горбань А.Н., Фриденберг В.И.** Новая игрушка человечества // МИР ПК. — 1993. — № 9. — С. 111—113.
71. **Горбань А.Н., Хлебопрос Р.Г.** Демон Дарвина. Идея оптимальности и естественный отбор. — М.: Наука, 1988. — 208 с.
72. **Грановская Р.М., Березная И.Я.** Интуиция и искусственный интеллект. — Л.: Изд-во ЛГУ, 1991. — 272 с.
73. **Гутчин И.Б., Кузичев А.С.** Бионика и надежность. — М.: Наука, 1967.
74. **Демиденко Е.З.** Линейная и нелинейная регрессия. — М.: Финансы и статистика, 1981. — 302 с.
75. **Деннис Дж. Мл., Шнабель Р.** Численные методы безусловной оптимизации и решения нелинейных уравнений. — М.: Мир, 1988. — 440 с.
76. **Дертоузос М.** Пороговая логика. — М.: Мир, 1967.
77. **Джордж Ф.** Мозг как вычислительная машина. — М.: Изд-во иностр. лит., 1963. — 528 с.
78. **Дискуссия** о нейрокомпьютерах / Под ред. В.И. Крюкова. — Пущино, 1988.
79. **Дорофеюк А.А.** Алгоритмы автоматической классификации: Обзор // Автоматика и телемеханика. — 1971. — № 12. — С. 78—113.
80. **Доррер М.Г., Горбань А.Н., Копытов А.Г., Зенкин В.И.** Психологическая интуиция нейронных сетей // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября, 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 114—127.
81. **Дуда Р., Харт П.** Распознавание образов и анализ сцен. — М.: Мир, 1976. — 512 с.
82. **Дунин-Барковский В.Л.** Нейронные схемы ассоциативной памяти // Моделирование возбудимых структур. — Пущино: ЦБИ, 1975. — С. 90—141.
83. **Дунин-Барковский В.Л.** Информационные процессы в нейронных структурах. — М.: Наука, 1978.
84. **Дюк В.А.** Компьютерная психодиагностика. — СПб.: Братство, 1994. — 364 с.
85. **Ермаков С.В., Мышов К.Д., Охонин В.А.** К вопросу о математическом моделировании базового принципа мышления человека. — Красноярск, 1992. — 36 с. — (Препр. / ИБФ СО РАН; № 173Б.)
86. **Журавлев Ю.И.** Об алгебраическом подходе к решению задач распознавания и классификации // Проблемы кибернетики. — М.: Наука, 1978. — Вып. 33. — С. 5—68.

87. **Загоруйко Н.Г.** Методы распознавания и их применение. — М.: Сов. радио, 1972. — 206 с.
88. **Загоруйко Н.Г., Елкина В.Н., Лбов Г.С.** Алгоритм обнаружения эмпирических закономерностей. — Новосибирск: Наука. Сиб. отд-ние, 1985. — 110 с.
89. **Захарова Л.Б., Полонская М.Г., Савченко А.А. и др.** Оценка антропологического напряжения пришлого населения промышленной зоны Заполярья (биологический аспект). — Красноярск, 1989. — 52 с. — (Препр. / ИБФ СО РАН; № 110Б.)
90. **Захарова Л.М., Киселева Н.Е., Мучник И.Б. и др.** Анализ развития гипертонической болезни по эмпирическим данным // Автоматика и телемеханика. — 1977. — № 9. — С. 114—122.
91. **Ивахненко А.Г.** Самообучающиеся системы распознавания и автоматического регулирования. — Киев: Техника, 1969. — 392 с.
92. **Ивахненко А.Г.** Персептроны. — Киев: Наук. думка, 1974.
93. **Искусственный интеллект:** В 3 кн. Кн. 1. Системы общения и экспертные системы: Справочник / Под ред. Э.В. Попова. — М.: Радио и связь, 1990. — 464 с.
94. **Искусственный интеллект:** В 3 кн. Кн. 2. Модели и методы: Справочник / Под ред. Д.А. Поспелова. — М.: Радио и связь, 1990. — 304 с.
95. **Итоги науки и техники.** Сер. "Физ. и матем. модели нейронных сетей" / Под ред. А.А. Веденова. — М.: Изд-во ВИНИТИ, 1990—1992. — Т. 1—5.
96. **Кендалл М., Стьюарт А.** Статистические выводы и связи. — М.: Наука, 1973. — 900 с.
97. **Костюк Ф.Ф.** Инфаркт миокарда. — Красноярск: Офсет, 1993. — 224 с.
98. **Кохонен Т.** Ассоциативная память. — М.: Мир, 1980.
99. **Кохонен Т.** Ассоциативные запоминающие устройства. — М.: Мир, 1982.
100. **Коченов Д.А., Миркес Е.М.** Синтез управляющих воздействий // Тез. докл. III Всероссийского семинара "Нейроинформатика и ее приложения". — Красноярск: изд. КГТУ, 1995. — С. 31.
101. **Коченов Д.А., Миркес Е.М.** Определение чувствительности нейросети к изменению входных сигналов // Там же. — С. 61.
102. **Коченов Д.А., Миркес Е.М., Россиев Д.А.** Автоматическая подстройка входных данных для получения требуемого ответа нейросети // Проблемы информатизации региона: Материалы межрегиональной конф. — Красноярск, 1995.
103. **Коченов Д.А., Миркес Е.М., Россиев Д.А.** Метод подстройки параметров примера для получения требуемого ответа нейросети // Нейроинформатика и ее приложения: Тез. докл. Всероссийского рабочего семинара, 7 — 10 октября 1994 г. — Красноярск, 1994. — С. 39.
104. **Коченов Д.А., Россиев Д.А.** Аппроксимация функций класса $C[a, b]$ нейросетевыми предикторами // Тез. докл. рабочего семинара "Нейроинформатика и нейрокомпьютеры", Красноярск, 8—11 октября 1993 г. — Красноярск, 1993. — С. 13.
105. **Крайзмер Л.П., Матюхин С.А., Майоркин С.Г.** Память кибернетических систем: Основы мнемологии. — М.: Сов. радио, 1971.
106. **Куссуль Э.М., Байдык Т.Н.** Разработка архитектуры нейроподобной сети для распознавания формы объектов на изображении // Автоматика. — 1900. — № 5. — С. 56—61.
107. **Кушаковский М.С.** Аритмии сердца. — СПб.: Гиппократ, 1992. — 544 с.
108. **Лбов Г.С.** Методы обработки разнотипных экспериментальных данных. — Новосибирск: Наука. Сиб. отд-ние, 1981. — 157 с.
109. **Лоули Д., Максвелл А.** Факторный анализ как статистический метод. — М.: Мир, 1967. — 144 с.
110. **Мазуров В.Д.** Метод комитетов в задачах оптимизации и классификации. — М.: Наука. Гл. ред. физ.-мат. лит., 1990. — 248 с.

