Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Институт радиоэлектроники и информационных технологий — РТФ

А. П. СЕРГЕЕВ
Д. А. ТАРАСОВ

ВВЕДЕНИЕ В НЕЙРОСЕТЕВОЕ МОДЕЛИРОВАНИЕ

Учебное пособие

Министерство образования и науки Российской Федерации
Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина

А. П. Сергеев, Д. А. Тарасов

Введение в нейросетевое моделирование

Учебное пособие

Под общей редакцией канд. физ.-мат. наук, доц. А. П. Сергеева

Рекомендовано методическим советом
Уральского федерального университета
для студентов вуза, обучающихся
по инженерно-техническим направлениям подготовки
ИРИТ — РТФ

Екатеринбург
Издательство Уральского университета
2017

УДК 004.032.26:004.81(075.8)
ББК 32.818.130в6я73
С32

Рецензенты:
д-р техн. наук, проф., директор Института химической переработки растительного сырья и промышленной экологии УГЛТУ *А. В. Вураско*;
д-р физ.-мат. наук, вед. науч. сотр. Института промышленной экологии УрО РАН *Л. М. Мартюшев*

На обложке использовано изображение с сайта https://pbs.twimg.com/media/CnqTWYvVIAAXfmK.jpg

Сергеев, А. П.
С32 Введение в нейросетевое моделирование : учеб. пособие / А. П. Сергеев, Д. А. Тарасов ; под общ. ред. А. П. Сергеева. — Екатеринбург : Изд-во Урал. ун-та, 2017. — 128 с.

ISBN 978-5-7996-2124-7

Учебное пособие содержит начальные сведения о моделировании на базе искусственных нейронных сетей. Разобраны биологические принципы построения и алгоритмы создания искусственного нейрона (типа «перцептрон») и сетей на его основе.

Издание рекомендуется исследователям, преподавателям, аспирантам, студентам, а также всем, кто интересуется современным состоянием дел в области искусственных нейронных сетей и моделирования.

Библиогр.: 42 назв. Табл. 7. Рис. 33.

УДК 004.032.26:004.81(075.8)
ББК 32.818.130в6я73

ISBN 978-5-7996-2124-7

1. Биологические прототипы нейронных сетей

1.1. Нейрон

Нейроны (или нервные клетки) являются базовыми элементами нервной системы, основными обработчиками и передатчиками информации в человеческом организме. Нервная система человека состоит примерно из ста миллиардов нервных клеток. Каждый нейрон представляет собой отдельную клетку и является базовой коммуникационной единицей нервной системы (рис. 1.1). Как правило, нервный импульс продвигается от дендритов и тела клетки к аксону (нейриту) до тех пор, пока не достигнет концевых синаптических луковичек. В этом месте нервный импульс встречается с дендритами другого нейрона, и его движение продолжается.

Нейроны бывают разных типов, и каждый из них предназначен для выполнения специфической нейронной функции. Некоторые нейроны играют роль рецепторных клеток органов чувств, воспринимающих извне определенные виды энергии, например свет, давление или химическую энергию. Эти нейроны трансдуцируют, или преобразуют, поглощенную ими энергию в нервные импульсы, направляемые затем другим нейронам, являющимся элементами нервной системы. Трансдукция — это превращение физической энергии в нейронную форму стимулирования, осуществляемое специализированными органами чувств. Сенсорные нейроны передают информацию от сенсорных рецепторов мозгу, мотонейроны — от мозга мышцам, а интернейроны осуществляют обмен информацией между нейронами.

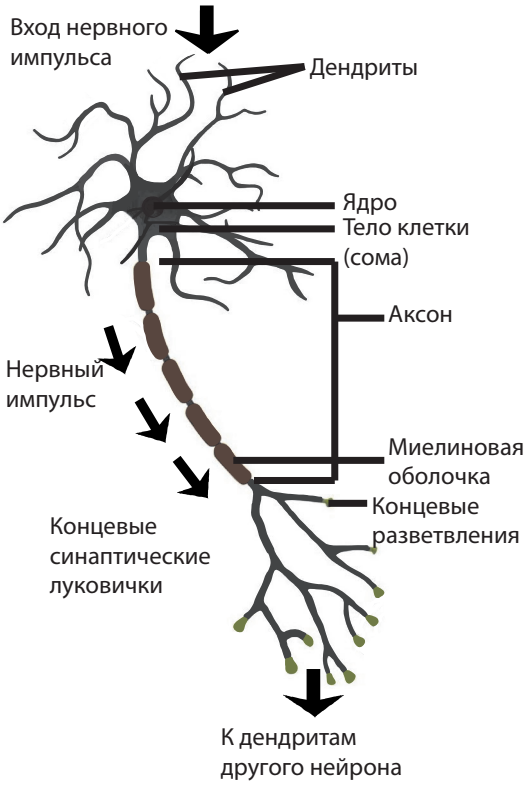


Рис. 1.1. Нейрон

Большинство нейронов независимо от их функции состоят из отдельной клетки, имеющей три отличные друг от друга части:

1) тело клетки, или сома, содержащее ядро клетки, которое регулирует химическую активность нейрона, а также принимает и сохраняет получаемую им информацию;
2) отходящие от тела клетки разветвленные структуры, называемые дендритами (от греческого слова dendron, что означает «дерево»), которые принимают информацию от других клеток и осуществляют связь с ними;
3) аксоны — тонкие длинные волокна, по которым информация поступает от сомы к другим нейронам.

Аксоны, как правило, передают информацию от нейрона. При этом они направляют ее либо соседним нейронам, либо мышце или железе, «приказывая» им совершить то или иное действие. Длина большинства аксонов не превышает 0,1 мм, однако некоторые аксоны нерв-

ной системы взрослого человека достигают метровой длины. Обычно аксоны связаны с дендритами других нервных клеток и передают им импульсы, т. е. аксон — элемент коммуникационной системы нейрона, передающий за пределы клетки информацию в виде нервного импульса, а дендриты «доставляют» информацию внутрь, в тело нервной клетки. Аксоны многих нейронов покрыты слоем миелина — белково-жирового комплекса, образованного защитными и питательными клетками и создающего изолирующую оболочку вокруг аксона. Благодаря этой оболочке скорость прохождения нервного импульса по нейрону значительно увеличивается.

Разветвленный конец аксона имеет древовидную форму, и каждая его ветвь заканчивается концевой синаптической луковичкой. Это место соединения аксона с дендритом другой нервной клетки.

1.2. Нейронная передача

Основа всех сенсорных процессов — зрения, слуха, осязания и других — передача информации. Информация в виде нервных импульсов передается по нейронам в результате сложного взаимодействия электрических и химических зарядов. Нервный импульс возникает в результате изменения концентрации катионов — положительно заряженных ионов — натрия (Na^+) и калия (K^+) внутри и снаружи нейрона. Для неактивного, или нестимулированного, нейрона характерны разные концентрации ионов внутри и снаружи, вне клеточной мембраны, причем концентрация отрицательно заряженных ионов снаружи несколько выше, чем внутри. Результатом подобного неравенства концентраций является возникновение разности потенциалов на мембране клетки. Электрический заряд внутри нервной клетки человека отличается от внешнего заряда примерно на –70 милливольт. Этот потенциал неактивного нейрона называется потенциалом покоя (или мембранным потенциалом).

Когда на нейрон воздействует раздражитель или аксон другого нейрона и внутри нейрона возникает избыточный по сравнению со средой положительный заряд, потенциал покоя изменяется за доли секунды. Результатом этого быстротечного процесса является электрический заряд, который с большой скоростью перемещается по аксону нервной клетки, после чего потенциал возвращается в исходное

состояние. Быстрое изменение электрического заряда — первая стадия возбуждения нейрона и передачи информации с помощью аксона внутри нервной системы. Подобный механизм характерен для всех сенсорных систем.

1.3. Потенциал действия

Нервные клетки не обязательно воспринимают и передают импульсы соседним нейронам всякий раз, когда они генерируют электрический заряд или когда на них воздействует раздражитель. Возникновение в нейроне потенциала действия и передача импульса возможны лишь тогда, когда достигнут определенный пороговый уровень его стимулирования. Этот минимальный уровень стимулирования, необходимый для возбуждения нейрона, называется нейронным порогом. Если внутри нервной клетки накапливается электрический заряд, превышающий нейронный порог, электрическое состояние нейрона быстро изменяется — заряд сохраняется в течение одной миллисекунды. Это изменение называется потенциалом действия, а также пиковым или спайковым потенциалом или просто спайком (от английского *spike*, что означает «острый выступ, шип»), поскольку в определенный момент электрический заряд нейрона быстро достигает пика, а затем быстро падает.

Потенциалы действия подчиняются принципу «все или ничего». Когда электрический заряд достигает нейронного порога, возникает потенциал действия и посылается импульс. С другой стороны, если общий электрический заряд падает ниже критического значения нейронного порога, потенциал действия не возникает. Иными словами, нейрон или генерирует потенциал действия, или нет, что и означает, что он функционирует по принципу «все или ничего».

Величина, или интенсивность, потенциала действия не зависит от интенсивности раздражителя, т. е. является постоянной величиной. Однако нам известно из собственного опыта, что по своей интенсивности раздражители весьма существенно отличаются друг от друга — они могут быть как весьма сильными, так и едва уловимыми. Каким же образом возникновение потенциала действия, подчиняющегося принципу «все или ничего», отражает интенсивность физического раздражителя? Влияние интенсивности раздражите-

ля проявляется в количестве потенциалов действия и во временном интервале между ними, т. е. в частоте следования импульсов потенциала действия. Чем сильнее раздражитель, тем выше частота следования импульсов потенциалов действия. Следовательно, разница между сенсорным воздействием ручного фонарика и фотовспышки — это разница частоты импульсов потенциалов действия, а не их продолжительности или величины.

1.4. Адаптация

Продолжительность ощущения зависит от времени генерирования потенциалов действия. В определенном смысле чем продолжительнее этот период, тем дольше сохраняется соответствующее ощущение. Однако при слишком продолжительном времени воздействия сенсорные рецепторы становятся менее чувствительными и скорость возникновения потенциалов действия уменьшается. В результате снижается и интенсивность ощущения. Подобное уменьшение чувствительности, наступающее вследствие продолжительного воздействия постоянного по интенсивности раздражителя, называется адаптацией. Она представляет собой явление, общее для всех сенсорных модальностей. С течением времени ощущение от постоянно воздействующего раздражителя может не только уменьшиться, но и совсем исчезнуть. Так, если долго находиться в помещении, в котором постоянно слышен какой-то шум, например шумит кондиционер или гудит старая лампа дневного света, звук сперва начинает казаться менее громким, а в конце концов его и вовсе перестаешь замечать. Наряду с продолжительностью интервалов между потенциалами действия и интенсивностью стимулов определенную роль играет также и то, что более слабые стимулы перестают восприниматься быстрее, чем сильные.

К адаптации способны все сенсорные модальности. К таким ощущениям, как тактильные и обонятельные, адаптация наступает относительно быстро. В то время как к другим ощущениям, например к боли, привыкнуть гораздо труднее, если вообще возможно. Например, чувствует ли наше тело прикосновение одежды, а запястье — давление, оказываемое ремешком от часов? Хотя адаптация, как правило, свидетельствует о снижении чувствительности, она одновременно приносит и немалую пользу. Снижая наше восприятие неизменного раздражи-

теля, который с течением времени не только может лишиться информативности, но и начать отвлекать, адаптация помогает, прежде всего, воспринимать те сигналы, которые свидетельствуют о переменах в окружающем мире. Как будет показано ниже, наша сенсорная система не только способна к адаптации, но и исключительно чувствительна к смене сигналов.

1.5. Рефрактерный период

Потенциалы действия ограничены во времени. После возникновения одного потенциала действия второй может возникнуть не ранее чем через 1 миллисекунду. Этот короткий промежуток времени, когда нейрон неактивен, называется рефрактерным периодом. Рефрактерный период ограничивает максимальную частоту возникновения импульсов 1000 потенциалами действия в секунду или менее. Когда речь идет о нервном возбуждении, это обстоятельство важно и с теоретической, и с практической точки зрения, ибо именно оно, прежде всего, определяет «пропускную способность» сенсорной системы. Другими словами, благодаря рефрактерному периоду скорость, с которой нервная система пропускает нейронные спайки, не может превышать 1000 спайков в секунду.

1.6. Скорость нейронной трансмиссии

У млекопитающих нейронный импульс, или потенциал действия, перемещается по демиелинизированному аксону за 2–3 миллисекунды. Однако, как отмечалось выше, в аксонах с миелиновой оболочкой, играющей роль электроизолятора, скорость прохождения потенциалов действия значительно возрастает. В миелинизированных аксонах потенциал действия может перемещаться со скоростью, превышающей 100 м/с. У человека миелинизирование аксонов завершается примерно к 12 годам. Это отчасти является причиной того, что дети не могут учиться или реагировать столь же быстро или действовать столь же целеустремленно, как взрослые. Их нервная система не может обрабатывать информацию со скоростью, необходимой для выполнения неко-

торых задач, особенно тех, которые требуют комплексного подхода. Такие заболевания, как рассеянный склероз, разрушают миелиновую оболочку, что приводит к уменьшению скорости передачи потенциалов действия и возможной потере сенсорно-моторной координации, для которой требуется интеграция нервной системы.

1.7. Синаптические связи

Вся сенсорная и моторная активность человека координируется совместными действиями миллиардов нервных клеток. Как они взаимодействуют друг с другом? Как сигнал, или информация, передается от одного нейрона к другому?

Потенциал действия создает импульс, который распространяется вдоль нейронной мембраны и далее продвигается по аксону до концевой синаптической луковички (рис. 1.1). Передача импульса от аксона одной нервной клетки (передающего, или пресинаптического, нейрона) к дендриту другой нервной клетки (принимающего, или постсинаптического, нейрона) происходит химическим путем. Область контакта нейронов называется синапсом (от греческого слова *synapsis*, что означает «соединение»). На рис. 1.2 представлен схематический вид синапса и синаптической щели, или синаптического пространства, микроскопического зазора между концевыми синаптическими луковичками пресинаптического нейрона и клеточной мембраной постсинаптического нейрона.

Когда потенциал действия доходит до конца аксона, до концевых синаптических луковичек, из крошечных пузырьков, или камер, называемых синаптическими пузырьками, выделяется микроскопическое количество содержащихся в них нейротрансмиттеров, которые заполняют синаптическую щель. Нейротрансмиттеры — это особые химические вещества, которые диффундируют в синаптическую щель и стимулируют прилегающий к ней постсинаптический нейрон.

Нервный импульс, проходящий по аксону, вызывает выделение нейротрансмиттеров из синаптических пузырьков концевых синаптических луковичек. Эти нейротрансмиттеры заполняют пространство между аксоном пресинаптического нейрона и дендритом принимающего, или постсинаптического, нейрона, «замыкая» электрическую цепь и обеспечивая тем самым прохождение нервного импульса.

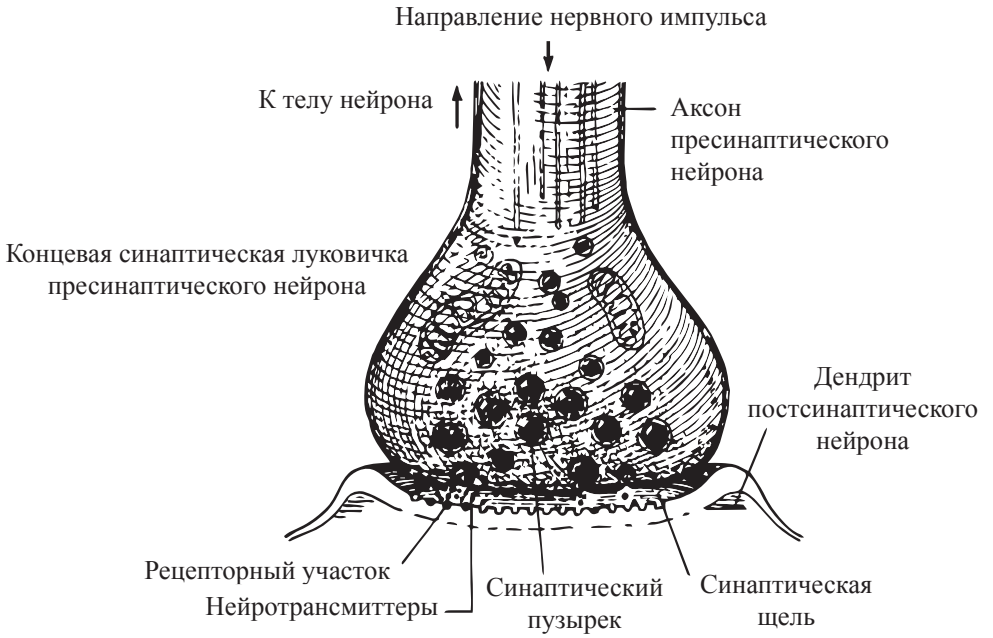


Рис. 1.2. Схематическое изображение нейронной трансмиссии в синапсе

1.8. Нейротрансмиттеры

Нейротрансмиттеры, которые высвобождаются в синапсе, либо возбуждают, либо тормозят активность постсинаптического нейрона. Когда потенциал действия запускает механизм возбуждающего синапса, происходит выделение трансмиттеров, возбуждающих нейрон, лежащий по ту сторону синаптического пространства. Если потенциал действия запускает тормозной синапс, выделяется другой нейротрансмиттер, который ингибирует противолежащий нейрон, в результате чего передача через него потенциала действия становится менее вероятной.

Ацетилхолин (АХ) является одним из важных возбуждающих трансмиттеров, на который оказывают заметное влияние такие вещества, как кофеин и кокаин. Агонисты — это вещества, усиливающие эффект специфического нейротрасмиттера. Ацетилхолин обнаружен в синапсах мозга, и доказано его участие в таких процессах, как память, внимание и пробуждение. Ацетилхолин также вызывает скелетно-мускульную активность за счет действия на двигательные нервы, и это лишь некоторые из его функций. Кураре, сильнодействующий яд,

вызывающий паралич, является антагонистом, веществом, ингибирующим ацетилхолин. Кураре иногда используют аборигены Южной Америки, живущие в бассейне реки Амазонки. Он селективно блокирует действие ацетилхолина тем, что препятствует его проникновению в синапсы нейронов нервов и мышц, в результате чего наступает полный паралич.

Норепинефрин (НЭ) — другой важный возбуждающий нейротрансмиттер, участвующий в процессе пробуждения и обеспечивающий готовность к действию. Хорошо известно, что кокаин и амфетамин, действуя как агонисты, продлевают действие норепинефрина, что еще более усиливает ярко выраженные стимулирующие психологические эффекты этих препаратов.

Гамма-аминомасляная кислота (ГАМК) является основным ингибирующим нейротрансмиттером. Она тормозит возникновение потенциала действия в нейронах, помогая тем самым контролировать точность движений мышц. Без ингибирующего действия ГАМК нервные импульсы становятся менее точными, и мышечная активность становится хаотичной, дискоординированной и даже конвульсивной.

Эндорфин — другой ингибирующий нейротрансмиттер, который блокирует нервные пути, проводящие боль. Такие широко известные вещества, как барбитураты и алкоголь, также играют роль антагонистов, поскольку подавляют выделение трансмиттеров.

1.9. Измерение потенциала действия

Потенциалы действия, или спайки, часто используются для непосредственных оценок сенсорной нейронной активности. Эти нейронно-электрические сигналы, или потенциалы действия, генерируются отдельно взятыми нейронами. Это специфический сенсорный код, обеспечивающий функционирование нервной системы. Как правило, эти нейронные сигналы измеряются и регистрируются точными чувствительными приборами, способными улавливать и усиливать очень слабые электрические сигналы. Одним из этих инструментов является микроэлектрод — тонкая металлическая проволочка с кончиком, диаметр которого менее 1 микрона, микроэлектрод в идеальном случае удается ввести или в сому нейрона, подающего сигнал, или в аксон. Электрическая активность нейрона измеряется с помощью вольтме-

тра, и результат высвечивается на осциллографе — электронно-лучевом приборе для преобразования электрических сигналов в видимое графическое изображение, фиксирующем изменение электрического заряда нейрона во времени. Экран осциллографа дает информацию о частоте импульсов потенциала действия, или ритмическую картину спайковых потенциалов.

На рис. 1.3 представлены результаты трех таких экспериментов, в которых с помощью микроэлектрода определялась активность одного и того же нейрона, реагирующего по-разному в зависимости от интенсивности сжатия пальца. Эти осциллограммы нейронной активности свидетельствуют о том, что по мере увеличения давления на палец увеличивается частота импульсов потенциала действия и изменяется ощущение.

Использование данных об активности единичных нейронов, полученных с помощью микроэлектрода, позволяет понять фундаментальные процессы, лежащие в основе функционирования нервной системы. Многие важные достижения сенсорной нейрофизиологии связаны именно с этим методом.

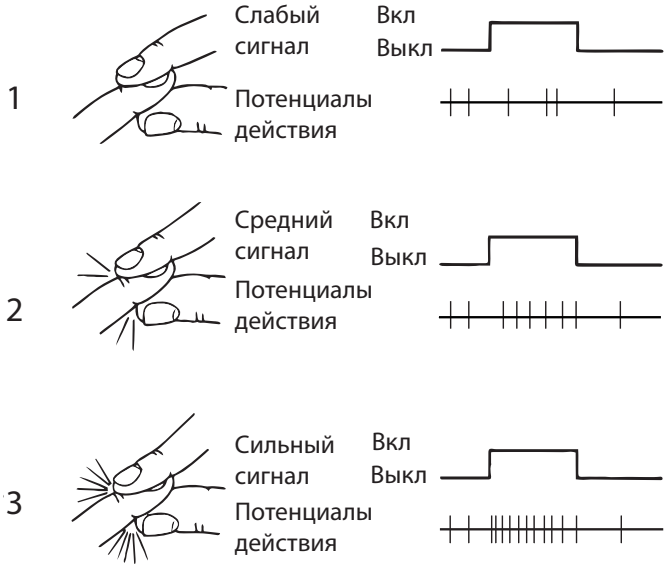


Рис. 1.3. Частота спайков в зависимости от силы сжатия пальца

Запись 1 соответствует легкому прикосновению к пальцу. Результатом такого прикосновения является слабое ощущение, и частота импульсов потенциала действия оказывается очень низкой. Запись 2 со-

ответствует более сильному сжатию пальца, при этом испытуемый чувствует оказываемое на палец давление, и частота импульсов потенциала действия возрастает. Запись 3 соответствует еще более сильному сжатию, при этом ощущается сильное давление (а возможно, и боль), и частота импульсов потенциала действия сравнительно высока. Из рисунка видно, что величина потенциалов действия (высота пиков на осциллограмме) одинакова для всех записей, но их частота следования (число потенциалов действия в единицу времени) изменяется в зависимости от интенсивности раздражителя. Таким образом, частота импульсов потенциалов действия отражает и интенсивность раздражителя, и силу ощущения [1].

1.10. Сенсорно-нейронная передача

С помощью синаптических связей единичный нейрон может быть связан с тысячами других нейронов. Поскольку человеческий мозг имеет примерно сто миллиардов нейронов, каждый из которых способен создать тысячи синаптических связей, общее количество синаптических связей в нервной системе равно по меньшей мере ста триллионам. Складывается такое впечатление, что способность нейронов к созданию сложных связей друг с другом практически безгранична и что именно этим объясняется чрезвычайное разнообразие функций нервной системы и ее сенсорные возможности.

Чтобы быть информативными для организма, электрические сигналы из нейрона поступают в мозг, что происходит благодаря нервам, трактам и ядрам центральной нервной системы. Нерв представляет собой пучок аксонов, образующих путь, по которому сигналы нейронов передаются от одного участка нервной системы другому. Наряду с сенсорными и моторными нейронами есть также и сенсорные и моторные нервы. Сенсорные нервы, называемые также афферентными нервами (от латинского affere, означающего «приносить»), передают сенсорную информацию головному и спинному мозгу, обеспечивая тем самым наш сенсорный опыт. Двигательные, или эфферентные, нервы (от латинского effere, означающего «выносить») передают информацию из головного и спинного мозга таким эфферентам, как мышцы и сухожилия.

Сенсорная информация передается нервами в центральную нервную систему, состоящую из спинного и головного мозга. Когда речь

идет о центральной нервной системе, для обозначения путей передачи сигнала чаще, чем термин «нервы», используется термин «тракты». Кроме того, в центральной нервной системе есть немало таких отделов, в которых синаптические связи образуются большими группами нейронов, называемых ядрами. Функция ядер — обработка, интеграция, трансформация и даже простейший анализ полученной сенсорной информации. Одним из важнейших ядер, в которых переключаются афферентные нервные пути, является таламус, расположенный в переднем мозге, ниже центра его полушарий. Он состоит из нескольких отделов, каждый из которых связан с определенной сенсорной модальностью. Таким образом, таламус является подкорковым центром всех видов чувствительности. Сигналы, передаваемые аксонами нейронов, поступают в определенные отделы коры головного мозга. Кора головного мозга (от латинского cortex, что значит «кора дерева») представляет собой тонкую внешнюю оболочку полушарий мозга. Ее толщина не превышает 2 миллиметров, но, будучи весьма извилистой, или складчатой, она занимает площадь примерно от 1468 до 1670 квадратных сантиметров.

Первичные проекционные зоны коры — это высокоспецифичные отделы мозга, которые связаны исключительно с определенными сенсорными модальностями. Основные проекционные зоны всех сенсорных систем лежат внутри определенных долей коры мозга (рис. 1.4).

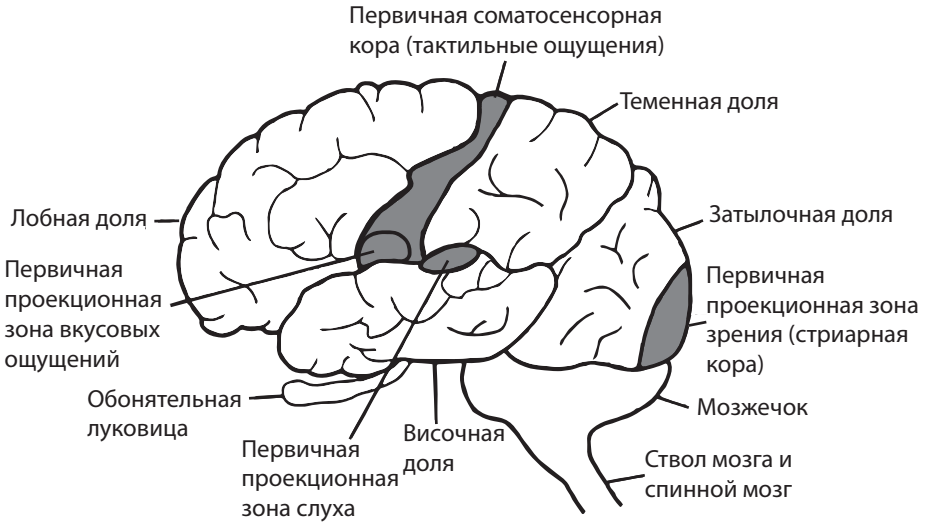


Рис. 1.4. Левое полушарие головного мозга человека

Доля — это анатомически различимая зона коры мозга, выполняющая специфическую функцию. Первичная проекционная, или принимающая, зона для слуха находится в височной доле; первичная проекционная зона для тактильных ощущений, которая также называется соматосенсорной корой — в теменной доле; первичная проекционная зона для зрения — в затылочной доле, называемой также стриарной корой, а отделом мозга, связанным с восприятием запахов, является обонятельная луковица, расположенная ниже височной доли, информация из которой проецируется на несколько отделов мозга.

2. Однослойный перцептрон

2.1. Введение

Большой вклад в становление теории нейронных сетей (1943–1958) внесли следующие исследователи:

— Мак-Каллок (McCulloch) и Питц (Pitt) в 1943 году ввели понятие нейронных сетей как вычислительных машин [2];

— Хебб (Hebb) в 1949 году постулировал первое правило самоорганизующегося обучения [3];

— Розенблатт (Rosenblatt) в 1958 году впервые предложил перцептрон в качестве модели для обучения с учителем [4].

Перцептрон является самой простой формой нейронной сети, используемой для классификации линейно разделимых моделей (то есть моделей, которые можно разделить какой-либо гиперплоскостью). Он состоит из одного нейрона с регулируемыми синаптическими весами и порога. Алгоритм, используемый для настройки свободных параметров такой нейронной сети, впервые появился в процедуре обучения, разработанной Розенблаттом [4, 5] для модели мозга. Розенблатт доказал, что если для обучения перцептрона используются образы (векторы) из двух линейно разделимых классов, то алгоритм перцептрона сходится и образует поверхность решений в виде гиперплоскости между этими двумя классами. Доказательство сходимости алгоритма известно как теорема о сходимости перцептрона. Задачи перцептрона, построенного на одном нейроне, ограничиваются разделением только двух классов (гипотез). Решение задач классификации с более чем двумя линейно разделимыми классами осуществляется включением в выходной (вычислительный) слой перцептрона нескольких нейронов. Важным моментом является то, что при описании перцептрона в качестве шаблона классификатора достаточно рассматривать слу-

чай с одним нейроном. Обобщение теории на случай более чем одного нейрона тривиально.

Единственный нейрон также формирует основу адаптивного фильтра — функционального блока, который является основным для обработки сигналов. Развитие адаптивной фильтрации во многом обязано открытию алгоритма минимизации среднеквадратичной ошибки LMS, также известного как дельта-правило. Адаптивные фильтры были успешно применены в разнообразных областях, таких как системы связи и управления, радиолокация, гидролокация, сейсмология и биомедицинская инженерия. Алгоритм LMS и перцептрон связаны между собой и поэтому рассматриваются совместно.

2.2. Задача адаптивной фильтрации

Рассмотрим динамическую систему с неизвестными математическими характеристиками. Дан набор маркированных данных ввода-вывода, генерируемых системой через равные промежутки времени. В частности, при m-мерном воздействии $\mathbf{x}(i)$ на m входных узлов системы формируется скалярный выходной сигнал $d(i)$, где $i = 1, 2, ..., n$, как показано на рис. 2.1, *а*. Внешнее поведение системы описывается набором данных

$$\mathbf{T}: \{\mathbf{x}(i), d(i); i = 1, 2, \ldots, n, \ldots\},$$

где $\mathbf{x}(i) = [x_1(i), x_2(i), \ldots, x_m(i)]^T$.

Элементы множества $\mathbf{T}$ распределены в соответствии с неизвестным вероятностным законом. Размерность m входного вектора $\mathbf{x}(i)$ называется размерностью входного пространства.

Вектор $\mathbf{x}(i)$ образуется пространственным и/или временным способом:

- когда m элементов $\mathbf{x}(i)$ возникают в различных точках пространства, $\mathbf{x}(i)$ — моментальный снимок данных;
- когда $\mathbf{x}(i)$ представляют собой набор текущих и $(m - 1)$ прошлых значений некоторого возбуждения, которые равномерно распределены во времени, $\mathbf{x}(i)$ снимается во временной области.

Рассматриваемая задача заключается в разработке модели выходного сигнала неизвестной динамической системы с несколькими вход-

ными сигналами на основе единичного нейрона. Нейронная модель, которая названа адаптивным фильтром, работает по алгоритму, контролирующему необходимые корректировки синаптических весов нейрона, со следующими ограничениями:

— работа начинается с произвольных значений синаптических весов;
— синаптические веса корректируются непрерывно в ответ на статистические изменения поведения системы (т. е. время включено в состав алгоритма);
— корректирующие значения синаптических весов вычисляются через равные промежутки времени.

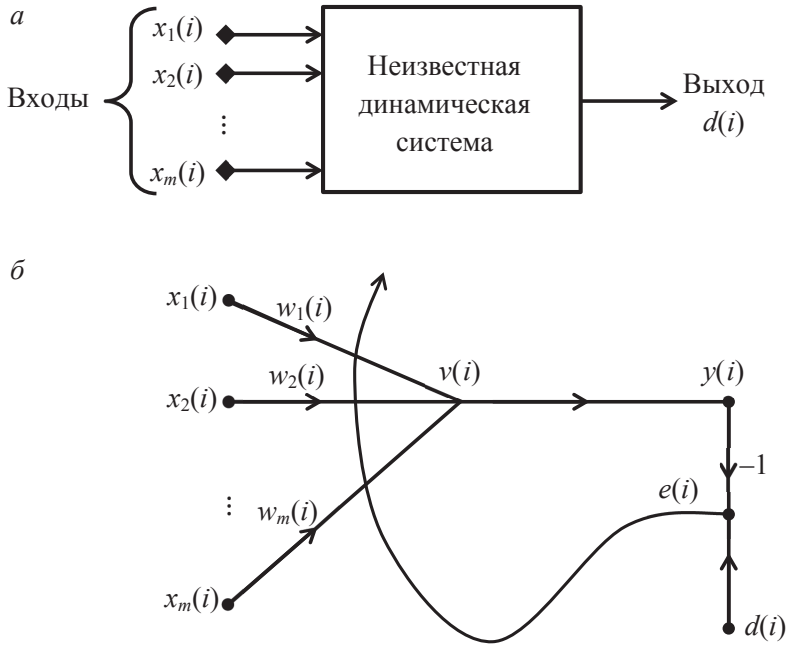


Рис. 2.1. Неизвестная динамическая система (*а*),
граф прохождения сигнала в адаптивной модели системы (*б*)

Работа адаптивного фильтра с контуром обратной связи:

1) процесс фильтрации — вычисление двух сигналов (выходного сигнала $y(i)$ и сигнала ошибки $e(i)$, который показывает отклонение $y(i)$ от выходного сигнала реальной системы $d(i)$; другие названия $e(i)$ — целевой сигнал или ожидаемый отклик);
2) процесс адаптации — автоматическая подстройка синаптических весов нейрона на основе $e(i)$.

Вследствие линейности нейрона выходной сигнал $y(i)$ совпадает с индуцированным локальным полем $v(i)$:

$$y(i) = v(i) = \sum_{k=1}^{m} w_k(i)\, x_k(i),$$

где $w_1(i)$, $w_2(i)$, ..., $w_m(i)$ — m синаптических весов нейрона, измеренных в момент времени i. В матричной форме выходной сигнал $y(i)$ можно представить как скалярное произведение векторов $\mathbf{w}(i)$ и $\mathbf{x}(i)$:

$$y(i) = \mathbf{x}^T(i)\,\mathbf{w}(i), \tag{2.1}$$

где $\mathbf{w}(i) = \left[w_1(i), w_i(i), \ldots, w_m(i)\right]^T$.

Поскольку рассматривается случай единственного нейрона, индексация синаптических весов не включает дополнительный индекс для определения нейрона.

Выходной сигнал нейрона $y(i)$ сравнивается с соответствующим фактическим выходным сигналом $d(i)$; как правило, они различаются и их разность определяет сигнал ошибки $e(i)$.

Способ использования сигнала ошибки $e(i)$ для корректировки синаптических весов нейрона определяется функцией стоимости, использованной для получения адаптивного алгоритма фильтрации. Эта задача тесно связана с задачей оптимизации. Поэтому целесообразно представить обзор некоторых методов оптимизации, который применим не только к линейным адаптивным фильтрам, но и нейронным сетям в целом.

2.3. Методы безусловной оптимизации

Рассмотрим непрерывно дифференцируемую функцию стоимости $\mathbf{E}(\mathbf{w})$, зависящую от некоторого неизвестного вектора $\mathbf{w}$. Функция $\mathbf{E}(\mathbf{w})$ отображает элементы вектора $\mathbf{w}$ в пространство действительных чисел и является мерой оптимальности выбранного для алгоритма адаптивной фильтрации вектора $\mathbf{w}$. Требуется отыскать такое решение $\mathbf{w}^*$, что

$$\mathbf{E}(\mathbf{w}^*) \le \mathbf{E}(\mathbf{w}). \tag{2.2}$$

Таким образом, необходимо решить задачу безусловной оптимизации, которая заключается в минимизации функции стоимости $\mathbf{E}(\mathbf{w})$ по отношению к вектору весов $\mathbf{w}$: $\mathbf{E}(\mathbf{w}) \to \min$.

