

КЛАССИЧЕСКОЕ УНИВЕРСИТЕТСКОЕ ИЗДАНИЕ

---

Серия основана в 2010 году



КЛАССИЧЕСКОЕ УНИВЕРСИТЕТСКОЕ ИЗДАНИЕ

---

Редакционный совет серии:

Председатель совета  
ректор Белорусского  
государственного университета  
*C. B. Абламейко*

Члены совета:

*C. Н. Ходин* (зам. пред.), *Н. Н. Герасимович* (отв. секретарь),  
*М. А. Журавков*, *А. Г. Кохановский*, *И. С. Ровдо*, *Н. В. Клебанович*,  
*В. В. Лысак*, *О. М. Самусевич*, *О. А. Ивашкевич* (зам. пред.),  
*В. М. Анишик*, *П. А. Мандрик*

В. А. Головко  
В. В. Краснопрошин

НЕЙРОСЕТЕВЫЕ  
ТЕХНОЛОГИИ  
ОБРАБОТКИ ДАННЫХ

*Допущено*  
Министерством образования Республики Беларусь  
в качестве учебного пособия для студентов  
учреждений высшего образования  
по специальностям «Информатика»,  
«Прикладная информатика»



МИНСК  
БГУ  
2017

УДК 0048.032.26(075.8)  
ББК 32.813я73-1  
Г61

Р е ц е н з е н т ы:  
кафедра электронных вычислительных машин  
Белорусского государственного  
университета информатики и радиоэлектроники  
(заведующий кафедрой кандидат технических наук,  
доцент *Д. И. Самаль*);  
доктор технических наук, профессор *А. А. Дудкин*

**Головко, В. А.**

Г61 Нейросетевые технологии обработки данных : учеб. пособие /  
В. А. Головко, В. В. Краснопрошин. – Минск : БГУ, 2017. – 263 с. –  
(Классическое университетское издание).  
ISBN 978-985-566-467-4.

Изложены математические и алгоритмические аспекты функционирования искусственных нейронных сетей. Детально проанализированы как конвенциональные модели нейронных сетей, так и глубокие нейронные сети. Рассмотрены парадигмы обучения нейронных сетей. Большое внимание уделено применению нейронных сетей для решения различного рода задач.

Для студентов учреждений высшего образования, обучающихся по специальностям «Информатика», «Прикладная информатика».

**УДК 0048.032.26(075.8)  
ББК 32.813я73-1**

**ISBN 978-985-566-467-4**

© Головко В. А.,  
Краснопрошин В. В., 2017  
© БГУ, 2017

## Уважаемые читатели!

Серия «Классическое университетское издание» была основана в 2010 году к 90-летию Белорусского государственного университета. Путь, который прошло наше учебное заведение в своем развитии, свидетельствует о становлении в нем собственной академической и научной традиции. Несомненно, опыт и знания, аккумулированные в стенах БГУ, являются не только предметом нашей гордости, но и достоянием всего белорусского общества. Одна из целей предлагаемой серии – сделать это достояние как можно более открытым и доступным.

Белорусский государственный университет всегда славился академичностью и фундаментальностью в подготовке специалистов. Однако сегодня этого уже недостаточно. От выпускника требуется умение быстро включаться в непосредственную практическую работу, которой свойствен синтез нескольких форм деятельности: собственно производственной, исследовательской, проектно-разработческой. В выигрыше в конечном итоге окажется тот, кто сегодня научится более эффективно создавать и применять знания, оперативно изменять технологии, совершенствовать и радикально трансформировать накопленный опыт. Вот почему совмещение преимуществ фундаментального и прагматического образования стало основой инновационно ориентированной подготовки будущих специалистов в нашем университете.

Серия отражает многолетний опыт научно-педагогической, методической и издательской работы БГУ. Ее цель – представить модель учебного текста, которая в своей структуре содержит набор программ образовательно-научно-производственной деятельности будущих специалистов. Реализация этой модели позволит обеспечить универ-

сализм выпускника, его способность к эффективному решению важных задач, стоящих перед Республикой Беларусь на национальном и международном уровне.

Классическое университетское издание, являя собой сплав научной и педагогической мысли, призвано формировать особую культуру знания – передового и доступного, теоретического и практического, общекультурного и специализированного. Словом, такого знания, которое будет работать.

Книги этой серии должны стать образцом научно-методического обеспечения современного образовательного процесса в высшей школе, утвердить ведущую роль нашего университета в качестве национального научно-методического центра Республики Беларусь.

Надеемся, что серия «Классическое университетское издание» состоится и как одно из слагаемых особой культурно-образовательной среды БГУ, которая будет способствовать интеллектуальному росту и творческой созидательной деятельности наших студентов.

*Ректор Белорусского  
государственного  
университета  
академик НАН Беларуси,  
профессор*



*C. V. Абламейко*

---

## ПРЕДИСЛОВИЕ

Слово «интеллект» (от лат. *intellectus* – ум, рассудок, разум) означает способность к мышлению [1]. Еще на заре развития вычислительной техники стояла проблема создания систем для различного рода интеллектуальной деятельности – так называемого искусственного интеллекта. Это связано с тем, что многие задачи не могут быть решены точными алгоритмическими методами. Другим аспектом данной проблемы является то, что искусственная система должна не просто функционировать по детерминированным алгоритмам, а должна быть обучаемой, генерировать знания и алгоритмы решения задач. Развитие таких систем происходит за счет самоорганизации, в результате которой осуществляется адаптация к решаемой задаче.

Существует два основных направления в теории искусственного интеллекта. В первом традиционном направлении используются методы логических рассуждений и символьной обработки информации, второе, связанное с построением сетей, состоящих из нейронных элементов, опирается на биологические основы естественного интеллекта, что позволяет проектировать системы, способные к обучению и самоорганизации. Искусственный интеллект, созданный на основе нейросетевых методов, называется *нейроинтеллектом*.

В данном пособии основное внимание уделено рассмотрению теории нейронных сетей (НС) и их использованию для решения различного рода задач.

В первой главе, посвященной различным аспектам нейронной организации головного мозга и обучения биологических систем, описываются процессы морфогенеза и самоорганизации биологических систем, а также основные этапы развития искусственных нейронных сетей и их классификация, показано, что самоорганизация происходит в результате динамической перестройки системы в целях адаптации к внешней среде. Приведены модели обучения биологических систем.

В второй главе рассматриваются однослойные персептроны, сыгравшие большую роль в создании нейроинтеллекта. Приведены основные определения, алгоритмы обучения и применения однослойных нейронных сетей. Показано, что однослойный персепtron с сигнальной функцией активации нейронных элементов способен решить задачу «**ИСКЛЮЧАЮЩЕЕ ИЛИ**».

В третьей главе излагаются различные аспекты построения и обучения многослойных персептронов, которые позволяют осуществить любое отображение входных сигналов в выходные. Отмечена несостоятельность некоторых мифов

о возможностях многослойных персепtronов, а также алгоритм обратного распространения ошибки, его модификации, ориентированные на улучшение процесса обучения. Анализируется применение многослойных персепtronов для решения задач распознавания, прогнозирования и управления.

В четвертой главе описываются функционирование и алгоритмы обучения рекуррентных сетей. Такие сети применяются в основном для прогнозирования различных временных процессов.

В пятой главе анализируются сверточные нейронные сети, которые широко используются для обработки изображений. Приводятся основные принципы построения и обучения сверточных нейронных сетей. Рассматривается упрощенная архитектура сверточной нейронной сети, которая позволяет классифицировать рукописные цифры с большей точностью, чем обычные сверточные сети архитектуры LeNet-5.

В шестой главе представлены автоэнкодерные нейронные сети, которые применяются для выделения наиболее важных информативных признаков (feature extraction) во входном пространстве образов, сжатия информации, очистки данных от шумов, визуализации и классификации данных, метод главных компонент, архитектура, обучение и применение автоэнкодерных нейронных сетей. Показано, что автоэнкодерные нейронные сети, в отличие от метода главных компонент, позволяют осуществить нелинейное преобразование информации.

Седьмая глава посвящена теоретическому анализу и различным аспектам применения релаксационных нейронных сетей с обратными связями и циркуляцией информации до тех пор, пока не установится стабильное состояние, а также архитектура, функционирование и обучение таких нейронных сетей. Для сети Хопфилда описываются механизм получения функции Ляпунова и анализ устойчивости.

В восьмой главе рассмотрены самоорганизующиеся сети Кохонена, которые характеризуются обучением без учителя. Они осуществляют топологическое упорядочивание входного пространства образов и основываются на конкурентном обучении. Приводятся архитектура и различные алгоритмы обучения таких нейронных сетей, а также алгоритм решения задачи коммивояжера с использованием сети Кохонена.

Девятая глава посвящена глубоким нейронным сетям (по этому вопросу полностью отсутствует информация в отечественной литературе), которые осуществляют глубокое иерархическое преобразование входного пространства образов и являются в настоящее время приоритетным направлением в области искусственного интеллекта. Приводятся основные принципы построения, обучения и функционирования таких нейронных сетей.

Авторы стремились изложить материал в доступной форме и восполнить пробелы в отечественной литературе по данной тематике, поэтому в учебном пособии подробно рассмотрены различные нейронные сети и приводится много примеров.

## **Глава 1**

### **БИОЛОГИЧЕСКИЕ ОСНОВЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА**

Интеллект характеризует способность в процессе мышления к генерации и выбору способа действий, адекватно отражающих решаемую проблему. Естественный интеллект возник и развивается в ходе биологической эволюции, адаптируясь к внешней среде. В той или иной мере он присущ биологическим формам жизни. Искусственный интеллект характеризует способность к мышлению искусственных систем. Важным его направлением являются биологически инспирированные (biologically inspired) интеллектуальные системы, которые опираются на биологические основы естественного интеллекта и пытаются в той или иной мере моделировать мыслительные процессы живых существ. В данной главе рассматриваются нейронная организация и механизмы обучения естественных систем. Обсуждаются процессы самоорганизации и морфогенеза. Приводятся основные этапы развития искусственных нейронных сетей и их классификация.

#### **1.1. НА СТЫКЕ НАУК**

Человеческий мозг — самая сложная в обозримом мире биологическая структура, представляющая собой результат миллионов лет эволюции. В отличие от компьютера он создавался без жестко определенной схемы, путем естественного отбора — главной движущей силы эволюции. Способность человеческого мозга к обучению, запоминанию сложной информации и логическому мышлению с давних пор привлекает философов, нейробиологов, психологов и кибернетиков. Важную роль в развитии головного мозга играет адаптация. *Адаптация* — это процесс приспособления организма или популяции организмов к внеш-

ней среде. Она определяется как внутренней структурой организма, которая характеризует его индивидуальность, так и воздействием внешней среды. В зависимости от этого возможны различные варианты адаптационного взаимодействия (модели поведения), например конформистская, ритуальная, агрессивная и т. д. В дальнейшем популяцию организмов будем называть биологической системой.

*Самоорганизация* — процесс динамической перестройки организма или биологической системы, происходящий с помощью обучения в целях адаптации к внешней среде. В прошлом многие ученые, например Р. Декарт, представляли психику как нечто нематериальное. Сейчас большинство исследователей сходятся к мысли, что психику можно объяснить материалистически, как следствие взаимодействия клеток мозга. До недавнего времени представители различных наук работали изолированно. Так, *нейробиологи* изучали функционирование мозга на молекулярном уровне, используя принцип «от молекулы к разуму». *Психологи* долгое время находились под влиянием бихевиоризма и исследовали мозг как «черный ящик», основываясь лишь на наблюдении поведения без учета морфологии мозга. В противовес этому представители *когнитивной нейробиологии* изучали нейронные основы мыслительных процессов, пытаясь выяснить, как психические события скоррелированы с электрическими сигналами в мозге, и описать высшие психические функции в терминах точных наук.

Биологически инспирированный *искусственный интеллект*, опираясь на достижения нейробиологии, создает и исследует искусственные системы, которые дают возможность воспроизводить поведение и мыслительные процессы биологических существ.

Теория искусственных нейронных сетей, моделируя в той или иной степени мыслительные процессы, позволяет глубже понять функционирование мозга.

## 1.2. БИОЛОГИЧЕСКИЙ НЕЙРОН

Головной мозг человека весит от 1,3 до 1,8 кг и содержит триллион клеток, из которых 100 млрд представлены соединенными в сети нейронами [2]. Это соизмеримо с числом звезд в Млечном Пути. Нейроны, соединенные разнообразными связями в сеть, определяют интеллект, творческие способности и память человека. Количество нейронов в головном мозге человека больше, чем у всех остальных известных форм жизни. *Нейрон* представляет собой особый вид клеток, которые обладают

электрической активностью [3]. Он получает информацию (рис. 1.1) при помощи сильно разветвленных отростков, называемых *дendritami*, и передает информацию вдоль тонкого волокна — *аксона*. Аксон имеет множество ответвлений, на конце каждого из которых находится область, называемая *синапсом*. Посредством синапсов осуществляется связь между различными нейронами. Каждый нейрон может иметь тысячи связей с соседними нейронами. Информация по аксонам передается в виде коротких электрических импульсов [2], амплитуда которых составляет около 100 мВ, а длительность — 1 мс. На участках контакта между нейронами (синапсы) электрические импульсы превращаются в химические сигналы, которые стимулируют проникновение в клетку нейрона положительных зарядов. Когда достигается критическое значение потенциала, называемое пороговым, в ядре нейрона возникает электрический импульс, распространяется, как волна, по аксону на следующий нейрон. Вклад одного синапса в установление соответствующего потенциала на выходе нейрона очень маленький. Для возникновения электрического импульса необходимо, чтобы нейрон непрерывно интегрировал множество синаптических входов.

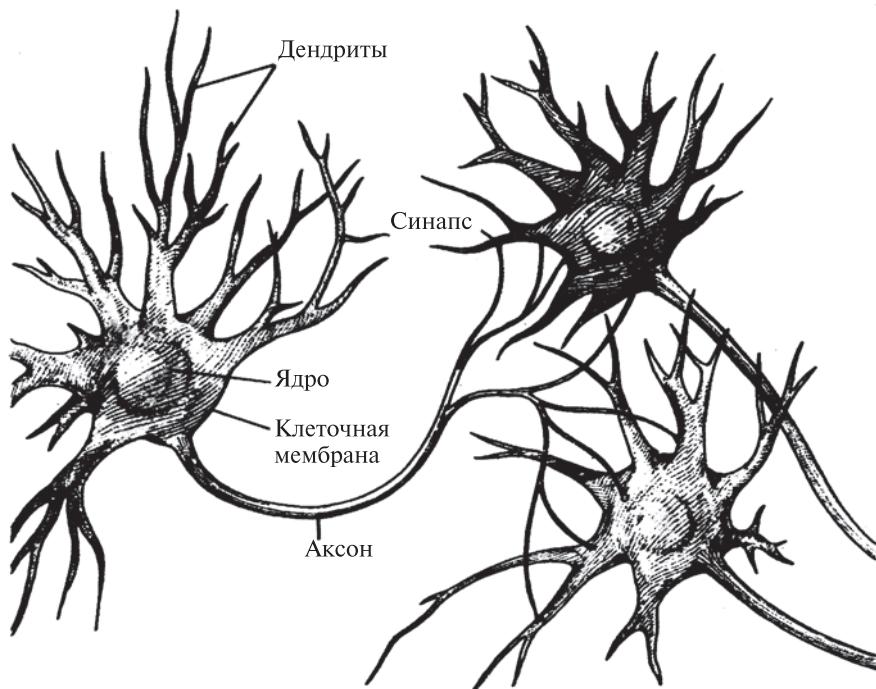


Рис. 1.1. Биологический нейрон

Такая интеграция является нелинейным преобразованием и не соответствует простой операции линейного суммирования. Использование технологии окраски нейронов солями серебра [2] позволяет выявить в коре головного мозга большое разнообразие типов нейронов. Существуют пирамидальные нейроны, нейроны таламуса, нейроны Пуркинье и т. д., всего около 50 типов. Из этого следует, что не все компоненты, из которых построен мозг, взаимозаменяемы.

Скорость распространения нервного импульса в аксоне составляет приблизительно 100 м/с, что в миллион раз меньше скорости распространения электрического сигнала по медной проволоке [2]. Однако параллельная обработка нейронами информации, одновременно распространяющаяся по множеству связей, компенсирует этот недостаток.

Таким образом, в процессе психической деятельности в коре головного мозга распространяются нервные импульсы, которые активизируют соответствующие области нейронов. Совокупность нейронов и связей между ними образуют *нейронную сеть*, от функционирования которой зависят эмоциональные реакции, сознательная деятельность и память человека.

### 1.3. НЕЙРОННАЯ ОРГАНИЗАЦИЯ МОЗГА

Структурно головной мозг состоит из двух симметричных полушарий, соединенных между собой пучком из примерно полумиллиарда аксонов [2], называемым *мозолистым телом* (рис. 1.2). Известно, что если мозолистое тело перерезать, то одна сторона мозга не будет осознавать, что видит другая. Мозг содержит белое и серое вещество. Серое вещество представляет собой совокупность дендритов, аксонов и нейронов. Белое вещество состоит главным образом из нервных волокон, которые соединяют различные области мозга друг с другом.

Анатомически мозг разделен на ряд зон, выполняющих разные функции. Это *префронтальная кора*, *гиппокамп*, *гипоталамус*, *зрительная кора*, *лимбическая система* и т. д. Каждая область представляет собой совокупность нейронов различных типов, соединенных между собой и другими частями мозга разнообразными связями. Как показывают исследования [2], молекулярный состав и функции отдельных нейронов приблизительно идентичны для биологических существ. Вероятно, именно организацией связей нейронов, которые формируются в результате взаимодействия с внешней средой, *Homo sapiens* отличается от других форм жизни.

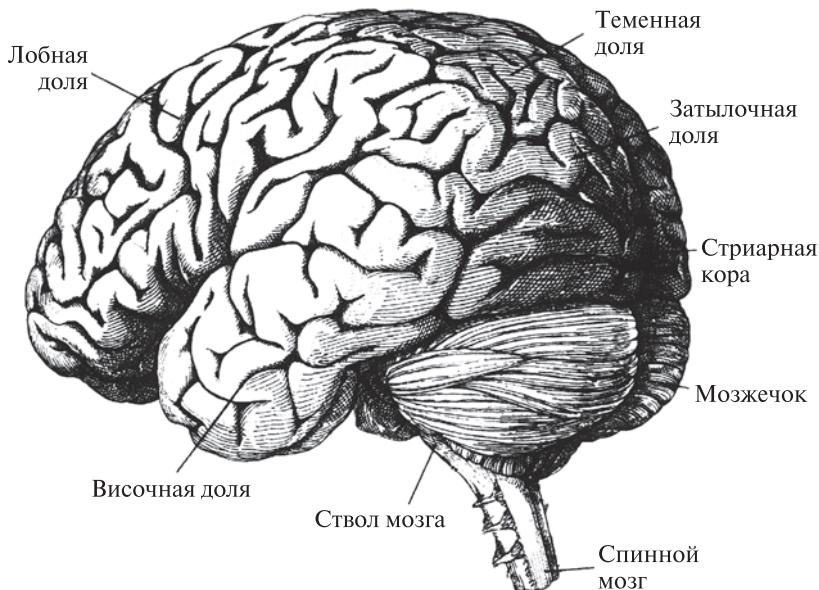


Рис. 1.2. Головной мозг

В настоящее время активно изучаются различные зоны мозга и их связь с психикой индивидуума. Так, *лимбическая система* участвует в эмоциональном поведении и долговременной памяти, которая хранит длительное время цифры, факты, правила и события [2].

Для того чтобы записи в этой памяти не забывались, необходимо через определенное время активизировать соответствующую нейронную структуру этой памяти. При повреждении нейронов долговременной памяти человек утрачивает связь со своим прошлым [4]. *Гиппокамп* выполняет функции кратковременной памяти, которая хранит информацию без реактивации соответствующей нейронной сети от нескольких минут до нескольких часов [5]. Время реактивации кратковременной памяти, чтобы информация не исчезла, значительно меньше, чем долговременной. Нейронная структура гиппокампа перерабатывает и хранит новую информацию, полученную в результате обучения и при соответствующей реактивации, в течение недель или месяцев, а затем передает ее в другую зону мозга для длительного хранения. *Предфронтальная кора* участвует в образовании оперативной памяти, которая необходима для извлечения фактов, событий и правил из долговременной памяти и манипулирования ими или промежуточными результатами в соответствии с обстоятельствами. В качестве примера оперативной

памяти можно привести планирование шахматного хода или операцию переноса при вычислении в уме, когда в процессе выполнения операции требуется запоминание промежуточных результатов. Соответствие перечисленных выше зон головного мозга различным типам памяти было подтверждено экспериментально [5]. Так, у больных с повреждением нейронов височной части – гиппокампа – нарушено запоминание новой информации. При этом они хорошо помнят прошлые события, но не воспринимают текущую информацию. Повреждение префронтальной коры является причиной многих психических расстройств, включая шизофрению [4].

Из разделения мозга на различные зоны вытекает концепция функциональной организации нейронных структур головного мозга [5]. Согласно ей различная информация обрабатывается и хранится в разных нейронных сетях мозга. Так, некоторые исследователи считают, что префронтальная кора представляет собой совокупность участков памяти, каждый из которых специализируется на информации определенного рода, например цвет, размеры объекта, семантические и математические знания. При этом разные характеристики одного объекта восприятия обрабатываются параллельно в различных нейронных схемах. Это подтвердилось при исследовании зрительной системы обезьяны [6], зрительная кора которой состоит из различных зон. Первая зона содержит с топографической точностью карту поверхности сетчатки, от которой информация параллельно поступает к остальным нейронным зонам. При этом информация, соответствующая форме объекта, поступает к одной нейронной сети, цвету объекта – к другой сети и т. д. Как показали эксперименты [6], повреждение первой зоны приводит к полному отказу зрительного тракта (слепоте), а поражение нейронов одной из специализированных зон – к недоступности и неосознаваемости соответствующего ей атрибута. Так, например, повреждение зоны цвета ведет к ахроматопсии, при которой все окружающее видится лишь в оттенках серого цвета.

Отсюда следует, что один и тот же объект представляется в мозге в виде совокупности различных атрибутов: зрительный образ, набор слов, звуков, запах, цвет и т. д. Эти представления об одном и том же объекте взаимодействуют между собой и охватывают множество нейронных структур, относящихся к разным функциональным областям мозга. Понимание и осознание увиденного происходит одновременно благодаря синхронизации активности соответствующих нейронных структур. Например, когда человек видит какой-то объект, его нейронная структура реагирует на цвет, форму, запах и функциональное назначение объекта. При этом вследствие нервной активности в мозге могут активизироваться также зоны нейронных элементов, которые

учитывают прошлый опыт человека о соответствующем объекте, т. е. происходит процесс воспоминания. Таким образом, активация одной из нейронных структур может генерировать работу другой и т. д. Результатом этой деятельности является совокупное представление об объекте. Аналогичные ассоциации возникают при воздействии слова, характеризующего тот или иной образ.

Каким образом взаимодействуют различные функциональные зоны мозга для получения единой интеграционной картины объекта? Простейшее решение – передача информации от различных зон некой высшей зоне, интегрирующей поступающую информацию. Однако анатомические данные не подтверждают этого предположения. Исследования показали [2], что между различными нейронными структурами мозга существуют обратные связи. Эти связи, как предполагается, синхронизируют активность различных функциональных зон, благодаря чему создается единая интеграционная картина об объекте.

Как уже отмечалось выше, скорость распространения сигналов в нервных волокнах намного меньше, чем скорость распространения сигналов в электрических схемах. Однако параллельная обработка нейронами образной информации и организация взаимодействия между нейронами, как было показано на примере зрительного тракта, позволяет нейтрализовать этот недостаток, поэтому на операциях распознавания образов человеческий мозг эффективнее конвенциональных компьютеров. Последовательные алгоритмы и обработка информации в компьютерах являются конструкцией нашего разума, а значит, в нем также должна быть заложена последовательная обработка информации. Так, при оперировании символьной информацией, например логический вывод или вычисление нескольких арифметических операций, конструкция нашего мозга использует последовательный алгоритм работы с привлечением оперативной памяти для запоминания промежуточной информации. Вследствие этого на операциях, требующих детерминированной последовательности действий, производительность вычислений человеческим мозгом значительно ниже, чем у компьютера. Другой аспект этой проблемы состоит в том, что мозг сам, в отличие от компьютера, может конструировать алгоритмы решения задачи. Он это производит путем поиска, методом проб и ошибок, используя также последовательный и параллельный режимы работы.

Исходя из приведенных выше рассуждений можно сделать следующие выводы [7]:

- мозг разделен на зоны, которые состоят из различных функциональных структур нейронных сетей. Разнообразная информация хранится и обрабатывается в разных нейронных структурах головного мозга;

- между нейронными структурами мозга существуют как прямые, так и обратные связи. Обратные связи в частности синхронизируют активность различных функциональных нейронных зон, благодаря чему создается единая интеграционная картина об объекте;
- осознание и понимание увиденного происходит одновременно, благодаря синхронизации активности соответствующих нейронных структур мозга, которые обрабатывают различные атрибуты информации (цвет, форма, запах и т. п.);
- для головного мозга характерна как последовательная, так и параллельная обработка информации;
- образная информация обрабатывается параллельно, а операции с символьной информацией производятся в общем случае последовательно;
- система памяти головного мозга состоит из кратковременной, оперативной и долговременной;
- существует последовательное и параллельное соединение нейронных структур мозга с точки зрения надежности. При нарушении одной из параллельно соединенных нейронных структур происходит частичная потеря функций соответствующей нейронной системы (неразличение цветов и т. д.). При дефекте одной из последовательно соединенных нейронных структур происходит отказ, ведущий к полной потере функций соответствующей нейронной системы.

#### **1.4. МОРФОГЕНЕЗ МОЗГА**

Развитие мозга – это непрерывный процесс, который происходит в результате обучения посредством взаимодействия с внешней средой с учетом внутренних факторов организма. Оно начинается в эмбриональный период и происходит в течение всей жизни. Как показывают исследования [7], именно в эмбриональный период закладываются основы разума путем образования нейронных структур и связей между ними. Мозг новорожденного составляет около четверти массы взрослого человека. В процессе развития индивидуума происходит увеличение размеров нейронов, числа их связей, изменяются структурная организация нейронных элементов и взаимодействие между ними. Некоторые ученые полагают [7], что формирование мозга в эмбриональный период происходит согласно заданной схеме аналогично сборке компьютера. Однако, как показывают проводимые исследования, это не совсем так. Установление и образование нейронных структур мозга зависят как от

генетической программы развития нервной системы, так и от внешних воздействий, стимулирующих нервную активность мозга [7]. Под управлением генетической программы происходит рост аксонов в определенных направлениях для установления синаптических связей между нейронами. Однако не существует точной спецификации каждого межнейронного соединения. Для этого потребовалось бы огромное количество генов. Как только растущие кончики аксонов достигают соответствующей им области, на выбор конкретной межнейронной связи начинают влиять события внешнего мира, стимулирующие определенную нервную активность. От воздействия этой активности и зависит установление соответствующих межнейронных связей.

Так, искусственное блокирование зрительной системы у новорожденных котят приводит к ненормальному развитию зрительной коры [7].

Таким образом, формирование нейронов и связей между ними начинается в эмбриональный период. Развитие индивида происходит посредством непрерывного процесса организации и модификации связей между нейронными элементами, а также роста размеров и количества нейронов. По мере взросления человека общее число нейронов в мозге снижается, уменьшаются размеры крупных нейронов и атрофируются связи между некоторыми из них [8]. Соответственно уменьшается вес мозга. Однако влияние этих изменений на интеллект у различных людей варьируется. Для некоторых индивидов структурные изменения становятся заметными только после 70 лет. Так, у людей от 40 до 70 лет исследователи [8] наблюдали рост дендритов в коре головного мозга с последующей регрессией их после 80 лет. Они предположили, что рост дендритов отражает попытку жизнеспособных нейронов нейтрализовать утрату с возрастом соседних нейронов. Это показывает, что мозг способен к динамической перестройке нейронных сетей даже в поздние годы жизни.

В процессе развития мозга непрерывно происходит динамическая перестройка нейронных сетей. При этом формируется сознание, а также то, что З. Фрейд называет бессознательным (инстинкты, влечения и т. д.) [9]. *Сознание* – это не состояние, а процесс, как отмечал американский психолог У. Джеймс [10]. К этому можно добавить, что сознание – это процесс непрерывного развития и структурной организации нейронов мозга, который происходит под воздействием внутренних и внешних факторов, во времени и в пространстве. Как уже отмечалось, формирование мозга происходит под воздействием генетической программы и факторов внешней среды. Поскольку каждый индивид имеет неповторимую генетическую информацию и развивается в разных условиях (испытывает воздействие разных сочетаний раздражителей), то

архитектура мозга формируется у каждого человека по-своему. Это составляет его **индивидуальность**. Человек живет, и его сознание развивается во времени и в пространстве. Поэтому, несмотря даже на одинаковую генетическую информацию, нельзя полностью воспроизвести архитектуру мозга и интеллект индивида.

Итак, в процессе морфогенеза непрерывно осуществляется динамическая перестройка нейронных сетей головного мозга. У различных индивидов это происходит по-разному в зависимости от генетической программы и внешних воздействий, стимулирующих нервную активность мозга. По мере взросления человека общее число нейронов в мозге снижается, однако путем перестройки соответствующих нейронных структур мозг стремится эту утрату компенсировать.

## 1.5. МЕХАНИЗМЫ ОБУЧЕНИЯ

*Обучение* – это процесс непрерывного развития и формирования сознания посредством взаимодействия с внешней средой с учетом индивидуальности организма. В результате обучения происходит динамическая перестройка нейронных сетей головного мозга. При этом, как уже отмечалось, увеличивается число связей между нейронами, совершенствуются сами нейроны и взаимодействие между ними. Способность синапсов и нейронных сетей динамически изменяться в результате воздействий называется *пластичностью*. Механизм пластичности лежит в основе обучения и создает в коре головного мозга соответствующие структуры нейронных сетей, которые определяют интеллект, память и эмоции индивида. *Синаптическая пластичность* возникает в результате изменения эффективности и количества связей между нейронами. В 1949 г. канадский психолог Д. Хебб сформировал принцип обучения [11], который был экспериментально подтвержден для некоторых типов синапсов. Согласно этому принципу для усиления связи между пресинаптическим и постсинаптическим нейронами необходимо совпадение их активности во времени (рис. 1.3). Нейроны, совпадение активности которых необходимо для усиления синаптической связи, изображены на рис. 1.3 черным цветом. Усиление связи приводит к повышению эффективности синаптической передачи между двумя нейронами. Данный механизм для некоторых синапсов гиппокампа, которые участвуют в образовании кратковременной памяти, подтвержден экспериментально [5]. Считается, что правило Хебба играет также большую роль в *эксплицитном обучении*, которое требует участия сознательных процессов.

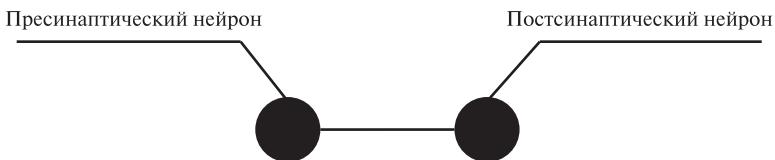


Рис. 1.3. Правило обучения Хебба

В 1963 г. французские исследователи Л. Тауц и Э. Кендел, изучая нервную систему морской улитки, сформулировали другой принцип ассоциативного обучения [5]. Они обнаружили, что если на пресинаптическую клетку действует некий третий нейрон, то для усиления синаптической связи между двумя нейронами не обязательна активность постсинаптического нейрона. Этот третий нейрон Тауц и Кендел назвали модулирующим. Если одновременно с пресинаптическим нейроном активен модулирующий нейрон, то происходит усиление связи между пресинаптическим и постсинаптическим нейроном (рис. 1.4.). Данный механизм обучения участвует в образовании условных рефлексов и *имплицитном обучении* [4], которое не требует участия сознания и сводится к механическому повторению определенных действий.

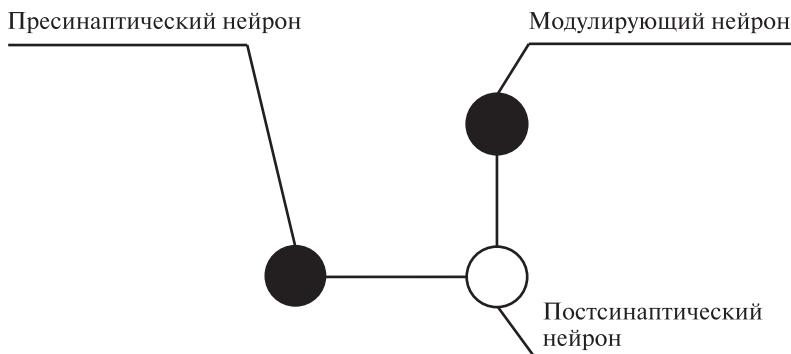


Рис. 1.4. Принцип ассоциативного обучения с модулирующим нейроном

Кроме механизма усиления связей существует также механизм их ослабления [5]. В этом случае сила связи между двумя нейронами уменьшается, если активность пресинаптического нейрона не сопровождается активностью постсинаптического нейрона. Исследования показали [4], что имеющиеся в префронтальной коре пирамидальные нейроны содержат симметричные и асимметричные синапсы. Симмет-

ричные синапсы соответствуют тормозным связям, а асимметричные – возбуждающим связям. Результатом действия тормозных связей может быть депрессия и стрессовые состояния [4].

Итак, в процессе обучения происходит усиление синаптических связей между соответствующими нейронами головного мозга, вследствие чего возникает *кратковременное* запоминание информации, которая хранится без реактивации соответствующей нейронной сети от нескольких минут до нескольких часов. При *долговременном* запоминании информации, длящемся месяцами, наблюдается активация и экспрессия генов, синтез соответствующих белков и рост новых связей [5]. При более сложных видах обучения участвуют оба механизма. Например, осмысливание нового материала требует определенного времени. В процессе этого усиливаются синаптические связи и в зависимости от сложности проблемы может осуществляться рост новых связей. Данный механизм будет действован до тех пор, пока не будет преодолено какое-то пороговое значение соответствующей нейронной сети головного мозга. В результате происходит понимание (инсайт) решаемой задачи. *Пороговое значение* нейронной сети характеризует степень незнания материала, индивидуальность организма и определяется пороговыми значениями нейронов, составляющих искомую сеть. При дальнейшем обучении в аналогичной области процесс понимания происходит быстрее за счет использования начальных знаний, заложенных в соответствующих синапсах.

В зависимости от вида взаимодействия обучающегося с внешней средой можно условно выделить обучение с учителем и без него. Обучение с учителем происходит при взаимодействии ученика с конкретным индивидом (учителем), с которым он находится в состоянии обратной связи.

В этом случае имеется конкретный желаемый выход и алгоритм его получения. В процессе взаимодействия реальная реакция ученика сравнивается с эталонной реакцией учителя. В зависимости от величины их несовпадения (целевая функция ошибки) происходит соответствующая перестройка синаптических связей в целях минимизации ошибки. При обучении без учителя нет конкретного учителя (учитель – внешняя среда) и ученик находится в состоянии обратной связи с внешней средой. Обучение здесь сводится к адаптации индивида к внешней среде. В обоих типах обучения используются как положительные, так и отрицательные обратные связи в соответствующих нейронных структурах головного мозга. Так, обучение с отрицательной обратной связью происходит для минимизации ошибки целевой функции. Положительная обратная связь может интенсифицировать процесс обучения при успешном взаимодействии индивида со средой.

Важной характеристикой процесса обучения является *обобщая способность*, характеризующая способность индивида интегрировать частные данные для определения закономерностей и пролонгации результатов. К этому относится способность после обучения на одних данных применять полученные знания для других данных или рассуждения от частного к общему. Обобщающая способность – важная черта нейронной организации мозга.

## 1.6. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СИСТЕМЫ

Интерес к нейроинтеллекту возник еще на ранних стадиях развития вычислительной техники. В его основе лежит нейронная организация искусственных систем, которая имеет биологические предпосылки. Способность биологических систем к обучению, самоорганизации и адаптации имеет большое преимущество по сравнению с современными вычислительными системами. Достоинство компьютерных систем – большая скорость распространения информации и возможность учета большого объема знаний, накопленных человечеством в этой области. Разработка искусственных разумных систем, соединяющих преимущества биологических существ и современной вычислительной техники, создает потенциальные предпосылки для перехода к качественно новому этапу эволюции в вычислительной технике.

Первыми в области искусственных нейронных сетей (1943 г.) были В. Мак-Каллох (W. McCulloch) и В. Питтс (W. Pitts). Они показали, что при помощи пороговых нейронных элементов можно реализовать исчисление любых логических функций [12]. В 1949 г. Д. Хебб (D. Hebb) [11] предложил правило, ставшее математической основой для обучения ряда нейронных сетей. В 1959 г. Ф. Розенблatt (F. Rosenblatt) создал модель нейронной сети, которую он назвал персепtronом. Результаты исследований он обобщил в книге «Принципы нейродинамики» [13]. В 1959 г. В. Видроу (W. Widrow) и М. Хофф (M. Hoff) предложили процедуру обучения для линейного адаптивного элемента ADALINE [14], получившую название «дельта-правило». В 1969 г. М. Минский (M. Minsky) и С. Пайперт (S. Papert) опубликовали монографию «Персептроны» [15], в которой осуществили математический анализ персептрана и отметили присущие ему ограничения. Выводы их были довольно пессимистичными, и это сыграло негативную роль в дальнейшем развитии нейронных сетей. Приостановленные в этой области работы

были возобновлены во второй половине 1970-х гг. В 1976 г. С. Гроссберг (S. Grossberg) разработал теорию адаптивного резонанса [18], которая может быть использована для построения ассоциативной памяти. Дж. Андерсон (J. Anderson) предложил в 1977 г. модель линейной ассоциативной памяти [16]. Исследования в этом направлении продолжил Т. Кохонен (T. Kohonen), создавший модель оптимальной линейной ассоциативной памяти [17].

Интенсивность исследований в области нейронных сетей значительно возросла в 1980-е гг. Дж. Хопфилд (J. Hopfield) в 1982 г. проанализировал устойчивость нейронных сетей с обратными связями и предложил применять их в качестве ассоциативной памяти [19]. Т. Кохонен разработал самоорганизующиеся нейронные сети [20]. В 1986 г. Д. Румельхат, Дж. Хинтон и Р. Вильямс создали алгоритм обратного распространения ошибки, который стал эффективным средством для обучения многослойных нейронных сетей [21]. При этом в научной среде вплоть до 2006 г. господствовала концепция, согласно которой не имеет смысла использовать более двух скрытых слоев в персептроне. Это было связано с тем, что алгоритм обратного распространения ошибки не давал никакого выигрыша в решении задач при обучении персептрона с более чем двумя скрытыми слоями. Также данная концепция базировалась на теореме об универсальной аппроксимации персептроном с одним или двумя скрытыми слоями любой функции со сколь угодно заданной точностью. К тому же появились машины опорных векторов (support vector machine, SVM), которые часто показывали большую эффективность на операциях распознавания образов по сравнению с персептроном. Поэтому интерес к многослойным персепtronам стал постепенно падать. И только после публикации работ [22–24] Дж. Хинтона (G. Hinton) с 2006 г. начался новый этап: появились глубокие нейронные сети (deep neural networks), представляющие собой результат развития многослойных персептронов и интегрирующие различные парадигмы обучения нейронных сетей. Благодаря многослойной архитектуре они позволяют обрабатывать и анализировать большой объем данных, а также моделировать когнитивные процессы в различных областях. В настоящее время большинство высокотехнологичных компаний используют глубокие нейронные сети для проектирования различных интеллектуальных систем. По версии ученых Массачусетского технологического института (США), глубокие нейронные сети входят в список 10 наиболее прорывных высоких технологий, способных в недалеком будущем

в значительной степени преобразить повседневную жизнь большинства людей на нашей планете. В настоящее время глубокие нейронные сети считаются революционным прорывом в области искусственного интеллекта.

## 1.7. КЛАССИФИКАЦИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Нейронные сети можно классифицировать в зависимости от различных качеств.

### *По характеру обучения*

1. С учителем, когда известно выходное пространство решений нейронной сети, а также предполагается, что имеются входные сигналы и эталонные реакции на них. В процессе обучения происходит целенаправленная модификация синаптических связей нейронной сети ( $NN$ ) для достижения наилучшего соответствия между реальными выходными значениями сети  $Y$  и их эталонными значениями  $e$  (рис. 1.5).

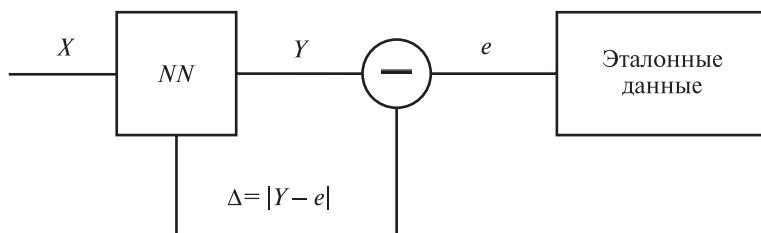


Рис. 1.5. Обучение с учителем

2. Без учителя. В этом случае нейронная сеть формирует выходное пространство решений только на основе входных воздействий. Такие сети называются самоорганизующимися (рис. 1.6).

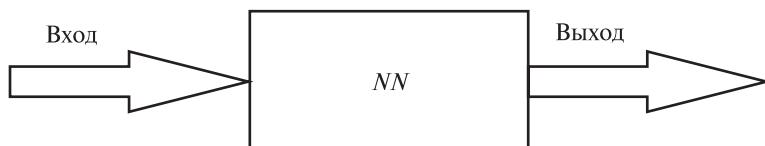


Рис. 1.6. Обучение без учителя

3. Подкрепляющее обучение (reinforcement learning). Происходит на основе сигнала подкрепления  $r$  от внешней среды (рис. 1.7).

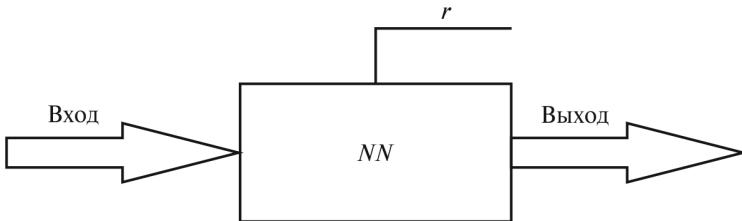


Рис. 1.7. Подкрепляющее обучение

#### *По характеру настройки синапсов*

1. Сети с фиксированными связями. В этом случае весовые коэффициенты нейронной сети выбираются сразу, исходя из условия задачи. При этом

$$\frac{dW}{dt} = 0,$$

где  $W$  – весовые коэффициенты сети.

2. Сети с динамическими связями. Для них в процессе обучения происходит настройка синаптических связей, т. е.

$$\frac{dW}{dt} \neq 0.$$

#### *По архитектуре и обучению*

I. Персептронные нейронные сети. Они характеризуются, как правило, обучением с учителем. Архитектура их базируется на многослойном персептроне, а в основе обучения лежит метод градиентного спуска. К ним относят следующие:

1. Многослойные персептроны.
2. Рекуррентные НС, в которых присутствует обратная связь между входом и выходом. Выходное значение при этом определяется в зависимости как от входных, так и предшествующих выходных значений нейронной сети (рис. 1.8).

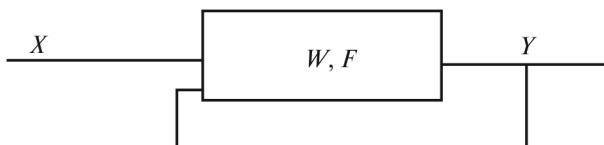


Рис. 1.8. Рекуррентная сеть

3. Рециркуляционные нейронные сети (PCA-сети, автоэнкодерные, автоассоциативные, репликаторные сети), которым присуще как прямое  $y = f(x)$ , так и обратное  $x = f^{-1}(y)$  преобразование информации (рис. 1.9).

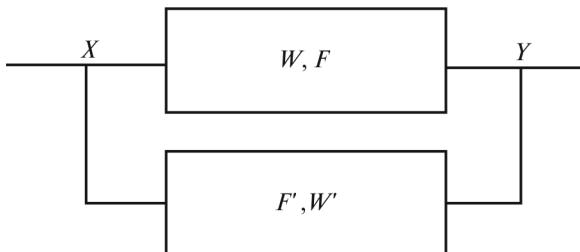


Рис. 1.9. Рециркуляционная нейронная сеть

4. Сверточные нейронные сети, представляющие дальнейшее развитие персептрана и неокогнитрона для обработки изображений.

5. Глубокие нейронные сети, осуществляющие глубокое нелинейное иерархическое преобразование информации.

II. Самоорганизующиеся нейронные сети, обучение без учителя. Такое обучение основывается только на сигналах от внешней среды. К ним относятся следующие:

1. Нейронные сети Кохонена.
2. Нейронные сети адаптивного резонанса.

III. Релаксационные нейронные сети, в которых циркуляция информации происходит до тех пор, пока не перестанут изменяться выходные значения нейронной сети (состояние равновесия). К ним относятся:

1. Нейронные сети Хопфилда.
2. Нейронные сети Хэмминга.
3. Двунаправленная ассоциативная память.

IV. Гибридные нейронные сети. Отличаются применением двух подходов к обучению – с учителем и без учителя. К ним относятся:

1. Нейронные сети встречного распространения (counter propagation networks).
2. Нейронные сети с радиально-базисной функцией активации (RBF networks).
3. Нечеткие нейронные сети, характеризуемые применением нечеткой логики и нейронных сетей.
- V. Нейронные иммунные сети, в которых применяются искусственные иммунные системы и нейронные сети.

## Г л а в а 2

# ОДНОСЛОЙНЫЕ ПЕРСЕПТРОНЫ

В главе рассматриваются нейронные сети с одним обрабатывающим слоем и прямыми связями (feed forward neural networks). В литературе они называются однослойными персепtronами (single layer perceptron). Большой вклад в их разработку внесли Ф. Розенблatt (1959), В. Видроу (B. Widrow) и М. Хофф (M. Hoff) (1959). *Однослойные* нейронные сети формируют линейную разделяющую поверхность, что ограничивает круг решаемых ими задач. Это установили в 1960-х гг. М. Минский и С. Пайперт на примере решения задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ», сделав пессимистические выводы по поводу дальнейших перспектив развития нейронных сетей. Хотя, как будет показано в данной главе, однослойный персептрон может решить задачу «ИСКЛЮЧАЮЩЕЕ ИЛИ» при использовании сигнальной функции активации. Также здесь приводятся основные определения, описываются алгоритмы обучения однослойных персептронов и применение однослойных нейронных сетей [13–15, 25–29].

### 2.1. ИСКУССТВЕННЫЙ НЕЙРОН

Основным элементом нейронной сети является *искусственный нейрон*. Он выполняет операцию нелинейного преобразования суммы произведений входных сигналов на весовые коэффициенты:

$$y = F \left( \sum_{i=1}^n \omega_i x_i \right) = F(WX),$$

где  $X = (x_1, x_2, \dots, x_n)^T$  – вектор входного сигнала;  $W = (\omega_1, \omega_2, \dots, \omega_n)$  – весовой вектор;  $F$  – оператор нелинейного преобразования.

Оператор нелинейного преобразования называется функцией активации нейронного элемента. Схема нейронного элемента изображена на рис. 2.1 и состоит из сумматора и блока нелинейного преобразования  $F$ . Каждому  $i$ -му входу нейрона соответствует весовой коэффици-

ент  $\omega_i$  (синапс), который характеризует силу синаптической связи по аналогии с биологическим нейроном.

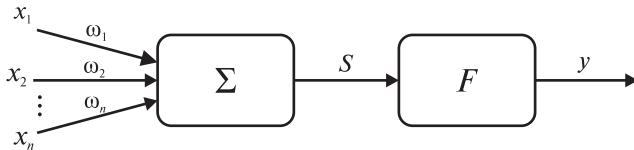


Рис. 2.1. Нейронный элемент

Сумма произведений входных сигналов на весовые коэффициенты называется *взвешенной суммой*. Она представляет собой скалярное произведение вектора весов на входной вектор:

$$S' = \sum_{i=1}^n \omega_i x_i = (W, X) = |W| |X| \cdot \cos \alpha,$$

где  $|W|$ ,  $|X|$  – длины векторов  $W$  и  $X$  соответственно;  $\alpha = \hat{W}, X$  – угол между векторами  $W$  и  $X$ .

Длины весового и входного векторов вычисляются через их координаты:

$$|W| = \sqrt{\omega_1^2 + \omega_2^2 + \dots + \omega_n^2},$$

$$|X| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}.$$

Поскольку для нейронного элемента длина весового вектора после обучения  $|W| = \text{const}$ , то величина взвешенной суммы определяется проекцией входного на весовой вектор:

$$S' = |W| |X| \cdot \cos \alpha = |W| \cdot X_W,$$

где  $X_W$  – проекция вектора  $X$  на вектор  $W$ .

Если входные векторы нормированные, т. е.  $|X| = \text{const}$ , то величина взвешенной суммы зависит только от угла между векторами  $X$  и  $W$ . Тогда при различных входных сигналах взвешенная сумма будет изменяться по косинусоидальному закону. Она достигает своего максимального значения при коллинеарности входного и весового векторов.

Если сила связи  $\omega_i$  отрицательная, то она называется *тормозящей*. В противном случае синаптическая связь является *усиливающей*. Вектор входного сигнала называется *паттерном входной активности* нейронной сети, а вектор выходного сигнала – *паттерном выходной активности*.

Рассмотренная модель искусственного нейрона – биологически инспирированная и используется для построения искусственных нейронных сетей.

## 2.2. ФУНКЦИИ АКТИВАЦИИ НЕЙРОННЫХ ЭЛЕМЕНТОВ

В качестве оператора нелинейного преобразования могут использоваться различные функции, которые определяются в соответствии с решаемой задачей и типом нейронной сети.

Пусть  $T$  – порог нейронного элемента, характеризующий расположение функции активации по оси абсцисс. Представим взвешенную сумму как

$$S = \sum_{i=1}^n \omega_i x_i - T = S' - T.$$

Рассмотрим наиболее распространенные функции активации нейронных элементов.

### 2.2.1. Линейная функция

В этом случае выходное значение нейронного элемента равняется взвешенной сумме  $y = kS$ , где  $k$  – коэффициент наклона прямой.

Изменение порога линейного элемента эквивалентно сдвигу функции активации по оси абсцисс  $S'$  (рис. 2.2).

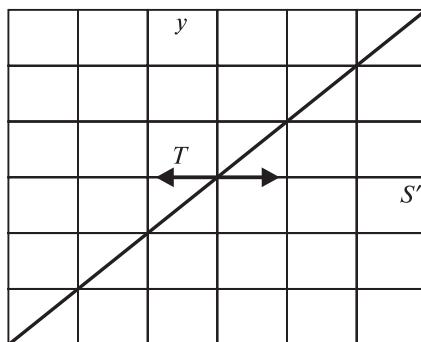


Рис. 2.2. Линейная функция активации

### 2.2.2. Пороговая функция

В качестве пороговой функции активации может использоваться или биполярная, или бинарная пороговая функция. В случае использования пороговой бинарной функции активации (рис. 2.3)

$$y = \text{sign}(S) = \begin{cases} 1, & S > 0, \\ 0, & S \leq 0, \end{cases} = \begin{cases} 1, & S' > T, \\ 0, & S' \leq T. \end{cases}$$

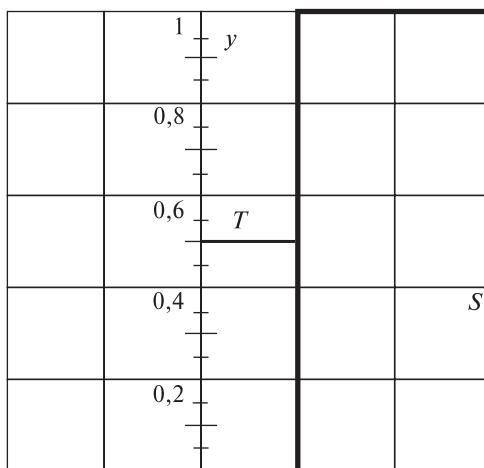


Рис. 2.3. Пороговая функция активации

Если применяется пороговая биполярная функция активации, то выходное значение нейронного элемента вычисляется как

$$y = F(S) = \begin{cases} 1, S > 0, \\ -1, S \leq 0. \end{cases}$$

Основным недостатком пороговых функций активации является их недифференцируемость.

### 2.2.3. Линейная ограниченная функция

В этом случае выходное значение нейрона определяется следующим образом (рис. 2.4):

$$y = \begin{cases} p, S > \alpha, \\ -p, S < -\alpha, \\ S, -\alpha \leq S \leq \alpha. \end{cases}$$

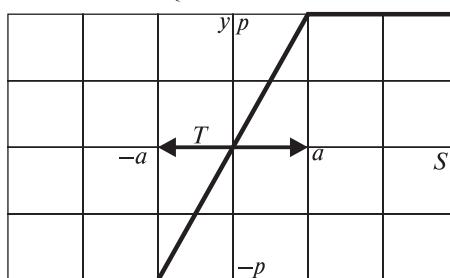


Рис. 2.4. Линейная ограниченная функция

### 2.2.4. Модифицированная пороговая функция

Данная пороговая функция используется в двунаправленной ассоциативной памяти. Выходное значение нейронного элемента характеризуется следующей функцией:

$$y(t+1) = \begin{cases} 1, & S > 0, \\ y(t), & S = 0, \\ -1, & S < 0. \end{cases}$$

Здесь  $t$  – параметр времени.

### 2.2.5. Сигмоидная функция

Эта функция является непрерывным аналогом бинарной пороговой функции активации и представляет собой непрерывную, возрастающую, ограниченную функцию в диапазоне значений  $[0, 1]$ :

$$y = \frac{1}{1 + e^{-cS}},$$

где  $c > 0$  – коэффициент, характеризующий ширину сигмоидной функции по оси абсцисс (рис. 2.5).