111. **МакКаллок У.С., Питтс В.** Логическое исчисление идей, относящихся к нервной активности // Нейрокомпьютер. — 1992. — № 3, 4. — С. 40—53.
112. **Минский М., Пайперт С.** Персептроны. — М.: Мир, 1971.
113. **Миркес Е.М.** Обучение сетей с пороговыми нейронами // Тез. докл. III Всероссийского семинара "Нейроинформатика и ее приложения". — Красноярск: изд. КГТУ, 1995. — С. 72.
114. **Миркес Е.М.** Оценки и интерпретаторы ответа для сетей двойственного функционирования // Там же. — С. 73—74.
115. **Миркес Е.М.** Использование весов примеров при обучении нейронных сетей // Там же. — С. 75.
116. **Миркес Е.М.** Глобальные и локальные оценки для сетей двойственного функционирования // Там же. — С. 76—77.
117. **Миркес Е.М., Свитин А.П.** Применение метода ассоциативных сетей для прогнозирования переносов заряда при адсорбции молекул // Эволюционное моделирование и кинетика. — Новосибирск: Наука. Сиб. отд-ние, 1992. — С. 30—35.
118. **Миркес Е.М., Свитин А.П., Фет А.И.** Массовые формулы для атомов // Математическое моделирование в химии и биологии: Новые подходы. — Новосибирск: Наука. Сиб. отд-ние, 1992. — С. 199—204.
119. **Миркин Б.Г.** Анализ качественных признаков и структур. — М.: Статистика, 1980. — 319 с.
120. **Мкртчян С.О.** Проектирование логических устройств ЭВМ на нейронных элементах. — М.: Энергия, 1977.
121. **Мостеллер Ф., Тьюки Дж.** Анализ данных и регрессия. — М.: Финансы и статистика, 1982. — 239 с.
122. **Муллат И.Э.** Экстремальные подсистемы монотонных систем. I—III // Автоматика и телемеханика. — 1976. — № 5. — С. 130—139; 1976. — № 8. — С. 169—178; 1977. — № 1. — С. 143—152.
123. **Мучник И.Б.** Анализ структуры экспериментальных графов // Автоматика и телемеханика. — 1974. — № 9. — С. 62—80.
124. **Мызников А.В., Россиев Д.А., Лохман В.Ф.** Нейросетевая экспертная система для оптимизации лечения облитерирующего тромбангиита и прогнозирования его непосредственных исходов // Ангиология и сосудистая хирургия. — 1995. — № 2.
125. **Мызников А.В., Россиев Д.А., Лохман В.Ф.** Прогнозирование непосредственных результатов лечения облитерирующего тромбангиита с помощью нейронных сетей // Молодые ученые — практическому здравоохранению. — Красноярск, 1994. — С. 42.
126. **Назаров Б.В.** Прогностические аспекты некоторых нарушений ритма и проводимости при остром инфаркте миокарда: Автореф. дис. ... канд. мед. наук. — Новосибирск, 1982. — 22 с.
127. **Народов А.А., Россиев Д.А., Захматов И.Г.** Оценка компенсаторных возможностей головного мозга при его органических поражениях с помощью искусственных нейронных сетей // Молодые ученые — практическому здравоохранению. — Красноярск, 1994. — С. 30.
128. **Научное** открытие в России... // Красноярский комсомолец (газета). — Красноярск, 1992. — 11 августа, № 86.
129. **Николаев П.П.** Трихроматическая модель констант восприятия окраски объектов // Сенсорные системы. — 1990. — Т. 4, вып. 4. — С. 421—442.
130. **Николаев П.П.** Методы представления формы объектов в задаче константного зрительного восприятия // Интеллектуальные процессы и их моделирование. Пространственно-временная организация. — М.: Наука, 1991. — С. 146—173.
131. **Нильсен Н.** Обучающиеся машины. — М.: Мир, 1967.

132. **Охонин В.А.** Вариационный принцип в теории адаптивных сетей. — Красноярск, 1987. — 18 с. — (Препр. / ИФ СО АН СССР; № 61Б.)
133. **Петров А.П.** Аксиоматика игры "в прятки" и генезис зрительного пространства // Интеллектуальные процессы и их моделирование. Пространственно-временная организация. — М.: Наука, 1991. — С. 174—182.
134. **Позин И.В.** Моделирование нейронных структур. — М.: Наука, 1970.
135. **Пшеничный Б.Н., Данилин Ю.М.** Численные методы в экстремальных задачах. — М.: Наука, 1975. — 319 с.
136. **Распознавание** образов и медицинская диагностика / Под ред. Ю.М. Неймарка. — М.: Наука, 1972. — 328 с.
137. **Розенблатт Ф.** Принципы нейродинамики. Перцептрон и теория механизмов мозга. — М.: Мир, 1965. — 480 с.
138. **Россиев Д.А.** Нейросетевые самообучающиеся экспертные системы в медицине // Молодые ученые — практическому здравоохранению. — Красноярск, 1994. — С. 17.
139. **Россиев Д.А., Бутакова Е.В.** Нейросетевая диагностика и дифференциальная диагностика злокачественных опухолей сосудистой оболочки глаза // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 167—194.
140. **Россиев Д.А., Бутакова Е.В.** Ранняя диагностика злокачественных опухолей сосудистой оболочки глаза с использованием нейронных сетей // Нейроинформатика и ее приложения: Тез. докл. II Всероссийского рабочего семинара, 7 — 10 октября 1994 г. — Красноярск, 1994. — С. 44.
141. **Россиев Д.А., Винник Н.Г.** Предсказание "удачности" предстоящего брака нейросетевыми экспертами // Там же. — С. 45.
142. **Россиев Д.А., Гилев С.Е., Коченов Д.А.** MultiNeuron, Версии 2.0 и 3.0 // Тез. докл. III Всероссийского семинара "Нейроинформатика и ее приложения". — Красноярск: изд. КГТУ, 1995. — С. 14.
143. **Россиев Д.А., Головенкин С.Е.** Прогнозирование осложнений инфаркта миокарда с помощью нейронных сетей // Нейроинформатика и ее приложения: Тез. докл. II Всероссийского рабочего семинара, 7 — 10 октября 1994 г. — Красноярск, 1994. — С. 40.
144. **Россиев Д.А., Головенкин С.Е., Назаров Б.В. и др.** Определение информативности медицинских параметров с помощью нейронной сети // Диагностика, информатика и метрология-94: Тез. науч.-техн. конф., Санкт-Петербург, 28—30 июня 1994 г. — СПб., 1994. — С. 348.
145. **Россиев Д.А., Головенкин С.Е., Шульман В.А., Матюшин Г.В.** Использование нейронных сетей для прогнозирования возникновения или усугубления застойной сердечной недостаточности у больных с нарушениями ритма сердца // Тез. докл. рабочего семинара "Нейроинформатика и нейрокомпьютеры", Красноярск, 8—11 октября 1993 г. — Красноярск, 1993. — С. 16.
146. **Россиев Д.А., Головенкин С.Е., Шульман В.А., Матюшин Г.В.** Прогнозирование осложнений инфаркта миокарда нейронными сетями // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 128—166.
147. **Россиев Д.А., Догадин С.А., Масленникова Е.В. и др.** Выявление накопленной дозы радиоактивного облучения с помощью нейросетевого классификатора // Современные проблемы и методологические подходы к изучению влияния факторов производственной и окружающей среды на здоровье человека: Тез. докл. республиканской конф. — Ангарск, Иркутск: Изд-во "Лисна", 1993. — С. 111—112.
148. **Россиев Д.А., Догадин С.А., Масленникова Е.В. и др.** Обучение нейросетей выявлению накопленной дозы радиоактивного облучения // Нейроинформатика и ней-