Необходимое условие оптимальности: $\nabla \mathbf{E}(\mathbf{w}^*) = 0$, где ∇ — оператор градиента, а $\nabla \mathbf{E}(\mathbf{w})$ — вектор градиента скорости.

$$\nabla \mathbf{E}(\mathbf{w}) = \left[\frac{\partial \mathbf{E}}{\partial w_1}, \frac{\partial \mathbf{E}}{\partial w_2}, \ldots, \frac{\partial \mathbf{E}}{\partial w_m} \right]^T.$$

Класс алгоритмов безусловной оптимизации, которые хорошо подходят для создания адаптивных фильтров, основан на алгоритме последовательного спуска: в предположении исходного значения $\mathbf{w}(0)$ генерируется последовательность весовых векторов $\mathbf{w}(1)$, $\mathbf{w}(2)$, ..., такая, что значение функции стоимости уменьшается при каждой итерации алгоритма:

$$\mathbf{E}(\mathbf{w}(n+1)) < \mathbf{E}(\mathbf{w}(n)),$$

где $\mathbf{w}(n)$ — предыдущее значение вектора весов, $\mathbf{w}(n + 1)$ — последующее.

Такой алгоритм будет сходиться к оптимальному решению $\mathbf{w}^*$ при выполнении определенных условий, которые реализованы в трех нижеизложенных методах безусловной оптимизации.

Метод наискорейшего спуска

Для удобства обозначим градиент $\mathbf{g} = \nabla \mathbf{E}(\mathbf{w})$.

Корректировка векторов весов выполняется в направлении максимального уменьшения функции стоимости, т. е. в направлении, противоположном вектору градиента:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta \mathbf{g}(n),$$

где $\eta = \text{const} > 0$ — параметр скорости обучения, $\mathbf{g}(n)$ — вектор градиента в точке $\mathbf{w}(n)$. Переходя от n-й итерации к $(n + 1)$-й, алгоритм выполняет коррекцию весовых коэффициентов:

$$\Delta \mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) = -\eta \mathbf{g}(n). \qquad (2.3)$$

Разложим $\mathbf{E}(\mathbf{w}(n + 1))$ в ряд Тейлора относительно $\mathbf{w}(n)$ с точностью до членов первого порядка, что допустимо при малых значениях η, чтобы показать соблюдение условия (2.2).

$$\mathbf{E}\big(\mathbf{w}(n+1)\big) = \mathbf{E}\big(\mathbf{w}(n)\big) + \mathbf{g}^T(n)\Delta\mathbf{w}(n).$$

Подставляя эту формулу в (2.3), получим

$$\mathbf{E}\big(\mathbf{w}(n+1)\big) = \mathbf{E}\big(\mathbf{w}(n)\big) - \eta\mathbf{g}^T(n)\mathbf{g}(n) = \mathbf{E}\big(\mathbf{w}(n)\big) - \eta\big\|\mathbf{g}(n)\big\|^2.$$

При малых η значение функции стоимости уменьшается на каждой итерации, но при этом алгоритм замедляется. При увеличении η алгоритм ускоряется, но при достижении некоторого критического значения η алгоритм становится неустойчивым (расходящимся).

Метод Ньютона

Основная идея метода Ньютона заключается в минимизации квадратичной аппроксимации функции стоимости $\mathbf{E}(\mathbf{w})$ в точке $\mathbf{w}(n)$ на каждом шаге алгоритма. Используя разложение функции стоимости в ряд Тейлора с точностью до членов второго порядка, можно записать:

$$\Delta\mathbf{E}\big(\mathbf{w}(n)\big) = \mathbf{E}\big(\mathbf{w}(n+1)\big) - \mathbf{E}\big(\mathbf{w}(n)\big) =$$
$$= \mathbf{g}^T(n)\Delta\mathbf{w}(n) + 1/2\Delta\mathbf{w}^T(n)\mathbf{H}(n)\Delta\mathbf{w}(n).$$

Здесь $\mathbf{g}(n)$ — вектор градиента функции $\mathbf{E}(\mathbf{w})$ размерности $m \times 1$, вычисленный в точке $\mathbf{w}(n)$. Матрица $\mathbf{H}(n)$ — матрица Гессе; или гессиан, также вычисленный в точке $\mathbf{w}(n)$ по следующей формуле:

$$\mathbf{H} = \nabla^2\mathbf{E}(\mathbf{w}) = \begin{bmatrix} \dfrac{\partial^2\mathbf{E}}{\partial w_1^2} & \dfrac{\partial^2\mathbf{E}}{\partial w_1\partial w_2} & \cdots & \dfrac{\partial^2\mathbf{E}}{\partial w_1\partial w_m} \\ \dfrac{\partial^2\mathbf{E}}{\partial w_2\partial w_1} & \dfrac{\partial^2\mathbf{E}}{\partial w_2^2} & \cdots & \dfrac{\partial^2\mathbf{E}}{\partial w_2\partial w_m} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial^2\mathbf{E}}{\partial w_m\partial w_1} & \dfrac{\partial^2\mathbf{E}}{\partial w_m\partial w_2} & \cdots & \dfrac{\partial^2\mathbf{E}}{\partial w_m^2} \end{bmatrix}$$

Для существования гессиана функция стоимости должна быть дважды непрерывно дифференцируемой по элементам вектора $\mathbf{w}$. Дифференцируя выражение для $\Delta\mathbf{E}\big(\mathbf{w}(n)\big)$ по вектору $\Delta\mathbf{w}$, получим, что инкремент $\Delta\mathbf{E}(\mathbf{w})$ достигает минимума при условии $\mathbf{g}(n) + \mathbf{H}(n)\Delta\mathbf{w}(n) = 0$.

Решая это уравнение относительно $\Delta\mathbf{w}(n)$, получим

$$\Delta\mathbf{w}(n) = -\mathbf{H}^{-1}(n)\mathbf{g}(n) \text{ и}$$
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta\mathbf{w}(n) = \mathbf{w}(n) - \mathbf{H}^{-1}(n)\mathbf{g}(n),$$

где $\mathbf{H}^{-1}(n)$ — матрица, обратная гессиану.

Для обеспечения работоспособности метода Ньютона матрица Гессе должна быть положительно определенной для всех n, иначе этот метод требует коррекции [6, 7].

Метод Гаусса — Ньютона

Применяется для минимизации функции стоимости, представленной в виде суммы квадратов ошибок:

$$\mathbf{E}(\mathbf{w}) = 1/2\sum_{i=1}^{n} e^2(i). \tag{2.4}$$

Коэффициент 1/2 введен для упрощения последующего анализа. Все слагаемые ошибок в этой формуле вычисляются на основании вектора весов **w**, фиксированного на всем интервале наблюдения $1 \leqslant i \leqslant n$.

Сигнал ошибки $e(i)$ является функцией от настраиваемого вектора весов **w**. Для текущего значения $\mathbf{w}(n)$ зависимость $e(i)$ от **w** можно линеаризовать следующим образом:

$$e'(i,\mathbf{w}) = e(i) + \left[\frac{\partial e(i)}{\partial \mathbf{w}}\right]^T_{\mathbf{w}=\mathbf{w}(n)} \left(\mathbf{w} - \mathbf{w}(n)\right),\ i = 1, 2, \ldots, n.$$

Эта же формула в матричном виде:

$$\mathbf{e}'(n,\mathbf{w}) = \mathbf{e}(n) + \mathbf{J}(n)\left(\mathbf{w} - \mathbf{w}(n)\right), \tag{2.5}$$

$\mathbf{e}(n) = \left[e(1), e(2), \ldots, e(m)\right]^T$ — вектор ошибки, $\mathbf{J}(n)$ — матрица якобиана ошибки:

$$\mathbf{J}(n) = \left[\begin{array}{cccc} \frac{\partial \mathrm{e}(1)}{\partial w_1} & \frac{\partial \mathrm{e}(1)}{\partial w_2} & \cdots & \frac{\partial \mathrm{e}(1)}{\partial w_m} \\ \frac{\partial \mathrm{e}(2)}{\partial w_1} & \frac{\partial \mathrm{e}(2)}{\partial w_2} & \cdots & \frac{\partial \mathrm{e}(2)}{\partial w_m} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \mathrm{e}(n)}{\partial w_1} & \frac{\partial \mathrm{e}(n)}{\partial w_2} & \cdots & \frac{\partial \mathrm{e}(n)}{\partial w_m} \end{array}\right]_{\mathbf{w}=\mathbf{w}(n)}$$

Якобиан — это транспонированная матрица градиента:

$$\nabla \mathbf{e}(n)=\left[\nabla e(1),\nabla e(2),\ldots,\nabla e(n)\right].$$

Обновленный вектор

$$\mathbf{w}(n+1)=\arg\min_{\mathbf{w}}\left\{1/2\mathbf{e}'(n,\mathbf{w})^2\right\}. \tag{2.6}$$

Используя формулу (2.5) для оценки квадратичной Евклидовой нормы $\left\|\mathbf{e}'(n,\mathbf{w})\right\|^2$, получим

$$\frac{1}{2}\left\|e'(n,\mathbf{w})\right\|^2=\frac{1}{2}\left\|e(n)\right\|^2+\mathbf{e}^T(n)\mathbf{J}(n)\left(\mathbf{w}-\mathbf{w}(n)\right)+$$
$$+\frac{1}{2}\left(\mathbf{w}-\mathbf{w}(n)\right)^T\mathbf{J}^T(n)\mathbf{J}(n)\left(\mathbf{w}-\mathbf{w}(n)\right).$$

Дифференцируя это выражение по **w** и приравнивая результат к нулю, получим:

$$\mathbf{J}^T(n)e(n)+\mathbf{J}^T(n)\mathbf{J}(n)\left(\mathbf{w}-\mathbf{w}(n)\right)=0.$$

Разрешая это уравнение относительно **w** и учитывая (2.6), получим выражение, описывающее метод Гаусса — Ньютона:

$$\mathbf{w}(n+1)=\mathbf{w}(n)-\left(\mathbf{J}^T(n)\mathbf{J}(n)\right)^{-1}\mathbf{J}^T(n)\mathbf{e}(n). \tag{2.7}$$

Для реализации этого метода матрица произведения $\mathbf{J}^T(n)\mathbf{J}(n)$ должна быть несингулярной, следовательно, n строк матрицы якобиана ошибки $\mathbf{J}(n)$ должны быть линейно независимы. Поскольку это условие выполняется не всегда, часто добавляют диагональную матрицу $\delta\mathbf{I}$, где **I** — единичная матрица. Параметр δ является малой положительной константой, обеспечивающей положительную определенность матрицы $\mathbf{J}^T(n)\mathbf{J}(n)+\delta\mathbf{I}$ для всех n. Теперь уравнение метода Гаусса — Ньютона запишется в виде:

$$\mathbf{w}(n+1)=\mathbf{w}(n)-\left(\mathbf{J}^T(n)\mathbf{J}(n)+\delta\mathbf{I}\right)^{-1}\mathbf{J}^T(n)\mathbf{e}(n). \tag{2.8}$$

Чем больше количество итераций, тем слабее влияние добавки $\delta\mathbf{I}$.

Выражение (2.8) также является решением задачи минимизации модифицированной функции стоимости:

$$E(w)=\frac{1}{2}\left\{\delta\left\|\mathbf{w}-\mathbf{w}(0)\right\|^2+\sum_{i=1}^{n}e^2(i)\right\},$$

где $\mathbf{w}(0)$ — начальное значение вектора весовых коэффициентов $\mathbf{w}(i)$.

2.4. Линейный фильтр, построенный по методу наименьших квадратов

Фильтр строится для отдельного линейного нейрона.

Функция стоимости $\mathbf{E}(\mathbf{w})$ представляет собой сумму квадратов ошибок и определяется в соответствии с формулой (2.4). Принимая во внимание формулу (2.1) и определение вектора ошибки $\mathbf{e}(n)$, можно записать следующее соотношение:

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{X}(n)\mathbf{w}(n), \tag{2.9}$$

где $\mathbf{d}(n) = [d(1), d(2), ..., d(n)]^T$ — вектор желаемого отклика размерности n; $\mathbf{X}(n) = [\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(n)]^T$ — матрица данных размерности $n \times m$.

Дифференцируя выражение (2.9) по $\mathbf{w}(n)$, получим матрицу градиента $\nabla\mathbf{E}(n) = -\mathbf{X}^T(n)$ и якобиан $\mathbf{e}(n)$ $\mathbf{J}(n) = -\mathbf{X}(n)$.

Так как уравнение ошибки (2.5) является линейным относительно вектора весовых коэффициентов $\mathbf{w}(n)$, метод Гаусса — Ньютона сходится за одну итерацию. Подставляя (2.9) и выражение для якобиана в (2.7), получим:

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \left(\mathbf{X}^T(n)\mathbf{X}(n)\right)^{-1}\mathbf{X}^T(n)\left(\mathbf{d}(n) - \mathbf{X}(n)\mathbf{w}(n)\right) = \\ &= \left(\mathbf{X}^T(n)\mathbf{X}(n)\right)^{-1}\mathbf{X}^T(n)\mathbf{d}(n).\end{aligned} \tag{2.10}$$

Выражение $\left(\mathbf{X}^T(n)\mathbf{X}(n)\right)^{-1}\mathbf{X}^T(n)$ называют псевдообратной матрицей для матрицы данных $\mathbf{X}(n)$ и обозначают как $\mathbf{X}^+(n)$. В этих обозначениях вектор весовых коэффициентов $\mathbf{w}(n + 1) = \mathbf{X}^+(n)\,\mathbf{d}(n)$ является решением линейной задачи фильтрации, решаемой по методу наименьших квадратов на интервале наблюдения длительности n.

Фильтр Винера как ограниченная форма линейного фильтра, построенного по методу наименьших квадратов для эргодической среды

Частным случаем, представляющим особый интерес, является получение вектора входного сигнала $\mathbf{x}(i)$ и желаемого отклика $\mathbf{d}(i)$ из эргодической стационарной среды. Для этого случая вместо усреднения

по времени можно использовать математическое ожидание или усреднение по множеству [8]. Такая среда частично описывается следующими статистическими характеристиками второго порядка.

1. Матрица корреляции $\mathbf{R}_{\mathbf{x}}$ вектора входного сигнала $\mathbf{x}(i)$:

$$\mathbf{R}_{\mathbf{x}} = E\left[\mathbf{x}(i)\mathbf{x}^T(i)\right] = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}(i)\mathbf{x}^T(i) = \lim_{n\to\infty}\frac{1}{n}\mathbf{X}^T(n)\mathbf{X}(n).$$

2. Вектор взаимной корреляции $\mathbf{r}_{\mathbf{x}d}$ между вектором входного сигнала $\mathbf{x}(i)$ и ожидаемого отклика $\mathbf{d}(i)$:

$$\mathbf{r}_{\mathbf{x}d} = E\left[\mathbf{x}(i)d(i)\right] = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}(i)d(i) = \lim_{n\to\infty}\frac{1}{n}\mathbf{X}^T(n)\mathbf{d}(n),$$

где E — статистический оператор математического ожидания. Тогда решением линейной задачи фильтрации по методу наименьших квадратов (2.10) является вектор весовых коэффициентов $\mathbf{w}_{\mathrm{o}}$, названный винеровским решением линейной задачи оптимальной фильтрации:

$$w_o = \lim_{n\to\infty} w(n+1) = \lim_{n\to\infty}\left(\mathbf{X}^T(n)\mathbf{X}(n)\right)^{-1}\mathbf{X}^T(n)\mathbf{d}(n) =$$
$$= \lim_{n\to\infty}\left(\mathbf{X}^T(n)\mathbf{X}(n)\right)^{-1}\lim_{n\to\infty}\mathbf{X}^T(n)\mathbf{d}(n) = \mathbf{R}_{\mathbf{x}}^{-1}\mathbf{r}_{\mathbf{x}d},$$

где $\mathbf{R}_{\mathbf{x}}^{-1}$ — обратная матрица для матрицы корреляции $\mathbf{R}_{\mathbf{x}}$.

Исходя из этого, можно сформулировать следующее утверждение: для эргодического процесса линейный фильтр, построенный по методу наименьших квадратов, асимптотически сходится к фильтру Винера по мере приближения количества наблюдений к бесконечности.

Часто недоступна информация о статистических характеристиках второго порядка: матрице корреляции $\mathbf{R}_{\mathbf{x}}$ для вектора входного сигнала $\mathbf{x}(n)$ и векторе взаимной корреляции $\mathbf{r}_{\mathbf{x}d}$ между $\mathbf{x}(n)$ и желаемым откликом $d(n)$. Поэтому на практике используют линейный адаптивный фильтр, который позволяет настраивать свободные параметры фильтра в соответствии со статистическими вариациями среды. Наиболее популярным алгоритмом такой настройки на непрерывной основе является алгоритм минимизации среднеквадратической ошибки, который самым тесным образом связан с фильтром Винера.

2.5. Алгоритм минимизации среднеквадратической ошибки (LMS)

Алгоритм основан на использовании дискретных значений функции стоимости:

$$\mathbf{E}(\mathbf{w}) = 1/2e^2(n),$$

где $e(n)$ — сигнал ошибки, измеренный в момент времени n. Дифференцируя $\mathbf{E}(\mathbf{w})$ по вектору весов $\mathbf{w}$, получим:

$$\frac{\partial \mathbf{E}(\mathbf{w})}{\partial \mathbf{w}} = e(n)\frac{\partial e(n)}{\partial \mathbf{w}}.$$

Поскольку алгоритм минимизации среднеквадратической ошибки работает с линейным нейроном, сигнал ошибки можно записать в следующем виде:

$$\mathbf{e}(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n). \tag{2.11}$$

Следовательно,

$$\frac{\partial e(n)}{\partial \mathbf{w}} = -\mathbf{x}(n),$$

$$\frac{\partial \mathbf{E}(\mathbf{w})}{\partial \mathbf{w}} = -\mathbf{x}(n)e(n).$$

Используя полученный результат, можно оценить вектор градиента

$$\hat{\mathbf{g}}(n) = -\mathbf{x}(n)e(n)$$

и использовать его в методе наискорейшего спуска. Тогда алгоритм минимизации среднеквадратической ошибки запишется в следующем виде:

$$\hat{\mathbf{w}}(n+1) = \mathbf{w}(n) + \eta\mathbf{x}(n)e(n), \tag{2.12}$$

где η — параметр скорости обучения. Контур обратной связи для вектора весов $\hat{\mathbf{w}}(n)$ в алгоритме минимизации среднеквадратической ошибки ведет себя как низкочастотный фильтр [9]. Усредненная временная константа этой фильтрации обратно пропорциональна параметру скорости обучения η. Следовательно, при малых значениях η процесс адаптации будет продвигаться медленно. При этом алгоритм

минимизации среднеквадратической ошибки будет запоминать большее количество предшествующих данных, а значит, более точной будет фильтрация. Другими словами, величина, обратная параметру скорости обучения η, является мерой памяти алгоритма минимизации среднеквадратической ошибки.

В формуле (2.12) вместо $\mathbf{w}(n)$ мы использовали $\widehat{\mathbf{w}}(n)$. Этим подчеркивается тот факт, что алгоритм минимизации среднеквадратической ошибки только оценивает вектор весовых коэффициентов на основе метода наискорейшего спуска. Следовательно, используя алгоритм минимизации среднеквадратической ошибки, можно пожертвовать одной из отличительных особенностей алгоритма наискорейшего спуска. В последнем вектор весовых коэффициентов $\mathbf{w}(n)$ изменяется по детерминированной траектории в пространстве весов при заранее выбранном параметре η. В противоположность этому в алгоритме минимизации среднеквадратической ошибки вектор $\widehat{\mathbf{w}}(n)$ перемещается по случайной траектории. По этой причине алгоритм минимизации среднеквадратической ошибки иногда называют стохастическим градиентным алгоритмом. При стремлении количества итераций в алгоритме LMS к бесконечности вектор $\widehat{\mathbf{w}}(n)$ выписывает хаотичную траекторию вокруг винеровского решения $\mathbf{w}_o$. Алгоритм минимизации среднеквадратической ошибки не требует знания статистических характеристик окружающей среды.

Краткое описание алгоритма минимизации среднеквадратической ошибки представлено в табл. 2.1, которая показывает, что для инициализации алгоритма достаточно обнулить его начальный вектор весовых коэффициентов.

Таблица 2.1.

Краткое описание алгоритма минимизации среднеквадратической ошибки

Обучающий пример	Вектор входного сигнала = $\mathbf{x}(n)$ Желаемый отклик = $d(n)$
Выбираемый пользователем параметр	η
Инициализация весов	$\hat{\mathbf{w}}(0) = 0$
Вычислительная схема	Для $n = 1, 2, \ldots$ $e(n) = d(n) - \mathbf{x}^T(n)\hat{\mathbf{w}}(n)$ $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta\mathbf{x}(n)e(n)$

Граф передачи сигнала для алгоритма минимизации среднеквадратической ошибки

Объединяя формулы (2.11) и (2.12), эволюцию вектора весов в алгоритме минимизации среднеквадратической ошибки можно представить в следующем виде:

$$\widehat{\mathbf{w}}(n+1) = \widehat{\mathbf{w}}(n) + \eta \mathbf{x}(n)\left[d(n) - \mathbf{x}^T(n)\widehat{\mathbf{w}}(n) \right] =$$

$$= \left[\mathbf{I} - \eta \mathbf{x}(n)\mathbf{x}^T(n) \right] \widehat{\mathbf{w}}(n) + \eta \mathbf{x}(n) d(n),$$

где $\mathbf{I}$ — единичная матрица. При использовании алгоритма минимизации среднеквадратической ошибки $\widehat{\mathbf{w}}(n) = z^{-1}\left[\widehat{\mathbf{w}}(n+1) \right]$, где z^{-1} — оператор единичной задержки, реализующей память алгоритма. Используя эти выражения, алгоритм минимизации среднеквадратической ошибки можно представить в виде графа передачи сигнала, показанного на рис. 2.2. На этом графе видно, что алгоритм минимизации среднеквадратической ошибки является всего лишь частным случаем стохастической системы с обратной связью. Наличие обратной связи оказывает первостепенное влияние на сходимость алгоритма LMS.

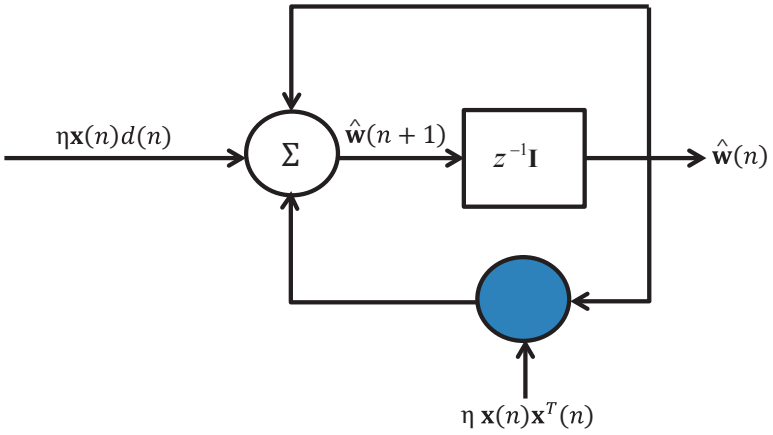


Рис. 2.2. Граф передачи сигнала для алгоритма минимизации среднеквадратической ошибки

Условия сходимости алгоритма LMS

Из теории управления известно, что стабильность системы с обратной связью определяется параметрами обратной связи. Из рис. 2.2 мы видим, что это нижняя часть обратной связи обе-

спечивает изменчивость поведения алгоритма LMS. В частности, существуют две различные величины, коэффициент скорости обучения η и входной вектор $\mathbf{x}(n)$, которые определяют пропускную способность этой цепи обратной связи. Таким образом, для каждой среды, из которой поступают входные векторы, следует подбирать собственный параметр скорости обучения η, который обеспечит сходимость алгоритма LMS.

Первым критерием сходимости алгоритма минимизации среднеквадратической ошибки является сходимость в среднем:

$$E\left[\hat{\boldsymbol{w}}(n)\right] \to \mathbf{w}_o \text{ при } n \to \infty,$$

где $\mathbf{w}_o$ — решение Винера. К сожалению, такой критерий сходимости не имеет большого практического значения, так как последовательность произвольных векторов, имеющих среднее значение 0, будет удовлетворять этому условию.

С практической точки зрения вопрос устойчивости играет роль именно в смысле среднеквадратической сходимости:

$$E\left[e^2(n)\right] \to \text{const при } n \to \infty.$$

Детальный анализ сходимости в смысле среднеквадратического значения для алгоритма LMS чрезвычайно сложен. Для математического упрощения алгоритма делают следующие предположения.

1. Векторы входных сигналов $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(n)$ являются статистически независимыми.

2. В момент времени n вектор $\mathbf{x}(n)$ является статистически независимым от всех предыдущих желаемых откликов, $d(1), d(2), \ldots, d(n-1)$.

3. В момент времени n желаемый отклик $d(n)$ зависит от вектора $\mathbf{x}(n)$, но статистически не зависит от всех предыдущих значений желаемого отклика.

4. Вектор входного сигнала $\mathbf{x}(n)$ и желаемый отклик $d(n)$ выбираются из множества, подчиняющегося распределению Гаусса.

Статистический анализ алгоритма минимизации среднеквадратической ошибки при наличии вышеуказанных допущений получил название теории независимости [10]. С учетом допущений теории независимости и достаточно малого значения параметра скорости обучения в [9] показано, что алгоритм минимизации среднеквадратической ошибки сходится в смысле среднеквадратического значения, если η удовлетворяет условию

$$0 < \eta < \frac{2}{\lambda_{max}},$$

где λ_{max} — наибольшее собственное значение матрицы корреляции $\mathbf{R}_x$. В типичных приложениях алгоритма LMS значение λ_{max} не известно. Чтобы обойти эту сложность, в качестве оценки сверху значения λ_{max} можно использовать след матрицы $\mathbf{R}_x$. В этом случае неравенство можно переписать в следующем виде:

$$0 < \eta < \frac{2}{\mathrm{tr}[\mathbf{R}_x]},$$

где tr [$\mathbf{R}_x$] — след матрицы $\mathbf{R}_x$. След квадратной матрицы определяется как сумма ее диагональных элементов. Так как каждый из диагональных элементов матрицы корреляции $\mathbf{R}_x$ является среднеквадратическим значением соответствующего входного сигнала, условие сходимости алгоритма минимизации среднеквадратической ошибки можно сформулировать в виде

$$0 < \eta < \frac{2}{\text{сумма среднеквадратических значений входных сигналов}}.$$

Если параметр скорости обучения удовлетворяет этому условию, то алгоритм минимизации среднеквадратической ошибки сходится также и в смысле среднего значения. Обратное утверждение верно не всегда.

Важным преимуществом алгоритма минимизации среднеквадратической ошибки является его простота (см. табл. 2.1). Кроме того, этот алгоритм независим от модели и, следовательно, при малой неопределенности в модели и небольших возмущениях (с малой энергией) сигнал ошибки также будет невелик. В более строгих математических терминах алгоритм минимизации среднеквадратической ошибки является оптимальным согласно минимаксному критерию (H^∞), основная идея оптимальности которого заключена в обеспечении наилучшего выполнения пессимистического сценария: если не знаете, с чем столкнулись, предположите наихудшее и оптимизируйте решение.

Долгое время алгоритм минимизации среднеквадратической ошибки рассматривался как частный случай алгоритма градиентного спуска. Однако оптимальность алгоритма минимизации среднеквадратической ошибки в смысле H^∞ придает ему более устойчивое положение

в рассматриваемой предметной области. В частности, он обеспечивает приемлемые результаты не только в стационарной, но и в нестационарной среде, в которой статистические характеристики изменяются во времени. В такой среде оптимальное решение Винера зависит от времени, а алгоритм минимизации среднеквадратической ошибки выполняет дополнительную задачу отслеживания изменения параметров фильтра Винера.

Основным ограничением алгоритма минимизации среднеквадратической ошибки является низкая скорость сходимости и чувствительность к изменению собственных значений матрицы входных сигналов [9]. Для сходимости алгоритма минимизации среднеквадратической ошибки обычно требуется в 10 раз больше итераций, чем размерность пространства входных сигналов. Медленная сходимость становится действительно серьезной преградой, если размерность пространства входных данных очень велика. Что же касается чувствительности, то наибольшая чувствительность алгоритма проявляется к изменению числа обусловленности или разброса собственных чисел матрицы корреляции $\mathbf{R}_x$ входного вектора $\mathbf{x}$. Число обусловленности $\chi(\mathbf{R}_x)$ определяется следующим выражением:

$$\chi(\mathbf{R}_x) = \frac{\lambda_{max}}{\lambda_{min}},$$

где λ_{max} и λ_{min} — соответственно максимальное и минимальное собственные числа матрицы $\mathbf{R}_x$. Чувствительность алгоритма минимизации среднеквадратической ошибки к вариациям числа обусловленности $\chi(\mathbf{R}_x)$ становится действительно опасной, когда обучающая выборка, которой принадлежит вектор $\mathbf{x}(n)$, является плохо обусловленной, т. е. число обусловленности $\chi(\mathbf{R}_x)$ достаточно велико.

Поскольку в алгоритме минимизации среднеквадратической ошибки матрица гессиана, определяемая как вторая производная от функции стоимости $\mathbf{E}(\mathbf{w})$ по вектору $\mathbf{w}$, эквивалентна матрице корреляции $\mathbf{R}_x$, то в дальнейших рассуждениях оба этих термина употребляются в одинаковом значении.

2.6. Графики процесса обучения

Одним из самых информативных способов проверки сходимости алгоритма минимизации среднеквадратической ошибки и всех адаптивных фильтров в целом является построение графиков процесса обучения, или так называемых кривых обучения, для различных условий внешней среды. Кривая обучения является графиком изменения среднеквадратического значения ошибки оценивания $\mathbf{E}_{av}(n)$ в зависимости от количества итераций n.

Представим себе эксперимент, проводимый над множеством адаптивных фильтров, когда каждый из них работает под управлением отдельного алгоритма. Предполагается, что начальные условия и принципы работы всех алгоритмов одинаковы. Различие между ними определяется случайностью выбора вектора входного сигнала $\mathbf{x}(n)$ и желаемого отклика $d(n)$ из имеющейся обучающей выборки. Для каждого из фильтров строится график изменения среднеквадратической ошибки оценивания (т. е. разности между желаемым откликом и фактическим выходным сигналом фильтра) относительно количества итераций. Полученное таким образом множество кривых обучения состоит из зашумленных графиков экспоненциального типа. Наличие шума связано со стохастической природой адаптивного фильтра. Чтобы построить усредненную по этому множеству кривую обучения (т. е. график $\mathbf{E}_{av}(n)$ относительно n), нужно вычислить средние значения по всем кривым, участвующим в эксперименте. Это способствует снижению влияния шума на результат.

В предположении устойчивости адаптивного фильтра усредненная по множеству кривая обучения начинается с достаточно большого значения $\mathbf{E}_{av}(0)$, определяемого начальными условиями, и затем ее значения уменьшаются с некоторой скоростью, зависящей от типа фильтра, а в пределе сходятся к некоторому устойчивому значению $\mathbf{E}_{av}(\infty)$ (рис. 2.3). Основываясь на этой кривой, можно определить скорость сходимости адаптивного фильтра как число итераций, необходимых для того, чтобы для любого произвольного начального значения $\mathbf{E}_{av}(0)$ величина $\mathbf{E}_{av}(n)$ уменьшилась в 10 раз.

Из усредненной кривой обучения можно получить еще одну характеристику адаптивного фильтра — рассогласование $\mathbf{M}$:

$$\mathbf{M} = \frac{\mathbf{E}(\infty) - \mathbf{E}_{\min}}{\mathbf{E}_{\min}} = \frac{\mathbf{E}(\infty)}{\mathbf{E}_{\min}} - 1.$$

где $\mathbf{E}_{\min}$ — минимальная среднеквадратическая ошибка, обеспечиваемая фильтром Винера, построенным на основе известных значений матрицы корреляции $\mathbf{R}_x$ и вектора взаимной корреляции $\mathbf{r}_{xd}$.

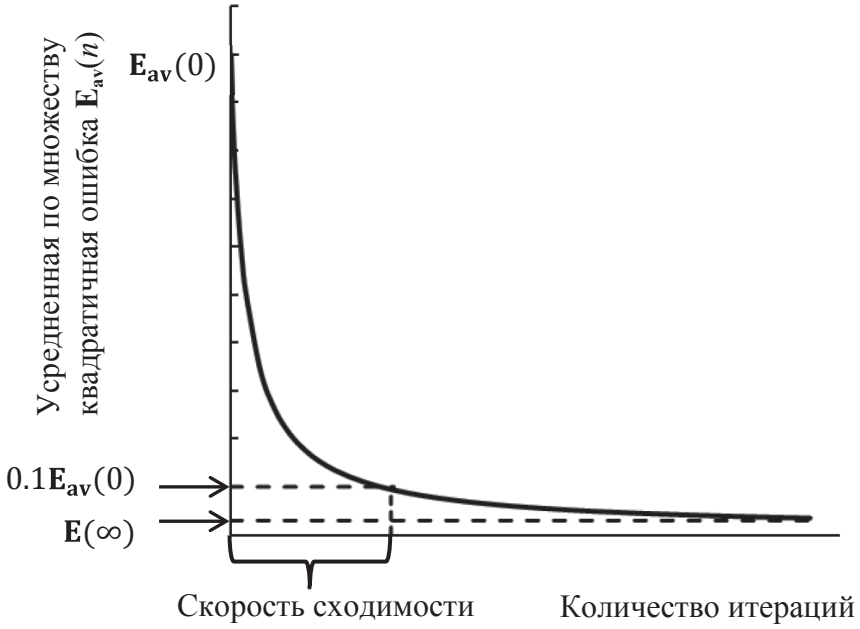


Рис. 2.3. Идеализированная кривая обучения алгоритма LMS

Рассогласование **M** является мерой измерения близости адаптивного фильтра к оптимальному в смысле среднеквадратической ошибки. Чем ближе значение **M** к единице, тем более точной является адаптивная фильтрация алгоритма.

Еще одной важной характеристикой алгоритма минимизации среднеквадратической ошибки является время установки. Однозначного определения этой величины не существует. Например, кривую обучения можно грубо аппроксимировать экспоненциальной функцией с усредненной временной константой t_{av}. Чем меньше значение t_{av}, тем короче время установки.

В качестве неплохой аппроксимации величины рассогласования **M** используют параметр скорости обучения η, который прямо пропорционален ей, в отличие от величины t_{av}, которая обратно пропор-

циональна η. Таким образом, получается противоречие: если уменьшить параметр скорости обучения для уменьшения рассогласования, то увеличивается время установки алгоритма LMS. Следовательно, для ускорения процесса обучения нужно увеличивать коэффициент скорости обучения, однако следует учесть, что одновременно с этим будет увеличиваться и рассогласование. Выбору параметра η уделяется большое внимание, так как он отвечает за общую производительность алгоритма LMS.

2.7. Изменение параметра скорости обучения по модели отжига

Сложность работы с алгоритмом минимизации среднеквадратической ошибки связана с тем, что параметр скорости обучения является эмпирической константой. Ее можно не изменять в течение всего процесса обучения. Это простейший способ задания коэффициента скорости обучения. В отличие от него в методах стохастической аппроксимации параметр интенсивности обучения изменяется со временем. Одной из самых распространенных форм изменения этого параметра, описанных в литературе по стохастической аппроксимации, является следующая:

$$\eta = \frac{\text{const}}{n}.$$

Такая форма гарантирует сходимость алгоритма стохастической аппроксимации. Если константа велика, существует опасность выхода алгоритма из-под контроля на первых шагах аппроксимации (при малых n). В качестве альтернативы можно использовать подход на основе поиска и сходимости

$$\eta = \frac{\eta_0}{1 + (n / \tau)},$$

где η и τ — заданные пользователем константы. На первых шагах адаптации число n является малым по сравнению с константой времени поиска τ. Поэтому параметр скорости $\eta(n)$ практически равен константе η_0 и алгоритм ведет себя как стандартный алгоритм минимиза-

ции среднеквадратической ошибки (рис. 2.4). При выборе больших значений η_0 настраиваемые весовые коэффициенты фильтра будут находиться вблизи приемлемых значений. При количестве итераций n, значительно превосходящих константу времени поиска τ, параметр интенсивности скорости $\eta(n)$ будет сходиться к функции $\eta = \text{const}/n$, где $\text{const} = \tau\eta_0$ (рис. 2.4). Алгоритм при этом будет вести себя как обычный алгоритм стохастической аппроксимации, а веса будут сходиться к своим оптимальным значениям. Таким образом, изменение коэффициента скорости обучения на основе метода поиска и сходимости обеспечивает сочетание полезных свойств обычного алгоритма минимизации среднеквадратической ошибки с методами традиционной теории стохастической аппроксимации.