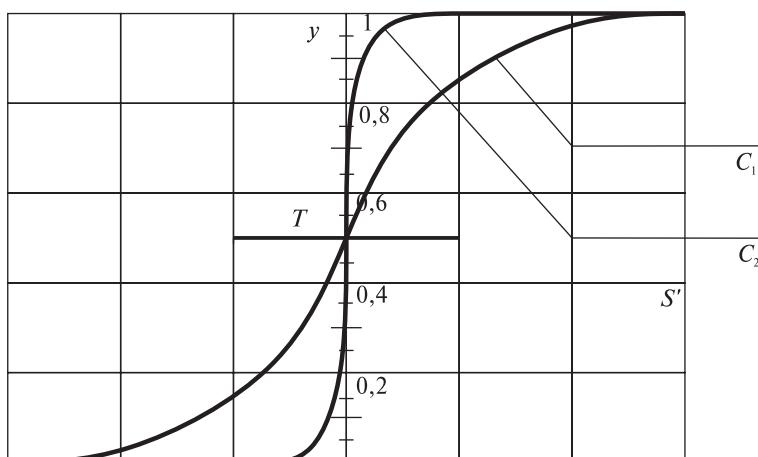


Рис. 2.5. Сигмоидная функция активации ( $C_1 > C_2$ )

На рис. 2.5 показано, что изменение порога линейного элемента эквивалентно сдвигу функции активации по оси абсцисс  $S'$ . Сигмоидная функция монотонна и всюду дифференцируема, поэтому она получила широкое распространение в искусственных нейронных сетях.

### 2.2.6. Биполярная сигмоидная функция

Данная функция представляет собой непрерывный аналог биполярной пороговой функции активации с диапазоном значений  $[-1, 1]$  (рис. 2.6).

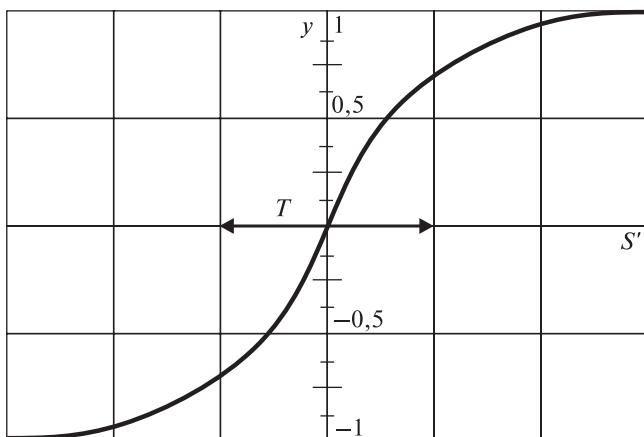


Рис. 2.6. Биполярная сигмоидная функция активации

Выходное значение нейронного элемента определяется как

$$y = \frac{2}{1+e^{-cS}} - 1.$$

### 2.2.7. Гиперболический тангенс

Функция гиперболического тангенса аналогична биполярной сигмоидной функции (рис. 2.7). Она определяется следующим образом:

$$y = \text{th}(cS) = \frac{e^{cS} - e^{-cS}}{e^{cS} + e^{-cS}} = \frac{2}{1+e^{-2cS}} - 1,$$

где  $c$ , как и в случае с сигмоидной функцией, характеризует ширину функции активации гиперболического тангенса по оси абсцисс.

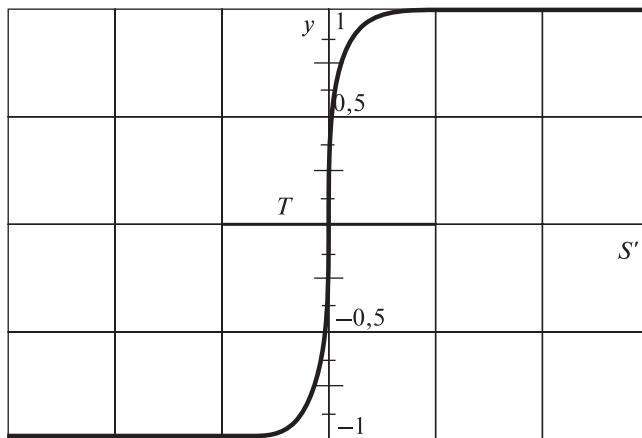


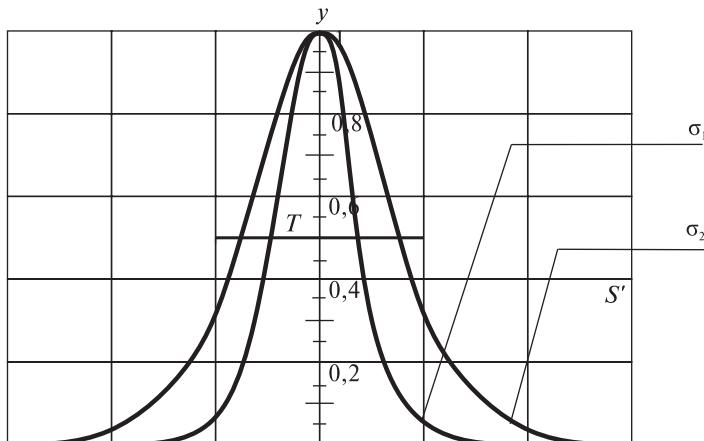
Рис. 2.7. Гиперболический тангенс

### 2.2.8. Радиально-базисная функция

Эта функция определяется функцией Гаусса для нормального закона распределения (рис. 2.8). В соответствии с ней

$$y = \exp\left(-\frac{S^2}{2\sigma^2}\right),$$

где  $\sigma$  – среднеквадратичное отклонение ширины радиально-базисной функции.

Рис. 2.8. Функция Гаусса ( $\sigma_2 > \sigma_1$ )

Величина  $S$  определяется, как правило, в соответствии с евклидовым расстоянием между входным и весовым вектором:

$$S^2 = |X - W|^2 = \sum_i (x_i - \omega_i)^2.$$

### 2.2.9. Ректификационная функция активации

Эта функция используется в глубоких нейронных сетях (рис. 2.9).

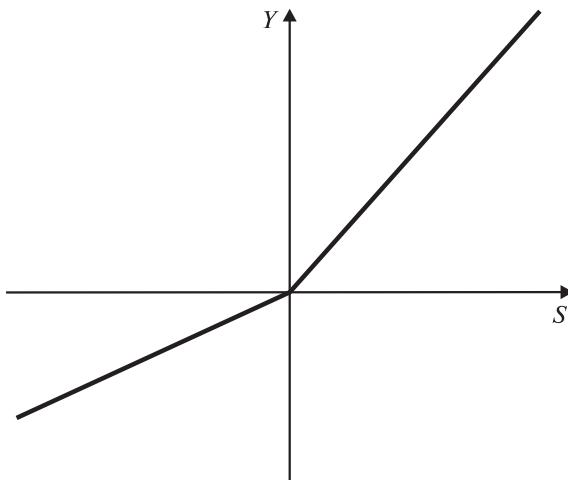


Рис. 2.9. Ректификационная функция

В соответствии с этой функцией активации выходное значение нейронного элемента вычисляется как

$$y = F(S) = \begin{cases} S, & S > 0, \\ kS, & S \leq 0. \end{cases}$$

Здесь  $k = 0$  или принимает небольшое значение, например  $k = 0,01$  или  $k = 0,001$ .

### 2.2.10. Функция активации softmax

Данная функция активации применяется обычно в последнем слое нейронной сети и используется в системах распознавания образов. Так, выходное значение  $j$ -го нейронного элемента в соответствии с данной функцией активации определяется следующим образом:

$$y_j = \text{softmax}(S_j) = \frac{e^{S_j}}{\sum_j e^{S_j}}.$$

В заключение отметим, что применение различных функций активации обусловливается классом решаемых нейронной сетью задач. Помимо перечисленных, могут применяться и другие функции активации нейронных элементов, которые адекватно отражают решаемую задачу.

### 2.3. НЕЙРОННЫЕ СЕТИ С ОДНИМ ОБРАБАТЫВАЮЩИМ СЛОЕМ

Совокупность нейронных элементов и связей между ними называется *нейронной сетью*. *Слой нейронной сети* – это множество нейронных элементов, на которые в каждый тик времени параллельно поступает информация от других нейронных элементов сети. В данном разделе будут рассматриваться нейронные сети, состоящие из одного слоя нейронных элементов, осуществляющего обработку входной информации. Такие сети принято изображать в виде двуслойной нейронной сети, где первый слой нейронных элементов является распределительным, а второй – обрабатывающим. Распределительный слой передает входные сигналы на обрабатывающий слой нейронных элементов, который преобразует входную информацию в соответствии с синаптическими связями и функцией активации (рис. 2.10). При этом каждый нейрон распределительного слоя имеет синаптические связи со всеми нейронами обрабатывающего слоя.

Тогда выходное значение  $j$ -го нейронного элемента второго слоя можно представить как

$$y_j = F(S_j) = F\left(\sum_{i=1}^n \omega_{ij} x_i - T_j\right), \quad (2.1)$$

где  $T_j$  – порог  $j$ -го нейронного элемента выходного слоя;  $\omega_{ij}$  – сила синаптической связи между  $i$ -м нейроном распределительного слоя и  $j$ -м нейроном обрабатывающего слоя.

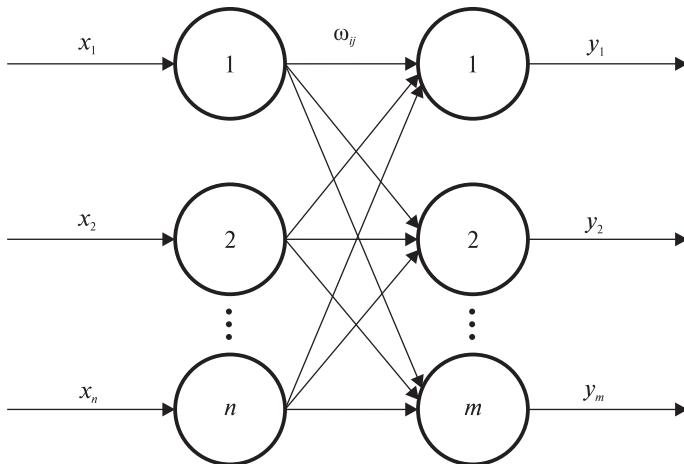


Рис. 2.10. Топология однослойной нейронной сети

Совокупность весовых коэффициентов сети можно представить матрицей размерностью  $n \times m$ :

$$W = \begin{bmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1m} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2m} \\ \dots & \dots & \dots & \dots \\ \omega_{n1} & \omega_{n2} & \dots & \omega_{nm} \end{bmatrix}.$$

Тогда вектор-столбец взвешенной суммы в матричном виде определяется следующим образом:

$$S = W^T X - T, \quad (2.2)$$

где  $T$  – вектор-столбец порогов нейронных элементов второго слоя.

## 2.4. ВОЗМОЖНОСТИ ОДНОСЛОЙНЫХ ПЕРСЕПТРОНОВ

Рассмотрим однослойный персептрон с двумя нейронами входного и одним нейроном выходного слоя (рис. 2.11).

Тогда взвешенная сумма вычисляется как

$$S = \omega_{11}x_1 + \omega_{21}x_2 - T_1, \quad (2.3)$$

а выходное значение нейронной сети  $y_1 = F(S)$  соответственно.

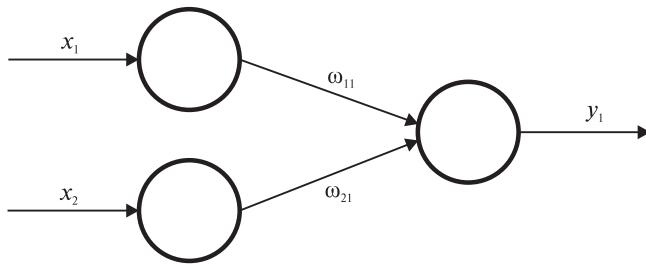


Рис. 2.11. Сеть с одним выходным нейроном

Следует отметить, что пороговое значение можно вынести за знак нейронного элемента и использовать в качестве весового коэффициента. В этом случае необходимо добавить в схему сети один входной нейрон и подать на его вход постоянное значение  $-1$  (рис. 2.12).

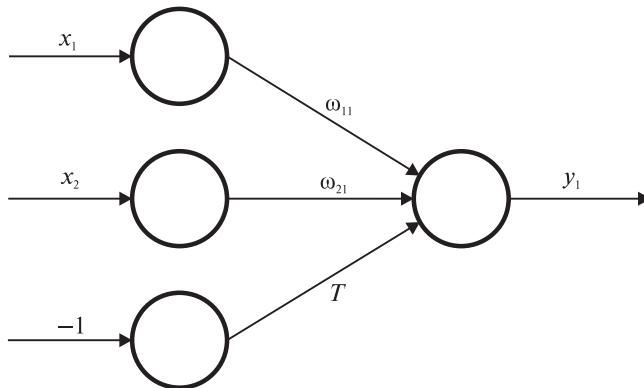


Рис. 2.12. Представление порога в качестве весового коэффициента

Пусть используется пороговая функция активации в выходном нейронном элементе. Тогда

$$y_1 = F(S) = \begin{cases} 1, & S > 0, \\ 0, & S \leq 0. \end{cases}$$

Такая сеть осуществляет *линейное разбиение* входного пространства образов на два класса (класс нулей и единиц) и может использоваться

для решения задач бинарной классификации образов. Уравнение разделяющей линии определяется при этом следующим образом:

$$\omega_{11}x_1 + \omega_{21}x_2 - T_1 = 0. \quad (2.4)$$

Она отделяет область решений, соответствующую одному классу, от другого класса и называется *дискриминантной линией*. Выражая из (2.4)  $x_2$ , получим

$$x_2 = \frac{T_1}{\omega_{21}} - \frac{\omega_{11}}{\omega_{21}}x_1. \quad (2.5)$$

В системе координат  $(x_2, x_1)$  последнее уравнение представляет собой прямую линию (рис. 2.13), которая отделяет один класс от другого.

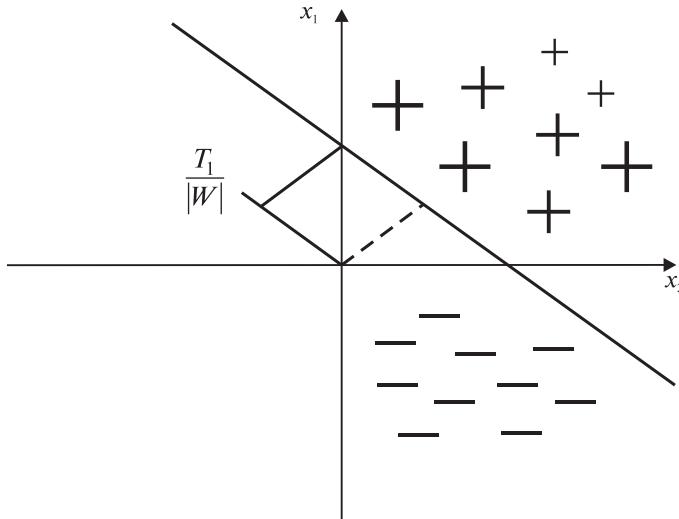


Рис. 2.13. Дискриминантная линия

При этом величина  $\frac{T_1}{|W|}$  – это расстояние между центром координат и прямой. Тогда если

$$X_W > \frac{T_1}{|W|},$$

то скалярное произведение  $S = (W, X) > T_1$  и  $y = 1$ . В противном случае  $y = 0$ . Здесь  $X_W$  – проекция входного вектора на весовой вектор.

Если размерность входного сигнала  $n = 3$ , то разделяющей поверхностью будет плоскость. При  $n > 3$  разделяющая поверхность – гиперплоскость.

Рассмотрим реализацию простейших логических операций. Если входное пространство образов с помощью линейной разделяющей поверхности можно разбить на два класса, то такие операции реализуются с использованием однослоистого персептрана. Схематично это представлено на рис. 2.14 для операций логического «И», «ИЛИ» и «ИСКЛЮЧАЮЩЕЕ ИЛИ» соответственно.

Из рис. 2.14 следует, что рассмотренный однослоистый персептран может решить задачу типа «И» и «ИЛИ», но не способен решить задачу «ИСКЛЮЧАЮЩЕЕ ИЛИ».

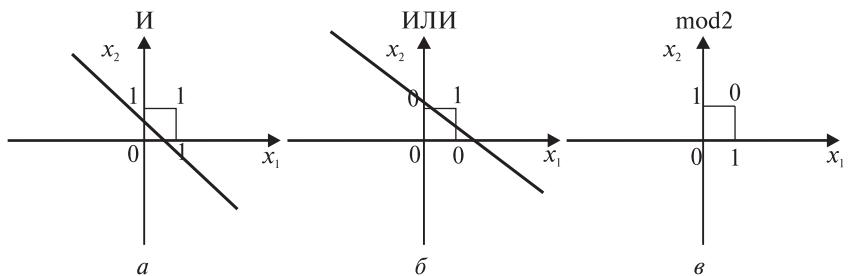


Рис. 2.14. Графическая интерпретация решения логических задач

Таким образом, линейная разделяющая поверхность, формируемая нейронной сетью с одним слоем обрабатывающих нейронных элементов, в общем случае ограничивает круг решаемых задач. Это в свое время показали американские ученые Минский и Пайперт [15]. Однако, как будет показано далее, однослоистый персептран с сигнальной функцией активации нейронных элементов может решить задачу «ИСКЛЮЧАЮЩЕЕ ИЛИ».

## 2.5. ПРАВИЛО ОБУЧЕНИЯ РОЗЕНБЛАТТА

*Самоадаптация и самоорганизация* нейронных сетей достигается в процессе их обучения, в ходе которого происходит определение синаптических связей между нейронными элементами. Цель задачи обучения — получение наилучшего *отображения* входных данных в выходные эталонные значения и *эффективной обобщающей способности сети* (корректная работа сети на данных, не входящих в обучающую выборку).

Обучающие правила определяют, как изменяются синаптические связи в ответ на входное воздействие. Рассмотрим правило обучения, которое предложил американский ученый Ф. Розенблатт в 1959 г. для нейронной сети, которую он назвал персептроном [13].

Персептрон — это сеть, состоящая из  $S$ ,  $A$  и  $R$  нейронных элементов (рис. 2.15). Нейроны слоя  $S$  называются *сенсорными* и предназначены для формирования входных сигналов, возникающих в результате внешних воздействий. Нейроны слоя  $A$  называются *ассоциативными* и предназначены для непосредственной обработки входной информации. Нейроны слоя  $R$  называются *эффекторными*. Они служат для передачи сигналов возбуждения к соответствующему объекту, например к мышцам.

Таким образом, персептрон Розенблатта содержит один слой обрабатывающих нейронных элементов с пороговой функцией активации, поэтому обучение персептрона происходит путем настройки весовых коэффициентов  $W$  и пороговых значений  $T$  между слоями  $S$  и  $A$ . В дальнейшем будем использовать стандартную интерпретацию нейронной сети Розенблатта, представляющую собой однослойный персептрон с пороговой функцией активации (см. рис. 2.10).

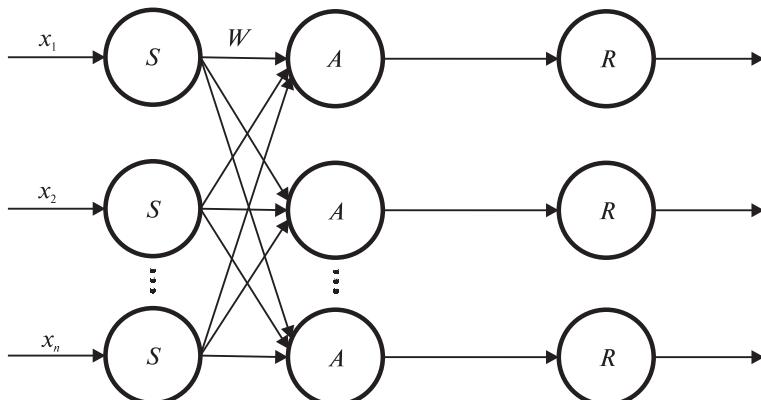


Рис. 2.15. Структура персептрона

В общем случае математическую формулировку правила обучения Розенблатта можно представить как

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha x_i(y_j - e_j), \quad (2.6)$$

$$T_j(t+1) = T_j(t) + \alpha(y_j - e_j), \quad (2.7)$$

где  $e_j$  – эталонное значение  $j$ -го выхода нейронной сети;  $\alpha$  – скорость или шаг обучения сети (learning rate).

Величина скорости обучения имеет следующее значение:

$$\alpha = \text{const}, \quad 0 < \alpha \leq 1.$$

Процедура обучения Розенблатта характеризуется тем, что весовые коэффициенты нейронной сети изменяются только в том случае, когда выходная реакция сети не совпадает с эталонной. Алгоритм обучения Розенблатта состоит из следующих основных шагов:

- весовые коэффициенты  $W$  и пороговые значения  $T$  нейронной сети инициализируются случайным образом в небольшом диапазоне значений, например  $[0, 1]$ ;
- на входы сети поочередно подаются входные образы  $X$  из обучающей выборки, которые трансформируются в выходные сигналы нейронных элементов  $Y$ ;
- после подачи каждого образа из обучающей выборки производится модификация синаптических связей сети в соответствии с выражениями (2.6) и (2.7);
- алгоритм продолжается до тех пор, пока все реальные выходные значения не станут равными эталонным, т. е.  $y_j = e_j$  для всех входных образов, или не перестанут изменяться весовые коэффициенты.

Согласно теореме о сходимости персептрана [13] если решение некоторой задачи существует, то рассмотренный выше алгоритм за конечное число шагов сходится. Входные образы из обучающей выборки в модели персептрана подаются до тех пор, пока не произойдет обучение сети.

*Линейная разделяющая поверхность*, формируемая персептраном с пороговой функцией активации, ограничивает круг решаемых им задач. В 1969 г. М. Минский и С. Пайперт [15] показали, что персептран не может решить задачу «ИСКЛЮЧАЮЩЕЕ ИЛИ». В связи с тем, что их выводы по поводу перспектив модели персептрана были весьма пессимистичными, их исследования в области нейронных сетей были почти полностью прекращены вплоть до конца 1970-х гг.

## 2.6. ГЕОМЕТРИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ ПРОЦЕДУРЫ ОБУЧЕНИЯ ПЕРСЕПТРОНА

Представим взвешенную сумму нейронной сети в виде скалярного произведения весового вектора на входной вектор:

$$S = (W, X) = |W| \|X| \cdot \cos \gamma, \quad (2.8)$$

где  $\gamma = \hat{W}, \hat{X}$ .

Тогда если  $0 < \gamma < 90^\circ$ , то  $S > 0$  и  $y = 1$ . Если  $90^\circ \leq \gamma < 270^\circ$ , то  $S \leq 0$  и  $y = -1$ . Рассмотрим векторы  $W$  и  $X$ . Пусть угол между ними  $\gamma \geq 90^\circ$ . В этом случае  $S \leq 0$  и  $y = 0$ . Предположим, что эталонное значение  $e = 1$ . В таком случае вектор  $W$  необходимо преобразовать так, чтобы угол  $\gamma$  стал меньше  $90^\circ$ . Это эквивалентно вращению вектора  $W$  по часовой стрелке. Отсюда правило преобразования вектора  $W$  будет следующим:

$$W' = W + \alpha X. \quad (2.9)$$

Схематично это представлено на рис. 2.16.

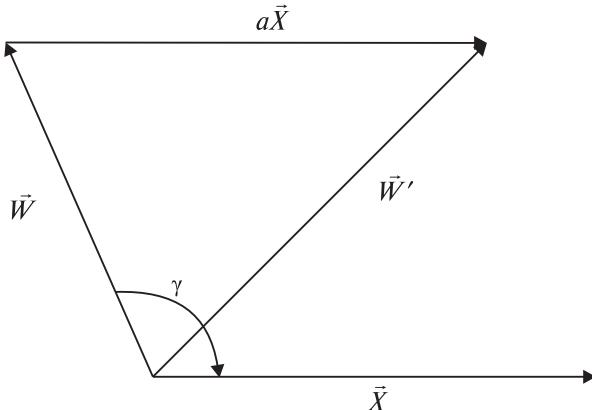


Рис. 2.16. Геометрическое представление обучения персептрона при  $\gamma$  больше  $90^\circ$

Рассмотрим теперь другой случай. Пусть  $\gamma < 90^\circ$ . Тогда  $S > 0$  и  $y = 1$ . Предположим, что эталонное значение  $e = 0$ . В таком случае вектор  $W$  необходимо преобразовать так, чтобы угол  $\gamma$  стал больше  $90^\circ$ . Это эк-

вивалентно вращению вектора  $\vec{W}$  по часовой стрелке. Таким образом, правило модификации вектора  $\vec{W}$  можно представить как

$$\vec{W}' = \vec{W} - \alpha \vec{X}. \quad (2.10)$$

Данный процесс показан на рис. 2.17.

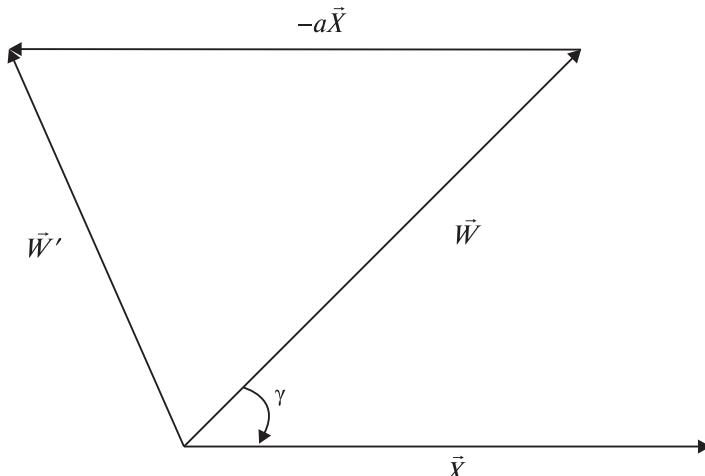


Рис. 2.17. Геометрическое представление обучения персептрона при  $\gamma$  меньше  $90^\circ$

Пусть  $\vec{W}' = \vec{W}(t+1)$ , а  $\vec{W} = \vec{W}(t)$ . В таком случае, объединяя два приведенные выше правила в одно, получим следующее выражение для настройки весовых коэффициентов:

$$\vec{W}(t+1) = \vec{W}(t) - \alpha(y - e)\vec{X} = \vec{W}(t) + \alpha(e - y)\vec{X}. \quad (2.11)$$

Последнее выражение характеризует правило обучения Розенблатта в общей форме.

## 2.7. ПРИМЕРЫ РЕШЕНИЯ ЗАДАЧ ОДНОСЛОЙНЫМ ПЕРСЕПТРОНОМ

Рассмотрим процедуру обучения Розенблатта на примере решения задачи «ЛОГИЧЕСКОЕ ИЛИ».

Пусть размерность входного сигнала  $n = 2$ . Тогда архитектура сети состоит из двух нейронов входного и одного нейрона выходного слоя (см. рис. 2.11). Значения параметров, полученных в результате процедуры пошагового обучения, представлены в табл. 2.1.

Таблица 2.1

№ шага	$x_1$	$x_2$	$S$	$y_1$	$e$	$w_{11}$	$w_{21}$	$T$
1	1	1	0	0	1	1	1	-1
2	1	0	2	1	1	1	1	-1
3	0	1	2	1	1	1	1	-1
4	0	0	1	1	0	1	1	0

Если дальше входные образы из обучающей выборки подавать на вход сети, то синаптические связи не изменяются. Подача всех входных образов из обучающей выборки на вход сети называется *эпохой*. В начальный момент времени для упрощения вычислений предполагалось, что весовые коэффициенты и пороговое значение равняются нулю, а скорость обучения  $\alpha = 1$ , хотя на практике это не рекомендуется делать.

В результате обучения получается следующее уравнение разделяющей линии:

$$x_1 + x_2 = 0.$$

Геометрическая интерпретация решения задачи «ЛОГИЧЕСКОЕ ИЛИ» приведена на рис. 2.18.

Как следует из рис. 2.18, получено не самое лучшее решение задачи, поскольку разделяющая прямая располагается на границе решения. Это происходит из-за неудачно выбранных начальных параметров сети (шага обучения и синаптических связей).

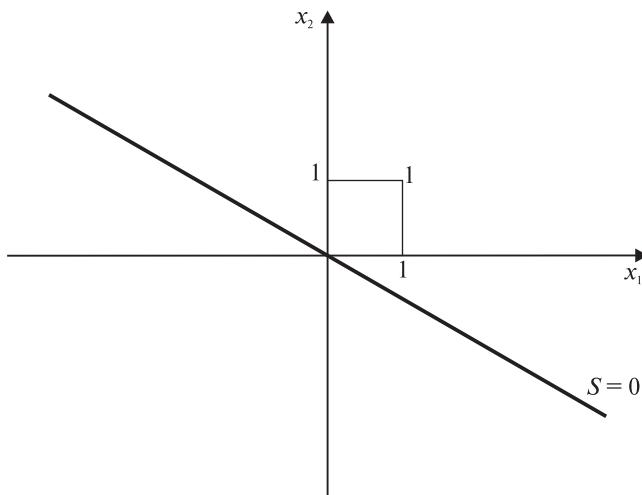


Рис. 2.18. Решение задачи «ЛОГИЧЕСКОЕ ИЛИ»

Рассмотрим геометрический способ настройки синаптических связей для решения задачи «ЛОГИЧЕСКОЕ ИЛИ» (обратная задача).

Пусть задана разделяющая линия для решения задачи «ЛОГИЧЕСКОЕ ИЛИ» (рис. 2.19). Необходимо найти весовые коэффициенты и пороговое значение однослойного персептрана. Дискриминантная линия описывается уравнением

$$x_2 = 0,5 - x_1.$$

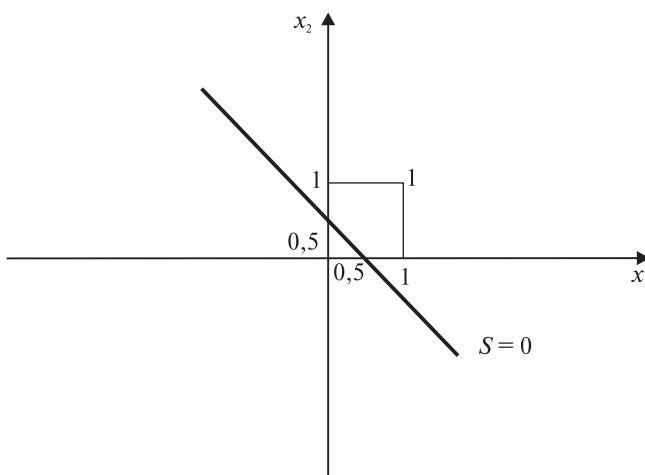


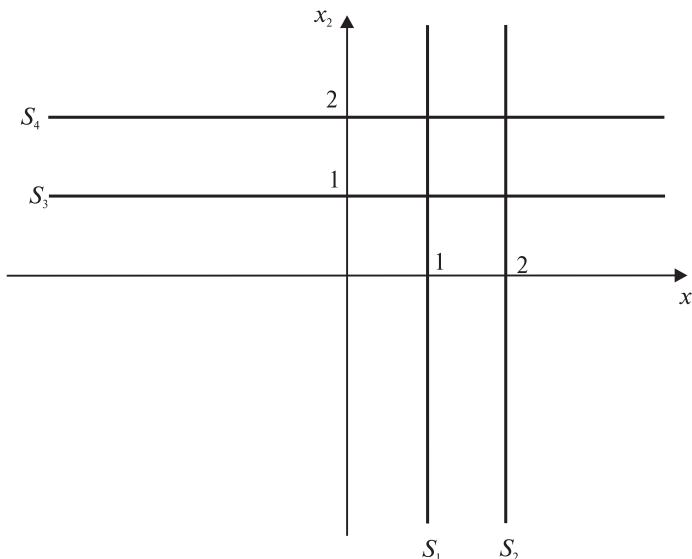
Рис. 2.19. Обратная задача «ЛОГИЧЕСКОЕ ИЛИ»

Поскольку в области выше разделяющей линии взвешенная сумма должна быть положительной ( $S > 0$ ), то  $S = x_2 + x_1 - 0,5$ . Учитывая, что для однослойного персептрана  $S = \omega_{11}x_1 + \omega_{21}x_2 - T$ , можно получить, что  $\omega_{11} = \omega_{21} = 1$ , а  $T = 0,5$ .

Приведем еще один пример геометрического способа настройки синаптических связей.

Пусть необходимо разбить двумерное пространство входных образов на классы, как показано на рис. 2.20, из которого следует, что общее количество классов равняется девяти.

Поскольку входное пространство образов двумерное, а разбиение на классы осуществляется четырьмя прямыми, то однослойный персептрон для решения этой задачи будет состоять из двух нейронов распределительного и четырех нейронов обрабатывающего слоя.



*Рис. 2.20.* Разбиение входного пространства образов на классы

Найдем синаптические связи такого персептрана. Так, уравнение первой прямой  $S_1$  имеет вид

$$x_1 = 1.$$

Если положительную область необходимо сделать справа от прямой  $S_1$ , т. е.  $S_1 > 0$ , то

$$S_1 = x_1 - 1.$$

Аналогичным образом для остальных прямых

$$S_2 : x_1 = 2 \rightarrow S_2 = x_1 - 2,$$

$$S_3 : x_2 = 1 \rightarrow S_3 = x_2 - 1,$$

$$S_4 : x_2 = 2 \rightarrow S_4 = x_2 - 2.$$

Как уже отмечалось, однослойный персептрон для решения данной задачи будет состоять из двух нейронов входного и четырех нейронов выходного слоя. Найдем синаптические связи. Так, для первого нейрона второго слоя

$$\left. \begin{array}{l} S_1 = x_1 - 1, \\ S_1 = w_{11}x_1 + w_{21}x_2 - T_1 \end{array} \right\} \rightarrow w_{11} = 1, w_{21} = 0, T_1 = 1.$$

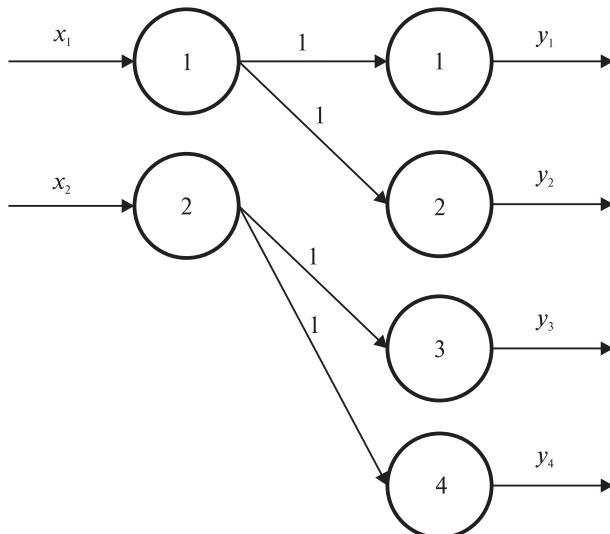


Рис. 2.21. Однослойный персептрон

Аналогично для остальных нейронов

$$w_{12} = 1, w_{22} = 0, T_2 = 2, \quad w_{13} = 0, w_{23} = 1, T_3 = 1, \quad w_{14} = 0, w_{24} = 1, T_4 = 2.$$

В результате однослойный персептрон для решения данной задачи будет иметь вид, как на рис. 2.21.

## 2.8. ПРАВИЛО ОБУЧЕНИЯ ВИДРОУ – ХОФФА (ДЕЛЬТА-ПРАВИЛО)

Данное правило используется для обучения однослойного персептрана, нейронные элементы обрабатывающего слоя которого имеют линейную функцию активации (рис. 2.22). Назовем однослойный персептрон с линейной функцией активации *линейным персептроном*.

Такая сеть называется адаптивным нейронным элементом или ADALINE (от англ. adaptive linear element) (предложено в 1960 г. Б. Видроу и М. Хоффом [14]). Правило обучения Видроу – Хоффа называется также *дельта-правилом* (delta rule). Выходное значение линейного персептрана определяется как

$$y_j = \sum_{i=1}^n \omega_{ij} x_i - T_j. \quad (2.12)$$

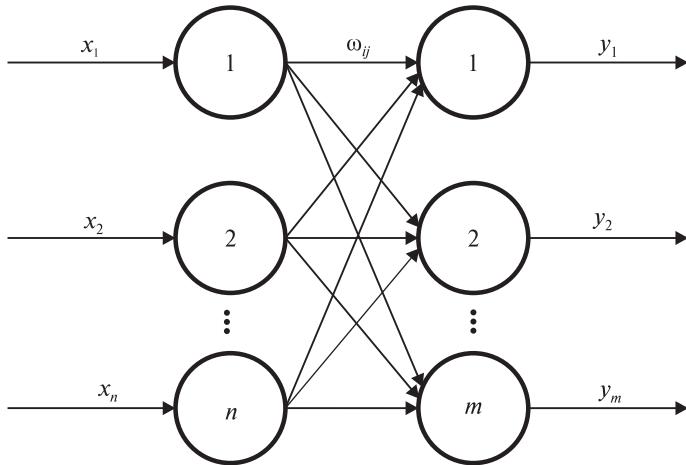


Рис. 2.22. Линейная сеть

Представим обучающую выборку в виде матрицы входных и эталонных образов размерности  $L$ :

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_n^1 \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \dots & \dots & \dots & \dots \\ x_1^L & x_2^L & \dots & x_n^L \end{bmatrix}, \quad e = \begin{bmatrix} e_1^1 & e_2^1 & \dots & e_m^1 \\ e_1^2 & e_2^2 & \dots & e_m^2 \\ \dots & \dots & \dots & \dots \\ e_1^L & e_2^L & \dots & e_m^L \end{bmatrix}.$$

Здесь  $x_i^k$  и  $e_j^k$  – соответственно входное и эталонное значение нейронной сети для  $k$ -го образа.

В общем, целью задачи обучения сети является достижение наилучшего отображения входных образов в эталонные. Это эквивалентно минимизации суммарной квадратичной ошибки нейронной сети, которая для  $L$  входных образов определяется следующим образом:

$$E_s = \sum_{k=1}^L E(k) = \frac{1}{2} \sum_{k=1}^L \sum_{j=1}^m (y_j^k - e_j^k)^2, \quad (2.13)$$

где  $E(k)$  – квадратичная ошибка сети для  $k$ -го образа;  $y_j^k$  и  $e_j^k$  – соответственно выходное и эталонное значение нейронной сети для  $k$ -го образа.

Критерий (2.13) характеризуется тем, что при малых ошибках ущерб представляет собой также малую величину, поскольку  $E$  меньше, чем

величина отклонения  $(y - e)$ . При больших ошибках ущерб возрастает, так как  $E$  увеличивается с ростом значения ошибки. Для минимизации суммарной квадратичной ошибки сети будем использовать метод градиентного спуска в пространстве весовых коэффициентов и пороговых значений сети. Существует два метода обучения: последовательное обучение (online learning) и групповое обучение (batch learning).

### 2.8.1. Последовательное обучение

*Последовательным* называется обучение, при котором модификация синаптических связей сети (веса и пороги) происходит после подачи каждого образа из обучающей выборки на нейронную сеть. В этом случае в методе градиентного спуска используется квадратичная ошибка нейронной сети для одного любого входного образа:

$$E = \frac{1}{2} \sum_{j=1}^m (y_j - e_j)^2. \quad (2.14)$$

Тогда в соответствии с методом градиентного спуска для минимизации суммарной квадратичной ошибки сети весовые коэффициенты и пороги нейронной сети необходимо изменять с течением времени по следующим выражениям:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial E}{\partial \omega_{ij}(t)}, \quad (2.15)$$

$$T_j(t+1) = T_j(t) - \alpha \frac{\partial E}{\partial T_j(t)}, \quad (2.16)$$

где  $i = \overline{1, n}$ ,  $j = \overline{1, m}$ ;  $\alpha$  – скорость или шаг обучения. Найдем производные среднеквадратичной ошибки  $E$  по настраиваемым параметрам сети  $\omega_{ij}$  и  $T_j$ :

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \omega_{ij}} = (y_j - e_j)x_i, \quad (2.17)$$

$$\frac{\partial E}{\partial T_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial T_j} = -(y_j - e_j). \quad (2.18)$$

Отсюда получим выражения для обучения нейронной сети, которые называются *дельта-правилом*:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha(y_j - e_j)x_i, \quad (2.19)$$

$$T_j(t+1) = T_j(t) + \alpha(y_j - e_j). \quad (2.20)$$

Последние выражения эквивалентны правилу обучения Розенблатта.

Алгоритм обучения однослойного персептрона, в основе которого лежит дельта-правило, состоит из следующих шагов:

- задается скорость обучения  $\alpha$  ( $0 < \alpha < 1$ ) и желаемая квадратичная ошибка сети  $E_e$ , которой необходимо достичь в процессе обучения;
- случайным образом инициализируются синаптические связи нейронной сети в достаточно узком диапазоне значений, например [0, 1]. Пороги могут инициализироваться нулевыми значениями

$$T_j(0) = 0;$$

• последовательно подаются входные образы из обучающей выборки на нейронную сеть и для каждого образа производятся следующие действия:

- вычисляются выходные значения сети;
- производится изменение весовых коэффициентов и порогов нейронной сети согласно выражениям (2.19) и (2.20);
- последовательно подаются входные образы из обучающей выборки на нейронную сеть и вычисляется значение суммарной квадратичной ошибки сети  $E_s$ ;
- алгоритм продолжается до тех пор, пока суммарная квадратичная ошибка сети не станет меньше заданной, т. е.  $E_s \leq E_e$ .

Другой критерий – остановка окончания алгоритма: алгоритм продолжается, пока не перестанут изменяться синаптические связи или уменьшаться значение суммарной квадратичной ошибки сети.

В алгоритме Видроу – Хоффа существует проблема выбора значения шага обучения  $\alpha$ . Если шаг  $\alpha$  слишком мал, то процесс обучения очень длительный. В случае когда шаг обучения большой, процесс обучения может оказаться расходящимся, т. е. может не привести к решению задачи. Таким образом, сходимость алгоритма обучения не избавляет от разумного выбора значения шага обучения. В некоторых работах предполагается выбирать значение  $\alpha$ , которое в процессе обучения уменьшается следующим образом:

$$\alpha = \frac{1}{p}, \quad (2.21)$$

где  $p$  – номер эпохи в алгоритме обучения. Напомним, что первая эпоха соответствует первой подаче в режиме обучения всех образов на ней-

ронную сеть, вторая эпоха – второй подаче и т. д. Однако здесь важным фактором является скорость, с которой величина  $\alpha$  приближается к нулю. Если скорость слишком велика, то процесс обучения может закончиться до того, как будут получены оптимальные результаты.

Таким образом, временная сложность алгоритма обучения и достижение оптимального решения зависят от выбора величины шага обучения.

### 2.8.2. Групповое обучение

*Групповым* называется обучение, при котором модификация синаптических связей происходит после подачи группы или всех образов из обучающей выборки на нейронную сеть. В этом случае в методе градиентного спуска используется суммарная квадратичная ошибка для группы из  $L_1$ -образов, после подачи которых на вход сети модифицируются синаптические связи:

$$\omega_{ij}(L_1) = \omega_{ij}(0) - \alpha \frac{\partial E_s(L_1)}{\partial \omega_{ij}}, \quad (2.22)$$

$$T_j(L_1) = T_j(0) - \alpha \frac{\partial E_s(L_1)}{\partial T_j}. \quad (2.23)$$

Здесь  $\omega_{ij}(0)$  и  $T_j(0)$  – синаптические связи до подачи  $L_1$ -образов из обучающей выборки на нейронную сеть,  $E_s(L_1)$  – суммарная квадратичная ошибка сети для  $L_1$ -образов. Поскольку

$$E_s(L_1) = \frac{1}{2} \sum_{k=1}^{L_1} \sum_{j=1}^m (y_j^k - e_j^k)^2,$$

$$y_j^k = \sum_{i=1}^n x_i^k w_{ij} - T_j, \quad (2.24)$$

то в случае группового обучения весовые коэффициенты и пороговые значения должны изменяться следующим образом:

$$\omega_{ij}(L_1) = \omega_{ij}(0) - \alpha(t) \sum_{k=1}^{L_1} (y_j^k - e_j^k) x_i^k, \quad (2.25)$$

$$T_j(L_1) = T_j(0) + \alpha(t) \sum_{k=1}^{L_1} (y_j^k - e_j^k). \quad (2.26)$$

Данные выражения минимизируют суммарную квадратичную ошибку сети для всех образов из обучающей выборки.

## 2.9. АДАПТИВНЫЙ ШАГ ОБУЧЕНИЯ

Для ускорения процедуры обучения градиентного спуска вместо постоянного шага обучения можно использовать *адаптивный шаг*  $\alpha(t)$  [25–27].

*Адаптивным* называется шаг обучения, который выбирается на каждом этапе алгоритма таким образом, чтобы минимизировать квадратичную ошибку сети. Рассмотрим вначале выбор адаптивного шага для последовательного обучения.

Пусть дана линейная нейронная сеть, которая состоит из распределительного слоя нейронных элементов и выходного слоя (рис. 2.23). В качестве нейронов выходного слоя используются нейронные элементы с линейной функцией активации. Каждый нейрон распределительного слоя имеет синаптические связи со всеми нейронами обрабатывающего слоя. Выходное значение  $j$ -го нейрона сети определяется как

$$y_j = \sum_i \omega_{ij} x_i - T_j. \quad (2.27)$$

Суммарная квадратичная ошибка сети для всей обучающей выборки вычисляется по формуле

$$E_s = \frac{1}{2} \sum_k \sum_j (y_j^k - e_j^k)^2. \quad (2.28)$$

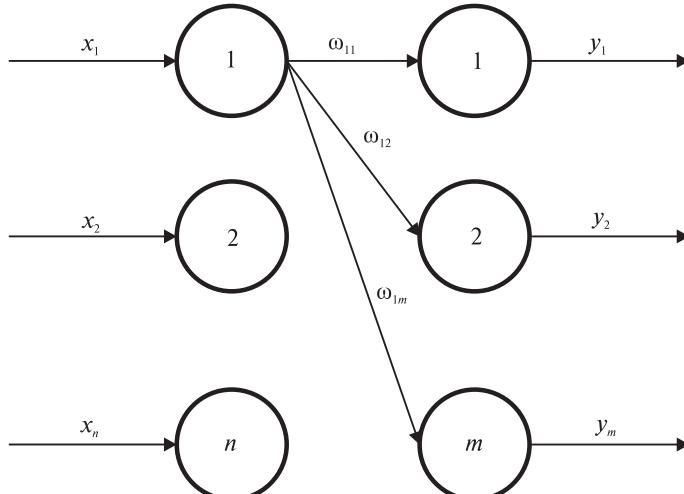


Рис. 2.23. Линейная нейронная сеть

Соответственно квадратичную ошибку для одного образа можно представить в виде

$$E = \frac{1}{2} \sum_j (y_j - e_j)^2. \quad (2.29)$$

Для нахождения адаптивного шага обучения будем использовать метод наискорейшего спуска.

На каждом шаге обучения нейронной сети необходимо выбирать такую скорость обучения  $\alpha(t)$ , чтобы минимизировать квадратичную ошибку:

$$\alpha(t) = \min E \left\{ \left( \omega_{ij}(t) - \alpha(t) \frac{\partial E}{\partial \omega_{ij}(t)} \right), \left( T_j - \alpha(t) \frac{\partial E}{\partial T_j(t)} \right) \right\}. \quad (2.30)$$

Подставляя в (2.27) выражения для изменения весовых коэффициентов и порогов нейронных элементов, получим

$$y'_j = \sum_i \left( \omega_{ij}(t) - \alpha(t) \frac{\partial E}{\partial \omega_{ij}(t)} \right) x_i - T_j(t) + \alpha(t) \frac{\partial E}{\partial T_j(t)}.$$

Запишем последнюю формулу в следующем виде:

$$y'_j = \sum_i \omega_{ij}(t) x_i - T_j(t) - \alpha(t) \left( \sum_i \frac{\partial E}{\partial \omega_{ij}(t)} x_i - \frac{\partial E}{\partial T_j(t)} \right). \quad (2.31)$$

Обозначим

$$b_j = \sum_i \left( \frac{\partial E}{\partial \omega_{ij}(t)} x_i - \frac{\partial E}{\partial T_j(t)} \right). \quad (2.32)$$

Тогда с учетом (2.32) выражение (2.31) можно записать как

$$y'_j = y_j - \alpha(t) b_j. \quad (2.33)$$

Для определения адаптивного шага обучения необходимо найти значение  $\alpha(t)$ , при котором среднеквадратичная ошибка минимальна:

$$E_1 = \frac{1}{2} \sum_j (y'_j - e_j)^2 = \frac{1}{2} \sum_j (y_j - \alpha b_j - e_j)^2 \rightarrow \min.$$

Тогда

$$\frac{\partial E_1}{\partial \alpha} = \sum_j \alpha b_j^2 - b_j (y_j - e_j) = 0.$$

Отсюда

$$\alpha = \frac{\sum_j b_j (y_j - e_j)}{\sum_j b_j^2}. \quad (2.34)$$

Поскольку  $\frac{\partial^2 E_1}{\partial^2 \alpha} > 0$ , то при заданной величине  $\alpha$  обеспечивается

минимум среднеквадратичной ошибки.

Найдем выражение для  $b_j$ . Поскольку

$$\begin{aligned} \frac{\partial E}{\partial \omega_{ij}(t)} &= (y_j - e_j)x_i, \\ \frac{\partial E}{\partial T_j} &= -(y_j - e_j), \end{aligned}$$

то

$$b_j = (y_j - e_j) \left( 1 + \sum_i x_i^2 \right). \quad (2.35)$$

Подставляя (2.35) в (2.34), получим окончательную формулу для адаптивного шага обучения на каждой итерации алгоритма:

$$\alpha(t) = \frac{1}{1 + \sum_i x_i^2(t)}. \quad (2.36)$$

В результате проведенных рассуждений доказана следующая теорема.

**Теорема 2.1.** Для линейной нейронной сети значение адаптивного шага обучения вычисляется на основе выражения (2.36).

Данная теорема справедлива, когда обучение происходит последовательно после подачи каждого образа на вход нейронной сети.

Рассмотрим групповое обучение, когда модификация синаптических связей производится после подачи на вход сети группы образов.

**Теорема 2.2.** Для линейной нейронной сети при групповом обучении величина адаптивного шага обучения определяется как

$$\alpha(t) = \frac{\sum_k \sum_j (y_j^k - e_j^k) a_j^k}{\sum_k \sum_j (a_j^k)^2},$$

$$a_j^k = \sum_p (y_j^p - e_j^p) \left( 1 + \sum_i x_i^p x_i^k \right), \quad (2.37)$$

где  $k = \overline{1, n}$ ;  $x_i^k$  –  $i$ -я компонента  $k$ -го образа. Данная теорема доказывается по аналогии с предыдущей.

Выражения (2.36) и (2.37) для выбора шага обучения нейронных сетей были получены в работах [25, 27]. Адаптивный шаг обучения позволяет значительно повысить скорость обучения линейной нейронной сети и достичь оптимального решения задачи.

## 2.10. ИСПОЛЬЗОВАНИЕ ПСЕВДООБРАТНОЙ МАТРИЦЫ ДЛЯ ОБУЧЕНИЯ ЛИНЕЙНЫХ НЕЙРОННЫХ СЕТЕЙ

Для настройки весовых коэффициентов линейной нейронной сети в целях минимизации среднеквадратичной ошибки можно использовать матричное решение системы линейных уравнений. Уравнение линейной нейронной сети в матричной форме можно представить в виде

$$Y = XW.$$

Матрицу выходных значений нейронной сети можно записать как

$$Y = \begin{bmatrix} y_1^1 & y_2^1 & \dots & y_m^1 \\ y_1^2 & y_2^2 & \dots & y_m^2 \\ \dots \\ y_1^L & y_2^L & \dots & y_m^L \end{bmatrix},$$

где  $L$  – размерность обучающей выборки;  $m$  – количество выходных нейронов сети.

Аналогичным образом запишем матрицу входных значений и весовых коэффициентов:

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_n^1 \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \dots \\ x_1^L & x_2^L & \dots & x_n^L \end{bmatrix}, \quad W = \begin{bmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1m} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2m} \\ \dots \\ \omega_{n1} & \omega_{n2} & \dots & \omega_{nm} \end{bmatrix}.$$

В случае использования порогов нейронных элементов в матрицу  $X$  добавляется строка, содержащая значения  $-1$ , а в матрицу  $W$  – строка с пороговыми значениями нейронных элементов  $T$ .

В работе [28] показано, что наилучшим приближенным решением системы линейных уравнений, при котором среднеквадратичная ошибка сети достигает своего наименьшего значения, является выражение

$$W = X^+ Y,$$

где  $X^+$  – псевдообратная матрица для матрицы  $X$ .

Наилучшее приближенное решение – единственное [28] и минимизирует суммарную квадратичную ошибку сети:

$$E = |Y - WX|^2.$$

*Псевдообратная матрица* – это обобщение обратной матрицы на случай произвольной размерности, определяемая следующим образом:

$$X^+ = (X^T X)^{-1} X^T.$$

Существуют различные способы нахождения псевдообратной матрицы [28], например метод Гревилля, который не требует вычисления детерминантов.

При использовании псевдообратной матрицы возникают проблемы, если матрица  $(X^T X)$  – вырожденная. В случае большой размерности входных сигналов определение псевдообратной матрицы требует значительных вычислительных затрат. Поэтому метод градиентного спуска имеет преимущество по сравнению с рассмотренным методом, особенно при использовании адаптивного шага обучения.

## 2.11. АНАЛИЗ ЛИНЕЙНЫХ НЕЙРОННЫХ СЕТЕЙ

Рассмотрим многослойную нейронную сеть, каждый нейрон которой имеет линейную функцию активации. Такая нейронная сеть представляет собой сеть с прямым распространением сигналов. Она характеризуется тем, что каждый нейронный элемент предыдущего слоя имеет синаптические связи со всеми нейронными элементами следующего слоя (рис. 2.24).

На рис. 2.24 изображены синаптические соединения только для одного нейрона каждого из слоев. Связи остальных нейронов будут идентичными.

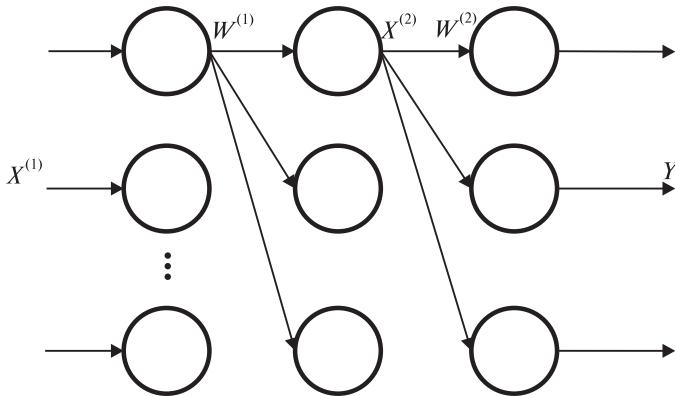


Рис. 2.24. Многослойная линейная сеть

**Теорема 2.3.** Многослойная нейронная сеть с линейной функцией активации нейронных элементов эквивалентна однослойной нейронной сети с соответствующей функцией активации нейронных элементов.