рокомпьютеры: Тез. докл. рабочего семинара. Красноярск, 8—11 октября 1993 г. — Красноярск, 1993. — С. 15.
149. **Россиев Д.А., Захматов И.Г.** Оценка компенсаторных возможностей головного мозга при его органических поражениях: Опыт применения нейросетевого векторного предиктора // Нейроинформатика и ее приложения: Тез. докл. II Всероссийского рабочего семинара, 7 — 10 октября 1994 г. — Красноярск, 1994. — С. 42.
150. **Россиев Д.А., Коченов Д.А.** Пакет программ "MultiNeuron" — "Configurator" — "Tester" для конструирования нейросетевых приложений // Там же. — С. 30.
151. **Россиев Д.А., Мызников А.В.** Прогнозирование непосредственных результатов лечения облитерирующего тромбангиита с помощью нейронных сетей // Там же. — С. 41.
152. **Россиев Д.А., Мызников А.В.** Нейросетевое моделирование лечения и прогнозирование его непосредственных результатов у больных облитерирующим тромбангиитом // Нейроинформатика и ее приложения: Материалы III Всероссийского семинара, 6—8 октября 1995 г. Ч. 1 / Под ред. А.Н. Горбаня. — Красноярск: изд. КГТУ, 1995. — С. 194—228.
153. **Россиев Д.А., Савченко А.А., Гилев С.Е., Коченов Д.А.** Применение нейросетей для изучения и диагностики иммунодефицитных состояний // Нейроинформатика и нейрокомпьютеры: Тез. докл. рабочего семинара 8—11 октября 1993 г. — Красноярск: Ин-т биофизики СО РАН, 1993. — С. 32.
154. **Россиев Д.А., Суханова Н.В., Швецкий А.Г.** Нейросетевая система дифференциальной диагностики заболеваний, проявляющихся синдромом "острого живота" // Нейроинформатика и ее приложения: Тез. докл. II Всероссийского рабочего семинара 7 — 10 октября 1994 г. — Красноярск, 1994. — С. 43.
155. **Руда М.Я., Зыско А.П.** Инфаркт миокарда. — М.: Медицина, 1981. — 288 с.
156. **Савченко А.А., Догадин С.А., Ткачев А.В. и др.** Обследование людей в районе возможного радиоактивного загрязнения с помощью нейросетевого классификатора // Нейроинформатика и ее приложения: Тез. докл. II Всероссийского рабочего семинара 7 — 10 октября 1994 г. — Красноярск, 1994. — С. 46.
157. **Савченко А.А., Митрошина Л.В., Россиев Д.А., Догадин С.А.** Моделирование с помощью нейросетевого предиктора реальных чисел иммуноэндокринного взаимодействия при заболеваниях щитовидной железы // Тез. докл. рабочего семинара "Нейроинформатика и нейрокомпьютеры", Красноярск, 8—11 октября 1993 г. — Красноярск, 1993. — С. 18.
158. **Савченко А.А., Россиев Д.А., Захарова Л.Б.** Применение нейросетевого классификатора для изучения и диагностики вилюйского энцефалита // Там же. — С. 17.
159. **Савченко А.А., Россиев Д.А., Ноздрачев К.Г., Догадин С.А.** Подтверждение с помощью нейросетевого классификатора существования гомеостатических уровней в группе практически здоровых людей // Нейроинформатика и ее приложения: Тез. докл. II Всероссийского рабочего семинара, 7 — 10 октября 1994 г. — Красноярск, 1994. — С. 49.
160. **Савченко А.А., Россиев Д.А., Ноздрачев К.Г., Догадин С.А.** Обучение нейросетевого классификатора дифференцировать пол человека по метаболическим и гормональным показателям // Там же. — С. 47.
161. **Савченко А.А., Смирнова С.В., Россиев Д.А.** Применение нейросетевого классификатора для изучения и диагностики аллергических и псевдоаллергических реакций // Там же. — С. 48.
162. **Смирнова Е.В.** Изменение корреляций между физиологическими параметрами при адаптации: Дис. ... канд. физ.-мат. наук. — Красноярск, 1989. — 128 с.
163. **Соколов Е.Н., Вайткявичус Г.Г.** Нейроинтеллект: от нейрона к нейрокомпьютеру. — М.: Наука, 1989. — 238 с.