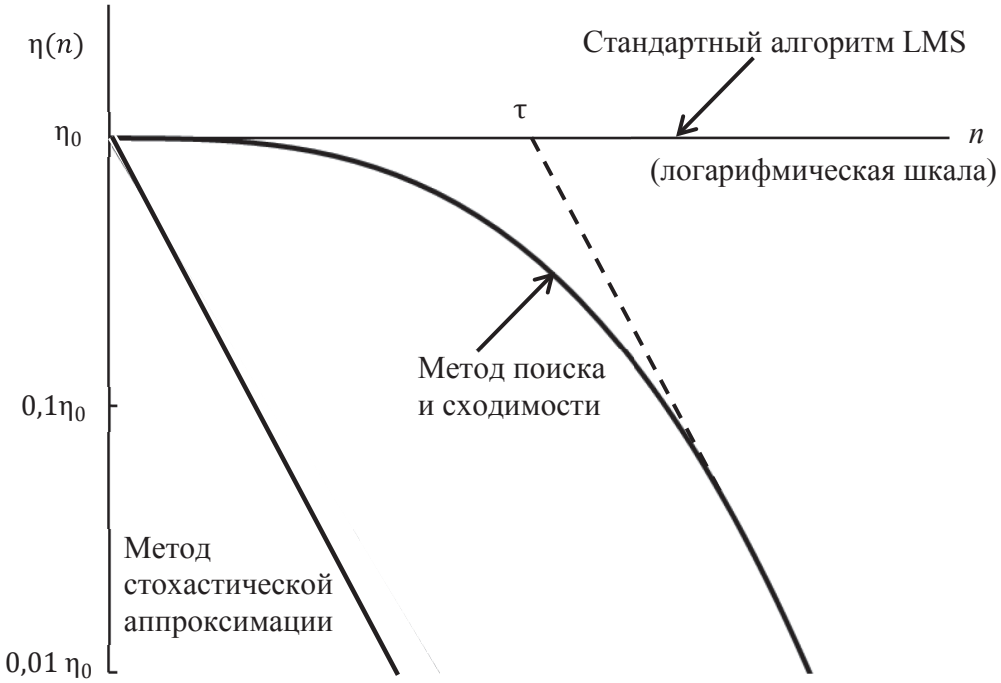


Рис. 2.4. Изменение коэффициента скорости обучения по методу моделирования отжига

2.8. Перцептрон

Если алгоритм LMS разработан для линейного нейрона, то перцептрон Розенблатта (далее просто перцептрон) строится для нелинейного, а именно модели нейрона Мак-Каллока — Питца. Такая нейронная модель состоит из линейного сумматора и ограничителя, реализованного в виде пороговой функции вычисления знака (рис. 2.5). Суммирующий узел этой нейронной модели вычисляет линейную комбинацию входных сигналов, поступающих на синапсы с учетом внешнего возмущения (порога). Полученная сумма (так называемое индуцированное локальное поле) передается на узел ограничителя. Выход нейрона принимает значение +1, если сигнал на выходе сумматора положителен, и −1, если отрицателен.

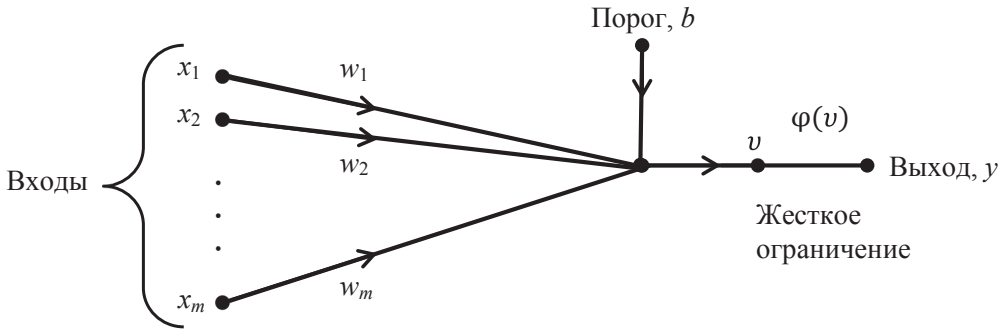


Рис. 2.5. Граф передачи сигнала для перцептрона

На диаграмме передачи сигнала (рис. 2.5) синаптические веса перцептрона обозначены $w_1, w_2, \ldots, w_m$, сигналы, поступающие на вход перцептрона, обозначены $x_1, x_2, \ldots, x_m$, а пороговое значение — b. Исходя из структуры модели, можно заключить, что входной сигнал ограничителя (т. е. индуцированное локальное поле) нейрона определяется выражением

$$\upsilon = \sum_{i=1}^{m} w_i x_i + b.$$

Целью перцептрона является корректное отнесение множества внешних стимулов $x_1, x_2, \ldots, x_m$ к одному из двух классов: $\mathbf{C}_1$ или $\mathbf{C}_2$. Основное правило такой классификации: входной сигнал относится к классу $\mathbf{C}_1$, если выход y равен +1, и к классу $\mathbf{C}_2$ в противном случае.

Чтобы глубже изучить поведение классификатора, целесообразно построить карту областей решения в m-мерном пространстве сигналов, определяемом переменными $x_1, x_2, ..., x_m$. В простейшем случае (когда в качестве классификатора выступает перцептрон) имеются всего две области решения, разделенные гиперплоскостью, определяемой формулой

$$\sum_{i=1}^{m} w_i x_i + b = 0.$$

Это проиллюстрировано на рис. 2.6 для случая двух переменных, x_1 и x_2, когда разделяющая гиперплоскость вырождается в прямую. Точки (x_1, x_2), лежащие выше этой прямой, относятся к классу $\mathbf{C}_1$, а точки, расположенные ниже прямой, принадлежат к классу $\mathbf{C}_2$. Пороговое значение определяет смещение разделяющей поверхности по отношению к началу координат.

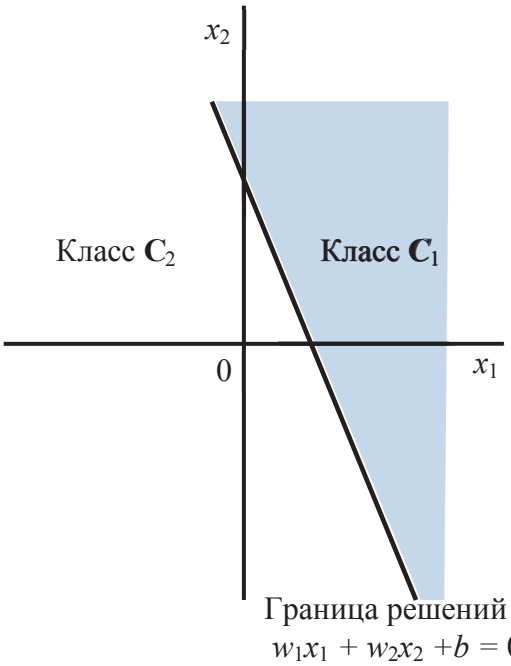


Рис. 2.6. Разделяющая поверхность в виде гиперплоскости для двумерной задачи классификации образов

Синаптические веса перцептрона (обозначены $w_1, w_2, ..., w_m$) можно адаптировать итеративным методом. В частности, для настройки

весовых коэффициентов можно использовать алгоритм, основанный на коррекции ошибок и получивший название алгоритма сходимости перцептрона.

2.9. Теорема о сходимости перцептрона

Чтобы вывести алгоритм обучения перцептрона, основанный на коррекции ошибок, удобно построить модифицированный граф передачи сигнала (рис. 2.7). В этой модели, которая эквивалентна модели нейрона, показанной на рис. 2.5, порог $b(n)$ рассматривается как синаптический вес связи с фиксированным входным сигналом +1.

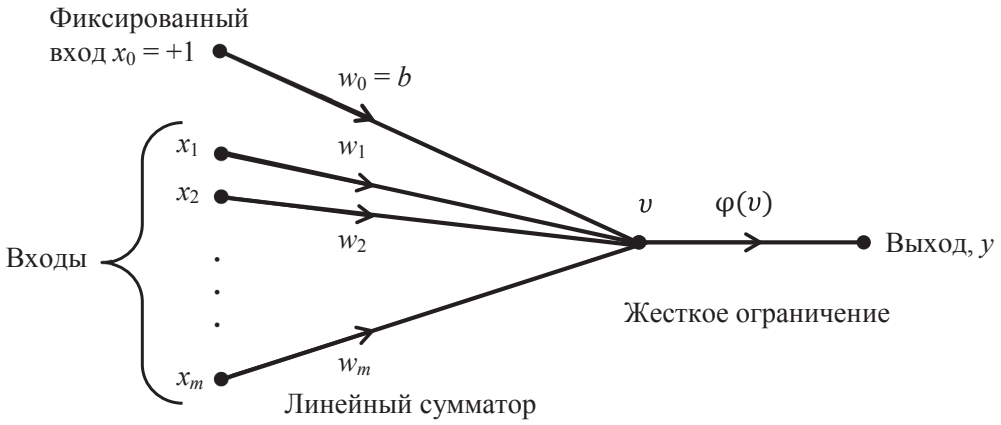


Рис. 2.7. Эквивалентный граф передачи сигнала для перцептрона

Это можно описать следующим входным вектором размерности $(m + 1)$:

$$\mathbf{x}(n) = [+1, x_1(n), x_2(n), ..., x_m(n)]^T,$$

где n — номер итерации алгоритма. Аналогично можно определить $(m + 1)$-мерный вектор весовых коэффициентов:

$$\mathbf{w}(n) = [b(n), w_1(n), w_2(n), ..., w_m(n)]^T.$$

Выход линейного сумматора запишется в более компактной форме:

$$\upsilon(n) = \sum_{i=0}^{m} w_i(n) x_i(n) = \mathbf{w}^T(n) x(n),$$

где $\mathbf{w}_0(n) = b(n)$ — пороговое значение. При фиксированном значении n уравнение $\mathbf{w}^T\mathbf{x} = 0$ в m-мерном пространстве с координатами x_1, x_2, ..., x_m определяет гиперплоскость (для некоторого заранее заданного значения порога), которая является поверхностью решений для двух различных классов входных сигналов.

Чтобы перцептрон функционировал корректно, два класса, $\mathbf{C}_1$ и $\mathbf{C}_2$, должны быть линейно разделимыми, то есть для правильной классификации образы должны быть значительно отдалены друг от друга, чтобы поверхность решений могла представлять собой гиперплоскость. Это требование проиллюстрировано на рис. 2.8 для случая двумерного перцептрона. На рис. 2.8, *а* два класса — $\mathbf{C}_1$ и $\mathbf{C}_2$ — значительно удалены друг от друга, и их можно разделить гиперплоскостью (в данном случае — прямой). Если эти два класса сдвинуть ближе друг к другу (рис. 2.8, *б*), они станут нелинейно разделимыми. Такая ситуация выходит за рамки вычислительных возможностей перцептрона.

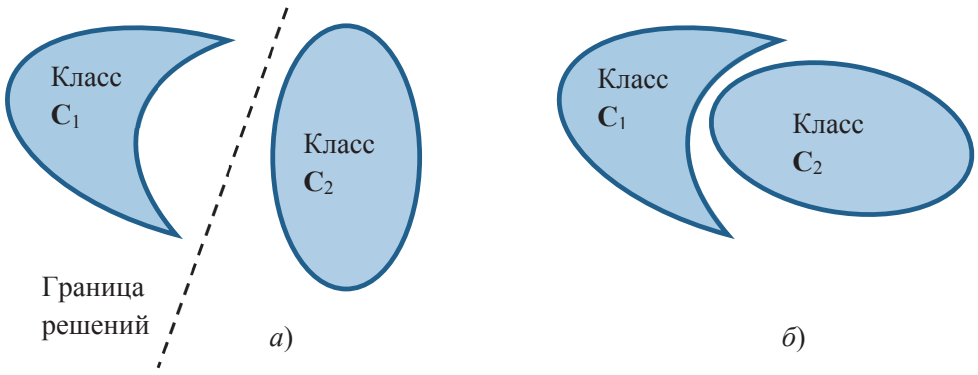


Рис. 2.8. Пара линейно разделимых (*а*) и нелинейно разделимых (*б*) образов

Теперь предположим, что входные переменные перцептрона принадлежат двум линейно разделимым классам. Пусть $\mathbf{X}_1$ — подмножество векторов обучения $\mathbf{x}_1(1)$, $\mathbf{x}_1(2)$, ..., которое принадлежит классу $\mathbf{C}_1$, $\mathbf{X}_2$ — подмножество векторов обучения $\mathbf{x}_2(1)$, $\mathbf{x}_2(2)$, ..., относящееся к классу $\mathbf{C}_2$. Объединение подмножеств $\mathbf{X}_1$ и $\mathbf{X}_2$ составляет все обучающее множество $\mathbf{X}$. Использование подмножеств $\mathbf{X}_1$ и $\mathbf{X}_2$ для обучения классификатора позволит настроить вектор весов $\mathbf{w}$ таким образом, что классы $\mathbf{C}_1$ и $\mathbf{C}_2$ будут линейно разделимыми. Это значит, что суще-

ствует такой вектор весовых коэффициентов $\mathbf{w}$, для которого истинно следующее утверждение:

$$\mathbf{w}^T\mathbf{x} > 0 \text{ для любого входного вектора } \mathbf{x}, \text{ принадлежащего классу } \mathbf{C}_1,$$

$$\mathbf{w}^T\mathbf{x} \leqslant 0 \text{ для любого входного вектора } \mathbf{x}, \text{ принадлежащего классу } \mathbf{C}_2. \quad (2.13)$$

Во второй строке утверждения (2.13) мы произвольно указали, что при равенстве $\mathbf{w}^T\mathbf{x} = 0$ входной вектор $\mathbf{x}$ принадлежит именно классу $\mathbf{C}_2$. При определенных таким образом подмножествах $\mathbf{X}_1$ и $\mathbf{X}_2$ задача обучения элементарного перцептрона сводится к нахождению такого вектора весов $\mathbf{w}$, для которого выполняются оба неравенства (2.13).

Алгоритм адаптации вектора весовых коэффициентов элементарного перцептрона можно сформулировать следующим образом.

Если n-й элемент $\mathbf{x}(n)$ обучающего множества корректно классифицирован с помощью весовых коэффициентов $\mathbf{w}(n)$, вычисленных на n-м шаге алгоритма, то вектор весов не корректируется, и действует следующее правило:

$$\mathbf{w}(n+1) = \mathbf{w}(n), \text{ если } \mathbf{w}^T\mathbf{x}(n) > 0 \text{ и } \mathbf{x}(n) \in \mathbf{C}_1,$$

$$\mathbf{w}(n+1) = \mathbf{w}(n), \text{ если } \mathbf{w}^T\mathbf{x}(n) \leq 0 \text{ и } \mathbf{x}(n) \in \mathbf{C}_2.$$

В противном случае вектор весов перцептрона подвергается коррекции в соответствии со следующим правилом:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n), \text{ если } \mathbf{w}^T(n)\,\mathbf{x}(n) > 0 \text{ и } \mathbf{x}(n) \in \mathbf{C}_2,$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n), \text{ если } \mathbf{w}^T(n)\,\mathbf{x}(n) \leq 0 \text{ и } \mathbf{x}(n) \in \mathbf{C}_1, \quad (2.14)$$

где интенсивность настройки вектора весов на шаге n определяется параметром скорости обучения $\eta(n)$.

Если $\eta(n) = \eta > 0$, где η — константа, вышеописанный алгоритм называется правилом адаптации с фиксированным приращением.

Ниже мы сначала докажем сходимость правила адаптации с фиксированным приращением для $\eta = 1$. Само значение η не играет особой роли, если оно положительно. Значение параметра $\eta \neq 1$ обеспечивает масштабирование образов, не влияющее на их разделимость. Случай с переменным коэффициентом $\eta(n)$ будет рассмотрен позже.

В приведенном доказательстве считается, что в начале процесса обучения вектор весовых коэффициентов равен нулю, $\mathbf{w}(0) = 0$. Если для $n = 1, 2, ...$, $\mathbf{w}^T(n)\,\mathbf{x}(n) < 0$, а входной вектор $\mathbf{x}(n)$ принадлежит подмножеству $\mathbf{X}_1$, то перцептрон некорректно классифицирует векторы $\mathbf{x}(1)$, $\mathbf{x}(2)$, ..., т. е. условие (2.13) не выполняется. Следовательно, для $\eta(n) = 1$ можно использовать вторую строку правила (2.14).

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mathbf{x}(n) \text{ для } \mathbf{x}(n) \in \mathbf{C}_1. \tag{2.15}$$

Поскольку начальное состояние $\mathbf{w}(0) = 0$, то данное уравнение для $\mathbf{w}(n+1)$ можно решить итеративно и получить следующий результат:

$$\mathbf{w}(n+1) = \mathbf{x}(1) + \mathbf{x}(2) + ... + \mathbf{x}(n). \tag{2.16}$$

Так как по предположению классы $\mathbf{C}_1$ и $\mathbf{C}_2$ являются линейно разделимыми, то существует такое решение $\mathbf{w}_0$, при котором будет выполняться условие $\mathbf{w}^T(n)\,\mathbf{x}(n) > 0$ для векторов $\mathbf{x}(1)$, $\mathbf{x}(2)$,.., $\mathbf{x}(n)$, принадлежащих подмножеству $\mathbf{X}_1$. Для фиксированного решения $\mathbf{w}_0$ можно определить такое положительное число α что

$$\alpha = \min_{x(n)\in X_1} \mathbf{w}_0^T \mathbf{x}(n). \tag{2.17}$$

Умножая обе части уравнения (2.16) на вектор-строку $\mathbf{w}_0^T$, получим:

$$\mathbf{w}_0^T \mathbf{w}(n+1) = \mathbf{w}_0^T \mathbf{x}(1) + \mathbf{w}_0^T \mathbf{x}(2) + \ldots \mathbf{w}_0^T \mathbf{x}(n).$$

Используя определение α (2.17), получим:

$$\mathbf{w}_0^T \mathbf{w}(n+1) \geq n\alpha.$$

Теперь можно использовать неравенство Гучи — Шварца (Gauchi-Schwartz inequality). Для двух векторов, $\mathbf{w}_0$ и $\mathbf{w}(n+1)$, его можно записать следующим образом:

$$\|\mathbf{w}_0\|^2 \|\mathbf{w}(n+1)\|^2 \geq \left[\mathbf{w}_0^T \mathbf{w}(n+1)\right]^2,$$

где $\|\cdot\|^2$ — евклидова норма векторного аргумента; $\mathbf{w}_0^T \mathbf{w}(n+1)$ — скалярное произведение векторов. Учитывая, что $\left[\mathbf{w}_0^T \mathbf{w}(n+1)\right]^2 \geq n^2\alpha^2$, получим:

$$\|\mathbf{w}_0\|^2 \|\mathbf{w}(n+1)\|^2 \geq n^2\alpha^2$$

или

$$\left\|\mathbf{w}(n+1)\right\|^2 \geq \frac{n^2\alpha^2}{\left\|\mathbf{w}_0\right\|^2}. \tag{2.18}$$

Перепишем уравнение (2.15) в следующем виде:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{x}(k) \text{ для } k = 1, 2, ..., n \text{ и } \mathbf{x}(k) \in \mathbf{X}_1.$$

Вычисляя евклидову норму векторов в обеих частях уравнения, получим:

$$\left\|\mathbf{w}(k+1)\right\|^2 = \left\|\mathbf{w}(k)\right\|^2 + \left\|\mathbf{x}(k)\right\|^2 + 2\mathbf{w}^T(k)\mathbf{x}(k). \tag{2.19}$$

Если перцептрон некорректно классифицировал входной вектор $\mathbf{x}(k)$, принадлежащий подмножеству $\mathbf{X}_1$, то $\mathbf{w}^T(k)\mathbf{x}(k) < 0$ и из (2.19) получим выражение

$$\left\|\mathbf{w}(k+1)\right\|^2 - \left\|\mathbf{w}(k)\right\|^2 \leq \left\|\mathbf{x}(k)\right\|^2 \text{ для } k = 1, 2, ..., n.$$

Применяя эти неравенства последовательно для $k = 1, ..., n$ и учитывая изначальное допущение, что $\boldsymbol{w}(0) = 0$, приходим к неравенству

$$\left\|\mathbf{w}(k+1)\right\|^2 \leq \sum_{k=1}^{n}\left\|\mathbf{x}(k)\right\|^2 \leq n\beta, \tag{2.20}$$

где $\beta = \max_{\mathbf{x}(k)\in\mathbf{X}_1}\left\|\mathbf{x}(k)\right\|^2 > 0$.

Уравнение (2.20) показывает, что евклидова норма вектора весов $\mathbf{w}(n + 1)$ линейно возрастает с увеличением номера итерации n.

Результат, описываемый неравенством (2.20), при больших n вступает в противоречие с полученным ранее результатом (2.18). Следовательно, номер итерации n не может превышать некоторого значения $n_{\max}$, при котором неравенства (2.18) и (2.20) удовлетворяются со знаком равенства. Это значит, что число $n_{\max}$ должно быть решением уравнения

$$\frac{n_{\max}^2\alpha^2}{\left\|\mathbf{w}_0\right\|^2} = n_{\max}\beta.$$

Решая это уравнение, получим

$$n_{\max} = \frac{\beta\left\|\mathbf{w}_0\right\|^2}{\alpha^2}. \tag{2.21}$$

В предположении существования вектора решения $\mathbf{w}_0$ для $\eta(n) = 1$ и $\mathbf{w}(0) = 0$ процесс адаптации синаптических весов перцептрона должен прекращаться не позднее итерации n_{max}, причем согласно формулам (2.17), (2.21) и определению β решение для $\mathbf{w}_0$ и n_{max} не единственно.

Теорема сходимости для алгоритма обучения перцептрона с фиксированным приращением для перцептрона [5]

Пусть подмножества векторов обучения $\mathbf{X}_1$ и $\mathbf{X}_2$ линейно разделимы. Пусть входные сигналы поступают перцептрону только из этих подмножеств. Тогда алгоритм обучения перцептрона сходится после некоторого числа n_0 итераций в том смысле, что

$$\mathbf{w}(n_0) = \mathbf{w}(n_0 + 1) = \mathbf{w}(n_0 + 2) = \ldots$$

является вектором решения для $n_0 \le n_{max}$.

Теперь рассмотрим абсолютную процедуру адаптации однослойного перцептрона на основе коррекции ошибок, в которой $\eta(n)$ — переменная величина. В частности, пусть $\eta(n)$ — наименьшее целое число, для которого выполняется соотношение

$$\eta(n)\mathbf{x}^T(n)\mathbf{x}(n) > \left|\mathbf{w}^T(n)\mathbf{x}(n)\right|.$$

Согласно этой процедуре, если скалярное произведение $\mathbf{w}^T(n)\mathbf{x}(n)$ на шаге n имеет неверный знак, то $\mathbf{w}^T(n+1)\mathbf{x}(n)$ на итерации $n + 1$ будет иметь правильный знак. Таким образом, предполагается, что если знак произведения $\mathbf{w}^T(n)\mathbf{x}(n)$ некорректен, то можно изменить последовательность обучения для итерации $n + 1$, приняв $\mathbf{x}(n + 1) = \mathbf{x}(n)$. Другими словами, каждый из образов представляется перцептрону до тех пор, пока он не будет классифицирован корректно.

Заметим, что использование отличного от нулевого исходного состояния $\mathbf{w}(0)$ приводит к увеличению или уменьшению количества итераций, необходимых для сходимости, в зависимости от того, насколько близким окажется исходное состояние $\mathbf{w}(0)$ к решению $\mathbf{w}_0$. Однако независимо от исходного значения $\mathbf{w}(0)$ сходимость все равно будет обеспечена.

В табл. 2.2 представлен алгоритм сходимости перцептрона [11] в краткой форме. Символ $\mathrm{sgn}(v)$, использованный на третьем шаге для вычисления фактического отклика перцептрона, означает функцию вычисления знака, которая равна +1, если $v > 0$, и –1 в противном случае.

Таблица 2.2

Алгоритм сходимости перцептрона

Параметры	Переменные
Постановка задачи	
Данные	$\mathbf{x}(n) = [+1, x_1(n), ..., x_m(n)]^T$ — вектор-строка размерности $m + 1$; $\mathbf{w}(n) = [b(n), w_1(n), ..., w_m(n)]^T$ — вектор-строка размерности $m + 1$; $b(n)$ — порог; $y(n)$ — фактический отклик (дискретизированный); $d(n)$ — желаемый отклик; $0 < \eta \leqslant 1$ — параметр скорости обучения
Алгоритм решения	
1. Инициализация	Пусть $\mathbf{w}(0) = 0$. Последующие вычисления выполняются для шагов $n = 1, 2, ...$
2. Активация	На шаге n активируем перцептрон, используя вектор $\mathbf{x}(n)$ с вещественными компонентами и желаемый отклик $d(n)$
3. Вычисление фактического ответа	$y(n) = \text{sgn}(\mathbf{w}^T(n)\,\mathbf{x}(n))$
4. Адаптация вектора весов	Изменяем вектор весов перцептрона: $\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\left[d(n) - y(n)\right]\mathbf{x}(n)$, где $d(n) = \begin{cases} +1, & \text{если } x(n) \in \mathbf{C}_1 \\ -1, & \text{если } x(n) \in \mathbf{C}_2 \end{cases}$
5. Продолжение	Увеличиваем номер итерации n на единицу и возвращаемся к п. 2 алгоритма

Таким образом, дискретный отклик $y(n)$ перцептрона можно выразить в компактной форме:

$$y(n) = \text{sgn}(\mathbf{w}^T(n)\mathbf{x}(n)).$$

Алгоритм адаптации вектора весовых коэффициентов $\mathbf{w}(n)$ соответствует правилу обучения на основе коррекции ошибок

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\left[d(n) - y(n)\right]\mathbf{x}(n),$$

где η — параметр скорости обучения, а разность $d(n) - y(n)$ выступает в роли сигнала ошибки. Выбирая значение параметра скорости обучения из диапазона [0; 1], следует учитывать два взаимоисключающих требования [11].

1. Усреднение предыдущих входных сигналов, обеспечивающее устойчивость оценки вектора весов, требует малых значений η.

2. Быстрая адаптация к реальным изменениям распределения процесса, отвечающего за формирование векторов входного сигнала x, требует больших значений η.

2.10. Взаимосвязь перцептрона и байесовского классификатора в гауссовой среде

Перцептрон имеет определенную связь с классической системой классификации образов, получившей название байесовского классификатора (Bayes classifier). В условиях гауссовой среды байесовский классификатор превращается в обычный линейный классификатор. Такую же форму имеет перцептрон. Однако линейная природа перцептрона не зависит от стохастических свойств среды. В этом разделе речь пойдет о взаимосвязи перцептрона и байесовского классификатора, что позволит глубже изучить природу и работу перцептрона.

Байесовский классификатор

В байесовской процедуре проверки гипотез (hypothesis testing procedure) минимизируется средний риск, который обозначается символом **R**. Для задачи двух классов ($\mathbf{C}_1$ и $\mathbf{C}_2$) средний риск в [12] определяется следующим образом:

$$\mathbf{R} = c_{11}p_1\int_{\mathbf{X}_1} f_{\mathbf{X}}\left(\mathbf{x}|C_1\right)d\mathbf{x} + c_{22}p_2\int_{\mathbf{X}_2} f_{\mathbf{X}}\left(\mathbf{x}|C_2\right)d\mathbf{x} +$$

$$+ c_{21}p_1\int_{\mathbf{X}_2} f_{\mathbf{X}}\left(\mathbf{x}|C_1\right)d\mathbf{x} + c_{12}p_2\int_{\mathbf{X}_1} f_{\mathbf{X}}\left(\mathbf{x}|C_2\right)d\mathbf{x},$$

где p_i — априорная вероятность того, что вектор наблюдения **x** (представляющий реализацию случайного вектора **X**) принадлежит подпространству $\mathbf{X}_i$ при $i = 1, 2$ и $p_1 + p_2 = 1$; c_{ij} — стоимость решения в пользу класса $\mathbf{C}_i$, представленного подпространством $\mathbf{X}_i$, когда истинным является класс $\mathbf{C}_j$ (т. е. вектор наблюдения принадлежит подпространству $\mathbf{X}_j$) при $(i, j) = 1, 2$; $f_X(\mathbf{x}|\mathbf{C}_i)$ — функция плотности условной веро-

ятности случайного вектора **X** при условии, что вектор наблюдения **x** принадлежит подпространству $\mathbf{X}_i$ для $i = 1, 2$.

Первые два слагаемых в правой части уравнения (2.22) представляют корректные решения (т. е. корректные классификации), а вторая пара слагаемых — некорректные (т. е. ошибки классификации). Каждое решение взвешивается произведением двух факторов: стоимости принятия решения и относительной частоты его принятия (т. е. априорной вероятности).

Целью является нахождение стратегии минимизации среднего риска. В процессе принятия решения каждому вектору наблюдения **x** из пространства **X** должно быть сопоставлено какое-либо из подпространств — $\mathbf{X}_1$ или $\mathbf{X}_2$: $\mathbf{X} = \mathbf{X}_1 + \mathbf{X}_2$. Соответственно, выражение (2.22) можно переписать в эквивалентной форме:

$$\mathbf{R} = c_{11} p_1 \int_{\mathbf{X}_1} f_{\mathbf{X}}\left(\mathbf{x}|C_1\right) d\mathbf{x} + c_{22} p_2 \int_{\mathbf{X}-\mathbf{X}_1} f_{\mathbf{X}}\left(\mathbf{x}|C_2\right) d\mathbf{x} +$$

$$+ c_{21} p_1 \int_{\mathbf{X}-\mathbf{X}_1} f_{\mathbf{X}}\left(\mathbf{x}|C_1\right) d\mathbf{x} + c_{12} p_2 \int_{\mathbf{X}_1} f_{\mathbf{X}}\left(\mathbf{x}|C_2\right) d\mathbf{x}, \qquad (2.23)$$

где $c_{11} < c_{21} < c_{12} < c_{22}$. Принимая во внимание, что

$$\int_{\mathbf{X}} f_{\mathbf{X}}\left(\mathbf{x}\,|\,C_1\right) d\mathbf{x} = \int_{\mathbf{X}} f_{\mathbf{X}}\left(\mathbf{x}\,|\,C_2\right) d\mathbf{x} = 1,$$

уравнение (2.23) можно свести к выражению

$$\mathbf{R} = c_{21} p_1 + c_{22} p_2 + c_{11} p_1 \int_{\mathbf{X}_1} \left[p_2 \left(c_{12} - c_{22}\right) f_{\mathbf{X}}\left(\mathbf{x}|C_2\right) - p_1 \left(c_{21} - c_{11}\right) f_{\mathbf{X}}\left(\mathbf{x}|C_1\right) \right] d\mathbf{x},$$

где первые два слагаемых представляют собой фиксированную стоимость. Теперь для минимизации среднего риска **R** можно вывести следующую стратегию оптимальной классификации.

1. Чтобы интеграл вносил отрицательный вклад в значение риска **R**, все значения вектора наблюдения **x**, для которых подынтегральное выражение является отрицательным, должны быть отнесены к подпространству $\mathbf{X}_1$ (т. е. к классу $\mathbf{C}_1$).

2. Чтобы интеграл вносил положительный вклад в значение риска **R**, все значения вектора наблюдения **x**, для которых подынтегральное выражение является положительным, должны быть исключены из подпространства $\mathbf{X}_1$ (т. е. отнесены к классу $\mathbf{C}_2$).

3. Значения **x**, при которых подынтегральное выражение равно нулю, не влияют на риск **R**. Их можно отнести к любому классу произвольным образом. В данном случае будем относить их к подпространству $\mathbf{X}_2$ (т. е. к классу $\mathbf{C}_2$).

Байесовский классификатор можно описать следующим образом.

Если выполняется условие $p_1(c_{21}-c_{11})f_{\mathbf{X}}(\mathbf{x}|C_1) > p_2(c_{12}-c_{22})f_{\mathbf{X}}(\mathbf{x}|C_2)$, то вектор наблюдения **x** следует относить к подпространству $\mathbf{X}_1$ (т. е. к классу $\mathbf{C}_1$), в противном случае — к подпространству $\mathbf{X}_2$ (т. е. к классу $\mathbf{C}_2$).

Для упрощения изложения введем следующие обозначения:

$$\Lambda(\mathbf{x}) = \frac{f_{\mathbf{X}}(\mathbf{x}|C_1)}{f_{\mathbf{X}}(\mathbf{x}|C_2)}, \tag{2.24}$$

$$\xi = \frac{p_2(c_{12}-c_{22})}{p_1(c_{21}-c_{11})}. \tag{2.25}$$

Величина $\Lambda(\mathbf{x})$, являющаяся частным двух функций плотности условной вероятности, называется отношением правдоподобия. Величина ξ называется пороговым значением процедуры проверки. Заметим, что обе эти величины всегда положительны. В терминах этих величин байесовский классификатор можно переопределить следующим образом.

Если для вектора наблюдения **x** отношение правдоподобия $\Lambda(\mathbf{x})$ превышает пороговый уровень ξ, то вектор **x** принадлежит классу $\mathbf{C}_1$, в противном случае — классу $\mathbf{C}_2$.

На рис. 2.9, *а* представлена блочная диаграмма байесовского классификатора. Его свойства:

1. Обработка данных в байесовском классификаторе ограничена исключительно вычислением отношения правдоподобия $\Lambda(\mathbf{x})$.

2. Эти вычисления полностью инвариантны по отношению к значениям априорной вероятности и стоимости, назначенным в процессе принятия решения. Эти значения влияют на величину порога ξ.

С вычислительной точки зрения более удобно работать с логарифмом отношения правдоподобия, а не с самим коэффициентом. К этому заключению приходим по двум причинам. Во-первых, логарифм является монотонной функцией. Во-вторых, значения $\Lambda(\mathbf{x})$ и ξ всегда положительны. Исходя из этого, байесовский классификатор можно реализовать в эквивалентной форме, показанной на рис. 2.9, *б*. Такой

подход называют логарифмическим критерием отношения правдоподобия.

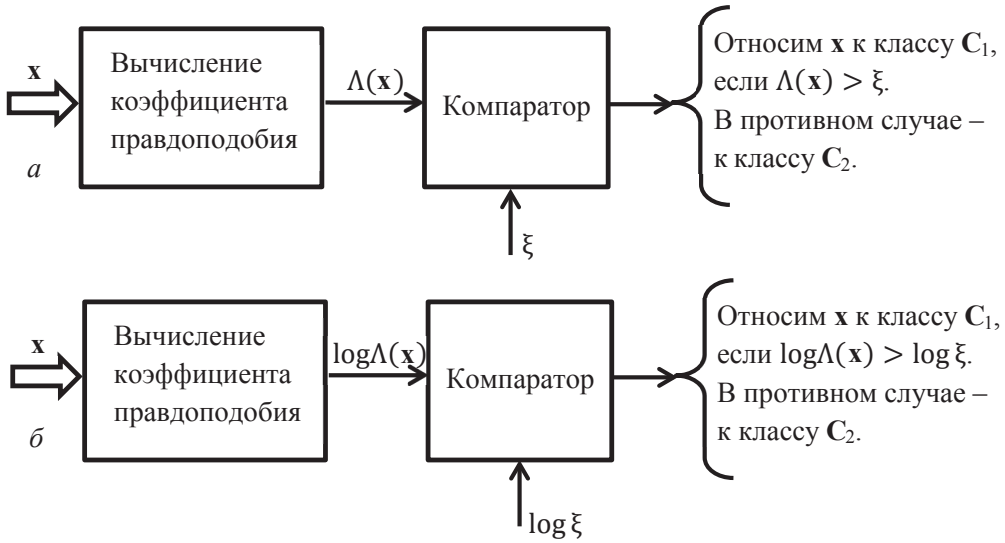


Рис. 2.9. Две эквивалентные реализации байесовского классификатора: на основе отношения правдоподобия (*а*) и его логарифма (*б*)

Байесовский классификатор и распределение Гаусса

Рассмотрим частный случай задачи классификации на два класса, в котором случайная величина имеет распределение Гаусса. Среднее значение случайного вектора **X** зависит от того, какому классу принадлежат его реализации — $\mathbf{C}_1 + \mathbf{C}_2$, однако матрица ковариации **X** остается одной и той же для обоих классов. Таким образом, можно записать следующее:

Класс $\mathbf{C}_1$

$$E[\mathbf{X}] = \mu_1,$$

$$E[(\mathbf{X} - \mu_1)(\mathbf{X} - \mu_1)^T] = \mathbf{C}.$$

Класс $\mathbf{C}_2$

$$E[\mathbf{X}] = \mu_2,$$

$$E[(\mathbf{X} - \mu_2)(\mathbf{X} - \mu_2)^T] = \mathbf{C}.$$

Матрица ковариации $\mathbf{C}$ не является диагональной, а это значит, что образы классов $\mathbf{C}_1$ и $\mathbf{C}_2$ коррелированы. Предполагается, что матрица $\mathbf{C}$ является несингулярной, поэтому существует обратная матрица $\mathbf{C}^{-1}$.