*Доказательство.* Выходные значения нейронной сети в матричной форме имеют вид

$$Y = X^{(2)}W^{(2)},$$

где  $X^{(2)}$  – матрица выходных сигналов промежуточного слоя;  $W^{(2)}$  – матрица весовых коэффициентов выходного слоя.

Аналогично матрицу выходных значений промежуточного слоя можно представить как

$$X^{(2)} = X^{(1)}W^{(1)}.$$

Отсюда получим, что  $Y = X^{(1)}W^{(1)}W^{(2)}$ .

Поскольку  $W^{(1)}W^{(2)} = W$ , то  $Y = X^{(1)}W$ , что и требовалось доказать.

Линейная нейронная сеть позволяет осуществить такое отображение входных сигналов в выходные, если между ними существует линейная зависимость. Такая сеть может решать задачи сложения или вычитания десятичных чисел, задачи линейного авторегрессионного анализа или использоваться в качестве памяти. Способность нейронной сети выдавать корректные результаты на данных, которые не входили в обучающую выборку, называется *обобщающей способностью* сети. Так, достаточно обучить линейную нейронную сеть небольшому количеству операций сложения десятичных чисел, чтобы она корректно складывала любые десятичные числа. Подробно применение линейных нейронных сетей будет рассмотрено далее.

## 2.12. ОДНОСЛОЙНЫЙ ПЕРСЕПТРОН ДЛЯ РЕШЕНИЯ ЗАДАЧИ «ИСКЛЮЧАЮЩЕЕ ИЛИ»

Однослойный персептрон с пороговой функцией активации формирует линейную разделяющую поверхность с точки зрения классификации образов и поэтому не может решить задачу «ИСКЛЮЧАЮЩЕЕ ИЛИ», что, как уже отмечалось, было доказано американскими учеными [15]. Однако последнее утверждение справедливо только для однослойного персептрана с пороговой или монотонной непрерывной функцией активации (например, сигмоидной). При использовании функции активации Гаусса или сигнальной функции однослойный персептрон может решить задачу «ИСКЛЮЧАЮЩЕЕ ИЛИ» ([29]). Покажем это для сигнальной функции активации. В этом случае вначале необходимо выделить область единиц или нулей при помощи двух прямых (рис. 2.25).

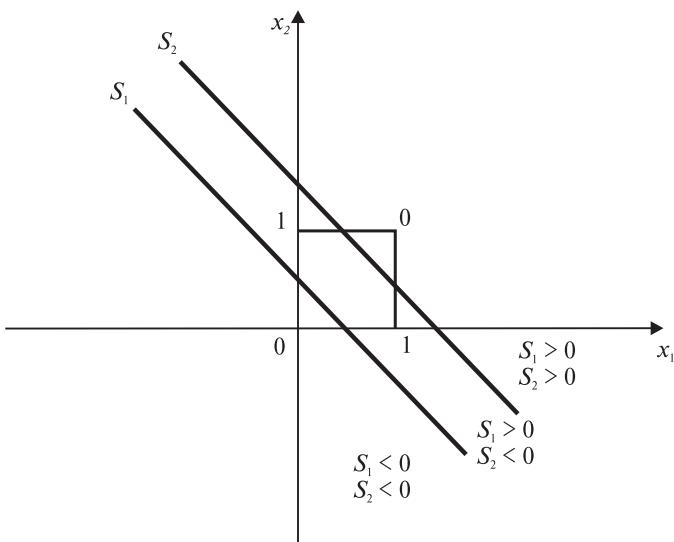


Рис. 2.25. Решение задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ»

Области  $A$ , характеризующей класс единиц, соответствует следующее условие:  $S_1 > 0 \wedge S_2 < 0$ . Уравнения двух прямых можно представить как

$$S_1 = x_2 + x_1 - 0,5,$$

$$S_2 = x_2 + x_1 - 1,5.$$

Отсюда  $S_2 = S_1 - 1$  (рис. 2.26).

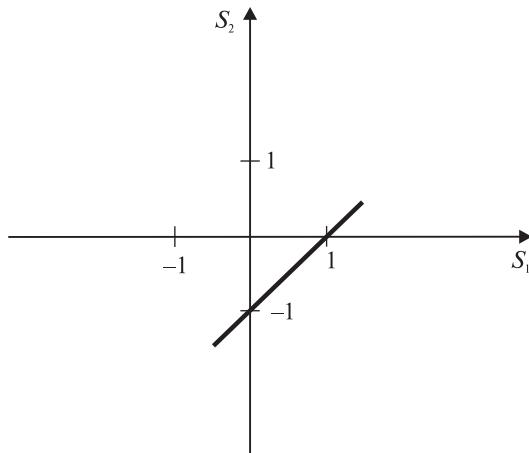


Рис. 2.26. Зависимость  $S_2$  от  $S_1$

Таким образом, для выделенной области  $A$ , когда

$$S_1 > 0 \wedge S_2 < 0 \rightarrow 0 < S_1 < 1,$$

можно получить следующую функцию активации (рис. 2.27):

$$y = F(S) = F(S_1) = \begin{cases} 1, & 0 < S_1 < 1, \\ 0, & \text{иначе.} \end{cases} \quad (2.38)$$

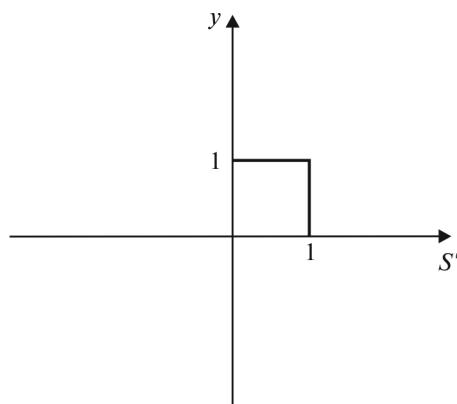
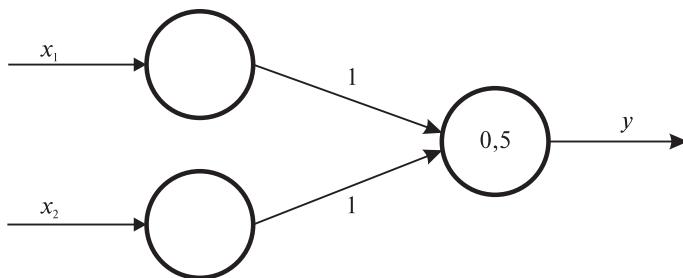


Рис. 2.27. Сигнальная функция активации

В результате однослойный персептрон для решения задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ» будет иметь веса и порог, как на рис. 2.28.



*Рис. 2.28. Однослойный персептрон  
для решения задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ»*

Таким образом, при использовании сигнальной функции активации однослойный персептрон может решить задачу «ИСКЛЮЧАЮЩЕЕ ИЛИ», так как она разбивает входное пространство образов на классы при помощи двух прямых. При использовании однослойного персептрана с радиально-базисной функцией активации для решения данной задачи необходимо принимать выходное значение персептрана, равное единице, если оно больше определенной величины.

## 2.13. ПРИМЕНЕНИЕ ОДНОСЛОЙНЫХ ПЕРСЕПТРОНОВ

Как уже отмечалось, однослойный персептрон может применяться для задач классификации образов, прогнозирования и т. д. В случае задачи классификации однослойный персептрон с пороговой или непрерывной, возрастающей, ограниченной функцией активации (например, сигмоидной) формирует линейную разделяющую поверхность, т. е. он способен решать только линейно разделимые задачи классификации. При использовании сигнальной функции активации он способен также решать некоторые линейно неразделимые задачи классификации, например «ИСКЛЮЧАЮЩЕЕ ИЛИ». В данном разделе рассмотрим решение системы линейных уравнений и задач прогнозирования при помощи линейного однослойного персептрана.

### 2.13.1. Решение системы линейных уравнений

Пусть дана следующая система линейных алгебраических уравнений (СЛАУ):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = y_1, \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = y_2, \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = y_m. \end{cases}$$

Здесь  $m$  – количество уравнений. Коэффициенты  $a_{11}, a_{12}, \dots, a_{mn}$  и свободные члены  $y_1, y_2, \dots, y_m$  являются известными. Необходимо определить  $x_1, x_2, \dots, x_n$ . Решение данной задачи сводится к нахождению таких значений  $X$ , чтобы минимизировать суммарную квадратичную ошибку между реальными выходными значениями сети и эталонными, где в качестве эталонных значений выступают свободные члены системы уравнений.

Структура линейного персептрана для решения СЛАУ изображена на рис. 2.29.

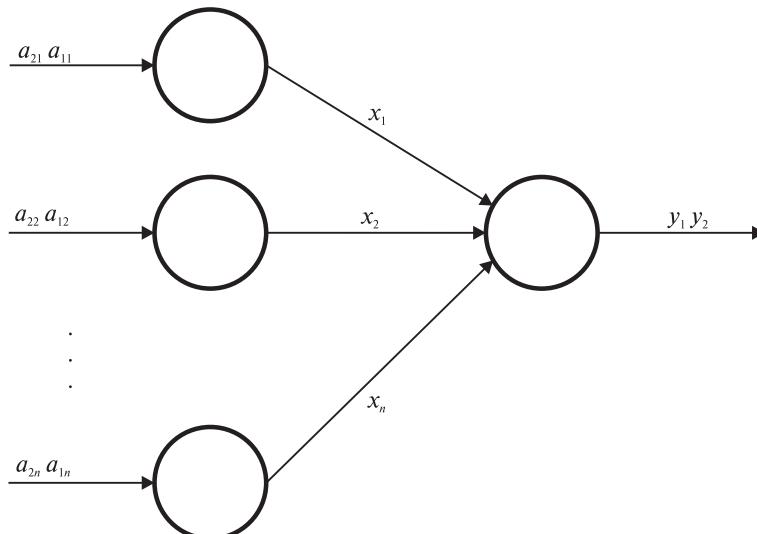


Рис. 2.29. Однослоиный персептрон для решения СЛАУ

В качестве входных данных нейронной сети выступают коэффициенты СЛАУ, а в качестве эталонных – свободные члены. Неизвестные  $X$ , которые нужно определить, являются весовыми коэффициентами однослойного персептрона. После обучения сети ее весовые коэффициенты и будут искомым решением задачи.

### 2.13.2. Использование линейной нейронной сети для прогнозирования

Способность нейронных сетей (после обучения) к *обобщению* и *пролонгации* результатов обучения создает потенциальные предпосылки для построения на их базе различного рода прогнозирующих систем. В данном разделе рассмотрим прогнозирование временных рядов при помощи линейных нейронных сетей.

Пусть дан временной ряд  $x(t)$  на промежутке  $t = \overline{1, m}$ . Тогда задача прогнозирования состоит в том, чтобы найти продолжение временного ряда на неизвестном промежутке, т. е. необходимо определить  $x(m+1)$ ,  $x(m+2)$  и т. д. (рис. 2.30).

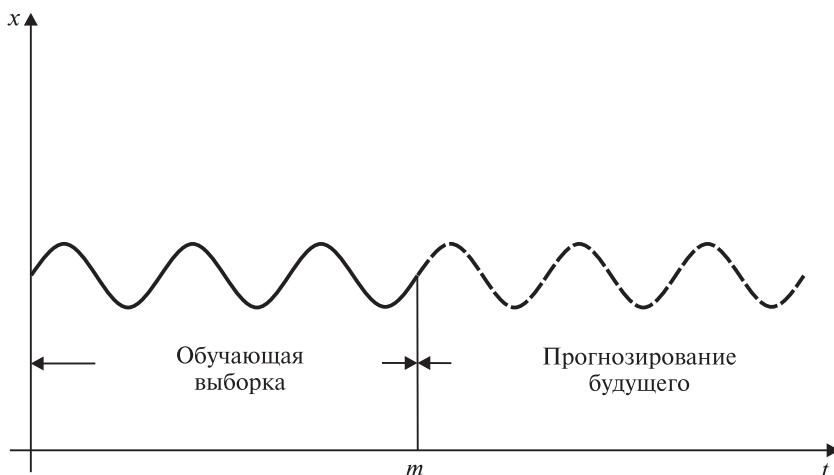


Рис. 2.30. Прогнозирование временного ряда

Совокупность известных значений временного ряда образует обучающую выборку, размерность которой определяется значением  $m$ . Для прогнозирования используется метод «скользящего окна». Он характеризуется длиной окна  $p$ , равной количеству элементов ряда, одновре-

менно подаваемых на нейронную сеть. Это определяет структуру нейронной сети, которая состоит из  $p$  распределительных и одного выходного нейрона.

Такая модель соответствует модели линейной авторегрессии и описывается выражением

$$\overline{x(t)} = \sum_{k=1}^p \omega_k x(t-p+k-1),$$

где  $\omega_k$ ,  $k = \overline{1, p}$ , – весовые коэффициенты нейронной сети;  $\overline{x(t)}$  – оценка значения ряда  $x(t)$  в момент времени  $t$ .

Ошибка прогнозирования определяется как

$$e(t) = \overline{x(t)} - x(t).$$

Модель линейной авторегрессии формирует значение ряда  $x(t)$  как взвешенную сумму предыдущих значений ряда. Обучающую выборку нейронной сети можно представить в виде матрицы, строки которой характеризуют векторы, подаваемые на вход сети:

$$X = \begin{bmatrix} x(1) & x(2) & \dots & x(p) \\ x(2) & x(3) & \dots & x(p+1) \\ \dots & \dots & \dots & \dots \\ x(m-p) & x(m-p+1) & \dots & x(m-1) \end{bmatrix}.$$

Это эквивалентно перемещению окна по ряду  $x(t)$  с единичным шагом.

Таким образом, для обучения нейронной сети прогнозированию используется выборка известных членов ряда. После обучения сеть должна прогнозировать временной ряд на упреждающий промежуток времени. Рассмотрим прогнозирование следующей функции:

$$y = 0,4\sin(0,3x) + 0,5.$$

Пусть размер окна –  $p = 4$ , а размерность обучающей выборки –  $L = 20$  для  $x = \overline{1, 20}$ . Тогда структура нейронной сети состоит из четырех входных и одного линейного нейрона. При обучении с постоянным шагом  $\alpha = 0,01$  достигается суммарная квадратичная ошибка  $E_s = 9,98 \cdot 10^{-3}$  при количестве итераций, равном 810. При обучении с адаптивным шагом суммарная квадратичная ошибка  $E_s = 9,98 \cdot 10^{-3}$  достигается при количестве итераций, равном 303. Таким образом, скорость обучения

с адаптивным шагом приблизительно в 2,7 раза больше, чем при использовании постоянного шага.

Таблица 2.2

Эталонное	Полученное	Отклонение	Квадрат отклонения
5,9867E-01	5,6086E-01	3,7809E-02	0,014295
6,4846E-01	6,0577E-01	4,2688E-02	0,0018222
6,9676E-01	6,5041E-01	4,6351E-01	0,0021485
7,4195E-01	6,9326E-01	4,8684E-02	0,0023701
7,8250E-01	7,3288E-01	4,9620E-02	0,0024622
8,1707E-01	7,6792E-01	4,9150E-02	0,0024157
8,4456E-01	7,9725E-01	4,7312E-02	0,0022384
8,6410E-01	8,1990E-01	4,4200E-02	0,0019536
8,7516E-01	8,3521E-01	3,9951E-02	0,0015961
8,7749E-01	8,4274E-01	3,4748E-02	0,0012074
8,7118E-01	8,4238E-01	2,8807E-02	0,0008298
8,5666E-01	8,3429E-01	2,2367E-02	0,0005003
8,3463E-01	8,1895E-01	1,5685E-02	0,000246
8,0609E-01	7,9706E-01	9,0249E-03	8,145E-05
7,7224E-01	7,6960E-01	2,6427E-03	6,984E-06
7,3450E-01	7,3771E-01	-3,2182E-03	1,036E-05
6,9437E-01	7,0271E-01	-8,3390E-03	6,954E-05
6,5347E-01	6,6600E-01	-1,2535E-02	0,0001571
6,1337E-01	6,2903E-01	-1,5659E-02	0,0002452
5,7561E-01	5,9322E-01	-1,7615E-02	0,0003103

Таблица 2.3

Эталонное	Полученное	Отклонение	Квадрат отклонения
5,9867E-01	5,9867E-01	1,9984E-15	3,9936E-30
6,4846E-01	6,4732E-01	1,1401E-03	1,2999E-06
6,9676E-01	6,9359E-01	3,1689E-03	1,0042E-05
7,4195E-01	7,3591E-01	6,0380E-03	3,6457E-05
7,8250E-01	7,7283E-01	9,6671E-03	9,3452E-05
8,1707E-01	8,0312E-01	1,3950E-02	1,9459E-04
8,4456E-01	8,2581E-01	1,8748E-02	3,5149E-04
8,6410E-01	8,4019E-01	2,3911E-02	5,7174E-04
8,7516E-01	8,4589E-01	2,9266E-02	8,5650E-04
8,7749E-01	8,4285E-01	3,4637E-02	1,1997E-03
8,7118E-01	8,3134E-01	3,9846E-02	1,5877E-03
8,5666E-01	8,1194E-01	4,4718E-02	1,9997E-03
8,3463E-01	7,8554E-01	4,9091E-02	2,4099E-03
8,0609E-01	7,5327E-01	5,2821E-02	2,7901E-03

*Окончание табл. 2.3*

Эталонное	Полученное	Отклонение	Квадрат отклонения
7,7224E-01	7,1645E-01	5,5787E-02	3,1122E-03
7,3450E-01	6,7660E-01	5,7897E-02	3,3521E-03
6,9437E-01	6,3528E-01	5,9090E-02	3,4916E-03
6,5347E-01	5,9413E-01	5,9338E-02	3,5210E-03
6,1337E-01	5,5472E-01	5,8650E-02	3,4398E-03
5,7561E-01	5,1854E-01	5,7063E-02	3,2562E-03

В табл. 2.2 и 2.3 приведены характеристики прогнозирующих свойств линейной нейронной сети, которая обучалась с постоянным и адаптивным шагом. При этом прогнозирование осуществлялось за пределами обучающей выборки на 20 шагов вперед.

## Глава 3

# МНОГОСЛОЙНЫЕ ПЕРСЕПТРОНЫ

В этой главе рассматриваются многослойные персептроны (multilayer perceptrons), представляющие собой многослойные нейронные сети с прямым распространением сигналов (multilayer feedforward neural networks). *Многослойный персепtron* способен осуществить любое отображение входных векторов в выходные. Это свойство отметили американские ученые еще в 1960-х гг. Однако на тот момент не было эффективного алгоритма обучения таких сетей. Поэтому их выводы по поводу перспектив развития многослойных персептронов были весьма пессимистичными. В 1986 г. Д. Румельхартом, Дж. Хинтоном и Р. Вильямсом независимо друг от друга был предложен *алгоритм обратного распространения ошибки* (backpropagation algorithm), который стал эффективным средством обучения многослойных персептронов [21]. До 2006 г. в научной среде была приоритетной парадигма, согласно которой персептрон с одним или максимум двумя скрытыми слоями является достаточным для решения различных задач, а использование персептрана с более чем двумя скрытыми слоями не имеет большого смысла. В настоящее время преобладает концепция, заключающаяся в том, что персептрон с более чем двумя скрытыми слоями более эффективен. В результате персептроны возродились под новым названием «глубокие нейронные сети» (deep neural networks), которые в общем случае представляют собой персептрон с более чем двумя скрытыми слоями. Благодаря многослойной архитектуре они позволяют обрабатывать и анализировать большие объемы данных, а также моделировать различные когнитивные процессы.

Также в главе описаны возможности и обучение многослойных персептронов, примеры их использования для решения различного рода прикладных задач [25–37].

### 3.1. ТОПОЛОГИЯ МНОГОСЛОЙНЫХ ПЕРСЕПТРОНОВ

Архитектура многослойного персептрана состоит из множества слоев нейронных элементов (рис. 3.1).

*Входной слой* (input layer) выполняет распределительные функции. *Выходной слой* (output layer) служит для обработки информации, полученной от предыдущих слоев, и выдачи окончательных результатов. Слои нейронных элементов, расположенные между входным и выходным слоями, называются *промежуточными* или *скрытыми* (hidden layers). Как и выходной, скрытые слои являются обрабатывающими. Выход каждого нейронного элемента предыдущего слоя нейронной сети соединен синаптическими связями со всеми входами нейронных элементов следующего слоя. Таким образом, топология многослойной нейронной сети однородна и регулярна.

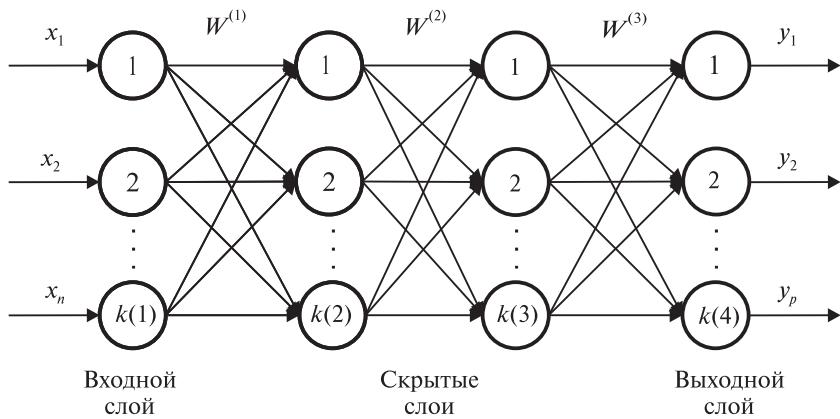


Рис. 3.1. Многослойный персептрон

Пусть  $W^{(i)}$  – матрица весовых коэффициентов  $i$ -го слоя многослойного персептрана. Тогда выходные значения  $Y$  для нейронной сети с двумя скрытыми слоями можно определить следующим образом:

$$Y = F(F(F(XW^{(1)})W^{(2)})W^{(3)}), \quad (3.1)$$

где  $X = (x_1, x_2, \dots, x_n)$  – вектор-строка входных сигналов;  $F$  – оператор нелинейного преобразования.

Общее количество синаптических связей многослойного персептрана равно

$$V = \sum_{i=1}^{p-1} k(i)k(i+1) + \sum_{i=1}^{p-1} k(i+1), \quad (3.2)$$

где  $p$  – общее количество слоев нейронной сети;  $k(i)$  – количество нейронных элементов в  $i$ -м слое.

## 3.2. АНАЛИЗ МНОГОСЛОЙНЫХ ПЕРСЕПТРОНОВ

Количество слоев характеризует способность многослойной нейронной сети к разбиению входного пространства образов на подпространства меньшей размерности. Как было показано в гл. 2, однослойный персептрон с помощью *гиперплоскости* разбивает входное пространство образов на классы. Персептрон с одним скрытым слоем и нелинейной функцией активации нейронных элементов позволяет формировать в пространстве решений *любые разделяющие поверхности*, как выпуклые, так и невыпуклые [29]. С помощью персептрана с двумя и более скрытыми слоями также можно получать области решений любой формы и сложности, в том числе и невыпуклой.

Рассмотрим возможности многослойного персептрана в зависимости от количества в нем скрытых слоев. По данному вопросу в литературе существует множество ошибок. Так, в работе [30] утверждается, что персептрон с одним скрытым слоем может формировать только выпуклую разделяющую поверхность с точки зрения классификации образов. Хотя позже было показано, что такой персептрон может формировать невыпуклую разделяющую поверхность. Данная ошибка продолжает попадать во многие учебники. Прежде всего следует отметить, что возможности персептрана с одним скрытым слоем различаются в зависимости от используемой в нем функции активации. Рассмотрим персептраны с различным количеством слоев.

### 3.2.1. Персептрон с одним скрытым слоем

В 1957 г. А. Н. Колмогоров показал [31], что любую непрерывную функцию  $n$  переменных на единичном отрезке  $[0, 1]$  можно представить в виде суммы конечного числа одномерных функций:

$$f(x_1, x_2, \dots, x_n) = \sum_{p=1}^{2n+1} g\left(\sum_{i=1}^n \lambda_i \varphi_p(x_i)\right), \quad (3.3)$$

где  $g$  и  $\varphi_p$  – одномерные и непрерывные функции;  $\lambda_i = \text{const}$  для всех  $i$ .

Данное утверждение легло в основу построения многослойных нейронных сетей для аппроксимации функций. Из него следует, что любую непрерывную функцию  $f : [0,1]^n \rightarrow [0,1]$  можно аппроксимировать нейронной сетью с одним скрытым слоем, если она имеет  $n$  входных нейронов,  $(2n+1)$  скрытых и один выходной. Основная проблема в этом случае связана с выбором соответствующих функций  $g$  и  $\phi$ . В 1989 г. ряд авторов [32, 33] обобщили приведенные выше результаты на многослойный персептрон с одним скрытым слоем, которые можно представить в виде следующей теоремы.

**Теорема 3.1.** Персептрон с одним скрытым слоем является *универсальным аппроксиматором*, т. е. он способен с любой степенью точности аппроксимировать любую непрерывную функцию, если в качестве функции активации нейронных элементов скрытого слоя используется непрерывная, монотонно возрастающая, ограниченная функция. При этом точность аппроксимации функции зависит от количества нейронов в скрытом слое. Чем больше количество нейронов, тем больше точность аппроксимации. Однако при слишком большой размерности скрытого слоя может наступить явление, которое называется перетренировкой сети, когда сеть имеет плохую обобщающую способность.

Приведенная выше теорема служит основой для аппроксимации и классификации функций с помощью многослойных нейронных сетей. Из теоремы следует, что в качестве функции активации нейронных элементов могут использоваться сигмоидальные функции активации: сигмоидная, bipolarная сигмоидная и гиперболический тангенс.

**Следствие 3.1.** Персептрон с одним скрытым слоем и непрерывной, монотонно возрастающей, ограниченной функцией активации нейронных элементов является *универсальным классификатором*.

Проведем анализ персептрона с одним скрытым слоем в качестве классификатора при использовании пороговой функции активации нейронных элементов (рис. 3.2).

Нейроны скрытого слоя представляют собой локальные детекторы признаков. Они разбивают входное пространство образов на области с помощью гиперплоскостей (прямых в двумерном случае). Число нейронов в скрытом слое характеризует количество таких областей:

$$c \leq 2^n. \quad (3.4)$$

Выходной слой нейронных элементов производит объединение полученных областей в соответствующие классы.

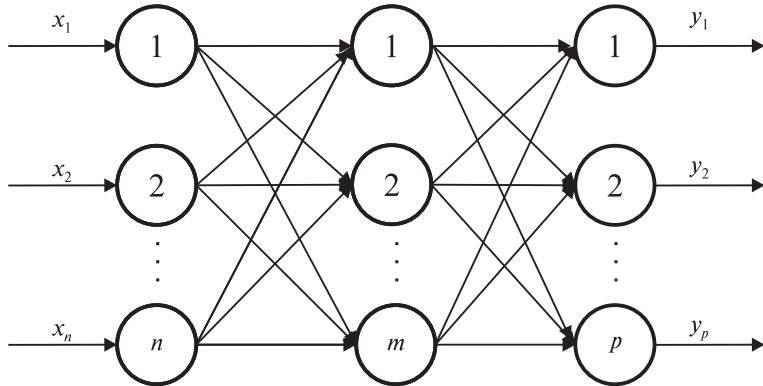


Рис. 3.2. Персептрон с одним скрытым слоем

Пусть  $p$  — количество классов, на которые необходимо разбить входное пространство образов. Число классов равно количеству нейронов выходного слоя. Для разбиения входного пространства образов на  $m$  классов необходимо, чтобы число областей было больше числа классов, т. е.  $c > p$ . Тогда количество нейронов в скрытом слое можно оценить следующим образом:

$$m > \log_2 p. \quad (3.5)$$

Рассмотрим возможности решения различных задач с помощью персептрона с одним скрытым слоем и бинарной пороговой функцией активации нейронных элементов, а также, что такой персептрон в общем случае не является универсальным классификатором.

**Формирование невыпуклой разделяющей поверхности.** Как уже отмечалось, во многих в работах утверждается, что персептрон с одним скрытым слоем может формировать только выпуклую разделяющую поверхность. Покажем на простом примере, что персептрон с одним скрытым слоем и бинарной пороговой функцией активации нейронных элементов позволяет формировать в пространстве решений невыпуклую разделяющую поверхность. На рис. 3.3. представлено разбиение входного пространства образов на два класса с помощью невыпуклой разделяющей линии, которая образуется тремя прямыми линиями  $S_1$ ,  $S_2$  и  $S_3$ .

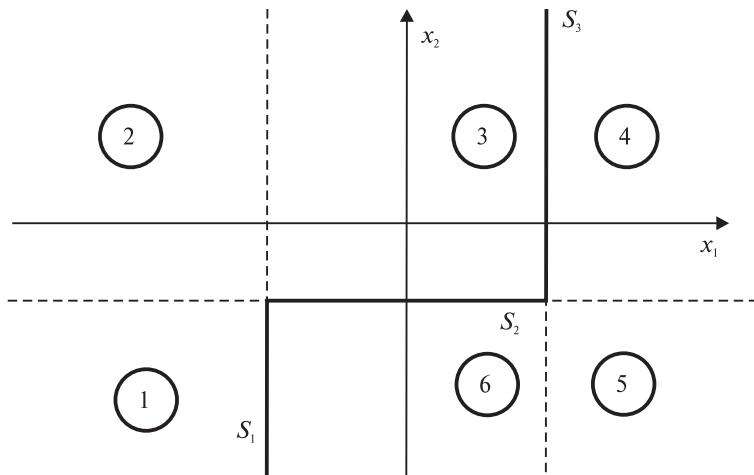


Рис. 3.3. Невыпуклая разделяющая поверхность

Построим персепtron, который решает данную задачу. Предположим, что справа от разделяющей линии находится класс единиц, а слева – класс нулей. Поскольку входное пространство образов двумерное, а разделяющая поверхность образуется тремя прямыми, то персепtron для бинарной классификации образов будет состоять из двух нейронов распределительного слоя, трех нейронов скрытого и одного выходного нейрона. Три нейрона скрытого слоя разбивают входное пространство образов на шесть областей (см. рис. 3.3). Найдем уравнения прямых линий и трансформируем их во взвешенные суммы нейронных элементов:

$$S_1 : x_1 = -1 \rightarrow S_1 = x_1 + 1,$$

$$S_2 : x_2 = -1 \rightarrow S_2 = x_2 + 1,$$

$$S_3 : x_1 = 1 \rightarrow S_3 = x_1 - 1.$$

Отсюда можно найти весовые коэффициенты и пороговые значения нейронных элементов скрытого слоя:

$$S_1 = \omega_{11}x_1 + \omega_{21}x_2 - T_1 \rightarrow \omega_{11} = 1, \omega_{21} = 0, T_1 = -1,$$

$$S_2 = \omega_{12}x_1 + \omega_{22}x_2 - T_2 \rightarrow \omega_{12} = 0, \omega_{22} = 1, T_2 = -1,$$

$$S_3 = \omega_{13}x_1 + \omega_{23}x_2 - T_3 \rightarrow \omega_{13} = 1, \omega_{23} = 0, T_3 = 1.$$

Поскольку используется бинарная пороговая функция активации нейронных элементов, то выходные значения нейронов скрытого слоя определяются следующим образом:

$$y_j = F(S_j) = \begin{cases} 1, & S_j > 0, \\ 0, & S_j \leq 0. \end{cases}$$

Найдем весовые коэффициенты и пороговое значение выходного нейронного элемента. Для этого проведем анализ работы персептрана в шести областях, на которые разбивают входное пространство образов три нейрона скрытого слоя. Результаты занесем в табл. 3.1.

Таблица 3.1

<i>A</i>	<i>y</i> <sub>1</sub>	<i>y</i> <sub>2</sub>	<i>y</i> <sub>3</sub>	<i>S</i>	<i>y</i>
1	0	0	0	-0,5	0
2	0	1	0	-1,5	0
3	1	1	0	-0,5	0
4	1	1	1	0,5	1
5	1	0	1	1,5	1
6	1	0	0	0,5	1

Исходя из анализа первых четырех столбцов таблицы и эталонных значений, которые необходимо получить в различных областях, можно найти взвешенную сумму и синаптические связи выходного нейрона:

$$S = y_1 - y_2 + y_3 - 0,5 \rightarrow \omega_1 = 1, \omega_2 = -1, \omega_3 = 1, T = 0,5.$$

Построенный персептран и его синаптические связи показаны на рис. 3.4.

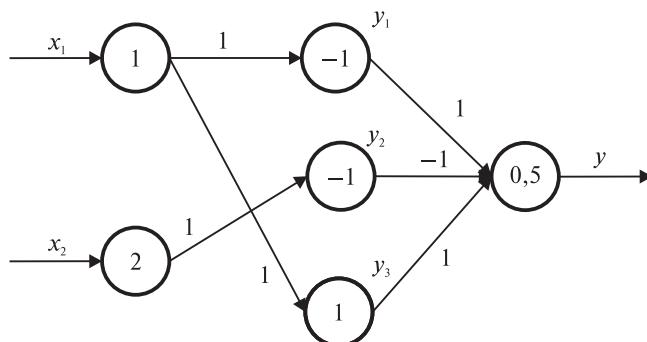


Рис. 3.4. Персептран для формирования невыпуклой разделяющей поверхности

**Задача «ИСКЛЮЧАЮЩЕЕ ИЛИ».** Проведем анализ решения данной задачи при помощи персептрана с одним скрытым слоем. На рис. 3.5 показаны весовые коэффициенты и пороговые значения нейронных элементов для решения задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ».

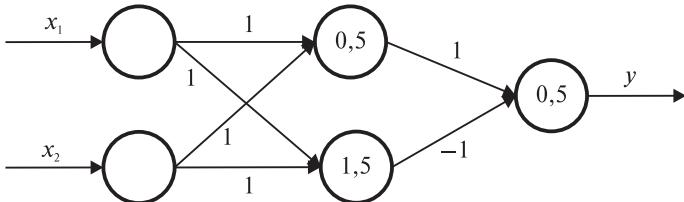


Рис. 3.5. Нейронная сеть для решения задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ»

В соответствии с рис. 3.5 выходные значения нейронных элементов скрытого слоя находятся следующим образом:

$$y_1 = F(S_1) = \begin{cases} 1, & S_1 > 0, \\ 0, & S_1 \leq 0, \end{cases}$$

$$S_1 = x_1 + x_2 - 0,5,$$

$$y_2 = F(S_2),$$

$$S_2 = x_1 + x_2 - 1,5.$$

Выходное значение нейрона последнего слоя определяется как

$$y = F(S),$$

$$S = y_1 - y_2 - 0,5.$$

Результаты функционирования персептрана представлены в табл. 3.2.

Таблица 3.2

$x_1$	$x_2$	$S_1$	$S_2$	$y_1$	$y_2$	$S$	$y$
0	0	-0,5	-1,5	0	0	-0,5	0
1	0	0,5	-0,5	1	0	0,5	1
0	1	0,5	-0,5	1	0	0,5	1
1	1	1,5	0,5	1	1	-0,5	0

Как показано на рис. 3.6, слой скрытых нейронных элементов производит разбиение входного пространства образов на области с помощью двух прямых.

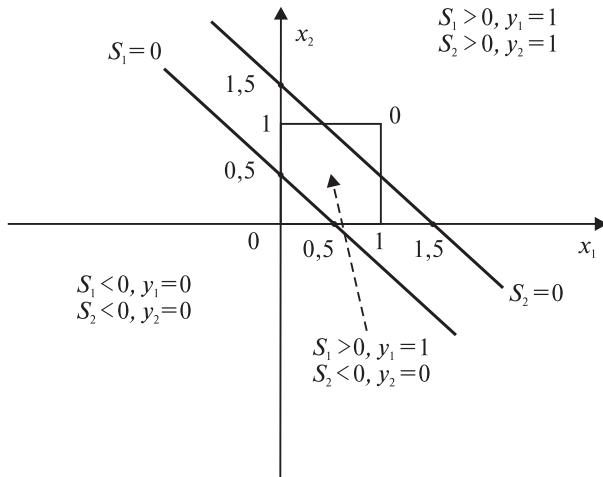


Рис. 3.6. Разбиение входного пространства образов

Выходной нейронный элемент осуществляет композицию полученных скрытым слоем областей в соответствующие классы для реализации функции «ИСКЛЮЧАЮЩЕЕ ИЛИ». Для этого он разбивает пространство образов скрытого слоя на классы (рис. 3.7).

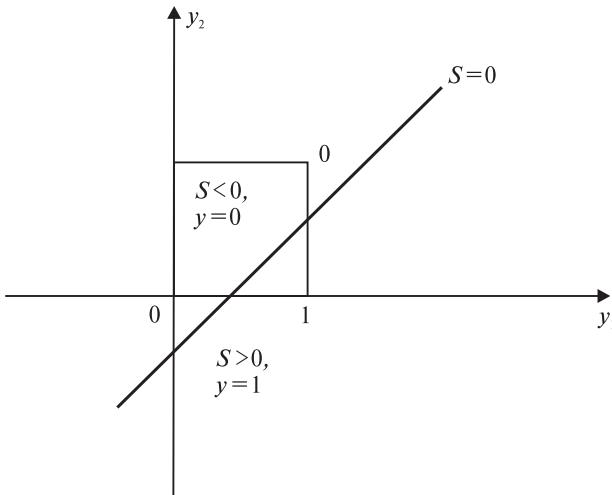


Рис. 3.7. Разбиение пространства образов скрытого слоя

Как следует из рис. 3.7, пространство образов скрытого слоя является линейно разделимым.

### 3.2.2. Персептрон с двумя скрытыми слоями

Как отмечалось выше, персептрон с одним скрытым слоем при использовании непрерывной, монотонной, возрастающей, ограниченной функции активации является *универсальным аппроксиматором*. Однако при слишком большой размерности скрытого слоя может наступить *перетренировка сети* (overfitting), имеющая плохую обобщающую способность. Поэтому вплоть до 2006 г. существовало мнение, что если на обучающей выборке с помощью одного скрытого слоя не удается получить требуемую точность аппроксимации функции и приемлемую обобщающую способность, то необходимо переходить к персептрону с двумя скрытыми слоями. Использование персептрона с более чем двумя скрытыми слоями считалось неэффективным.

Рассмотрим функции различных слоев нейронных элементов. Нейроны первого скрытого слоя служат локальными детекторами признаков. Они выделяют примитивные признаки из входного пространства образов. Второй скрытый слой нейронных элементов детектирует пространство признаков более высокого уровня абстракции на основе композиции признаков первого скрытого слоя. Выходной слой нейронных элементов отображает пространство признаков скрытого слоя в соответствующие классы.

Рассмотрим персептрон с двумя скрытыми слоями и пороговой функцией активации нейронных элементов для решения задачи бинарной классификации, изображенной на рис. 3.8.

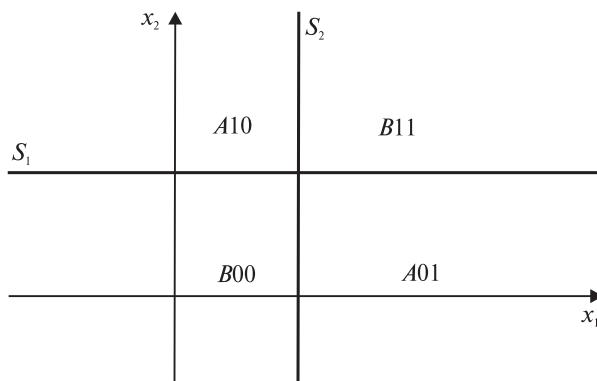


Рис. 3.8. Задача бинарной классификации двух классов  $A$  и  $B$

Предположим, что в правом верхнем и левом нижнем квадранте необходимо получить нуль (класс  $B$ ), а в остальных двух квадрантах – единицу (класс  $A$ ). Здесь ситуация меняется. Персептрон с одним скрытым слоем и пороговой функцией активации нейронных элементов не может решить такую задачу. В данном случае только персептрон с двумя скрытыми слоями способен сделать это.

Как видно из рис. 3.8, классы разделяются прямыми  $S_1$  и  $S_2$ . Уравнения, задающие прямые, описываются формулами:  $S_1 = x_2 - 1$  и  $S_2 = x_1 - 1$ . В результате данная задача сводится к проблеме «ИСКЛЮЧАЮЩЕЕ ИЛИ». Искомый персептрон для ее решения будет состоять из двух скрытых слоев, где первый слой формируют разделяющие линии  $S_1$  и  $S_2$ , а остальные слои нейронных элементов (рис. 3.9) представляют персептрон с одним скрытым слоем для решения задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ» (см. рис. 3.5).

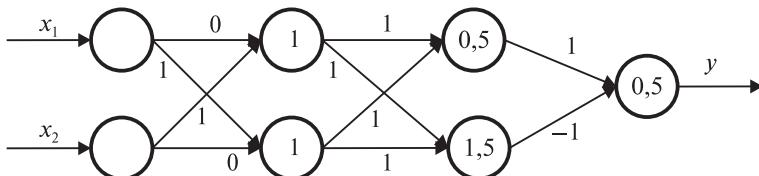


Рис. 3.9. Персептрон с двумя скрытыми слоями для решения задачи бинарной классификации

Таким образом, доказано следующее утверждение.

**Утверждение 3.1.** Персептрон с одним скрытым слоем и пороговой функцией активации нейронных элементов не является *универсальным классификатором*.

Рассмотрим еще один случай, когда используется сигнальная функция активации в отдельных слоях нейронных элементов. Тогда для решения приведенной выше задачи достаточно персептрана с одним скрытым слоем и сигнальной функцией активации в выходном нейроне, как это показано на рис. 3.10. Здесь нейроны скрытого слоя формируют дискриминантные линии  $S_1$  и  $S_2$ , а выходной нейронный элемент с сигнальной функцией активации – окончательное решение задачи.

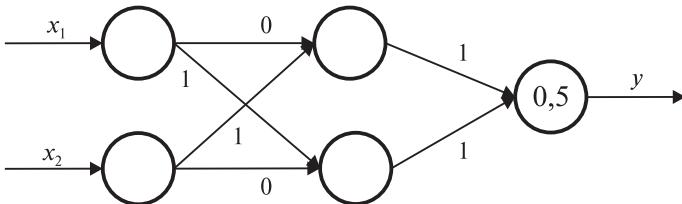


Рис. 3.10. Персептрон с одним скрытым слоем для решения задачи бинарной классификации

Как уже отмечалось, до 2006 г. основная научная парадигма заключалась в том, что персептрон с одним или максимум двумя скрытыми слоями является достаточным для решения различных задач. Использование персептрана с более чем двумя скрытыми слоями не имеет большого смысла.

### 3.3. НЕЙРОННЫЕ СЕТИ ВЫСОКОГО ПОРЯДКА

Нейронные сети *высокого порядка* (higher order networks) позволяют так же, как и многослойные нейронные сети, разбивать входное пространство образов на классы с помощью нелинейных разделяющих поверхностей [27]. Взвешенная сумма нейронных элементов многослойного персептрана содержит только слагаемые первого порядка. В отличие от этого взвешенная сумма нейронного элемента сети высокого порядка содержит произведения двух или более компонент входного вектора  $X$ . Так, для нейронной сети второго порядка

$$S = \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i x_j - T.$$

Разделяющая поверхность, определяемая уравнением  $S = 0$ , является поверхностью второго порядка и называется *гиперквадрикой*.

Если продолжить вводить дополнительные элементы в произведение компонент вектора  $X$ , например такие, как  $w_{ij} x_i x_j x_n$ , то получится класс полиномиальных разделяющих поверхностей. Рассмотрим решение задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ» с помощью нейронной сети второго порядка. Тогда взвешенная сумма равна

$$S = w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 - T.$$

На рис. 3.11 изображена структура такой сети для решения задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ».

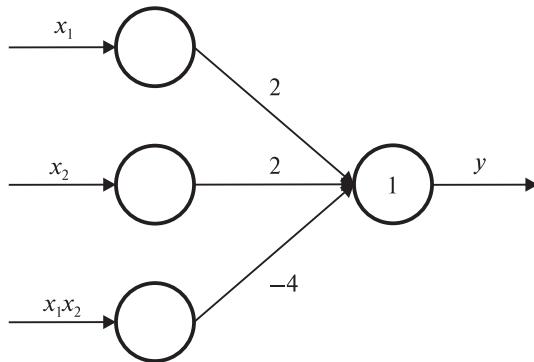


Рис. 3.11. Нейронная сеть второго порядка

В качестве функции активации выходного нейрона применяется пороговая функция. С учетом определенных на рис. 3.11 весовых коэффициентов и порогов нейронной сети взвешенную сумму можно представить как

$$S = 2x_1 + 2x_2 - 4x_1x_2 - T.$$

Для обучения нейронных сетей высокого порядка может использоваться алгоритм обратного распространения ошибки [34]. Однако более эффективно применение многослойных персепtronов по сравнению с нейронной сетью второго порядка.

### 3.4. МАТЕМАТИЧЕСКИЕ ОСНОВЫ АЛГОРИТМА ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

*Алгоритм обратного распространения ошибки*, предложенный в работе [34], представляет собой эффективное средство обучения многослойных нейронных сетей.

Рассмотрим нейронную сеть, состоящую из четырех слоев (рис. 3.12). Обозначим слои нейронных элементов от входа к выходу  $l, k, i, j$  соответственно.

Выходное значение  $j$ -го нейрона последнего слоя определяется следующим образом:

$$y_j = F(S_j), \quad (3.6)$$

$$S_j = \sum_i \omega_{ij} y_i - T_j, \quad (3.7)$$

где  $S_j$  – взвешенная сумма  $j$ -го нейрона выходного слоя;  $y_i$  – выходное значение  $i$ -го нейрона предпоследнего слоя;  $\omega_{ij}$  и  $T_j$  – весовой коэффициент и порог  $j$ -го нейрона выходного слоя соответственно.

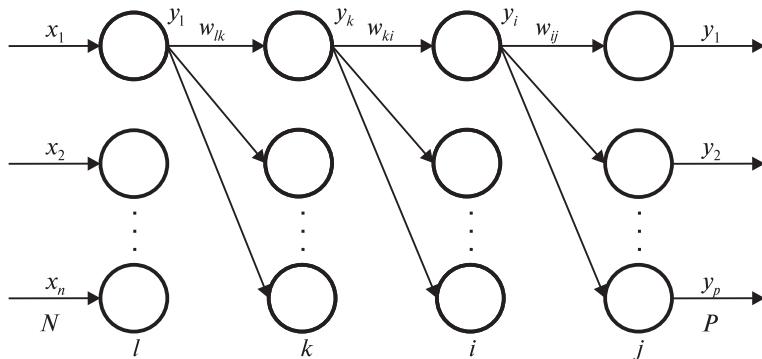


Рис. 3.12. Нейронная сеть с двумя скрытыми слоями

Аналогично выходное значение  $i$ -го нейрона предпоследнего слоя находится как

$$y_i = F(S_i), \quad (3.8)$$

$$S_i = \sum_k \omega_{ki} y_k - T_i. \quad (3.9)$$

Соответственно для  $k$ -го слоя

$$y_k = F(S_k), \quad (3.10)$$

$$S_k = \sum_l \omega_{lk} x_l - T_k. \quad (3.11)$$

Алгоритм обратного распространения ошибки минимизирует суммарную квадратичную ошибку нейронной сети, которая характеризует разницу между реальными и эталонными выходными значениями для всех образов обучающей выборки:

$$E_s = \frac{1}{2} \sum_{k=1}^L \sum_{j=1}^p (y_j^k - e_j^k)^2, \quad (3.12)$$

где  $y_j^k$  и  $e_j^k$  – выходное и эталонное значение нейронной сети для  $k$ -го образа соответственно;  $L$  – размерность обучающей выборки.

Для минимизации суммарной квадратичной ошибки сети будем использовать метод градиентного спуска в пространстве весовых коэффициентов и порогов нейронной сети. Как уже отмечалось в гл. 2, существует последовательное (online learning) и групповое (batch learning) обучение.

### 3.4.1. Последовательное обучение

В этом случае модификация синаптических связей сети (веса и пороги) происходит после подачи на нейронную сеть каждого образа обучающей выборки. Тогда для минимизации суммарной квадратичной ошибки сети, согласно методу градиентного спуска, весовые коэффициенты и пороговые значения нейронной сети должны изменяться следующим образом:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial E}{\partial \omega_{ij}(t)}, \quad (3.13)$$

$$T_j(t+1) = T_j(t) - \alpha \frac{\partial E}{\partial T_j(t)}, \quad (3.14)$$

где  $E$  – квадратичная ошибка нейронной сети для одного образа –

$$E = \frac{1}{2} \sum_j (y_j - e_j)^2, \quad (3.15)$$

где  $e_j$  – эталонное выходное значение  $j$ -го нейрона.

Ошибка  $j$ -го нейрона выходного слоя равна

$$\gamma_j = y_j - e_j. \quad (3.16)$$

**Теорема 3.2.** Для любого скрытого слоя  $i$  ошибка  $i$ -го нейронного элемента определяется рекурсивным образом через ошибки нейронов следующего  $j$ -го слоя:

$$\gamma_i = \sum_{j=1}^m \gamma_j F'(S_j) \omega_{ij}, \quad (3.17)$$

где  $m$  – количество нейронов следующего слоя по отношению к слою  $i$ ;  
 $\omega_{ij}$  – синаптическая связь между  $i$ -м и  $j$ -м нейроном различных слоев;

$S_j$  – взвешенная сумма  $j$ -го нейрона.

*Доказательство.* Ошибка  $i$ -го нейронного элемента рассчитывается как

$$\gamma_i = \frac{\partial E}{\partial y_i},$$

где  $y_i$  – выходное значение  $i$ -го нейрона.

Ошибка  $\gamma_i$  зависит от ошибки нейронных элементов следующего слоя  $j$  и вычисляется по формуле

$$\gamma_i = \frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial S_j} \frac{\partial S_j}{\partial y_i}.$$

Так как

$$\frac{\partial E}{\partial y_j} = \gamma_j, \quad \frac{\partial y_j}{\partial S_j} = F'(S_j), \quad \frac{\partial S_j}{\partial y_i} = \omega_{ij},$$

то

$$\gamma_i = \sum_j \gamma_j F'(S_j) \omega_{ij},$$

что и требовалось доказать.

Используя результаты теоремы, можно найти ошибки нейронов любого скрытого слоя через ошибки нейронов следующего слоя по отношению к скрытому. Так, например, ошибки нейронных элементов  $k$ -го слоя (рис. 3.12) определяются как

$$\gamma_k = \sum_i \gamma_i F'(S_i) \omega_{ki}.$$

**Теорема 3.3.** Производные среднеквадратичной ошибки по весовым коэффициентам и порогам нейронных элементов для любых двух слоев  $i$  и  $j$  многослойного персептрона вычисляются следующим образом:

$$\frac{\partial E}{\partial \omega_{ij}} = \gamma_j F'(S_j) y_i, \tag{3.18}$$

$$\frac{\partial E}{\partial T_j} = -\gamma_j F'(S_j). \tag{3.19}$$

*Доказательство.* Найдем производные среднеквадратичной ошибки для различных слоев нейронной сети (см. рис. 3.12). Тогда для выходного слоя имеем

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial S_j} \frac{\partial S_j}{\partial \omega_{ij}},$$

$$\frac{\partial E}{\partial T_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial S_j} \frac{\partial S_j}{\partial T_j}.$$

Если

$$\frac{\partial E}{\partial y_j} = \gamma_j = y_j - e_j, \quad \frac{\partial y_j}{\partial S_j} = F'(S_j), \quad \frac{\partial S_j}{\partial \omega_{ij}} = y_i, \quad \frac{\partial S_j}{\partial T_j} = -1,$$

то

$$\frac{\partial E}{\partial \omega_{ij}} = (y_j - e_j) F'(S_j) y_i,$$

$$\frac{\partial E}{\partial T_j} = -(y_j - e_j) F'(S_j).$$

Таким образом, для выходного слоя теорема доказана. Рассмотрим теперь скрытый слой  $i$ . Для него производные среднеквадратичной ошибки по весовым коэффициентам и порогам вычисляются как

$$\frac{\partial E}{\partial \omega_{ki}} = \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial S_j} \frac{\partial S_j}{\partial y_i} \frac{\partial y_i}{\partial S_i} \frac{\partial S_i}{\partial \omega_{ki}},$$

$$\frac{\partial E}{\partial T_i} = \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial S_j} \frac{\partial S_j}{\partial y_i} \frac{\partial y_i}{\partial S_i} \frac{\partial S_i}{\partial T_i}.$$

Поскольку

$$\gamma_i = \frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial S_j} \frac{\partial S_j}{\partial y_i} = \sum_j \gamma_j F'(S_j) \omega_{ij},$$

то

$$\frac{\partial E}{\partial \omega_{ki}} = \gamma_i \frac{\partial y_i}{\partial S_i} \frac{\partial S_i}{\partial \omega_{ki}}, \tag{3.20}$$

$$\frac{\partial E}{\partial T_i} = \gamma_i \frac{\partial y_i}{\partial S_i} \frac{\partial S_i}{\partial T_i}. \tag{3.21}$$

Найдем оставшиеся производные:

$$\frac{\partial y_i}{\partial S_i} = F'(S_i), \tag{3.22}$$

$$\frac{\partial S_i}{\partial \omega_{ki}} = y_k, \tag{3.23}$$

$$\frac{\partial S_i}{\partial T_i} = -1. \tag{3.24}$$

Подставляя (3.22), (3.23) и (3.24) в выражения (3.20) и (3.21), получим

$$\frac{\partial E}{\partial \omega_{ki}} = \gamma_i F'(S_i) y_k,$$

$$\frac{\partial E}{\partial T_i} = -\gamma_i F'(S_i),$$

что и требовалось доказать.

**Следствие 3.2.** Для минимизации суммарной квадратичной ошибки сети при последовательном обучении весовые коэффициенты и пороги нейронных элементов любых двух слоев  $i$  и  $j$  нейронной сети с течением времени должны изменяться следующим образом:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \gamma_j F'(S_j) y_i, \quad (3.25)$$

$$T_j(t+1) = T_j(t) + \alpha \gamma_j F'(S_j), \quad (3.26)$$

где  $\alpha$  – скорость обучения.

Данное следствие является очевидным. Оно определяет правило обучения многослойных персепtronов в общем виде, которое называется *обобщенным дельта-правилом*.