164. **Становление** химии как науки. Всеобщая история химии / Под ред. Ю.И. Соловьева. — М.: Наука, 1983. — 464 с.
165. **Степанян А.А., Архангельский С.В.** Построение логических схем на пороговых элементах. — Куйбышев: Куйбышев. кн. изд-во, 1967.
166. **Сударикov В.А.** Исследование адаптивных нейросетевых алгоритмов решения задач линейной алгебры // Нейрокомпьютер. — 1992. — № 3, 4. — С. 13—20.
167. **Тарасов К.Е., Великов В.К., Фролова А.И.** Логика и семиотика диагноза: Методологические проблемы. — М.: Медицина, 1989. — 272 с.
168. **Тепляков А.Т., Гаргапеева А.А.** Ишемия и инфаркт миокарда. — Томск: Изд-во Томск. ун-та, 1994. — 406 с.
169. **Томов Л., Томов Ил.** Нарушения ритма сердца: Пер. с болг. — София: Медицина и физкультура, 1976. — 276 с.
170. **Транспьютерные и нейронные ЭВМ** / Под ред. В.К. Левина и А.И. Галушкина. — М.: Российский Дом знаний, 1992.
171. **Уидроу Б., Стирнз С.** Адаптивная обработка сигналов. — М.: Мир, 1989. — 440 с.
172. **Уоссермен Ф.** Нейрокомпьютерная техника. — М.: Мир, 1992.
173. **Федотов Н.Г.** Методы стохастической геометрии в распознавании образов. — М.: Радио и связь, 1990. — 144 с.
174. **Фор А.** Восприятие и распознавание образов. — М.: Машиностроение, 1989. — 272 с.
175. **Фролов А.А., Муравьев И.П.** Нейронные модели ассоциативной памяти. — М.: Наука, 1987. — 160 с.
176. **Фролов А.А., Муравьев И.П.** Информационные характеристики нейронных сетей. — М.: Наука, 1988.
177. **Ф[illegible] К.** Структурные методы в распознавании образов. — М.: Мир, 1977. — 320 с.
178. **Фукунга К.** Введение в статистическую теорию распознавания образов. — М.: Наука, 1979. — 367 с.
179. **Харман Г.** Современный факторный анализ. — М.: Статистика, 1972. — 486 с.
180. **Хинтон Дж.Е.** Обучение в параллельных сетях // Реальность и прогнозы искусственного интеллекта. — М.: Мир, 1987. — С. 124—136.
181. **Химмельблау Д.** Прикладное нелинейное программирование. — М.: Мир, 1975. — 534 с.
182. **Цыганков В.Д.** Нейрокомпьютер и его применение. — М.: Сол Систем, 1993.
183. **Цыпкин Я.З.** Основы теории обучающихся систем. — М.: Наука, 1970. — 252 с.
184. **Шенк Р., Хантер Л.** Познать механизмы мышления // Реальность и прогнозы искусственного интеллекта. — М.: Мир, 1987. — С. 15—26.
185. **Шульман В.А.** Осложнения инфаркта миокарда: Методические указания для самостоятельной работы студентов. — Красноярск: изд. КГМИ, 1990. — 22 с.
186. **Шайдуров В.В.** Многосеточные методы конечных элементов. — М.: Наука, 1989.
187. **Шевченко Н.М., Олишевко С.В., Рыжов В.М.** Основы клинической кардиологии для врачей-терапевтов: Профессиональный минимум. — Калининград, 1992. — 145 с.
188. **Щербаков П.С.** Библиографическая база данных по методам настройки нейронных сетей // Нейрокомпьютер. — 1993. — № 3, 4. — С. 5—8.
189. **The Adaptive** Brain / S. Grossberg (Ed.). — North-Holland, 1987. — Vol. 1. Cognition, Learning, Reforcement, and Rhythm. — 498 p.; vol. 2. Vision, Speech, Language, and Motor Control. — 514 p.
190. **Alexander S. Th.** Adaptive Signal Processing: Theory and Applications. — Springer Verlag, 1986. — 179 p.
191. **Alexander I., Morton H.** The logic of neural cognition // Adv. Neural Comput. — Amsterdam e.a., 1990. — P. 97—102.

192. **Amari Sh., Maginu K.** Statistical Neurodynamics of Associative Memory // Neural Networks. — 1988. — Vol. 1, № 1. — P. 63—74.
193. **Arbib M.A.** Brains, Machines, and Mathematics. — Springer Verlag, 1987. — 202 p.
194. **Baba N.** New Topics in Learning Automate Theory and Applications. — Springer Verlag, 1985. — 131 p. — (Lec. Not. Control and Information; № 71.)
195. **Bartsev S.I., Okhonin V.A.** The algorithm of dual functioning (back-propagation): general approach, versions and applications. — Krasnoyarsk, 1989. — 16 p. — (Prepr. / Biophysics Institute SB AS USSR; № 107B.)
196. **Bartsev S.I., Okhonin V.A.** Variation principle and algorithm of dual functioning: examples and applications // Proc. Intern. Workshop "Neurocomputers and attention II". — Manchester Univ. Press, 1991. — P. 445—452.
197. **Bartsev S.I., Okhonin V.A.** Self-learning neural networks playing "Two coins" // Ibid. — P. 453—458.
198. **Bartsev S.I., Okhonin V.A.** Optimization and Monitoring Needs: Possible Mechanisms of Control of Ecological Systems // Nanobiology. — 1993. — Vol. 2. — P. 165—172.
199. **Bartsev S.I., Okhonin V.A.** Variation principle and algorithm of dual functioning: examples and applications // Proc. Intern. Workshop "Neurocomputers and attention II". — Manchester Univ. Press, 1991. — P. 445—452.
200. **Borisov A.G., Gilev S.E., Golovenkin S.E. et al.** "MultiNeuron" neural simulator and its medical applications // Modelling, Measurement & Control C. — 1996. — Vol. 55, № 1. — P. 1—5.
201. **Bruck J., Goodman J.** On the power of neural networks for solving hard problems // J. Complex. — 1990. — Vol. 6, № 2. — P. 129—135.
202. **Carpenter G.A., Grossberg S.** A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine // Computer Vision, Graphics, and Image Processing. — 1987. — Vol. 37. — P. 54—115.
203. The Computer and the Brain. Perspectives of Human and Artificial Intelligence / Ed. by J.R. Brinc, C.R. Haden, C. Burava. — North-Holland, 1989. — 300 p.
204. Connectionism in Perspective / Ed. by R. Pfeifer, Z. Schreter, F. Fogelman-Soulie, and L. Steels. — North-Holland, 1989. — 518 p.
205. **Cybenko G.** Approximation by superposition of a sigmoidal function // Mathematics of Control, Signals, and Systems. — 1989. — Vol. 2. — P. 303—314.
206. **Diday E., Simon J.C.** Clustering analysis (dans Digital Pattern Recognition), Redacteur: K.S.F.U. — Berlin: Springer Verlag, 1980. — P. 47—93.
207. Disordered Systems and biological Organization / Ed. by Bienenstock F., Fogelman-Soulie G. — Weisbuch. Springer, 1986. — 405 p.
208. **Dorrer M.G., Gorban A.N., Kopytov A.G., Zenkin V.I.** Psychological intuition of neural networks // Proc. WCNN'95 (World Congress on Neural Networks'95). — Washington, DC, July, 1995. — P. 193—196.
209. **Dorrer M.G., Gorban A.N., Zenkin V.I.** Neural networks in psychology: classical explicit diagnoses // Neuroinformatics and Neurocomputers: Proc. Second RNNS-IEEE Symp. — Rostov-na-Donu, September, 1995. — P. 281—284.
210. **Draper N.R.** Applied regression analysis bibliography update 1988/89 // Commun. Statist. Theory and Meth. — 1990. — Vol. 19, № 4. — P. 1205—1229.
211. **Ferretti C., Mauri G.** NNET: some tools for neural Networks simulation // 9th Ann. Int. Phoenix Conf. Comput. and Commun., Scottsdate, Ariz., March 21—23 1990. — Los Alamitos, Calif., e.a., 1990. — P. 38—43.
212. **Filho E.C.D.B.C., Fairhurst M.C., Bisset D.L.** Adaptive pattern recognition using goal seeking neurons // Pattern Recogn. Lett. — 1991. — Vol. 12, № 3. — P. 131—138.
213. **Forbes A.B., Mansfield A.J.** Neural implementation of a method for solving systems of linear algebraic equations // Nat. Phys. Lab. Div. Inf. Technol. and Comput. Rept. — 1989. — № 155. — P. 1—14.