Используя эти соглашения, функцию плотности условной вероятности $\mathbf{X}$ можно представить в следующем виде:

$$f_{\mathbf{X}}\left(\mathbf{x}|\mathbf{C}_i\right)=\frac{\exp\left(-1/2\left(x-\mu_i\right)^T\mathbf{C}^{-1}\left(x-\mu_i\right)\right)}{\left(2\pi\right)^{m/2}\left(\det\left(\mathbf{C}\right)\right)^{1/2}}, \tag{2.26}$$

где m — размерность вектора наблюдения $\mathbf{x}$, $i = 1, 2$.

Введем следующие предположения.

1. Вероятность принадлежности образа обоим классам — $\mathbf{C}_1$ и $\mathbf{C}_2$ — одинакова, т. е. $p_1 = p_2 = 1/2$.

2. Ошибка классификации имеет постоянную стоимость, а корректная классификация стоимости не имеет: $c_{12} = c_{21}$ и $c_{11} = c_{22} = 0$.

Подставляя (2.26) в (2.24) и вычисляя натуральный логарифм, можно построить байесовский классификатор для двух классов.

$$\begin{aligned}\log\Lambda\left(\mathbf{x}\right)&=-1/2\left(\mathbf{x}-\mu_1\right)^T\mathbf{C}^{-1}\left(\mathbf{x}-\mu_1\right)+1/2\left(\mathbf{x}-\mu_1\right)^T\mathbf{C}^{-1}\left(\mathbf{x}-\mu_1\right)=\\&=\left(\mu_1-\mu_2\right)^T\mathbf{C}^{-1}\mathbf{x}+1/2\left(\mu_2^T\mathbf{C}^{-1}\mu_2-\mu_1^T\mathbf{C}^{-1}\mu_1\right).\end{aligned} \tag{2.27}$$

Применяя принятые предположения к выражению (2.25), приходим к соотношению

$$\log\xi=0. \tag{2.28}$$

Выражения (2.27) и (2.28) свидетельствуют о том, что байесовский классификатор для данной задачи является линейным классификатором, описываемым соотношением

$$y=\mathbf{w}_T\mathbf{x}+b, \tag{2.29}$$

где

$$y=\log\Lambda\left(\mathbf{x}\right),$$

$$\mathbf{w}=\mathbf{C}^{-1}\left(\mu_1-\mu_2\right),$$

$$b=1/2\left(\mu_2^T\mathbf{C}^{-1}\mu_2-\mu_1^T\mathbf{C}^{-1}\mu_1\right).$$

Более точно классификатор представляет собой линейный сумматор с вектором весов $\mathbf{w}$ и порогом b (рис. 2.10).

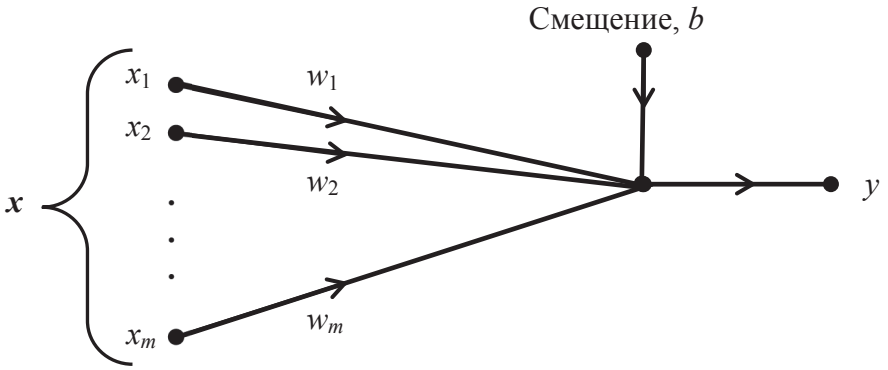


Рис. 2.10. Граф передачи сигнала байесовского классификатора в гауссовой среде

Теперь с учетом (2.29) логарифмический критерий отношения правдоподобия для задачи классификации на два класса можно описать следующим образом.

Если выходной сигнал у линейного сумматора, содержащего порог b, положителен, вектор наблюдения $\mathbf{x}$ относится к классу $\mathbf{C}_1$, в противном случае — к классу $\mathbf{C}_2$.

Описанный выше байесовский классификатор в гауссовой среде аналогичен перцептрону, так как оба классификатора являются линейными. Однако между ними существует ряд мелких и важных различий [11].

1. Перцептрон работает при условии, что классифицируемые образы линейно разделимы. Распределение Гаусса в контексте байесовского классификатора предполагает их пересечение, что исключает их линейную разделимость. Границы этого пересечения определяются посредством векторов μ_1 и μ_2 и матрицы ковариации $\mathbf{C}$. Природа пересечения проиллюстрирована на рис. 2.11 для частного случая скалярной случайной переменной (т. е. размерность $m = 1$). Если входные сигналы неразделимы и их функции распределения пересекаются так, как показано на рисунке, алгоритм обучения перцептрона не сходится, так как границы областей решения могут постоянно смещаться.

2. Байесовский классификатор минимизирует вероятность ошибки классификации. Эта минимизация не зависит от пересечения между распределениями Гаусса двух классов. Например, в частном случае, показанном на рис. 2.11, байесовский классификатор всегда будет помещать границу областей решения в точку пересечения функций гауссова распределения классов $\mathbf{C}_1$ и $\mathbf{C}_2$.

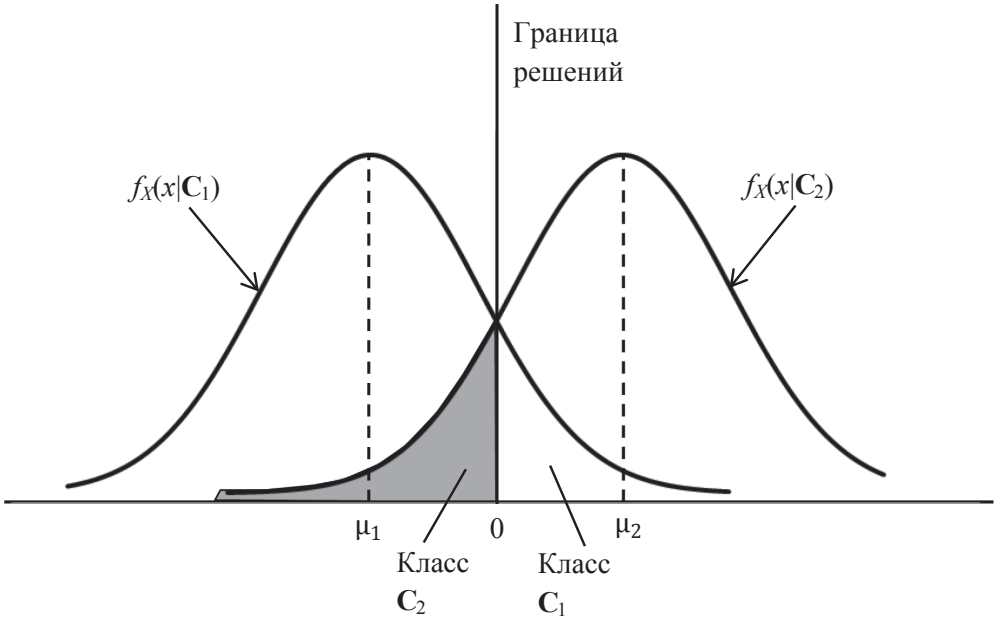


Рис. 2.11. Две пересекающиеся одномерные функции распределения Гаусса

3. Алгоритм работы перцептрона является непараметрическим, т. е. относительно формы рассматриваемых распределений никаких предварительных предположений не делается. Работа алгоритма базируется на коррекции ошибок, возникающих в точках пересечения функций распределения. Таким образом, перцептрон хорошо работает с входными сигналами, генерируемыми нелинейными физическими процессами, даже если их распределения не симметричны и не являются гауссовыми. В отличие от перцептрона байесовский классификатор является параметрическим. Он предполагает, что распределения случайных величин являются гауссовыми, а это может ограничить область применения классификатора.

4. Алгоритм сходимости перцептрона является адаптивным и простым для реализации. Его требования к хранению информации ограничиваются множеством синаптических весов и порогов. Архитектура байесовского классификатора является фиксированной; ее можно сделать адаптивной только за счет усиления требований к хранению информации и усложнения вычислений.

2.11. Резюме и обсуждение

Перцептрон и адаптивный фильтр на основе алгоритма LMS связаны самым естественным образом, что проявляется в процессе модификации синаптических связей. Более того, они представляют различные реализации однослойного перцептрона, обучаемого на основе коррекции ошибок. Термин «однослойный» здесь используется для того, чтобы подчеркнуть, что в обоих случаях вычислительный слой состоит из единственного нейрона. Однако перцептрон и алгоритм минимизации среднеквадратической ошибки отличаются друг от друга в некоторых фундаментальных аспектах.

1. Алгоритм минимизации среднеквадратической ошибки использует линейный нейрон, в то время как перцептрон основан на формальной модели нейрона Мак-Каллока — Питца.

2. Процесс обучения перцептрона завершается за конечное число итераций. Алгоритм минимизации среднеквадратической ошибки предполагает непрерывное обучение, т. е. обучение происходит до тех пор, пока выполняется обработка сигнала. Этот процесс никогда не останавливается.

Модель Мак-Каллока — Питца накладывает жесткие ограничения на форму нелинейности нейрона. Устойчивое принятие решений перцептроном не зависит от вида нелинейности нейрона [13], если эти жесткие ограничения заменить сигмоидальной нелинейностью. Таким образом, можно формально утверждать, что при использовании модели нейрона, которая состоит из линейного сумматора и нелинейного элемента, независимо от формы используемой нелинейности однослойный перцептрон будет выполнять классификацию образов только для линейно разделимых классов.

В завершение краткая историческая справка. Перцептрон и алгоритм минимизации среднеквадратической ошибки появились приблизительно в одно и то же время — в конце 1950-х годов. Алгоритм минимизации среднеквадратической ошибки достойно выдержал испытание временем. Он хорошо зарекомендовал себя в качестве «рабочей лошадки» адаптивной обработки сигналов благодаря своей эффективности и простоте реализации. Перцептрон Розенблатта имеет более интересную историю.

Первая критика перцептрона Розенблатта появилась в [14], где утверждалось, что перцептрон Розенблатта не способен к обобщению даже в задаче «исключающего ИЛИ» (двоичной четности), не говоря уже о более общих абстракциях. Вычислительные ограничения перцептрона Розенблатта были математически обоснованы в знаменитой книге Минского и Пейперта «Перцептроны» [15, 16]. После блестящего и в высшей степени подробного математического анализа перцептрона Минский и Пейперт доказали, что перцептрон в определении Розенблатта внутренне не способен на глобальные обобщения на базе локальных примеров обучения. В последней главе своей книги Минский и Пейперт высказали предположение, что недостатки перцептрона Розенблатта остаются в силе и для его вариаций, в частности для многослойных нейронных сетей. Приведем цитату из [15] (раздел 13.2).

«Перцептрон достоин изучения вопреки (и даже благодаря) своим ограничениям. Он имеет множество свойств, заслуживающих внимания: линейность, интригующую теорему обучения, образцовую простоту в смысле организации параллельных вычислений. Нет оснований полагать, что некоторые из его преимуществ сохранятся в многослойной версии. Тем не менее, мы рассматриваем его как важный объект исследований, для того чтобы обосновать (или отвергнуть) наш интуитивный приговор: расширение перцептрона в сторону многослойных систем потенциально безрезультатно.»

Это заключение в основном и несет ответственность за возникновение серьезных сомнений в вычислительных возможностях не только перцептрона, но и нейронных сетей в целом вплоть до середины 1980-х годов.

Однако история показала, что предположение, высказанное Минским и Пейпертом, было бездоказательным. В настоящее время существует ряд усовершенствованных форм нейронных сетей, которые с вычислительной точки зрения мощнее перцептрона Розенблатта. Например, многослойные перцептроны, обучаемые с помощью алгоритма обратного распространения ошибки, сети на основе радиальных базисных функций и машины опорных векторов преодолевают вычислительные ограничения однослойного перцептрона различными способами.

3. Многослойный перцептрон

3.1. Введение

Эта глава посвящена важному классу нейронных сетей — многослойным сетям прямого распространения. Обычно сеть состоит из множества сенсорных элементов (входных узлов или узлов источника), которые образуют входной слой; одного или нескольких скрытых слоев (hidden layer) вычислительных нейронов и одного выходного слоя (output layer) нейронов. Входной сигнал распространяется по сети в прямом направлении, от слоя к слою. Подобную архитектуру сетей обычно называют многослойными перцептронами (multilayer perceptron), которые представляют собой обобщение однослойного перцептрона. Многослойные перцептроны (МСП) успешно применяются для решения разнообразных задач. При этом выполняется обучение с учителем с помощью алгоритма обратного распространения ошибки (error back-propagation algorithm), который основывается на коррекции ошибок (error-correction learning rule). Его можно рассматривать как обобщение столь же популярного алгоритма адаптивной фильтрации — аналога алгоритма минимизации среднеквадратической ошибки (LMS). Термин «обратное распространение» активно используется после 1986 года, когда он был популяризован в известной книге [17]. Появление алгоритма обратного распространения стало знаковым событием в области развития нейронных сетей, так как он реализует вычислительно эффективный (computationally efficient) метод обучения многослойного перцептрона, частично решивший проблемы, обозначенные в [16]. Обучение методом обратного распространения ошибки предполагает два прохода по всем слоям сети: прямой и обратный. При прямом проходе (forward pass) образ (входной вектор) подается на сенсорные узлы сети, после чего распространяет-

ся по сети от слоя к слою. В результате генерируется набор выходных сигналов, который и является реакцией сети на данный входной образ. Во время прямого прохода все синаптические веса сети фиксированы. Во время обратного прохода (backward pass) синаптические веса настраиваются в соответствии с правилом коррекции ошибок, а именно: фактический выход сети вычитается из желаемого (целевого) отклика, в результате чего формируется сигнал ошибки (error signal). Этот сигнал впоследствии распространяется по сети в направлении, обратном направлению синаптических связей. Отсюда и происходит название алгоритма. Синаптические веса настраиваются с целью максимального приближения выходного сигнала сети к желаемому в статистическом смысле. Алгоритм обратного распространения ошибки часто называют упрощенно — алгоритмом обратного распространения (back-propagation algorithm). Процесс обучения, реализуемый этим алгоритмом, называется обучением на основе обратного распространения (back-propagation learning).

Многослойные перцептроны имеют три отличительных признака.

1. Каждый нейрон сети имеет нелинейную функцию активации (nonlinear activation function). Данная нелинейная функция является гладкой (т. е. всюду дифференцируемой), в отличие от жесткой пороговой функции, используемой в перцептроне Розенблатта. Самой популярной формой функции, удовлетворяющей этому требованию, является сигмоидальная (sigmoidal nonlinearity), определяемая логистической функцией (logistic function)

$$y_i = \frac{1}{1+\exp(-v_i)}$$

где v_i — индуцированное локальное поле (т. е. взвешенная сумма всех синаптических входов плюс пороговое значение) нейрона i; y_j — выход нейрона. Наличие нелинейности играет очень важную роль, так как в противном случае отображение «вход-выход» сети можно свести к обычному однослойному перцептрону. Более того, использование логистической функции мотивировано биологически, так как в ней учитывается восстановительная фаза реального нейрона. Сигмоидальные функции получили свое название благодаря форме своего графика в виде буквы *S*. В [18] исследованы два класса сигмоид: *простые сигмоиды* — произвольные асимптотически ограниченные и строго монотонные функции одной переменной и *гиперболические сигмоиды* — полное

подмножество простых сигмоид, являющихся обобщением функции гиперболического тангенса.

2. Сеть содержит один или несколько слоев скрытых нейронов, не являющихся частью входа или выхода сети. Эти нейроны позволяют сети обучаться решению сложных задач, последовательно извлекая наиболее важные признаки из входного образа (вектора).

3. Сеть обладает высокой степенью связности, реализуемой посредством синаптических соединений. Изменение уровня связности сети требует изменения множества синаптических соединений или их весовых коэффициентов.

Комбинация всех этих свойств наряду со способностью к обучению на собственном опыте обеспечивает вычислительную мощность многослойного перцептрона. Однако эти же качества являются причиной неполноты современных знаний о поведении такого рода сетей. Во-первых, распределенная форма нелинейности и высокая связность сети существенно усложняют теоретический анализ многослойного перцептрона. Во-вторых, наличие скрытых нейронов делает процесс обучения более трудным для визуализации. Именно в процессе обучения необходимо определить, какие признаки входного сигнала следует представлять скрытыми нейронами. Тогда процесс обучения становится еще более сложным, поскольку поиск должен выполняться в очень широкой области возможных функций, а выбор должен производиться среди альтернативных представлений входных образов [19].

3.2. Общая информация о МСП

На рис. 3.1 показан архитектурный граф многослойного перцептрона с двумя скрытыми слоями и одним выходным слоем. Показанная на рисунке сеть является полносвязной, что характерно для многослойного перцептрона общего вида, в котором каждый нейрон любого слоя сети связан со всеми нейронами (узлами) предыдущего слоя. Сигнал передается по сети исключительно *в прямом направлении, слева направо, от слоя к слою.* Для этого типа сети выделяют два типа сигналов.

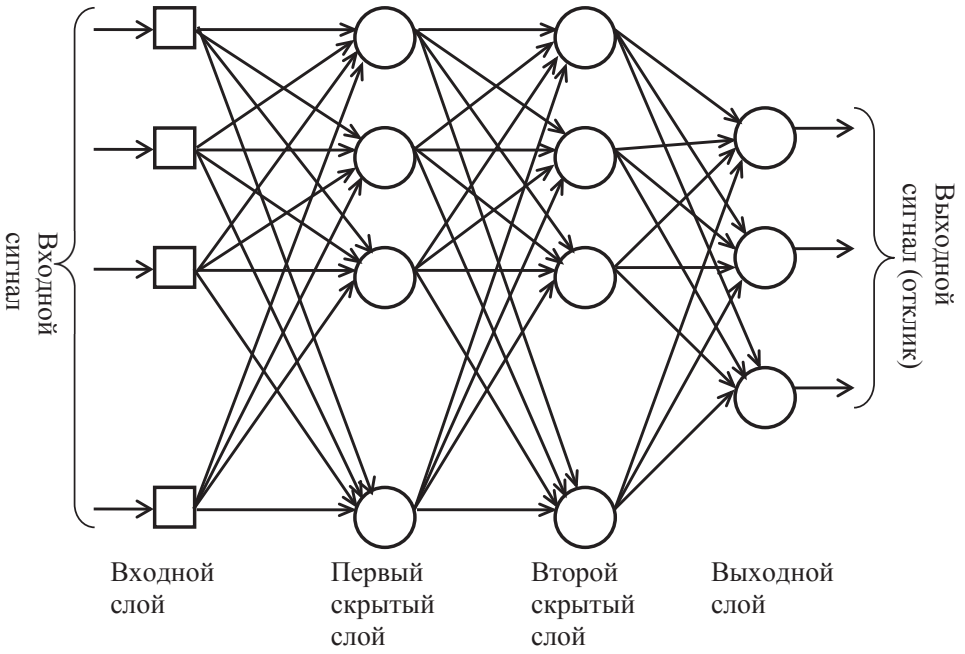


Рис. 3.1. Архитектурный граф многослойного перцептрона с двумя скрытыми слоями

1. Функциональный сигнал (function signal). Это входной сигнал (стимул), поступающий в сеть и передаваемый вперед от нейрона к нейрону по всей сети. Такой сигнал достигает конца сети в виде выходного сигнала. Название сигнала связано с тем, что его предназначение — выполнение некоторой функции, и с тем, что в каждом нейроне, через который передается этот сигнал, вычисляется некоторая функция с учетом весовых коэффициентов.

2. Сигнал ошибки (error signal), берущий свое начало на выходе сети и распространяющийся в обратном направлении (от слоя к слою). Он вычисляется каждым нейроном сети на основе функции ошибки, представленной в той или иной форме.

На рис. 3.2 показан фрагмент многослойного перцептрона. Выходные нейроны (вычислительные узлы) составляют выходной слой сети. Остальные нейроны (вычислительные узлы) относятся к скрытым слоям. Первый скрытый слой получает данные из входного слоя, составленного из сенсорных элементов (входных узлов). Результирующий сигнал первого скрытого слоя, в свою очередь, поступает на следующий скрытый слой и т. д. до самого конца сети.

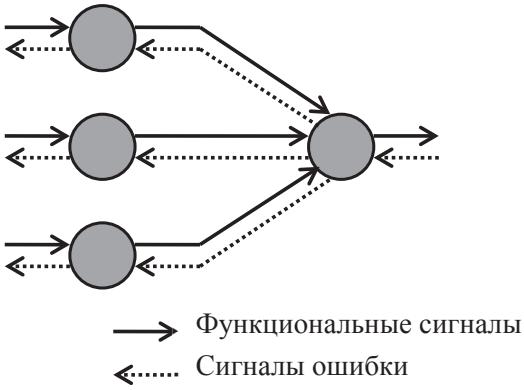


Рис. 3.2. Направление двух основных потоков сигнала для многослойного перцептрона: прямое распространение функционального сигнала и обратное распространение сигнала ошибки

Любой скрытый или выходной нейрон многослойного перцептрона может выполнять два типа вычислений.

1. Вычисление функционального сигнала на выходе нейрона, реализуемое в виде непрерывной нелинейной функции от входного сигнала и синаптических весов, связанных с данным нейроном.

2. Вычисление оценки вектора градиента (т. е. градиента поверхности ошибки по синаптическим весам, связанным с входами данного нейрона), необходимого для обратного прохода через сеть.

Для упрощения понимания математических выкладок вывода алгоритма обратного распространения приведем полный список используемых обозначений.

— Индексы i, j и k относятся к различным нейронам сети. Когда сигнал проходит по сети слева направо, считается, что нейрон j находится на один слой правее нейрона i, а нейрон k — еще на один слой правее нейрона j, если последний принадлежит скрытому слою.
— Итерация (такт времени) n соответствует n-му обучающему образу (примеру), поданному на вход сети.
— $\mathbf{E}(n)$ — текущая сумма квадратов ошибок (или энергия ошибки) на итерации n. Среднее значение $\mathbf{E}(n)$ по всем значениям n (т. е. по всему обучающему множеству) называется средней энергией ошибки $\mathbf{E}_{av}$.
— $e_j(n)$ — сигнал ошибки на выходе нейрона j на итерации n.
— $d_j(n)$ — желаемый отклик нейрона j, используется для вычисления $e_j(n)$.

— $y_j(n)$ — функциональный сигнал, генерируемый на выходе нейрона j на итерации n.
— $w_{ji}(n)$ — синаптический вес, связывающий выход нейрона i со входом нейрона j на итерации n. Коррекция, применяемая к этому весу на шаге n, обозначается $\Delta w_{ji}(n)$.
— $v_j(n)$ — индуцированное локальное поле (т. е. взвешенная сумма всех синаптических входов плюс порог) нейрона j на итерации n. Это значение передается в функцию активации, связанную с нейроном j.
— $\varphi_j(\cdot)$ — функция активации, соответствующая нейрону j и описывающая нелинейную взаимосвязь входного и выходного сигналов этого нейрона.
— b_j — порог, применяемый к нейрону j. Его влияние представлено синапсом с весом $w_{jo} = b_j$, соединенным с фиксированным входным сигналом, равным +1.
— $x_i(n)$ — i-й элемент входного вектора.
— $o_k(n)$ — k-й элемент выходного вектора (образа).
— η — параметр скорости (интенсивности) обучения.
— m_l — размерность (количество узлов) слоя l многослойного перцептрона; $l = 1, 2, ..., L$, где L — «глубина» сети. Таким образом, символ m_0 означает размерность входного слоя; m_1 — первого скрытого слоя; m_L — выходного слоя. Для описания размерности выходного слоя будет также использоваться символ M.

3.3. Алгоритм обратного распространения ошибки

Сигнал ошибки выходного нейрона j на итерации n определяется соотношением

$$e_j(n) = d_j(n) - y_j(n). \tag{3.1}$$

Текущее значение энергии ошибки нейрона j определим как $\frac{1}{2}e_j^2(n)$. Соответственно, текущее значение $\mathbf{E}(n)$ общей энергии ошибки сети вычисляется путем сложения величин $\frac{1}{2}e_j^2(n)$ по всем нейронам выходного слоя. Это «видимые» нейроны, для которых сигнал ошибки может быть вычислен непосредственно. Таким образом, можно записать:

$$\mathbf{E}(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n), \tag{3.2}$$

где множество C включает все нейроны выходного слоя сети. Пусть N — общее число образов в обучающем множестве (т. е. мощность этого множества). Энергия среднеквадратической ошибки в таком случае вычисляется как нормализованная по N сумма всех значений энергии ошибки $\mathbf{E}(n)$:

$$\mathbf{E}_{\mathbf{av}}(n) = \frac{1}{N}\sum_{n=1}^{N}\mathbf{E}(n). \tag{3.3}$$

Текущая энергия ошибки $\mathbf{E}(n)$, а значит и средняя энергия ошибки $\mathbf{E}_{\mathbf{av}}$ являются функциями всех свободных параметров (т. е. синаптических весов и значений порога) сети. Для данного обучающего множества энергия $\mathbf{E}_{\mathbf{av}}$ представляет собой функцию стоимости (cost function) — меру эффективности обучения. Целью процесса обучения является настройка свободных параметров сети с целью минимизации величины $\mathbf{E}_{\mathbf{av}}$. В процессе минимизации используется аппроксимация, аналогичная применяемой для вывода алгоритма LMS. В частности, рассмотрим простой метод обучения, в котором веса обновляются для каждого обучающего примера в пределах одной эпохи (epoch), т. е. всего обучающего множества. Настройка весов выполняется в соответствии с ошибками, вычисленными для каждого образа, представленного в сети. Арифметическое среднее отдельных изменений для весов сети по обучающему множеству может служить оценкой реальных изменений, произошедших в процессе минимизации функции стоимости $\mathbf{E}_{\mathbf{av}}$ на множестве обучения. Оценка качества этого критерия обсуждается ниже.

На рис. 3.3 изображен нейрон j, на который поступает поток сигналов от нейронов, расположенных в предыдущем слое. Индуцированное локальное поле $v_j(n)$, полученное на входе функции активации, связанной с данным нейроном, равно

$$v_j(n) = \sum_{i=0}^{m} w_{ij}(n)\, y_i(n), \tag{3.4}$$

где m — общее число входов (за исключением порога) нейрона j. Синаптический вес w_{j0} (соответствующий фиксированному входу $y_0 = +1$) равен порогу b_j, применяемому к нейрону j. Функциональный сигнал $y_j(n)$ на выходе нейрона j на итерации n равен

$$y_j(n) = \varphi_j\left(v_j(n)\right). \tag{3.5}$$

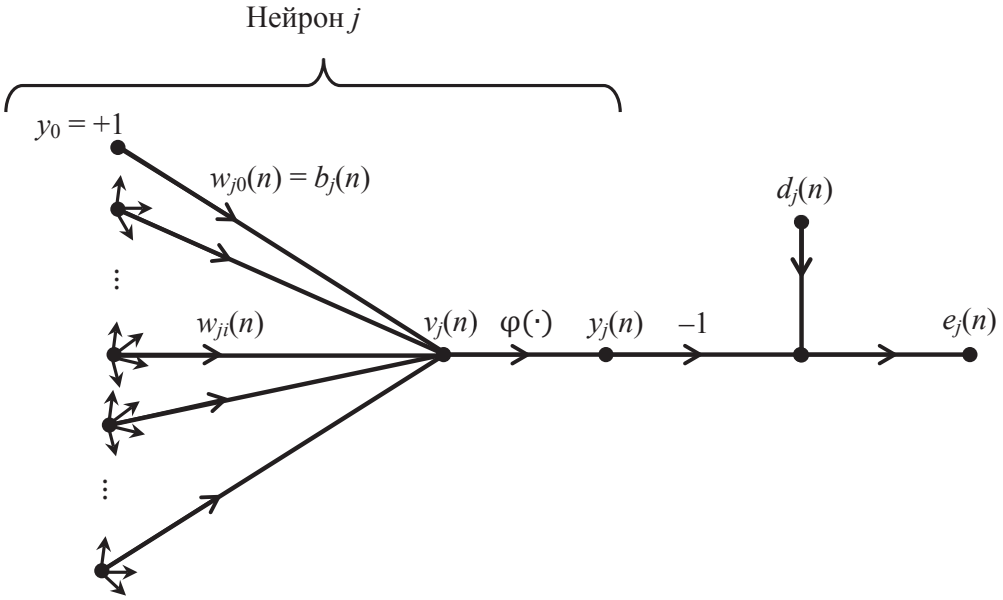


Рис. 3.3. Граф передачи сигнала в пределах нейрона *j*

Аналогично алгоритму LMS алгоритм обратного распространения состоит в применении к синаптическому весу $w_{ji}(n)$ коррекции $\Delta w_{ji}(n)$, пропорциональной частной производной $\partial \mathbf{E}(n) / \partial w_{ji}(n)$. В соответствии с правилом цепочки, или цепным правилом (chain rule), градиент можно представить следующим образом:

$$\frac{\partial \mathbf{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathbf{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial \mathbf{w}_{ji}(n)}. \tag{3.6}$$

Частная производная $\partial \mathbf{E}(n) / \partial w_{ji}(n)$ представляет собой фактор чувствительности (sensitivity factor), определяющий направление поиска в пространстве весов для синаптического веса $w_{ji}(n)$.

Дифференцируя обе части уравнения (3.2) по $e_j(n)$, получим:

$$\frac{\partial \boldsymbol{E}(n)}{\partial e_j(n)} = \boldsymbol{e}_j(n). \tag{3.7}$$

Дифференцируя обе части уравнения (3.1) по $y_j(n)$, получим:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1. \tag{3.8}$$

Затем, дифференцируя (3.5) по $v_j(n)$, получим:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'_j\left(v_j(n)\right), \tag{3.9}$$

где штрих справа от имени функции обозначает дифференцирование по аргументу. И, наконец, дифференцируя (3.4) по $w_{ji}(n)$, получим:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_j(n). \tag{3.10}$$

Подставляя результаты (3.7)–(3.10) в (3.6), получим:

$$\frac{\partial \mathbf{E}(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'_j\left(v_j(n)\right)y_j(n). \tag{3.11}$$

Коррекция $\Delta w_{ji}(n)$, применяемая к $w_{ji}(n)$, определяется согласно дельта-правилу (delta rule):

$$\Delta w_{ji}(n) = -\eta\frac{\partial \mathbf{E}(n)}{\partial w_{ji}(n)}, \tag{3.12}$$

где η — параметр скорости обучения (learning-rate parameter) алгоритма обратного распространения. Использование знака «минус» в (3.12) связано с реализацией градиентного спуска (gradient descent) в пространстве весов (т. е. поиском направления изменения весов, уменьшающего значение энергии ошибки $\mathbf{E}(n)$). Следовательно, подставляя (3.11) в (3.12), получим:

$$\Delta w_{ji}(n) = \eta\delta_j(n)y_j(n), \tag{3.13}$$

где локальный градиент (local gradient) $\delta_j(n)$ определяется выражением

$$\delta_j(n) = -\frac{\partial \mathbf{E}(n)}{\partial v_j(n)} = -\frac{\partial \mathbf{E}(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi'_j\left(v_j(n)\right). \tag{3.14}$$

Локальный градиент указывает на требуемое изменение синаптического веса. В соответствии с (3.14) локальный градиент $\delta_j(n)$ выходного нейрона j равен произведению соответствующего сигнала ошибки $e_j(n)$ этого нейрона и производной $\varphi'_j(v_j(n))$ соответствующей функции активации.

Из выражений (3.13) и (3.14) видно, что ключевым фактором в вычислении величины коррекции $\Delta w_{ji}(n)$ весовых коэффициентов явля-

ется сигнал ошибки $e_j(n)$ нейрона j. В этом контексте можно выделить два различных случая, определяемых положением нейрона j в сети. В первом случае нейрон j является выходным узлом. Это довольно простой случай, так как для каждого выходного узла сети известен соответствующий желаемый отклик. Следовательно, вычислить сигнал ошибки можно с помощью простой арифметической операции. Во втором случае нейрон j является скрытым узлом. Однако даже если скрытый нейрон непосредственно недоступен, он несет ответственность за ошибку, получаемую на выходе сети. Вопрос состоит в том, как персонифицировать вклад (положительный или отрицательный) отдельных скрытых нейронов в общую ошибку. Эта проблема называется задачей назначения коэффициентов доверия (credit assignment problem) и решается с помощью метода обратного распространения сигнала ошибки по сети.

Случай 1. Нейрон j — выходной узел

Если нейрон j расположен в выходном слое сети, для него известен соответствующий желаемый отклик. Значит, с помощью выражения (3.1) можно определить сигнал ошибки $e_j(n)$ (см. рис. 3.3) и вычислить градиент $\delta_j(n)$ по формуле (3.14).

Случай 2. Нейрон j — скрытый узел

Если нейрон j расположен в скрытом слое сети, желаемый отклик для него неизвестен. Следовательно, сигнал ошибки скрытого нейрона должен рекурсивно вычисляться на основе сигналов ошибки всех нейронов, с которыми он непосредственно связан. Именно здесь алгоритм обратного распространения сталкивается с наибольшей сложностью. Рассмотрим ситуацию, изображенную на рис. 3.4, где представлен скрытый нейрон j. Согласно (3.14), локальный градиент $\delta_j(n)$ скрытого нейрона j можно переопределить следующим образом:

$$\delta_j(n) = -\frac{\partial \mathbf{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \mathbf{E}(n)}{\partial y_j(n)} \varphi_j'\left(v_j(n)\right), \qquad (3.15)$$

где j — скрытый нейрон, а для получения второго результата использовалась формула (3.9). Для вычисления частной производной $\partial \mathbf{E}(n) / \partial y_j(n)$ выполним некоторые преобразования. На рис. 3.4 видно, что

$$\mathbf{E}(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n), \tag{3.16}$$

где k — выходной нейрон. Соотношение (3.16) — это выражение (3.2), в котором индекс j заменен индексом k. Это сделано для того, чтобы избежать путаницы с индексом j, использованным ранее для скрытого нейрона. Дифференцируя (3.16) по функциональному сигналу $y_j(n)$, получим:

$$\frac{\partial \mathbf{E}(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)}. \tag{3.17}$$

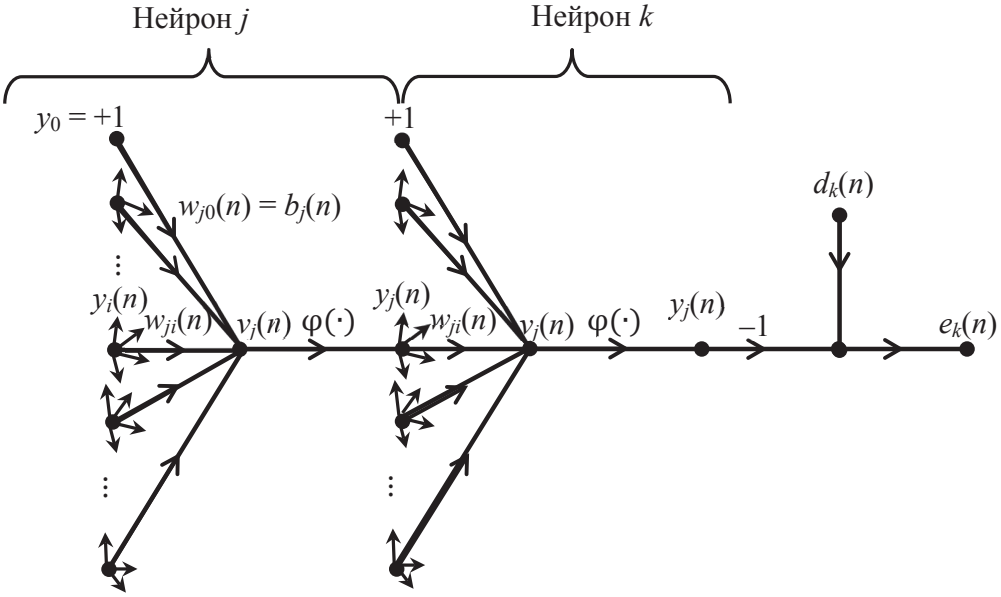


Рис. 3.4. Граф передачи сигнала, детально отражающий связь выходного нейрона k со скрытым нейроном j

Теперь к частной производной $\partial e_k(n)/\partial y_j(n)$ можно применить цепное правило и переписать (3.17) в эквивалентной форме:

$$\frac{\partial \mathbf{E}(n)}{\partial y_j(n)} = \sum_k e_k(n)\frac{\partial e_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)}. \tag{3.18}$$

Однако на рис. 3.4 видно, что для выходного нейрона k

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_j(v_k(n)). \tag{3.19}$$

Отсюда

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi'_k\left(v_k(n)\right). \tag{3.20}$$

На рис. 3.4 также видно, что индуцированное локальное поле нейрона k составляет

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n)\, y_j(n), \tag{3.21}$$

где m — общее число входов (за исключением порога) нейрона k. Здесь синаптический вес $w_{k0}(n)$ равен порогу b_k, приходящемуся на нейрон k, а соответствующий вход имеет фиксированное значение +1. Дифференцируя (3.21) по $y_j(n)$, получим:

$$\frac{\partial v_k(n)}{\partial v_j(n)} = w_{kj}(n). \tag{3.22}$$

Подставляя (3.20) и (3.22) в (3.18), получим:

$$\frac{\partial \mathbf{E}(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi'_k\left(v_k(n)\right) w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n), \tag{3.23}$$

где для получения второго результата использовано определение локального градиента (3.14), в котором индекс j заменен индексом k по уже упомянутым соображениям.