### 3.4.2. Групповое обучение

В случае *группового обучения* модификация синаптических связей происходит после подачи группы или всех образов из обучающей выборки на нейронную сеть. В этом случае в методе градиентного спуска используется суммарная квадратичная ошибка для группы из  $L_1$  образов, после подачи которых на вход сети модифицируются синаптические связи:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial E_s(L_1)}{\partial \omega_{ij}}, \quad (3.27)$$

$$T_j(t+1) = T_j(t) - \alpha \frac{\partial E_s(L_1)}{\partial T_j}. \quad (3.28)$$

Здесь  $\omega_{ij}(t)$  и  $T_j(t)$  – синаптические связи до подачи на нейронную сеть  $L$  образов из обучающей выборки,  $E_s(L_1)$  – суммарная квадратичная ошибка сети для  $L_1$ :

$$E_s(L_1) = \frac{1}{2} \sum_{k=1}^{L_1} \sum_{j=1}^p (y_j^k - e_j^k)^2.$$

Выходное значение  $j$ -го нейронного элемента для  $k$ -го образа

$$y_j^k = F(S_j^k), \quad (3.28)$$

$$S_j^k = \sum_i \omega_{ij} y_i^k - T_j. \quad (3.29)$$

С учетом результатов предыдущего раздела на основе теоремы 3.3 можно сформулировать следствие для группового обучения.

**Следствие 3.3.** В случае группового обучения для минимизации суммарной квадратичной ошибки весовые коэффициенты и пороговые значения любых двух слоев  $i$  и  $j$  нейронной сети должны изменяться следующим образом:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \sum_{k=1}^{L_1} \gamma_j^k F'(S_j^k) y_i^k, \quad (3.30)$$

$$T_j(t+1) = T_j(t) + \alpha \sum_{k=1}^{L_1} \gamma_j^k F'(S_j^k). \quad (3.31)$$

Здесь  $F'(S_j^k) = \frac{\partial y_j^k}{\partial S_j^k}$ .

Данное правило обучения называется *обобщенным дельта-правилом для группового обучения*.

**Примечание.** Метод градиентного спуска для последовательного или группового обучения, если используются группы образов небольшого размера (*minibatch*) и происходит случайный выбор образов из обучающей выборки, называется методом *стохастического градиентного спуска* (*stochastic gradient descent*, SGD).

### 3.5. ОБОБЩЕННОЕ ДЕЛЬТА-ПРАВИЛО ДЛЯ РАЗЛИЧНЫХ ФУНКЦИЙ АКТИВАЦИИ НЕЙРОННЫХ ЭЛЕМЕНТОВ

В выражениях для обобщенного дельта-правила присутствует производная функции активации нейронных элементов по взвешенной сумме:

$$F'(S_j) = \frac{\partial y_j}{\partial S_j}.$$

Определим ее для различных функций активации нейронных элементов.

### 3.5.1. Сигмоидная функция

Выходное значение  $j$ -го нейронного элемента в соответствии с сигмоидной функцией активации определяется следующим образом:

$$y_j = F(S_j) = \frac{1}{1 + e^{-S_j}}, \quad (3.32)$$

$$S_j = \sum_i \omega_{ij} y_i - T_j. \quad (3.33)$$

Тогда

$$F'(S_j) = \frac{\partial y_j}{\partial S_j} = \frac{e^{-S_j}}{(1 + e^{-S_j})^2} = \frac{1}{(1 + e^{-S_j})} - \frac{1}{(1 + e^{-S_j})^2} = y_j(1 - y_j). \quad (3.34)$$

В результате в случае последовательного обучения обобщенное дельта-правило для сигмоидной функции активации можно представить в виде

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \gamma_j y_j(1 - y_j) y_i, \quad (3.35)$$

$$T_j(t+1) = T_j + \alpha \gamma_j y_j(1 - y_j). \quad (3.36)$$

Ошибка для  $j$ -го нейрона выходного слоя вычисляется как

$$\gamma_j = y_j - e_j. \quad (3.37)$$

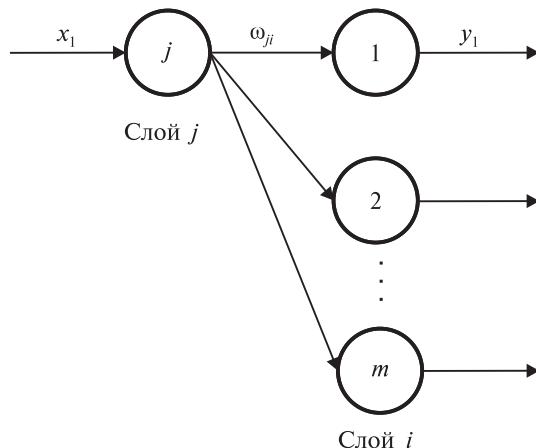


Рис. 3.13. Определение ошибки  $j$ -го нейронного элемента

Для  $j$ -го нейронного элемента скрытого слоя

$$\gamma_j = \sum_{i=1}^m \gamma_i y_i (1 - y_i) \omega_{ji}, \quad (3.38)$$

где  $m$  – количество нейронных элементов следующего слоя по отношению к слою  $j$  (рис. 3.13).

### 3.5.2. Биполярная сигмоидная функция

Выходное значение  $j$ -го нейрона определяется как

$$y_j = F(S_j) = \frac{2}{1 + e^{-S_j}} - 1. \quad (3.38)$$

Тогда

$$\begin{aligned} F'(S_j) &= \frac{\partial y_j}{\partial S_j} = \frac{2}{(1 + e^{-S_j})^2} = \frac{2}{(1 + e^{-S_j})} - \frac{2}{(1 + e^{-S_j})^2} = \\ &= -\frac{1}{2} \left( \frac{4}{(1 + e^{-S_j})^2} - \frac{4}{(1 + e^{-S_j})} + 1 - 1 \right) = -\frac{1}{2} \left( \left( \frac{2}{(1 + e^{-S_j})} - 1 \right)^2 - 1 \right). \end{aligned}$$

В результате получим выражение для производной биполярной сигмоидной функции активации:

$$F'(S_j) = \frac{\partial y_j}{\partial S_j} = \frac{1}{2}(1 - y_j^2). \quad (3.39)$$

Из (3.25) и (3.26) получим формулу для обучения нейронной сети с биполярной сигмоидной функцией активации:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \gamma_j \frac{1}{2}(1 - y_j^2) y_i, \quad (3.40)$$

$$T_j(t+1) = T_j(t) + \alpha \gamma_j \frac{1}{2}(1 - y_j^2). \quad (3.41)$$

Ошибка для  $j$ -го нейрона выходного и скрытого слоев соответственно находится как

$$\gamma_j = y_j - e_j, \quad (3.42)$$

$$\gamma_j = \frac{1}{2} \sum_i \gamma_i (1 - y_i^2) \omega_{ji}. \quad (3.43)$$

### 3.5.3. Гиперболический тангенс

Для данной функции активации выходное значение  $j$ -го нейрона находится следующим образом:

$$y_j = \text{th}(S_j) = \frac{e^{S_j} - e^{-S_j}}{e^{S_j} + e^{-S_j}}. \quad (3.44)$$

Учитывая, что функцию гиперболического тангенса можно представить в виде биполярной сигмоидной функции активации

$$y_j = \text{th}(S_j) = \frac{2}{1+e^{-2S_j}} - 1,$$

то производную функции гиперболического тангенса – как

$$F'(S_j) = \frac{\partial y_j}{\partial S_j} = (1 - y_j^2). \quad (3.45)$$

Тогда правило обучения можно записать в виде следующих выражений:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \gamma_j (1 - y_j^2) y_i, \quad (3.46)$$

$$T_j(t+1) = T_j(t) + \alpha \gamma_j (1 - y_j^2). \quad (3.47)$$

Ошибка для  $j$ -го нейрона выходного и скрытого слоев равна соответственно

$$\gamma_j = y_j - e_j, \quad (3.48)$$

$$\gamma_j = \sum_i \gamma_i (1 - y_i^2) \omega_{ji}. \quad (3.49)$$

### 3.5.4. Функция активации softmax

Как уже отмечалось выше, данная функция активации обычно применяется в последнем слое нейронной сети и используется в системах распознавания образов. В соответствии с этой функцией активации выходное значение  $r$ -го нейрона выходного слоя определяется как

$$z_r = \text{softmax}(S_r) = \frac{e^{S_r}}{\sum_j e^{S_j}}.$$

Тогда если  $r = j$ , то

$$F'(S_j) = \frac{\partial z_r}{\partial S_j} = \frac{\partial z_j}{\partial S_j} = \frac{-e^{2S_j}}{\left(\sum_j e^{S_j}\right)^2} + \frac{e^{S_j}}{\sum_j e^{S_j}} = z_j(1 - z_j) \quad (3.50)$$

и

$$F'(S_j) = \frac{\partial z_r}{\partial S_j} = -e^{S_r} \left(\sum e^{S_j}\right)^{-2} e^{S_j} = -z_r z_j, \text{ если } r \neq j.$$

Таким образом,

$$F'(S_j) = \frac{\partial z_r}{\partial S_j} = \begin{cases} z_j(1 - z_j), & r = j, \\ -z_r z_j, & r \neq j. \end{cases}$$

Рассмотрим обобщенное дельта-правило для последнего слоя нейронной сети (рис. 3.12), нейронные элементы которого используют функцию активации softmax. Найдем производные квадратичной ошибки по весовым коэффициентам и пороговым значениям. Квадратичная ошибка для одного образа равна

$$E = \frac{1}{2} \sum_{r=1}^p (z_r - e_r)^2.$$

В таком случае

$$\frac{\partial E}{\partial \omega_{ij}} = \sum_r \frac{\partial E}{\partial z_r} \frac{\partial z_r}{\partial S_j} \frac{\partial S_j}{\partial \omega_{ij}} = y_i \sum_r (z_r - e_r) \frac{\partial z_r}{\partial S_j},$$

$$\frac{\partial E}{\partial T_j} = \sum_r \frac{\partial E}{\partial z_r} \frac{\partial z_r}{\partial S_j} \frac{\partial S_j}{\partial T_j} = - \sum_r (z_r - e_r) \frac{\partial z_r}{\partial S_j}.$$

В результате для последнего слоя нейронной сети

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha y_i \sum_r (z_r - e_r) \frac{\partial z_r}{\partial S_j},$$

$$T_j(t+1) = T_j(t) + \alpha \sum_r (z_r - e_r) \frac{\partial z_r}{\partial S_j}.$$

Для остальных слоев нейронной сети обобщенное дельта-правило определяется аналогичным способом. Так, например, для скрытого слоя (см. рис. 3.12)

$$\omega_{ki}(t+1) = \omega_{ki}(t) - \alpha \gamma_i F'(S_i) y_k,$$

$$T_i(t+1) = T_i(t) + \alpha \gamma_i F'(S_i).$$

Ошибка нейронов скрытого слоя определяется как

$$\gamma_i = \frac{\partial E}{\partial y_i} = \sum_{j=1}^p \sum_{r=1}^p \frac{\partial E}{\partial z_r} \frac{\partial z_r}{\partial S_j} \frac{\partial S_j}{\partial y_i} = \sum_{j=1}^p \sum_{r=1}^p (z_r - e_r) \frac{\partial z_r}{\partial S_j} \omega_{ij}.$$

Обозначим

$$\gamma_j = \frac{\partial E}{\partial S_j} = \sum_{r=1}^p (z_r - e_r) \frac{\partial z_r}{\partial S_j}.$$

Тогда ошибка  $i$ -го нейрона скрытого слоя равна

$$\gamma_i = \sum_{j=1}^p \gamma_j \omega_{ij}.$$

### 3.5.5. Ректификационная функция активации ReLU

В этом случае

$$y_j = F(S_j) = \begin{cases} S_j, & S_j > 0, \\ kS_j, & S_j \leq 0, \end{cases}$$

где  $k = 0$  или принимает небольшое значение, например  $k = 0,01$  или  $k = 0,001$ . Тогда

$$\frac{\partial y_j}{\partial S_j} = F'(S_j) = \begin{cases} 1, & S_j > 0, \\ k, & S_j \leq 0. \end{cases}$$

Аналогично вычисляется производная линейной функции активации по взвешенной сумме.

Используя полученные в данном разделе выражения, можно определить алгоритм обратного распространения ошибки для различных функций активации нейронных элементов.

## 3.6. АЛГОРИТМ ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

Рассмотрим алгоритм обратного распространения ошибки (backpropagation algorithm) для последовательного обучения. Алгоритм был предложен в 1986 г. несколькими учеными независимо друг от друга. Он явля-

ется эффективным средством обучения нейронных сетей и состоит из следующих шагов:

1. Задается шаг обучения  $\alpha$  ( $0 < \alpha < 1$ ) и желаемая среднеквадратичная ошибка нейронной сети  $E_e$ .

2. Случайным образом инициализируются весовые коэффициенты и пороговые значения нейронной сети в достаточно узком диапазоне значений, например  $[-0,1; 0,1]$ .

3. Последовательно подаются входные образы из обучающей выборки  $k = 1, L$  на вход нейронной сети и для каждого входного образа выполняются следующие действия:

а) производится фаза прямого распространения входного образа по нейронной сети и вычисляется выходная активность всех нейронных элементов сети:

$$y_j = F\left(\sum_i \omega_{ij} y_i - T_j\right),$$

где индекс  $j$  характеризует нейроны следующего слоя по отношению к слою  $i$ ;

б) производится фаза обратного распространения сигнала, в результате которой определяется ошибка  $\gamma_j$ ,  $j = 1, 2, \dots$ , нейронных элементов для всех слоев сети. При этом для выходного и скрытого слоев соответственно

$$\gamma_j = y_j - e_j,$$

$$\gamma_j = \sum_i \gamma_i F'(S_i) \omega_{ji}.$$

В последнем выражении индекс  $i$  характеризует нейронные элементы следующего слоя по отношению к слою  $j$ ;

в) для каждого слоя нейронной сети происходит изменение весовых коэффициентов и порогов нейронных элементов в соответствии с обобщенным дельта-правилом:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \gamma_j F'(S_j) y_i,$$

$$T_j(t+1) = T_j(t) + \alpha \gamma_j F'(S_j).$$

4. Последовательно подаются входные образы из обучающей выборки  $k = 1, L$  и вычисляется суммарная квадратичная ошибка нейронной сети:

$$E_s = \frac{1}{2} \sum_{k=1}^L \sum_j (y_j^k - e_j^k)^2,$$

где  $L$  – размерность обучающей выборки.

5. Если  $E_s > E_e$  то происходит переход к шагу 3 алгоритма. В противном случае алгоритм обратного распространения ошибки заканчивается.

Таким образом, данный алгоритм функционирует до тех пор, пока суммарная квадратичная ошибка сети не станет меньше заданной, т. е.  $E_s \leq E_e$ .

Существует также другой критерий остановки алгоритма: алгоритм продолжается до тех пор, пока не перестанут изменяться синаптические связи или значения суммарной квадратичной ошибки сети. Аналогичным образом алгоритм обратного распространения ошибки определяется для группового обучения. В этом случае модификация синаптических связей сети происходит после подачи группы образов на нейронную сеть.

### 3.7. НЕДОСТАТКИ АЛГОРИТМА ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

При использовании для обучения многослойных нейронных сетей алгоритма обратного распространения ошибки, в основе которого лежит градиентный метод, могут возникнуть следующие проблемы:

- в общем случае неизвестен выбор количества слоев и количества нейронных элементов в слое для многослойных сетей;
- медленная сходимость градиентного метода с постоянным шагом обучения;
- выбор подходящей скорости обучения  $\alpha$ . Так, слишком малая скорость обучения увеличивает время обучения и может привести к скатыванию нейронной сети в ближайший локальный минимум функции суммарной квадратичной ошибки сети. Большая скорость обучения может привести к пропуску глобального минимума и сделать процесс обучения расходящимся;
- градиентный метод не различает точек локального и глобального минимумов (рис. 3.14). Рассмотрим общий вид зависимости суммарной квадратичной ошибки от настраиваемых параметров сети:

$$E_s = f(W, T).$$

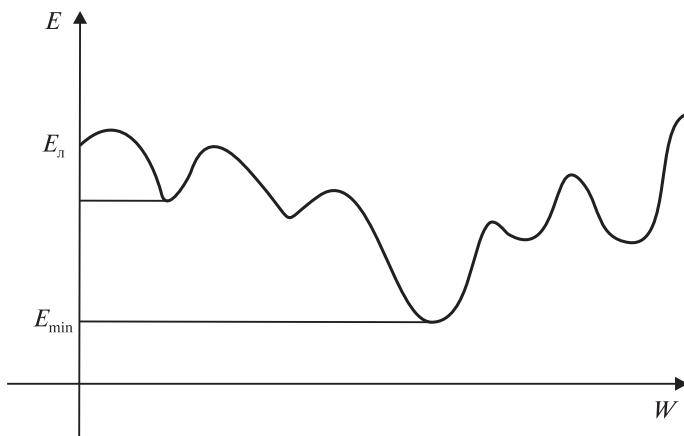


Рис. 3.14. Функция суммарной квадратичной ошибки:  
 $E_{\text{л}}$  – локальный минимум;  $E_{\text{min}}$  – глобальный минимум

Для многослойного персептрона такая зависимость имеет множество локальных минимумов (см. рис. 3.14). В общем случае алгоритм обратного распространения ошибки не позволяет достигнуть глобального минимума функции суммарной квадратичной ошибки сети. Однако это не умаляет его достоинств, так как для многих практических задач точка глобального минимума на обучающей выборке не является оптимальным решением задачи. *Основная задача обучения сети* – достижение суммарной квадратичной ошибки, при которой сеть обладает хорошей *обобщающей способностью*, при этом не имеет значения, какой минимум локальный или глобальный;

- влияние случайной инициализации весовых коэффициентов нейронной сети на поиск минимума функции суммарной квадратичной ошибки.

Последний пункт означает, что при разной инициализации синаптических связей могут получаться различные решения задачи. Это характеризует неустойчивость алгоритма обучения, когда нейронная сеть в одних случаях может обучаться до требуемой суммарной квадратичной ошибки, а в других нет. В следующем разделе данной главы будут рассмотрены модификации алгоритма обратного распространения ошибки в целях нейтрализации приведенных выше недостатков.

### 3.8. РЕКОМЕНДАЦИИ ПО ОБУЧЕНИЮ И АРХИТЕКТУРЕ МНОГОСЛОЙНЫХ НЕЙРОННЫХ СЕТЕЙ

Эффективность обучения многослойных нейронных сетей зависит от количества слоев, числа элементов в скрытых слоях нейронной сети и начальной инициализации весовых коэффициентов. Рассмотрим эти вопросы более подробно.

**Случайная инициализация.** Должна производиться в достаточно узком диапазоне значений. Как отмечалось в предыдущем разделе, разная инициализация весовых коэффициентов нейронной сети может приводить к различным решениям задачи. Важную роль здесь играет размер случайно инициализируемых синаптических связей [27]. Так, для сигмоидной функции активации нейронных элементов, когда весовые коэффициенты имеют большие значения (положительные и отрицательные), повышается вероятность того, что взвешенная сумма нейронных элементов также будет большой. В этом случае выходная активность нейронных элементов будет стремиться к единице или нулю. Соответственно значение выражения производной сигмоидной функции активации  $y_j(1 - y_j)$  будет близко к нулю и, согласно обобщенному дельта-правилу (3.35), весовые коэффициенты будут изменяться незначительно. Это приведет к тому, что процесс обучения может остановиться в ближайшем от стартовой точки локальном минимуме. То же самое можно показать для других функций активации.

Некоторые авторы рекомендуют выбирать значения весовых коэффициентов случайным образом в интервале

$$\omega_{ij}(0) \approx \left[ -\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}} \right],$$

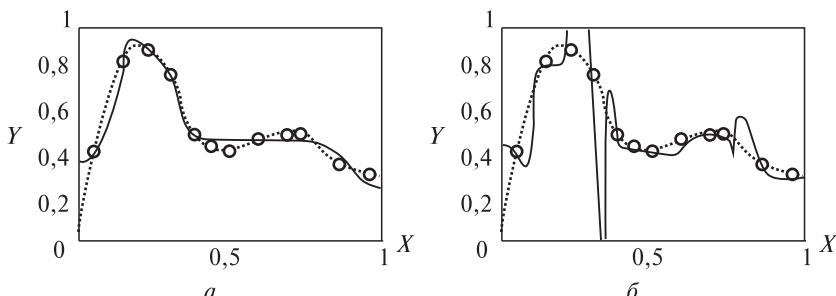
где  $m$  – количество нейронных элементов в слое  $i$ .

Другие авторы предлагают инициализировать весовые коэффициенты случайным образом в диапазоне  $[-0,05; 0,05]$  или  $[-0,1; 0,1]$ . При этом пороговые значения нейронных элементов в начальный момент времени могут устанавливаться в нулевые значения, т. е.  $T(0) = 0$ .

**Архитектура нейронной сети.** Большую роль для эффективности обучения нейронной сети играет ее архитектура. Рассмотрим традиционный подход к выбору архитектуры персептрона. Размерность входного и выходного слоев нейронной сети определяется из условия решаемой задачи или обучающей выборки. Как было отмечено в разделе 3.1, с помощью персептрона с одним скрытым слоем и сигмоидальной

функции активации нейронных элементов можно аппроксимировать любую функцию со сколь угодно заданной точностью. При этом точность аппроксимации зависит от количества нейронов в скрытом слое.

Чем больше количество нейронных элементов в скрытом слое, тем выше точность аппроксимации функции на обучающей выборке. Однако при слишком большой размерности скрытого слоя может возникнуть ситуация перетренировки сети, когда сеть имеет низкую обобщающую способность. На рис. 3.15 изображены примеры обучения нейронной сети для аппроксимации функции: рис. 3.15, *а* соответствует нейронной сети с пятью скрытыми нейронными элементами, а рис. 3.15, *б* – с 20. Объем обучающей выборки при этом равен 12. Как следует из рис. 3.15, *а*, *б*, нейронная сеть с пятью скрытыми нейронами достаточно точно аппроксимирует функцию на данных, которые не использовались в процессе обучения (сплошная линия). Нейронная сеть с 20 скрытыми нейронами хорошо аппроксимирует функцию только на обучающих наборах (на рис. 3.15 они представлены кружками). Отсюда следует, что слишком большое количество нейронов в скрытом слое ухудшает обобщающую способность нейронных сетей. Поэтому их количество должно быть меньше количества тренировочных образцов. С другой стороны, при слишком малой размерности скрытого слоя можно не получить желаемой суммарной квадратичной ошибки сети, поэтому необходим разумный компромисс между требуемой обобщающей способностью и желаемой суммарной квадратичной ошибкой сети. Для хорошего обучения сети необходимо, чтобы объем обучающей выборки  $L$  был больше количества настраиваемых параметров сети  $V$ , т. е.  $L > V$ .



*Рис. 3.15.* Обучение нейронной сети:  
а – сеть имеет пять скрытых нейронов;  
б – количество скрытых нейронов равно 20

Количество настраиваемых параметров для сети с архитектурой  $n-m-p$  (см. рис. 3.2) находится следующим образом:

$$V = nm + mp + m + p = m(n + p + 1) + p,$$

тогда

$$L > m(n + p + 1) + p.$$

Исходя из этого можно оценить верхнюю границу количества нейронных элементов в скрытом слое:

$$m < \frac{L - p}{n + p + 1}. \quad (3.51)$$

С учетом выражения (3.5), которое определяет нижнюю границу количества нейронов в скрытом слое для решения задачи классификации образов, общее выражение для количества нейронов в скрытом слое имеет вид

$$\log_2 p < m < \frac{L - p}{n + p + 1}. \quad (3.52)$$

Если при использовании персептрона с одним скрытым слоем не удается получить требуемую точность и обобщающую способность сети, то применяется нейронная сеть с двумя скрытыми слоями.

**Выход из локальных минимумов.** Как отмечалось ранее, одним из недостатков метода градиентного спуска является возможное попадание функции суммарной квадратичной ошибки сети в нежелательные локальные минимумы. Для выхода из небольших локальных минимумов можно использовать метод тяжелого шарика [35]. В этом случае модификация синаптических связей любых двух слоев  $i$  и  $j$  нейронной сети происходит в соответствии со следующим выражением:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \gamma_j F'(s_j) y_i + \beta(\omega_{ij}(t) - \omega_{ij}(t-1)), \quad (3.53)$$

где  $0 < \beta < 1$  – величина постоянная, которая называется моментным параметром.

Исходя из выражения (3.53) можно сделать вывод, что метод тяжелого шарика заключается в добавлении в обобщенное дельта-правило дополнительного слагаемого, позволяющего выбраться из небольших локальных минимумов. Обычно данный метод записывается в виде выражения

$$\Delta\omega_{ij}(t+1) = -\alpha \gamma_j F'(s_j) y_i + \beta \Delta\omega_{ij}(t). \quad (3.54)$$

На практике часто используется значение  $\beta = 0,9$  [27]. Последнее слагаемое в (3.54) называется *моментным термом* (momentum term).

**Регуляризация параметров сети (weight decay).** Как уже отмечалось, если синаптические связи сети принимают большие по абсолютной величине значения, то взвешенная сумма нейронных элементов в случае сигмоидной функции активации стремится к нулю или к единице. В результате ухудшается обобщающая способность сети и увеличивается вероятность остановки процесса обучения в нежелательном локальном минимуме. Во избежание переобучения нейронной сети используется регуляризация ее настраиваемых параметров, которая лимитирует неограниченный рост весов и порогов. Рассмотрим последовательный метод обучения. В этом случае в квадратичную ошибку сети добавляют слагаемое  $E_2$ , которое характеризует штраф за использование больших значений параметров:

$$E = E_1 + E_2 = \frac{1}{2} \sum_j (y_j - e_j)^2 + \frac{1}{2} \sum_{i,j} \omega_{ij}^2.$$

Тогда

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial E_1}{\partial \omega_{ij}(t)} - \lambda \frac{\partial E_2}{\partial \omega_{ij}(t)},$$

где  $0 < \lambda < 1$  – параметр регуляризации.

В результате обобщенное дельта-правило можно представить в следующем виде:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \gamma_j F'(S_j) y_i - \lambda \omega_{ij}(t).$$

Отсюда, приведя подобные слагаемые, получим

$$\omega_{ij}(t+1) = \omega_{ij}(t)(1 - \lambda) - \alpha \gamma_j F'(S_j) y_i. \quad (3.55)$$

Для пороговых значений

$$T_j(t+1) = T_j(t)(1 - \lambda) + \alpha \gamma_j F'(S_j). \quad (3.56)$$

Аналогичные выражения можно легко получить для группового способа обучения.

**Метод раннего останова (early stopping).** Этот метод используется для достижения максимальной обобщающей способности сети. В данном случае процесс обучения останавливается, когда нейронная сеть имеет максимальную обобщающую способность. Разделим все множество известных образов размерностью  $F$  на три подмножества:

1. Обучающее подмножество размерностью  $L = 0,8 F$ , которое используется непосредственно для обучения нейронной сети.

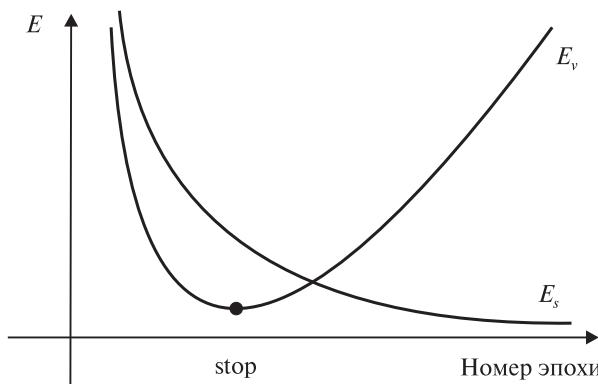
2. Валидационное (проверочное) подмножество размерностью  $V = 0,1F$ , применяемое для определения обобщающей способности на этапе обучения нейронной сети.

3. Тестовое подмножество размерностью  $T = 0,1F$ , используемое для оценки обобщающей способности после окончания процесса обучения.

Оценка обобщающей способности сети определяется как

$$E_v = \frac{1}{2} \sum_{k=1}^V \sum_j (y_j^k - e_j^k)^2. \quad (3.57)$$

В соответствии с методом раннего останова в процессе обучения производится оценка обобщающей способности сети на каждой эпохе (п. 4 алгоритма, разд. 3.6). При достижении минимального значения  $E_v$ , т. е. максимальной обобщающей способности, процесс обучения останавливается, хотя, как видно из рис. 3.16, суммарная квадратичная ошибка сети может продолжать уменьшаться.



*Rис. 3.16. Иллюстрация метода раннего останова*

Исходя из проведенных рассуждений можно сделать следующие выводы:

1. Персепtron с одним скрытым слоем и непрерывной, монотонно возрастающей, ограниченной функцией активации нейронных элементов является *универсальным аппроксиматором*, т. е. он позволяет осуществить любое отображение входных сигналов в выходные. При этом точность аппроксимации функции зависит от количества нейронов в скрытом слое. Чем больше количество нейронов в скрытом слое, тем выше точность аппроксимации функции. Однако при слишком большом количестве нейронов в скрытом слое ухудшается обобщающая способность сети.

2. Для качественного обучения сети необходимо, чтобы объем обучающей выборки  $L$  был больше количества настраиваемых параметров сети.

3. Мощность нейронной сети можно увеличивать как за счет количества нейронов в слое, так и за счет количества слоев. Если на нейронную сеть накладываются ограничения п. 1 и она не может решить поставленную задачу, то необходимо увеличивать количество скрытых слоев.

4. Случайная инициализация весовых коэффициентов нейронной сети должна проходить в достаточно узком диапазоне значений.

### 3.9. ГЕТЕРОГЕННЫЕ ПЕРСЕПТРОНЫ

В предыдущих разделах данной главы описывались сети, нейронные элементы которых имеют одинаковую функцию активации. Рассмотрим многослойный персептрон, различные слои которого могут иметь разные функции активации нейронных элементов. Такие сети называются *гетерогенными*. Простейшая гетерогенная сеть состоит из одного скрытого слоя с нелинейной функцией активации нейронных элементов и выходного линейного нейрона (рис. 3.17) и является универсальным аппроксиматором. Она часто используется для того, чтобы выйти за пределы диапазона значений, который ограничивается нелинейной функцией активации.

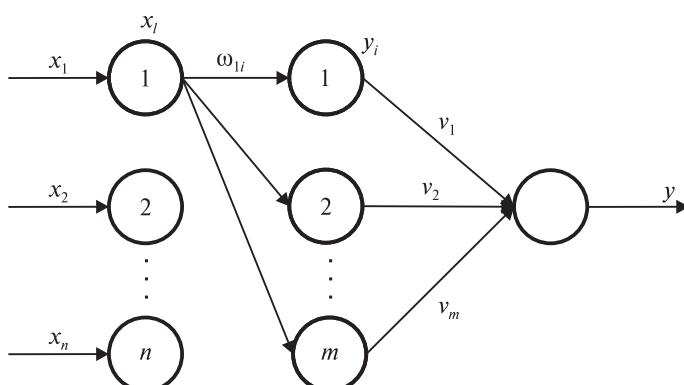


Рис. 3.17. Гетерогенная нейронная сеть

Выходное значение нейронной сети равно

$$y = \sum_i v_i y_i - T, \quad (3.58)$$

где  $v_i$  –  $i$ -й весовой коэффициент выходного нейрона.

Выходные значения нейронных элементов скрытого слоя определяются как

$$y_i = F(S_i) = F\left(\sum_l \omega_{li} x_l - T_i\right). \quad (3.59)$$

Рассмотрим обобщенное делта-правило для такой сети. Найдем ошибки нейронных элементов различных слоев нейронной сети. Так, для выходного слоя

$$\gamma = (y - e).$$

Ошибки нейронных элементов скрытого слоя вычисляются следующим образом:

$$\gamma_i = (y - e)v_i. \quad (3.60)$$

Обучающие правила для выходного слоя можно представить как

$$v_i(t+1) = v_i(t) - \alpha(y - e)y_i, \quad (3.61)$$

$$T(t+1) = T(t) + \alpha(y - e), \quad (3.62)$$

для скрытого слоя соответственно

$$\omega_{li}(t+1) = \omega_{li}(t) - \alpha\gamma_i F'(S_i)x_l, \quad (3.63)$$

$$T_i(t+1) = T_i(t) + \alpha\gamma_i F'(S_i), \quad (3.64)$$

где  $\gamma_i = (y - e)v_i$ .

Аналогично можно определить правила обучения для различного типа гетерогенных сетей. Использование стандартных алгоритмов обратного распространения ошибки для гетерогенных нейронных сетей приводит к нестабильности процесса обучения. Это происходит из-за применения в сетях различных функций активации нейронных элементов. Поэтому для обучения может использоваться алгоритм многократного распространения ошибки.

### 3.10. АЛГОРИТМ МНОГОКРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

Алгоритм обратного распространения ошибки предполагает для каждого тренировочного набора модификацию синаптических связей всех слоев нейронной сети. При этом изменение весовых коэффициентов одного слоя нейронной сети происходит без учета изменения весовых коэффициентов остальных слоев, что может привести к нестабильности процесса обучения, который характеризуется отсутствием тенденции к снижению среднеквадратичной ошибки сети. Особенно актуальной является эта проблема для гетерогенных нейронных сетей. Из-за различия функций активации нейронных элементов может возникнуть рассинхронизация процесса обучения между разными слоями сети, в результате чего процесс обучения будет характеризоваться нестабильностью. Для устранения таких явлений можно использовать алгоритм многократного распространения ошибки. Он предполагает на каждой итерации обучения модификацию синаптических связей только одного слоя нейронной сети. В соответствии с этим каждый образ будет последовательно подаваться на нейронную сеть столько раз, сколько настраиваемых слоев имеет сеть. Пусть  $p$  – число настраиваемых слоев нейронной сети. Тогда алгоритм многократного распространения ошибки состоит из следующих шагов:

1. Задается желаемая квадратичная ошибка нейронной сети  $E_p$ .
2. Происходит случайная инициализация синаптических связей сети.
3. В счетчик количества настраиваемых слоев записывается число  $p$ .
4. Подается первый тренировочный набор на вход нейронной сети.

Выполняется фаза прямого и обратного распространения сигнала. В результате осуществляется модификация весовых коэффициентов и порогов нейронных элементов только для  $p$ -го слоя нейронной сети:

$$\omega_{lj}(t+1) = \omega_{lj}(t) - \alpha(t) \gamma_i F'(S_i) x_l,$$

$$T_p(t+1) = T_p(t) + \alpha(t) \gamma_p F'_p(S_p),$$

где  $F_p(S_p)$  – функция активации нейронных элементов  $p$ -го слоя,  $i = p-1$ .

5. Устанавливается  $p = p-1$ .
6. Если  $p \neq 0$ , то перейти к шагу 4. В противном случае перейти к шагу 7 алгоритма.
7. Повторить процесс обучения, начиная с шага 3, для всех тренировочных наборов из обучающей выборки.

8. Вычислить суммарную квадратичную ошибку нейронной сети:

$$E_s = \frac{1}{2} \sum_{k=1}^L \sum_j (y_j^k - e_j^k)^2,$$

где  $L$  – общее количество тренировочных наборов.

9. Если  $E_s > E_e$ , то перейти к шагу 3. В противном случае закончить процесс обучения.

### 3.11. ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ВХОДНЫХ ДАННЫХ

Для эффективного обучения нейронной сети необходимо использовать методы предварительной обработки входных данных. Рассмотрим распространенные методы, которые могут применяться как отдельно, так и совместно.

#### 1. Преобразование входных данных к приемлемому диапазону значений

Входные данные нейронной сети необходимо преобразовать в диапазон значений в соответствии с используемой функцией активации нейронных элементов.

Пусть входные данные находятся в диапазоне  $[x_{\min}, x_{\max}]$ . Нужно преобразовать их в диапазон  $[a, b]$ , т. е.

$$x \in [x_{\min}, x_{\max}] \rightarrow [a, b].$$

В этом случае каждая компонента входного образа преобразуется следующим образом:

$$\overline{x_i^k} = \frac{(x_i^k - x_{\min})(b - a)}{(x_{\max} - x_{\min})} + a. \quad (3.65)$$

Так, например, если  $[x_{\min}, x_{\max}] \rightarrow [0, 1]$  в соответствии с сигмоидной функцией активации, то выражение (3.65) будет иметь вид

$$\overline{x_i^k} = \frac{x_i^k - x_{\min}}{x_{\max} - x_{\min}}.$$

#### 2. Преобразование разнородных входных данных

Разнородные входные данные описывают различные свойства одного и того же процесса, например давление, температуру и т. д. Для уменьшения степени разнородности данных можно использовать следующее преобразование:

$$\overline{x_i^k} = \frac{x_i^k - \mu(x_i)}{\sigma(x_i)}, \quad (3.66)$$

где  $x_i^k$  –  $i$ -я компонента  $k$ -го образа;  $\mu(x_i)$  – математическое ожидание  $i$ -й компоненты;  $\sigma(x_i)$  – среднеквадратичное отклонение.

Математическое ожидание и среднеквадратичное отклонение находятся в соответствии со следующими выражениями:

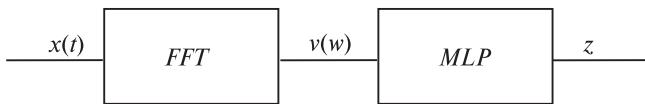
$$\mu(x_i) = \frac{1}{L} \sum_{k=1}^L x_i^k,$$

$$\sigma(x_i) = \sqrt{\frac{1}{L} \sum_{k=1}^L (x_i^k - \mu(x_i))^2}.$$

### *3. Преобразование входных данных из временного пространства в другие области*

Часто для лучшего обучения нейронной сети необходимо перейти из временной области в другие пространства. Существуют следующие методы такого преобразования входных данных:

- быстрое преобразование Фурье (fast Fourier transform, FFT). В этом случае входные данные из временной области преобразуются в частотную область и подаются на многослойный персептрон (MLP), как показано на рис. 3.18.



*Рис. 3.18. Использование преобразования Фурье*

Аналогичным образом могут использоваться другие преобразования, приведенные ниже:

- дискретное косинусное преобразование (ДКП);
- вейвлет-преобразование;
- метод главных компонент (PCA).

## **3.12. ПРИМЕНЕНИЕ МНОГОСЛОЙНЫХ ПЕРСЕПТРОНОВ**

Рассмотрим примеры использования персепtronов для решения задач классификации образов, прогнозирования и управления.

### 3.12.1. Классификация образов

Стандартная процедура распознавания образов в общем случае состоит из двух основных этапов: выделения основных признаков объекта распознавания (feature extraction) и построения классификатора (рис. 3.19). В качестве классификатора может использоваться персептрон с одним скрытым слоем.

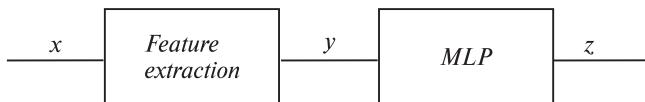


Рис. 3.19. Стандартная схема распознавания образов

Пусть  $p$  – количество классов, на которые необходимо разбить входное пространство образов. Тогда персептрон будет состоять из  $n$  входных нейронных элементов, где  $n$  – размерность входного образа,  $m$  – нейронных элементов скрытого слоя и  $p$  – нейронных элементов выходного слоя. Как уже отмечалось, количество областей, на которые разбивает скрытый слой входное пространство образов, должно быть больше количества классов:

$$m > \log_2 p.$$

С точки зрения классификации образов существует детерминированная и вероятностная интерпретация выходных значений последнего слоя персептрона. При детерминированном подходе на выходе нейронной сети присутствует одно единичное и остальные нулевые значения. Номер нейронного элемента с единичным выходным значением определяет принадлежность входного образа к соответствующему классу. Для этого можно использовать следующую кодировку выходных сигналов нейронной сети. Определяется номер  $k$  выходного нейронного элемента, который имеет максимальное выходное значение:

$$y_k = \arg \max y_j.$$

Выходному значению нейрона с номером  $k$  присваивается единичное, а выходные значения остальных нейронных элементов равняются нулю:

$$y_j = \begin{cases} 1, & j = k, \\ 0, & \text{иначе.} \end{cases} \quad (3.67)$$

Другой способ детерминированного кодирования выходных значений состоит в сравнении выходного значения с пороговым значением в соответствии с используемой функцией активации нейронных элементов. Так, при использовании сигмоидной функции активации имеем

$$y_j = \begin{cases} 1, & y_j > 0,5, \\ 0, & \text{иначе.} \end{cases} \quad (3.68)$$

При вероятностной кодировке выходные значения нейронной сети определяют вероятности принадлежности входного образа к соответствующему классу. В этом случае при применении, например, сигмоидной функции активации выходное значение  $j$ -го нейрона выходного слоя определяется следующим образом:

$$y_j = \frac{y_j}{\sum_j y_j}. \quad (3.69)$$

При использовании в последнем слое функции активации softmax выходные значения нейронов последнего слоя также определяют вероятность принадлежности входного образа к соответствующему классу:

$$y_j = \text{softmax}(S_j) = \frac{e^{S_j}}{\sum_j e^{S_j}}.$$

### 3.12.2. Экспертные системы

Данные системы характеризуются тем, что знания в них формируются экспертом-специалистом в конкретной предметной области. Продукционная модель экспертной системы работает по принципу «если (условие), то (действие)». Для построения экспертных систем можно использовать многослойный персептрон, который функционирует по аналогичному принципу «если вход  $X$ , то выход  $Y$ ». Основная задача проектирования конвенциальной экспертной системы состоит в формализации знаний эксперта. В случае применения многослойного персептрона основная проблема заключается в подготовке (с помощью квалифицированного эксперта) обучающей выборки. После этого проводится обучение персептрона, в результате которого осуществляется процесс обобщения знаний эксперта. Одним из основных преимуществ экспертных систем, построенных на основе нейронных сетей, является возможность их применения для решения различного рода прикладных задач в трудноформализуемых (с точки зрения математики) предметных областях.

### 3.12.3. Прогнозирование

Рассмотрим задачу прогнозирования временных рядов. Задача предсказания временного ряда может быть описана следующим образом: для заданной последовательности  $x(1), x(2), \dots, x(t)$  необходимо найти ее продолжение  $x(t+1), x(t+2) \dots$

Для прогнозирования временных рядов применяется метод скользящего окна. Он характеризуется длиной окна  $k$ , которая перемещается по ряду с единичным шагом и определяет количество нейронов распределительного слоя сети. Оценка временного ряда в момент времени  $t$  находится как нелинейное преобразование от предыдущих  $k$  членов ряда:

$$x(t) = F(x(t-1), x(t-2), \dots, x(t-k)),$$

где  $t = \overline{k+1, \dots}$ ;  $F$  – нелинейная функция.

Прогнозирующая нейронная сеть представляет собой многослойный персептрон, который состоит из  $k$  нейронов распределительного слоя, скрытых слоев и одного выходного нейрона. Такая сеть называется также TDNN (time delay neural network). Для обучения нейронной сети прогнозированию используется выборка известных членов ряда. После обучения сеть должна прогнозировать временной ряд на упра-дающий промежуток времени. Рассмотрим более детально применение многослойного персептрона для прогнозирования хаотических временных рядов [36]. В качестве хаотических систем будем исследовать аттрактор Лоренца, который описывается системой трех дифференциальных уравнений

$$\begin{cases} \frac{dx}{dt} = G(y - x), \\ \frac{dy}{dt} = -xz + rx - y, \\ \frac{dz}{dt} = xy - bz. \end{cases} \quad (3.70)$$

Процесс Лоренца является хаотической системой для значений параметров  $G = 10$ ,  $r = 28$  и  $b = 8/3$ . Последовательность значений координаты  $x$ , полученная путем численного решения системы (3.70) с использованием метода Рунге – Кутты четвертого порядка с шагом 0,01, изображена на рис. 3.20.

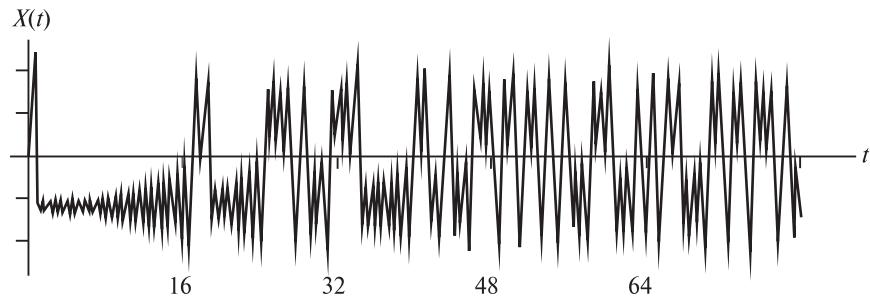


Рис. 3.20. Оригинальный процесс Лоренца (координата  $X$ )

Для прогнозирования ряда Лоренца по координате  $X$  возьмем многослойный персептрон с семью входными элементами, пятью скрытыми нейронами с сигмоидной функцией активации и одним линейным выходным нейроном. Размерность тренировочного множества равна 800 образом.

На рис. 3.21 показаны результаты предсказания для временного ряда Лоренца на 30 шагов. Как видно из рисунка, сеть позволяет достаточно точно предсказать ряд Лоренца на 12 отсчетов вперед, а дальше ошибка предсказания увеличивается. В этом проявляются основные свойства хаотических систем.

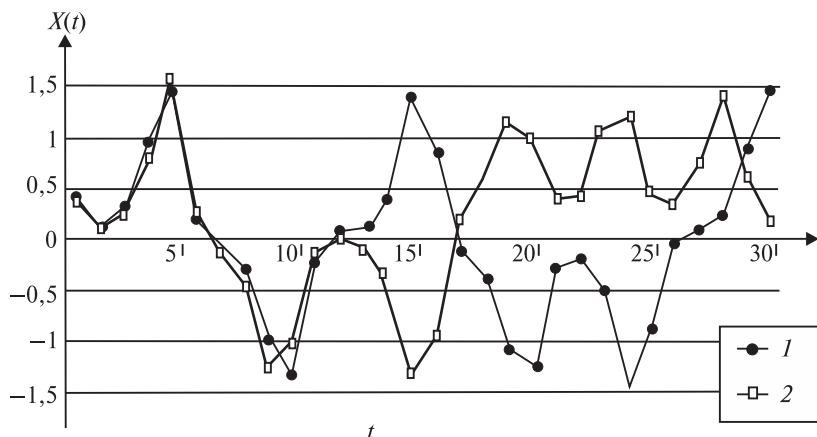


Рис. 3.21. Прогноз процесса Лоренца для 30 итераций прогноза:  
1 – прогноз; 2 – оригинальная последовательность

Рассмотрим теперь прогнозирование состояния динамических систем. Пусть состояние динамической системы в каждый момент времени задается  $n$ -мерным вектором

$$X(t) = (X_1(t), X_2(t), \dots, X_n(t)).$$

Предположим, что известны временные зависимости  $X_i(t)$  соответствующих переменных на определенном интервале времени. Стандартный математический подход состоит в аппроксимации динамической системы нелинейными дифференциальными уравнениями. Нейросетевой подход заключается в применении многослойного персептрона, который на основе предыдущего состояния динамической системы прогнозирует следующее, имеющее  $n$  входных,  $m$  скрытых и  $n$  выходных нейронных элементов (рис. 3.22).

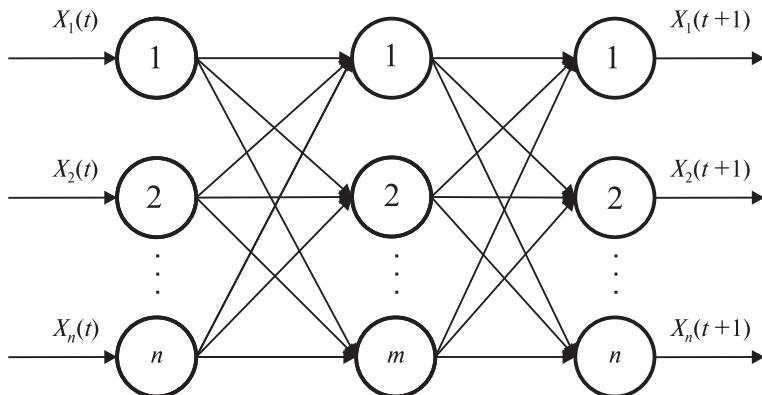


Рис. 3.22. Прогнозирование состояния динамической системы

Выходные значения нейронной сети определяются как

$$X(t+1) = F(X(t)). \quad (3.71)$$

После обучения такой сети, используя только наблюдаемые реализации, можно установить как состояние динамической системы в произвольный момент времени, так и эволюцию точек фазовой траектории.

### 3.12.4. Автономное управление автомобилем

Нейронные сети могут эффективно применяться в управлении автомобилем. В этом случае достаточно системе управления задать координаты конечной точки движения, и автомобиль без участия человека

будет двигаться к указанной цели. Такая система разработана в университете Карнеги – Меллона в рамках проекта ALVINN (autonomous land vehicle in a neural networks) [37]. В ней предполагается, что автомобиль оборудован видеокамерой, которая отображает дорогу с разметкой. Центральным элементом системы является многослойный персептрон с одним скрытым слоем (рис. 3.23). Входной слой содержит 30×32 нейронных элемента, на которые подается преобразованное от видеокамеры изображение пути. Скрытый слой состоит из пяти, а выходной – из 30 нейронных элементов. В качестве функции активации используется сигмоидная функция. Активность выходных нейронов характеризует повороты руля. Так, если максимальной активностью обладает центральный нейрон, то это означает движение прямо. Когда наибольшую

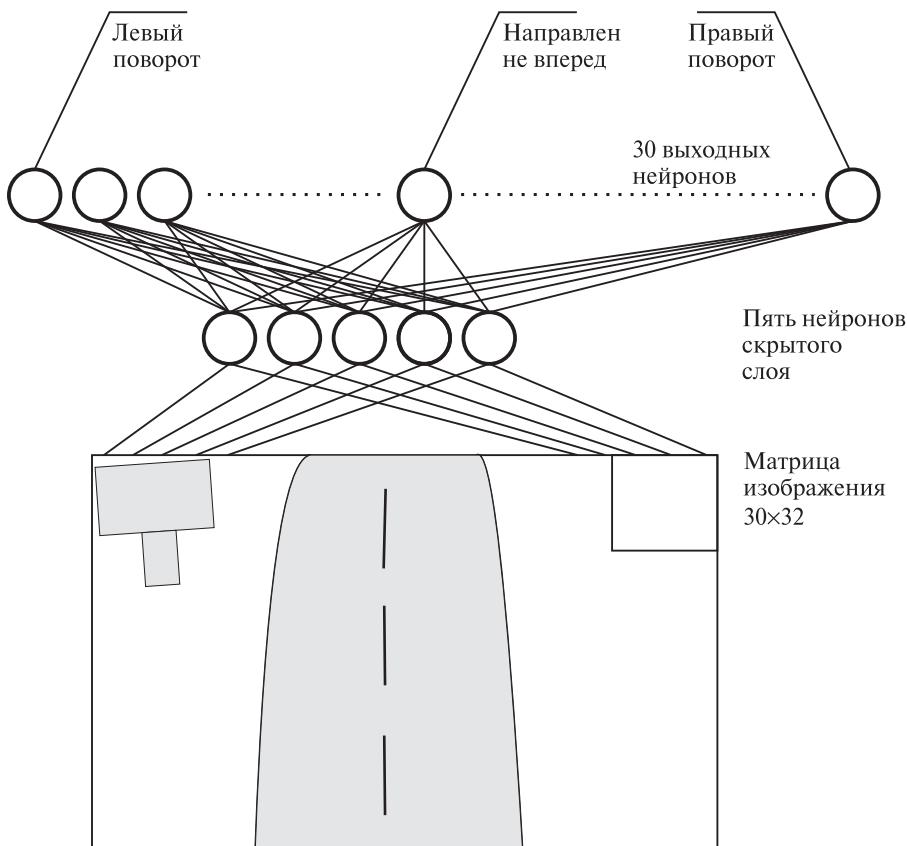


Рис. 3.23. Отображение изображения дороги на нейронную сеть

активность имеет крайний левый нейрон, то это соответствует повороту налево на определенное число градусов. Обучающая выборка генерируется при управлении автомобилем оператором.

Для подготовки обучающей выборки оператор управлял автомобилем при движении со скоростью 9,5 км/ч, моделируя различные ситуации. Изображение от видеокамеры использовалось как вход, а текущее направление руля — как желаемый выход. В целях упрощения процесса получения обучающей выборки применялось программное вращение изображения от видеокамеры (рис. 3.24). При этом соответствующим образом изменялась эталонная реакция нейронной сети.

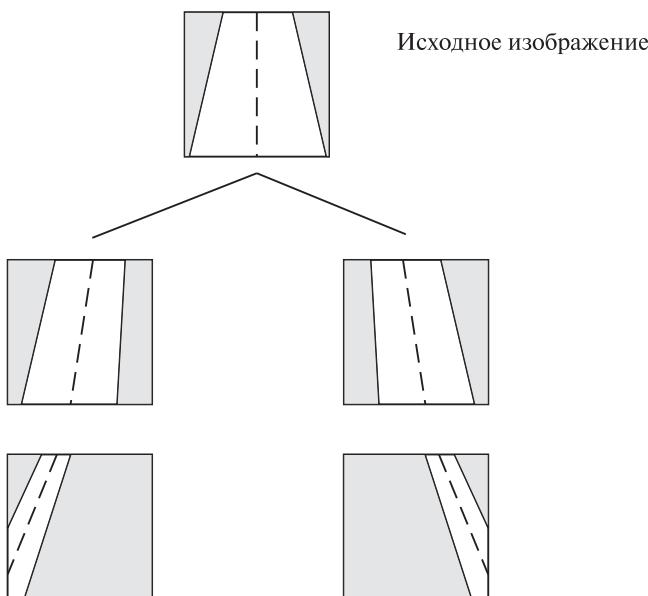


Рис. 3.24. Вращение исходного изображения от видеокамеры

В результате была создана обучающая выборка, объем которой составил 1200 тренировочных наборов. Обучение нейронной сети проводилось с использованием трех станций «Sun-4». Время обучения методом обратного распространения ошибки составило пять минут. После обучения, как показали эксперименты, нейронная сеть смогла автономно управлять автомобилем. В результате в рамках этого проекта была достигнута скорость движения автомобиля до 70 миль/час. При этом автомобиль проехал 90 миль к северу от Питтсбурга, что свидетельствует о большом потенциале нейронных сетей при решении задач управления.