214. **Fukushima K.** Neocognitron: A self-organizing Neural Network model for a Mechanism of Pattern Recognition uneffected by shift in position // Biol. Cybernetics. — 1980. — Vol. 36, № 4. — P. 193—202.
215. **Fulcher J.** Neural networks: promise for the future? // Future Generat. Comput. Syst. — 1990—1991. — Vol. 6, № 4. — P. 351—354.
216. **Gallant A.R., White H.** There exists a neural network that does not make avoidable mistakes // IEEE Second Intern. Conf. on Neural Networks, San Diego, CA. — N.Y.: IEEE Press, 1988. — Vol. 1. — P. 657—664.
217. **Gecseg F.** Products of Automata. — Springer Verlag, 1986. — 107 p.
218. **Gemignani M.C.** Liability for malfunction of an expert system // Proc. IEEE Conf. Manag. Expert Syst. Program and Proj., Bethesda, Md., Sept. 10—12, 1990. — Los Alamitos, Calif., e.a., 1990. — P. 8—15.
219. **Genis C.T.** Relaxation and neural learning: points of convergence and divergence // J. Parallel and Distrib. Comput. — 1989. — Vol. 6, № 2. — P. 217—244.
220. **George N., Wang Hen-Ge, Venable D.L.** Pattern recognition using the ring-wedge detector and neural-network software: [Pap.] Opt. Pattern Recogn. II: Proc. Meet., Paris, 26—27 Apr., 1989 // Proc. Soc. Photo-Opt. Instrum. Eng. — 1989. — P. 96—106.
221. **Gileva L.V., Gilev S.E.** Neural Networks for Binary Classification // AMSE Trans. Scientific Siberian, A. — 1993. — Vol. 6: Neurocomputing. — P. 135—167.
222. **Gilev S.E., Gorban A.N., Kochenov D.A. et al.** "MultiNeuron" neural simulator and its medical applications // Proc. Intern. Conf. on Neural Information Processing, Oct. 17—20, 1994, Seoul, Korea. — 1994. — Vol. 2. — P. 1261—1264.
223. **Gilev S.E., Gorban A.N., Mirkes E.M.** Several Methods for Accelerating the Training Process of Neural Networks in Pattern Recognition // Adv. Modelling & Analysis, A. AMSE Press. — 1992. — Vol. 12, № 4. — P. 29—53.
224. **Gilev S.E., Gorban A.N., Mirkes E.M.** Small Experts and Internal Conflicts in Learnable Neural Networks // Adv. Modelling & Analysis. AMSE Press. — 1992. — Vol. 24, № 1. — P. 45—50.
225. **Gilev S.E., Gorban A.N., Mirkes E.M.** Internal Conflicts in Neural Networks // Trans. IEEE-RNNS Symposium, Rostov-on-Don, September, 1992. — 1992. — Vol. 1. — P. 219—226.
226. **Gluck M.A., Parker D.B., Reifsnider E.S.** Some Biological Implications of a Differential-Hebbian Learning Rule // Psychobiology. — 1988. — Vol. 16, № 3. — P. 298—302.
227. **Gorban A.N.** Neurocomputing in Siberia // Adv. Modelling & Analysis. AMSE Press. — 1992. — Vol. 34 (2). — P. 21—28.
228. **Gorban A.N.** Systems with inheritance and effects of selection // Scientific Siberian A. AMSE Press. — 1992. — Vol. 1. Ecology. — P. 82—126.
229. **Gorban A.N., Rossiev D.A., Butakova E.V. et al.** Medical psychological and physiological applications of MultiNeuron neural simulator // Neuroinformatics and Neurocomputers: Proc. the Second RNNS-IEEE Symp. — Rostov-na-Donu, September, 1995. — 1995. — P. 7—14.
230. **Gorban A.N., Rossiev D.A., Gilev S.E. et al.** Medical and physiological applications of MultiNeuron neural simulator // Proc. WCNN'95. (World Congress on Neural Networks'95). — Washington, DC, July, 1995. — P. 170—175.
231. **Gorban A.N., Waxman C.** How many neurons are sufficient to elect the U.S.A. President? // AMSE Trans., Scientific Siberian, A. — 1993. — Vol. 6. Neurocomputing. — P. 168—188.
232. **Gorban A.N., Waxman C.** How many neurons are sufficient to elect the U.S.A. President? Two! (Siberian neurocomputer forecast results of U.S.A. Presidential elections). — Krasnoyarsk: Institute of Biophysics. SB RAS, 1992. — 29 p. — (Prepr. / SB RAS, Institute of Biophysics; № 191Б.)