В заключение, подставляя выражение (3.23) в (3.15), получим формулу обратного распространения (back-propagation formula) для локального градиента $\delta_j(n)$ скрытого нейрона j:

$$\delta_j(n) = \varphi'_j\left(v_j(n)\right)\sum_k \delta_k(n) w_{kj}(n). \tag{3.24}$$

На рис. 3.5 графически представлена формула (3.24) в предположении, что выходной слой состоит из m_L нейронов.

Множитель $\varphi'_j(v_j(n))$, использованный в формуле (3.24) для вычисления локального градиента $\delta_j(n)$, зависит исключительно от функции активации, связанной со скрытым нейроном j. Второй множитель формулы (сумма по k) зависит от двух множеств параметров. Первое из них — $\delta_k(n)$ — требует знания сигналов ошибки $e_k(n)$ для всех нейронов слоя, находящегося правее скрытого нейрона j, которые непосредственно связаны с нейроном j (см. рис. 3.4). Второе множество — $w_{kj}(n)$ — состоит из синаптических весов этих связей.

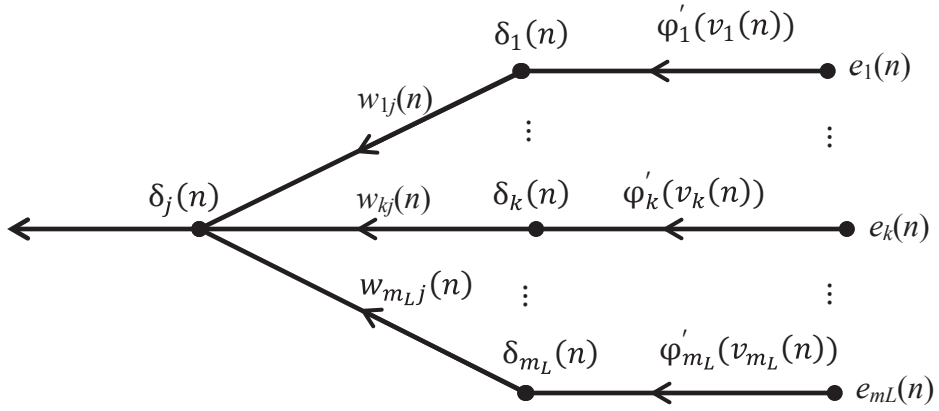


Рис. 3.5. Граф передачи сигнала для фрагмента системы, задействованного в обратном распространении сигналов ошибки

Теперь можно свести воедино все соотношения, которые мы вывели для алгоритма обратного распространения. Во-первых, коррекция $\Delta w_{ji}(n)$, применяемая к синаптическому весу, соединяющему нейроны i и j, определяется следующим дельта-правилом:

$$\begin{pmatrix}\text{коррекция}\\ \text{веса}\\ \Delta w_{ji}(n)\end{pmatrix}=\begin{pmatrix}\text{параметр}\\ \text{скорости обучения}\\ \eta\end{pmatrix}\cdot\begin{pmatrix}\text{локальный}\\ \text{градиент}\\ \delta_j(n)\end{pmatrix}\cdot\begin{pmatrix}\text{входной сигнал}\\ \text{нейрона } j\\ y_j(n)\end{pmatrix} \quad (3.25)$$

Во-вторых, значение локального градиента $\delta_j(n)$ зависит от положения нейрона в сети.

1. Если нейрон j — выходной, то градиент $\delta_j(n)$ равен произведению производной $\varphi'_j(v_j(n))$ на сигнал ошибки $e_j(n)$ для нейрона j (см. (3.14)).

2. Если нейрон j — скрытый, то градиент $\delta_j(n)$ равен произведению производной $\varphi'_j(v_j(n))$ на взвешенную сумму градиентов, вычисленных для нейронов следующего скрытого или выходного слоя, которые непосредственно связаны с данным нейроном j (см. (3.24)).

Два прохода вычислений

При использовании алгоритма обратного распространения различают два прохода, выполняемых в процессе вычислений. Первый проход называется прямым, второй — обратным. При прямом проходе

(forward pass) синаптические веса остаются неизменными во всей сети, а функциональные сигналы вычисляются последовательно, от нейрона к нейрону. Функциональный сигнал на выходе нейрона j вычисляется по формуле

$$y_j(n) = \varphi\left(v_j(n)\right), \tag{3.26}$$

где $v_j(n)$ — индуцированное локальное поле нейрона j; определяемое выражением

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n)\, y_j(n), \tag{3.27}$$

где m — общее число входов (за исключением порога) нейрона j; $w_{ji}(n)$ — синаптический вес, соединяющий нейроны i и j; $y_i(n)$ — входной сигнал нейрона j или выходной сигнал нейрона i. Если нейрон j расположен в первом скрытом слое сети, то $m = m_0$, а индекс i относится к i-му входному терминалу сети, для которого можно записать

$$y_i(n) = x_i(n), \tag{3.28}$$

где $x_i(n)$ — i-й элемент входного вектора (образа). С другой стороны, если нейрон j расположен в выходном слое сети, то $m = m_L$, а индекс j означает j-й выходной терминал сети, для которого можно записать

$$y_i(n) = o_i(n), \tag{3.29}$$

где $o_j(n)$ — i-й элемент выходного вектора. Выходной сигнал сравнивается с желаемым откликом $d_j(n)$, в результате чего вычисляется сигнал ошибки $e_j(n)$ для j-го выходного нейрона. Таким образом, прямая фаза вычислений начинается с представления входного вектора первому скрытому слою сети и завершается в последнем слое вычислением сигнала ошибки для каждого нейрона этого слоя.

Обратный проход начинается с выходного слоя предъявлением ему сигнала ошибки, который передается справа налево от слоя к слою с параллельным вычислением локального градиента для каждого нейрона. Этот рекурсивный процесс предполагает изменение синаптических весов в соответствии с дельта-правилом (3.25). Для нейрона, расположенного в выходном слое, локальный градиент равен соответствующему сигналу ошибки, умноженному на первую производную нелинейной функции активации. Затем соотношение (3.25) используется для вычисления изменений весов, связанных с выходным слоем нейронов. Зная локальные градиенты для всех нейронов выход-

ного слоя, с помощью (3.24) можно вычислить локальные градиенты всех нейронов предыдущего слоя, а значит, и величину коррекции весов связей с этим слоем. Такие вычисления проводятся для всех слоев в обратном направлении. Заметим, что очередной обучающий пример «фиксируется» на все время прямого и обратного проходов.

Функция активации

Вычисление локального градиента δ для каждого нейрона многослойного перцептрона требует знания производной функции активации φ (·), связанной с этим нейроном. Для существования такой производной функция активации должна быть непрерывной. Другими словами, дифференцируемость является единственным требованием, которому должна удовлетворять функция активации. Примером непрерывно дифференцируемой нелинейной функции активации, которая часто используется в многослойных перцептронах, является сигмоидальная нелинейная функция (sigmoidal nonlinearity), две формы которой описываются ниже.

Логистическая функция

Эта форма сигмоидальной нелинейности в общем виде определяется следующим образом:

$$\varphi_j\left(v_j(n)\right)=\frac{1}{1+\exp\left(-av_j(n)\right)},\ a>0,\ -\infty<v_j(n)<+\infty, \tag{3.30}$$

где $v_j(n)$ — индуцированное локальное поле нейрона j. Из (3.30) видно, что амплитуда выходного сигнала нейрона с такой активационной функцией лежит в диапазоне $0 \leqslant y_j \leqslant 1$. Дифференцируя (3.30) по $v_i(n)$, получим:

$$\varphi_j'\left(v_j(n)\right)=\frac{a\exp\left(-av_j(n)\right)}{[1+\exp\left(-av_j(n)\right)]^2}. \tag{3.31}$$

Так как $y_i(n) = \varphi_j(v_j(n))$, то можно избавиться от экспоненты в выражении (3.31) и представить производную функции активации в следующем виде:

$$\varphi_j'\left(v_j(n)\right)=ay_j(n)\left[1-y_j(n)\right]. \tag{3.32}$$

Для нейрона j, расположенного в выходном слое, $y_j(n) = o_j(n)$. Отсюда локальный градиент нейрона j можно выразить следующим образом:

$$\delta_j(n) = e_j(n)\varphi'_j\left(v_j(n)\right) = a\left[d_j(n) - o_j(n)\right]o_j(n)\left[1 - o_j(n)\right], \quad (3.33)$$

где $o_j(n)$ — функциональный сигнал на выходе нейрона j; $d_j(n)$ — его желаемый сигнал. С другой стороны, для произвольного скрытого нейрона j локальный градиент можно выразить так:

$$\delta_j(n) = \varphi'_j\left(v_j(n)\right)\sum_k \delta_k(n)w_{kj}(n) = ay_j(n)\left[1 - y_j(n)\right]\sum_k \delta_k(n)w_{kj}(n). \quad (3.34)$$

Обратите внимание, что, согласно выражению (3.32), производная $\varphi'_j(v_j(n))$ достигает своего максимального значения при $y_j(n) = 0{,}5$, а минимального (нуля) — при $y_j(n) = 0$ и $y_j(n) = 1$. Так как величина коррекции синаптических весов в сети пропорциональна производной $\varphi'_j(v_j(n))$, то для максимального изменения синаптических весов нейрона его функциональные сигналы (вычисленные в соответствии с сигмоидальной функцией) должны находиться в середине диапазона. Согласно [20], именно это свойство алгоритма обратного распространения вносит наибольший вклад в его устойчивость как алгоритма обучения.

Функция гиперболического тангенса

Еще одной часто используемой формой сигмоидальной нелинейности является функция гиперболического тангенса, которая в общем виде описывается выражением

$$\varphi_j\left(v_j(n)\right) = a\tanh\left(bv_j(n)\right),\ (a,b) > 0, \quad (3.35)$$

где a и b — константы. В действительности функция гиперболического тангенса является не более чем логистической функцией, только масштабированной и смещенной. Ее производная по аргументу $v_j(n)$ определяется следующим образом:

$$\varphi'_j\left(v_j(n)\right) = ab\,\mathrm{sech}^2(by_j(n)) = ab\,(1 - \tanh^2\left(bv_j(n)\right) =$$

$$= \frac{b}{a}\left[a - y_j(n)\right]\left(a + y_j(n)\right). \quad (3.36)$$

Для нейрона j, расположенного в выходном слое, локальный градиент будет иметь вид

$$\delta_j(n) = e_j(n)\varphi'_j\left(v_j(n)\right) = \frac{b}{a}\left[d_j(n) - o_j(n)\right]\left[a - o_j(n)\right]\left[a + o_j(n)\right]. \quad (3.37)$$

Для нейрона *j*, расположенного в скрытом слое, локальный градиент приобретает вид

$$\delta_j(n) = \varphi'_j\left(v_j(n)\right)\sum_k \delta_k(n) w_{kj}(n) =$$

$$= \frac{b}{a}\left[a - y_j(n)\right]\left[a + y_j(n)\right]\sum_k \delta_k(n) w_{kj}(n). \quad (3.38)$$

Используя формулы (3.33) и (3.34) для логистической функции и формулы (3.37) и (3.38) для гиперболического тангенса, можно вычислить локальный градиент даже без явного знания активационной функции.

Скорость обучения

Алгоритм обратного распространения обеспечивает построение в пространстве весов «аппроксимации» для траектории, вычисляемой методом ускоренного спуска. Чем меньше параметр скорости обучения η, тем меньше корректировка синаптических весов, осуществляемая на каждой итерации, и тем более гладкой является траектория в пространстве весов. Однако это улучшение происходит за счет замедления процесса обучения. С другой стороны, если увеличить параметр η для повышения скорости обучения, то результирующие большие изменения синаптических весов могут привести систему в неустойчивое состояние. Простейшим способом повышения скорости обучения без потери устойчивости является изменение дельта-правила (3.13) за счет добавления к нему момента инерции [20] (3.39). Для частного случая алгоритма LMS было показано, что использование постоянной момента α уменьшает диапазон устойчивости параметра скорости обучения η и может привести к неустойчивости, если последний выбран неверно. Более того, с увеличением значения α рассогласование возрастает (см. [21]).

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_j(n), \quad (3.39)$$

где α, как правило, положительное значение, называемое постоянной момента (momentum constant). Как показано на рис. 3.6, она является управляющим воздействием в контуре обратной связи для $\Delta w_{ji}(n)$, где

z^{-1} — оператор единичной задержки. Уравнение (3.39) называется обобщенным дельта-правилом (generalized delta rule). При $\alpha = 0$ оно вырождается в обычное дельта-правило. Вывод алгоритма обратного распространения с учетом постоянной момента содержится в [22].

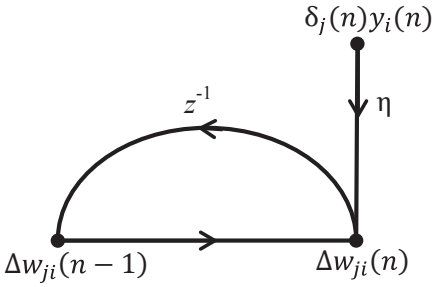


Рис. 3.6. Граф прохождения сигнала, иллюстрирующий эффект постоянной момента α

Для того чтобы оценить влияние последовательности представления обучающих примеров на синаптические веса в зависимости от константы α, перепишем (3.39) в виде временного ряда с индексом *t*. Значение индекса *t* будет изменяться от нуля до текущего значения *n*. При этом выражение (3.39) можно рассматривать как разностное уравнение первого порядка для коррекции весов $\Delta w_{ji}(n)$. Решая это уравнение относительно $\Delta w_{ji}(n)$, получим временной ряд длины $n + 1$:

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t). \tag{3.40}$$

Из уравнений (3.11) и (3.14) видно, что $\delta_j(n) y_i(n) = -\partial E(n) / \partial w_{ji}(n)$. Следовательно, соотношение (3.40) можно переписать в эквивалентной форме:

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}. \tag{3.41}$$

Основываясь на (3.41), можно сделать следующие наблюдения [23, 24].

1. Текущее значение коррекции весов $\Delta w_{ji}(n)$ представляет собой сумму экспоненциально взвешенного временного ряда, для схождения которого постоянная момента должна находиться в диапазоне $0 \le |\alpha| \le 1$. Если константа α равна нулю, алгоритм обратного распространения работает без момента. Заметим, что константа α может быть

и отрицательной, хотя эти значения не рекомендуется использовать на практике.

2. Если частная производная $\partial E(n)/\partial w_{ji}(n)$ имеет один и тот же алгебраический знак на нескольких последовательных итерациях, то экспоненциально взвешенная сумма $\Delta w_{ji}(n)$ возрастает по абсолютному значению, поэтому веса $w_{ji}(n)$ могут изменяться на очень большую величину. Включение момента в алгоритм обратного распространения ведет к ускорению спуска (accelerate descent) в некотором постоянном направлении.

3. Если частная производная $\partial E(n)/\partial w_{ji}(n)$ на нескольких последовательных итерациях меняет знак, экспоненциально взвешенная сумма $\Delta w_{ji}(n)$ уменьшается по абсолютной величине, поэтому веса $w_{ji}(n)$ изменяются на небольшую величину. Таким образом, добавление момента в алгоритм обратного распространения ведет к стабилизирующему эффекту (stabilized effect) для направлений, изменяющих знак.

Включение момента в алгоритм обратного распространения обеспечивает незначительную модификацию метода корректировки весов, оказывая положительное влияние на работу алгоритма обучения. Кроме того, слагаемое момента может предотвратить нежелательную остановку алгоритма в точке какого-либо локального минимума на поверхности ошибок. При выводе алгоритма обратного распространения предполагалось, что параметр интенсивности обучения представлен константой η. Однако на практике он может задаваться как η_{ji}. Это значит, что параметр скорости обучения определяется для каждой конкретной связи. Применяя различные параметры скорости обучения в разных областях сети, можно добиться интересных результатов. Более подробно этот вопрос рассматривается далее.

При реализации алгоритма обратного распространения можно изменять как все синаптические веса сети, так и только часть из них, оставляя прочие на время адаптации фиксированными. В последнем случае сигнал ошибки распространяется по сети в обычном порядке, однако фиксированные синаптические веса будут оставаться неизменными. Этого можно добиться, установив для соответствующих синаптических весов $w_{ji}(n)$ параметр интенсивности обучения η_{ji} равным нулю.

Последовательный и пакетный режимы обучения

В практических приложениях алгоритма обратного распространения в процессе обучения многослойного перцептрона ему многократно предъявляется предопределенное множество обучающих примеров. Как уже отмечалось, один полный цикл предъявления полного набора примеров обучения называют *эпохой*. Процесс обучения проводится от эпохи к эпохе, пока синаптические веса и уровни порога не стабилизируются, а среднеквадратическая ошибка на всем обучающем множестве не сойдется к некоторому минимальному значению. Целесообразно случайным образом изменять порядок представления примеров обучения для разных эпох. Такой принцип предъявления образов делает поиск в пространстве весов стохастическим, предотвращая потенциальную возможность появления замкнутых циклов в процессе эволюции синаптических весов. Замкнутые циклы рассматриваются в последующих главах. Для данного обучающего множества алгоритм обратного распространения можно реализовать двумя способами.

1. Последовательный режим

Последовательный режим (sequential mode) обучения по методу обратного распространения также иногда называют стохастическим (stochastic) или интерактивным (on-line). В этом режиме корректировка весов проводится после подачи каждого примера. Это тот самый режим, для которого мы выводили алгоритм обратного распространения ранее в этой главе. Для примера рассмотрим эпоху, состоящую из N обучающих примеров, упорядоченных следующим образом: $(\mathbf{x}(1), \mathbf{d}(1))$, ..., $(\mathbf{x}(N), \mathbf{d}(N))$. Сети предъявляется первый пример $(\mathbf{x}(1), \mathbf{d}(1))$ этой эпохи, после чего выполняются описанные выше прямые и обратные вычисления. В результате проводится корректировка синаптических весов и уровней порогов в сети. После этого сети предъявляется вторая пара $(\mathbf{x}(2), \mathbf{d}(2))$ в эпохе, повторяются прямой и обратный проходы, приводящие к следующей коррекции синаптических весов и уровня порога. Этот процесс повторяется, пока сеть не завершит обработку последнего примера (пары) данной эпохи — $(\mathbf{x}(N), \mathbf{d}(N))$.

2. Пакетный режим

В пакетном режиме (batch mode) обучения по методу обратного распространения корректировка весов проводится после подачи в сеть

примеров обучения (эпохи). Для конкретной эпохи функция стоимости определяется как среднеквадратическая ошибка (3.2) и (3.3), представленная в составной форме:

$$\mathbf{E}_{\mathbf{av}}(n) = \frac{1}{2N}\sum_{n=1}^{N}\sum_{j\in C} e_j^2(n), \tag{3.42}$$

где сигнал ошибки $e_j(n)$ соответствует нейрону j для примера обучения n и определяется формулой (4.1). Ошибка $e_j(n)$ равна разности между $d_j(n)$ и $y_j(n)$ для j-го элемента вектора желаемых откликов $\mathbf{d}(n)$ и соответствующего выходного нейрона сети. В выражении (3.42) внутреннее суммирование по j выполняется по всем нейронам выходного слоя сети, в то время как внешнее суммирование по n выполняется по всем образам данной эпохи. При заданном параметре скорости обучения η корректировка, применяемая к синаптическому весу $w_{ji}(n)$, связывающему нейроны i и j, определяется следующим дельта-правилом:

$$\Delta w_{ji}(n) = -\eta\frac{\partial E_{av}}{\partial w_{ji}} = -\frac{\eta}{N}\sum_{n=1}^{N} e_j \frac{\partial e_j(n)}{\partial w_{ji}}. \tag{3.43}$$

Для вычисления частной производной $\partial e_j(n)/\partial w_{ji}$ нужно проделать тот же путь, что и ранее. Согласно (3.43), в пакетном режиме корректировка веса $\Delta w_{ji}(n)$ выполняется только после прохождения по сети всего множества примеров.

С точки зрения процессов реального времени, последовательный режим является более предпочтительным, чем пакетный, так как требует меньшего объема внутреннего хранилища для каждой синаптической связи. Более того, предъявляя обучающие примеры в случайном порядке (в процессе последовательной корректировки весов), поиск в пространстве весов можно сделать действительно стохастическим, что сокращает до минимума возможность остановки алгоритма в точке какого-либо локального минимума.

Следует отметить, что стохастическая природа последовательного режима усложняет построение теоретического фундамента для нахождения условий сходимости алгоритма. В противовес этому использование пакетного режима обеспечивает точную оценку вектора градиента. Таким образом, сходимость алгоритма к локальному минимуму гарантируется при довольно простых условиях. Помимо того, в пакетном режиме легче распараллелить вычисления.

Если данные обучения являются избыточными, т. е. содержат по несколько копий одних и тех же примеров, то предпочтительнее использовать последовательный режим, так как примеры все равно подаются по одному. Это преимущество особенно заметно при больших наборах данных с высокой степенью избыточности. Последовательный режим алгоритма обратного распространения остается очень популярным (особенно при решении задач распознавания образов) по двум практическим причинам:

— этот алгоритм прост в реализации;

— он обеспечивает эффективное решение сложных и больших задач.

Критерий останова

В общем случае не существует доказательства сходимости алгоритма обратного распространения, как не существует и какого-либо четко определенного критерия его останова. Известно лишь несколько обоснованных критериев, которые можно использовать для прекращения корректировки весов. Каждый из них имеет свои практические преимущества. Для того чтобы сформулировать такой критерий, вполне логично рассуждать в терминах уникальных свойств локального и глобального минимумов поверхности ошибок. Обозначим символом $\mathbf{w}^*$ вектор весов, обеспечивающий минимум, будь то локальный или глобальный. Необходимым условием минимума является то, что вектор градиента $\mathbf{g}(\mathbf{w})$ (т. е. вектор частных производных первого порядка) для поверхности ошибок в этой точке равен нулевому. Следовательно, можно сформулировать разумный критерий сходимости алгоритма обучения обратного распространения [25]. Вектор $\mathbf{w}^*$ называется локальным минимумом функции F, если в этой точке она достигает наименьшего значения в некоторой области, т. е. если существует такое значение ε, что [26] $F(\mathbf{w}^*) \leqslant F(\mathbf{w})$ для всех $\mathbf{w}$, удовлетворяющих неравенству $\|\mathbf{w} - \mathbf{w}^*\| < \varepsilon$. Вектор $\mathbf{w}^*$ называется глобальным минимумом функции F, если в этой точке она достигает наименьшего значения на всей своей области определения, т. е. если $F(\mathbf{w}^*) \leqslant F(\mathbf{w})$ для всех $\mathbf{w} \in R^n$, где n — размерность вектора $\mathbf{w}$.

Считается, что алгоритм обратного распространения сошелся, если евклидова норма вектора градиента достигает достаточно малых значений.

Недостаток этого критерия сходимости в том, что для сходимости обучения может потребоваться довольно много времени. Кроме того,

необходимо постоянно вычислять вектор градиента $\mathbf{g}(\mathbf{w})$. Другое уникальное свойство минимума заключается в том, что функция стоимости (или мера ошибки) $\mathbf{E}_{av}(\mathbf{w})$ в точке $\mathbf{w} = \mathbf{w}^*$ стабилизируется. Отсюда можно вывести еще один критерий сходимости.

Критерием сходимости алгоритма обратного распространения является достаточно малая абсолютная интенсивность изменений среднеквадратической ошибки в течение эпохи.

Интенсивность изменения среднеквадратической ошибки обычно считается достаточно малой, если она лежит в пределах 0,1–1 % за эпоху. Иногда используется уменьшенное значение — 0,01 %. К сожалению, этот критерий может привести к преждевременной остановке процесса обучения. Существует еще один полезный и теоретически подкрепленный критерий сходимости. После каждой итерации обучения сеть тестируется на эффективность обобщения. Процесс обучения останавливается, когда эффективность обобщения становится удовлетворительной или когда оказывается, что пик эффективности уже пройден.

3.4. Алгоритм обратного распространения в краткой форме

На рис. 3.1 представлена архитектурная схема многослойного перцептрона. Соответствующий граф передачи сигнала в процессе обучения по методу обратного распространения, иллюстрирующий как прямую, так и обратную фазу вычислений, представлен на рис. 3.7 для случая $L = 2$ и $m_0 = m_1 = m_2 = 3$. В верхней части графа передачи сигнала показан прямой проход, в нижней — обратный. Последний еще носит название графа чувствительности (sensitivity graph) для вычисления локальных градиентов в алгоритме обратного распространения [27]. Ранее мы упоминали, что последовательная корректировка весов является более предпочтительным режимом алгоритма обратного распространения для реализации в реальном времени. В этом режиме алгоритм циклически обрабатывает примеры из обучающего множества $\{(x(n), d(n))\}_{n=1}^{N}$ следующим образом.

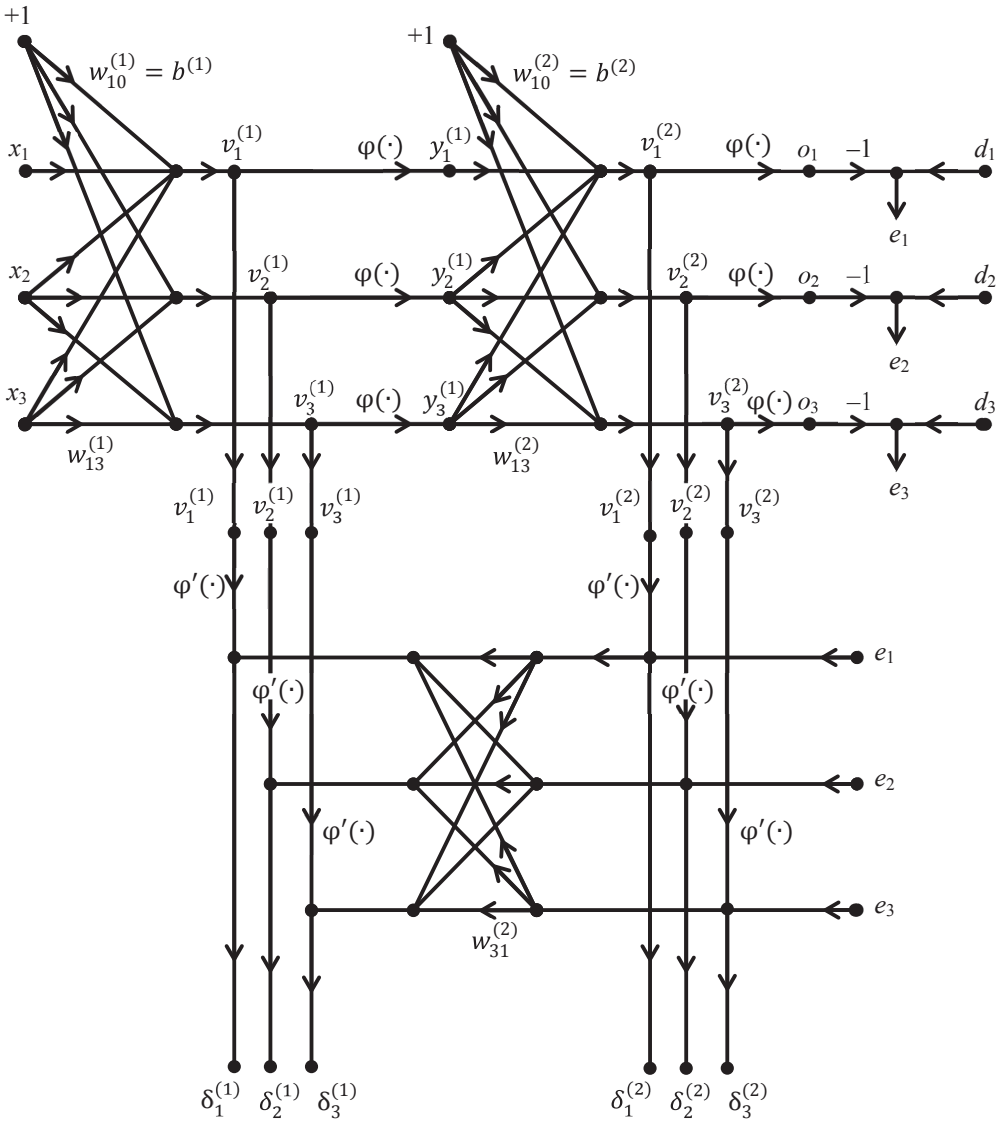


Рис. 3.7. Граф передачи сигнала для процесса обучения по методу обратного распространения. Верхняя часть графа — прямой проход, нижняя часть графа — обратный

1. *Инициализация* (initialization). Предполагая отсутствие априорной информации, генерируем синаптические веса и пороговые значения с помощью датчика равномерно распределенных чисел со средним значением 0. Дисперсия выбирается таким образом, чтобы стандарт-

ное отклонение индуцированного локального поля нейронов приходилось на линейную часть сигмоидальной функции активации (и не достигало области насыщения).

2. *Предъявление примеров обучения* (presentation of training examples). В сеть подаются образы из обучающего множества (эпохи). Для каждого образа последовательно выполняются прямой и обратный проходы, описанные далее в пп. 3 и 4.

3. *Прямой проход* (forward computation). Пусть пример обучения представлен парой $(\mathbf{x}(n), \mathbf{d}(n))$, где $\mathbf{x}(n)$ — входной вектор, предъявляемый входному слою сенсорных узлов; $\mathbf{d}(n)$ — желаемый отклик, предоставляемый выходному слою нейронов для формирования сигнала ошибки. Вычисляем индуцированные локальные поля и функциональные сигналы сети, проходя по ней послойно в прямом направлении. Индуцированное локальное поле нейрона j слоя l вычисляется по формуле:

$$v_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n)\, y_i^{(l-1)}(n), \tag{3.44}$$

где $y_i^{(l-1)}(n)$ — выходной (функциональный) сигнал нейрона i, расположенного в предыдущем слое $l-1$, на итерации n; $w_{ji}^{(l)}(n)$ — синаптический вес связи нейрона j слоя l с нейроном i слоя $l-1$. Для $i=0$ $y_0^{(l-1)}(n) = +1$, а $w_{j0}^{(l)}(n) = b_j^l(n)$ — порог, применяемый к нейрону j слоя l. Если используется сигмоидальная функция, то выходной сигнал нейрона j слоя l выражается $y_j^{(l)}(n) = \varphi_j\left(v_j(n)\right)$. Если нейрон j находится в первом скрытом слое (т. е. $l=1$), то $y_j^{(0)}(n) = x_j(n)$, где $x_j(n)$ — j-й элемент входного вектора $\mathbf{x}(n)$. Если нейрон j находится в выходном слое (т. е. $l = L$, где L — глубина сети), то $y_j^{(L)}(n) = o_j(n)$.

Вычисляем сигнал ошибки

$$e_j(n) = d_j(n) - o_j(n), \tag{3.45}$$

где $d_j(n)$ — j-й элемент вектора желаемого отклика $\mathbf{d}(n)$.

4. *Обратный проход* (backward computation). Вычисляем локальные градиенты узлов сети по следующей формуле:

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\,\varphi_j'\left(v_j^{(L)}(n)\right), & \text{для нейрона } j \text{ слоя } L, \\ \varphi_j'\left(v_j^{(l)}(n)\right)\sum_k \delta_k^{l+1}(n)\,\delta_{kj}^{l+1}(n), & \text{для нейрона } j \text{ слоя } l, \end{cases} \tag{3.46}$$

где штрих в функции $\varphi_j'(\cdot)$ обозначает дифференцирование по аргументу. Изменение синаптических весов слоя l сети выполняется в соответствии с обобщенным дельта-правилом

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^{(l)}(n)\, y_j^{(l-1)}(n), \qquad (3.47)$$

где η — параметр скорости обучения; α — постоянная момента.

5. *Итерации* (iteration). Последовательно выполняем прямой и обратный проходы (согласно пп. 3, 4), предъявляя сети все примеры обучения из эпохи, пока не будет достигнут критерий останова. Порядок представления примеров обучения может случайным образом меняться от эпохи к эпохе. Параметры момента и скорости обучения настраиваются (и обычно уменьшаются) по мере роста количества итераций.

3.5. Задача исключающего ИЛИ (XOR)

В однослойном перцептроне нет скрытых нейронов. Следовательно, он не может классифицировать входные образы, которые линейно неразделимы. Однако нелинейно разделимые области встречаются не так уж редко. Например, именно такая проблема возникает в известной задаче «исключающего ИЛИ» (XOR), которую можно рассматривать как частный случай более общей задачи классификации точек единичного гиперкуба. Каждая точка этого куба принадлежит одному из двух классов — 0 или 1. Для частного случая задачи XOR рассмотрим четыре угла единичного квадрата, которые соответствуют входным парам (0, 0), (0, 1), (1, 0) и (1, 1). Первая и четвертая пары принадлежат классу 0, т. е. $0 \oplus 0 = 0$ и $1 \oplus 1 = 0$, где знак $\oplus$ обозначает оператор булевой функции XOR. Входные образы (0, 0) и (1,1) располагаются в противоположных углах единичного квадрата (unit square), но генерируют одинаковый выходной сигнал, равный нулю. С другой стороны, пары (0, 1) и (1, 0) также располагаются в противоположных углах квадрата, но принадлежат классу 1: $0 \oplus 1 = 1$ и $1 \oplus 0 = 1$.

Очевидно, что при использовании всего одного нейрона с двумя входами граница решений в пространстве входов будет линейной. Для всех точек, расположенных по одну сторону этой линии, выход нейрона будет равен единице, а для остальных точек — нулю. Положение

и ориентация этой линии в пространстве входов определяется синаптическими весами, соединяющими нейрон с входными узлами, и порогом. Имея две пары точек, расположенных в противоположных углах квадрата и относящихся к разным классам, мы, естественно, не сможем построить прямую линию так, чтобы точки одного класса лежали по одну сторону от разделяющей поверхности. Другими словами, элементарный перцептрон не может решить задачу XOR.

Для решения этой задачи введем скрытый слой с двумя нейронами (рис. 3.8, *а*) [28]. Граф передачи сигнала этой сети показан на рис. 3.8, *б*. При этом сделаем следующие допущения:

— каждый из нейронов представлен моделью Мак-Каллока — Питца, в которой функцией активации является пороговая;

— биты 0 и 1 представлены соответственно уровнями сигнала 0 и +1.