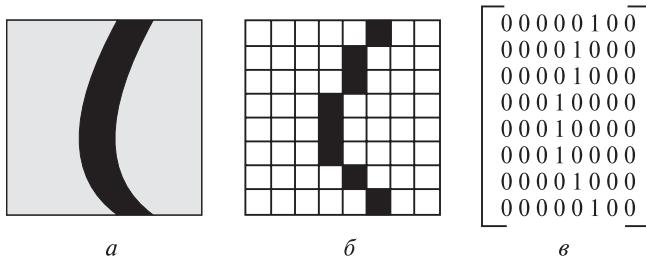
### 3.12.5. Автономное управление роботом

Одной из задач, возникающих при управлении мобильным роботом, является его движение по заранее заданной траектории. При этом траектория может задаваться в виде разметки на дороге или в цеху. Задача робота состоит в том, чтобы, двигаясь по известной траектории, он достиг заданной конечной точки. Такой подход позволяет создавать относительно дешевые автономные системы для перевозки грузов на предприятии. В этом случае к ведущему роботу, который отслеживает траекторию движения, могут подсоединяться ведомые роботы, образующие автопоезд.

Общая структура системы автономного управления мобильным роботом приведена на рис. 3.25. Мобильный робот оборудован видеокамерой, которая предназначена для отображения заданной траектории. Блок обработки видеоизображений при помощи пороговой обработки преобразует изображение от видеокамеры в бинарную матрицу, как показано на рис. 3.26.



*Рис. 3.25. Структура системы автономного управления мобильным роботом*



*Рис. 3.26. Этапы преобразования изображения траектории от видеокамеры:*

*а* – исходное видеоизображение траектории;

*б* – проекция изображения на матрицу размерностью  $8 \times 8$ ;

*в* – бинарный массив

Размерность матрицы может быть различной. Для моделирования такой системы применялась бинарная матрица размером  $8 \times 8$ . Блок определения направления движения представляет собой многослойный персептрон. В каждый момент времени он формирует направление движения робота, которое подается в органы управления. Задача системы состоит в обеспечении устойчивого управления роботом при движении по различным траекториям. При этом робот должен корректно проходить участки траектории, с формой которых он не был знаком на этапе обучения. Для формирования направления движения робота используется персептрон с одним скрытым слоем (рис. 3.27). Входной слой, на который подается бинарная матрица изображения траектории, состоит из 64 нейронных элементов, скрытый слой состоит из 15, а выходной слой – из 19 нейронных элементов.

Каждому входному нейрону сети поставлено в соответствие определенное направление движения из диапазона от  $-45^\circ$  до  $45^\circ$ . Шаг дисcretизации составляет при этом  $5^\circ$ . В каждый момент времени на выходе нейронной сети активным является только один нейронный элемент, который определяет текущее направление движения робота. В качестве функции нелинейного преобразования используется сигмоидная функция активации.

Для обучения нейронной сети следует сформировать обучающую выборку. Для этого необходимо генерировать тренировочные наборы, которые отражают наиболее типичные участки траектории. Выходной вектор должен соответствовать при этом требуемому углу поворота робота (рис. 3.28).

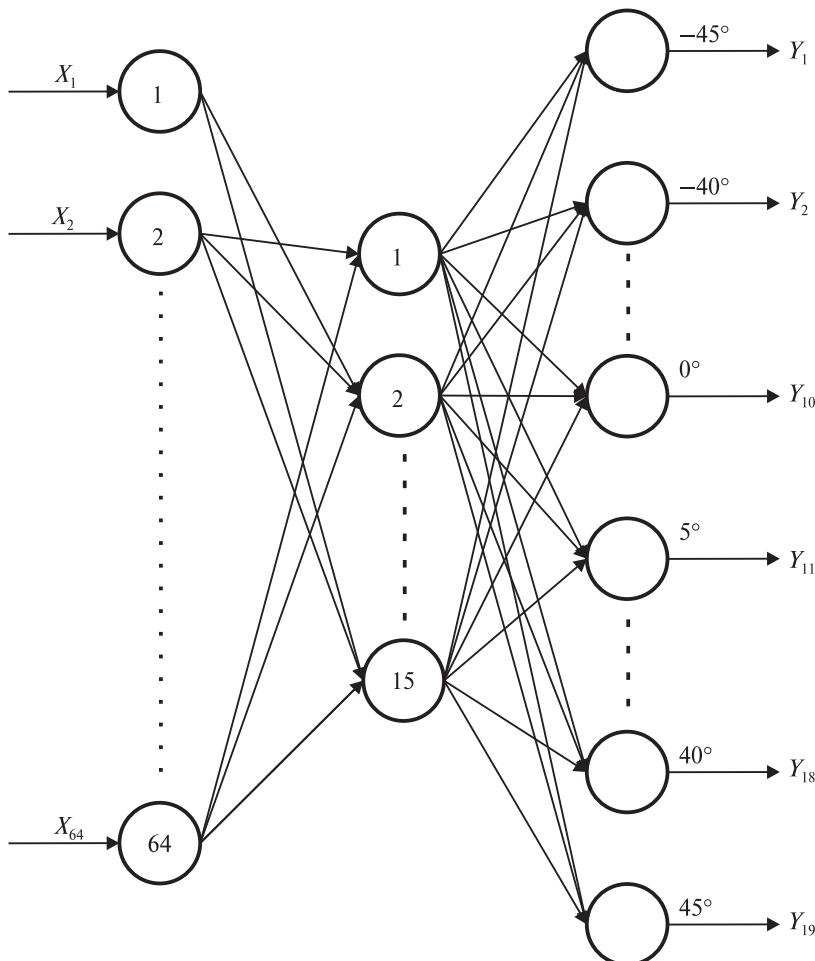
Для представленного на рис. 3.28 фрагмента траектории обучающий паттерн приведен на рис. 3.29.

В целях обучения нейронной сети были разработаны обучающие наборы, которые можно условно разделить на два класса. Первый класс представляет собой фрагменты прямых линий, наблюдаемые под различными углами, а также рекомендуемые углы поворота для каждого фрагмента. Примеры тренировочных наборов для данного класса приведены на рис. 3.30.

Второй класс тренировочных наборов представляет собой фрагменты кривых линий, которые имеют различную степень кривизны. Положение робота относительно заданной траектории изображено на рис. 3.31, *а*, обучающие паттерны – на рис. 3.31, *б*.

Для обучения нейронной сети применялся алгоритм обратного распространения ошибки. В целях проведения экспериментов разработано программное обеспечение для эмуляции нейронной сети и моделиро-

вания движения робота. При этом на экране компьютера может задаваться произвольная траектория движения робота. Результаты компьютерного моделирования подтвердили устойчивое движение робота по произвольным траекториям с различными углами поворота (рис. 3.32).



*Рис. 3.27. Архитектура нейронной сети:*  
 $X_1 \dots X_{64}$  – элементы бинарного массива;  
 $Y_1 \dots Y_{19}$  – результирующий вектор направления движения

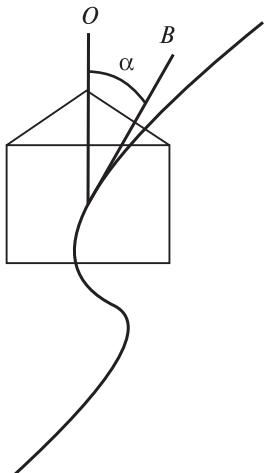


Рис. 3.28. Определение желаемого направления робота:  
 $O$  – текущее направление;  
 $B$  – желаемое направление;  
 $\alpha = 20^\circ$  – требуемый угол поворота

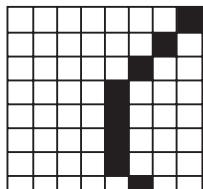
*a*

Рис. 3.29. Формирование обучающего паттерна:  
 $a$  – входной паттерн;  $b$  – выходной паттерн

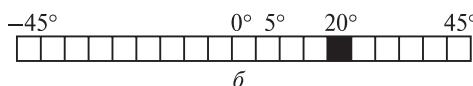
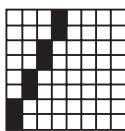
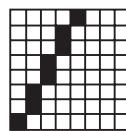
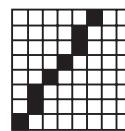
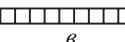
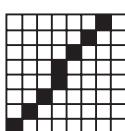
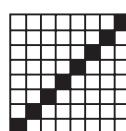
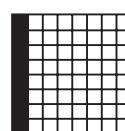
*b* $\alpha = 5^\circ$  $\alpha = 15^\circ$  $\alpha = 25^\circ$ *a**b* $\alpha = -25^\circ$  $\alpha = 35^\circ$  $\alpha = 40^\circ$  $\alpha = -25^\circ$ *c**d**e*

Рис. 3.30. Пример формирования шести тренировочных наборов (*a*, *b*, *c*, *d*, *e*) для фрагментов прямой линии

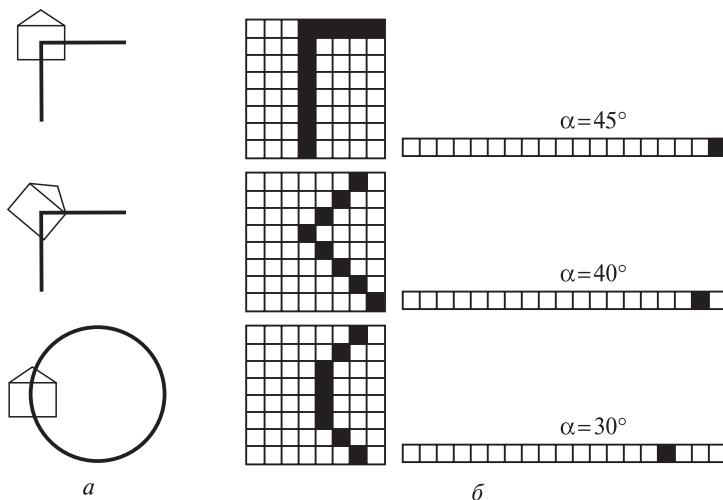


Рис. 3.31. Формирование тренировочных наборов для кривых линий:  
 а – положение робота относительно траектории;  
 б – формирование обучающей выборки

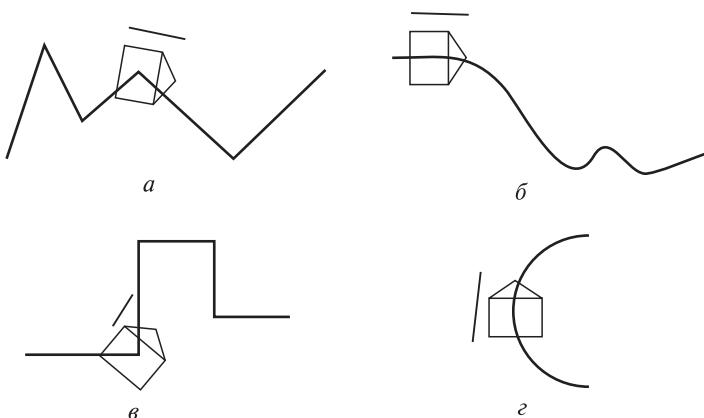


Рис. 3.32. Примеры движения робота по различным траекториям:  
 а – движение робота по ломаной кривой; б – движение робота  
 по извилистой кривой; в – движение робота по прямоугольной линии;  
 г – движение робота по окружности

Это происходит за счет обобщения знаний на неизвестные типы трасс, которые не входили в обучающую выборку нейронной сети. При любых отклонениях робота в пределах видимости видеокамеры от заданной траектории он всегда возвращается на требуемый путь.

## Глава 4

# РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

В данной главе рассматриваются *рекуррентные нейронные сети* (recurrent neural networks), которые являются дальнейшим развитием многослойных персепtronов, описываются их архитектура, обучение и применение [27, 38–42]. Они характеризуются как прямым (feedforward) так и обратным (feedback) распространением информации, а также обучением с учителем и обратными связями, по которым передаются результаты обработки сетью данных на предыдущем этапе. В итоге в каждый фиксированный момент времени входом рекуррентной нейронной сети являются вектор входных данных и результаты обработки информации сетью на предыдущем этапе. Обучение таких сетей базируется на алгоритме обратного распространения ошибки. Рекуррентные сети могут использоваться для обработки динамических данных, временных образов, решения задач прогнозирования, идентификации систем, распознавания речи, обработки естественного языка и управления.

### 4.1. ОБЩАЯ АРХИТЕКТУРА РЕКУРРЕНТНОЙ НЕЙРОННОЙ СЕТИ

*Рекуррентными* нейронными сетями называются такие сети, в которых выходы нейронных элементов последующих слоев имеют синаптические соединения с нейронами предшествующих слоев. Это приводит к возможности учета результатов преобразования нейронной сетью информации на предыдущем этапе для обработки входного вектора на следующем этапе функционирования сети. На рис. 4.1 изображена общая архитектура рекуррентной нейронной сети, которая имеет обратные связи как от нейронов выходного, так и от нейронов скрытого слоя [38]. Такая сеть называется *мультирекуррентной*.

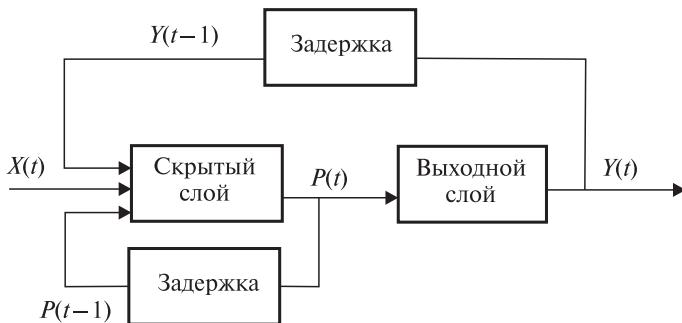


Рис. 4.1. Мультирекуррентная нейронная сеть

Рекуррентная нейронная сеть является динамической системой. Поэтому здесь необходимо использовать параметр времени. Как видно из рис. 4.1, выходные значения рекуррентной нейронной сети зависят от входного образа  $X(t)$ , значений нейронов скрытого  $P(t-1)$  и выходного  $Y(t-1)$  слоев:

$$y(t) = f(x(t), y(t-1), p(t-1)). \quad (4.1)$$

Выходные значения нейронов скрытого и выходного слоев определяются следующим образом:

$$P(t) = F(W_1 X(t) + W_2 P(t-1) + W_4 Y(t-1)),$$

$$Y(t) = F(W_3 P(t)),$$

где  $W_1, W_2, W_3$  – матрицы весовых коэффициентов скрытого слоя для входного образа  $X(t)$ ; для нейронов обратной связи скрытого  $P(t-1)$  и выходного слоев  $Y(t-1)$  соответственно;  $W_4$  – матрица весовых коэффициентов выходного слоя нейронной сети.

Теорему универсальной аппроксимации можно обобщить на рекуррентную нейронную сеть.

**Теорема 4.1.** Любая нелинейная динамическая система при использовании достаточного количества сигмоидальных нейронных элементов в скрытом слое с любой точностью может быть аппроксимирована рекуррентной нейронной сетью.

Здесь под сигмоидальными нейронами понимаются нейроны с сигмоподобной функцией активации (гиперболический тангенс, биполярная сигмоидная и сигмоидная). Рекуррентные нейронные сети позво-

ляют моделировать конечные автоматы, например машину Тьюринга. В зависимости от организации обратных связей существуют различные варианты архитектур рекуррентных нейронных сетей [38–40].

## 4.2. РЕКУРРЕНТНАЯ СЕТЬ ДЖОРДАНА

В 1986 г. М. Джордан (M. Jordan) предложил рекуррентную сеть (рис. 4.2), в которой выходы нейронных элементов последнего слоя посредством специальных входных нейронов соединены с нейронами промежуточного слоя [38, 39].

Входные нейроны, которые используются для организации обратных связей, называются *контекстными* (context units). Они распределяют выходные данные нейронной сети на нейронные элементы промежуточного слоя (рис. 4.2).

Количество контекстных нейронов равно числу выходных нейронных элементов рекуррентной сети. В качестве выходного слоя таких сетей можно использовать нейроны с линейной функцией активации. Выходное значение  $j$ -го нейронного элемента последнего слоя в этом случае вычисляется как

$$y_j(t) = \sum_{i=1}^m v_{ij} p_i(t) - T_j, \quad (4.2)$$

где  $v_{ij}$  — весовой коэффициент между  $i$ -м нейроном промежуточного и  $j$ -м нейроном выходного слоя;  $p_i(t)$  — выходное значение  $i$ -го нейрона промежуточного слоя;  $T_j$  — пороговое значение  $j$ -го нейрона выходного слоя.

Взвешенная сумма  $i$ -го нейрона промежуточного слоя определяется следующим выражением:

$$S_i(t) = \sum_{k=1}^n w_{ki} x_k(t) + \sum_{j=1}^p w_{ji} y_j(t-1) - T_i, \quad (4.3)$$

где  $w_{ki}$  — весовой коэффициент между  $k$ -м нейроном входного и  $i$ -м нейроном скрытого слоя;  $T_i$  — пороговое значение  $i$ -го нейрона скрытого слоя,  $n$  — размерность входного вектора;  $p$  — количество нейронов выходного слоя;  $w_{ji}$  — весовой коэффициент между  $j$ -м контекстным нейроном и  $i$ -м нейроном скрытого слоя.

Тогда выходное значение  $i$ -го нейрона скрытого слоя равно

$$p_i(t) = F(S_i(t)). \quad (4.4)$$

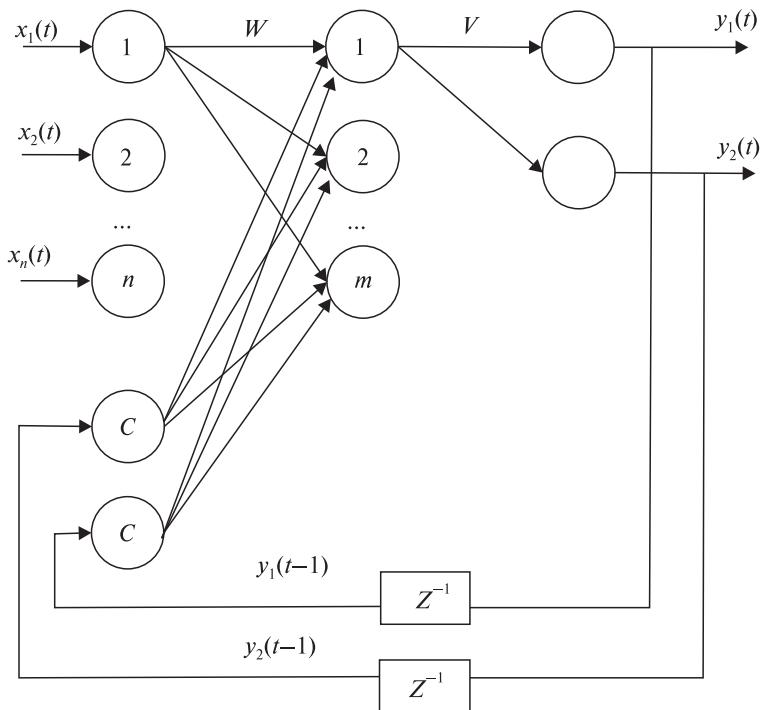


Рис. 4.2. Рекуррентная сеть Джордана  
( $Z^{-1}$  – элемент задержки;  $C$  – контекстный нейрон)

В качестве функции нелинейного преобразования  $F$  обычно используется гиперболический тангенс или сигмоидная функция. Сеть Джордана можно рассматривать как нелинейную модель авторегрессии со скользящим средним (NARMA).

### 4.3. РЕКУРРЕНТНАЯ СЕТЬ ЭЛМАНА

Другой вариант рекуррентной нейронной сети предложил в 1990 г. Дж. Элман (J. Elman) [40]. Выходы нейронных элементов промежуточного слоя такой сети соединяются с контекстными нейронами входного слоя (рис. 4.3).

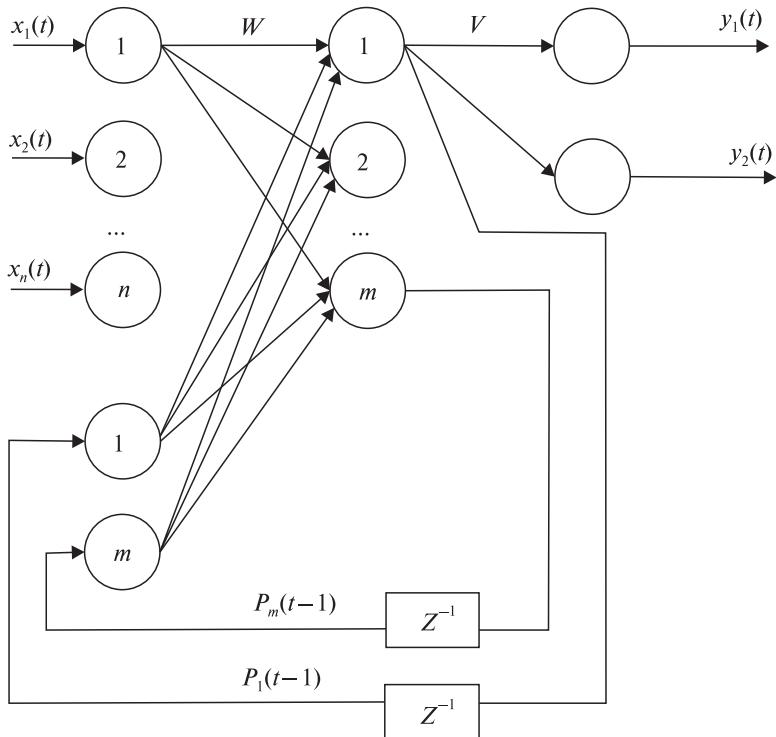


Рис. 4.3. Рекуррентная сеть Элмана

Взвешенная сумма  $i$ -го нейрона промежуточного слоя находится по следующей формуле:

$$S_i(t) = \sum_{k=1}^n w_{ki} x_i(t) + \sum_{l=1}^m w_{li} p_l(t-1) - T_i, \quad (4.5)$$

где  $m$  – количество нейронов скрытого слоя;  $p_l(t-1)$  – выходное значение  $l$ -го нейрона скрытого слоя.

Выходное значение  $l$ -го нейрона скрытого слоя определяется выражением

$$p_l(t-1) = F(S_l(t-1)). \quad (4.6)$$

Для построения рекуррентных нейронных сетей можно использовать оба приведенных выше подхода. В этом случае получается мульти-рекуррентная нейронная сеть, в которой имеются обратные связи к кон-

текстным нейронам от нейронных элементов как выходного, так и скрытого слоя. Количество контекстных нейронов входного слоя равняется общему числу нейронов скрытого и выходного слоев. Взвешенная сумма нейронных элементов скрытого слоя в этом случае вычисляется по формуле

$$S_i(t) = \sum_{k=1}^n w_{ki}x_k(t) + \sum_{j=1}^p w_{ji}y_j(t-1) + \sum_{l=1}^m w_{li}p_l(t-1) - T_i, \quad (4.7)$$

где  $p$  – количество нейронов выходного слоя.

#### 4.4. ОБУЧЕНИЕ РЕКУРРЕНТНОЙ СЕТИ

Для обучения рекуррентных нейронных сетей применяется алгоритм обратного распространения ошибки. Рассмотрим его применение для мультирекуррентной нейронной сети (рис. 4.4), которая имеет один нейронный элемент с линейной функцией активации в последнем слое. Такая сеть может использоваться для прогнозирования временных рядов.

Вычислим квадратичную ошибку для одного входного образа:

$$E(t) = \frac{1}{2}(y(t) - e)^2,$$

где  $e$  – эталонное значение нейронной сети для соответствующего входного образа.

Выходное значение нейронной сети определяется как

$$y(t) = \sum_{i=1}^m v_i p_i(t) - T,$$

где  $v_i$  – синаптическая связь от  $i$ -го нейрона скрытого слоя к выходному нейрону.

Найдем выходное значение и взвешенную сумму  $i$ -го нейрона скрытого слоя по следующим формулам соответственно:

$$p_i(t) = F(S_i(t)),$$

$$S_i(t) = \sum_{k=1}^n w_{ki}x_k(t) + \sum_{l=1}^m w_{li}p_l(t-1) + w_{li}y(t-1) - T_i,$$

где  $w_{li}$  – синаптическая связь от  $l$ -го контекстного нейрона к  $i$ -му нейрону скрытого слоя;  $w_{li}$  – весовой коэффициент между выходным и  $i$ -м нейроном скрытого слоя.

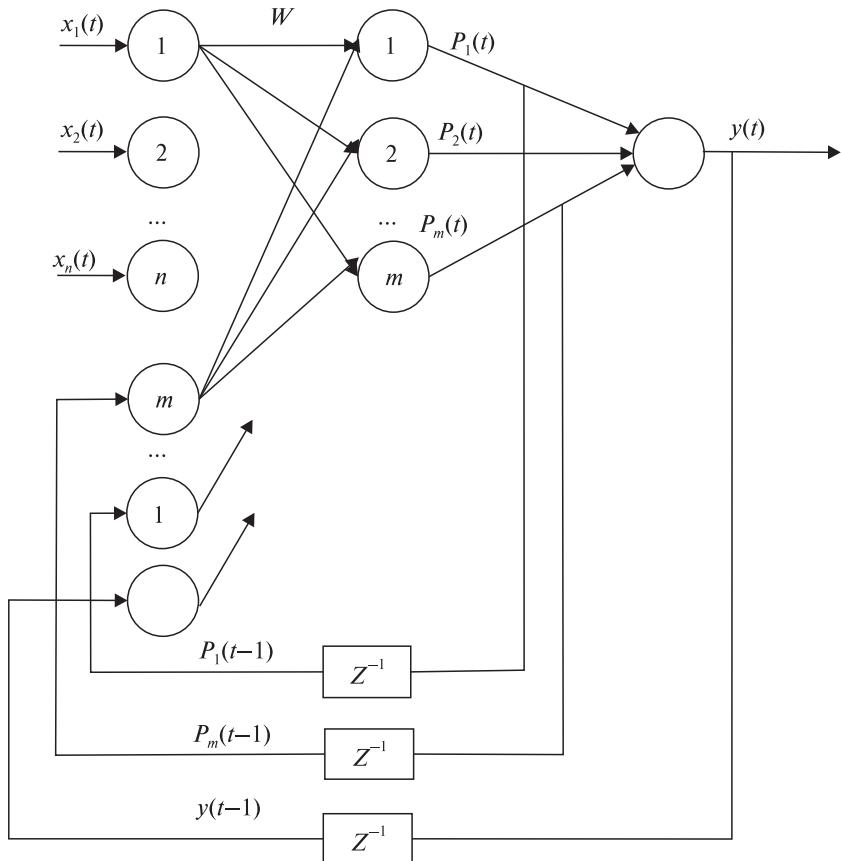


Рис. 4.4. Мультирекуррентная сеть

Для определения обобщенного дельта-правила найдем производные функции квадратичной ошибки по настраиваемым параметрам сети. Для выходного слоя это

$$\frac{\partial E}{\partial v_i} = \frac{\partial E}{\partial y(t)} \frac{\partial y(t)}{\partial v_i} = (y(t) - e) p_i(t),$$

$$\frac{\partial E}{\partial T} = \frac{\partial E}{\partial y(t)} \frac{\partial y(t)}{\partial T} = -(y(t) - e).$$

Для нейронных элементов скрытого слоя

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial y(t)} \frac{\partial y(t)}{\partial p_i(t)} \frac{\partial p_i(t)}{\partial S_i(t)} \frac{\partial S_i(t)}{\partial w_{ki}} = (y(t) - e)v_i F'(S_i(t))x_k(t),$$

$$\frac{\partial E}{\partial w_{li}} = \frac{\partial E}{\partial y(t)} \frac{\partial y(t)}{\partial p_i(t)} \frac{\partial p_i(t)}{\partial S_i(t)} \frac{\partial S_i(t)}{\partial w_{li}} = (y(t) - e)v_i F'(S_i(t))p_i(t-1),$$

$$\frac{\partial E}{\partial w_{li}} = \frac{\partial E}{\partial y(t)} \frac{\partial y(t)}{\partial p_i(t)} \frac{\partial p_i(t)}{\partial S_i(t)} \frac{\partial S_i(t)}{\partial w_{li}} = (y(t) - e)v_i F'(S_i(t))y(t-1),$$

$$\frac{\partial E}{\partial T_i} = \frac{\partial E}{\partial y(t)} \frac{\partial y(t)}{\partial p_i(t)} \frac{\partial p_i(t)}{\partial S_i(t)} \frac{\partial S_i(t)}{\partial T_i} = -(y(t) - e)v_i F'(S_i(t)),$$

где  $F'(S_i(t))$  – производная функции активации скрытого слоя.

Отсюда получим выражения для настройки параметров обучения рекуррентной сети:

$$v_i(t+1) = v_i(t) - \alpha(y(t) - e)p_i(t),$$

$$T_i(t+1) = T_i(t) + \alpha(y(t) - e),$$

$$w_{ki}(t+1) = w_{ki}(t) - \alpha\gamma_i F'(S_i(t))x_k(t),$$

$$w_{li}(t+1) = w_{li}(t) - \alpha\gamma_i F'(S_i(t))p_i(t-1),$$

$$w_{li}(t+1) = w_{li}(t) - \alpha\gamma_i F'(S_i(t))y(t-1),$$

$$T_i(t+1) = T_i(t) + \alpha\gamma_i F'(S_i(t)).$$

Здесь  $\gamma_i = (y(t) - e)v_i$  – ошибка  $i$ -го нейрона скрытого слоя.

В качестве функции активации нейронных элементов может использоваться функция гиперболического тангенса или сигмоидная.

Для сигмоидной функции

$$F'(S_i(t)) = \frac{\partial p_i(t)}{\partial S_i(t)} = p_i(t)(1 - p_i(t)),$$

для функции гиперболического тангенса

$$F'(S_i(t)) = (1 - p_i^2(t)).$$

Алгоритм обучения рекуррентной нейронной сети в общем случае состоит из следующих шагов:

1. В начальный момент времени  $t=1$  все контекстные нейроны устанавливаются в нулевое состояние, т. е. их выходные значения равняются нулю.

2. Входной образ подается на сеть, и происходит прямое распространение его в нейронной сети.

3. В соответствии с алгоритмом обратного распространения ошибки производится модификация весовых коэффициентов и пороговых значений нейронных элементов.

4. Устанавливается  $t = t + 1$ , и осуществляется переход к п. 2.

Обучение рекуррентной сети производится до тех пор, пока суммарная квадратичная ошибка сети не станет меньше заданной.

#### 4.5. ПРИМЕНЕНИЕ РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЕЙ

Рекуррентные сети могут использоваться для обработки динамических данных и временных образов, прогнозирования и идентификации систем, распознавания речи, видео, обработки естественного языка и управления. Рассмотрим решение некоторых задач при помощи рекуррентных нейронных сетей.

##### *Временная версия задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ»*

Рассмотрим пример использования рекуррентной нейронной сети для решения временной задачи «ИСКЛЮЧАЮЩЕЕ ИЛИ» (temporal version of the **XOR** problem). Как ранее, стандартная задача XOR заключается в преобразовании входных двумерных данных вида (00, 11, 01, 10) в выходной одномерный вектор (0, 0, 1, 1).

Для отображения указанной задачи во временной ряд необходимо случайным образом выбрать два произвольных бита и провести над ними операцию XOR. В результате получается третий бит временного ряда. Далее снова генерируются два случайных бита. Процесс продолжается до тех пор, пока не получится временная последовательность размерностью  $X(n)$ . Например,

$$X(n) = [101110000011110101 \dots], \quad (4.8)$$

т. е. задача прогнозирования заключается в предсказании каждого последующего бита временной последовательности на основе знания о значении предыдущего:

$$x(i+1) = f(x(i)), \quad \forall i. \quad (4.9)$$

В результате выходные значения нейронной сети временного ряда (4.8) образуют последовательность битов

$$Y(m) = [0111 0000011110101 \dots].$$

Для решения задачи можно применять рекуррентную сеть Элмана, архитектура которой состоит из одного входного, двух скрытых, двух

контекстных и одного выходного нейронного элемента [40]. Для обучения такой сети использовалась временная последовательность, состоящая из 3000 бит. После ее обучения был точно предсказан каждый третий выходной бит для входных данных:  $i = 2, 5, 8, \dots$ .

### *Прогнозирование странных аттракторов*

Прогнозирующие свойства рекуррентной нейронной сети можно применять для идентификации поведения нелинейных динамических систем, например для построения странных аттракторов. Странные аттракторы имеют фрактальную размерность и характеризуют хаотические процессы.

Пусть для заданного временного ряда необходимо спрогнозировать последующие его значения, т. е.  $x(t+1) = f(x(1), x(2), \dots, x(t))$ .

Ф. Такенс показал [42], что, используя только одну координату динамической системы, можно реконструировать исходный аттрактор в пространстве точек с задержками  $(x(t), x(t+\tau), \dots, x(t+(m-1)\tau))$ . При этом сохраняются важнейшие динамические и топологические свойства оригинального аттрактора. Размерность  $m$  определяется по формуле

$$m \geq 2[d] + 1,$$

где  $d$  – фрактальная размерность аттрактора, а  $[d]$  обозначает целую часть фрактальной размерности.

В данном случае фразу *важнейшие динамические свойства* следует понимать как диссипативность и хаотичность системы, а *важнейшие геометрические свойства* – как топологические инварианты аттрактора, такие, например, как фрактальная размерность. Выполнение неравенства  $m \geq 2[d] + 1$  гарантирует сохранение указанных свойств. Однако эксперименты показывают, что данная оценка размерности пространства вложения является несколько завышенной. Для некоторых простых хаотических систем сохранение упомянутых свойств наблюдается даже при  $m = [d + 1]$ . Для максимальной предсказуемости хаотического процесса необходимо вначале провести псевдофазовую реконструкцию одномерного сигнала.

*Псевдофазовая реконструкция* – это отображение, которое точке  $x(t)$  временного ряда ставит в соответствие точку

$$(x(t), x(t+\tau), \dots, x(t+(m-1)\tau)) \in \mathbb{R}^m,$$

где  $t$  – дискретное время ( $t = \overline{(m-1)\tau + 1, N}$ );  $\tau$  – временная задержка (в дискретах времени);  $m$  – размерность пространства вложения.

Таким образом, для прогнозирования хаотического сигнала необходимо определить параметры вложения динамической системы, а именно: подходящую временную задержку сигнала  $\tau$  [38] и размерность  $m$  [38] пространства вложения для псевдофазовой реконструкции. Тогда общая структура прогнозирующей нейронной сети будет состоять из  $k \geq m - 1$  входных нейронов ( $m$  – размерность пространства вложения в соответствии с теоремой Такенса),  $p$  скрытых и одного выходного нейронного элемента [38]. Рассмотрим пример использования рекуррентных нейронных сетей для построения странных аттракторов.

Пусть сигнал  $x(t)$  известен на определенном промежутке времени. С помощью рекуррентной сети его можно пролонгировать на упреждающий промежуток времени. Отображая полученные значения  $x(t)$  на псевдофазовую плоскость в системе координат  $x(t)$  и  $x(t+\tau)$ , можно построить соответствующий аттрактор динамической системы. Рассмотрим данный процесс на примере отображения Энона, которое описывается следующими уравнениями:

$$x_{n+1} = 1 - \alpha x_n^2 + y_n,$$

$$y_{n+1} = \beta x_n,$$

где  $\alpha = 1,4$ ;  $\beta = 0,3$ .

Общий вид оригинального аттрактора, рассчитанный для 1500 точек по приведенным выше формулам, представлен на рис. 4.5. Предположим, что известны значения координаты  $x$  для  $n = 1500$ . В качестве рекуррентной сети будем использовать сеть Элмана (см. рис. 4.3), которая имеет один линейный нейронный элемент в последнем слое. Поскольку отображение Энона имеет фрактальную размерность  $d = 1,26$ , то размерность окна должна выбираться исходя из условия  $n \geq 2$ .

Пусть количество входных нейронов сети  $n = 7$ , а количество нейронов в скрытом слое равно 5. Для обучения и прогнозирования будем применять метод скользящего окна с шагом 1.

В качестве функции активации нейронных элементов скрытого слоя будем использовать сигмоидную функцию. После обучения сети спрогнозируем переменную  $x$  на 1500 шагов вперед. Результаты прогноза для 50 шагов показаны на рис. 4.6.

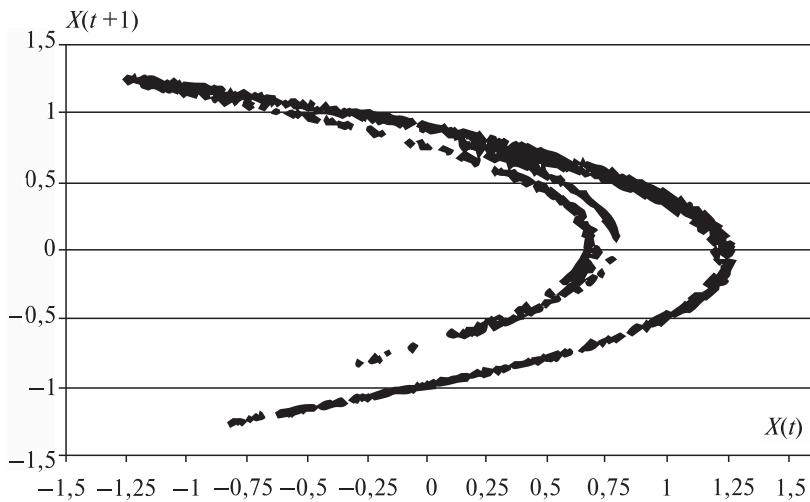


Рис. 4.5. Оригинальный аттрактор Энона, построенный для 1500 точек

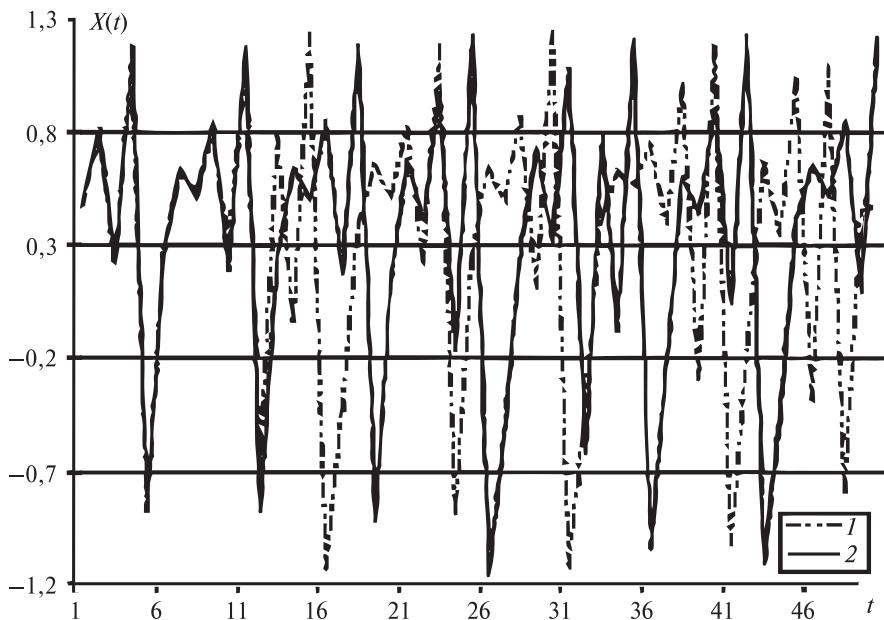


Рис. 4.6. Многошаговый прогноз процесса Энона для 50 шагов:  
1 – оригинальная последовательность; 2 – прогноз

Как следует из рис. 4.6, сеть достаточно точно прогнозирует для первых 12 шагов, а затем наблюдается отличие между эталонным и прогнозирующим рядами. Такое отличие обусловлено хаотической природой сигнала  $x$ , которая чувствительно зависит от начальных условий. При этом расстояние между двумя достаточно близкими точками на хаотическом аттракторе с течением времени увеличивается по экспоненциальному закону:

$$d(t) = d_0 e^{\lambda t},$$

где  $d_0$  – начальное расстояние между двумя точками;  $\lambda$  – старший показатель Ляпунова, который характеризуется положительными значениями для хаотических процессов.

Для отображения Энона  $\lambda = 0,418$ . Рассмотрим построение аттрактора Энона. Для этого 1500 спрогнозированных значений, полученные при помощи рекуррентной сети, отобразим на псевдофазовую плоскость  $X(t)$ ,  $X(t+1)$  (рис. 4.7).

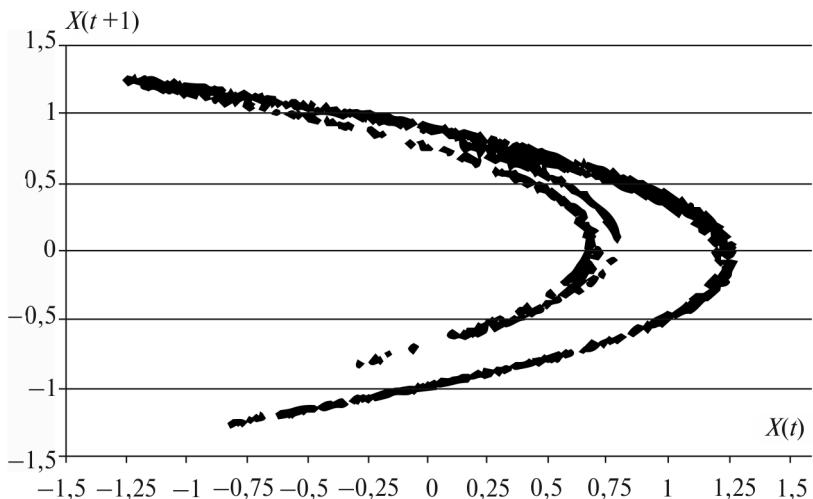


Рис. 4.7. Восстановленный аттрактор Энона, построенный для 1500 точек многошагового прогноза

Как видно из рис. 4.7, форма прогнозируемого аттрактора соответствует оригинальному. Таким образом, рекуррентная нейронная сеть демонстрирует высокую обобщающую способность для хаотических процессов и позволяет моделировать поведение нелинейных динамических систем.

## Глава 5

### СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

Сверточные нейронные сети (convolutional neural networks, CNN) являются дальнейшим развитием многослойного персептрона и неокогнитрона и широко используются для обработки изображений. В отличие от многослойного персептрона, сверточные нейронные сети позволяют учитывать топологию изображений и инвариантны к сдвигам, масштабированию и другим искажениям входного образа. В данной главе рассматриваются основные принципы построения и обучения сверточных нейронных сетей [43–45], приводится упрощенная архитектура сверточной нейронной сети [46], которая дает возможность классифицировать рукописные цифры с большей точностью, чем обычные сверточные сети архитектуры LeNet-5.

#### 5.1. ПОСТРОЕНИЕ СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ

Сверточная нейронная сеть объединяет три подхода при обработке изображений. Это использование локального рецептивного поля для каждого нейрона сверточного слоя, формирование сверточных слоев в виде набора карт, нейронные элементы которых имеют одинаковые синаптические связи, и наличие карт подвыборочного (subsampling) слоя, повышающего инвариантность сети к искажениям [43–45]. Применение локального рецептивного поля позволяет нейронам одной карты признаков детектировать один и тот же стимул в разных фрагментах изображения. Использование идентичных нейронов в каждой карте дает возможность сократить количество настраиваемых синаптических связей сети. Подвыборочный слой осуществляет локальное усреднение или операцию максимума для каждого неперекрывающегося фрагмента карты сверточного слоя, что позволяет уменьшить размерность соответствующих карт признаков. Общая архитектура сверточной сети показана на рис. 5.1. Она состоит из совокупности сверточных слоев  $C_1, C_3, C_5$  и подвыборочных слоев  $S_2, S_4$ .

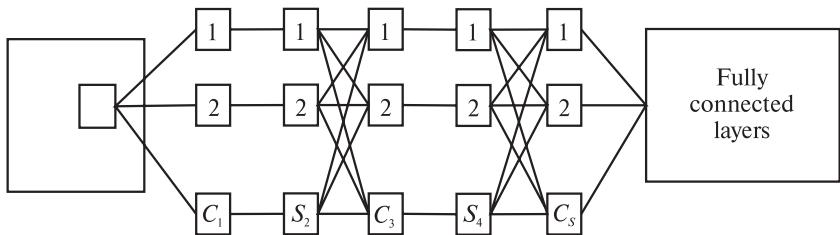


Рис. 5.1. Архитектура сверточной нейронной сети

Как уже отмечалось, сверточный слой состоит из множества карт признаков, где нейроны каждой карты содержат одни и те же наборы весов и порогов. В результате нейронные элементы каждой карты признаков выполняют одни и те же операции над различными частями изображения.

Для сканирования изображения используется метод скользящего окна, которое называется локальным рецептивным полем (receptive field) для соответствующего ему нейрона карты признаков. Поэтому если размер скользящего окна, которое называется ядром, равняется  $p \times p$  (рецептивное поле), то каждый нейрон сверточного слоя связан с  $p^2$  элементами соответствующей области рецептивного поля изображения. Каждое рецептивное поле (окно) во входном пространстве образов отображается на специальный (отдельный) нейрон в каждой карте признаков. Если скользящее окно сканирует изображение с единичным шагом, то количество нейронов в каждой карте признаков определяется следующим образом:

$$D(C_1) = (n - p + 1)(n - p + 1), \quad (5.1)$$

где  $n \times n$  – размерность исходного изображения.

Если скользящее окно сканирует изображение с шагом  $s$ , то количество нейронов в каждой карте признаков в общем случае вычисляется по формуле

$$D(C_1) = \left( \frac{n - p}{s} + 1 \right) \left( \frac{n - p}{s} + 1 \right). \quad (5.2)$$

Общее количество различных синаптических связей в сверточном слое равно

$$V(C_1) = M(p^2 + 1), \quad (5.3)$$

где  $M$  – общее количество карт признаков в сверточном слое.

Как следует из последнего выражения, применение сверточной сети позволяет сократить общее количество настраиваемых синаптических связей по сравнению с многослойным персепtronом за счет использования идентичных нейронов в каждой карте признаков.

Для упрощения математического описания сверточного слоя представим пиксели входного изображения в одномерном пространстве. Тогда выходное значение  $ij$ -го нейрона для  $k$ -й карты признаков в конволюционном слое определяется как

$$y_{ij}^k = F(S_{ij}^k), \quad (5.4)$$

$$S_{ij}^k = \sum_c w_{cij}^k x_c - T_{ij}^k, \quad (5.5)$$

где  $c = \overline{1, p^2}$ ;  $F$  – функция активации;  $S_{ij}^k$  – взвешенная сумма  $ij$ -го нейрона в  $k$ -й карте признаков;  $w_{cij}^k$  – весовой коэффициент между  $c$ -м нейроном входного слоя и  $ij$ -м нейроном в  $k$ -й карте признаков;  $T_{ij}^k$  – пороговое значение  $ij$ -го нейрона в  $k$ -й карте признаков.

Как уже отмечалось, нейроны каждой карты признаков имеют одинаковый набор весов и порогов. В результате из одного и того же изображения можно извлечь множество различных признаков. Далее в целях уменьшения размерности карт признаков эти признаки обрабатываются следующим слоем  $S_2$ , который называется объединяющим (pooling) или подвыборочным (subsampling). Он осуществляет сжатие карт признаков сверточного слоя при помощи операции максимизации или локального усреднения различных областей карт признаков. Для этого каждая карта признаков сверточного слоя разбивается на неперекрывающиеся области размером  $k \times k$ . Каждая область отображается в один нейрон соответствующей карты подвыборочного слоя. Следует также отметить, что каждая карта признаков сверточного слоя связана лишь с одной картой в пулинговом слое. Каждый нейрон подвыборочного слоя вычисляет среднее или максимальное значение  $k^2$  нейронов в сверточном слое:

$$z_j = \frac{1}{k^2} \sum_{j=1}^{k \times k} y_j \text{ или } z_j = \max(y_j). \quad (5.6)$$

Количество нейронов в каждой карте подвыборочного слоя

$$D(S_2) = D(C_1) / k^2. \quad (5.7)$$

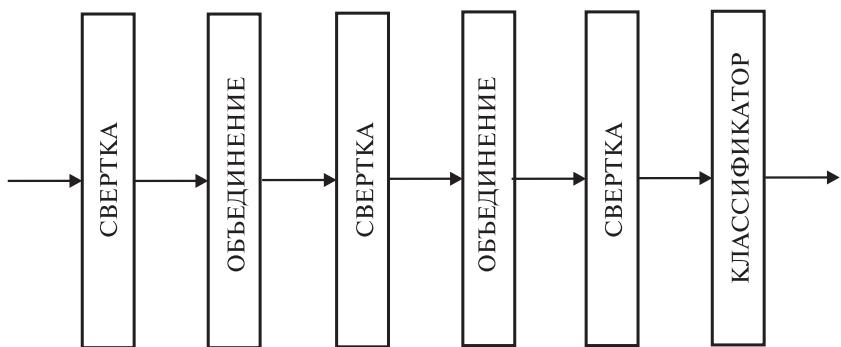


Рис. 5.2. Общая структура сверточной нейронной сети

Число карт признаков в слое пулинга будет таким же, как и в сверточном слое. Далее опять следуют сверточный ( $C_3$ ), пулинговый ( $S_4$ ) и сверточный ( $C_5$ ) слои. При этом каждый нейронный элемент пулингового слоя в общем случае имеет синаптические связи со всеми нейронами сверточного слоя (см. рис. 5.1). Таким образом, сверточная нейронная сеть представляет собой сочетание сверточных и пулинговых слоев, которые выполняют нелинейное иерархическое преобразование входного пространства образов (рис. 5.2). Последний блок сверточной нейронной сети является многослойным персепtronом, машиной опорных векторов или другим классификатором.

## 5.2. АРХИТЕКТУРА СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ LENET-5

Рассмотрим традиционную сверточную нейронную сеть (LeNet-5) для классификации рукописных цифр (рис. 5.3).

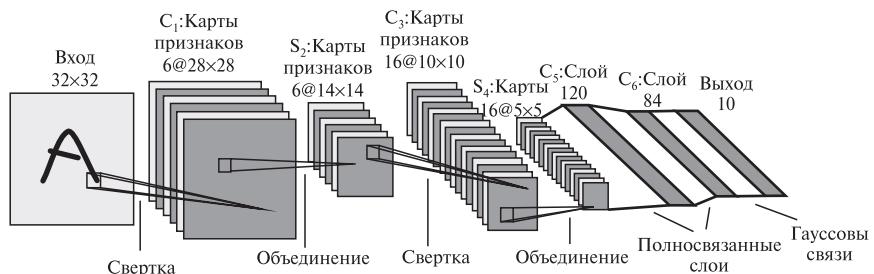


Рис. 5.3. Архитектура сети LeNet-5

Входное изображение имеет размер  $32 \times 32$  пикселя. Скользящее окно размерностью  $5 \times 5$  сканирует изображение с шагом 1, и каждый фрагмент изображения поступает на соответствующий нейронный элемент карты признаков сверточного слоя  $C_1$  нейронной сети. Слой  $C_1$  состоит из шести карт признаков, где каждая карта согласно выражению (5.1) содержит  $28 \times 28$  нейронов. Слой  $S_2$  представляет собой подвыборочный слой с шестью картами признаков. Каждый нейрон подвыборочного слоя вычисляет среднее значение четырех нейронов в сверточном слое, т. е.  $k = 2$ . Тогда в соответствии с выражением (5.7) каждая карта подвыборочного слоя  $S_2$  содержит  $14 \times 14$  нейронных элементов. Слой  $C_3$  является сверточным слоем с 16 картами признаков и ядром сканирования  $5 \times 5$  для каждой карты подвыборочного слоя. Тогда размерность каждой карты признаков сверточного слоя  $C_3$  равняется  $10 \times 10$  нейронных элементов. В LeNet-5 слои  $S_2$  и  $C_3$  образуют не полностью связанный нейронную сеть, где нейроны этих слоев имеют синаптические связи между собой в соответствии с табл. 5.1 [45].

Таблица 5.1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X				X	X	X			X	X	X	X		X	X
2	X	X				X	X	X			X	X	X	X		X
3	X	X	X				X	X	X			X		X	X	X
4		X	X	X			X	X	X	X			X		X	X
5			X	X	X			X	X	X	X		X	X		X
6				X	X	X			X	X	X	X		X	X	X

Каждые столбец и строка таблицы характеризуют соответственно карты признаков сверточного  $C_3$  и подвыборочного  $S_2$  слоев. Знак  $X$  указывает на наличие связей между нейронами соответствующих карт.

Слой  $S_4$  является подвыборочным слоем с 16 картами признаков и ядром  $2 \times 2$  для каждой карты признаков сверточного слоя. Поэтому каждый нейрон подвыборочного слоя  $S_4$  вычисляет среднее значение четырех нейронов в сверточном слое  $C_3$ , т. е.  $k = 2$ . Поэтому размерность карты признаков данного слоя составляет  $5 \times 5$  нейронных элементов. Следующий сверточный слой  $C_5$  сканирует подвыборочный слой окном размерностью  $5 \times 5$ .

При этом каждый нейрон подвыборочного слоя имеет синаптические связи со всеми нейронами сверточного слоя. Слой  $F_6$  содержит 84 нейронных элемента с функцией активации гиперболического тангенса и функционирует как классический персептронный слой. Выходной слой состоит из 10 нейронных элементов, каждый из которых формирует выходное значение в соответствии с радиально-базисной функцией активации:

$$y_j = \exp\left(\frac{-\sum_i (x_i - w_{ij})^2}{2\sigma^2}\right), \quad (5.8)$$

где  $x_i$  — выходное значение  $i$ -го нейронного элемента слоя  $F_6$ ;  $w_{ij}$  — весовые коэффициенты между последними слоями;  $\sigma$  — среднеквадратичное отклонение, характеризующее ширину радиально-базисной функции.

Согласно выражению (5.2) чем больше отличается входной вектор от весового вектора соответствующего выходного нейронного элемента, тем меньше будет выходное значение этого нейрона.

### 5.3. ОБУЧЕНИЕ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Для обучения сверточных нейронных сетей используется алгоритм обратного распространения ошибки, адаптированный к архитектуре сверточной сети. Целью обучения является минимизация суммарной квадратичной ошибки сети, которая характеризует разницу между реальными и эталонными выходными значениями сети. Значение суммарной квадратичной ошибки для  $L$  тренировочных наборов определяется следующим образом:

$$E_s = \frac{1}{2} \sum_{k=1}^L \sum_{j=1}^m (y_j^k - e_j^k)^2, \quad (5.9)$$

где  $y_j^k$  и  $e_j^k$  — соответственно реальное и эталонное выходное значение  $j$ -го нейрона выходного слоя для  $k$ -го образа.