233. **Gorban A.N., Waxman C.** Neural networks for political forecast // Proc. WCNN'95 (World Congress on Neural Networks'95). — Washington, DC, July, 1995. — P. 179—184.
234. **Gordienko P.** Construction of efficient neural networks: Algorithms and tests // Proc. Intern. Joint Conf. on Neural Networks IJCNN'93. — Nagoya, Japan, 1993. — P. 313 — 316.
235. **Gordienko P.** Construction of efficient neural networks // Proc. Conf. on Neural Information Processing (Oct. 17—20, 1994, Seoul, Korea). — 1994. — Vol. 1. — P. 366—371.
236. **Gordienko P.** How to obtain a maximum of skills with minimum numbers of connections // AMSE Trans., Scientific Siberian, A. — 1993. — Vol. 6: Neurocomputing. — P. 204—208.
237. **Grossberg S.** Nonlinear Neural Networks: Principles, Mechanism and Architectures // Neural Networks. — 1988. — Vol. 1, № 1. — P. 17—62.
238. **Hecht-Nielsen R.** Neurocomputing: Picking the Human Brain // IEEE Spectrum. — 1988, March. — P. 36—41.
239. **Hecht-Nielsen R.** Theory of the backpropagation neural network // Neural Networks for Human and Machine Perception / Ed. by H. Wechsler. — Boston, MA: Acad. Press, 1992. — Vol. 2. — P. 65—93.
240. **Hod H., Lew A.S., Keltai M. et al.** Early atrial fibrillation during evolving myocardial infarction: a consequence of impaired left atrial perfustion // Circulation. — 1987. — Vol. 75, № 1. — P. 146—150.
241. **Hopfield J.J.** Neural networks and physical systems with emergent collective computational abilities // Proc. Nat. Sci. USA. — 1982. — Vol. 79. — P. 2554—2558.
242. **Hornik K., Stinchcombe M., White H.** Multilayer Feedforward Networks are Universal Approximators // Neural Networks. — 1989. — Vol. 2. — P. 359—366.
243. **Jeffries C.** Code recognition with neural network dynamical systems // SIAM Rev. — 1990. — Vol. 32, № 4. — P. 636—651.
244. **Kalman R.E.** A theory for the identification of linear relations // Frontiers Pure and Appl. Math.: Collect. Pap. Dedicat. Jacques-Louis Occas. His 60th Birthday: Sci. Meet., Paris, 6—10 June, 1988. — Amsterdam e.a., 1991. — P. 117—132.
245. **Keller J.M., Yager R.R., Tahani H.** Neural network implementation of fuzzy logic // Fuzzy Sets and Syst. — 1992. — Vol. 45, № 1. — P. 1—12.
246. **Kochenov D.A., Rossiev D.A.** Approximations of functions of C [A, B] class by neural-net predictors (architectures and results) // AMSE Trans., Scientific Siberian, A. — 1993. — Vol. 6. Neurocomputing. — P. 189—203.
247. **Koopmans T.** Serial correlation and quadratic forms in normal variables // Ann. Math. Statist. — 1942. — Vol. 13. — P. 14—33.
248. **Kosko B.** Bidirectional Associative Memories // IEEE Trans. on Systems, Man. and Cybernetics. — Jan., 1988. — Vol. SMC-18. — P. 49—60.
249. **Lichtman A.J., Keilis-Borok V.I.** Pattern Recognition as Applied to Presidential Elections in U.S.A., 1860—1980; Role of Integral Social, Economic and Political Traits, Contribution № 3760. — California Institute of Technology, Division of Geological and Planetary Sciences, 1981.
250. **Macukow B.** Robot control with neural networks // Artif. Intell. and Inf.-Contr. Syst. Rob.-89: Proc. 5th Intern. Conf., Strbske Pleso, 6—10 Nov., 1989. — Amsterdam e.a., 1989. — P. 373—376.
251. **Mirkes E.M., Svitin A.P.** The usage of adaptive neural networks for catalytic activity predictions // CHISA — 10th Intern. Congr. of Chem. Eng., Chem. Equipment Design and Automation. — Praha, 1990. — 7 p. — (Prepr. / B3. 80 [1418].)
252. **Modha D.S., Hecht-Nielsen R.** Multilayer Functionals // Mathematical Approaches to Neural Networks / Ed. by J.G. Taylor. — Elsevier, 1993. — P. 235—260.