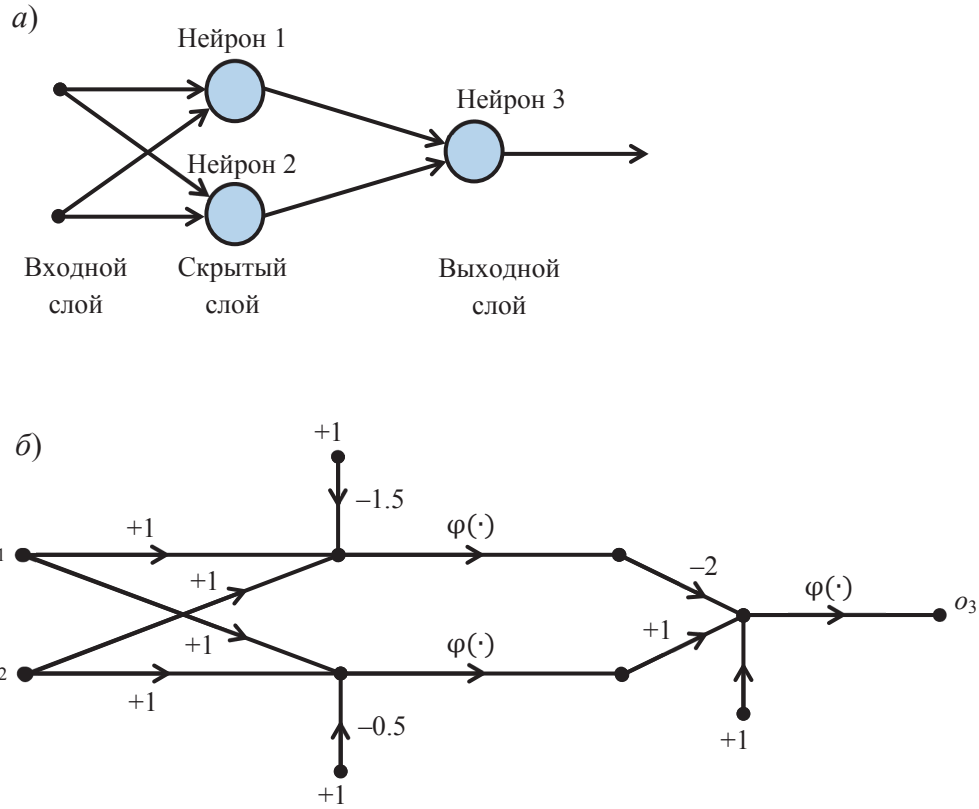


Рис. 3.8. Архитектурный граф сети для решения задачи XOR (*а*) и граф передачи сигнала для этой сети (*б*)

Верхний нейрон (с меткой 1 в скрытом слое) характеризуется следующим образом: $w_{11} = w_{12} = +1$, $b_1 = -3/2$.

Наклон границы решений для этого скрытого нейрона равен -1, что и показано на рис. 3.9, *а*. Нижний нейрон (с меткой 2 в скрытом слое) обладает свойствами $w_{21} = w_{22} = +1$, $b_2 = -1/2$.

Ориентация и расположение границы решений для второго скрытого нейрона показаны на рис. 3.9, *б*. Для выходного нейрона (обозначенного меткой 3 на рис. 3.8, *а*) $w_{31} = -2$, $w_{32} = +1$, $b_1 = -1/2$.

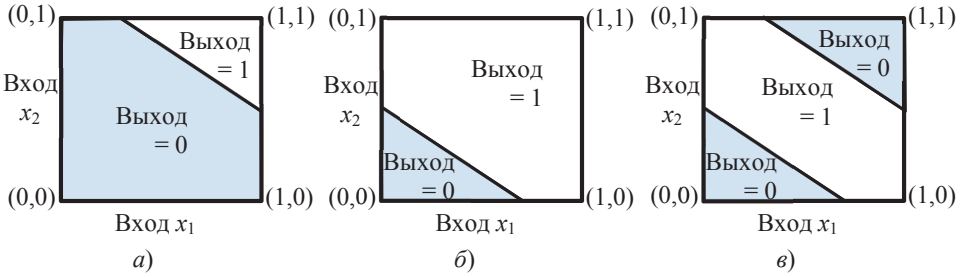


Рис. 3.9. Границы решений, построенные для скрытого нейрона 1 сети, показанной на рис. 3.8, *а* (*а*); границы решений для скрытого нейрона 2 сети (*б*) и для всей сети (*в*)

Функцией выходного нейрона является построение линейной комбинации границ решений, сформированных двумя скрытыми нейронами. Результат вычислений показан на рис. 3.9, *в*. Нижний скрытый нейрон соединен возбуждающей (положительной) связью с выходным нейроном, в то время как верхний скрытый нейрон — тормозящей (отрицательной). Если оба скрытых нейрона заторможены (что соответствует входному образу (0, 0)), выходной нейрон также остается неактивным. Если оба скрытых нейрона возбуждены (что соответствует входному образу (1, 1)), выходной нейрон остается неактивным, так как тормозящее влияние большого отрицательного веса (верхний скрытый нейрон) преобладает над возбуждающим воздействием с меньшим весом (нижний скрытый нейрон). Когда верхний скрытый нейрон находится в заторможенном состоянии, а нижний — в возбужденном (что соответствует входным образам (0, 1) и (1, 0)), выходной нейрон переходит в возбужденное состояние, так как возбуждающий сигнал с положительным весом приходит от нижнего скрытого нейрона. Таким образом, сеть, изображенная на рис. 3.8, *а*, и в самом деле решает задачу XOR.

3.6. Рекомендации по улучшению работы алгоритма обратного распространения

Часто утверждают, что проектирование нейронных сетей, использующих алгоритм обратного распространения, является скорее искусством, чем наукой. При этом имеют в виду тот факт, что многочисленные параметры этого процесса определяются только на основе личного практического опыта разработчика. В этом утверждении есть доля правды. Тем не менее, приведем некоторые общие методы, улучшающие производительность алгоритма обратного распространения.

1. *Режим: последовательный или пакетный* (sequential versus batch update). Как уже говорилось ранее, последовательный режим обучения методом обратного распространения (использующий последовательное предоставление примеров эпохи с обновлением весов на каждом шаге) в вычислительном смысле оказывается значительно быстрее. Это особенно сказывается тогда, когда обучающее множество является большим и в высокой степени избыточным. (Избыточные данные вызывают вычислительные проблемы при оценке якобиана, необходимой для пакетного режима.)

2. *Максимизация информативности* (maximizing information content). Как правило, каждый обучающий пример, предоставляемый алгоритму обратного распространения, нужно выбирать из соображений наибольшей информационной насыщенности в области решаемой задачи [29]. Для этого существуют два общих метода:

— использование примеров, вызывающих наибольшие ошибки обучения;
— использование примеров, которые радикально отличаются от ранее использованных.

Эти два эвристических правила мотивированы желанием максимально расширить область поиска в пространстве весов.

В задачах классификации, основанных на последовательном обучении методом обратного распространения, обычно применяется метод случайного изменения порядка следования примеров, подаваемых на вход многослойного перцептрона, от одной эпохи к другой. В идеале такая рандомизация приводит к тому, что успешно обрабатываемые примеры будут принадлежать к различным классам.

Более утонченным приемом является схема акцентирования (emphasizing scheme), согласно которой более сложные примеры подаются в систему чаще, чем более легкие [29]. Простота или сложность отдельных примеров выявляется с помощью анализа динамики ошибок (в разрезе итераций), генерируемых системой при обработке обучающих примеров. Однако использование схемы акцентирования приводит к двум проблемам, которые следует учесть.

1) распределение примеров в эпохе, представляемой сети, искажается;
2) наличие исключений или немаркированных примеров может привести к катастрофическим последствиям с точки зрения эффективности алгоритма. Обучение на таких исключениях подвергает риску способность сети к обобщению в наиболее правдоподобных областях пространства входных сигналов.

3. *Функция активации* (activation function). Многослойный перцептрон, обучаемый по алгоритму обратного распространения, может в принципе обучаться быстрее (в терминах требуемого для обучения количества итераций), если сигмоидальная функция активации нейронов сети является антисимметричной, а не симметричной. Функция активации $\varphi(v)$ называется антисимметричной (т. е. четной функцией своего аргумента), если $\varphi(-v) = -\varphi(v)$, что показано на рис. 3.10, *а*. Стандартная логистическая функция не удовлетворяет этому условию (рис. 3.10, *б*).

Известным примером антисимметричной функции активации является сигмоидальная нелинейная функция гиперболического тангенса (hyperbolic tangent) $\varphi(v) = a \tanh(bv)$, где a и b — константы. Удобными значениями для констант являются следующие [29]: $a = 1{,}7159$, $b = 2/3$.

Определенная таким образом функция гиперболического тангенса имеет ряд полезных свойств:

— $\varphi(1) = 1$ и $\varphi(-1) = -1$;
— в начале координат тангенс угла наклона (т. е. эффективный угол) функции активации близок к единице: $\varphi(0) = ab = 1{,}7159 \times 2/3 = 1{,}1424$;
— вторая производная $\varphi(v)$ достигает своего максимального значения при $v = 1$.

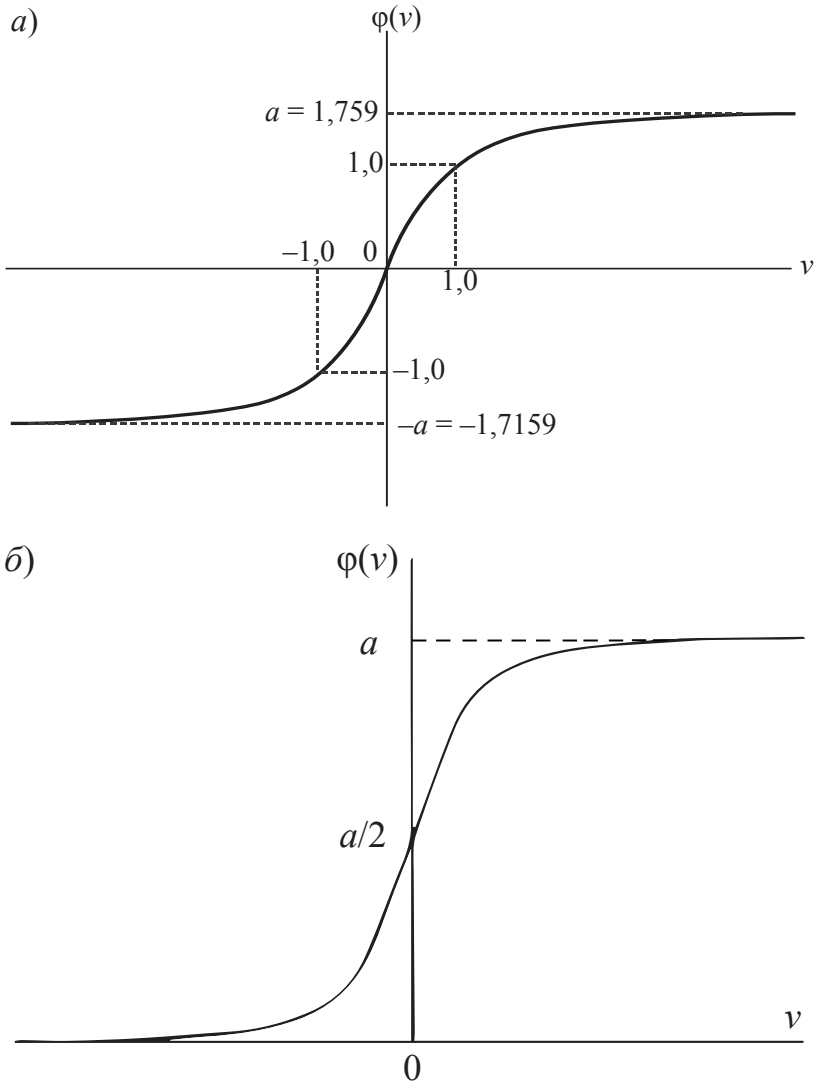


Рис. 3.10. Антисимметричная (*а*) и асимметричная (*б*) функции активации

4. *Целевые значения* (target value). Очень важно, чтобы целевые значения выбирались из области значений сигмоидальной функции активации. Более точно: желаемый отклик d_j нейрона j выходного слоя многослойного перцептрона должен быть смещен на некоторую величину ε от границы области значений функции активации в сторону ее внутренней части. В противном случае алгоритм обратно-

го распространения будет модифицировать свободные параметры сети, устремляя их в бесконечность, замедляя таким образом процесс обучения и доводя скрытые нейроны до предела насыщения. В качестве примера рассмотрим антисимметричную функцию активации, показанную на рис. 3.10, *а*. Для предельного значения $+a$ выберем $d_j = a - \varepsilon$. Аналогично, для предельного значения $-a$ установим $d_j = -a + \varepsilon$, где ε — соответствующая положительная константа. Для выбранного ранее значения $a = 1{,}7159$ установим $\varepsilon = 0{,}7159$. В этом случае желаемый отклик d_j будет находиться в диапазоне от -1 до $+1$ (см. рис. 3.10, *а*).

5. *Нормализация входов* (normalizing the inputs). Все входные переменвые должны быть предварительно обработаны так, чтобы среднее значение по всему обучающему множеству было близко к нулю, иначе их будет сложно сравнивать со стандартным отклонением [29]. Для оценки практической значимости этого правила рассмотрим экстремальный случай, когда все входные переменвые положительны. В этом случае синаптические веса нейрона первого скрытого слоя могут либо одновременно увеличиваться, либо одновременно уменьшаться. Следовательно, вектор весов этого нейрона будет менять направление, что приведет к зигзагообразному движению по поверхности ошибки. Такая ситуация обычно замедляет процесс обучения и, таким образом, неприемлема.

Чтобы ускорить процесс обучения методом обратного распространения, входные векторы необходимо нормализовать в двух следующих аспектах [29].

1) Входные переменные, содержащиеся в обучающем множестве, должны быть некоррелированны (uncorrelated). Этого можно добиться с помощью анализа главных компонентов.
2) Некоррелированные входные переменные должны быть масштабированы так, чтобы их ковариация была приближенно равной (approximately equal). Тогда различные синаптические веса сети будут обучаться приблизительно с одной скоростью.

На рис. 3.11 показан результат трех шагов нормализации: смещения среднего, декорреляции и выравнивания ковариации, примененных в указанном порядке.

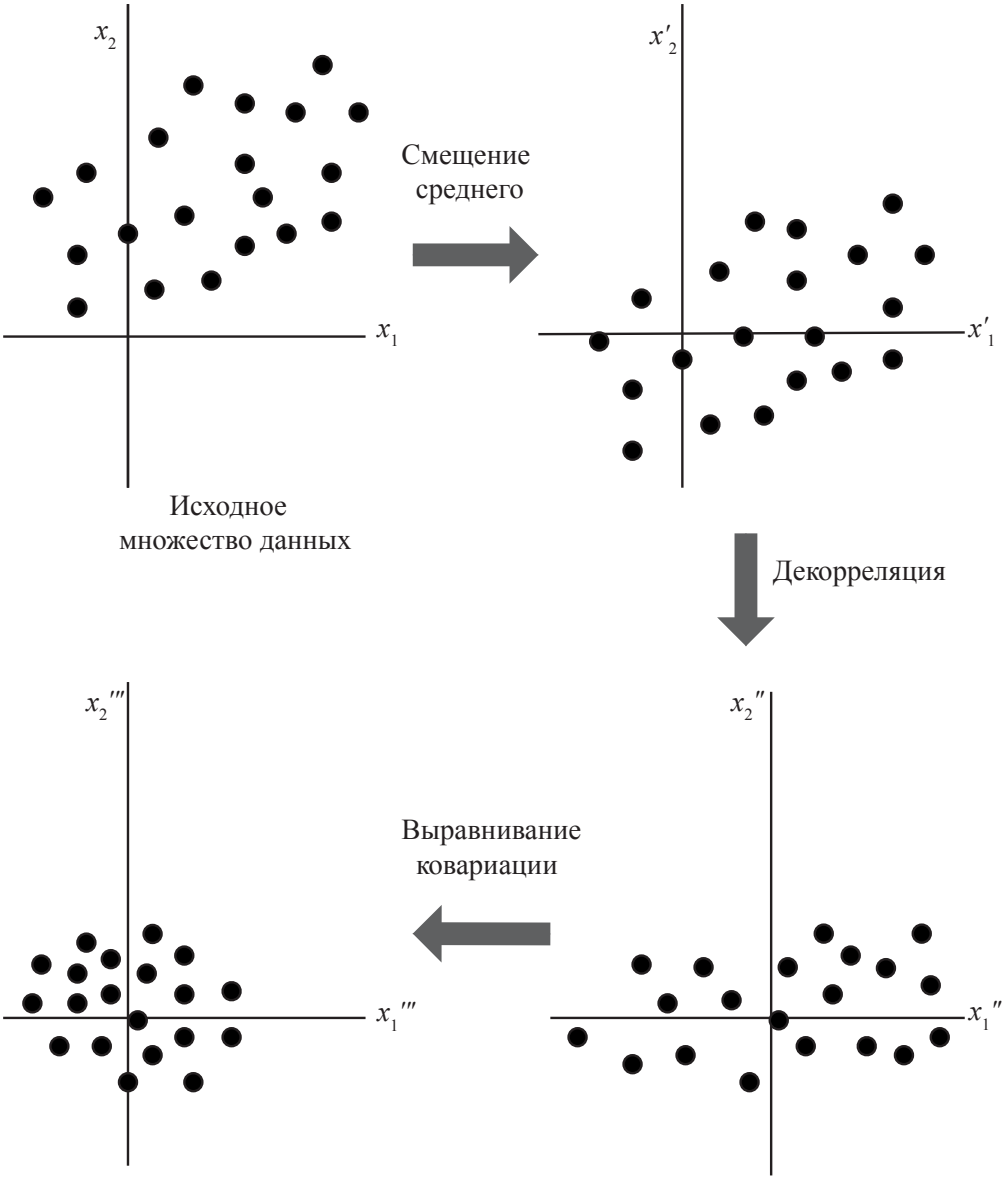


Рис. 3.11. Результаты трех шагов нормализации:
смещения среднего, декорреляции и выравнивания ковариации

6. *Инициализация* (initialization). Хороший выбор начальных значений синаптических весов и пороговых значений (threshold) сети может оказать неоценимую помощь в проектировании. Естественно, возникает вопрос: «А что такое “хорошо”?» Если синаптические веса

принимают большие начальные значения, то нейроны, скорее всего, достигнут режима насыщения. Если такое случится, то локальные градиенты алгоритма обратного распространения будут принимать малые значения, что, в свою очередь, вызовет торможение процесса обучения. Если же синаптическим весам присвоить малые начальные значения, алгоритм будет очень вяло работать в окрестности начала координат поверхности ошибок. В частности, это верно для случая антисимметричной функции активации, такой как гиперболический тангенс. К сожалению, начало координат является седловой точкой (saddle point), т. е. стационарной точкой, где образующие поверхности ошибок вдоль одной оси имеют положительный градиент, а вдоль другой — отрицательный. По этим причинам нет смысла использовать как слишком большие, так и слишком маленькие начальные значения синаптических весов. Как всегда, золотая середина находится между этими крайностями.

Для примера рассмотрим многослойный перцептрон, в котором в качестве функции активации используется гиперболический тангенс. Пусть пороговое значение, применяемое к нейронам сети, равно нулю. Исходя из этого, индуцированное локальное поле нейрона j можно выразить следующим образом: $v_j = \sum_{i=1}^{m} w_{ji} y_i$.

Предположим, что входные значения, передаваемые нейронам сети, имеют нулевое среднее значение и дисперсию, равную единице, т. е. $\mu_y = E[y_i] = 0$ для всех i, $\sigma_y^2 = E\left[(y_i - \mu_i)^2\right] = E\left[y_i^2\right] = 1$ для всех i.

Далее предположим, что входные сигналы некоррелированы:

$$E[y_i y_k] = \begin{cases} 1 \text{ для } k = i, \\ 0 \text{ для } k \neq i, \end{cases}$$

и синаптические веса выбраны из множества равномерно распределенных чисел с нулевым средним $\mu_w = E\left[w_{ji}\right] = 0$ для всех пар (j, i) и дисперсией $\sigma_w^2 = E\left[\left(w_{ji} - \mu_w\right)^2\right] = E\left[w_{ji}^2\right]$ для всех пар (j, i). Следовательно, математическое ожидание и дисперсию индуцированного локального поля можно выразить так:

$$\mu_v = E[v_j] = E\left[\sum_{i=1}^{m} w_{ji} y_i\right] = \sum_{i=1}^{m} E[w_{ji}] E[y_i] = 0,$$

$$\sigma_v^2 = E[(v_i - \mu_v)^2] = E[v_j^2] = E\left[\sum_{i=1}^{m}\sum_{k=1}^{m} w_{ji} w_{jk} y_i y_k\right] =$$
$$= \sum_{i=1}^{m}\sum_{k=1}^{m} E[w_{ji} w_{jk}] E[y_i y_k] = \sum_{i=1}^{m} E[w_{ji}^2] = m\sigma_w^2, \qquad (4.48)$$

где m — число синаптических связей нейрона.

На основании этого результата можно описать хорошую стратегию инициализации синаптических весов таким образом, чтобы стандартное отклонение индуцированного локального поля нейрона лежало в переходной области между линейной частью сигмоидальной функции активации и областью насыщения. Например, для случая гиперболического тангенса с параметрами a и b (см. определение функции) эта цель достигается при $\sigma_v = 1$ в (3.48). Исходя из этого, получим [29]:

$$\sigma_w = m^{-1/2}. \qquad (4.49)$$

Таким образом, желательно, чтобы равномерное распределение, из которого выбираются исходные значения синаптических весов, имело нулевое среднее значение и дисперсию, обратную корню квадратному из количества синаптических связей нейрона.

7. *Обучение по подсказке* (hints). Обучение на множестве примеров связано с аппроксимацией неизвестной функцией отображения входного сигнала на выходной. В процессе обучения из примеров извлекается информация о функции $f(\cdot)$ и строится некоторая аппроксимация этой функциональной зависимости. Процесс обучения на примерах можно обобщить, добавив обучение по подсказке, которое реализуется путем предоставления некоторой априорной информации о функции $f(\cdot)$ [30]. Такая информация может включать свойства инвариантности, симметрии и прочие знания о функции $f(\cdot)$, которые можно использовать для ускорения поиска ее аппроксимации и, что более важно, для повышения качества конечной оценки. Использование соотношения (3.49) является одним из примеров такого подхода.

8. *Скорость обучения* (leaming rates). Все нейроны многослойного перцептрона в идеале должны обучаться с одинаковой скоростью. Однако последние слои обычно имеют более высокие значения локальных градиентов, чем начальные слои сети. Исходя из этого, параметру скорости обучения η, следует назначать меньшие значения для последних слоев сети и большие — для первых. Чтобы время обучения для всех нейронов сети было примерно одинаковым, нейроны

с большим числом входов должны иметь меньшее значение параметра обучения, чем нейроны с малым количеством входов. В [29] предлагается назначать параметр скорости обучения для каждого нейрона обратно пропорционально квадратному корню из суммы его синаптических связей.

3.7. Представление выхода и решающее правило

Теоретически для задачи классификации на M классов (M-class classification problem), в которой объединение M классов формирует все пространство входных сигналов, для представления всех возможных результатов классификации требуется M выходов (рис. 3.12). На этом рисунке вектор $\mathbf{x}_j$ является j-м прототипом (prototype) (т. е. отдельной реализацией) m-мерного случайного вектора $\mathbf{x}$, который должен быть классифицирован многослойным перцептроном. $\mathbf{C}_k$ обозначает k-й из M возможных классов, которому принадлежит данный входной сигнал. Пусть y_{kj} — k-й выход сети, генерируемый в ответ на прототип $\mathbf{x}_i$:

$$y_{k,j} = F_k\left(\mathbf{x}_j\right),\ k = 1, 2, \ldots, M, \tag{3.50}$$

где функция $F_k(\cdot)$ определяет отображение, которому обучается сеть при передаче входного примера на k-й выход. Для удобства представления обозначим

$$y_j = \left[y_{1,j}, y_{2,j} \ldots y_{M,j}\right]^T = \left[F_1\left(x_j\right), F_2\left(x_j\right) \ldots F_M\left(x_j\right)\right]^T = F\left(x_3\right), \tag{3.51}$$

где $\mathbf{F}(\cdot)$ — вектор-функция.

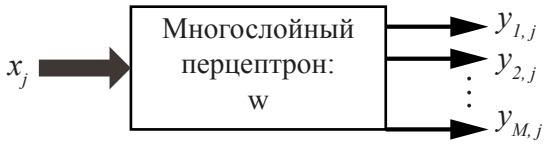


Рис. 3.12. Блочная диаграмма классификатора входных сигналов

Главный вопрос этого раздела звучит так: каким должно быть оптимальное решающее правило, применяемое для классификации M выходов сети после обучения многослойного перцептрона?

Естественно, решающее правило должно основываться на знании вектор-функции:

$$F : R^m x \to y \in R^M. \tag{3.52}$$

В общем случае о вектор-функции $\mathbf{F}(\cdot)$ определенно известно лишь то, что это непрерывная функция, минимизирующая функционал эмпирического риска (empirical risk functional):

$$R = \frac{1}{2N}\sum_{j=1}^{N} \| d_j - F\left(x_j\right) \|^2, \tag{3.53}$$

где $\mathbf{d}_j$ — желаемый (целевой) выход для прототипа $\mathbf{x}_j$; $\|\cdot\|$ — евклидова норма вектора; N — общее число примеров, представленных сети для обучения. Сущность критерия (3.53) та же, что и у функции стоимости (3.3). Вектор-функция $\mathbf{F}(\cdot)$ строго зависит от выбора примеров $(\mathbf{x}_j, \mathbf{d}_j)$, использованных для обучения сети. Это значит, что разные значения пар $(\mathbf{x}_j, \mathbf{d}_j)$ приведут к построению различных вектор-функций $\mathbf{F}(\cdot)$. Обратите внимание, что используемое здесь обозначение $(\mathbf{x}_j, \mathbf{d}_j)$ является эквивалентом употреблявшегося ранее обозначения $(\mathbf{x}(j), \mathbf{d}(j))$.

Предположим, что сеть обучается на двоичных целевых значениях (которые случайно совпадают с верхней и нижней границами области значений логистической функции):

$$d_{kj} = \begin{cases} 1, \text{если прототип } \mathbf{x}_j \text{ принадлежит классу } \mathbf{C}_k, \\ 0, \text{ если прототип } \mathbf{x}_j \text{ не принадлежит классу } \mathbf{C}_k. \end{cases} \tag{3.54}$$

Основываясь на этом допущении, класс $\mathbf{C}_k$ можно представить M-мерным целевым вектором

$$\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow k\text{-й элемент.}$$

Многослойный классификатор перцептронного типа, обученный по алгоритму обратного распространения на конечном множестве независимых и равномерно распределенных примеров, обеспечивает асимптотическую аппроксимацию соответствующей апостериорной вероятности класса. Это свойство можно обосновать следующим образом [31].

1. Согласно закону больших чисел, при бесконечном увеличении размера N обучающего множества вектор $\mathbf{w}$, минимизирующий функционал стоимости R из (3.53), достигает оптимального значения $\mathbf{w}^*$, минимизирующего ожидание случайной величины $\frac{1}{2}\|\mathbf{d} - \mathbf{F}(\mathbf{w}, \mathbf{x})\|^2$, где $\mathbf{d}$ — вектор желаемого отклика; $\mathbf{F}(\mathbf{w}, \mathbf{x})$ — аппроксимация, реализованная многослойным перцептроном для вектора весовых коэффициентов $\mathbf{w}$ и входа $\mathbf{x}$ [31]. Функция $\mathbf{F}(\mathbf{w}, \mathbf{x})$, в которой явным образом показана зависимость от вектора $\mathbf{w}$, — это не что иное, как использованная ранее функция $\mathbf{F}(\mathbf{x})$.

2. Оптимальный вектор весов $\mathbf{w}^*$ обладает тем свойством, что соответствующий ему вектор фактического выхода сети $\mathbf{F}(\mathbf{w}^*, \mathbf{x})$ является аппроксимацией, построенной по методу наименьших квадратов и минимизирующей ошибку условного ожидания вектора желаемого отклика при данном входном векторе $\mathbf{x}$ [31].

3. Для задачи классификации входных сигналов на M классов k-й элемент вектора желаемого отклика равен единице, если входной вектор $\mathbf{x}$ принадлежит к классу $\mathbf{C}_k$, и нулю в противном случае. Отсюда следует, что условное ожидание вектора желаемого отклика при данном векторе $\mathbf{x}$ равно апостериорной вероятности класса $P(\mathbf{C}_k|\mathbf{x})$, $k = 1, 2, ..., M$.

Отсюда следует, что многослойный перцептрон (с логистической активационной функцией) действительно аппроксимирует апостериорную (a posteriori) вероятность распознавания класса при условии, что размерность обучающего множества достаточно велика и что процесс обучения методом обратного распространения не прекратится в точке локального минимума. Теперь можно ответить на поставленный ранее вопрос. В частности, можно утверждать, что соответствующее решающее правило является (приближенно) байесовским правилом, обобщенным для апостериорной вероятности оценок.

Случайный вектор $\mathbf{x}$ относится к классу $\mathbf{C}_k$, если

$$F_k(\mathbf{x}) > F_j(\mathbf{x}) \text{ для всех } j \neq k, \tag{3.55}$$

где $F_k(\mathbf{x})$ и $F_j(\mathbf{x})$ — элементы вектор-функции отображения

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} F_1(x) \\ F_2(x) \\ \cdots \\ F_M(x) \end{bmatrix}.$$

Единственное наибольшее значение выходного сигнала существует с вероятностью 1, если соответствующие апостериорные распределения классов различаются. Здесь предполагается использование арифметики с бесконечной точностью. Это решающее правило имеет определенное преимущество по сравнению с моделью «отжига», поскольку позволяет разделить однозначные (unambiguous) решения. Это значит, что вектор x относится к определенному классу, если соответствующее выходное значение превышает заданный порог (в логистических формах функции активации обычно используется значение 0,5), в противном случае классификация не однозначна.

Ранее было указано, что двоичные целевые значения [0, 1], соответствующие логистической функции (3.30), на практике во время обучения сети должны измениться на небольшое значение ε во избежание насыщения синаптических весов (в связи с далеко не бесконечной точностью представления чисел). В результате этой модификации целевые значения перестают быть двоичными, и асимптотические аппроксимации $F_k(\mathbf{x})$ не являются апостериорными вероятностями $P(\mathbf{C}_k|\mathbf{x})$ интересующих нас M классов [32]. Вместо этого $P(\mathbf{C}_k|\mathbf{x})$ линейно отображается на закрытый отрезок [ε, 1 — ε] так, что $P(\mathbf{C}_k|\mathbf{x}) = 0$ соответствует выходу ε, а $P(\mathbf{C}_k|\mathbf{x}) = 1$ — выходу 1 — ε. Так как такое отображение сохраняет относительный порядок, это не влияет на результат применения выходного решающего правила (3.55).

Интересно также отметить следующее. Если граница решений формируется пороговым отсечением выходов многослойного перцептрона относительно некоторых фиксированных значений, ее общая форма и ориентация могут быть выражены эвристически (для случая единственного скрытого слоя) в терминах количества скрытых нейронов и относительных величин связанных с ними синаптических весов [33]. Однако такой анализ не применим к границе решений, сформированной в соответствии с выходным решающим правилом (3.55). Скрытые нейроны лучше рассматривать как нелинейные детекторы признаков (nonlinear feature detector), призванные отобразить классы исходного входного пространства (возможно, линейно неразделимые) в пространство активности скрытого слоя, где их линейная разделимость более вероятна.

3.8. Компьютерный эксперимент

В этом разделе с помощью компьютерного моделирования будет проиллюстрировано поведение многослойного перцептронного классификатора в процессе обучения. Целью обучения является разделение двух перекрывающихся двумерных классов с гауссовым распределением, обозначенных цифрами 1 и 2. Пусть $\mathbf{C}_1$ и $\mathbf{C}_2$ — множества событий, для которых случайный вектор $\mathbf{x}$ принадлежит к классам 1 и 2 соответственно. Функцию плотности условной вероятности можно представить в следующем виде.

Для класса $\mathbf{C}_1$:

$$f_x(x \mid C_1) = \frac{1}{2\pi\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2} \|x - \mu_1\|^2 \right), \tag{3.56}$$

где вектор среднего значения $\mu_1 = [0, 0]^T$, а дисперсия $\sigma_1^2 = 1$.

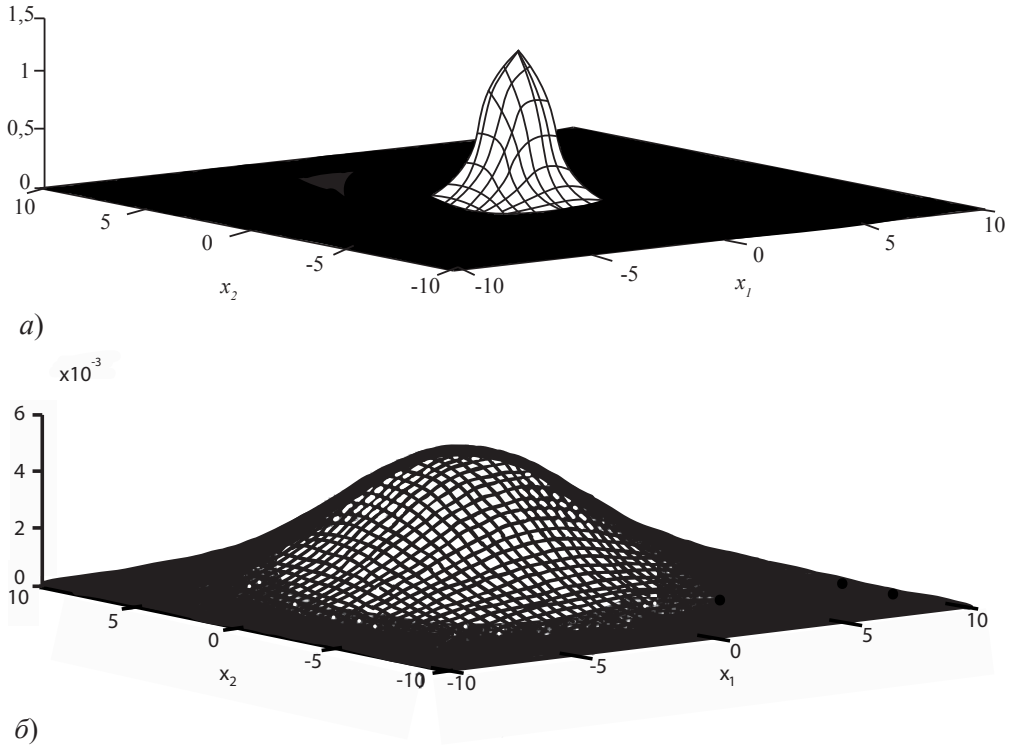
Для класса $\mathbf{C}_2$:

$$f_x(x \mid C_2) = \frac{1}{2\pi\sigma_2^2} \exp\left(-\frac{1}{2\sigma_2^2} \|x - \mu_2\|^2 \right), \tag{3.57}$$

где вектор среднего значения $\mu_2 = [2, 0]^T$, а дисперсия $\sigma_2^2 = 4$.

Вероятность принадлежности образа обоим классам одинакова, т. е. $p_1 = p_2 = 1/2$.

На рис. 3.13 показаны трехмерные графики гауссовых распределений, определенных выражениями (3.56) и (3.57) для входного вектора $\mathbf{x} = [x_1, x_2]^T$ и размерности пространства входных сигналов $m_0 = 2$. На рис. 3.14 показаны корреляционные диаграммы классов $\mathbf{C}_1$ и $\mathbf{C}_2$ в отдельности и общая корреляционная диаграмма, отражающая суперпозицию графиков рассеяния по 500 точкам, выбранным для каждого процесса. На последней диаграмме четко видно, что распределения существенно перекрываются, а значит, высока вероятность неправильной классификации.