Тогда согласно методу градиентного спуска для минимизации суммарной квадратичной ошибки сети синаптические связи  $ij$ -го нейрона карты признаков сверточного слоя в случае группового обучения (mini-batch) должны изменяться следующим образом:

$$w_{cij}(t+1) = w_{cij}(t) - \alpha \frac{\partial E(r)}{\partial w_{cij}(t)}, \quad (5.10)$$

где  $\alpha$  – скорость обучения;  $E(r)$  – суммарная квадратичная ошибка для группы размерностью  $r$  образов;  $w_{cij}$  – весовой коэффициент между  $c$ -м и  $ij$ -м нейроном соответственно предыдущего и сверточного слоя;  $ij$  – номер (координаты) нейронного элемента в карте признаков.

Поскольку нейроны каждой карты признаков сверточного слоя имеют одинаковые синаптические связи (веса и пороги), то частная производная  $\frac{\partial E(r)}{\partial w_{cij}}$  определяется как сумма частных производных всех нейронов соответствующей карты признаков:

$$\frac{\partial E(r)}{\partial w_{cij}} = \sum_{i,j} \frac{\partial E(r)}{\partial \omega_{cij}}. \quad (5.11)$$

Таким образом, для вычисления определенного весового коэффициента необходимо взять производные по этой связи для всех нейронных элементов карты признаков и просуммировать их. Затем согласно выражению (5.10) полученное значение весового коэффициента присвоить одноименным связям для всех нейронных элементов карты признаков сверточного слоя. Используя результаты раздела 3.3, можно получить следующее выражение для модификации весовых коэффициентов:

$$\omega_{cij}(t+1) = \omega_{cij}(t) - \alpha(t) \cdot \sum_{i,j} \sum_{k=1}^r \gamma_{ij}^k \cdot F'(S_{ij}^k) x_c^k, \quad (5.12)$$

где  $c = \overline{1, p^2}$ ,  $F'(S_{ij}^k) = \frac{\partial y_{ij}^k}{\partial S_{ij}^k}$  – производная функции активации для  $k$ -го образа;  $S_{ij}^k$  – взвешенная сумма нейрона с номером  $ij$  в карте признаков;  $\gamma_{ij}^k$  – ошибка  $ij$ -го нейрона для  $k$ -го образа;  $x_c^k$  –  $c$ -я компонента  $k$ -го входного образа соответствующей карты признаков сверточного слоя.

## 5.4. РЕДУЦИРОВАННАЯ АРХИТЕКТУРА СВЕРТОЧНОЙ СЕТИ ДЛЯ РАСПОЗНАВАНИЯ РУКОПИСНЫХ ЦИФР

Во многих практических приложениях важной проблемой являются ограниченные вычислительные ресурсы, которые не позволяют использовать сложные нейронные сети. Поэтому снижение сложности нейронных сетей при сопоставимой точности решения задачи — достаточно важная проблема. В данном разделе рассматривается сверточная нейронная сеть с более простой архитектурой по сравнению с LeNet-5. Архитектура упрощенной сверточной нейронной сети для распознавания рукописных цифр изображена на рис. 5.4.

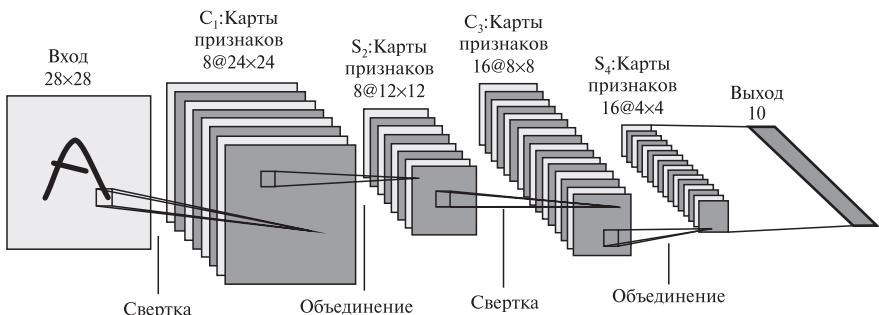


Рис. 5.4. Архитектура редуцированной сверточной нейронной сети

Упрощенная сверточная нейронная сеть состоит из сверточного ( $C_1$ ), пулингового ( $S_1$ ), сверточного ( $C_3$ ), пулингового ( $S_4$ ) и сверточного ( $C_5$ ) слоев [46]. Входное изображение имеет размер  $28 \times 28$  пикселей. Изображение сканируется скользящим окном размерностью  $5 \times 5$  с шагом 1, и каждый фрагмент изображения поступает на соответствующий нейронный элемент карты признаков сверточного слоя  $C_1$  нейронной сети. Сверточный слой  $C_1$  состоит из 8 карт признаков, где каждая карта содержит  $24 \times 24$  нейрона. Карты признаков сверточного слоя разбиваются на неперекрывающиеся области размером  $2 \times 2$ , каждая из которых отображается на соответствующий нейронный элемент пулингового слоя. Пулинговый слой  $S_2$  содержит 8 карт признаков размерностью  $12 \times 12$  нейронов для каждой карты признаков. Сверточный слой  $C_3$  содержит 16 карт признаков размерностью  $8 \times 8$  и ядром обхода  $5 \times 5$ . По сравнению с традиционной сетью LeNet-5 слои  $S_2$

и  $C_3$  – полностью связанные, т. е. каждый нейрон слоя  $S_2$  имеет связи со всеми нейронными элементами слоя  $C_3$ . Слой  $S_4$  представляет собой пулинговый слой с ядром обхода сверточного слоя  $2 \times 2$ , который состоит из 16 карт признаков размерностью  $4 \times 4$  нейронных элементов. Последний слой  $C_5$  – выходной, состоящий из 10 нейронных элементов и выполняющий классификацию образов. Основные отличия представленной архитектуры сверточной нейронной сети от LeNet-5 следующие: 1) отсутствуют два слоя размерностью 120 и 84 нейронных элементов; 2) слои  $S_2$  и  $C_3$  – полностью связанные; 3) сигмоидная функция активации используется во всех сверточных слоях.

Рассмотрим применение предложенной сверточной сети для распознавания рукописных цифр на основе базы данных MNIST. База данных MNIST содержит 60 000 изображений для обучения и 10 000 изображений для тестирования. Каждое изображение имеет размерность  $28 \times 28$  пикселей в градациях серого. Для обучения сети будем использовать групповое обучение с алгоритмом обратного распространения ошибки.

Размерность группы (minibatch) составляет 50 образов; значение шага обучения изменяется от 0,8 до 0,0001. После обучения сети ошибка распознавания на тестовом множестве составила 0,71 %. Сравнительный анализ различных сетей этого класса приведен в табл. 5.2 (<http://yann.lecun.com/exdb/mnist/>).

Таблица 5.2

## Сравнительный анализ

Классификатор	Ошибка распознавания (%)
Сверточная сеть LeNet-1	1,7
Сверточная сеть LeNet-4	1,1
Сверточная сеть LeNet-5, [no distortions]	0,95
Сверточная сеть LeNet-5, [distortions]	0,8
Редуцированная сверточная сеть, [no distortions]	0,71

Как следует из данных табл. 5.2, лучшим результатом сверточной сети LeNet-5 без использования искусственных искажений (distortions) в обучающей выборке является ошибка распознавания 0,95 %, а при использовании искажений – 0,8 %. Таким образом, применение редуцированной сверточной нейронной сети позволяет достичь более высокой точности распознавания по сравнению с традиционной архитектурой сверточной сети.

## Г л а в а 6

### АВТОЭНКОДЕРНЫЕ НЕЙРОННЫЕ СЕТИ

Автоэнкодерные нейронные сети характеризуются как прямым  $Y = f(X)$ , так и обратным  $\bar{X} = f^{-1}(Y)$  преобразованием информации (рис. 6.1). При прямом преобразовании данных происходит ее сжатие, а при обратном – восстановление. Автоэнкодерные нейронные сети называются также рециркуляционными, автоассоциативными, NPCA (nonlinear principal component analysis) нейронными сетями [27, 29].

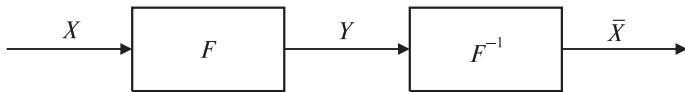


Рис. 6.1. Преобразование информации в автоэнкодере

Автоэнкодерные нейронные сети применяются в общем случае для выделения наиболее важных информативных признаков (feature extraction) во входном пространстве образов, сжатия информации, очистки данных от шумов, визуализации и классификации данных. Задача такого преобразования заключается в достижении наилучшего автопрогноза вектора  $X$  при отображении его в пространство меньшей размерности:

$$|X - \bar{X}| \rightarrow \min. \quad (6.1)$$

Отсюда следует, что сжатие информации здесь осуществляется таким образом, чтобы восстановить информацию с наименьшими потерями. Обучение данных сетей производится без учителя в соответствии с критерием (6.1). Впервые использование автоэнкодерных нейронных сетей было предложено в 1982 г. в работе [47]. В настоящее время их эволюция происходит по пути увеличения количества слоев (deep auto encoder – глубокий автоэнкодер). Теоретической основой данного класса сетей является *анализ главных компонент* (principal component analysis) [48–50]. В данной главе описываются метод главных компонент, архи-

тектура, обучение и применение автоэнкодерных нейронных сетей. Показано, что автоэнкодерные нейронные сети, в отличие от метода главных компонент, позволяют осуществить нелинейное преобразование информации.

## 6.1. МЕТОД ГЛАВНЫХ КОМПОНЕНТ

Метод главных компонент применяется в статистике для сжатия информации без существенных потерь ее информативности [48–50]. Он состоит в линейном ортогональном преобразовании входного вектора  $X(n)$  размерностью  $n$  в выходной вектор  $Y(p)$  размерностью  $p$ , где  $p < n$ . Такое преобразование эквивалентно переходу к новой системе координат и проецированию входных данных на новые оси координат. Новая координатная ось, соответствующая первой главной компоненте, направлена в сторону максимального разброса данных, т. е. в сторону максимальной дисперсии (рис. 6.2). Вторая координатная ось перпендикулярна первой оси и т. д. При этом компоненты вектора  $Y$  являются некоррелированными между собой, а общая дисперсия после преобразования при  $p = n$  остается неизменной.

Недостаток метода главных компонент состоит в том, что он линейный, т. е. проецирование входных данных осуществляется на линейные оси координат, что не позволяет в большинстве случаев произвести эффективное сжатие информации.

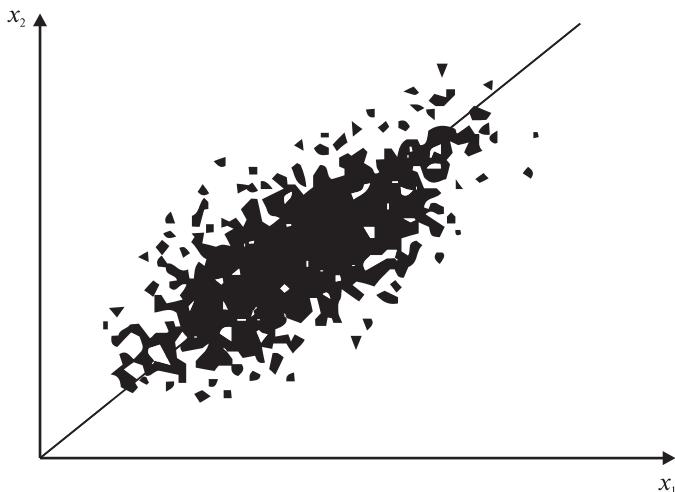


Рис. 6.2. Ось первой главной компоненты

Совокупность входных образов представим в виде матрицы

$$X = \begin{bmatrix} x^1 \\ x^2 \\ \dots \\ x^L \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_n^1 \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \dots \\ x_1^L & x_2^L & \dots & x_n^L \end{bmatrix},$$

где  $X^k = (x_1^k, x_2^k \dots x_n^k)$  соответствует  $k$ -му входному образу;  $L$  – общее количество образов.

Рассмотрим линейные комбинации переменных входного пространства образов:

$$y_1 = w_{11}x_1 + w_{21}x_2 + \dots + w_{n1}x_n,$$

$$y_2 = w_{12}x_1 + w_{22}x_2 + \dots + w_{n2}x_n,$$

.....

$$y_p = w_{1p}x_1 + w_{2p}x_2 + \dots + w_{np}x_n.$$

Переменные  $y_j$ ,  $j = \overline{1, p}$ , называются главными компонентами.

В матричном виде

$$Y = XW,$$

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \dots \\ W_p \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1p} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2p} \\ \dots \\ \omega_{n1} & \omega_{n2} & \dots & \omega_{np} \end{bmatrix}, \quad X = [x_1 \ x_2, \dots, x_n].$$

Для определения главных компонент необходимо определить матрицу преобразования  $W$ . Метод главных компонент состоит в нахождении таких линейных комбинаций исходных переменных, при которых выполняются следующие условия:

1. Главные компоненты должны быть некоррелированными между собой, т. е.

$$\sigma(y_i, y_j) = \frac{1}{L} \sum_{k=1}^L y_i^k y_j^k = 0, \quad i, j = \overline{1, n}.$$

2. Дисперсия первой главной компоненты должна быть больше дисперсии второй главной компоненты и т. д.:

$$\sigma(y_1) > \sigma(y_2) > \dots > \sigma(y_n),$$

где дисперсия  $j$ -й главной компоненты

$$\sigma(y_j) = \frac{1}{L} \sum_{k=1}^L (y_j^k)^2 = 0.$$

3. Сумма дисперсий в исходном пространстве образов должна равняться сумме дисперсий в пространстве главных компонент:

$$\sum_i^n \sigma(x_i) = \sum_i^n \sigma(y_i).$$

Дисперсия характеризует количество информации входного пространства образов. В методе главных компонент первая главная компонента аккумулирует большую часть дисперсии и большую часть информации входного пространства образов соответственно. Вторая главная компонента накапливает следующую по величине дисперсию входного пространства образов и т. д. В результате такого преобразования  $p$  первых главных компонент содержат большую часть информации исходного пространства образов и сжатие информации осуществляется с наименьшими потерями соответственно.

### 6.1.1. Сжатие информации

Сжатие информации заключается в нахождении главных компонент и отбрасывании компонент с малой величиной дисперсии. Алгоритм нахождения главных компонент состоит из следующих шагов:

1. Преобразование исходной матрицы входных образов в центрированную матрицу, вектор математических ожиданий которой, рассчитанный по столбцам матрицы, равняется нулю:

$$\mu = (\mu^1, \mu^2, \dots, \mu^n) = 0.$$

Для этого необходимо вначале вычислить математическое ожидание (выборочное среднее) каждого столбца матрицы входных образов:

$$\mu^i = \sum_{k=1}^L x_i^k / L. \quad (6.2)$$

Затем из каждой компоненты столбца следует вычесть математическое ожидание соответствующего столбца матрицы:

$$x_i^k = x_i^k - \mu^i. \quad (6.3)$$

Далее будем оперировать только центрированными входными образами.

2. Матрица ковариаций входных образов размерностью  $n \times n$  имеет вид

$$K = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \dots & \sigma_{2n} \\ \dots & \dots & \dots & \dots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_{nn} \end{bmatrix},$$

где  $\sigma_{ij}$  – ковариация между  $i$ -й и  $j$ -й компонентой входных образов.

Элементы матрицы ковариаций вычисляются на основе центрированной матрицы входных образов:

$$\sigma_{ij} = \frac{1}{L} \sum_{k=1}^L x_i^k x_j^k, \quad (6.4)$$

где  $i, j = \overline{1, n}$ .

Главная диагональ ковариационной матрицы характеризует дисперсию соответствующих входных образов. Ковариационная матрица является действительной симметрической матрицей.

3. Определение собственных значений  $\lambda_i$ ,  $i = \overline{1, n}$ , ковариационной матрицы требует решения характеристического уравнения

$$\det(K - \lambda I) = 0, \quad (6.5)$$

где  $I$  – единичная матрица, элементы главной диагонали которой принимают единичные значения, а остальные – нулевые значения.

В соответствии с этим уравнением необходимо раскрыть определитель и найти собственные числа ковариационной матрицы

$$\begin{bmatrix} \sigma_{11} - \lambda & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} - \lambda & \dots & \sigma_{2n} \\ \dots & \dots & \dots & \dots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_{nn} - \lambda \end{bmatrix} = 0. \quad (6.6)$$

Поскольку ковариационная матрица  $K$  симметрична, то уравнение (6.6) имеет  $n$  вещественных корней.

4. Расположение собственных чисел ковариационной матрицы в порядке убывания:

$$\lambda_1 > \lambda_2 > \dots > \lambda_n.$$

Следует отметить, что собственные числа ковариационной матрицы характеризуют дисперсию главных компонент. Так, дисперсия  $j$ -й главной компоненты определяется следующим образом:

$$\lambda_j = \sigma(y_j) = \frac{1}{L} \sum_{k=1}^L (y_j^k)^2. \quad (6.7)$$

Если собственное число  $\lambda_j = 0$ , то можно отбросить  $j$ -ю главную компоненту без потери информации.

5. Нахождение собственных векторов ковариационной матрицы  $W_1$ ,  $W_2$ ,  $W_p$ , которые являются столбцами матрицы преобразований  $W$ . Первый собственный вектор соответствует максимальному собственному значению (максимальной дисперсии)  $\lambda_1$ . Поэтому для нахождения первого собственного вектора  $W_1$  следует решить систему уравнений:

$$(K - \lambda_1 I) W_1^T = 0,$$

где  $W_1 = [\omega_{11}, \omega_{21}, \dots, \omega_{n1}]$ .

В обычной форме система уравнений записывается как

$$(\sigma_{11} - \lambda_1) \omega_{11} + \sigma_{12} \omega_{21} + \dots + \sigma_{1n} \omega_{n1} = 0,$$

$$\sigma_{21} \omega_{11} + (\sigma_{22} - \lambda_1) \omega_{21} + \dots + \sigma_{2n} \omega_{n1} = 0,$$

.....

$$\sigma_{n1} \omega_{11} + \sigma_{n2} \omega_{21} + \dots + (\sigma_{nn} - \lambda_1) \omega_{n1} = 0.$$

Для нахождения второго собственного вектора  $W_2$  необходимо решить систему уравнений:

$$(K - \lambda_2 I) W_2^T = 0.$$

Аналогичным образом находятся остальные собственные векторы.

6. Ортонормирование собственных векторов. Как известно, собственные векторы действительной симметрической матрицы  $K$  являются ортогональными, т. е.

$$W_i W_j^T = \begin{cases} 0, & i \neq j, \\ \text{const}, & i = j. \end{cases}$$

В методе главных компонент собственные векторы должны быть ортонормированными. Для получения ортонормированного вектора  $W_i$  его требуется пронормировать:

$$W_i = \left( \frac{w_{1i}}{|W_i|}, \frac{w_{2i}}{|W_i|}, \dots, \frac{w_{ni}}{|W_i|} \right),$$

где  $|W_i| = \sqrt{w_{1i}^2 + w_{2i}^2 + \dots + w_{ni}^2}$ .

Как уже отмечалось, метод главных компонент состоит в переходе к новой системе координат для входных данных. При этом первый собственный вектор характеризует новую координатную ось  $W_1$ , которая проходит, например, в двумерном случае через точки с координатами  $(0,0)$  и  $(w_{11}, w_{21})$  и направлена в сторону максимальной дисперсии входных данных. Второй собственный вектор  $W_2$  характеризует вторую координатную ось, которая перпендикулярна первой оси и т. д.

7. Нахождение главных компонент для всех входных образов. Так, для  $k$ -го входного образа  $x^k$  главные компоненты

$$y^k = x^k W.$$

Приведенный выше алгоритм позволяет найти главные компоненты входного пространства образов. Получаемая в результате матрица преобразований  $W$  является ортогональной, т. е.

$$WW^T = I.$$

Собственные числа ковариационной матрицы характеризуют дисперсию главных компонент. При этом сумма дисперсий в пространстве исходных образов равняется сумме дисперсий в пространстве выходных образов:

$$\sum_{i=1}^n \sigma(x_i) = \sum_{i=1}^n \lambda_i = \sum_{i=1}^n \sigma(y_i). \quad (6.8)$$

### 6.1.2. Восстановление информации

Рассмотрим отображение выходного вектора  $Y$  во входной вектор  $X$ . Такое отображение называется восстановлением информации или автопрогнозом [48]. Восстановленная информация определяется как

$$\bar{X} = YQ, \quad (6.9)$$

где  $Q$  – матрица размерности  $n \times p$ .

Задача восстановления информации заключается в достижении минимальной ошибки между входными и восстановленными данными:

$$\varepsilon = |X - \bar{X}| \rightarrow \min.$$

**Теорема 6.1.** Минимальное значение ошибки восстановления информации в методе главных компонент достигается, когда строки матрицы  $Q$  равняются собственным векторам  $W_i$ , вычисленным в соответствии с методом главных компонент, т. е.

$$Q = W^T = \begin{bmatrix} w_{11} & w_{21} & \dots & w_{n1} \\ w_{12} & w_{22} & \dots & w_{n2} \\ \dots & \dots & \dots & \dots \\ w_{1p} & w_{2p} & \dots & w_{np} \end{bmatrix}.$$

Таким образом, наилучший автопрогноз имеет место, когда

$$\bar{X} = YW^T. \quad (6.10)$$

Величина абсолютной ошибки восстановления информации выражается через собственные числа ковариационной матрицы:

$$\Delta = \sum_{i=1}^{n-p} \lambda_{p+i}. \quad (6.11)$$

Относительная ошибка вычисляется как

$$\delta = \frac{\Delta}{\sum_{i=1}^n \lambda_i} \cdot 100 \%. \quad (6.12)$$

Рассмотрим нахождение количества главных компонент. Для этого можно использовать следующий критерий информативности:

$$J = \frac{\sum_{i=1}^p \lambda_i}{\sum_{i=1}^n \lambda_i} = \frac{\lambda_1 + \lambda_2 + \dots + \lambda_p}{\lambda_1 + \lambda_2 + \dots + \lambda_n}, \quad (6.13)$$

который позволяет ориентировочно определить число главных компонент  $p$ . Так, анализируя при помощи выражения (6.13) изменение  $J$  в зависимости от числа  $p$ , можно подобрать необходимое количество компонент без существенной потери информативности  $J$  (рис. 6.3).

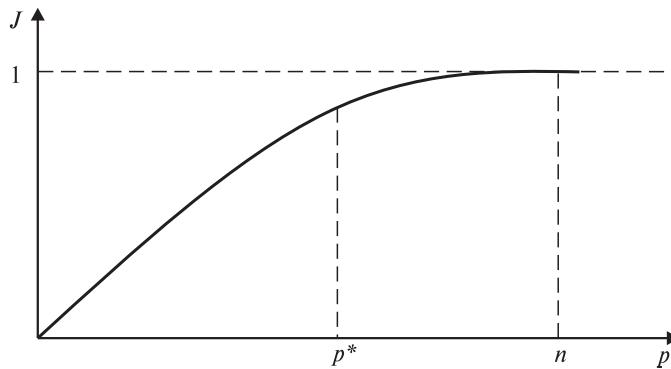


Рис. 6.3. Выбор количества главных компонент

Метод главных компонент является эффективным средством для сжатия и выделения наиболее важных признаков (feature extraction) во входном пространстве образов.

### 6.1.3. Пример расчета по методу главных компонент

Пусть имеем

$$X = \begin{bmatrix} x_1^1 & x_2^1 \\ x_1^2 & x_2^2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} - \text{1-й образ, } \\ - \text{2-й образ.}$$

1. Центрирование матрицы входных образов. Вычислим выборочные средние первого и второго столбца матрицы:

$$\mu^1 = \frac{1+2}{2} = 1,5, \quad \mu^2 = \frac{2+1}{2} = 1,5.$$

В результате получим следующую центрированную матрицу входных образов:

$$X = \begin{bmatrix} -0,5 & 0,5 \\ 0,5 & -0,5 \end{bmatrix}.$$

## 2. Нахождение ковариационной матрицы:

$$\sigma_{11} = \frac{(-0,5)^2 + (0,5)^2}{2} = 0,25;$$

$$\sigma_{12} = \frac{-0,5 \cdot 0,5 + 0,5(-0,5)}{2} = -0,25;$$

$$\sigma_{22} = 0,25; \quad \sigma_{21} = -0,25.$$

Тогда

$$K = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} 0,25 & -0,25 \\ -0,25 & 0,25 \end{bmatrix}.$$

3. Определение собственных чисел ковариационной матрицы:

$$\det(K - \lambda I) = 0,$$

$$\begin{vmatrix} 0,25 - \lambda & -0,25 \\ -0,25 & 0,25 - \lambda \end{vmatrix} = 0,$$

$$(0,25 - \lambda)^2 - (0,25)^2 = 0.$$

Отсюда получим, что  $\lambda_1 = 0$ ,  $\lambda_2 = 0,5$ .

4. Расположение собственных чисел ковариационной матрицы в порядке убывания, т. е.  $\lambda_1 = 0,5$ ,  $\lambda_2 = 0$ .

5. Вычисление собственных векторов ковариационной матрицы для для  $n = p = 2$ :

а) первый собственный вектор:

$$(K - \lambda_1 I)w_1^T = 0,$$

$$\begin{cases} (0,25 - \lambda_1)w_{11} + (-0,25)w_{21} = 0, & -0,25w_{11} - 0,25w_{21} = 0 \\ (-0,25)w_{11} + (0,25 - \lambda_1)w_{21} = 0, & -0,25w_{11} - 0,25w_{21} = 0 \end{cases} \Rightarrow w_{11} = -w_{21}.$$

Пусть  $w_{11} = 1 \rightarrow w_{21} = -1$ ;

б) второй собственный вектор:

$$(K - \lambda_2 I)w_2^T = 0,$$

$$\begin{cases} (0,25 - \lambda_2)w_{12} + (-0,25)w_{22} = 0 \\ (-0,25)w_{12} + (0,25 - \lambda_2)w_{22} = 0 \end{cases} \Rightarrow w_{12} = w_{22}.$$

Пусть  $w_{12} = 1 \rightarrow w_{22} = 1$ .

6. Ортонормирование собственных векторов:

$$|w_1| = \sqrt{2} \rightarrow w_{11} = \frac{1}{\sqrt{2}}, \quad w_{21} = \frac{-1}{\sqrt{2}}; \quad |w_2| = \sqrt{2} \rightarrow w_{12} = \frac{1}{\sqrt{2}}, \quad w_{22} = \frac{1}{\sqrt{2}}.$$

В результате получим следующую матрицу трансформации:

$$W = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

7. Нахождение главных компонент для всех центрированных входных образов:

$$Y^1 = X^1 W, \quad Y^1 = [-0,5; 0,5] \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}.$$

В результате главные компоненты для первого входного образа можно записать в виде

$$y_1^1 = -\frac{1}{\sqrt{2}}, \quad y_2^1 = 0.$$

Аналогично для второго входного паттерна

$$Y^2 = X^2 W, \quad Y^2 = [0,5 - 0,5] \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}.$$

В итоге

$$y_1^2 = \frac{1}{\sqrt{2}}, \quad y_2^2 = 0.$$

Рассмотрим дисперсию главных компонент:

$$\sigma(y_1) = \lambda_1 = \frac{1}{2} \left( (y_1^1)^2 + (y_2^1)^2 \right) = \frac{1}{2} \left( \frac{1}{2} + \frac{1}{2} \right) = 0,5,$$

$$\sigma(y_2) = \lambda_2 = \frac{1}{2} \left( (y_1^2)^2 + (y_2^2)^2 \right) = \frac{1}{2} (0 + 0) = 0.$$

8. Восстановление информации. В общем случае восстановленная информация определяется следующим образом:

$$\bar{X} = YQ; \quad Q = W^T.$$

Восстановленный первый входной паттерн вычисляется как

$$\bar{X}^1 = y^1 W^T; \quad y^1 = [y_1^1 \ y_2^1].$$

Тогда

$$\begin{bmatrix} \bar{x}_1^1 \\ \bar{x}_2^1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix}.$$

Аналогично второй восстановленный образ:

$$\begin{bmatrix} -2 \\ x_1 \\ -2 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \sqrt{2} \\ 0 \\ \sqrt{2} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0,5 \\ -0,5 \end{bmatrix}.$$

Поскольку  $\lambda_2 = 0$ , то без потери информации можно выбрать раз мерность сжатого пространства  $p = 1$ . Тогда для сжатия и восстановления информации используется только первый собственный вектор:

$$W = W_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}.$$

В таком случае формула, описывающая сжатую информацию для первого и второго входного образа (первые главные компоненты), имеет вид

$$y_1^1 = x^1 W = -\frac{1}{\sqrt{2}}, \quad y_1^2 = x^2 W = \frac{1}{\sqrt{2}}.$$

Найдем восстановленную информацию:

$$\begin{bmatrix} -1 \\ x_1 \\ -1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix},$$

$$\begin{bmatrix} -2 \\ x_1 \\ -2 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0,5 \\ -0,5 \end{bmatrix}.$$

Исходя из последних выражений можно сделать вывод, что информация восстановилась без потерь. В заключение этого раздела рассмотрим геометрическую интерпретацию метода главных компонент (рис. 6.4).

Геометрический смысл метода главных компонент состоит в проецировании входных образов на координатные оси  $W_1$  и  $W_2$ . Ось первой главной компоненты  $W_1$  соответствует максимальному разбросу входных данных и для рассматриваемого примера проходит через точки  $x^1$

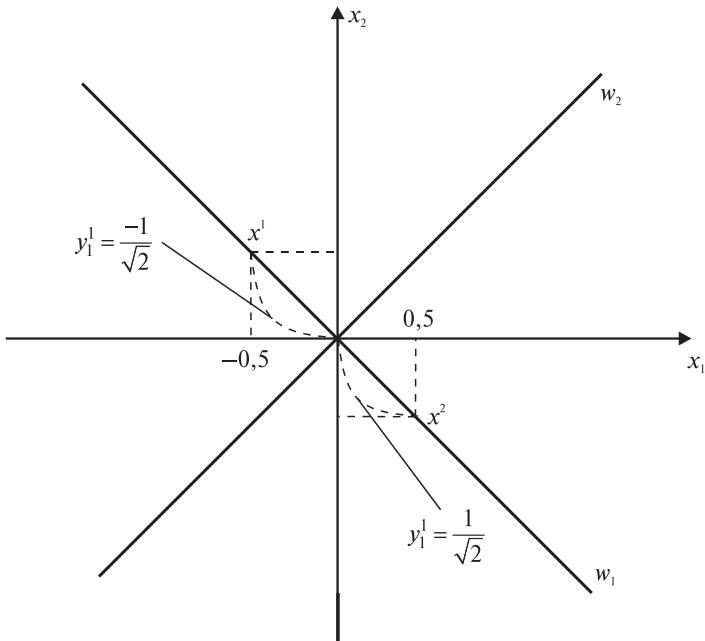


Рис. 6.4. Геометрическая интерпретация метода главных компонент

и \$x^2\$. Ось второй главной компоненты \$W\_2\$ проводится перпендикулярно оси первой главной компоненты. Поэтому из рис. 6.4 можно найти первый и второй собственный вектор:

$$w_{11} = \frac{1}{\sqrt{2}}, \quad w_{21} = -\frac{1}{\sqrt{2}},$$

$$w_{12} = \frac{1}{\sqrt{2}}, \quad w_{22} = -\frac{1}{\sqrt{2}}.$$

Главные компоненты представляются как проекция входных образов на координатные оси, соответствующие собственным векторам ковариационных матриц. Согласно рис. 6.4 получим

$$y_1^1 = -\frac{1}{\sqrt{2}}, \quad y_1^2 = \frac{1}{\sqrt{2}}, \quad y_2^1 = 0, \quad y_2^2 = 0.$$

Используя выражение (6.7), можно определить также собственные числа ковариационной матрицы.

## 6.2. АРХИТЕКТУРА АВТОЭНКОДЕРНОЙ НЕЙРОННОЙ СЕТИ

Автоэнкодерные нейронные сети позволяют осуществить как линейное, так и нелинейное преобразование данных [27, 29]. Рассмотрим простую автоэнкодерную сеть. Существует два представления автоэнкодерной нейронной сети [27].

1. Автоэнкодерная нейронная сеть представляет собой совокупность двух слоев нейронных элементов, которые соединены между собой двунаправленными связями (рис. 6.5).

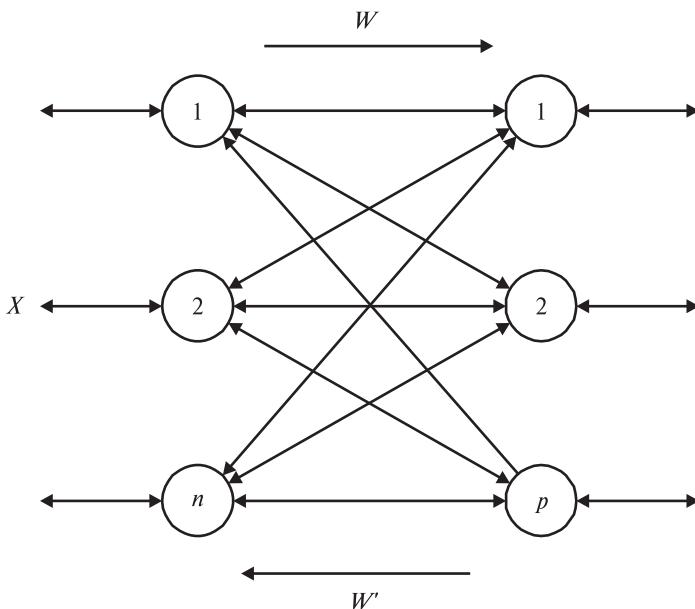


Рис. 6.5. Архитектура автоэнкодерной нейронной сети

Каждый из слоев нейронных элементов может использоваться в качестве входного или выходного. Если слой нейронных элементов служит в качестве входного, то он выполняет распределительные функции. В противном случае нейронные элементы слоя являются обрабатывающими. Весовые коэффициенты, соответствующие прямым и обратным связям, характеризуются матрицей весовых коэффициентов  $W$  и  $W'$ . Данное представление рециркуляционной нейронной сети эквивалентно ограниченной машине Больцмана (restricted Boltzmann machine (RBM)) [29].

2. Развёрнутое представление автоэнкодерной нейронной сети. В этом случае автоэнкодерная сеть представляет собой персептрон с одним скрытым слоем (рис. 6.6).

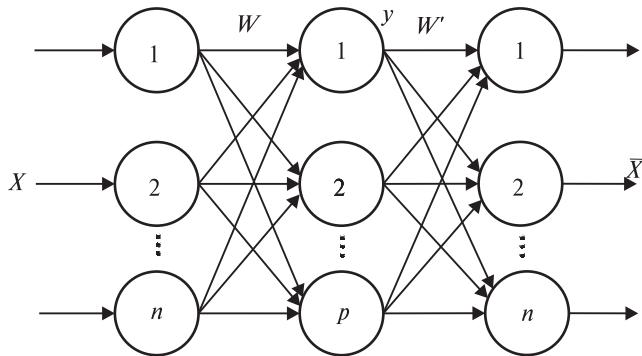


Рис. 6.6. Эквивалентное представление автоэнкодерной сети

Такое представление сети является эквивалентным и характеризует полный цикл преобразования информации. При этом скрытый слой (bottleneck) нейронных элементов производит кодирование (сжатие) входных данных  $X$ , а последний слой осуществляет восстановление сжатой информации  $Y$ . Назовем слой нейронной сети, матрица связи которой  $W$ , прямым (сжимающим), а слой, соответствующий матрице связей  $W'$  — обратным (восстанавливающим).

Рециркуляционная сеть предназначена как для сжатия данных, так и для восстановления сжатой информации. Сжатие данных осуществляется при прямом преобразовании информации согласно выражению

$$Y = F(XW). \quad (6.14)$$

Восстановление или реконструкция данных происходит при обратном преобразовании информации:

$$\bar{X} = F(YW'). \quad (6.15)$$

В качестве функции активации нейронных элементов  $F$  может применяться как линейная, так и нелинейная функция. При использовании линейной функции активации

$$Y = XW, \quad (6.16)$$

$$\bar{X} = YW'. \quad (6.17)$$

Такие сети называются линейными. В предыдущем разделе отмечалось, что наилучший автопрогноз достигается, когда матрица весовых коэффициентов определяется методом главных компонент. При этом

строки матрицы  $W'$  равняются собственным векторам ковариационной матрицы. Тогда  $W' \equiv W^T$ .

Таким образом, весовые коэффициенты линейной рециркуляционной нейронной сети можно определить при помощи метода главных компонент. В этом случае матрица  $W$  ортогональна и  $WW^T = I$ .

Линейные автоэнкодерные сети, в которых весовые коэффициенты определяются согласно методу главных компонент, называются РСА-сетями. Существуют следующие методы обучения автоэнкодерных нейронных сетей: правило обучения Ойя, кумулятивное дельта-правило, алгоритм обратного распространения ошибки и метод послойного обучения. Правило обучения Ойя автоэнкодерной нейронной сети приводит к тем же результатам, что и метод главных компонент, т. е. весовые векторы (собственные векторы) являются ортогональными. Остальные методы в общем случае не приводят к тем же результатам, что и метод главных компонент. Рассмотрим методы обучения автоэнкодерных нейронных сетей.

### 6.3. ПРАВИЛО ОБУЧЕНИЯ ОЙЯ

Данное правило обучения было предложено в 1982 г. финским ученым Е. Ойя (Erkki Oja) [47]. Для его вывода Ойя использовал модификацию правила Хебба. Покажем, что данное правило обучения можно легко получить на основе классического метода градиентного спуска [27, 51].

Автоэнкодерные нейронные сети должны обеспечивать такое преобразование информации, чтобы достигалась минимальная суммарная квадратичная ошибка между входными и реконструированными образами:

$$E_S = \frac{1}{2} \sum_{k=1}^L \sum_{i=1}^n (\bar{x}_i^k - x_i^k)^2, \quad (6.18)$$

где  $\bar{x}_i^k$  –  $i$ -я компонента  $k$ -го выходного образа сети;  $x_i^k$  –  $i$ -я компонента  $k$ -го входного образа сети;  $L$  – общее количество образов.

Для минимизации суммарной квадратичной ошибки сети будем использовать метод градиентного спуска для последовательного обучения. Тогда

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha \frac{\partial E}{\partial w'_{ji}(t)}.$$

Здесь  $E$  – квадратичная ошибка сети для одного образа, которая вычисляется как

$$E = \frac{1}{2} \sum_{i=1}^n (\bar{x}_i - x_i)^2.$$

В случае использования линейной функции активации нейронных элементов выходное значение  $j$ -го нейрона скрытого слоя определяется следующим образом:

$$y_j = S_j = \sum_{i=1}^n w_{ij} x_i.$$

Аналогично для  $i$ -го нейрона выходного слоя:

$$\bar{x}_i = S_i = \sum_{j=1}^p w'_{ji} y_j.$$

Тогда

$$\frac{\partial E}{\partial w'_{ji}} = \frac{\partial E}{\partial \bar{x}_i} \frac{\partial S_i}{\partial w'_{ji}} = (\bar{x}_i - x_i) y_j = -(x_i - \bar{x}_i) y_j = -y_j \left( x_i - \sum_{j=1}^p w'_{ji} y_j \right).$$

В результате правила модификации весовых коэффициентов восстанавливающего слоя можно представить как

$$w'_{ji}(t+1) = w'_{ji} + \alpha y_j \left( x_i - \sum_{j=1}^p w'_{ji} y_j \right). \quad (6.19)$$

Поскольку в соответствии с методом главных компонент матрица весовых коэффициентов сжимающего и восстанавливающего слоя автоэнкодерной сети связана операцией транспонирования

$$W = (W')^T,$$

то весовые коэффициенты нейронных элементов сжимающего слоя будут изменяться в соответствии со следующим выражением:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha y_j \left( x_i - \sum_{j=1}^p w_{ij} y_j \right). \quad (6.20)$$

Приведенные выше выражения (6.19) и (6.20) называются правилом обучения Ояя. Последнее слагаемое в выражении (6.20) препятствует

неограниченному возрастанию весовых коэффициентов. Теоретически доказано, что Ойя-правило эквивалентно методу главных компонент.

Проведем анализ изменения весовых коэффициентов нейронной сети, которая обучается согласно выражению (6.20). Для  $j$ -го нейронного элемента весовые коэффициенты изменяются с течением времени:

$$\Delta W_j = \alpha y_j (X - WY),$$

где

$$X = [x_1, x_2, \dots, x_n]^T,$$

$$Y = [y_1, y_2, \dots, y_n]^T,$$

$$\Delta W_j = [\Delta w_{1j}, \Delta w_{2j}, \dots, \Delta w_{nj}]^T,$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1p} \\ w_{21} & w_{22} & \dots & w_{2p} \\ \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & \dots & w_{np} \end{bmatrix}.$$

Рассмотрим весовые коэффициенты нейронной сети в состоянии равновесия, т. е.

$$\Delta W_j = \alpha y_j (X - WY) = 0.$$

Умножим обе части последнего выражения на  $W^T$ . Тогда получим

$$W^T X - W^T W Y = 0.$$

Поскольку  $W^T X = Y$ , то

$$W^T W = I,$$

где  $I$  – единичная матрица.

Как известно, последнее равенство определяет необходимое и достаточное условие ортогональности матрицы  $W$ . Отсюда следует, что столбцы матрицы  $W$  являются ортонормированными, т. е.

$$\sum_{i=1}^n w_{ij} w_{ik} = \begin{cases} 1, & j = k, \\ 0, & j \neq k \end{cases}$$

для любых  $j, k$ .

Таким образом, правило обучения Ойя эквивалентно методу главных компонент. Рассмотрим правило обучения Ойя для нелинейной

рециркуляционной сети. В этом случае выходные значения сжимающего и восстанавливающего слоя определяются следующим образом:

$$y_j = F(S_j) = F\left(\sum_{i=1}^n w_{ij}x_i\right),$$

$$\bar{x}_i = F(S_i) = F\left(\sum_{j=1}^p w'_{ji}y_j\right).$$

Тогда правило обучения Ойя для нелинейной сети имеет вид

$$w_{ij}(t+1) = w_{ij}(t) + \alpha y_j \left( x_i - F\left(\sum_{j=1}^p w_{ij}y_j\right) \right). \quad (6.21)$$

Как следует из последнего выражения, правило обучения Ойя для нелинейной сети записывается в том же виде, что и для линейной сети, где в качестве выходных значений сжимающего и восстанавливающего слоя используется нелинейная функция активации.

Рассмотрим соответствие нейронов сжимающего слоя главным компонентам. Существуют следующие способы определения соответствия нейронного элемента конкретной главной компоненте. Для того чтобы первый нейрон сжимающего слоя соответствовал первой главной компоненте, а второй нейронный элемент — второй главной компоненте и т. д., необходимо вначале обучать только первый нейрон сжимающего слоя, затем второй нейрон и т. д. В этом случае правило обучения можно представить как

$$w_{ij}(t+1) = w_{ij}(t) + \alpha y_j \left( x_i - \sum_{k=1}^j w_{ik}y_k \right).$$

Второй способ состоит в том, что вначале нейронная сеть обучается в соответствии с правилом Ойя (6.21). После обучения необходимо вычислить дисперсию выходных значений сжимающего слоя:

$$\lambda_j = \sigma(y_j) = \frac{1}{L} \sum_{k=1}^L (y_j^k)^2, \quad j = \overline{1, p}.$$

Нейронный элемент сжимающего слоя, который имеет максимальное значение дисперсии, определяет первую главную компоненту, нейронный элемент со следующим значением дисперсии определяет вторую компоненту и т. д.

## 6.4. ОБОБЩЕННОЕ ДЕЛЬТА-ПРАВИЛО

Задачей обучения автоэнкодерной нейронной сети является минимизация суммарной среднеквадратичной ошибки между входными и реконструированными образами. Как уже отмечалось, квадратичная ошибка сети для одного образа вычисляется как

$$E = \frac{1}{2} \sum_{i=1}^n (\bar{x}_i - x_i)^2.$$

В таком случае согласно методу градиентного спуска

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \frac{\partial E}{\partial w_{ij}(t)},$$

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha \frac{\partial E}{\partial w'_{ji}(t)}.$$

Рассмотрим правило модификации весовых коэффициентов нейронной сети в случае использования произвольной функции активации нейронных элементов. В таком случае взвешенная сумма и выходное значение  $j$ -го нейрона скрытого слоя определяются следующим образом:

$$S_j = \sum_{i=1}^n w_{ij} x_i,$$

$$y_j = F(S_j).$$

Аналогично  $i$ -й нейрон выходного слоя:

$$S_i = \sum_{j=1}^p w'_{ji} y_j,$$

$$\bar{x}_i = F(S_i).$$

Тогда

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial \bar{x}_i} \frac{\partial \bar{x}_i}{\partial S_i} \frac{\partial S_i}{\partial w'_{ji}} = (\bar{x}_i - x_i) F'(S_i) y_j,$$

$$\frac{\partial E}{\partial w_{ij}} = \sum_{i=1}^n \frac{\partial E}{\partial \bar{x}_i} \frac{\partial \bar{x}_i}{\partial S_i} \frac{\partial S_i}{\partial y_j} \frac{\partial y_j}{\partial S_j} \frac{\partial S_j}{\partial w_{ij}} = F'(S_j) x_i \gamma_j.$$

Здесь  $\gamma_j$  – ошибка  $j$ -го нейрона скрытого слоя. Она вычисляется как

$$\gamma_j = \sum_{i=1}^n (\bar{x}_i - x_i) F'(S_i) w'_{ji}.$$

В соответствии с этим правило модификации весовых коэффициентов прямого и обратного слоя можно представить как

$$w_{ij}(t+1) = w_{ij}(t) - \alpha F'(S_j) x_i \gamma_j, \quad (6.22)$$

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha F'(S_i) y_j (\bar{x}_i - x_i). \quad (6.23)$$

Подставляя в (6.22) и (6.23) производные используемой функции активации  $F'(S_j)$  и  $F'(S_i)$ , можно получить конкретные выражения для модификации весовых коэффициентов автоэнкодерной нейронной сети.

Как уже отмечалось, алгоритм обратного распространения ошибки в общем случае не приводит к тем же результатам, что и метод главных компонент, т. е. весовые векторы нейронов скрытого слоя после обучения не являются ортогональными. Для обеспечения ортогональности соответствующих весовых векторов весовые коэффициенты скрытого слоя в процессе обучения ортонормируются согласно процедуре Грамма – Шмидта [52]:

а) в качестве первого вектора ортонормированного базиса выберем

$$w'_1 = \left[ \frac{w_{11}}{|w_1|}, \frac{w_{21}}{|w_1|}, \dots, \frac{w_{n1}}{|w_1|} \right],$$

где  $|w_1| = \sqrt{w_{11}^2 + w_{21}^2 + \dots + w_{n1}^2}$ ;

б) остальные весовые векторы определяются рекурсивным образом по следующим выражениям:

$$w_i = w_i - \sum_{j=1}^{i-1} (w_i^T w'_j) w'_j,$$

$$w'_j = \left[ \frac{w_{1j}}{|w_j|}, \frac{w_{2j}}{|w_j|}, \dots, \frac{w_{nj}}{|w_j|} \right],$$

$$|w_j| = \sqrt{w_{1j}^2 + w_{2j}^2 + \dots + w_{nj}^2},$$

где  $i = 2, p$ .

## 6.5. КУМУЛЯТИВНОЕ ДЕЛЬТА-ПРАВИЛО

Кумулятивное дельта-правило базируется на раздельной минимизации суммарной квадратичной ошибки сети в восстанавливающем и сжимающем слоях [27, 53]. Рассмотрим применение данного правила для обучения нелинейных автоэнкодерных нейронных сетей.

В процессе обучения автоэнкодерной сети для каждого входного образа производится три цикла преобразования информации: прямое (сжатие), обратное (восстановление) и прямое (сжатие). Такое преобразование информации эквивалентно сэмплированию Гиббса с единичным шагом. После этого производится настройка весовых коэффициентов сети. Для наглядности процесса распространения информации введем обозначения. Пусть  $x_i(0)$ ,  $i = 1, n$ , — входной вектор, поступающий на вход сети в начальный момент времени  $t = 0$ . Тогда формулы выходных значений сжимающего слоя можно записать как

$$y_j(0) = F(S_j) = F\left(\sum_{i=1}^n w_{ij}x_i(0)\right), \quad (6.24)$$

где  $j = \overline{1, p}$ .

Выходные значения восстанавливающего слоя в момент времени  $t = 1$ :

$$x_i(1) = F(S_j) = F\left(\sum_{j=1}^p w'_{ji}y_j(0)\right), \quad (6.25)$$

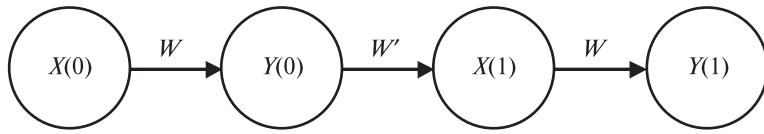
где  $i = \overline{1, n}$ .

На следующем этапе распространения информации выходной вектор восстанавливающего слоя подается на вход сети и опять определяются выходные значения сжимающего слоя:

$$y_j(1) = F(S_j(1)) = F\left(\sum_{i=1}^n w_{ij}x_i(1)\right), \quad (6.26)$$

где  $j = \overline{1, p}$ .

Такое преобразование информации можно представить в виде цепочки, изображенной на рис. 6.7.



*Рис. 6.7.* Последовательное преобразование информации в автоэнкодерной сети

Тогда ошибка восстановления информации в последнем слое нейронной сети определяется как

$$E_v = \frac{1}{2} \sum_i (x_i(1) - x_i(0))^2. \quad (6.27)$$

Ошибку воспроизведения информации в сжимающем слое нейронной сети можно представить следующим образом:

$$E_h = \frac{1}{2} \sum_j (y_j(1) - y_j(0))^2. \quad (6.28)$$

Обучение автоэнкодерной нейронной сети производится в целях минимизации суммарной квадратичной ошибки восстановления, а также сжатия информации. При этом значение  $y_j(0)$  в выражении (6.28) принимается за эталонное. Тогда при последовательном обучении согласно методу градиентного спуска в пространстве весовых коэффициентов получим

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \frac{\partial E_h}{\partial w_{ij}(t)}, \quad (6.29)$$

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha \frac{\partial E_v}{\partial w'_{ji}(t)}. \quad (6.30)$$

Найдем соответствующие производные:

$$\frac{\partial E_h}{\partial w_{ij}(t)} = (y_j(1) - y_j(0)) F'(S_j(1)) x_i(1), \quad (6.31)$$

$$\frac{\partial E_v}{\partial w'_{ji}(t)} = (x_i(1) - x_i(0)) F'(S_i(1)) y_j(0). \quad (6.32)$$

В результате выражения для настройки весовых коэффициентов автоэнкодерной нейронной сети примут следующий вид:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha x_i(1) F' \left( S_j(1) (y_j(1) - y_j(0)) \right), \quad (6.33)$$

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha y_j(0) F' \left( S_i(1) (x_i(1) - x_i(0)) \right). \quad (6.34).$$

Как уже отмечалось, в процессе обучения рециркуляционной сети для каждого входного образа происходит три цикла распространения информации. После этого выполняется модификация весовых коэффициентов сети. Процедура обучения осуществляется до тех пор, пока суммарная квадратичная ошибка сети не станет меньше заданной.

В начальный момент времени производится случайная инициализация весовых коэффициентов. При этом желательно обеспечивать симметричность весовых коэффициентов прямого и обратного слоя ( $w_{ij} = w'_{ji}$ ).

## 6.6. МЕТОД ПОСЛОЙНОГО ОБУЧЕНИЯ

Метод послойного обучения характеризуется последовательной модификацией весовых коэффициентов различных слоев нейронной сети и состоит из двух отдельных фаз обучения [27, 51]. На первом этапе определяется весовая матрица  $W' = [w'_{ji}]$ ,  $j = \overline{1, p}$ ,  $i = \overline{1, n}$ , и матрица выходных значений скрытого слоя  $\bar{y} = [\bar{y}_j^k]$ ,  $j = \overline{1, p}$ ,  $k = \overline{1, L}$ , для осуществления наилучшего автопрогноза. Для этого необходимо минимизировать суммарную квадратичную ошибку между восстановленными данными  $\bar{X}$  и входными данными  $X$ :

$$E_S = \frac{1}{2} \sum_{k=1}^L \sum_{i=1}^n (x_i^k - \bar{x}_i^k)^2, \quad (6.35)$$

где  $L$  – общее количество тренировочных образцов.

На втором этапе алгоритма находится весовая матрица прямого слоя  $W = [w_{ij}]$ ,  $i = \overline{1, n}$ ,  $j = \overline{1, p}$ . При этом в качестве эталонных выходов используются значения  $\bar{y}$ , полученные на предыдущем этапе алгоритма. Тогда обучение на втором этапе происходит в целях минимизации следующего выражения:

$$E'_S = \frac{1}{2} \sum_{k=1}^L \sum_{j=1}^p (y_i^k - \bar{y}_i^k)^2. \quad (6.36)$$

Рассмотрим теоретические основы метода послойного обучения.

### 6.6.1. Теоретические основы метода

Определим правила обучения в случае применения в слоях рециркуляционной сети нелинейной функции активации. Тогда на первом этапе алгоритма необходимо найти весовые коэффициенты обратного слоя  $W'$ , а также для каждого входного вектора  $X$  определить такой вектор  $Y$ , чтобы обеспечить минимизацию суммарной среднеквадратичной ошибки (6.35).

Рассмотрим определение матрицы выходных значений скрытого слоя  $\bar{Y}$  для минимизации выражения (6.35). В соответствии с методом градиентного спуска компоненты вектора  $Y$  должны изменяться с течением времени:

$$y_j(t+1) = y_j(t) - \alpha \frac{\partial E}{\partial y_j(t)}. \quad (6.37)$$

Ошибка  $E$  определяется следующим образом:

$$E = \frac{1}{2} \sum_{i=1}^n (\bar{x}_i - x_i)^2, \quad (6.38)$$

где

$$\bar{x}_i = F(S_i) = F\left(\sum_{j=1}^p w'_{ji} y_j\right). \quad (6.39)$$

Дифференцируя выражение (6.38) по переменной  $y_j$ , получим

$$\gamma_j = \frac{\partial E}{\partial y_j} = \sum_i \frac{\partial E}{\partial \bar{x}_i} \frac{\partial \bar{x}_i}{\partial S_i} \frac{\partial S_i}{\partial y_j} = \sum_{i=1}^n (\bar{x}_i - x_i) F'(S_i) w'_{ji}. \quad (6.40)$$

Тогда

$$y_j(t+1) = y_j(t) - \alpha_1 \gamma_j. \quad (6.41)$$

Найдем выражение для настройки весовых коэффициентов  $w'_{ji}$  обратного слоя согласно методу градиентного спуска:

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha_2 \frac{\partial E}{\partial w'_{ji}(t)}. \quad (6.42)$$

Производная среднеквадратичной ошибки вычисляется как

$$\frac{\partial E}{\partial w'_{ji}} = \frac{\partial E}{\partial \bar{x}_i} \frac{\partial \bar{x}_i}{\partial S_i} \frac{\partial S_i}{\partial w'_{ji}} = (\bar{x}_i - x_i) F'(S_i) y_j. \quad (6.43)$$

Отсюда следует, что модификация весовых коэффициентов обратного слоя должна производиться следующим образом:

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha_2 (\bar{x}_i - x_i) F'(S_i) y_j. \quad (6.44)$$

В результате при выполнении первого этапа алгоритма вычисляются весовые коэффициенты восстанавливающего слоя  $w'_{ji}$  и эталонных выходов  $\bar{y}_j^k$ ,  $j = \overline{1, p}$ ,  $k = \overline{1, L}$ , скрытого слоя. Необходимо отметить, что значения эталонных выходов скрытого слоя  $\bar{y}_j^k$  могут не принадлежать диапазону значений, определяемых функцией активации нейронных элементов. Поэтому необходимо проводить нормализацию эталонных выходов согласно используемой функции активации.