253. **Narendra K.S., Amnasway A.M.** A Stable Adaptive System. — Prentice-Hall, 1988. — 350 p.
254. **Neural** Computers / Ed. by R. Eckmiller, Ch. Malsburg. — Springer Verlag, 1989. — 556 p.
255. **Pedrycz W.** Neurocomputations in relational systems // IEEE Trans. Pattern Anal. and Mach. Intell. — 1991. — Vol. 13, № 3. — P. 289—297.
256. **Pham D.T., Liu X.** Statespace identification of dynamic systems using neural networks // Eng. Appl. Artif. Intell. — 1990. — Vol. 3 , № 3. — P. 198—203.
257. **Pineda F.J.** Recurrent backpropagation and the dynamical approach to adaptive neural computation // Neural Comput. — 1989. — Vol. 1. — P. 161—172.
258. **Real** Brains, Artificial Minds / Ed. by J.L. Casti, A. Karlqvist. — North-Holland, 1987. — 226 p.
259. **Rossiev D.A., Golovenkin S.E., Shulman V.A., Matyushin G.V.** The employment of neural networks to model implantation of pacemaker in patients with arrhythmias and heart blocks // Modelling, Measurement & Control, C. — 1995. — Vol. 48, № 2. — P. 39—46.
260. **Rossiev D.A., Golovenkin S.E., Shulman V.A., Matyushin G.V.** The employment of neural networks to model implantation of pacemaker in patients with arrhythmias and heart blocks // Proc. Intern. Conf. on Neural Information Processing, Oct. 17—20, 1994, Seoul, Korea. — 1994. — Vol. 1. — P. 537—542.
261. **Rossiev D.A., Golovenkin S.E., Shulman V.A., Matyushin G.V.** Neural networks for forecasting of myocardial infarction complications // Proc. the Second IEEE — RNNS Intern. Symp. on Neuroinformatics and Neurocomputers, Rostov-on-Don, September 20—23, 1995. — 1995. — P. 292—298.
262. **Rossiev D.A., Golovenkin S.E., Shulman V.A., Matyushin G.V.** Forecasting of myocardial infarction complications with the help of neural networks // Proc. WCNN'95. (World Congress on Neural Networks'95). — Washington, DC, July 1995.
263. **Rossiev D.A., Savchenko A.A., Borisov A.G., Kochenov D.A.** The employment of neural-network classifier for diagnostics of different phases of immunodeficiency // Modelling, Measurement & Control. C. — 1994. — Vol. 42, № 2. — P. 55—63.
264. **Rumelhart D.E., Hinton G.E., Williams R.J.** Learning internal representations by error propagation // Parallel Distributed Processing: Exploration in the Microstructure of Cognition / Ed. by D.E. Rumelhart and J.L. McClelland. — Cambridge, MA: MIT Press, 1986. — Vol. 1. — P. 318—362.
265. **Rummelhart D.E., Hinton G.E., Williams R.J.** Learning representations by back-propagating errors // Nature. — 1986. — Vol. 323. — P. 533—536.
266. **Saaf L.A., Morris G.M.** Filter synthesis using neural networks: [Pap.] Opt. Pattern Recogn. II: Proc. Meet., Paris, 26—27 Apr., 1989 // Proc. Soc. Photo-Opt. Instrum. Eng. — 1989. — Vol. 1134. — P. 12—16.
267. **Sandberg I.W.** Approximation for Nonlinear Functionals // IEEE Trans. on Circuits and Systems. 1: Fundamental Theory and Applications, Jan. — 1992. — Vol. 39, № 1. — P. 65—67.
268. **Savchenko A.A., Zakharova L.B., Rossiev D.A.** The employment of neural networks for investigation & diagnostics of Viliuisk encephalomyelitis // Modelling, Measurement & Control, C. — 1995. — Vol. 48, № 4. — P. 1—15.
269. **Sussman H.J.** Uniqueness of the weights for minimal feedforward nets with a given input-output map // Neural Networks. — 1992. — № 5. — P. 589—593.
270. **Tabatabai A., Troudet T.P.** A neural net based architecture for the segmentation of mixed gray-level and binary pictures // IEEE Trans. Circuits and Syst. — 1991. — Vol. 38, № 1. — P. 66—77.

271. **Tao K.M., Morf M.** A lattice filter type of neuron model for faster nonlinear processing // 23d Asilomar Conf. Signals, Syst. and Comput., Pacific Grove, Calif., Oct. 30-Nov. 1, 1989: Conf. Rec. Vol. 1. — San Jose (Calif.), 1989. — P. 123—127.
272. **Vakhrushev S.G., Rossiev D.A., Burenkov G.I., Toropova L.A.** Neural network forecasting of optimal parameters of laserotherapy in patients after tonsillectomy // Proc. WCNN'95. (World Congress on Neural Networks'95). — Washington, DC, July 1995.
273. **Waxman C.** Neurocomputers in the human sciences. Program: predictions of US presidential elections // Modelling, Measurement & Control, D. — 1992. — Vol. 5, № 1. — P. 41—53.
274. **Weckert J.** How expert can expert systems really be? // Libr. and Expert Syst.: Proc. Conf. and Workshop [Centre Inf. Stud.], Riverina, July, 1990. — L., 1991. — P. 99—114.
275. **Wiedermann J.** On the computation efficiency of symmetric neural networks // Theor. Comput. Sci. — 1991. — Vol. 80, № 2. — P. 337—345.
276. **Wong K.Y.M., Kahn P.E., Sherrington D.** A neural network model of working memory exhibiting primacy and recency // J. Phys. A. — 1991. — Vol. 24, № 5. — P. 1119—1133.

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ 3

Часть 1

НЕЙРОИМИТАТОРЫ И ИХ ПРИЛОЖЕНИЯ

ВВЕДЕНИЕ 6

1. Область приложения нейроимитаторов . . . . . —
2. Задачи классификации 8
3. Как обучается нейросеть 9
4. Тестирование (работа обученной нейросети) 10
5. Прогнозирование результата выборов президента США 13
6. Распознавание направления сдвига . . . . . —
7. Коррекция классификационной модели 14
8. Автоматическая минимизация числа параметров 16
9. Нейросетевые предикторы 17
10. Создание нейросетевых экспертных систем 18

Глава 1. КАК МЫ "ВЫБИРАЛИ" КЛИНТОНА 19

1. Данные для обучения . . . . . —
2. Результаты тестирования 21
3. Голосование "консилиума" 23
4. Анализ значимости признаков 24

Глава 2. ПСИХОЛОГИЧЕСКАЯ ИНТУИЦИЯ НЕЙРОННЫХ СЕТЕЙ . . 26

1. Проблемы: адаптация тестов и измеренная индивидуальность . . —
2. Обработка результатов классического тестирования (*эксперимент 1*) —
3. Предсказание отношений в группе минуя явные описания (*эксперимент 2*) 28

Приложение 30

Глава 3. ОБЗОР МЕДИЦИНСКИХ И ФИЗИОЛОГИЧЕСКИХ ПРИМЕНЕНИЙ НЕЙРОИМИТАТОРА "MultiNeuron" 35

1. Применение нейросетей для изучения и диагностики иммунодефицитных состояний . . . . . —
2. Информативность иммунологических, ферментативных и гормональных показателей в модели нейросетевого классификатора "здоровые — лица с иммунодефицитом" 36
3. Диагностика и изучение аллергических и псевдоаллергических реакций . . . . . —
4. Классификация лиц с аллергиями по уровню метаболических процессов в лимфоцитах 37
5. Прогнозирование возникновения или усугубления застойной сердечной недостаточности у больных с нарушениями ритма сердца . 38
6. Прогнозирование сердечной недостаточности у больных с различными поражениями проводящей системы сердца 39
7. Прогнозирование осложнений инфаркта миокарда . . . . . —
8. Определение прогностически значимых медицинских параметров 41
9. Прогнозирование непосредственных результатов лечения облитерирующего тромбангиита 42
10. Обучение нейросетей выявлению накопленной дозы радиоактивного облучения 43
11. Нейротехнология для ранней диагностики злокачественных опухолей сосудистой оболочки глаза 44
12. Нейронные сети для определения вида лазеротерапии после тонзиллэктомии . . . . . —
13. Нейросетевая дифференциальная диагностика на основе субъективных данных 45
14. Нейронная сеть определяет пол человека по метаболическим и гормональным показателям 46
15. Нейросетевая система дифференциальной диагностики заболеваний, проявляющихся синдромом "острого живота" 47
16. Нейросетевая экспертная система прогнозирования исходов и осложнений у больных с циррозами печени . . . . . —
17. Сравнительное изучение метаболических показателей лимфоцитов у больных гепатитами А и В с помощью нейросетевого классификатора 48
18. Оценка компенсаторных возможностей головного мозга при его органических поражениях с помощью искусственных нейросетей 49