а)

б)

Рис. 3.13. Функции плотности вероятности: *а*) $f_x(\mathbf{x}|C_1)$; *б*) $f_x(\mathbf{x}|C_2)$

Байесовская граница решений

Байесовский критерий оптимальной классификации уже обсуждался выше. Для данной задачи классификации на два класса предположим, что классы $\mathbf{C}_1$ и $\mathbf{C}_2$ равновероятны, стоимость корректной классификации равна нулю и стоимости ошибок классификации одинаковы. Тогда для нахождения оптимальной границы решений можно использовать критерий отношения правдоподобия

$$\Lambda(x) \underset{C_2}{\overset{C_1}{\lessgtr}} \xi, \tag{3.58}$$

где $\Lambda(\mathbf{x})$ — отношение правдоподобия (likelihood ratio), определяемое формулой

$$\Lambda(x) = \frac{f_x(x|C_1)}{f_x(x|C_2)}, \tag{3.59}$$

а ξ — порог критерия (threshold of the test), определяемый формулой

$$\xi = p_2 \Big/ p_1 = 1. \tag{3.60}$$

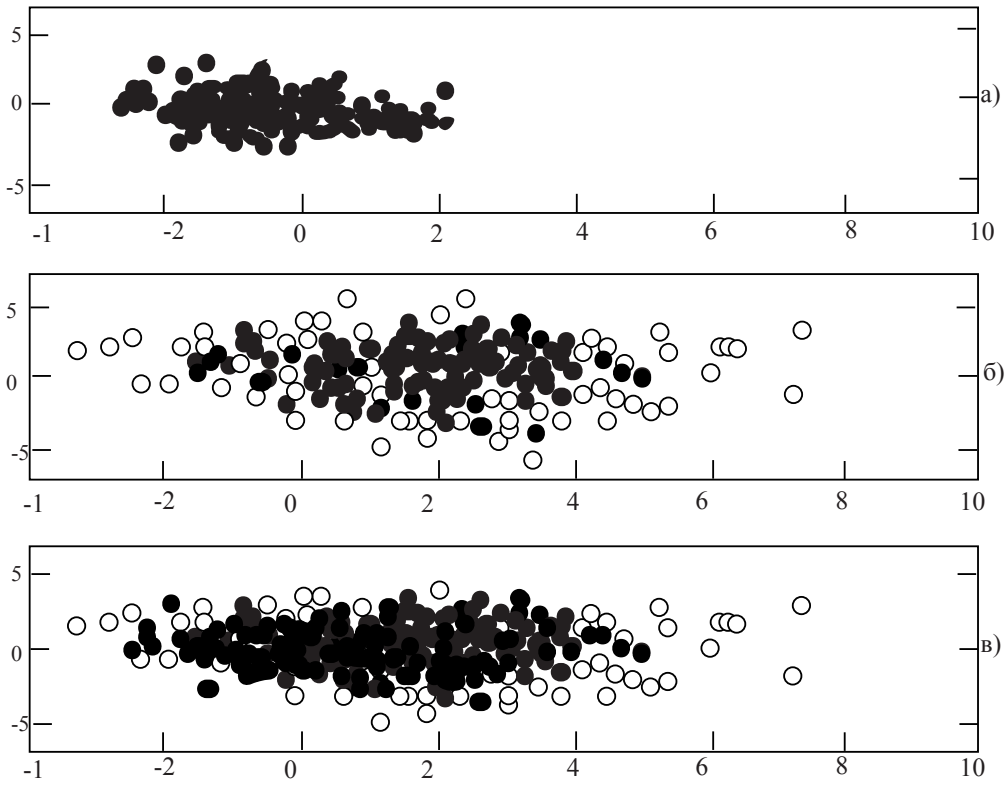


Рис. 3.14. Корреляционные диаграммы классов $\mathbf{C}_1$ (*а*) и $\mathbf{C}_2$ (*б*), а также общая корреляционная диаграмма для обоих классов (*в*)

В рассматриваемом примере имеем:

$$\Lambda(x) = \frac{\sigma_2^2}{\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2} x - \mu_1{}^2 + \frac{1}{2\sigma_2^2} x - \mu_2{}^2 \right).$$

Таким образом, оптимальная (байесовская) граница решений определяется соотношением

$$\frac{\sigma_2^2}{\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2} x - \mu_1{}^2 + \frac{1}{2\sigma_2^2} x - \mu_2{}^2 \right) = 1$$

или

$$\frac{1}{\sigma_2^2}\left\|x-\mu_2\right\|^2+\frac{1}{\sigma_1^2}\left\|x-\mu_1\right\|^2=4\log\left(\frac{\sigma_1}{\sigma_2}\right). \tag{3.61}$$

Используя простые преобразования, оптимальную границу решений (3.61) можно переопределить в виде

$$\|\mathbf{x}-\mathbf{x}\|^2=r^2, \tag{3.62}$$

где

$$x_c=\frac{\sigma_2^2\mu_1-\sigma_1^2\mu_2}{\sigma_2^2-\sigma_1^2}, \tag{3.63}$$

и

$$x_c=\frac{\sigma_2^2\sigma_1^2}{\sigma_2^2-\sigma_1^2}\left[\frac{\left\|\mu_1-\mu_2\right\|^2}{\sigma_2^2-\sigma_1^2}+4\log\left(\frac{\sigma_2}{\sigma_1}\right)\right]. \tag{3.64}$$

Уравнение (3.62) описывает окружность с центром в точке x и радиусом r. Обозначим область, расположенную внутри этого круга, символом Ω_1. Байесовское правило классификации для этой задачи можно сформулировать следующим образом.

Вектор наблюдения $\mathbf{x}$ *относится к классу* $\mathbf{C}_1$, *если отношение правдоподобия* $\Delta(x)$ *больше порогового значения* ξ, *и к классу* $\mathbf{C}_2$ *в остальных случаях.*

В данном эксперименте центр круговой границы решений находится в точке

$$x_c=\begin{bmatrix}-2/3\\0\end{bmatrix},$$

а радиус области составляет $r\cong 2{,}34$.

Обозначим символом c множество результатов корректной классификации, а символом e — множество результатов некорректной классификации. Тогда вероятность ошибки (probability of error) P_e классификатора, работающего на основе байесовского решающего правила, можем определить в виде

$$P_e=p_1P\left(e\,|\,C_1\right)+p_2P\left(e\,|\,C_2\right), \tag{3.65}$$

где $P(e|\mathbf{C}_1)$ — условная вероятность ошибки для входного вектора из класса $\mathbf{C}_1$; $P(e|\mathbf{C}_2)$ — условная вероятность ошибки для входного вектора из класса $\mathbf{C}_2$; p_1 и p_2 — априорные вероятности классов $\mathbf{C}_1$ и $\mathbf{C}_2$ соответственно. Для нашей задачи можно найти числовые значения вышеуказанных величин:

$$P(e|\mathbf{C}_1) \cong 0{,}1056;\ P(e|\mathbf{C}_2) \cong 0{,}2642.$$

Так как классы равновероятны, т. е. $p_1 = p_2 = 1/2$, то $P_e \cong 0{,}1849$.

Следовательно, вероятность правильной классификации (probability of correct classification) составляет $P_c = 1 - P_e \cong 0{,}8151$.

Экспериментальное построение оптимального многослойного перцептрона

В табл. 3.1 приведены параметры многослойного перцептрона с одним слоем скрытых нейронов, который обучается с помощью алгоритма обратного распространения в последовательном режиме. Поскольку единственной целью классификатора является достижение приемлемого уровня корректной классификации, этот критерий и применяется для проверки оптимальности переменных параметров MLP.

Таблица 3.1

Переменные параметры многослойного перцептрона

Параметр	Символ	Типичный диапазон
Количество скрытых нейронов	m_1	$(2, \infty)$
Параметр скорости обучения	η	(0, 1)
Константа момента	α	(0, 1)

Оптимальное число скрытых нейронов

Учитывая практический подход к задаче определения оптимального количества скрытых нейронов (m_1), будем использовать следующий критерий. Необходимо найти минимальное количество скрытых нейронов, которое обеспечивает производительность, близкую (в пределах 1 %) к байесовскому методу. Исходя из этого, эксперимент начинается с двух скрытых нейронов. Результаты моделирования приведены в табл. 3.2.

Поскольку целью первого этапа моделирования является проверка достаточности для двух скрытых нейронов, параметрам α и η произвольно присвоены некоторые номинальные значения. Для каждого запуска процесса моделирования генерировалось множество примеров обучения, с равной вероятностью содержащее образы классов $\mathbf{C}_1$ и $\mathbf{C}_2$ с гауссовым распределением. Примеры из этого множества по-

следовательно подавались на вход сети в течение нескольких циклов, или эпох (epoch). Количество эпох выбиралось так, чтобы общее число обучающих примеров в процессе обучения оставалось постоянным. Таким образом, можно усреднить результаты для обучающих множеств разного размера.

Таблица 3.2

Результаты моделирования для двух скрытых нейронов (коэффициент скорости обучения равен 0,1, фактор момента — 0)

Количество проходов	Размер обучающего множества	Количество эпох	Среднеквадратическая ошибка	Вероятность корректной классификации P_c, %
1	500	320	0,2375	80,36
2	2000	80	0,2341	80,33
3	8000	20	0,2244	80,47

В таблице 3.2 и последующих таблицах среднеквадратическая ошибка вычислялась в соответствии с функционалом, определенным формулой (3.53). Необходимо подчеркнуть, что для этих таблиц определялась именно среднеквадратическая ошибка, хотя минимум среднеквадратической ошибки не всегда отражает хороший уровень обобщения (т. е. хорошую производительность на ранее не использованных данных).

В результате обучения сети на N примерах вероятность корректной классификации теоретически определяется выражением

$$P(c, N) = p_1 P(c, N|C_1) + p_2 P(c, N|C_2), \tag{3.66}$$

где $p_1 = p_2 = 1/2$ и

$$P(c, N \mid C_1) = \int_{\Omega_1(N)} f_x\left(x \mid C_1\right) dx, \tag{3.67}$$

$$P(c, N \mid C_2) = 1 - \int_{\Omega_1(N)} f_x\left(x \mid C_2\right) dx, \tag{3.68}$$

$\Omega_1(N)$ — область пространства решений, в которой многослойный перцептрон (обученный на N примерах) относит вектор $\mathbf{x}$ (представляющий реализацию случайного вектора $\mathbf{X}$) к классу $\mathbf{C}_1$. Эту область обычно находят экспериментально, оценивая функцию отображения, которой обучалась сеть, и применяя выходное решающее правило (3.55). К сожалению, численно оценить величины $P(c, N|\mathbf{C}_1)$ и $P(c, N|\mathbf{C}_2)$ непросто, так как сложно отыскать формальное определение границы ре-

шений $\Omega_1(N)$. Поэтому воспользуемся экспериментальным подходом и протестируем обученный многослойный перцептрон на независимом множестве примеров, которые снова случайно сгенерируем на основе гауссовых распределений классов $\mathbf{C}_1$ и $\mathbf{C}_2$ с равной вероятностью.

Введем случайную переменную A, которая означает количество корректно классифицированных примеров из множества мощности N. Тогда частное $p_N = A/N$ является случайной величиной, которая обеспечивает максимально правдоподобную несмещенную оценку реальной эффективности классификации p. Предполагая, что p является константой для N пар «вход-выход», можно использовать предел Чернова [34]:

$$P(|p_N - p| > \varepsilon) < 2 \exp(-2\varepsilon^2 N) = \delta.$$

Применяя предел Чернова, получим $N = 26500$ для $\delta = 0{,}01$ и $\varepsilon = 0{,}01$ (т. е. с достоверностью 99 % можно утверждать, что оценка p характеризуется данным допустимым отклонением). Рассмотрим тестовое множество, содержащее $N = 32000$ образов. В последнем столбце табл. 3.2 показана вероятность корректной классификации для тестового множества такой мощности, усредненная по десяти независимым экспериментам.

Таблица 3.3

Результаты моделирования для четырех скрытых нейронов ($\eta = 0{,}1$, $\alpha = 0$)

Количество проходов	Размер обучающего множества	Количество эпох	Среднеквадратическая ошибка	Вероятность корректной классификации P_c, %
1	500	320	0,2129	80,80
2	2000	80	0,2108	80,81
3	8000	20	0,2142	80,19

Эффективность классификации многослойного перцептрона с двумя скрытыми нейронами (см. табл. 3.2) достаточно близка к производительности байесовского классификатора, равной $P_c = 81{,}51\,\%$. Отсюда логично заключить, что рассматриваемую задачу классификации можно решить с помощью многослойного перцептронного классификатора с двумя скрытыми нейронами. Чтобы подтвердить это заключение, в табл. 3.3 представлены результаты моделирования для четырех скрытых нейронов (остальные параметры остались без изменений). Хотя среднеквадратическая ошибка в табл. 3.3 для четырех скрытых нейронов несколько ниже, чем в табл. 3.2 для двух скрытых

нейронов, средний уровень корректной классификации практически не улучшился — в одном из тестов результаты оказались даже несколько хуже. Поэтому продолжим вычислительный эксперимент для двух скрытых нейронов.

Оптимальное обучение и константа момента

Для оценки оптимальности значений параметров скорости обучения η и момента α можно использовать любое из трех следующих определений.

1. Оптимальными считаются константы α и η, при которых сходимость сети к локальному минимуму на поверхности ошибок достигается в среднем за минимальное количество эпох.

2. Оптимальными считаются константы α и η, при которых сходимость сети к локальному минимуму на поверхности ошибок (в наихудшем случае или в среднем) достигается за минимальное число эпох.

3. Оптимальными считаются константы α и η, при которых средний показатель сходимости к конфигурации, имеющей наилучшие обобщающие способности во всем пространстве входных сигналов, достигается за минимальное количество эпох.

Используемые здесь термины «средний» и «наихудший» означают распределение обучающих пар типа «вход-выход». Определение 3 является наиболее ценным для практики, однако его сложно применить, так как минимизация среднеквадратической ошибки — это математический критерий оптимальности, применяемый при обучении сети, и, как говорилось ранее, снижение среднеквадратической ошибки не всегда ведет к улучшению обобщающей способности. С исследовательской точки зрения второе определение представляет больший интерес, нежели первое. Например, в [35] были представлены результаты серьезного исследования оптимальной настройки параметра скорости обучения η, при котором многослойному перцептрону требуется минимальное количество эпох для аппроксимации глобально оптимальной матрицы синаптических весов с заданной точностью (правда, только для частного случая линейных нейронов). Тем не менее, в общем случае при экспериментальном и эвристическом подходе определения оптимальных значений η и α используется определение 1. Поэтому в своем эксперименте будем руководствоваться

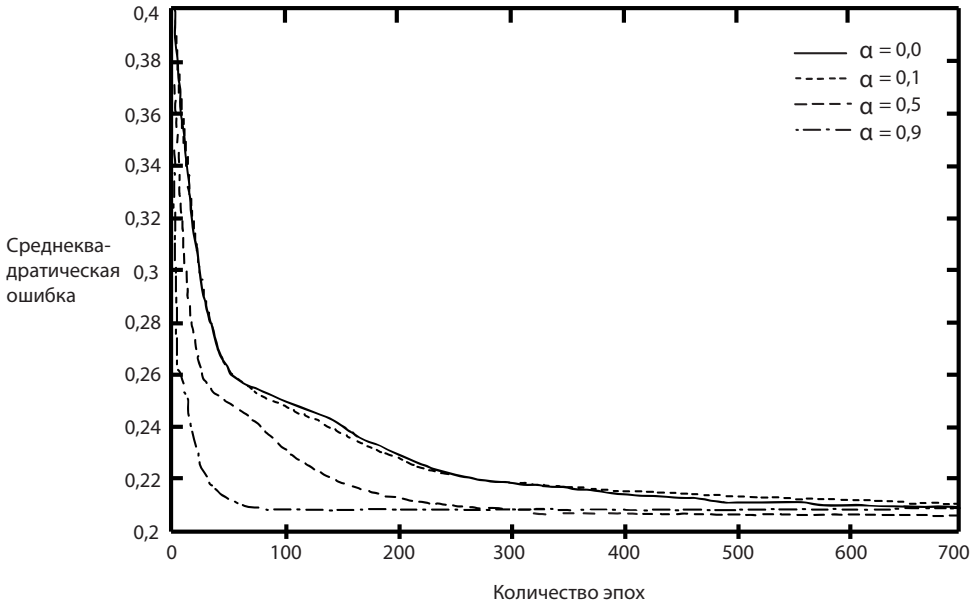
именно этим определением. Используя многослойный перцептрон с двумя скрытыми нейронами, а также различные комбинации значений параметра скорости обучения $\eta \in \{0{,}01;\ 0{,}1;\ 0{,}5;\ 0{,}9\}$ и константы момента $\alpha \in \{0{,}0;\ 0{,}1;\ 0{,}5;\ 0{,}9\}$, исследуем скорость сходимости сети. Каждая из конфигураций обучается на одном и том же множестве примеров при одном и том же наборе исходных значений синаптических весов для $N = 500$. Поэтому результаты можно сравнивать напрямую, без внесения поправок. Процесс обучения длился в течение 700 эпох. Такую продолжительность обучения мы посчитали достаточной для достижения алгоритмом обратного распространения некоторого локального минимума на поверхности ошибок. Усредненные кривые процесса обучения показаны на рис. 3.15, *а–г* для различных значений параметра η. Показанные экспериментальные кривые отражают следующие тенденции обучения.

1. В общем случае малые значения параметра η обеспечивают более медленную сходимость. При этом более точно определяется точка локального минимума на поверхности ошибок. Это интуитивно понятно, так как при меньших значениях η поиск минимума выполняется в более широкой области поверхности ошибок.

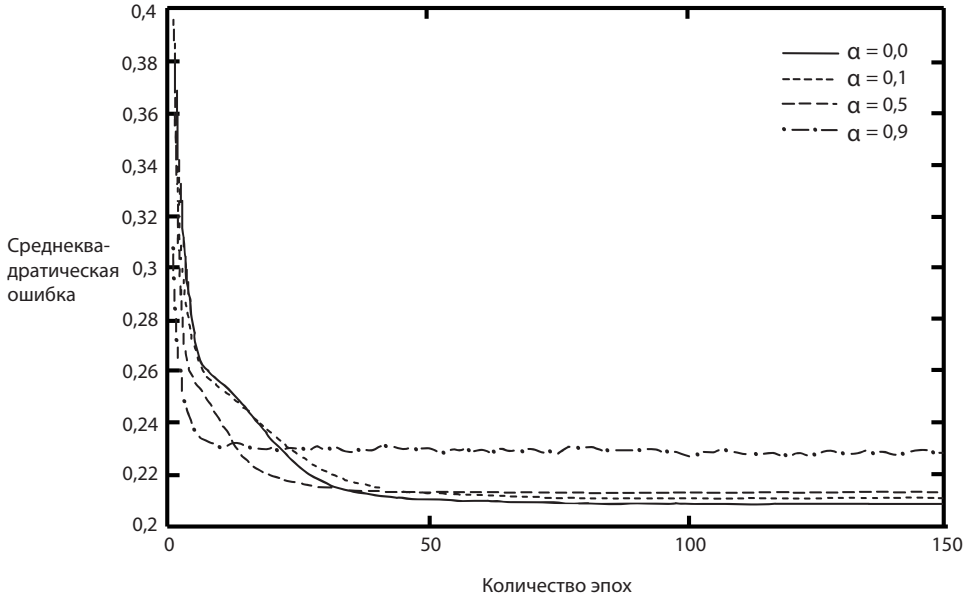
2. При $\eta \to 0$ выбор больших значений константы момента ($\alpha \to 1$) приводит к увеличению скорости сходимости. С другой стороны, при $\eta \to 1$ малые значения фактора момента $\alpha \to 0$ повышают устойчивость обучения.

3. Использование значений $\eta = \{0{,}5;\ 0{,}9\}$ и $\alpha = 0{,}9$ приводит к колебаниям среднеквадратической ошибки в процессе обучения и более высокому ее значению по завершении процесса сходимости. И то, и другое — нежелательно.

На рис. 3.16 показаны графики «наилучших» кривых обучения по каждой из групп, представленных на рис. 3.15, для выбора наилучшей кривой в смысле определения 1. На рисунке видно, что оптимальное значение параметра скорости обучения η_{opt} составляет порядка 0,1, а α_{opt} — около 0,5. В табл. 3.4 приведены оптимальные значения параметров сети, которые будут использоваться в оставшейся части эксперимента. Как видно на рис. 3.16, конечная среднеквадратическая ошибка мало отличается для различных кривых. Это значит, что поверхность ошибок в нашей задаче достаточно гладкая.



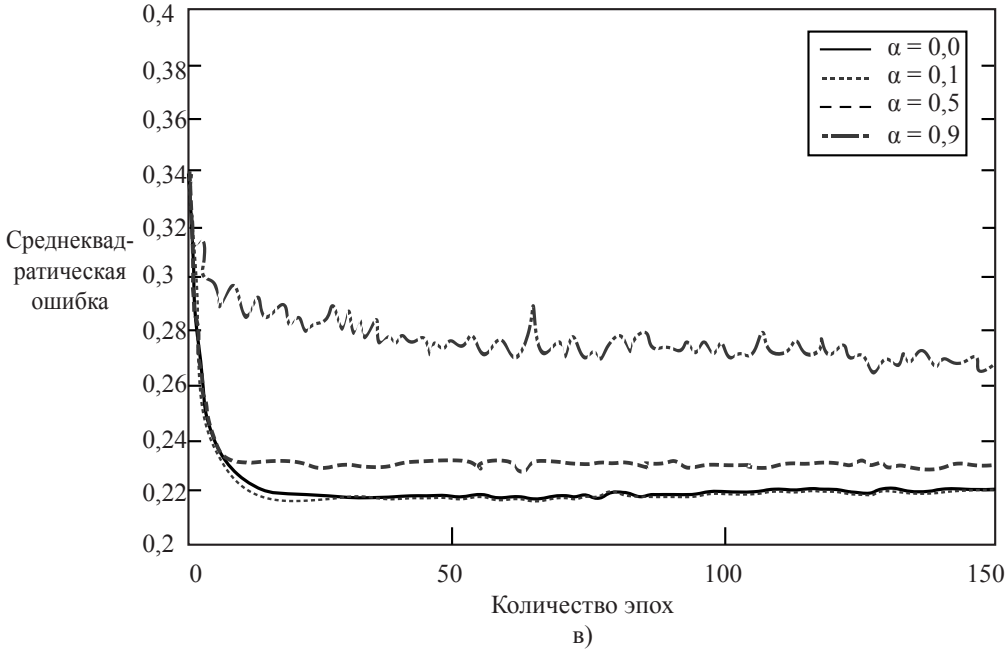
а)



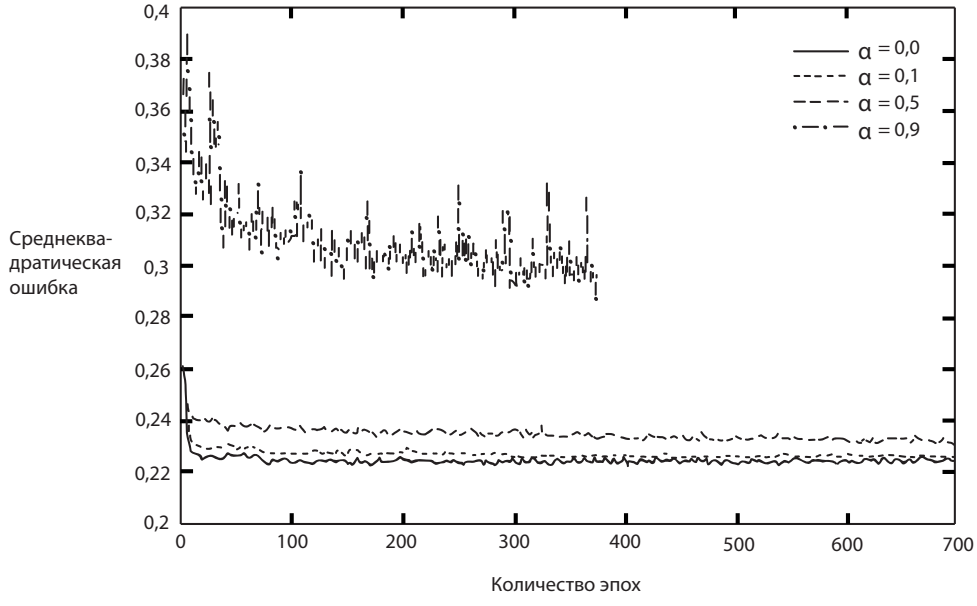
б)

Рис. 3.15. Усредненные кривые обучения для различных значений константы момента α и параметра скорости обучения:

η = 0,01 (*а*); η = 0,1 (*б*); η = 0,5 (*в*); η = 0,9 (*г*) (окончание см. на с. 103)



в)



г)

Рис. 3.15. Окончание (начало см. на с. 102)

Таблица 3.4

Конфигурация оптимизированного многослойного перцептрона

Параметр	Символ	Значение
Оптимальное число скрытых нейронов	m_{opt}	2
Оптимальный параметр скорости обучения	η_{opt}	0,1
Оптимальная константа момента	α_{opt}	0,5

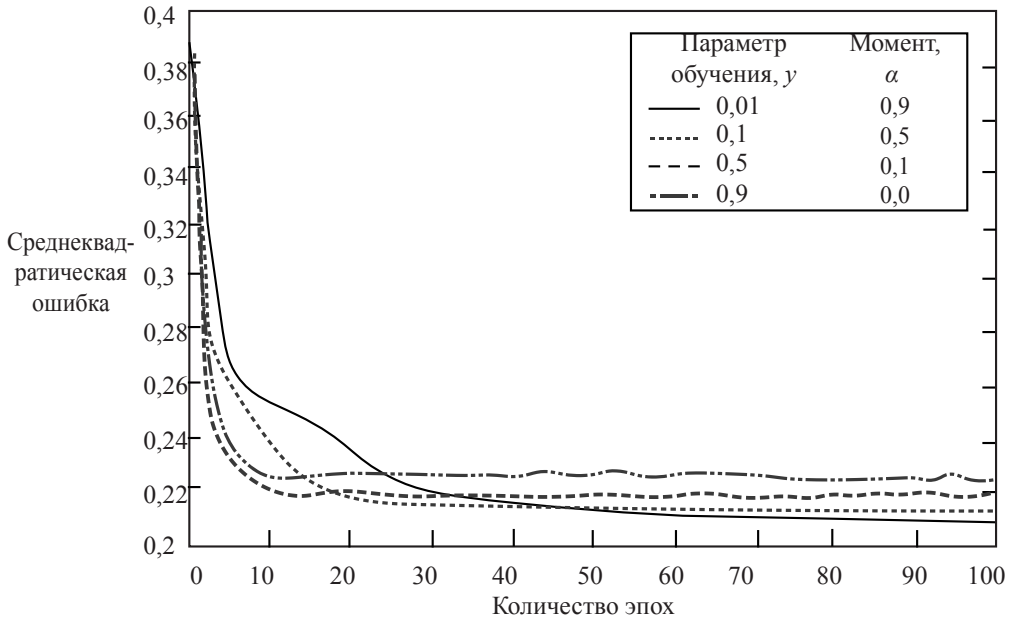


Рис. 3.16. Лучшие кривые обучения, выбранные на рис. 3.15

Оценка оптимальной архитектуры сети

Для «оптимизированного» многослойного перцептрона, параметры которого приведены в табл. 3.4, была исследована граница решений, построена усредненная по ансамблю кривая обучения и оценена вероятность корректной классификации. Если обучающее множество конечно, то функциональное преобразование сети, обученной при оптимальных значениях параметров, стохастично по своей природе. Поэтому показатели производительности усреднялись по двадцати независимо обучаемым сетям. Каждое из обучающих множеств состояло из 1000 примеров, выбранных из классов $\mathbf{C}_1$ и $\mathbf{C}_2$ с одинако-

вой вероятностью. Эти примеры предъявлялись сетям в случайной последовательности. Как и ранее, обучение выполнялось в течение 700 эпох. Для экспериментального определения вероятности корректной классификации использовалось сгенерированное ранее тестовое множество из 32000 примеров. На рис. 3.17, *а* показаны три «лучшие» границы решений для трех из 20 сетей. На рис. 3.17, *б* показаны три «наихудшие» границы решений для трех других сетей из этой же группы. На этих рисунках видно, что границы решений, построенные с помощью алгоритма обратного распространения, являются выпуклыми по отношению к области классификации вектора наблюдения **x**.

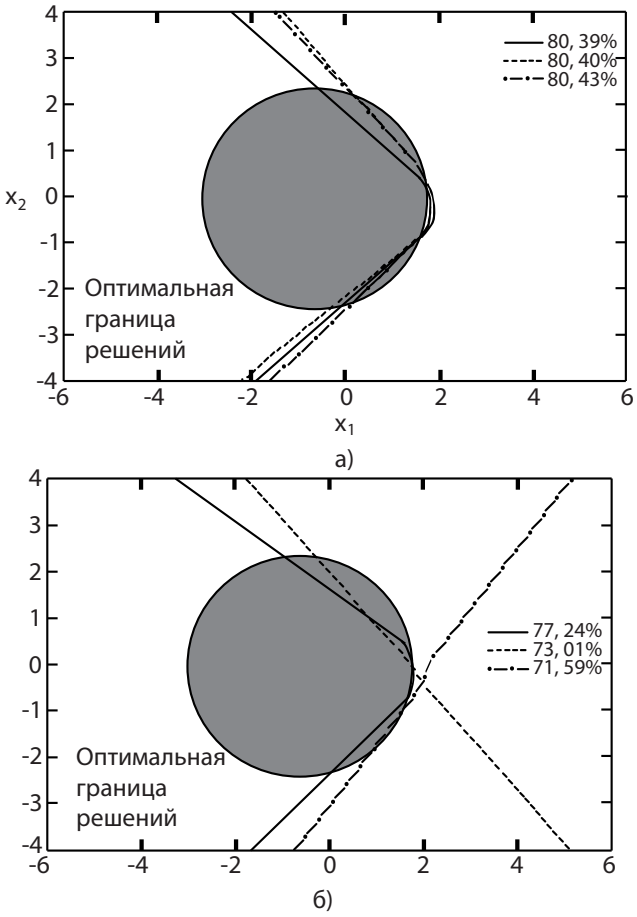


Рис. 3.17. Графики трех «лучших» (*а*) и трех «наихудших» (*б*) границ решений с точностью классификации 80,39; 80,40; 80,43 и 77,24; 73,01 и 71,59 % соответственно

Статистические данные о производительности, вероятности корректной классификации и конечной среднеквадратической ошибке, вычисленной для обучающего множества, приведены в табл. 3.5. Напомним, что вероятность корректной классификации для оптимального байесовского классификатора составляет 81,51 %.

Таблица 3.5

Обобщенные статистические показатели производительности

Показатель производительности	Среднее значение	Стандартное отклонение
Вероятность корректной классификации, %	79,70	0,44
Окончательная среднеквадратическая ошибка	0,2277	0,0118

3.9. Извлечение признаков

Скрытые нейроны (hidden neuron) играют крайне важную роль в работе многослойного перцептрона, обучаемого методом обратного распространения, поскольку они выступают в роли детекторов признаков (feature detector). В ходе обучения скрытые нейроны постепенно «выявляют» характерные черты данных обучения. Это осуществляют с помощью нелинейного преобразования входных данных в новое, скрытое пространство (hidden space) (или пространство признаков (feature space)). В новом пространстве классы (если взять для примера задачу классификации) легче отделить друг от друга, чем в исходном пространстве.

Чтобы перевести обсуждение в математический контекст, рассмотрим многослойный перцептрон с одним слоем, состоящим из m_1 нелинейных скрытых нейронов, и одним слоем линейных выходных нейронов размерности $m_2 = M$. Выбор линейных нейронов в выходном слое обусловлен необходимостью сфокусировать внимание только на роли скрытых нейронов в работе многослойного перцептрона. Синаптические веса сети нужно настроить так, чтобы минимизировать среднеквадратическую ошибку (по N образам) между целевым выходом (желаемым откликом) и фактическим выходным сигналом сети, генерируемым в ответ на m_0-мерный вектор входного сигнала. Пусть $z_j(n)$ — выходной сигнал скрытого нейрона j, генерируемый в ответ на пред-

ставление n-го входного вектора. Он является нелинейной функцией образа (вектора), подаваемого на входной слой сети. С каждым нейроном связана сигмоидальная функция активации. Выходной сигнал нейрона k выходного слоя описывается выражением

$$y_k(n) = \sum_{j=0}^{m_1} \omega_{kj} z_j(n),\ k = 1, 2, \ldots, M;\ n = 1, 2, \ldots, N, \tag{3.69}$$

где w_{k0} — смещение, связанное с нейроном k. Функция стоимости, которую следует минимизировать, записывается в виде

$$E_{\text{av}} = \frac{1}{2N} \sum_{n=1}^{N} \sum_{k=1}^{M} \left(d_k(n) - y_k(n)\right)^2. \tag{3.70}$$

Заметим, что здесь подразумевается использование пакетного режима обучения. Используя (3.69) и (3.70), можно довольно легко переписать функцию стоимости в более компактной матричной форме:

$$E_{\text{av}} = \frac{1}{2N} \left\| \tilde{\mathrm{D}} - \mathrm{W}\tilde{\mathrm{Z}} \right\|^2, \tag{3.71}$$

где $\mathbf{W}$ — матрица синаптических весов размерности $M \times m_1$, относящаяся к выходному слою сети. Матрица $\tilde{\mathrm{Z}}$ выходов скрытых нейронов (за вычетом их среднего значения) имеет размерность $m_1 \times N$. Эта матрица состоит из сигналов, полученных в результате обработки N входных образов (входных сигналов), т. е.

$$\tilde{\mathrm{Z}} = \left\{ \left(z_j(n) - \mu_{zj} \right);\ j = 1, 2, \ldots, m_1;\ n = 1, 2, \ldots, N \right\},$$

где μ_z — среднее значение сигнала $z_j(n)$. Соответственно, матрица $\tilde{\mathrm{D}}$ желаемых откликов размерности $M \times N$, представляемых выходному слою сети, имеет вид

$$\tilde{\mathrm{D}} = \left\{ \left(d_k(n) - \mu_{dk} \right); k = 1, 2, \ldots, M;\ n = 1, 2, \ldots, N \right\},$$

где μ_{dk} — среднее значение $d_k(n)$. Минимизация функции E_{av}, определяемой выражением (3.70), — это линейная задача, решение которой описывается уравнением

$$\mathrm{W} = \bar{\mathrm{D}}\bar{\mathrm{Z}}^{+}, \tag{3.72}$$

где $\bar{\mathrm{Z}}^{+}$ — псевдообратная матрица для $\bar{\mathrm{Z}}$. Минимальное значение функции E_{av}, можно представить в виде:

$$E_{\text{av.min}} = \frac{1}{2N} tr\left[\tilde{\mathrm{D}}\tilde{\mathrm{D}}^T - \tilde{\mathrm{D}}\tilde{\mathrm{Z}}^T\left(\tilde{\mathrm{Z}}\tilde{\mathrm{Z}}^T\right) + \tilde{\mathrm{Z}}\tilde{\mathrm{D}}^T\right], \tag{3.73}$$

где $tr[\cdot]$ — оператор следа. Так как целевые образы, представленные матрицей $\bar{\mathrm{D}}$, фиксированы, задача минимизации функции стоимости E_{av} относительно синаптических весов многослойного перцептрона эквивалентна максимизации дискриминантной функции [36]:

$$D = tr\,[\mathbf{C}_b\mathbf{C}_t^+], \tag{3.74}$$

где матрицы $\mathbf{C}_b$ и $\mathbf{C}_t$ определяются следующим образом:

— $\mathbf{C}_t\,(m_1 \times m_1)$ — матрица общей ковариации (total covariance matrix) выходов скрытых нейронов при представлении N входных сигналов

$$\mathbf{C}_t = \bar{\mathrm{Z}}\bar{\mathrm{Z}}^T. \tag{3.75}$$

Матрица $\mathbf{C}_t^+$ является псевдообратной по отношению к матрице $\mathbf{C}_t$.

— Матрица $\mathbf{C}_b$ размерности $m_1 \times m_1$ определяется выражением

$$\mathbf{C}_b = \overline{\mathrm{ZD}}^T\,\overline{\mathrm{DZ}}^T. \tag{3.76}$$

Обратите внимание, что дискриминантная функция D согласно (3.74) определяется исключительно скрытыми нейронами многослойного перцептрона. Заметим также, что количество скрытых слоев, участвующих в нелинейном преобразовании и генерации дискриминантной функции, не ограничено. Если многослойный перцептрон содержит несколько скрытых слоев, то матрица $\bar{\mathrm{Z}}$ описывает все множество образов (сигналов) в пространстве, определенном последним слоем скрытых нейронов. Чтобы интерпретировать матрицу $\mathbf{C}_b$, рассмотрим частный случай схемы кодирования «один из M» (one-from-M coding scheme) [36]. Это значит, что k-й элемент целевого вектора (желаемого отклика) равен единице, если входной сигнал принадлежит классу k, и нулю в противном случае:

$$d(n) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow k\text{-й элемент, } d(n) \in C_k.$$

Таким образом, для M классов C_k, $k = 1, 2, ..., M$, к каждому из которых относится N_k образов, таких, что $\sum_{k=1}^{M} N_k = N$, матрицу $\mathbf{C}_b$ для данной схемы кодирования можно представить в виде

$$\mathbf{C}_b = \sum_{k=1}^{M} N_k^2 \left(\mu_{z,k} - \mu_z\right)\left(\mu_{z,k} - \mu_z\right)^T, \tag{3.77}$$

где вектор $\boldsymbol{\mu}_{z,k}$ размерности $m_1 \times 1$ является средним значением вектора выходов скрытых нейронов для N входных образов. В соответствии с (3.77) матрицу $\mathbf{C}_b$ можно интерпретировать как матрицу взвешенной межклассовой ковариации (weighted between-class covariance matrix) выходных сигналов скрытого слоя. Таким образом, для схемы кодирования «один из M» многослойный перцептрон максимизирует дискриминантную функцию, представляющую собой след произведения двух величин: матрицы взвешенной межклассовой ковариации и псевдообратной матрицы для матрицы общей ковариации. Из этого примера видно, что при обучении многослойного перцептрона методом обратного распространения в качестве предварительной информации учитываются пропорции образов в рамках отдельных классов.