На втором этапе метода послойного обучения рассчитываются весовые коэффициенты  $w_{ij}$  сжимающего слоя, где в качестве эталонных выходов  $\bar{y}_j^k$  берутся значения, полученные на предыдущем этапе. Для этого необходимо минимизировать выражение (6.36):

$$w_{ij}(t+1) = w_{ij}(t) - \alpha_3 \frac{\partial E'}{\partial w_{ij}(t)}, \quad (6.45)$$

где

$$E' = \frac{1}{2} \sum_{j=1}^p (y_j - \bar{y}_j)^2. \quad (6.46)$$

Определим производную среднеквадратичной ошибки:

$$\frac{\partial E'}{\partial w_{ij}} = \frac{\partial E'}{\partial y_j} \frac{\partial y_j}{\partial S_j} \frac{\partial S_j}{\partial w_{ij}} = (y_j - \bar{y}_j) F'(S_j) x_i, \quad (6.47)$$

где  $S_j$  – взвешенная сумма  $j$ -го нейрона скрытого слоя –

$$S_j = \sum_i w_{ij} x_i. \quad (6.48)$$

Тогда

$$w_{ij}(t+1) = w_{ij}(t) - \alpha_3 (y_j - \bar{y}_j) F'(S_j) x_i. \quad (6.49)$$

Подставляя в выражения (6.47), (6.49) значения  $F'(S_j)$ , соответствующие используемой функции активации, можно получить конкретные правила обучения для рециркуляционных сетей.

### 6.6.2. Алгоритм послойного обучения

Алгоритм послойного обучения состоит из следующих шагов [27, 51]:

1. Случайно инициализируются весовые коэффициенты и задается желаемая суммарная квадратичная ошибка сети  $E_e$ .
2. Последовательно подаются  $L$  образов на нейронную сеть. При этом для каждого образа происходит только прямое распространение информации. В результате выполнения данного этапа определяется базовый набор сжатых образов  $Y$ , которые будут использоваться на следующем этапе алгоритма.

3. Рассматривается только обратный слой сети. В качестве входной информации берутся значения  $\bar{Y}$ , полученные на предыдущем шаге алгоритма. В качестве эталонных значений выходов обратного слоя принимается вектор исходных данных  $X$ . При этом производится следующая последовательность действий:

- 3.1. Для  $L$  входных образов производится настройка весовых коэффициентов обратного слоя  $W'$  по выражению (6.44).
- 3.2. Для  $L$  входных образов настраиваются желаемые выходы прямого слоя  $\bar{Y}$  согласно выражению (6.41).
- 3.3. Пункты 3.1 и 3.2 повторяются, пока суммарная квадратичная ошибка обратного слоя не станет меньше заданной  $E_e$ .
4. Модифицируются весовые коэффициенты прямого слоя в соответствии с выражением (6.49). Для этого входные образы последовательно подаются на сеть и для каждого образа происходит только прямое распространение (сжатие) информации. В качестве эталонных данных применяются значения  $\bar{Y}$ , полученные на шаге 3 алгоритма.
5. Пункт 4 продолжается до тех пор, пока суммарная квадратичная ошибка прямого слоя не станет меньше заданной.

Существуют также другие варианты приведенного выше алгоритма. Например, после выполнения шага 5 можно перейти к шагу 2 и повторять алгоритм до тех пор, пока не будет достигнуто лучшее решение с точки зрения суммарной квадратичной ошибки сети и обобщающей способности. Также на заключительном этапе можно применить алгоритм обратного распространения ошибки ко всей сети для более точной настройки параметров сети.

## 6.7. АНАЛИЗ АВТОЭНКОДЕРНЫХ НЕЙРОННЫХ СЕТЕЙ

В разделах 6.2–6.6 рассматривалась автоэнкодерная нейронная сеть с одним скрытым слоем. При применении линейной функции активации нейронных элементов такая сеть формирует линейные оси

главных компонент, что не подходит для многих практических приложений. При использовании нелинейных нейронных элементов автоэнкодер с одним скрытым слоем слабо формирует нелинейные оси главных компонент, что не позволяет адекватно отобразить нелинейное распределение входных данных. Так, например, если входные данные распределены в виде подковы (рис. 6.8), то ось первой главной компоненты для отображения максимального количества данных должна быть нелинейной.

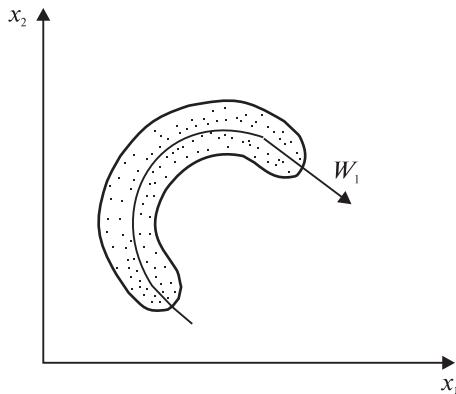


Рис. 6.8. Нелинейная ось первой главной компоненты

Для осуществления такого отображения необходимо увеличить количество скрытых слоев сети. Рассмотрим автоэнкодерную нейронную сеть с пятью слоями нейронных элементов (рис. 6.9).

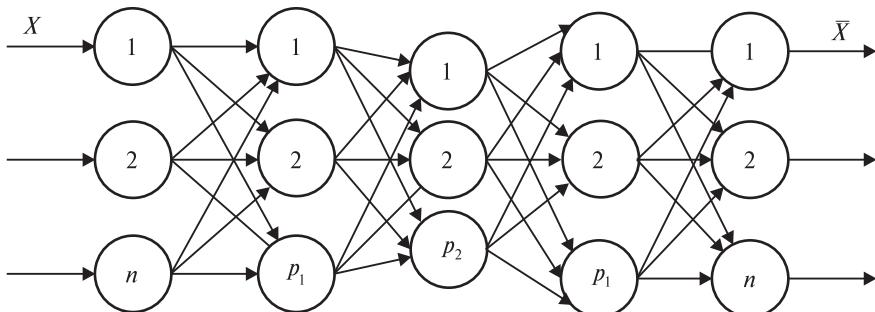


Рис. 6.9. Автоэнкодер с тремя скрытыми слоями

Здесь следует отметить, что количество нейронных элементов  $p_1$  в первом скрытом слое не обязательно должно быть меньше количества входных нейронов  $n$  ( $p_1 < n$ ). Рассмотрим теорему Ковера (T. Cover) о разделимости образов [41].

**Теорема 6.2.** При нелинейном отображении входных данных в пространство большей размерности увеличивается вероятность их линейной разделимости.

Поэтому в зависимости от решаемой задачи количество нейронных элементов  $p_1$  в первом скрытом слое может быть как больше, так и меньше количества входных нейронов  $n$ .

**Теорема 6.3.** Автоэнкодер с тремя скрытыми слоями является универсальным аппроксиматором для сжатия и восстановления информации. Поэтому для эффективной нелинейной обработки информации минимальная конфигурация автоэнкодерной нейронной сети должна содержать пять слоев нейронных элементов, где первые три слоя используются для сжатия, а последние два слоя – для восстановления информации.

*Доказательство.* Согласно формулировке теоремы минимальная конфигурация автоэнкодерной нейронной сети (NPCA) содержит три скрытых слоя нейронных элементов и состоит из двух персепtronов: сжимающего ( $MLP_e$ ) и восстанавливающего ( $MLP_r$ ), т. е.

$$NPCA = MLP_e + MLP_r.$$

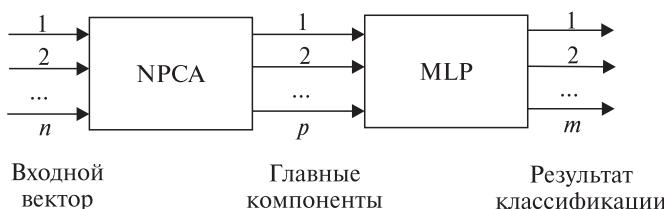
При этом каждый из персепtronов имеет один скрытый слой, т. е. является универсальным аппроксиматором при использовании сигмоидальной функции активации нейронных элементов. В результате обеспечивается универсальная аппроксимация при сжатии и восстановлении информации.

## 6.8. ПРИМЕНЕНИЕ АВТОЭНКОДЕРНЫХ НЕЙРОННЫХ СЕТЕЙ

Как уже отмечалось, автоэнкодерная нейронная сеть может применяться для решения задач классификации и визуализации образов, сжатия и восстановления изображений и т. д. В данном разделе рассмотрим применение автоэнкодерных сетей для предварительной обработки данных, обнаружения аномалий, классификации и визуализации данных, сжатия изображений и разделения сигналов.

### 6.8.1. Предварительная обработка данных

Предварительная обработка данных применяется для выделения наиболее информативных признаков (feature selection) из исходного пространства образов и сокращения размерности входных данных. Для распознавания образов можно использовать схему, состоящую из автоэнкодера (feature selection) и многослойного персептрона в качестве классификатора (рис. 6.10).



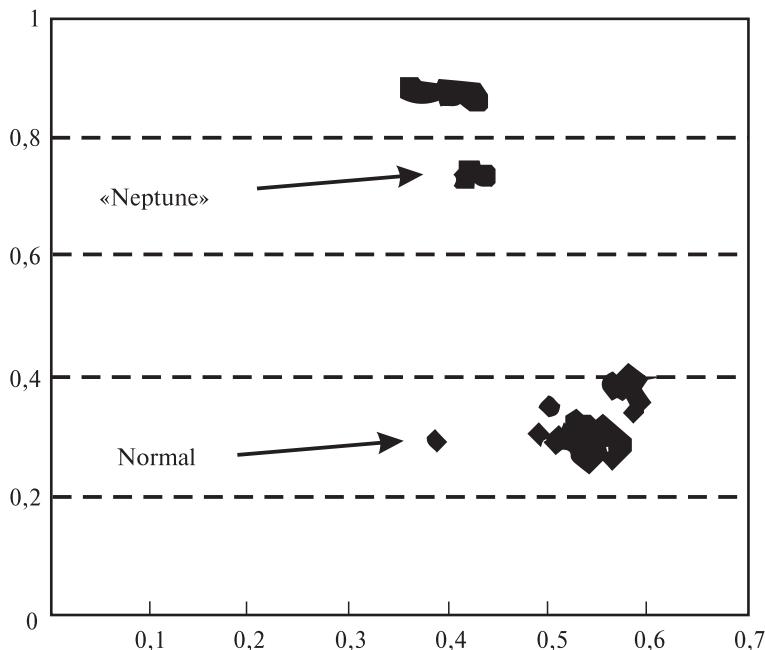
*Рис. 6.10. Комбинация двух нейронных сетей для распознавания образов*

В этом случае вначале обучается автоэнкодер для сжатия данных и выделения наиболее информативных признаков из входного пространства образов, а затем с использованием сжатых данных обучается многослойный персептрон классификации образов.

### 6.8.2. Классификация и визуализация данных

Для визуализации и классификации входного пространства данных необходимо отобразить их на плоскость двух или трех главных компонент [54]. В результате данные, принадлежащие одному классу, будут группироваться в одной области пространства главных компонент. Рассмотрим, например, отображение входного пространства образов для нормального состояния и атаки (тип атаки «Neptune») на плоскость двух первых главных компонент (рис. 6.11).

Как следует из рис. 6.11, данные, соответствующие классу атаки и нормальным соединениям, концентрируются в различных областях. Для проведения эксперимента использовалась база данных о сетевых соединениях KDD-99 (<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>). Каждая запись в базе соответствует одному TCP/IP-соединению и представляет собой характеристический 41-размерный вектор, который описы-



*Рис. 6.11. Отображение нормальных соединений и атаки на плоскость главных компонент*

вает конкретный тип атаки или нормальное состояние. Для эксперимента применялся нелинейный автоэнкодер с архитектурой 41–2–41 и сигмоидной функцией активации нейронных элементов [54].

### 6.8.3. Обнаружение аномалий

Аномалией называется любое отклонение от нормы или типового поведения, например компьютерные атаки, вирусы, та или иная болезнь и т. д. Автоэнкодерные нейронные сети могут использоваться для обнаружения аномальных образов, т. е. в качестве детектора аномалий [55]. Это базируется на таком свойстве сетей, как умение восстанавливать на выходе входную информацию. Для создания детектора аномалий необходимо составить обучающую выборку, состоящую из нормальных данных, и обучить автоэнкодер воспроизводить на выходе нормальные векторы. Тогда если после обучения подать на вход сети нормальный образ, то ошибка реконструкции информации для него будет меньше, чем для аномального образа. В результате чем

больше входной образ похож на нормальный, тем меньше ошибка реконструкции:

$$E(k) = \sum_j (\bar{x}_j^k - x_j^k)^2,$$

где  $x_j^k$  —  $j$ -й элемент  $k$ -го входного вектора;  $\bar{x}_j^k$  —  $j$ -й элемент  $k$ -го выходного вектора.

Если  $E(k) > T$ , где  $T$  — порог идентификации аномального образа, то входной образ считается аномальным, иначе — нормальным образом.

Отдельный автоэнкодер может применяться для определения принадлежности входного вектора  $X^k$  к одному из двух классов: или к тому, на котором он обучался (класс  $A$ ), или к тому (класс  $\bar{A}$ ), которому соответствуют далеко отстающие векторы:

$$\begin{cases} X^k \in A, \text{ если } E(k) \leq T, \\ X^k \in \bar{A}, \text{ если } E(k) > T. \end{cases}$$

Достоинством такого подхода является использование для обучения только нормальных образов, так как не всегда легко подготовить для обучения аномальные образы.

#### 6.8.4. Разделение гауссовских сигналов

Разделение гауссовских сигналов позволяет выделить из смеси сигналов исходные независимые источники сигналов. Пусть имеется два независимых источника сигналов  $x_1(t)$  и  $x_2(t)$ , а также два приемника гауссовых сигналов  $S_1(t)$  и  $S_2(t)$  (рис. 6.12). Допустим, что при передаче сигналов по каналам связи происходит линейное смещивание сигналов:

$$S_1(t) = w_{11}x_1(t) + w_{21}x_2(t),$$

$$S_2(t) = w_{12}x_1(t) + w_{22}x_2(t).$$

Задача разделения сигналов состоит в том, чтобы на основе смешанных сигналов  $S_1(t)$  и  $S_2(t)$  получить исходные сигналы  $x_1(t)$  и  $x_2(t)$ . Здесь подразумевается, что источники сигналов гауссовые и статистически независимые. Поэтому задача получения исходных сигналов заключается в выделении из смеси статистически независимых сигналов, которые и будут исходными. Для этого можно использовать

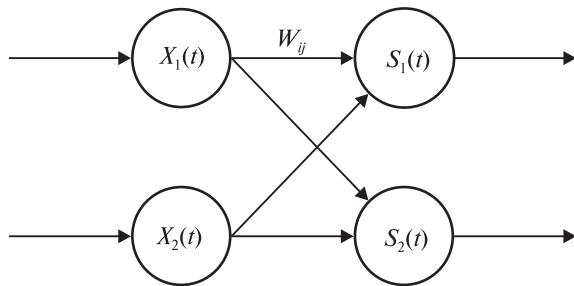


Рис. 6.12. Смешивание сигналов

автоэнкодерные нейронные сети. Теоретически это базируется на том, что главные компоненты (выходные значения скрытого слоя) некоррелированы между собой. А поскольку для гауссовых сигналов некоррелированность эквивалентна статистической независимости сигналов, то автоэнкодерные нейронные сети позволяют в скрытом слое получить исходные источники сигналов. При этом количество источников должно быть меньше или равно количеству смешанных сигналов. В случае если источники сигналов не являются гауссовскими сигналами, то для их разделения применяется метод независимых компонент (independent component analysis (ICA)) [56].

### 6.8.5. Сжатие изображений

Для упрощения процедуры обработки изображение размерностью  $n \times n$  можно разделить на множество блоков размерностью  $c \times c$ , где  $c < n$  (рис. 6.13). При этом количество таких блоков можно найти по формуле

$$k = \left( \frac{n}{c} \right)^2.$$

Назовем блок размерностью  $p \times p$  окном, которому поставим в соответствие рециркуляционную нейронную сеть. Количество нейронов первого слоя такой сети отвечает размерности окна и равно  $p \times p$ , число нейронов второго слоя обозначим через  $r$ . Сканируя изображение при помощи окна и подавая его на нейронную сеть, можно сжать входное изображение.

Такой подход использовал Т. Сангер (T. Sanger) в 1989 г. для сжатия изображений [27]. В качестве рециркуляционной сети он применял PCA-сеть, в которой весовые коэффициенты нейронов прямого слоя со-

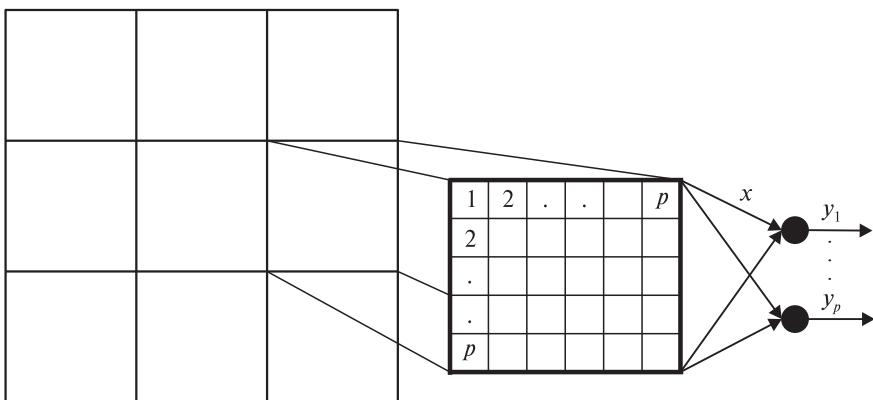


Рис. 6.13. Разбиение исходного изображения на блоки размерностью  $c \times c$

ответствовали собственным векторам ковариационной матрицы. В качестве теста использовалось изображение, представленное на рис. 6.14, а. Его размерность составляет  $256 \times 256$  пикселей, где каждый пиксель содержит 8 бит. Размерность окна равна  $8 \times 8$ . PCA-сеть состоит из 64 нейронов первого и восьми нейронов второго слоя. На рис. 6.14, б показаны весовые коэффициенты восьми нейронов второго слоя.

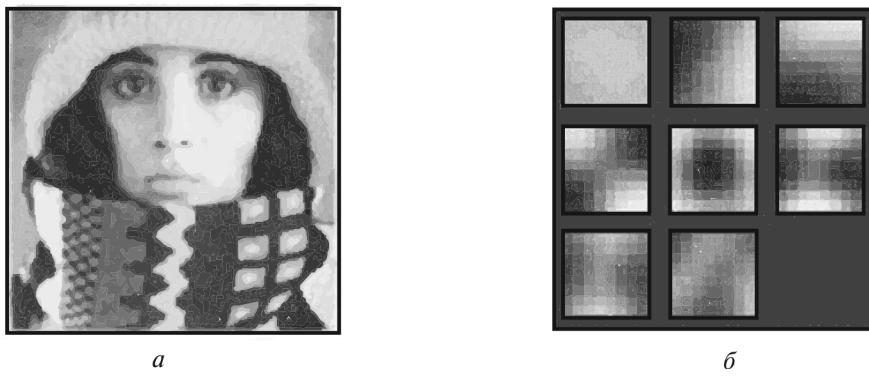


Рис. 6.14. Сжатие изображения при помощи РСА-сети:

а – исходное изображение; б – весовые коэффициенты нейронов сети

Более темным линиям на рис. 6.14, б соответствует большая величина синаптических связей. Сжатое изображение может восстанавливаться при помощи обратного распространения информации.

## Г л а в а 7

# РЕЛАКСАЦИОННЫЕ НЕЙРОННЫЕ СЕТИ

*Релаксационные* нейронные сети характеризуются прямым и обратным распространением информации между слоями нейронной сети. В основе функционирования таких сетей лежит итерационный принцип работы. Он заключается в том, что на каждой итерации процесса происходит обработка данных, полученных на предыдущем шаге. Такая циркуляция информации продолжается до тех пор, пока не установится состояние равновесия. При этом состояния нейронных элементов перестают изменяться и характеризуются стационарными значениями. Поскольку релаксация – это процесс установления равновесия, то такие сети называются релаксационными. Для анализа устойчивости релаксационных нейронных сетей используются *функции Ляпунова*. Такие сети применяются в качестве ассоциативной памяти и для решения комбинаторных задач оптимизации. К релаксационным сетям относятся: нейронные сети Хопфилда, Хэмминга; двунаправленная ассоциативная память [19, 27, 57–61]. В данной главе рассматривается архитектура и функционирование таких нейронных сетей. Для сети Хопфилда приводится механизм получения функции Ляпунова и анализ устойчивости.

### 7.1. УСТОЙЧИВОСТЬ ДИНАМИЧЕСКИХ СИСТЕМ

Для исследования устойчивости нелинейных динамических систем используется второй метод Ляпунова [62–65]. Рассмотрим динамическую систему, которая описывается нелинейными дифференциальными уравнениями:

$$\dot{X} = f(X), \quad X(0) = X_0. \quad (7.1)$$

Здесь  $\dot{X}$  и  $f(X)$  –  $n$ -мерные векторы:

$$\dot{X}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix}, \quad f(X) = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) \end{bmatrix}.$$

Совокупность величин  $x_1, x_2, \dots, x_n$  представляет собой **фазовые координаты** точки в  $n$ -мерном пространстве. С течением времени фазовые координаты будут меняться, что приведет к соответствующему перемещению точки. Такая точка называется *изображающей*, ее траектория – *фазовой траекторией*, а  $n$ -мерное пространство – *фазовым пространством*. Фазовая траектория является наглядной геометрической иллюстрацией динамического поведения системы в пространстве.

Второй метод Ляпунова [62] базируется на использовании совместно с уравнениями движения системы специальной функции – функции Ляпунова. В качестве функции Ляпунова применяется функция фазовых координат  $E(x_1, x_2, \dots, x_n)$ , обладающая определенными свойствами [63].

Назовем функцию  $E(x)$  *знакопостоянной*, если в некоторой области фазового пространства она может принимать значения только одного определенного знака или обращаться в нуль. Знакопостоянная функция называется *знакоопределенной*, если она обращается в нуль только при  $x_1 = x_2 = \dots = x_n = 0$ .

Если функция  $E(x)$  знакоопределенная, то уравнение

$$E(x) = C = \text{const}$$

характеризует замкнутую поверхность, охватывающую начало координат. При убывании функции  $E(x)$ , т. е. производная  $\frac{dE}{dt} < 0$ , фазовая траектория пересекает поверхность  $E(x) = C$  в направлении начала координат (рис. 7.1).

Это соответствует устойчивой системе. Существуют различные варианты второго метода Ляпунова [62–65]. Здесь приведен только один, рассмотренный в работе [70].

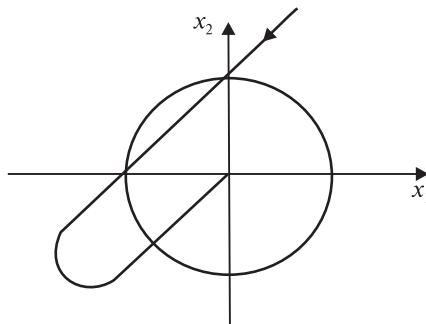


Рис. 7.1. Фазовая траектория устойчивой системы

**Теорема 7.1.** Если существует такая знакопредeterminedная функция  $E(x)$ , полная производная по времени которой в силу дифференциальных уравнений движения знакопостоянная или тождественно равняется нулю, то динамическая система является устойчивой.

Полная производная по времени функции  $E(x)$  находится следующим образом:

$$\frac{d}{dt} E(x(t)) = \sum_{i=1}^n \frac{\partial E}{\partial x_i} \frac{dx_i}{dt}. \quad (7.2)$$

Основная проблема применения метода Ляпунова заключается в определении функции  $E(x)$ . Общего аналитического метода построения функций Ляпунова не существует. Имеются только рекомендации. Так, для линейных систем функция Ляпунова строится в виде квадратичной формы координат [65]:

$$E(X, X) = \sum_i \sum_k a_{ik} x_i x_k = X^T A X, \quad (7.3)$$

где  $a_{ik} = a_{ki}$  и  $A$  – симметрическая матрица соответственно.

Если система содержит множество нелинейностей  $f_i(\sigma)$ , то функцию Ляпунова рекомендуется [65] строить в виде суммы квадратичной формы и интеграла от нелинейности:

$$E = X^T A X + \sum_i b \int_0^z f_i(\sigma) d\sigma, \quad (7.4)$$

где  $b$  – число.

Функции Ляпунова служат хорошим инструментом для исследования различных динамических систем и применяются для определения устойчивости релаксационных нейронных сетей, которые будут рассмотрены в следующих разделах.

## 7.2. НЕЙРОННАЯ СЕТЬ ХОПФИЛДА

Нейронная сеть Хопфилда представляет собой нейронную сеть с обратными связями. Функционирование таких сетей характеризуется релаксационным процессом обработки информации, который происходит до тех пор, пока не установится состояние равновесия. Многие исследователи описывали аналогичные нейронные сети, например Дж. Андерсон [16] (J. Anderson, 1977). В 1982 г. американский биофизик Дж. Хопфилд (J. Hopfield) представил математический анализ релаксационных сетей с обратными связями [19], основанный на теории изинговых спинов, которая используется для изучения ферромагнетиков при низких температурах. Поэтому данные нейронные сети и получили название сетей Хопфилда.

### 7.2.1. Архитектура нейронной сети Хопфилда

Нейронная сеть Хопфилда характеризуется обратными связями. В ней каждый нейрон имеет синаптические связи со всеми остальными нейронами сети.

Представим архитектуру такой сети в виде двух слоев нейронных элементов (рис. 7.2).

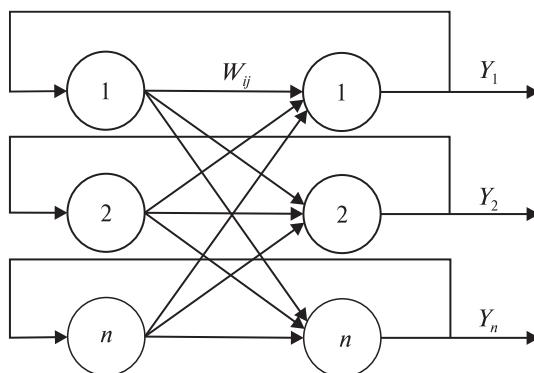


Рис. 7.2. Архитектура сети Хопфилда

При этом первый слой является распределительным, а второй слой нейронных элементов осуществляет нелинейное преобразование взвешенной суммы:

$$y_j(t+1) = F(S_j(t)) = F\left(\sum_{\substack{i=1 \\ j \neq i}}^n \omega_{ij} y_i(t) - T_j\right), \quad (7.5)$$

где  $y_j(t+1)$  – выходное значение  $j$ -го нейронного элемента в момент времени  $t+1$ ;  $F$  – оператор нелинейного преобразования;  $T_j$  – пороговое значение  $j$ -го нейрона.

В матричной форме модель Хопфилда можно представить как

$$Y(t+1) = F(S(t)), \quad (7.6)$$

$$S(t) = W^T Y(t) - T.$$

При этом используемые векторы имеют следующий вид:

$$\begin{aligned} S &= [S_1, S_2, \dots, S_n]^T, \\ Y &= [y_1, y_2, \dots, y_n]^T, \\ T &= [T_1, T_2, \dots, T_n]^T, \\ W &= \begin{bmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1n} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2n} \\ \dots & \dots & \dots & \dots \\ \omega_{n1} & \omega_{n2} & \dots & \omega_{nn} \end{bmatrix}. \end{aligned} \quad (7.7)$$

В качестве матрицы весовых коэффициентов Хопфилд применял симметрическую матрицу ( $\omega_{ij} = \omega_{ji}$ ) с нулевой главной диагональю ( $\omega_{ii} = 0$ ).

Последнее условие соответствует отсутствию обратной связи нейронного элемента на себя самого. В качестве функции активации нейронных элементов  $F$  может использоваться как пороговая, так и непрерывная функция, например сигмоидная или гиперболический тангенс.

Рассмотрим нейронную сеть Хопфилда с дискретным временем. При использовании пороговой функции активации она называется нейронной сетью с *дискретным состоянием и временем*, а нейронная сеть с непрерывной функцией активации – нейронной сетью с *непрерывным состоянием и дискретным временем*.

Для описания функционирования таких сетей Хопфилд использовал аппарат статистической физики. При этом каждый нейрон имеет

два состояния активности  $(1, -1)$ , которые аналогичны значениям спина некоторой частицы. Весовой коэффициент  $\omega_{ij}$  можно интерпретировать как вклад поля  $j$ -й частицы в величину потенциала  $i$ -й частицы. Хопфилд показал, что поведение такой сети аналогично поведению изингового спинового стекла. При этом он ввел понятие вычислительной энергии, которую можно интерпретировать в виде ландшафта с долинами и впадинами. Структура соединений сети определяет очертания ландшафта. Нейронная сеть выполняет вычисления, следя по пути, уменьшающему вычислительную энергию сети. Это происходит до тех пор, пока путь не приведет на дно впадины. Данный процесс аналогичен скатыванию капли жидкости по склону, когда она минимизирует свою потенциальную энергию в поле тяготения. Впадины и долины в сети Хопфилда соответствуют наборам информации, которую хранит сеть. Если процесс начинается с приближенной или неполной информации, то он следует по пути, который ведет к ближайшей впадине. Это соответствует операции ассоциативного распознавания.

### 7.2.2. Нейронная сеть Хопфилда как динамическая система

Нейронная сеть Хопфилда является динамической системой с непрерывным или дискретным временем. Состояния нейронов такой сети характеризуют фазовое пространство системы, когда каждому состоянию соответствует точка в фазовом пространстве. Тогда релаксационный процесс изменения состояния нейронов можно интерпретировать как движение точки в фазовом пространстве в устойчивое положение. Траектория движения этой точки может определить процесс воспоминания информации.

Существует параллельная и последовательная динамика работы сети Хопфилда [19].

*Параллельная динамика* обусловлена синхронным функционированием нейронных элементов сети. При этом за один такт работы сети все нейроны одновременно изменяют свое состояние:

$$y_j(t+1) = F \left( \sum_{\substack{i=1 \\ j \neq i}}^n \omega_{ij} y_i(t) - T_j \right), \quad (7.8)$$

где  $j = \overline{1, n}$ .

*Последовательной динамике* свойствен асинхронный процесс работы нейронной сети. В этом случае за один такт работы нейронной сети изменяется состояние только одного нейронного элемента:

$$\begin{aligned} y_j(t+1) &= y_j(t), \quad \forall j \neq k, \\ y_j(t+1) &= F\left(\sum_{\substack{i=1 \\ i \neq j}}^n \omega_{ij} y_i(t) - T_j\right) \text{ при } j = k. \end{aligned} \quad (7.9)$$

Выбор нейрона, который на данном такте должен изменить свое состояние, производится случайным образом. Асинхронную модель функционирования исследовал Хопфилд [19].

Назовем динамическую систему *диссипативной* [65], если производная энергии ее по времени всегда отрицательна или равна нулю (в равновесном состоянии). В диссипативных системах на каждом шаге происходит необратимое уменьшение энергии. В установившемся режиме ( $t = \infty$ ) все состояния таких систем сосредоточиваются на некотором подмножестве  $A$  фазового пространства:

$$\lim_{t \rightarrow \infty} Y(t) = A. \quad (7.10)$$

Такое подмножество называется *аттрактором* и характеризует область притяжения системы. Оно является предельным множеством для фазовых траекторий, близких к аттрактору.

В работе [66] приводится следующая теорема.

**Теорема 7.2.** Нейронная сеть Хопфилда, матрица весовых коэффициентов  $W$  которой имеет нулевую главную диагональ, является диссипативной.

Аттракторами диссипативных динамических систем могут быть устойчивые стационарные точки или устойчивые предельные циклы различной длины [66].

*Устойчивыми стационарными точками* называются такие точки, для которых в установившемся режиме выполняется условие

$$y_j(t+1) = y_j(t), \quad \forall j = \overline{1, n}. \quad (7.11)$$

*Устойчивыми предельными циклами* длины  $k$  называются такие циклы, для которых в установившемся режиме выполняется условие

$$y_j(t+k) = y_j(t), \quad \forall j = \overline{1, n}. \quad (7.12)$$

При использовании симметричной матрицы весовых коэффициентов ( $\omega_{ij} = \omega_{ji}$ ) аттракторами диссипативной динамической системы Хопфилда являются устойчивые стационарные точки или предельные циклы длины два.

### 7.2.3. Энергия сети Хопфилда

Рассмотрим нейронную сеть Хопфилда с дискретным временем и непрерывным состоянием. В этом случае, как уже отмечалось, в качестве функции активации нейронных элементов можно использовать, например, функцию гиперболического тангенса. Тогда

$$y_j(t+1) = \text{th}(S_j(t)), \quad (7.13)$$

$$S_j(t) = \sum_{i=1}^n \omega_{ij} y_i(t) - T_j, \quad (7.14)$$

где  $i \neq j$ .

Состояние нейронного элемента  $y_j(t+1)$  зависит от взвешенной суммы  $S_j(t)$ , которая характеризует входную активность  $j$ -го нейрона. Энергия одного нейронного элемента должна определяться таким образом, чтобы изменение состояния и входной активности нейрона приводило к уменьшению энергии. Тогда выходное значение  $j$ -го нейрона должно быть пропорционально градиенту энергии:

$$y_j = -\frac{dE(y_j, t)}{dS_j}, \quad (7.15)$$

где  $E(y_j, t)$  – энергия  $j$ -го нейронного элемента в момент времени  $t$ .

Такие системы называются *градиентными*. Понятие энергии в градиентных системах соответствует функции Ляпунова. Из выражения (7.15) найдем энергию  $j$ -го нейронного элемента:

$$E(y_j, t) = - \int_0^{y_j(t)} y_j(t) dS_j = -y_j(t) S_j(t) + \int_0^{y_j(t)} S_j(t) dy_i. \quad (7.16)$$

Входную активность  $S_i$  можно найти при помощи обратной функции  $F$ :

$$S_j = F^{-1}(y_j). \quad (7.17)$$

Тогда

$$E(y_j, t) = -y_j(t) S_j(t) + \int_0^{y_j(t)} F^{-1}(y_j) dy_j. \quad (7.18)$$

С учетом (7.14) энергия  $j$ -го нейронного элемента в момент времени  $t$  вычисляется по формуле

$$E(y_j, t) = -\sum_i \omega_{ij} y_i(t) y_j(t) + T_j y_j(t) + \int_0^{y_j(t)} F^{-1}(y_j) dy_j. \quad (7.19)$$

Изменение энергии для одного нейрона можно найти следующим образом:

$$\Delta E(y_j, t+1) = E(y_j, t+1) - E(y_j, t). \quad (7.20)$$

Определим изменение энергии при асинхронном функционировании сети:

$$\begin{aligned} E(y_j, t+1) &= -\sum_i \omega_{ij} y_i(t+1) y_j(t) + T_j y_j(t+1) + \\ &+ \int_0^{y_j(t+1)} F^{-1}(y_j) dy_j = -y_i(t+1) S_i(t) + \int_0^{y_j(t+1)} F^{-1}(y_j) dy_j. \end{aligned} \quad (7.21)$$

Отсюда

$$\Delta E(y_j, t+1) = -\Delta y_j(t+1) S_j(t) + \int_{y_j(t)}^{y_j(t+1)} F^{-1}(y_j) dy_j. \quad (7.22)$$

Согласно теореме о среднем определенного интеграла [67] получим

$$\int_{y_j(t)}^{y_j(t+1)} F^{-1}(y_j) dy_j = F^{-1}(\varepsilon)(y_j(t+1) - y_j(t)), \quad (7.23)$$

где  $y_j(t) \leq \varepsilon \leq y_j(t+1)$ .

Преобразуя последнее выражение, имеем

$$\int_{y_j(t)}^{y_j(t+1)} F^{-1}(y_j) dy_j = \Delta y_j(t+1) S_j(\varepsilon), \quad (7.24)$$

а (7.22) можно представить как

$$\Delta E(y_j, t+1) = \Delta y_j(t+1)(S_j(\varepsilon) - S_j(t)). \quad (7.25)$$

Рассмотрим, как меняется энергия нейрона при изменении его входной и выходной активности.

Пусть  $S_j(t) > S_j(t-1)$ . Тогда  $y_j(t+1) > y_j(t)$ ,  $\Delta y_j(t+1) > 0$  и  $S_j(\varepsilon) < S_j(t)$ .

В результате получим, что  $\Delta E(y_j, t+1) < 0$ .

Предположим, что  $S_j(t) < S_j(t-1)$ . Тогда  $y_j(t+1) < y_j(t)$  и  $\Delta y_j(t+1) < 0$ . Поскольку в этом случае  $S_j(\varepsilon) > S_j(t)$ , то  $\Delta E(y_j, t+1) < 0$ .

Отсюда следует, что изменение входной активности одного нейрона приводит к необратимому уменьшению его энергии. В результате согласно теореме Ляпунова обеспечивается сходимость нейронного элемента.

Определим полную энергию нейронной сети для системы с дискретным временем и непрерывным состоянием. Энергия сети представляет собой сумму энергий всех нейронных элементов, образующих нейронную сеть:

$$E(t) = -\sum_j E(y_j, t) = -\sum_j S_j(t)y_j(t) + \sum_j \int_0^{y_j(t)} F^{-1}(y_j) dy_j. \quad (7.26)$$

Преобразуя первое слагаемое последнего выражения, получим

$$E(t) = -\sum_j \sum_i \omega_{ij} y_i(t) y_j(t) + \sum_j y_j(t) T_j + \sum_j \int_0^{y_j(t)} F^{-1}(y_j) dy_j. \quad (7.27)$$

Поскольку для нейронной сети Хопфилда матрица весовых коэффициентов является симметрической ( $\omega_{ij} = \omega_{ji}$ ), то первое слагаемое выражения (7.27) обычно умножается на  $\frac{1}{2}$ :

$$E(t) = -\frac{1}{2} \sum_j \sum_i \omega_{ij} y_i(t) y_j(t) + \sum_j y_j(t) T_j + \sum_j \int_0^{y_j(t)} F^{-1}(y_j) dy_j. \quad (7.28)$$

В матричной форме энергию системы можно представить в следующем виде:

$$E(t) = -\frac{1}{2} Y^T W Y + Y^T T + \sum_i \int_0^{y_j(t)} F^{-1}(y_j) dy_j. \quad (7.29)$$

Таким образом, энергия сети определяется композицией квадратичной формы и интеграла от нелинейности. Это соответствует функции Ляпунова (7.4), приведенной в разд. 7.1.

Определим теперь энергию сети Хопфилда для системы с дискретными временем и состоянием. В этом случае в качестве функции нелинейного преобразования используется пороговая функция

$$y_j(t+1) = \text{sign}(S_j(t)), \quad (7.30)$$

$$S_j(t) = \sum_{i=1}^n \omega_{ij} y_i(t) - T_j, \quad (7.31)$$

где  $i \neq j$ .

Энергия нейронного элемента в момент времени вычисляется по формуле

$$E(y_j, t) = -y_j(t)S_j(t) = -\sum_i \omega_{ij} y_i(t) y_j(t) + T_j y_j(t). \quad (7.32)$$

Рассмотрим изменение энергии нейронного элемента при асинхронном режиме функционирования сети:

$$E(y_j, t+1) = -y_j(t+1)S_j(t). \quad (7.33)$$

Отсюда

$$\Delta E(y_j, t+1) = -\Delta y_j(t+1)S_j(t). \quad (7.34)$$

Пусть  $S_j(t) \geq 0$ . Тогда  $y_j(t+1) = 1$  и  $\Delta y_j(t+1) \geq 0$ . Получим  $\Delta E(y_j, t+1) \leq 0$ . Если  $S_j(t) \leq 0$ , то  $y_j(t+1) = -1$  и  $\Delta y_j(t+1) \leq 0$ . В результате  $\Delta E(y_j, t+1) \leq 0$ .

Таким образом, в результате изменения входной активности нейронного элемента энергия его остается отрицательной. Согласно теореме 7.1 это гарантирует устойчивость нейронного элемента. Выходное значение нейронного элемента при асинхронном режиме функционирования в зависимости от изменения энергии нейрона можно представить следующим образом:

$$y_j(t+1) = F(S_j(t)) = F\left(-\frac{\Delta E(y_j, t+1)}{\Delta y_j(t+1)}\right). \quad (7.35)$$

Энергия системы вычисляется как

$$E(t) = -\sum_j S_j(t) y_j(t). \quad (7.36)$$

Преобразуя последнее выражение, получим

$$E(t) = -\frac{1}{2} \sum_j \sum_i \omega_{ij} y_i(t) y_j(t) + T_j y_j(t). \quad (7.37)$$

Представим (7.37) в матричной форме:

$$E(t) = -\frac{1}{2}Y^T W Y + Y^T T. \quad (7.38)$$

Отсюда следует, что энергия сети с дискретными временем и состоянием определяется квадратичной формой. Это соответствует функции Ляпунова (7.3). Следует отметить, что энергия сети с непрерывным состоянием сводится в пределе к энергии сети с дискретным состоянием при  $c \rightarrow \infty$ , где  $c$  – коэффициент, характеризующий ширину функции активации (подразд. 2.2.7).

Таким образом, состояние нейронной сети Хопфилда описывается функцией энергии, которая эквивалентна функции Ляпунова. В процессе функционирования сети состояния нейронных элементов изменяются таким образом, что функция энергии их уменьшается. Это гарантирует сходимость нейронных элементов и сети в целом.

#### 7.2.4. Анализ атTRACTоров

В подразд. 7.2.2 отмечалось, что атTRACTорами нейронной сети Хопфилда при использовании симметричной матрицы весовых коэффициентов с нулевой главной диагональю могут быть как устойчивые стационарные точки, так и предельные циклы длины два. Исследуем атTRACTоры сети Хопфилда [27].

**Утверждение 7.1.** Нейронная сеть Хопфилда сходится к устойчивым стационарным точкам, если в установившемся режиме, когда  $y_j(t+1) = y_j(t)$  для всех  $j$ , изменение энергии  $\Delta E(t)$  равняется нулю.

**Утверждение 7.2.** Сеть Хопфилда сходится к предельному циклу длины два, если в установившемся режиме, когда  $y_j(t+2) = y_j(t)$  для всех  $j$ , изменение энергии  $\Delta E(t)$  равняется нулю.

**Утверждение 7.3.** Для сети Хопфилда с дискретными временем и состоянием, если  $y_j(t+2) = y_j(t)$  для всех  $j$ , то  $y_j(t) = -y_j(t+1)$  для всех  $j$ .

**Утверждение 7.4.** Устойчивый предельный цикл длины два будет существовать, когда при  $y_j(t) = -y_j(t+1)$  для всех  $j$  изменение энергии нейронной сети  $\Delta E(t)$  равняется нулю.

Данные утверждения являются очевидными. Рассмотрим атTRACTоры нейронной сети с дискретными временем и состоянием.

**Теорема 7.3.** Нейронная сеть Хопфилда с асинхронным режимом функционирования сходится только к устойчивым стационарным точкам.

*Доказательство.* Энергия сети в два последовательных момента времени при асинхронном функционировании определяется следующим образом:

$$E(t) = -\sum_j S_j(t) y_j(t), \quad (7.39)$$

$$E(t+1) = -\sum_{i \neq k} S_i(t) y_i(t) - S_k(t) y_k(t+1), \quad (7.40)$$

где  $k$  – номер нейронного элемента, для которого производится изменение состояния.

Тогда изменение энергии можно вычислить как

$$\Delta E(t+1) = -\Delta y_k(t+1) S_k(t), \quad (7.41)$$

где  $\Delta y_k(t+1) = y_k(t+1) - y_k(t)$ .

Рассмотрим наличие устойчивых стационарных точек в такой сети. Поскольку при  $y_k(t+1) = y_k(t)$  изменение энергии  $\Delta E(t) = 0$ , то согласно утверждению (7.1) сеть сходится к устойчивым стационарным точкам.

Исследуем существование предельных циклов длины два. Если  $y_k(t) = -y_k(t+1)$ , то  $\Delta y_k(t+1) \neq 0$  и, следовательно,  $\Delta E(t+1) \neq 0$ .

Теорема доказана.

**Теорема 7.4.** Аттракторами нейронной сети Хопфилда с синхронной динамикой являются устойчивые стационарные точки или предельные циклы длины два.

*Доказательство.* При синхронном режиме функционирования энергия сети в два последовательных момента времени определяется как

$$E(t) = -\sum_j S_j(t) y_j(t), \quad (7.42)$$

$$E(t+1) = -\sum_j S_j(t+1) y_j(t+1). \quad (7.43)$$

Тогда изменение энергии вычисляется по формуле

$$\Delta E(t+1) = -\left( \sum_j S_j(t+1) y_j(t+1) - \sum_j S_j(t) y_j(t) \right). \quad (7.44)$$

Рассмотрим наличие устойчивых стационарных точек. Если  $y_j(t+1) = y_j(t)$ , то  $S_j(t+1) = S_j(t)$ . Отсюда получим  $\Delta E(t+1) = 0$ , что доказывает наличие устойчивых точек.

Определим существование предельных циклов длины два. Если  $y_j(t+1) = -y_j(t)$ , то  $S_j(t+1) = -S_j(t)$ . Из выражения (7.44) получим, что  $\Delta E(t+1) = 0$ , что доказывает наличие циклов длины два.

Теорема доказана.

**Пример 7.1.** Рассмотрим нейронную сеть Хопфилда с двумя нейронными элементами и пороговыми значениями, равными нулю (рис. 7.3).

В качестве функции активации нейронных элементов второго слоя используем пороговую функцию. Выходные значения сети являются биполярными, т. е.  $y_i \in [1, -1]$ .

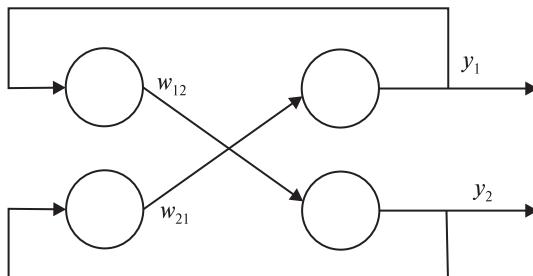


Рис. 7.3. Сеть Хопфилда с двумя нейронными элементами

Пусть  $w_{12} = w_{21} = -1$ . Матрица весовых коэффициентов сети имеет вид

$$W = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.$$

Тогда вектор выходных значений определяется как

$$Y(t+1) = \text{sign}(W^T Y(t)).$$

Пусть в момент времени  $t=0$   $y_1(0)=1$ ,  $y_2(0)=-1$ . Тогда  $y_1(1)=1$ ,  $y_2(1)=-1$  и т. д. Отсюда следует, что точки 1 и  $-1$  являются устойчивыми стационарными точками.

Предположим теперь, что  $y_1(0)=1$  и  $y_2(0)=1$ . Тогда  $y_1(1)=-1$ ,  $y_2(1)=-1$ ;  $y_1(2)=1$ ,  $y_2(2)=1$  и т. д., т. е.  $y_j(t+2) = y_j(t)$ . Отсюда следует, что в такой сети присутствуют осцилляции в виде циклов длины два.

Аналогичная картина наблюдается для нейронных сетей с дискретным временем и непрерывным состоянием. В работе [66] доказана тео-

рема о том, что если матрица весовых коэффициентов нейронной сети Хопфилда с синхронной динамикой положительно полуопределенная (все ее собственные значения неотрицательны), то атTRACTорами такой системы являются только точки покоя. Если матрица синаптических связей несимметрична, то в такой сети возможно существование циклов различной длины [66].

Рассмотрим квадратичную форму функции энергии:

$$E(t) = -\frac{1}{2} Y^T W Y = -\frac{1}{2} \sum_j \sum_i \omega_{ij} y_i(t) y_j(t). \quad (7.45)$$

При помощи ортогонального преобразования ее можно представить в канонической форме [73]:

$$E(t) = -\frac{1}{2} \sum_i \lambda_j y_j^2, \quad (7.46)$$

где  $\lambda_j$ ,  $j = \overline{1, n}$ , — характеристические числа матрицы синаптических связей.

Можно показать, что если  $y_j \in \{-1, 1\}$ , то минимумы функции энергии (7.46) достигаются в узлах  $n$ -мерного куба (гиперкуба). Нейронная сеть с  $n$  нейронами имеет  $2^n$  состояний. При установке сети в начальное состояние происходит релаксационный процесс достижения минимума энергии, который определяется ближайшей вершиной гиперкуба.

### 7.2.5. Правило обучения Хебба

*Самоадаптация и самоорганизация* нейронных сетей достигается в процессе их обучения, в ходе которого определяются синаптические связи между нейронными элементами. Обучающие правила устанавливают, как изменяются весовые коэффициенты в ответ на входное воздействие. Для обучения сети Хопфилда используется правило обучения Хебба.

Правило обучения Хебба [11], как отмечалось в гл. 1, имеет биологические предпосылки. Оно является основой многих методов обучения нейронных сетей. Согласно правилу Хебба обучение происходит в результате усиления силы связи (синаптического веса) между одновременно активными нейронами. Исходя из этого часто используемые в сети связи усиливаются, что объясняет феномен обучения путем повторения и привыкания. Пусть имеются два нейронных элемента  $i$  и  $j$ , между которыми существует сила связи, равная  $\omega_{ij}$  (рис. 7.4).

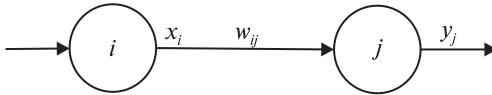


Рис. 7.4. Взаимосвязь двух нейронных элементов

Тогда правило обучения Хебба можно записать как

$$\omega_{ij}(t+1) = \omega_{ij}(t) + x_i y_j,$$

где  $t$  – время;  $x_i$  и  $y_j$  – выходные значения  $i$ -го и  $j$ -го нейрона соответственно.

В начальный момент времени предполагается, что

$$\omega_{ij}(t=0) = 0, \quad \forall i, j.$$

Правило обучения Хебба можно получить исходя из минимизации энергии сети Хопфилда:

$$E(t) = -\frac{1}{2} \sum_j \sum_i \omega_{ij} y_i(t) y_j(t) + T_j y_j(t).$$

В соответствии с методом градиентного спуска имеем

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial E(t)}{\partial \omega_{ij}(t)}.$$

Учитывая, что

$$\frac{\partial E(t)}{\partial \omega_{ij}(t)} = -y_i y_j,$$

правило Хебба можно представить в следующем виде:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha y_i y_j.$$

### 7.2.6. Ассоциативная память

Сеть Хопфилда может использоваться в качестве ассоциативной памяти. В этом случае она способна распознавать зашумленные или искаженные образы. Для обучения нейронной сети Хопфилда используется правило Хебба. Представим входные образы, которые необходимо запомнить, в виде следующей матрицы:

$$Y = \begin{bmatrix} y^1 \\ y^2 \\ \dots \\ y^L \end{bmatrix} = \begin{bmatrix} y_1^1 & y_2^1 & \dots & y_n^1 \\ y_1^2 & y_2^2 & \dots & y_n^2 \\ \dots \\ y_1^L & y_2^L & \dots & y_n^L \end{bmatrix},$$

где  $L, n$  – общее количество и размерность входных образов соответственно.

Пусть входные векторы являются биполярными, т. е.  $y_i \in \{-1, 1\}$ . Тогда синаптические коэффициенты нейронной сети в матричной форме определяются как

$$W = \sum_{k=1}^L (y^k)^T y^k = Y^T Y, \quad (7.47)$$

или в обычной форме:

$$\omega_{ij} = \sum_{k=1}^L y_i^k y_j^k, \quad (7.48)$$

где  $i = \overline{1, n}; j = \overline{1, n}$ , причем  $i \neq j$ .

Поскольку главная диагональ матрицы весовых коэффициентов сети Хопфилда должна быть нулевой, то

$$W = \sum_{k=1}^L \left( (y^k)^T y^k - I \right) = Y^T Y - LI, \quad (7.49)$$

где  $I$  – единичная матрица.

При использовании бинарных векторов, когда  $y_i \in \{0, 1\}$ , правило обучения можно представить в следующем виде:

$$w_{ij} = \sum_{k=1}^L (2y_i^k - 1)(2y_j^k - 1), \quad (7.50)$$

$$W = (2Y - 1)^T (2Y - 1) - LI, \quad (7.51)$$

где  $i, j = \overline{1, n}; i \neq j$ .

Приведенные выше выражения являются модификацией правила обучения Хебба. Они допускают усиление силы связи не только между одновременно активными, но и одновременно неактивными нейронами. На практике обычно используется биполярное представление входных сигналов.

Для настройки порогов нейронных элементов можно применить выражение

$$T_j = -\sum_{k=1}^L y_j^k, \quad (7.52)$$

где  $y_j^k$  –  $j$ -я компонента  $k$ -го входного образа.

В ассоциативной памяти пороговые значения нейронных элементов обычно принимаются равными нулю. Матрица весовых коэффициентов, получаемая в соответствии с правилом Хебба, симметрична и характеризуется нулевой главной диагональю. Она представляет собой ковариационную матрицу входных образов, собственные векторы которой ортогональны. Тогда взвешенную сумму  $i$ -го нейрона можно представить как

$$S_j(t) = \lambda_j y_j(t), \quad (7.53)$$

где  $\lambda_j$  –  $j$ -е собственное значение матрицы  $W$ .