Глава 4. ПРОГНОЗИРОВАНИЕ ОСЛОЖНЕНИЙ ИНФАРКТА МИОКАРДА НЕЙРОННЫМИ СЕТЯМИ 50

1. Задача прогнозирования осложнений . . . . . —
2. Обучающая выборка и входные параметры 53
3. Прогнозирование факта и времени возникновения фибрилляции предсердий 58
4. Прогнозирование возникновения перикардита 65
5. Прогнозирование факта и времени возникновения тромбоэмболических осложнений 68

6. Прогнозирование факта и времени возникновения или усугубления сердечной недостаточности 73
7. Изучение влияния исследуемых параметров на прогноз осложнений инфаркта миокарда 77

Часть 2

НЕЙРОННЫЕ СЕТИ

ВВЕДЕНИЕ. ИДЕЯ НЕЙРОКОМПЬЮТЕРА 81

Глава 1. ЭЛЕМЕНТЫ НЕЙРОННЫХ СЕТЕЙ 84

1. Перечень элементов . . . . . —
2. Задачи для одноэлементных систем 88
3. Настройка параметров одноэлементных систем для решения задач 89
4. Значимость параметров и сигналов. Сокращение описания . . . 107
5. Сокращение описания "сверху вниз" — подбор достаточного семейства наиболее значимых параметров 114
6. Рекурсивное контрастирование и бинаризация 118

Глава 2. НЕЙРОННЫЕ СЕТИ 123

1. Основные архитектуры . . . . . —
2. Входы, выходы и функционирование 127
3. Сети с включением макронейронов — новое средство моделирования 129
4. Задачи для нейронных сетей 132
5. Настройка нейронных сетей для решения задач 144

Глава 3. ОБУЧЕНИЕ КАК ОПТИМИЗАЦИЯ 196

1. Универсальный метод построения нейронных сетей для решения задач . . . . . —
2. Двойственное функционирование сетей автоматов при вычислении градиентов сложных функций 198
3. Двойственное функционирование сетей автоматов. Обобщения . . 206
4. Смысл двойственных переменных 213
5. Несколько заключительных замечаний о двойственности 215
6. Особенности задачи оптимизации, возникающей при обучении нейронных сетей 216
7. Учет ограничений при обучении 217
8. Одномерная оптимизация 218
9. Наискорейший спуск, случайный поиск и партан-методы 221
10. Одношаговый квазиньютоновский метод и сопряженные градиенты 224
11. Экспериментальное сравнение методов обучения нейронных сетей 225
12. Использование случайных возмущений в обучении 227
13. Несколько слов о философии оптимизации 229
14. Четыре типа устойчивости 230
15. Метод виртуальных частиц 232
16. Тестирование метода виртуальных частиц 233
17. Тестирование нейронных сетей с квадратичными сумматорами . 241

18. Характеристики нейрона и функции, вычисляемые сетью. Теорема о полноте 243
19. Сравнение различных функций активации в вычислительном эксперименте 246
20. Контрастирование и логически прозрачные нейронные сети . . 247

ПРИЛОЖЕНИЕ 250

СПИСОК ЛИТЕРАТУРЫ 257

Научное издание

Горбань Александр Николаевич
Россиев Дмитрий Анатольевич

Нейронные сети на персональном компьютере

Редактор *Л.В. Филиппова*
Художественный редактор *Л.В. Матвеева*
Художник *В.И. Шумаков*
Технический редактор *Л.П. Минеева*
Корректоры *С.М. Погудина,*
Л.А. Щербакова
Оператор электронной верстки
Т.А. Клименкова

ИБ № 1847

ЛР № 020297 от 27.11.91. Сдано в набор 04.06.96. Подписано в печать 23.08.96. Бумага офсетная. Формат 60 × 90 1/16 . Гарнитура таймс. Офсетная печать. Усл. печ. л. 17,5. Уч.-изд. л. 16,2. Доп. тираж 500 экз. Заказ № 604

"Наука". Сибирская издательская фирма РАН. 630099 Новосибирск, ул. Советская, 18.

Оригинал-макет изготовлен на настольной издательской системе.

Сибирское предприятие "Наука" РАН. 630077 Новосибирск, ул. Станиславского, 25.

Качество печати соответствует оригиналам, предоставленным издательством

ГОРБАНЬ
Александр Николаевич

Родился в 1952 г. в Омске. Доктор физико-математических наук, профессор. Заведует отделом вычислительной математики Вычислительного центра СО РАН в г. Красноярске и кафедрой НейроЭВМ в Красноярском государственном техническом университете. Автор более 200 работ по различным аспектам математического моделирования в химии, физике, биологии, по теории динамических систем, математической физике и др., а также 11 монографий, включая: **Горбань А.Н., Хлебопрос Р.Г.** Демон Дарвина. Идея оптимальности и естественный отбор. М.: Наука, 1988. 208 с.; **Горбань А.Н.** Обучение нейронных сетей. М.: СССР—США СП "Параграф", 1990. 160 с.; **Yablonskii G.S., Bykov V.I., Gorban A.N., Elokhin V.I.** Kinetic models of catalytic reactions. Amsterdam: Elsevier, 1991. 400 p.

РОССИЕВ
Дмитрий Анатольевич

Родился в 1965 г. в Красноярске. По образованию врач-терапевт, специализировался в области иммунологии и иммуногенетики. В 1993 г. защитил кандидатскую диссертацию по иммунологии язвенной болезни желудка. В течение нескольких последних лет работает в области медицинской информатики и нейроинформатики, заведует лабораторией медицинской нейроинформатики Красноярской медицинской академии. Автор более 100 научных публикаций о применении нейронных сетей в области медицины и биологии. В 1996 г. защитил докторскую диссертацию, посвященную технологии создания нейросетевых экспертных систем.