Связь с линейным дискриминантом Фишера

Дискриминантная функция, определенная в (3.74), характерна лишь для многослойного перцептрона. Однако она тесно связана с линейным дискриминантом Фишера (Fisher's linear discriminant), который описывает линейное преобразование многомерной задачи в одномерную. Рассмотрим переменную y, представляющую собой линейную комбинацию элементов входного вектора $\mathbf{x}$. Более точно: пусть y является скалярным произведением вектора $\mathbf{x}$ и вектора настраиваемых весов $\mathbf{w}$ (содержащего в качестве первого элемента порог b): $y = \mathbf{w}^T\mathbf{x}$.

Вектор $\mathbf{x}$ выбирается из множества C_1 или C_2, которые, в свою очередь, отличаются векторами средних значений $\boldsymbol{\mu}_1$ и $\boldsymbol{\mu}_2$ соответственно. Критерий Фишера (Fisher's criterion), определяющий степень различия между двумя классами, задается следующим образом:

$$J(w) = \frac{w^T \mathbf{C}_b w}{w^T \mathbf{C}_t w},$$

где $\mathbf{C}_b$ — матрица межклассовой ковариации (between-class covariance matrix), а $\mathbf{C}_t$ — общая матрица внутриклассовой ковариации (within-class covariance matrix),

$$\mathbf{C}_b = \sum_{n \in C_1} (x_n - \mu_1)(x_n - \mu_1)^T + \sum_{n \in C_2} (x_n - \mu_2)(x_n - \mu_2)^T.$$

Матрица внутриклассовой ковариации $\mathbf{C}_t$ пропорциональна матрице ковариации обучающего множества. Это симметричная и неотрицательно определенная матрица, которая обычно является несингулярной, если размер множества обучения достаточно велик. Матрица межклассовой ковариации $\mathbf{C}_b$ также является симметричной и неотрицательно определенной, однако она сингулярна. Следует отметить, что матричное произведение $\mathbf{C}_b\mathbf{w}$ всегда направлено в сторону вектора разности между средними значениями $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$. Это свойство непосредственно следует из определения матрицы $\mathbf{C}_b$. Выражение, определяющее критерий Фишера $J(\mathbf{w})$, называют обобщенным фактором Рэлея (generalized Rayleigh quotient). Вектор $\mathbf{w}$, максимизирующий эту величину, должен удовлетворять условию

$$\mathbf{C}_b\mathbf{w} = \lambda\mathbf{C}_t\mathbf{w}. \tag{3.78}$$

Уравнение (3.78) описывает обобщенную задачу нахождения собственных чисел. В нашем случае матричное произведение $\mathbf{C}_b\mathbf{w}$ всегда направлено в сторону разности $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$, так что уравнение (3.78) достаточно легко разрешить относительно $\mathbf{w}$:

$$\mathbf{w} = \mathbf{C}_t^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2). \tag{3.79}$$

Это решение называется линейным дискриминантом Фишера (Fisher's linear discriminant) [37]. Возвращаясь к вопросу извлечения признаков, вспомним, что дискриминантная функция D, определяемая формулой (3.74), связывает матрицы межклассовой и общей ковариации образов, трансформированных в пространство скрытых нейронов сети. Дискриминантная функция D аналогична линейному дискриминанту Фишера. Именно поэтому нейронные сети так хорошо решают задачу классификации.

3.10. Обратное распространение ошибки и дифференцирование

Метод обратного распространения ошибки является специфической реализацией градиентного спуска (gradient descent) в пространстве весов многослойных сетей прямого распространения. Основная идея этого метода заключается в эффективном вычислении частных производных (partial derivative) функции сети $F(\mathbf{w}, \mathbf{x})$ по всем элементам настраиваемого вектора весов $\mathbf{w}$ для данного входного вектора $\mathbf{x}$. В этом и заключается вычислительная мощность алгоритма обратного распространения. Более полно описание обратного распространения ошибки для эффективной оценки градиента представлено в [38].

Для большей конкретизации рассмотрим многослойный перцептрон с входным слоем, состоящим из m_0 узлов, двумя скрытыми слоями и одним выходным нейроном (рис. 3.18). Элементы вектора весов $\mathbf{w}$ упорядочены по слоям (начиная с первого скрытого), затем по нейронам и, наконец, по синаптическим связям каждого нейрона.

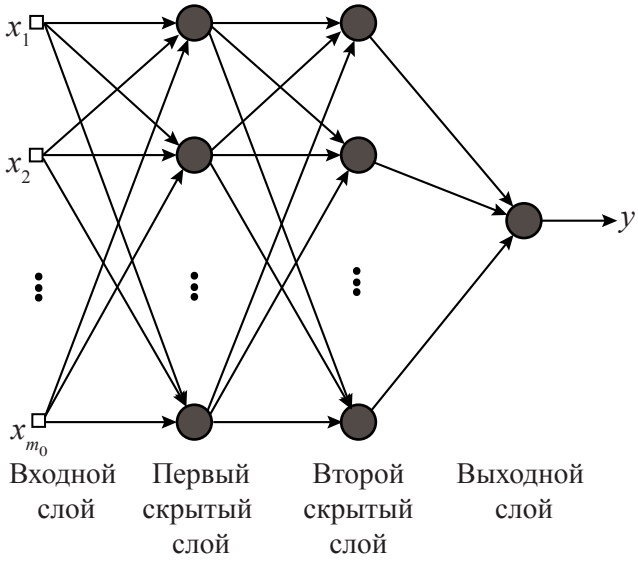


Рис. 3.18. Многослойный перцептрон с двумя скрытыми слоями и одним выходным нейроном

Пусть $w_{ji}^{(l)}$ — синаптический вес, связывающий нейрон i с нейроном j слоя $l = 0, 1, \ldots$. Для первого скрытого слоя ($l = 1$) индекс i относится не к нейрону, а к входному узлу. Для $l = 3$, что соответствует выходному слою, $j = 1$. Требуется вычислить производные функции $F(\mathbf{w}, \mathbf{x})$ по всем элементам вектора весов $\mathbf{w}$ для заданного входного вектора $\mathbf{x} = [x_1, x_2, \ldots, x_{mo}]^T$. Заметим, что для $l = 2$ (т. е. для второго скрытого слоя) функция $F(\mathbf{w}, \mathbf{x})$ имеет форму, аналогичную правой части выражения (3.69). Вектор весов $\mathbf{w}$ включен в список аргументов функции $F(\mathbf{w}, \mathbf{x})$, чтобы привлечь к нему внимание.

Многослойный перцептрон на рис. 3.18 определяется архитектурой $\mathbf{A}$ (представляющей собой дискретный параметр) и вектором весов $\mathbf{w}$ (составленным из вещественных элементов). Пусть $\mathbf{A}_j^{(l)}$ — часть архитектуры, включающая в себя фрагмент нейронной сети от входного слоя ($l = 0$) до узла j слоя l ($l = 1, 2, 3$). Соответственно, можно записать

$$F(\mathbf{w}, \mathbf{x}) = \varphi(\mathbf{A}_1^{(3)}), \tag{3.80}$$

где φ — функция активации. Однако $\mathbf{A}_1^{(3)}$ необходимо интерпретировать исключительно как символ обозначения архитектуры, а не переменную. Таким образом, адаптируя выражения (3.1), (3.2), (3.11) и (3.23) к данной ситуации, получим следующий результат:

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial \omega_{1k}^{(3)}} = \varphi'\left(A_1^{(3)}\right)\varphi\left(A_k^{(2)}\right), \tag{3.81}$$

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial \omega_{kj}^{(2)}} = \varphi'\left(A_1^{(3)}\right)\varphi'\left(A_k^{(2)}\right)\varphi\left(A_j^{(1)}\right)\omega_{1k}^{(3)}, \tag{3.82}$$

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial \omega_{ji}^{(1)}} = \varphi'\left(A_1^{(3)}\right)\varphi'\left(A_j^{(1)}\right)x_i\left[\sum_k \omega_{1k}^{(3)}\varphi'\left(A_k^{(2)}\right)\omega_{kj}^{(2)}\right], \tag{3.83}$$

где φ' — частная производная нелинейной функции φ по своим аргументам; x_i — i-й элемент входного вектора $\mathbf{x}$. Аналогично можно вывести выражения для частных производных любой общей сети с большим числом слоев и выходных нейронов. Выражения (3.81)–(3.83) являются основой для вычисления степени чувствительности функции сети $F(\mathbf{w}, \mathbf{x})$ к вариациям элементов вектора весов $\mathbf{w}$. Пусть ω — некоторый элемент вектора весов $\mathbf{w}$. Чувствительность (sensitivity) $F(\mathbf{w}, \mathbf{x})$ к элементу ω формально определяется выражением

$$S_{\omega}^{F} \frac{\partial F / F}{\partial \omega / \omega}, \omega \in w.$$

Именно по этой причине нижнюю часть графа передачи сигнала на рис. 3.7 мы назвали графом чувствительности (sensitivity graph).

Матрица якобиана

Пусть W — общее количество свободных параметров (т. е. синаптических весов и порогов) многослойного перцептрона, которые указанным выше способом формируют вектор весов **w**. Пусть N — общее количество примеров, использованных для обучения сети. Применяя метод обратного распространения, можно вычислить множество W частных производных аппроксимирующей функции $F[\mathbf{w}, \mathbf{x}(n)]$ по элементам вектора весов **w** для каждого примера **x** (n) из обучающего множества. Повторяя эти вычисления для $n = 1, 2, ..., N$, можно получить матрицу частных производных размерности $N \times W$. Эта матрица называется якобианом **J** многослойного перцептрона, вычисленным в точке **x**(n). Каждая строка якобиана соответствует одному примеру из обучающего множества. Экспериментально подтверждается, что многие задачи обучения нейросетей внутренне плохо обусловлены (ill-conditioned), что ведет к неполноте ранга якобиана **J** [39]. Ранг (rank) матрицы равен количеству ее линейно независимых строк или столбцов (наименьшему из них). Ранг якобиана считается неполным (rank deficient), если он меньше значения min (N, W). Неполнота ранга якобиана приводит к тому, что алгоритм обратного распространения получает только частичную информацию о возможных направлениях поиска, что, в свою очередь, значительно удлиняет время обучения.

4. Геостатистика. Гибридные методы на основе ИНС и геостатистики

Современная геостатистика — это широкий набор статистических моделей, используемый для анализа и обработки пространственно распределенной информации. Центральная идея геостатистики состоит в использовании знаний о пространственной корреляции экспериментальных данных для построения пространственных оценок и интерполяций. При решении задач пространственной интерполяции необходима гипотеза о пространственной стационарности функции распределения. Условие стационарности является строгим, поэтому на практике используются более мягкие условия стационарности второго порядка или внутренняя гипотеза (intrinsic hypothesis). Из внутренней гипотезы следует определение одного из основных понятий геостатистики — вариограммы. Вариограмма — ключевой инструмент для оценки степени пространственной корреляции, имеющейся в данных, и для ее моделирования. Модель вариограммы является функцией, определяющей зависимость изменения исследуемой величины в пространстве от расстояния [40].

Случайная функция $V(\mathbf{x})$ обладает стационарностью второго порядка, если:

— математическое ожидание $m(\mathbf{x})$ существует и не зависит от местоположения $\mathbf{x}$:

$$m(\mathbf{x}) = \{V(\mathbf{x})\} = \text{const}, \ \forall \mathbf{x};$$

— для каждой пары значений случайной переменной $\{V(\mathbf{x}), V(\mathbf{x}+\mathbf{h})\}$ ковариация существует и зависит только от разности координат $\mathbf{h}$:

$$C(h) = E\{V(\mathbf{x})V(\mathbf{x}+\mathbf{h})\} - m^2, \ \forall \mathbf{x}.$$

Таким образом, стационарность второго порядка — это стационарность только для моментов первого и второго порядка.

Случайная функция $V(\mathbf{x})$ удовлетворяет внутренней гипотезе, если:

— математическое ожидание $m(\mathbf{x})$ существует и не зависит от местоположения $\mathbf{x}$:

$$m(\mathbf{x}) = E\{V(\mathbf{x})\} = \text{const}, \ \forall \mathbf{x};$$

— для любого вектора $\mathbf{h}$ разность $V(\mathbf{x}) - V(\mathbf{x} + \mathbf{h})$ имеет конечную вариацию, не зависящую от $\mathbf{x}$ (стационарность приращений):

$$\text{Var}\{V(\mathbf{x}+\mathbf{h}) - V(\mathbf{x})\} = E\left\{V(\mathbf{x}+\mathbf{h}) - \left[V(\mathbf{x})\right]^2\right\} = 2\gamma(\mathbf{h}), \forall \mathbf{x}.$$

В рамках предположения о стационарности второго порядка, в частности, работает базовый метод геостатистики — кригинг (kriging) [41]. Модели из семейства кригинга — наилучшего линейного несмещенного оценивателя (Best Linear Unbiased Estimator — BLUE) — используются для получения наилучшей в статистическом смысле пространственной оценки. Кригинг является «наилучшим» оценивателем, так как его оценка обладает минимальной дисперсией. Важным свойством кригинга является точное воспроизведение значений измерений в имеющихся точках (свойства интерполятора). В отличие от многочисленных детерминистических методов оценка кригинга сопровождается оценкой ошибки интерполяции в каждой точке. Полученная ошибка позволяет охарактеризовать неопределенность интерполяционной оценки данных при помощи доверительных интервалов.

При применении любой модели интерполяции встает вопрос о подборе оптимальных модельно зависимых параметров. Даже в случае использования одного и того же метода интерполяции можно получить качественно разные результаты в зависимости от выбора модельных параметров. Выбор оптимальных параметров опирается на пошаговое исследование характера и структуры данных при помощи методов геостатистики. При проведении анализа на реальных данных часто возникает проблема недостаточного количества измерений по интересующей переменной, например вследствие их дороговизны или невозможности взятия проб. При этом в наличии имеется избыточное количество измерений переменной, которая достаточно сильно коррелированна с основной. Эту информацию можно использовать для улучшения оценки переменной, информация по которой труднодо-

ступна. В рамках многопеременной геостатистики существует модель совместной пространственной интерполяции нескольких коррелированных переменных — кокригинг. Кокригинг позволяет значительно улучшить качество оценки, перейти из области экстраполяции в область интерполяции, уменьшить ошибку оценки за счет использования дополнительной информации по коррелированным переменным [40].

Все модели семейства кригинга сводятся к линейной регрессионной оценке:

$$V(x) - m(x) = \sum_{i-1}^{n} w_i(x)\left[V(x_i) - m(x_i)\right], \tag{4.1}$$

где $w_i(x)$ — веса, присваиваемые данным $V(x_i)$, которые в свою очередь являются реализациями пространственной переменной V. Значения $m(x)$ и $m(x_i)$ являются математическими ожиданиями (средними) пространственных переменных $V(x)$ и $V(x_i)$. Количество данных n, используемых для оценивания, как и их веса, могут меняться в зависимости от точки оценивания x.

Будем считать, что модель поведения пространственных данных строится в виде неизвестной случайной функции V, которая является функцией случайных переменных $\{V(x), V(x_1), ..., V(x_n)\}$. При этом значения $V(x_1), ..., V(x_n)$ заданы в виде исходных данных и представляют собой единственную реализацию случайных переменных в соответствующих точках пространства, а значение $V(x)$ неизвестно.

Случайная функция $V(x)$ обычно раскладывается на две компоненты — детерминистический тренд $m(x)$ и случайную невязку $R(x)$:

$$V(x) = R(x) + m(x). \tag{4.2}$$

Компонента невязки $R(x)$ моделируется как стационарная случайная функция с нулевым математическим ожиданием $m_R(x)$ и ковариацией $C_R(h)$:

$$m_R(x) = E\{R(x)\} = 0; \tag{4.3}$$

$$Cov\{R(x), R(x+h)\} = E\{R(x)R(x+h)\} = C_R(h). \tag{4.4}$$

Математическое ожидание пространственной переменной V в точке x, таким образом, будет равно значению тренда:

$$E\{V(x)\} = m(x). \tag{4.5}$$

Первое условие, которому удовлетворяют все модели кригинга — это несмещенность оценки (4.1) в точке x (среднее значение ошибки оценивания равно нулю):

$$R^*(x) = V^*(x) - V(x); \tag{4.6}$$

$$E\{R^*(x)\} = 0. \tag{4.7}$$

Здесь $V^*(x)$ — это оценка значения $V(x)$, которое неизвестно.

Следующим условием, которое используется для получения уравнений кригинга, является условие минимизации вариации ошибки, что дает «наилучшую» в статистическом смысле оценку:

$$\sigma_R^2(x) = E\left\{\left(R^2(x) - m_R(x)\right)^2\right\}. \tag{4.8}$$

Различают три типа кригинга в зависимости от модели тренда $m(x)$.

1. Простой кригинг (Simple Kriging, SK) предполагает среднее $m(x)$ постоянным и известным в области исследования S:

$$m(x) = m = \text{const, известно } \forall\ x \in S. \tag{4.9}$$

2. Обычный кригинг (Ordinary Kriging, OK) Значение среднего в области неизвестно, но постоянно:

$$m(x) = m = \text{const, неизвестно } \forall\ x \in A(x). \tag{4.10}$$

3. Универсальный кригинг (Universal Kriging, UK) предполагает, что неизвестное среднее значение $m(x)$ гладко меняется во всей области исследования S. Компонента тренда моделируется как линейная комбинация известных функций $f_k(x)$:

$$m(x) = \sum_{k=0}^{K} a_k(x) f_k(x), \tag{4.11}$$

где коэффициенты $a_k(x)$ определяются из системы уравнений универсального кригинга.

Обычный кригинг — это один из наиболее часто используемых в геостатистике методов интерполяции. Этот метод имеет свойство «наилучшего линейного несмещенного оценивателя». Как следует из (4.1), метод является линейным. Несмещенность обеспечивается условием (4.7). Термин «наилучший» означает, что веса w_i в (4.1) выбираются так, чтобы минимизировать вариацию ошибки σ_R^2 (4.8).

Условие (4.10) позволяет переписать выражение (4.1) в другом виде:

$$V^*(x) = \sum_{i=1}^{n} w_i * V(x_i) + \left[1 - \sum_{i=1}^{n} w_i\right] * m. \tag{4.12}$$

Если подставить (4.12) в условие несмещенности (4.7), получим:

$$E\{V^*(x) - V(x)\} = E\left\{\sum_{i=1}^{n} w_i * V(x_i)\right\} - E\{V(x)\} - \left[1 - \sum_{i=1}^{n} w_i\right] * m \equiv 0. \tag{4.13}$$

Условие несмещенности для уравнения (4.12) выполнено автоматически, если сумма весов равняется единице:

$$1 - \sum_{i=1}^{n} w_i = 0. \tag{4.14}$$

Таким образом, оценка обычного кригинга вычисляется по формуле:

$$V^*(x) = \sum_{i=1}^{n} w_i * V(x_i). \tag{4.15}$$

Веса w_i должны быть вычислены при ограничении (4.14).

Теперь обратимся к анализу вариации ошибки σ_R^2 для обычного кригинга. По определению в силу условия (4.7):

$$\mathrm{Var}\{R(x)\} = E\left\{\left(R(x) - m_R(x)\right)^2\right\} = E\{R^2(x)\} - mR(x) = E\{R(x)\}.$$

Тогда:

$$\begin{aligned} E\{R^2(x)\} &= E\left\{\left[V^*(x) - V(x_0)\right]^2\right\} = E\left\{\left[\left(V^*(x) - m\right) - \left(V(x) - m\right)\right]^2\right\} = \\ &= \mathrm{Var}\{V^*(x)\} - 2\mathrm{Cov}\{V^*(x)V(x)\} + \mathrm{Var}\{V(x)\}. \end{aligned} \tag{4.16}$$

При условии, что функция V удовлетворяет внутренней гипотезе, все написанные здесь ковариации существуют. Используя (4.14) и (4.15), получим:

$$\mathrm{Var}\{V^*(x)\} = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j C_{ij}.$$

Вариация неизвестной случайной переменной $V(x)$ также существует и равна априорной вариации исходных данных:

$$\mathrm{Var}\{V^*(x)\} = \sigma^2 = \mathrm{const}.$$

Второе слагаемое в (4.16) с учетом (4.14) и (4.15) равно:

$$2\mathrm{Cov}\left\{V^*(x)V(x)\right\} = 2\sum_{i=1}^{n} w_i C_{i0}.$$

В итоге получим:

$$\sigma_R^2 = \sigma^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j C_{ij} - 2\sum_{i=1}^{n} C_{i0}, \tag{4.17}$$

где C_{ij} — ковариации случайных переменных $\mathrm{Cov}\{V_i V_j\}$, $i, j = 1, ..., n$. Теперь, чтобы сделать простой кригинг «наилучшим» в статистическом смысле, нужно подобрать веса w_i так, чтобы они минимизировали (4.17) при ограничении (4.14). Решение такой задачи стандартно и получается образованием из (4.17) лагранжиана путем включения в него условия (4.14) с весом μ. Вес μ называется множителем Лагранжа:

$$L = \sigma^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j C_{ij} - 2\sum_{i=1}^{n} C_{i0} - 2\mu\left(\sum_{i=1}^{n} w_i - 1\right). \tag{4.18}$$

Для минимизации лагранжиана (4.18) необходимо вычислить частные производные от него по всем весам w_i и по μ и приравнять их к нулю. В результате получается следующая система из $n+1$ уравнений обычного кригинга (ordinary kriging):

$$\sum_{j=1}^{n} w_j C_{ij} + \mu = C_{i0}\ , \forall\, i = 1, ..., n$$

$$\sum_{i=1}^{n} w_i = 1. \tag{4.19}$$

Система уравнений решается относительно w_i и μ. Далее найденные веса w_i используются для получения оценки линейной регрессии. Вариация ошибки оценки вычисляется путем умножения n уравнений на w_i и их суммирования:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j C_{ij} = \sum_{i=1}^{n} w_i C_{i0} - \mu.$$

Подставив это в (4.17), получим итоговое выражение для вариации ошибки обычного кригинга:

$$\sigma_R^2 = \sigma^2 - \sum_{i=1}^{n} w_i C_{i0} - \mu. \tag{4.20}$$

Для того чтобы решить систему уравнений кригинга, необходимо знание значений ковариаций (или вариограмм) для всех пар измерений (*ij*) и их пар с точкой оценивания (i_0). Таким образом, в общем случае для вычисления оценки кригинга в произвольных точках пространства требуется знать значения ковариации $C(\mathbf{h})$ (или вариограммы $\gamma(\mathbf{h})$) для любого вектора $\mathbf{h}$ в анализируемой области.

Применение различных детерминистических или геостатистических моделей интерполяции/оценивания всегда дает единственное и сглаженное, не воспроизводящее изначальную вариабельность данных, значение оценки в интересующей точке при выбранных модельных параметрах. Стохастическое моделирование является альтернативным подходом, дающим возможность воспроизвести исходную вариабельность и получить сколь угодно много равновероятных реализаций пространственной функции в области. Равновероятные реализации позволяют описать пространственную вариабельность (изменчивость) и неопределенность пространственной функции, оценить вероятности и риск. При использовании стохастического моделирования удается избежать сглаженной картины оценки, которая присуща большинству моделей интерполяции. Это позволяет получать корректные результаты в таких задачах, как, например, расчет объема резервуара, «длины» береговой линии и т. п.

Еще одной проблемой, часто осложняющей проведение анализа пространственных данных, является пространственная нестационарность. Поведение данных в природе обычно зависит от множества различных факторов. Это приводит к появлению разномасштабных пространственных структур. Проблема моделирования и удаления крупномасштабного тренда из данных решается различными способами. Одним из эффективных подходов представляется применение искусственных нейронных сетей (ИНС). В процессе обучения ИНС адаптируются к исходным данным и хорошо моделируют крупномасштабные нелинейные эффекты. Смешанные модели ИНС в сочетании с геостатистикой продемонстрировали свою высокую эффективность по сравнению с другими существующими методами на различных данных, имеющих сложный пространственный характер (нестационарность, периодичность, пятнистость). Идея метода заключается в моделировании нелинейного крупномасштабного тренда при помощи ИНС и последующего моделирования невязок геостатистическими методами. Этот подход был впервые предложен М. Ф. Каневским [42]. Преи-

мущество использования ИНС для моделирования тренда перед другими моделями тренда (полиномы, сплайн и др.) заключается в том, что ИНС является универсальным оценивателем и хорошо моделирует нелинейные структуры. ИНС не предполагает фиксированной аналитической зависимости, а, наоборот, способна получить эту зависимость на основе имеющихся данных в процессе обучения. В качестве ИНС может использоваться как наиболее популярный многослойный перцептрон, так и более сложные нейронные сети (обобщенной регрессии, радиальных базисных функций). Ключевым моментом является анализ и моделирование корреляционной структуры невязок, оставшихся после вычета из данных оценки ИНС.

При анализе невязок — разницы между данными и оценками ИНС — возможно несколько вариантов. Если невязки не обладают пространственной корреляцией, а распределены совершенно случайно, это может означать, что ИНС полностью промоделировала структуру данных. В этом случае оценку ИНС можно принять как окончательную. Если невязки имеют пространственную структуру, а также коррелированы с исходными данными, необходимо проводить дальнейшее моделирование невязок. Обычно корреляция невязок и исходных данных слабее, чем корреляция данных с оценками ИНС (в случае корректного обучения и использования ИНС). Можно видеть, что невязки обладают пространственной корреляцией на меньших расстояниях, чем данные. Это обусловлено тем, что ИНС уже промоделировала корреляцию на более крупных масштабах. Это свойство невязок часто позволяет предположить их стационарность на всей области исследования, чего нельзя было предположить для исходных данных с трендом. Таким образом, вариограммная модель для невязок отличается коротким радиусом и стабилизированным плато. Использование такой модели в кригинге дает корректные и точные результаты. После выделения тренда простой или обычный кригинг используется на невязках модели к измеренным значениям поля.

Кригинг невязок, как и универсальный кригинг, предполагает, что неизвестное среднее значение $m(x)$ меняется во всей области исследования S так, что нельзя допустить постоянство даже локального среднего. В этом случае компонента тренда моделируется отдельно, используя другие математические или физические подходы [40].

Процедура оценки кригингом невязок предсказанных значений ИНС начинается с получения данных, предсказанных нейронной се-

тью. Невязка нейронной сети может быть определена следующим образом:

$$r(x_i) = Z(x_i) - \widehat{Z}_{ANN}(x_i),$$

где $r(x_i)$ — невязка нейронной сети на площадке x_i; $Z(x_i)$ — измеренное значение; $\widehat{Z}_{ANN}(x_i)$ — значение, предсказанное ИНС. Новая переменная $r(x_i)$ сохраняет пространственную изменчивость, определяемую внутренними факторами. Учитывая невязки $r(x_1)$, $r(x_2)$, ..., $r(x_n)$ на узлах x_1, x_2, ..., x_n, значение случайной величины r оценивается наиболее распространенным на практике обычным кригингом:

$$\hat{r}_{OK}(x_0) = \sum_{i=1}^{n} \lambda_i r(x_i),$$

где $\hat{r}_{OK}(x_0)$ представляет собой расчетную величину r на площадке x_0, выполненную обычным кригингом; λ_i — оптимальный вес при условии, что $\sum \lambda_i = 1$ и $r(x_i)$ — невязка нейронной сети на площадке x_i. Пространственные структуры обычно описываются с помощью экспериментальной вариограммы $\hat{\gamma}(h)$, которая измеряет среднее несходство между данными, разделенными расстоянием h. Рассчитывается как половина среднего квадрата разности между компонентами пар данных:

$$\hat{\gamma}(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} \left[r(x_i) - r(x_i + h) \right]^2,$$

где $N(h)$ — число пар участков отбора проб, разделенных расстоянием h. Эта модель используется для интерполяции ИНС остатков кригингом. Окончательное содержание моделируемого компонента $\widehat{Z}(x_i)$ есть сумма значения предсказанного ИНС $\widehat{Z}_{ANN}(x_i)$ и невязка нейронной сети, полученная обычным кригингом $\hat{r}_{OK}(x_i)$:

$$\widehat{Z}(x_i) = \widehat{Z}_{ANN}(x_i) + \hat{r}_{OK}(x_i).$$

Применение гибридных моделей на основе ИНС и геостатистики улучшает точность предсказания пространственно распределенных данных, особенно при значительной неоднородности (гетерогенности) последних.

Список библиографических ссылок

1. Dember W. N., Jenkins J. J., Teyler T. J. General Psychology. Hillsdale, NJ: Lawrence Erlbaum Associates, 1984. 915 p.

2. McCulloch W. S., Pitts W. A logical calculus of the ideas immanent in nervous activity // Bulletin of Mathematical Biophysics. 1943. Vol. 5. P. 115–133.

3. Hebb D. O. The Organization of Behavior: A Neuropsychological Theory. New York : Wiley, 1949.

4. Rosenblatt F. The Perceptron: A probabilistic model for information storage and organization in the brain // Psychological Review. 1958. Vol. 65. P. 386–408.

5. Rosenblatt F. Principles of Neurodynamics. Washington, DC : Spartan Books, 1962.

6. Bertsekas O. P. Dynamic Programming and Optimal Control: vol. I and vol. II. Belmont, MA : Athenas Scientific, 1995.

7. Powell M. J. D. Radial basis function aproximatiol1s to polynomials // Numerical Analysis 1987 Proceedings. Dundee, UK, 1988. P. 223–241.

8. Gray R. M., Davisson L. D. Random Processes: A Mathematical Aproach for Engineers. Englewood Cliffs, NJ: Prentice-Hall, 1986.

9. Haykin S. Adaptive Filter Theory. 3rd ed. Englewood Cliffs, NJ : Prentice-Hall, 1996.

10. Swindlehurst A. L., Goris M. J., Ottersten B. Some experiments with array data collected in actual urban and suburban environments // IEEE Workshop on Signal Processing Advances in Wireless Communications. Paris, France, 1997. P. 301–304.

11. Lipmann R. P. An introduction to computing with neural nets // IEEE ASSP Magazine. 1987. Vol. 4. P. 4–22.

12. Van Trees H. L. Detection, Estimation and Modulation Theory. New York : Wiley, 1968. Part 1.

13. Shynk J. J. Performance surfaces of a single-layer perceptron // IEEE Transactions on Neural Networks. 1990. Vol. 1. P. 268–274.

14. Minsky M. L., Selfridge O. G. Learning in random nets // Information Theory, Fourth London Symposium. London: Butterworths, 1961.

15. Minsky M. L., Papert S. A. Perceptrons, expanded edition. Cambridge, MA : MIT Press, 1988.

16. Minsky M. L., Papert S. A. Perceptrons. Cambridge, MA : MIT Press, 1969.

17. Parallel Distributed Processing: Explorations in the Microstructure of Cognition / D. E. Rumelhart, J. L. McClelland, eds. Cambridge, MA : MIT Press, 1986. Vol. 1.

18. Characterization of a class of sigmoid functions with applications to neural networks / A. Mennon, K. Mehrotra, C. K. Mohan, S. Ranka // Neural Networks. 1996. Vol. 9. P. 819–835.

19. Hinton G. E. Deterministic Boltzmann machine learning performs steepest descent in weight-space // Neural Computation. 1989. Vol. 1. P. 143–150.

20. Rumelhart O. E., Hinton G. E., Williams R. J. Learning representations of back-propagation errors // Nature (London). 1986. Vol. 323. P. 533–536.

21. Roy S., Shynk J. J. Analysis of the momentum LMS algorithm // IEEE Transactions on Acoustics, Speech and Signal Processing. 1990. Vol. ASSP-38. P. 2088–2098.

22. Hagiwara M. Theoretical derivation of momentum term in back-propagation // International Joint Conference on Neural Networks. 1992. Vol. 1. P. 682–686. Baltimore.

23. Jacobs R. A. Increased rates of convergel1ce through 1earning rate adaptation // Neura1 Networks. 1988. Vol. 1. P. 295–307.

24. Watrous R. L. Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization // First IEEE Informational Conference on Neural Networks. San Diego, CA. 1987. Vol. 2. P. 619–627.

25. Kramer A. H., Sangiovanni-Vincentelli A. Efficient paralle1leaming algorithms for neural networks // Advances in neural Information Processing Systems. San Mateo, CA : Morgan Kaufmann. 1989. Vo1. 1. P. 40–48.

26. Bertsekas D. P. NonlinearProgramming. Belmont, MA : Athenas Scientific, 1995.

27. Narendra K. S., Parthasarathy K. Identification and control of dynamical systems using neura1 networks // IEEE Transactions on Neural Networks. 1990. Vol. 1. P. 4–27.

28. Touretzky D. S., Pomerleau O. A. What is hidden in the hidden layers? // Byte. 1989. Vol. 14. P. 227–233.

29. LeCun Y. Efficient Learning and Second-order Methods: A Tutorial at NIPS 93. Denver, 1993.

30. Abu-Mostafa Y. S. Hints // Neural computation. 1995. Vol. 7. P. 639–671.

31. White H. Learning in artificial neural networks: A statistical perspective // Neural Computation. 1989. Vol. 1. P. 425–464.

32. Hampshire J. B., Pearlmutter B. Equivalence proofs for multilayer perceptron classifiers and Bayesian discriminant function // Proceedings of the 1990 Connectionist Models Summer School. San Mateo, CA : Morgan Kaufmann, 1990. P. 159–172.

33. Lui H. C. Analysis of decision contour of neural network with sigmoidal nonlinearity // International Joint Conference on Neural Networks. Washington, DC. 1990. Vol. 1. P. 655–659.

34. Devroye L. Exponential inequalities in nonparametric estimation // Nonparametric Functional Estimation and Related Topics / G. Roussas, ed. Boston : Kluwer. 1991. P. 31–44.

35. Luo Z. On the convergence of the LMS algorithm with adaptive learning rate for linear feedforward networks // Neural Computation. 1991. Vo1. 3. P. 226–245.

36. Webb A. R., Lowe O. The optimal internal representation of multilayer classifier networks performs nonlinear discriminant analysis // Neural Networks. 1990. Vol. 3. P. 367–375.

37. Duda R. O., Hart P. E. Pattern Classification and Scene Analysis. New York : Wiley, 1973.

38. Suga N. Cortical computational maps for auditory imaging // Neural Networks. 1990. Vol. 3. P. 3–21.

39. Saarinen S., Bramley R., Cybenko G. The numerical solution of neural network training problems // CRSD Report № 1089, Center for Supercomputing Research and Development, University of Illinois, Urbana, IL, 1991.

40. Демьянов В. В., Савельева Е. А. Геостатистика: теория и практика / под ред. Р. В. Арутюняна. Институт проблем безопасного развития атомной энергетики РАН: Наука, 2010. 327 с.

41. Матерон Ж. Основы прикладной геостатистики. М. : Мир, 1968. 407 с.

42. Элементарное введение в геостатистику / М. Каневский [и др.]. М., 1999. 136 с. (Проблемы окружающей среды и природных ресурсов / ВИНИТИ ; № 11).

Оглавление

Учебное издание

Сергеев Александр Петрович
Тарасов Дмитрий Александрович

ВВЕДЕНИЕ В НЕЙРОСЕТЕВОЕ МОДЕЛИРОВАНИЕ

Редактор Т. Е. Мерц
Верстка О. П. Игнатьевой

Подписано в печать 28.06.2017. Формат 70×100/16.
Бумага писчая. Печать цифровая. Гарнитура Newton.
Уч.-изд. л. 6,3. Усл. печ. л. 10,3. Тираж 50 экз.
Заказ 155

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ
620049, Екатеринбург, ул. С. Ковалевской, 5
Тел.: 8(343)375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

Отпечатано в Издательско-полиграфическом центре УрФУ
620075, Екатеринбург, ул. Тургенева, 4
Тел.: 8 (343) 350-90-13, 358-93-06, 350-58-20
Факс: 8 (343) 358-93-06
E-mail: press-urfu@mail.ru
Сайт: print.urfu.ru

СЕРГЕЕВ АЛЕКСАНДР ПЕТРОВИЧ

Кандидат физико-математических наук, доцент департамента информационных технологий и автоматики ИРИТ — РТФ УрФУ; заведующий лабораторией Института промышленной экологии УрО РАН.

Область научных интересов: искусственные нейронные сети, нейронаука, геостатистика.

ТАРАСОВ ДМИТРИЙ АЛЕКСАНДРОВИЧ

Старший преподаватель департамента информационных технологий и автоматики ИРИТ — РТФ УрФУ; научный сотрудник Института промышленной экологии УрО РАН.

Область научных интересов: искусственные нейронные сети, моделирование сложных систем, оптическая спектроскопия бумаги, светотехника, цифровое репродуцирование.