В результате вычислим энергию сети по формуле

$$E(t) = -\frac{1}{2} \sum_j S_j(t) y_j(t) = -\frac{1}{2} \sum_i \lambda_j y_j^2(t). \quad (7.54)$$

Из этого выражения следует, что если  $\lambda_j < 0$ , то при минимизации энергии значение  $y_j(t)$  должно стремиться к нулю. В противном случае если  $\lambda_j > 0$ , то значение  $y_j(t)$  для минимизации энергии будет увеличиваться. Таким образом, в процессе релаксации сети она будет ослаблять одни компоненты вектора  $Y$  и усиливать другие. На этом основана коррекция нейронной сетью Хопфилда ошибок во входных образах.

После обучения нейронной сети входные образы создают впадины, соответствующие определенным наборам информации (рис. 7.5).

На рис. 7.5 ось ординат соответствует энергии сети Хопфилда, а  $y^k$  характеризует  $k$ -й образ, хранимый в сети. Таким образом, информация хранится в локальных минимумах функции энергии, которые соответствуют стабильным состояниям сети.

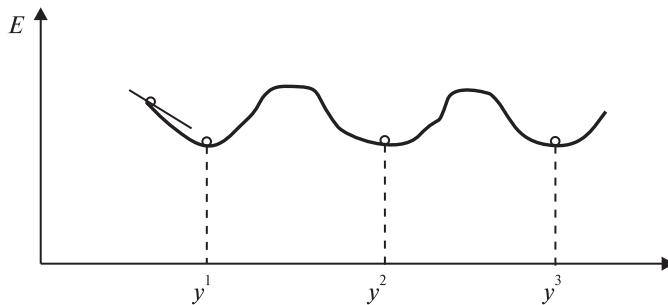


Рис. 7.5. Хранение образов в нейронной сети Хопфилда

Совокупность точек, которые при равновесном состоянии сети отображаются в точку локального минимума, называется *бассейном притяжения*. Паттерны, соответствующие бассейну притяжения, характеризуют зашумленные образы памяти, которая при подаче на вход сети зашумленного образа за конечное число шагов переходит в состояние, соответствующее ближайшему локальному минимуму (см. рис. 7.5). Количество шагов определяется расстоянием Хэмминга между искаженным и эталонным образом. Чем дальше расположен входной образ от локального минимума, тем больше итераций необходимо сети для перехода в стабильное состояние. Если наборы входных данных оказываются слишком похожими или слишком многочисленными, то в этом случае локальные минимумы будут располагаться слишком близко друг к другу и начинают интерферировать между собой (рис. 7.6). При этом появляются локальные минимумы, соответствующие комбинациям из достаточно больших фрагментов эталонных образов. Такое явление Хопфилд назвал химерами [58]. Если количество образов, хранящихся в сети, больше  $0,15n$ , то локальные минимумы начинают исчезать и поведение сети становится хаотическим (состояние спинового стекла). Размер бассейна притяжения можно регулировать при помощи синаптических связей и порогов нейронных элементов.



Рис. 7.6. Иллюстрация ложных минимумов

Так, если сила тормозящих соединений будет возрастать, то и бассейн притяжения увеличится, а впадины станут глубже. Если уменьшить порог нейрона, то впадина также углубится. При достаточном уменьшении порога (в отрицательную сторону) бассейн притяжения может охватить все пространство конфигураций, ликвидировав впадины, соответствующие другим стабильным состояниям.

Хопфилд установил экспериментально, что объем памяти для устойчивой работы сети определяется как

$$L = 0,15n, \quad (7.55)$$

где  $n$  – количество нейронов сети.

Позже были получены различные аналитические оценки допустимого объема памяти сети Хопфилда, например

$$L = \frac{n}{4\ln(n)}. \quad (7.56)$$

Данные выражения ориентировочные. Так, при увеличении корреляции между входными образами и, соответственно, уменьшении расстояния Хэмминга емкость памяти сократится. В связи с этим рекомендуется в качестве запоминаемых паттернов применять линейно независимые, а лучше – ортогональные векторы [68]. Для получения ортогональных векторов можно воспользоваться матрицей Адамара порядка  $n$ , элементами которой являются действительные числа  $\pm 1$ . Так, матрица Адамара второго порядка вычисляется по формуле

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Матрица Адамара более высокого порядка определяется на основе матриц меньшего порядка:

$$H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}.$$

Так, например, матрица четвертого порядка имеет следующий вид:

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

Вычислим для ортогональных векторов расстояние Хэмминга:

$$\delta = n / 2. \quad (7.57)$$

Назовем *радиусом притяжения* число битов, на которое может отличаться входной вектор от эталонного, чтобы произошел процесс его идентификации. Для ортогональных векторов радиус притяжения найдем следующим образом [68]:

$$r = \frac{n}{2L}, \quad (7.58)$$

где  $L$  – количество образцов, хранимых сетью.

При увеличении емкости памяти уменьшается радиус притяжения и, соответственно, корректирующие свойства сети Хопфилда. В некоторых работах для расширения емкости памяти рекомендуется использовать сети Хопфилда *высокого порядка* [69]. Так, например, для сети второго порядка выходное значение  $i$ -го нейронного элемента можно вычислить как

$$y_j(t+1) = F(S_j(t)), \quad (7.59)$$

$$S_j(t) = \sum_{h=1}^n \sum_{i=1}^n w_{hij} y_h(t) y_i(t), \quad (7.60)$$

где  $w_{hij}$  – синаптическая связь от  $h$ -го и  $i$ -го к  $j$ -му нейрону.

Обобщая правило обучения Хебба, можно получить выражение для настройки весовых коэффициентов сети второго порядка:

$$w_{hij} = \sum_{k=1}^L X_h^k X_i^k X_j^k - I, \quad (7.61)$$

где  $I$  – трехмерная единичная матрица.

Весовые коэффициенты для такой нейронной сети являются также симметричными, т. е.

$$w_{hji} = w_{jhi} = w_{ijh}.$$

Использование нейронной сети Хопфилда второго порядка позволяет повысить емкость памяти приблизительно до  $0,3n$  [69].

### 7.2.7. Функционирование сети Хопфилда

Функционирование сети Хопфилда представляет собой релаксационный процесс, в ходе которого сеть достигает устойчивого состояния. Алгоритм функционирования можно представить в виде их шагов.

1. На вход сети подается неизвестный образ.

2. В зависимости от синхронного или асинхронного режима работы сети производятся следующие вычисления:

а) при синхронном режиме функционирования сети выходы нейронных элементов изменяются одновременно:

$$y_j(t+1) = F\left(\sum_i \omega_{ij} y_i(t)\right), \quad \forall i,$$

где в качестве оператора нелинейного преобразования  $F$  обычно используется пороговая функция;

б) в случае асинхронного режима работы в каждый тик времени изменяется состояние только одного нейронного элемента  $k$ :

$$y_j(t+1) = F\left(\sum_i \omega_{ij} y_i(t)\right), \text{ если } j = k,$$

$$y_j(t+1) = y_j(t), \text{ для } j \neq k.$$

При этом на каждом шаге работы сети такой нейронный элемент выбирается случайно из всей совокупности нейронных элементов или из тех нейронов, которые на предыдущем шаге не изменили свое состояние.

3. Пункт 2 повторяется до тех пор, пока сеть не перейдет в стабильное состояние. При этом состояния всех нейронных элементов переходят изменяться, т. е.

$$y_j(t+1) = y_j(t), \quad \forall j.$$

**Пример 7.2.** Пусть требуется, чтобы сеть Хопфилда хранила один бинарный образ:

$$Y = (1 \ 1 \ 1 \ 0).$$

Тогда

$$\begin{aligned} W &= (2Y - 1)^T (2Y - 1) - I = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\ &= \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}. \end{aligned}$$

В весовой матрице  $i$ -й столбец соответствует весовым связям  $i$ -го нейронного элемента. Пусть на вход сети поступает зашумленный образ  $Y = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$ . Рассмотрим различные режимы функционирования.

### 1. Асинхронный режим

Допустим, что в начальный момент времени  $t=1$  выбран первый нейрон, состояние которого будет изменяться:

$$y_1 = \text{sign}(YW_1),$$

$$S_1 = YW_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix} = 0 + 0 + 1 + 0 = 1,$$

где  $W_1$  – первый столбец весовой матрицы.

Отсюда

$$y_1 = \text{sign}(1) = 1.$$

В результате этого на выходе сети получим следующее:

$$Y(1) = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}.$$

На втором шаге  $t=2$  выберем четвертый нейрон, состояние которого изменяется:

$$y_4 = \text{sign}(Y(1)W_4) = \text{sign}(-2) = 0,$$

$$Y(2) = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}.$$

При изменении в следующий момент времени состояния третьего нейрона имеем

$$y_3 = F(Y(2)W_3) = F(1) = 1,$$

$$Y(3) = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}.$$

Аналогично для второго нейронного элемента

$$y_2 = F(Y(3)W_2) = F(2) = 1,$$

$$Y(4) = \begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix}.$$

Если продолжить процесс дальше, то можно показать, что состояния нейронных элементов перестанут изменяться. Это свидетельствует о переходе сети в стабильное состояние. Следовательно, она распознала зашумленный образ.

## 2. Синхронный режим

В матричной форме функционирование сети в синхронном режиме можно представить как

$$\begin{aligned}y(t+1) &= \text{sign}(S(t)), \\ S(t) &= Y(t)W.\end{aligned}$$

Пусть на вход сети подается зашумленный образ  $Y = [0 \ 0 \ 1 \ 0]$ .

Тогда в начальный момент времени

$$S(0) = Y(0)W = [0 \ 0 \ 1 \ 0] \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix} = [1 \ 1 \ 0 \ -1],$$

$$Y(1) = \text{sign}(S(0)) = [1 \ 1 \ 0 \ 0].$$

Для следующего такта работы состояние сети

$$S(1) = Y(1)W = [1 \ 1 \ 2 \ -2],$$

$$Y(2) = \text{sign}(S(1)) = [1 \ 1 \ 1 \ 0].$$

При дальнейшем функционировании сеть перестает изменять свое состояние. В результате произойдет распознавание зашумленного образа.

### 7.2.8. Решение задач оптимизации

Нейронные сети Хопфилда можно применять для решения различного комбинаторных задач оптимизации [66]. Это основано на том, что стабильные состояния нейронной сети соответствуют локальным минимумам функции энергии. Пусть необходимо найти минимум целевой функции

$$\min F(x). \quad (7.62)$$

Для решения такой задачи с использованием нейронной сети Хопфилда следует поставить в соответствие целевую функцию и функцию энергии:

$$F(x) = E. \quad (7.63)$$

Решая полученное уравнение относительно  $W$ , можно определить весовые коэффициенты сети Хопфилда для минимизации целевой

функции. Запрограммированная подобным образом нейронная сеть будет находить решение задачи оптимизации в виде минимума функции энергии. В результате получается решение, как правило, в виде локального, а не глобального оптимума. Это является недостатком сети Хопфилда, а ее достоинство заключается в высокой скорости вычислений. Особенно это важно для NP-полных задач оптимизации. Другой аспект этой проблемы состоит в том, что во многих областях требуется найти решение задачи в реальном масштабе времени. При этом главным фактором выступает быстродействие алгоритма, а не определение глобального оптимума. Для таких задач эффективно применение сети Хопфилда, которая позволяет достаточно быстро найти локальное решение задачи оптимизации. Рассмотрим решение задачи коммивояжера при помощи нейронной сети Хопфилда.

Задача *коммивояжера* (Travelling salesman problem) формулируется следующим образом. Пусть даны города и расстояния между ними. Требуется найти замкнутый маршрут коммивояжера, который начинается и заканчивается в одном и том же городе и проходит через все города. При этом маршрут должен иметь минимальную длину, а также входить в каждый город и выходить из него по одному разу. Задача коммивояжера является NP-полной.

Рассмотрим ее нейропостановку [66, 70]. Пусть количество городов равняется  $n$ . Тогда структура нейронной сети Хопфилда представляет собой матрицу нейронных элементов размерностью  $n \times n$ . В такой матрице номер строки соответствует городу, а номер столбца характеризует позицию города в путешествии. Пусть  $n = 4$ , а города обозначим как  $A, B, C, D$ .

Матрица нейронных элементов для этого случая изображена на рис. 7.7. Определим выходные значения нейронных элементов следующим образом:

$$Y_{Ai} = \begin{cases} 1, & \text{если город } A \text{ стоит в маршруте на } i\text{-м месте,} \\ 0, & \text{иначе.} \end{cases}$$

Тогда, если активными будут нейроны, показанные на рис. 7.7, то маршрут коммивояжера имеет вид

$$C \rightarrow A \rightarrow B \rightarrow D \rightarrow C.$$

Целевая функция задачи коммивояжера будет состоять из двух частей:

- одна отвечает за допустимость решения, которое определяет синтаксис задачи;
- другая представляет качество решения и непосредственно отвечает за минимизацию целевой функции.

	Номер маршрута			
	1	2	3	4
A	○	⊗	○	○
B	○	○	⊗	○
C	⊗	○	○	○
D	○	○	○	⊗

Город ↓

Рис. 7.7. Матрица нейронных элементов для четырех городов

Синтаксис задачи должен удовлетворять следующим условиям:

- Каждый город должен стоять в маршруте не более чем на одном месте.
- На каждом месте маршрута должен стоять не более чем один город.
- Общее число единиц в матрице нейронных элементов должно быть равно  $n$ .

Тогда целевую функцию, ответственную за синтаксис задачи, можно представить как

$$F_c = \frac{a}{2} \sum_A \sum_i \sum_{j \neq i} Y_{Ai} Y_{Aj} + \frac{b}{2} \sum_i \sum_A \sum_{j \neq i} Y_{Ai} Y_{Aj} + \frac{c}{2} \left( \sum_A \sum_i Y_{Ai} - n \right)^2, \quad (7.64)$$

где  $a, b, c$  – постоянные, которые больше нуля. Минимум данного выражения равен нулю, когда выполняются три перечисленных выше условия синтаксиса задачи.

Целевая функция, отвечающая за качество решения, должна обеспечивать минимальное расстояние выбранного маршрута. Она определяется следующим образом:

$$F_e = \frac{d}{2} \sum_A \sum_{B \neq A} \sum_i \text{dist}(A, B) Y_{Ai} (Y_{Bi+1} + Y_{Bi-1}), \quad (7.65)$$

где  $\text{dist}(A, B)$  – расстояние между городами  $A$  и  $B$ ;  $D$  – постоянная, которая больше нуля.

Найдем энергию дискретной сети Хопфилда для матрицы нейронных элементов по формуле

$$E(t) = \frac{1}{2} \sum_A \sum_{B \neq A} \sum_i \sum_{j \neq i} \omega_{Ai, Bj} Y_{Ai}(t) Y_{Bj}(t) + \sum_A \sum_i T_{Ai} Y_{Ai}(t). \quad (7.66)$$

Составим уравнение

$$E(t) = Fc + Fe. \quad (7.67)$$

Тогда решение уравнения (7.66) можно представить в виде

$$W_{Ai,Bj} = W_{Ai,Bj}(1) + W_{Ai,Bj}(2) + W_{Ai,Bj}(3) + W_{Ai,Bj}(4). \quad (7.68)$$

Вычислим слагаемые последнего выражения:

$$W_{Ai,Bj}(1) = -a\delta_{AB}(1 - \delta_{ij}), \quad (7.69)$$

$$W_{Ai,Bj}(2) = -b\delta_{ij}z(1 - \delta_{AB}), \quad (7.70)$$

$$W_{Ai,Bj}(3) = -c, \quad (7.71)$$

$$W_{Ai,Bj}(4) = -d(1 - \delta_{AB})(\delta_{ij+1} + \delta_{ij-1})\text{dist}(A, B). \quad (7.72)$$

Пороги нейронных элементов при этом одинаковые:

$$T_{Ai} = -cn. \quad (7.73)$$

Тогда уравнение динамики сети Хопфилда можно записать как

$$S_{Ai}(t) = -a \sum_{j \neq i} Y_{Aj} - b \sum_{B \neq A} Y_{Bi} - c \sum_{B \neq A} \sum_{j \neq i} Y_{Bj} - d \sum_{B \neq A} \text{dist}(A, B)(Y_{Bi+1} + Y_{Bi-1}) + cn.$$

Можно также задать динамику функционирования сети Хопфилда через изменение функции энергии:

$$Y_{Ai}(t+1) = \text{sign}\left(\frac{-\Delta E(t)}{\Delta Y_{Ai}(t)}\right). \quad (7.74)$$

Хопфилд использовал для моделирования задачи коммивояжера следующие константы:

$$a = b = 500, \quad c = 200, \quad d = 500.$$

Недостатком такой сети является сложность выбора подходящих постоянных  $a, b, c, d$ . В работе [66] приводится методика выбора данных постоянных. Как уже отмечалось, сеть Хопфилда дает решение задач оптимизации в основном в виде локального минимума. По оценке Хопфилда, только 50 % всех решений получается в области, близкой к глобальному оптимуму. Другие авторы [71] снижают эту цифру до 15 %. Для улучшения решения задачи можно применять метод имитационного отжига, который является разновидностью процедуры случайного поиска [72] и позволяет выбраться из локального минимума целевой функции. Имитационный отжиг моделирует процесс релаксации системы, когда она при некоторой конечной температуре стремится к состоянию равновесия.

### 7.3. НЕЙРОННАЯ СЕТЬ ХЭММИНГА

Нейронная сеть Хэмминга (Hamming network) была предложена в 1987 г. Р. Липпманом (R. Lippman) [73]. Она представляет собой релаксационную многослойную сеть, в которой применяются обратные связи между отдельными слоями. Сеть Хэмминга используется в качестве ассоциативной памяти. При распознавании образов в ней в качестве меры близости применяют расстояние Хэмминга. Весовые коэффициенты и пороги сети Хэмминга определяются из условия задачи. Поэтому такая сеть является нейронной сетью с фиксированными связями.

#### 7.3.1. Архитектура сети

Сеть Хэмминга многослойная и состоит из различных классов нейронных сетей. Пусть задано  $m$  образов, каждый из которых имеет размерность  $n$ :

$$X^1 = \left[ x_1^1, x_2^1, \dots, x_n^1 \right],$$

$$X^2 = \left[ x_1^2, x_2^2, \dots, x_n^2 \right],$$

.....

$$X^m = \left[ x_1^m, x_2^m, \dots, x_n^m \right].$$

Тогда нейронная сеть Хэмминга будет состоять из сети с прямыми связями, сети Хопфилда и слоя выходных нейронов (рис. 7.8).

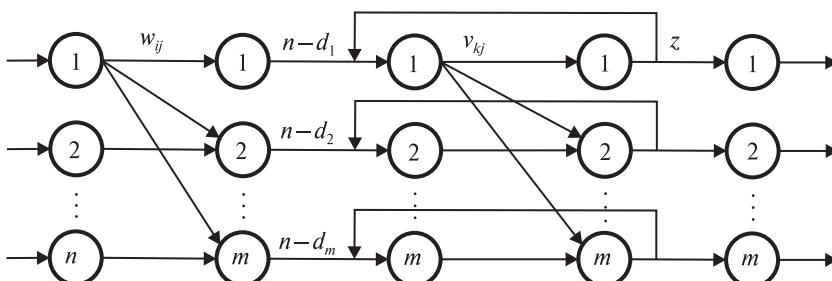


Рис. 7.8. Архитектура нейронной сети Хэмминга

Сеть с прямыми связями состоит из  $n$  входных распределительных и  $m$  выходных нейронных элементов. Она вычисляет меру подобия между входным и эталонным образом, хранящимися в сети. В качестве меры подобия используется количество одинаковых разрядов между входным и эталонным образом. Тогда выходное значение  $i$ -го нейрона второго слоя представляет собой меру подобия  $P_i$  между входным и  $i$ -м эталонным образом:

$$P_i = n - d_i, \quad (7.75)$$

где  $d_i$  – расстояние Хэмминга между входным и  $i$ -м эталонным паттерном.

Сеть Хопфилда используется для разрешения возникающих конфликтов, когда входной паттерн подобен нескольким эталонным образам, хранящимся в сети. В процессе релаксации сети Хопфилда на ее выходе остается только один нейронный элемент с положительной выходной активностью.

Выходной слой нейронной сети состоит из  $m$  нейронов, каждый из которых имеет пороговую функцию активации. Он предназначен для преобразования положительной выходной активности нейрона сети Хэмминга в единичное значение. При этом значения всех остальных нейронов выходного слоя устанавливаются в нулевое состояние. Таким образом, происходит идентификация входного паттерна, который кодируется номером нейрона выходного слоя, имеющим единичное значение. Если входной образ не совпадает с эталонным, то на выходе сети Хэмминга будет формироваться эталонный паттерн с минимальным расстоянием Хэмминга по отношению к выходному образу.

### 7.3.2. Обучение

Рассмотрим правила определения весовых коэффициентов для различных слоев нейронной сети Хэмминга.

Весовые коэффициенты и пороги сети с прямыми связями настраиваются таким образом, чтобы выходное значение  $i$ -го нейронного элемента соответствовало  $i$ -й мере подобия  $P_i$  между входным и  $i$ -м эталонным образом. Для этого необходимо, чтобы

$$w_{ij}^j = \frac{x_i^j}{2}, \quad T_j = \frac{n}{2}, \quad (7.76)$$

где  $i = \overline{1, n}$ ;  $j = \overline{1, m}$ .

Тогда выходная активность  $j$ -го нейронного элемента сети с прямыми связями определяется как

$$y_j = \sum_{i=1}^n w_{ij}x_i + T_j = \sum_{i=1}^n \frac{x_i^j}{2}x_i + \frac{n}{2} = \frac{1}{2} \left( \sum_{i=1}^n x_i^j x_i + n \right). \quad (7.77)$$

Легко показать, что выражение (7.77) эквивалентно мере подобия между входным и  $j$ -м эталонным образом:

$$y_j = P_j = n - d_j. \quad (7.78)$$

Сеть Хопфилда, как уже отмечалось, предназначена для устранения возможных конфликтов, когда входной паттерн похож на несколько эталонных образов, хранящихся в сети. Найдем для этого весовые коэффициенты сети Хопфилда:

$$v_{kj} = \begin{cases} 1, & \text{если } k = j, \\ -e, & \text{если } k \neq j, \end{cases} \quad (7.79)$$

где  $e = \text{const}$ . Параметр  $e$  обычно лежит в диапазоне

$$0 < e < \frac{1}{m}. \quad (7.80)$$

Таким образом, каждый нейрон связан с остальными нейронными элементами сети Хопфилда только тормозящими связями. Начальная инициализация сети Хопфилда происходит на основе нейронной сети с прямыми связями:

$$z_j(0) = y_j, \quad j = \overline{1, m}.$$

В процессе релаксации сеть Хопфилда изменяет свое состояние:

$$z_j(t+1) = F(S_j(t)),$$

$$S_j(t) = \sum_{k=1}^m v_{kj} z_k(t).$$

Преобразуем данное выражение с учетом того, что при  $k = j$ ,  $v_{ji} = 1$ :

$$S_j(t) = \sum_{k=1}^m v_{kj} z_k(t) = z_j(t) - e \sum_{k=1}^m z_k(t),$$

где  $k \neq j$ .

В качестве функции активации нейронных элементов сети Хопфилда используется следующая функция:

$$z_j(t+1) = F(S_j(t)) = \begin{cases} S_j(t), & \text{если } S_j(t) > 0, \\ 0, & \text{если } S_j(t) \leq 0. \end{cases}$$

Релаксационный процесс происходит до тех пор, пока только один нейронный элемент сети Хопфилда не останется с положительной активностью. Такой нейронный элемент является победителем в конкурентной борьбе. Выходной слой сети Хэмминга преобразует выходную активность нейрона-победителя в единичное значение, а остальных нейронов – в нулевое значение. Для этого нейроны выходного слоя используют пороговую функцию активации. Номер нейрона победителя идентифицирует распознанный образ. Количество образов, хранимых в сети, равняется количеству нейронных элементов выходного слоя.

### 7.3.3. Алгоритм функционирования

Рассмотрим алгоритм функционирования нейронной сети Хэмминга. Он состоит из следующих шагов:

1. Определяются весовые коэффициенты и пороговые значения для соответствующих слоев нейронной сети.
2. На вход сети подается неизвестный образ и производится инициализация нейронных элементов сети Хопфилда в соответствии с выражениями (7.78).
3. Производится итерационная процедура расчета выходных значений нейронной сети Хопфилда до тех пор, пока она не стабилизируется. В этом случае на выходе сети Хэмминга один нейронный элемент будет иметь единичное состояние, а остальные – нулевое состояние.
4. Если в выходном слое существует несколько нейронных элементов-победителей, то выбор одного из них производится случайным образом.

## 7.4. ДВУНАПРАВЛЕННАЯ АССОЦИАТИВНАЯ ПАМЯТЬ

Двунаправленную ассоциативную память (Bidirectional Associative Memory) предложил в 1987 г. Б. Коско (B. Kosko) [60, 61]. Она является дальнейшим развитием сети Хопфилда и представляет собой релаксационную сеть с циркуляцией информации.

### 7.4.1 Архитектура

Двунаправленная ассоциативная память представляет собой нейронную сеть, состоящую из двух слоев нейронных элементов (рис. 7.9). Нейронные элементы каждого из слоев могут быть как входными, так и выходными.

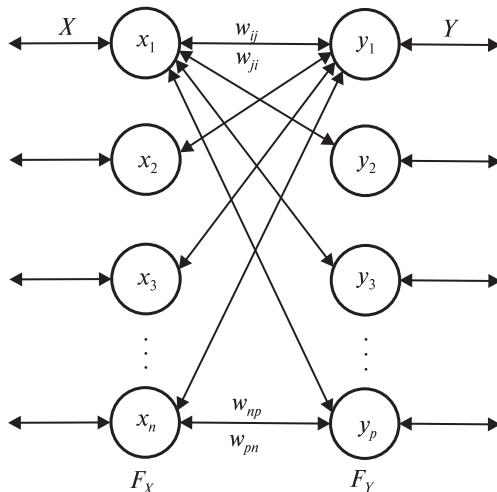


Рис. 7.9. Архитектура двунаправленной ассоциативной памяти

При прямом распространении информации нейронные элементы слоя  $F_x$  являются входными, а слоя  $F_y$  – выходными. При обратном распространении информации входными являются нейроны слоя  $F_y$ , а выходными – слоя  $F_x$ . Каждый нейронный элемент слоя  $F_x$  имеет синаптические связи  $w_{ij}$  со всеми нейронами слоя  $F_y$ . Синаптические связи в обратном направлении от слоя  $F_y$  к слою  $F_x$  обозначим как  $w_{ji}$  соответственно. Архитектура двунаправленной памяти аналогична рециркуляционной нейронной сети. Однако принципы функционирования таких сетей разные. Двунаправленная ассоциативная память, в отличие от рециркуляционной сети, представляет собой релаксационную сеть.

### 7.4.2. Обучение и функционирование

Для обучения двунаправленной памяти используется правило Хебба. Пусть задано  $L$  пар векторов  $X$  и  $Y$ :

$$X_k = [x_1^k, x_2^k, \dots, x_n^k],$$

$$Y_k = [y_1^k, y_2^k, \dots, y_p^k],$$

где  $k = \overline{1, L}$ ;  $L$  – количество образов одного типа, хранимых в нейронной сети.

Тогда в соответствии с правилом Хебба весовые коэффициенты такой сети определяются следующим образом:

$$w_{ij} = \sum_{k=1}^L x_i^k y_j^k \quad (7.81)$$

или в матричной форме:

$$W = X^T Y. \quad (7.82)$$

Найдем весовые коэффициенты  $W'$  обратного слоя:

$$W' = W^T. \quad (7.83)$$

Взвешенную сумму  $S_j$   $j$ -го нейронного элемента слоя  $F_y$  вычислим по формуле

$$S_j = \sum_{i=1}^n x_i w_{ij}. \quad (7.84)$$

Получим выходные значения нейронов слоя  $F_v$ :

$$y_j(t+1) = F(S_j) = \begin{cases} 1, & \text{если } S_j > 0, \\ y_j(t), & \text{если } S_j = 0, \\ -1, & \text{если } S_j < 0, \end{cases}$$

где  $j = \overline{1, p}$ ;  $F$  – оператор нелинейного преобразования. Взвешенную сумму  $i$ -го нейронного элемента слоя  $F_x$  можно представить как

$$S_i = \sum_{j=1}^n y_j w'_{ji}. \quad (7.85)$$

Найдем выходные значения нейронов слоя  $F_x$ :

$$x_i(t+1) = F(S_i) = \begin{cases} 1, & \text{если } S_i > 0, \\ x_i(t), & \text{если } S_i = 0, \\ -1, & \text{если } S_i < 0. \end{cases}$$

В матричной форме выходные значения слоев  $F_y$  и  $F_x$  определяются как

$$Y = F(XW), \quad (7.86)$$

$$X = F(YW^T). \quad (7.87)$$

Емкость памяти можно вычислить при помощи следующего выражения [66]:

$$L = \frac{m}{4\ln(m)}, \quad (7.88)$$

где  $m = \min(n, p)$ .

Устойчивость двунаправленной ассоциативной памяти доказал Коско [61]. Для этого он использовал функцию Ляпунова, которая определяется так же, как и для нейронной сети Хопфилда.

### 7.4.3. Алгоритм функционирования

Алгоритм функционирования двунаправленной ассоциативной памяти можно представить в виде следующих шагов.

1. На входные слои  $F_X$  и  $F_Y$  подаются входные образы  $X$  и  $Y$  или только один из них.
2. Активация нейронных элементов в слое  $F_X$  передается через весовую матрицу  $W$  на слой  $F_Y$ .
3. Вычисляются выходные значения слоя  $F_Y$ .
4. Выходные значения слоя  $F_Y$  через весовую матрицу  $W'$  поступают на слой  $F_X$ .
5. Рассчитываются выходные значения нейронных элементов слоя  $F_X$ .
6. Процедура повторяется, начиная с п. 2, пока все выходные значения нейронных элементов слоев  $F_X$  и  $F_Y$  не перестанут изменяться. Это положение равновесия называется резонансом.

**Пример 7.3.** Пусть имеется две пары входных образов, которые должна запомнить нейронная сеть:

$$\begin{aligned} X_1 &= \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}, Y_1 = \begin{bmatrix} -1 & 1 & -1 & -1 \end{bmatrix}, \\ X_2 &= \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}, Y_2 = \begin{bmatrix} 1 & -1 & 1 & 1 \end{bmatrix}. \end{aligned}$$

В таком случае в соответствии с правилом (7.82) матрица синаптических связей определяется следующим образом:

$$W = X^T Y = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 2 & 2 \\ -2 & 2 & -2 & -2 \\ 2 & -2 & 2 & 2 \end{bmatrix}.$$

Проверим работоспособность сети при распознавании эталонных образов. Подадим на вход  $F_X$  сети образ  $X_1$ . В соответствии с выражениями (4.98), (4.99) на выходах сети получим

$$\begin{aligned} Y &= F(X_1 W) = F([-6 \ 6 \ -6 \ -6]) = [-1 \ 1 \ -1 \ -1] = Y_1, \\ X &= F(Y_1 W^T) = F([-8 \ 8 \ -8]) = [-1 \ 1 \ -1] = X_1. \end{aligned}$$

Отсюда следует, что сеть корректно функционирует в качестве ассоциативной памяти. Рассмотрим функционирование сети при распознавании зашумленных образов. Пусть

$$X_1 = [-1 \ -1 \ -1].$$

Тогда

$$\begin{aligned} Y &= F(X_1 W) = F([-2 \ 2 \ 2 \ -2]) = [-1 \ 1 \ 1 \ -1] = Y_1, \\ X &= F(Y_1 W^T) = F([-4 \ 4 \ -4]) = [-1 \ 1 \ -1] = X_1. \end{aligned}$$

Таким образом, в результате релаксации сеть установилась в состояние равновесия и распознала искаженный образ.

## Глава 8

# САМООРГАНИЗУЮЩИЕСЯ НЕЙРОННЫЕ СЕТИ КОХОНЕНА

*Самоорганизующиеся нейронные сети* (self-organising neural networks) характеризуются обучением без учителя, в результате которого происходит адаптация сети к решаемой задаче. Их разработал в 80-е гг. XX в. финский ученый Т. Кохонен (T. Kohonen) [20, 74–76]. Нейронные сети Кохонена осуществляют топологическое упорядочивание входного пространства образов, поступающих на сеть. Они широко применяются в задачах распознавания и визуализации образов, оптимизации и управления. В данной главе рассматривается архитектура, обучение и функционирование самоорганизующихся нейронных сетей. Приводится алгоритм решения задачи коммивояжера с использованием сети Кохонена.

### 8.1. ОБЩАЯ ХАРАКТЕРИСТИКА СЕТЕЙ КОХОНЕНА

Данные сети осуществляют отображение  $F$  входного  $n$ -мерного пространства образов в выходное  $m$ -мерное пространство, т. е.  $F: R^n \rightarrow R^m$ . При этом обучение здесь происходит без учителя на основе образов, поступающих на сеть. Если сеть осуществляет кластеризацию данных, то  $m$  характеризует количество кластеров, на которые разбивается входное пространство образов. Архитектура нейронной сети в общем случае представляет собой двуслойную нейронную сеть с прямыми связями (рис. 8.1).

Первый слой выполняет чисто распределительные функции, причем каждый нейрон его имеет соединения со всеми нейронными элементами выходного слоя. Второй слой нейронных элементов является обрабатывающим.

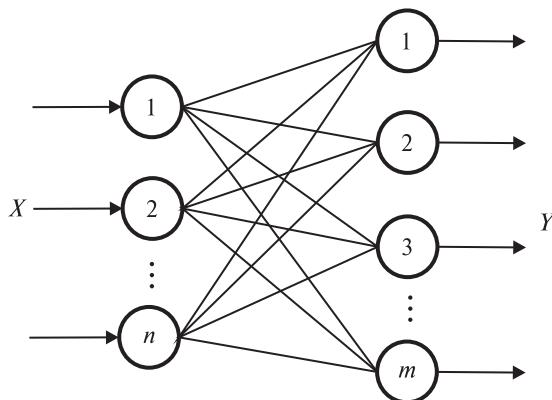


Рис. 8.1. Архитектура нейронной сети Кохонена

Нейронная сеть Кохонена использует *конкурентный принцип* обучения и функционирования. В соответствии с этим принципом при подаче на сеть входного образа значение только одного нейронного элемента выходного слоя принимается равным 1, а выходные значения остальных нейронов – 0. Нейронный элемент, имеющий выходное значение 1, называется победителем в конкурентной борьбе. По мере поступления входных образов на такую сеть посредством обучения происходит разбиение  $n$ -мерного входного пространства на различные области решений, каждой из которых соответствует отдельный нейрон обрабатывающего слоя. Границы отдельной области перпендикулярны линиям, проведенным между центроидами соседних областей решений. Такое разделение пространства называется диаграммой Вороного или картами Кохонена. Для двумерного случая ( $n = 2, m > n$ ) область решений представляет собой правильные шестиугольники (рис. 8.2), в результате чего получается наименьшая ошибка. Для  $n > 2$  наилучшая форма областей решений является неизвестной.

Таким образом, самоорганизация таких сетей происходит в результате топологического упорядочивания входной информации по различным зонам, количество которых равно  $m$ . Такие зоны или области решений называются кластерами. Топологическое упорядочивание информации напоминает процессы, происходящие в головном мозге при его развитии, когда осуществляется формирование топологически упорядоченных нейронных структур.

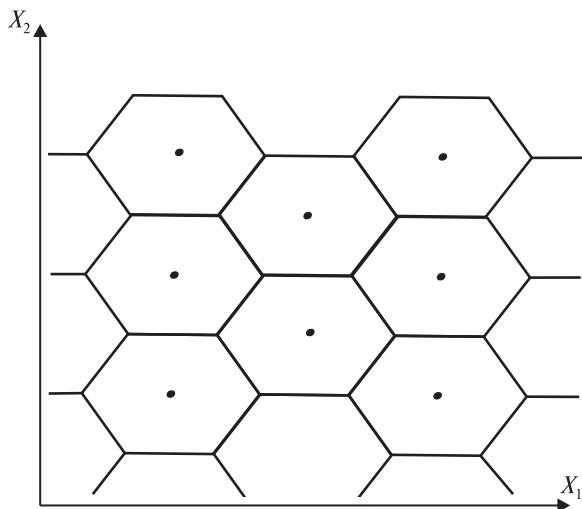


Рис. 8.2. Разбиение входного пространства образов

Самоорганизующиеся нейронные сети применяются для решения различных проблем: распознавание, кластеризация, векторное квантование, выделение характерных признаков и т. д. При кластеризации входные образы группируются в кластеры, причем каждому кластеру ставится в соответствие отдельный нейрон. *Векторное квантование* применяется для сжатия данных.

## 8.2. КОНКУРЕНТНОЕ ОБУЧЕНИЕ

Конкурентное обучение (competitive learning) — основной метод для обучения нейронных сетей Кохонена. Нейронные сети, использующие такой метод обучения, называются *конкурентными*. При конкурентном обучении сеть Кохонена функционирует по принципу «*победитель берет все*» (winner take all). Это означает, что выходное значение только нейрона-победителя с номером  $k$  равно единице, а выходные значения остальных нейронных элементов — нулю:

$$y_j = F(S_j) = \begin{cases} 1, & \text{если } j = k, \\ 0, & \text{если } j \neq k. \end{cases}$$

Для нейрона-победителя в процессе обучения синаптические связи усиливаются, а для остальных нейронов в общем случае не изменяются. Для определения нейрона-победителя применяются следующие методы:

1) по максимальному значению взвешенной суммы нейронных элементов второго слоя;

2) по максимальному значению евклидового расстояния между входным образом и весовыми векторами нейронов обрабатывающего слоя.

### 8.2.1. Определение нейрона-победителя по взвешенной сумме

В этом случае победителем в конкурентной борьбе является такой нейронный элемент с номером  $k$ , который в результате подачи на вход сети определенного образа имеет максимальное значение взвешенной суммы:

$$S_k = \max_j S_j,$$

где взвешенная сумма  $j$ -го нейрона вычисляется как

$$S_j = \sum_i w_{ij} x_i = W_j X^T. \quad (8.1)$$

Здесь  $X = \{x_1, x_2, \dots, x_n\}$  – входной образ,  $W_j = \{w_{1j}, w_{2j}, \dots, w_{nj}\}$  – вектор-столбец весовых коэффициентов  $j$ -го выходного нейрона.

Тогда активность выходных нейронов

$$y_j = F(S_j) = \begin{cases} 1, & \text{если } j = k, \\ 0, & \text{если } j \neq k, \end{cases} \quad (8.2)$$

где  $j = \overline{1, m}$ .

Таким образом, после обучения нейронной сети при подаче входного образа активность нейрона-победителя принимается равной единице, а остальных нейронов – нулю. Выражение (8.1) эквивалентно скалярному произведению вектора весов соответствующего нейронного элемента на входной вектор нейронной сети:

$$S_j = |W_j| \|X\| \cos \alpha, \quad (8.3)$$

где  $\alpha = \hat{W_j} X$ ,  $|W_j|$  и  $\|X\|$  – длины векторов  $W_j$  и  $X$ .

Обозначим  $P = |X| \cdot \cos \alpha$ , где  $P$  – проекция вектора  $X$  на вектор  $W$ :

$$S_j = |W_j| \cdot P. \quad (8.4)$$

Если векторы  $|W_j|$  и  $|X|$  ненормированы, то происходит неадекватное определение нейрона-победителя (рис. 8.3). Как следует из рис. 8.3, нейрон, вектор весов которого  $W_2$  больше отличается от входного образа  $X$ , чем нейрон, имеющий весовой вектор  $W_1$ , становится победителем в конкурентной борьбе.

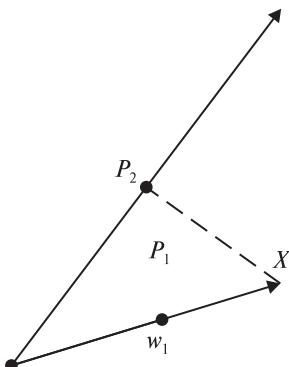


Рис. 8.3. Геометрическая интерпретация определения нейрона-победителя при ненормированных весовом векторе и входном образе

Поэтому при определении нейрона-победителя по взвешенной сумме (8.3) необходимо нормализовать входные или весовые векторы нейронов обрабатывающего слоя. Нормализация осуществляется таким образом, чтобы длина соответствующих векторов вычислялась по следующим формулам:

$$|X| = \sqrt{\sum_i x_i^2} = 1, \quad (8.5)$$

$$|W_j| = \sqrt{\sum_i w_{ij}^2} = 1. \quad (8.6)$$

При нормализации входных и весовых векторов взвешенную активность  $j$ -го нейрона можно представить как

$$S_j = |W_j||X| \cdot \cos \alpha = \cos \alpha. \quad (8.7)$$

Из этого выражения следует, что максимальную активность будет иметь тот нейрон, весовой вектор которого коллинеарен входному вектору. Концы векторов при этом находятся на поверхности  $n$ -мерной сферы (гиперсферы) с радиусом 1 (рис. 8.4).

В выражении (8.7) взвешенная сумма эквивалентна коэффициенту взаимной корреляции между входным и весовым вектором. Он будет иметь значение 1, когда угол между векторами равен нулю. Отсюда следует, что правило настройки весовых коэффициентов нейрона-победителя должно соответствовать вращению вектора  $W_k$  в сторону вектора  $X$  (рис. 8.5).

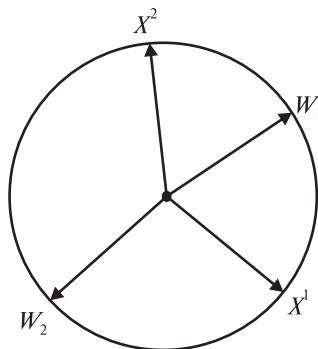


Рис. 8.4. Гиперсфера входных и весовых векторов

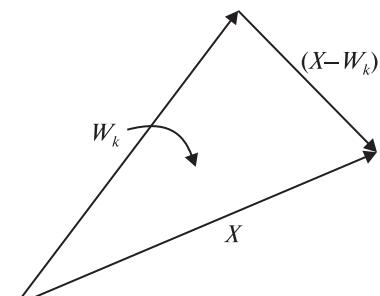


Рис. 8.5. Модификация весового вектора нейрона-победителя

В результате можно записать следующее правило обучения для вектора весов нейрона-победителя:

$$W_k(t+1) = W_k(t) + \gamma(X - W_k(t)), \quad (8.8)$$

где  $0 < \gamma < 1$  характеризует скорость обучения.

В качестве нейрона-победителя выбирается такой нейрон, весовой вектор которого наиболее близок к входному вектору. В обычной форме правило обучения для  $k$ -го нейрона-победителя можно представить как

$$w_{ik}(t+1) = w_{ik}(t) + \gamma(x_i - w_{ik}(t)),$$

где  $i = \overline{1, n}$ .

При применении данного правила к  $k$ -му нейрону усиливается его выходная активность (рис. 8.6).

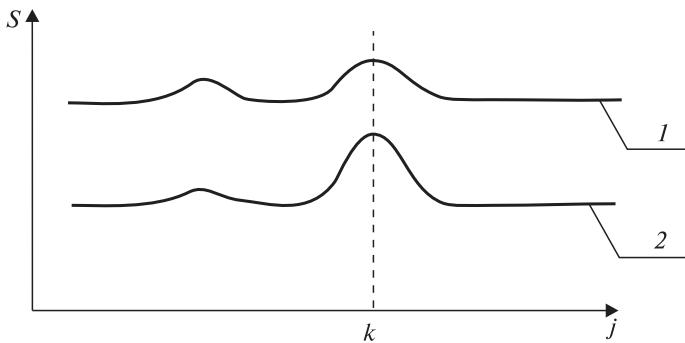
При нормировании весового вектора правило (8.8) изменения весового вектора в процессе обучения модифицируется следующим образом:

$$W_k(t+1) = \frac{W_k(t) + \gamma(X(t) - W_k(t))}{|W_k(t) + \gamma(X(t) - W_k(t))|}. \quad (8.9)$$

В скалярной форме весовые коэффициенты  $k$ -го нейронного элемента определяются как

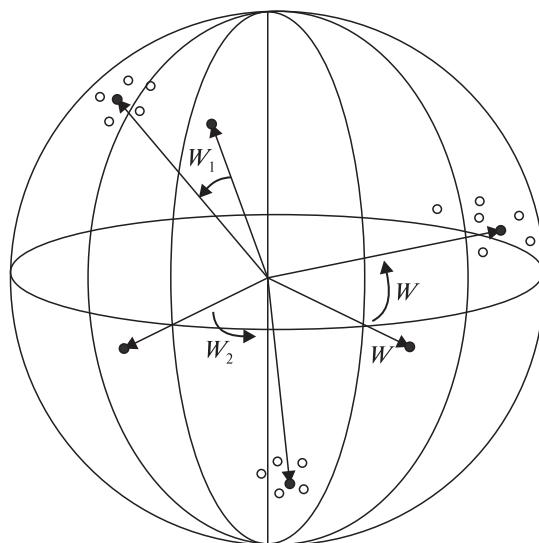
$$w_{ik}(t+1) = \frac{w_{ik}(t) + \gamma_i(x_i(t) - w_{ik}(t))}{|W_k(t) + \gamma(X(t) - W_k(t))|}, \quad (8.10)$$

где  $i = \overline{1, n}$ .



*Рис. 8.6. Профили активности нейронов:*  
 1 – профиль активности нейронов до обучения;  
 2 – профиль активности нейронов после обучения

При применении этого правила для обучения нейронной сети весовые векторы нейронов будут вращаться в направлении кластеров входных образов, как показано на рис. 8.7.



*Рис. 8.7. Изменение весовых векторов в процессе обучения:*  
 $p$  – вектор паттернов;  $W$  – вектор весов

Таким образом, обучение сети Кохонена сводится к настройке весовых векторов нейронных элементов на существующие во входном пространстве образов кластеры. Это эквивалентно вращению весового вектора нейрона-победителя в сторону входного вектора. В процессе обучения весовые векторы нейронов второго слоя настраиваются на центры кластеров.

### 8.2.2. Определение нейрона-победителя по евклидовому расстоянию

В этом случае не накладывается строгих ограничений на нормализацию входных или весовых векторов. Для определения нейрона-победителя необходимо определить евклидово расстояние между входным образом и весовыми векторами нейронов обрабатывающего слоя. Так, для  $j$ -го нейрона второго слоя евклидово расстояние вычисляется по формуле

$$D_j = |X - W_j| = \sqrt{(x_1 - w_{1j})^2 + (x_2 - w_{2j})^2 + \dots + (x_n - w_{nj})^2}. \quad (8.11)$$

Далее находится нейрон-победитель с номером  $k$ , который соответствует минимальному евклидовому расстоянию между входным и весовым вектором:

$$D_k = \min_j |X - W_j|. \quad (8.12)$$

Тогда модификация весового вектора нейрона-победителя происходит следующим образом:

$$W_k(t+1) = W_k(t) + \gamma(X(t) - W_k(t)). \quad (8.13)$$

При применении выражения (8.13) для обучения одного нейронного элемента на интервале входных значений  $[a, b]$ , ( $x \in [a, b]$ ) вес нейрона с течением времени будет стремиться к середине интервала [27].

**Теорема 8.1.** При использовании конкурентного обучения (выражение (8.13)) для обучения нейронных элементов происходит минимизация суммарной квадратичной между входными образами и весовыми векторами соответствующих нейронных элементов.

*Доказательство.* Суммарная квадратичная ошибка для  $k$ -го нейронного элемента определяется следующим образом:

$$E_k = \frac{1}{2} \sum_{l=1}^{L_k} \sum_{i=1}^n (x_i^l - w_{ik})^2,$$

где  $L_k$  – количество входных образов, принадлежащих  $k$ -му кластеру.

Квадратичная ошибка для одного образа в случае  $k$ -го нейрона победителя вычисляется по формуле

$$E = \frac{1}{2} \sum_i (x_i - w_{ik})^2. \quad (8.14)$$

Тогда в соответствии с методом градиентного спуска для последовательного обучения получим

$$w_{ik}(t+1) = w_{ik}(t) - \gamma \frac{\partial E}{\partial w_{ik}(t)}.$$

Найдем градиент

$$\frac{\partial E}{\partial w_{ik}(t)} = -(x_i - w_{ik}).$$

Отсюда

$$w_{ik}(t+1) = w_{ik}(t) + \gamma(x_i - w_{ik}),$$

что и требовалось доказать.

Недостаток описанного выше метода обучения с одним победителем заключается в том, что при случайной инициализации весовых векторов может получиться так, что некоторые нейроны никогда не будут победителями в конкурентной борьбе. Для нейтрализации этого недостатка можно расширить правило обучения следующим образом:

$$W_j(t+1) = W_j(t) + \gamma(X - W_j(t)), \forall j = k,$$

$$W_j(t+1) = W_j(t) + \gamma'(X - W_j(t)), \forall j \neq k, \quad (8.15)$$

где  $\gamma' \ll \gamma$ . Данное правило позволяет отобразить весовые векторы побежденных нейронов в такую область, где увеличиваются их шансы в конкуренции. Другим вариантом является частотно чувствительное конкурентное обучение (sensitive competitive learning). Здесь для каждого нейрона ведется статистика его побед. Пусть  $f_j$  – частота нахождения  $j$ -го нейрона в состоянии победителя. Тогда нейрон-победитель определяется как

$$D_k = \min_j |X - W_j| f_j. \quad (8.16)$$

Чем чаще нейрон становится победителем, тем меньше шансов он имеет в конкуренции.

Итак, при конкурентном обучении все множество входных образов разбивается на кластеры, каждому из которых соответствует свой нейрон. При поступлении на вход нейронной сети неизвестного образа она будет его относить к такому кластеру, на который он больше всего похож. В этом заключается обобщающая способность такого типа нейронных сетей.

### 8.3. ВЕКТОРНЫЙ КВАНТОВАТЕЛЬ

Нейронная сеть для векторного квантования была предложена в 1982 г. Т. Кохоненом [20]. Векторное квантование используется для сжатия данных и основано на идеи сопоставления входного вектора с эталоном [75]. Пусть имеется предварительно сформированное множество эталонных данных, каждое из которых называется кодовым вектором. Совокупность кодовых векторов называется кодовой книгой. При поступлении входного вектора происходит его сравнение с вектором из кодовой книги. В процессе этого выбирается такой кодовый вектор, который наилучшим образом аппроксимирует входной вектор и его номер применяется в качестве кода (рис. 8.8). В качестве меры подобия между входным и эталонными векторами может использоваться евклидово расстояние, а в качестве векторного квантователя – нейронная сеть Кохонена. Тогда целью обучения сети является такая настройка весовых коэффициентов нейрона, которая минимизирует погрешность аппроксимации между входными образами, создающими  $k$ -й кластер и весовыми коэффициентами  $k$ -го нейрона:

$$E_k = \frac{1}{2} \sum_{l=1}^{L_k} \sum_{i=1}^n (x_i^l - w_{ik})^2,$$

где  $E_k$  – ошибка квантования для  $k$ -го кластера.

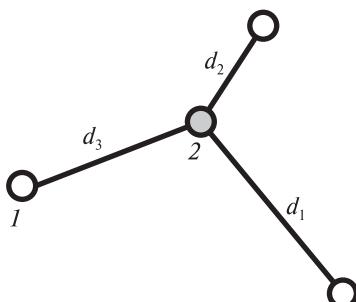


Рис. 8.8. Пример векторного квантования:  
код после квантования равен 10;  
1 – кодовый вектор;  
2 – входной вектор ( $d_2 < d_3 < d_1$ )

Общая ошибка квантования определяется как

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

Нейронную сеть для векторного квантования принято называть обучающимся векторным квантователем (learning vector quantization). Она представляет собой двуслойную сеть с прямыми связями, как было показано на рис. 8.1. В процессе поступления эталонных векторов на сеть она обучается так, что образуются кластеры различных эталонов, каждому из которых соответствует свой нейрон. При поступлении на вход такой нейронной сети неизвестного образа он идентифицируется в соответствии с мерой близости к эталонным векторам и кодируется на выходе сети номером нейрона. Существует большое количество вариантов обучения векторного квантователя [75]. Рассмотрим некоторые из них.

### 8.3.1. Конкурентное обучение с одним победителем

Здесь в отдельный квант времени только один нейрон может быть победителем. Процедура обучения векторного квантователя состоит из следующих шагов:

1. Случайно инициализируются весовые коэффициенты нейронной сети в диапазоне  $[0,1]$ .

2. Задается начальное значение момента времени  $t = 0$ .

3. Входные образы последовательно подаются на нейронную сеть  $x^l$ ,  $l = \overline{1, L}$ , и для каждого образа производятся вычисления:

а) вычисляется евклидово расстояние между входным образом и весовыми векторами нейронов выходного слоя:

$$D_j = |X - W_j| = \sqrt{(x_1 - w_{1j})^2 + (x_2 - w_{2j})^2 + \dots + (x_n - w_{nj})^2},$$

где  $j = \overline{1, m}$ ;

б) определяется нейрон-победитель, обеспечивающий минимальное расстояние:

$$D_k = \min_j D_j;$$

в) производится модификация весовых коэффициентов нейронного элемента победителя, а весовые коэффициенты остальных нейронов не изменяются:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \gamma(t)(x_i - w_{ij}(t)), \text{ если } j = k, \\ w_{ij}(t+1) &= w_{ij}(t), \text{ если } j \neq k, \end{aligned}$$

где  $i = \overline{1, n}$ ;  $j = \overline{1, m}$ .

4. Изменяется значение времени  $t = t + 1$  и процесс повторяется, начиная с п. 3.

Обучение производится до получения желаемой степени согласования между входными и весовыми векторами или до тех пор, пока не перестанут изменяться весовые коэффициенты. Для остановки процесса обучения можно использовать следующие правила:

а) чтобы весовые коэффициенты в процессе обучения перестали изменяться, шаг обучения  $\gamma(t)$  должен уменьшаться с течением времени, например, по следующему закону:

$$\gamma(t) = \gamma_0 e^{-\frac{t}{c}},$$

где  $\gamma_0 = 0,1$ ;  $c = 1000$ ;

б) обычно, как показывает практика, рекомендуется выбирать общее количество эпох обучения

$$t = (50 \div 200).$$

### 8.3.2. Конкурентное обучение со многими победителями

Для того чтобы похожие кластеры отображались на соседние нейроны второго слоя, можно использовать конкурентное обучение со многими победителями. В этом случае вокруг нейрона-победителя формируются соседние нейроны, для которых также производится модификация весовых коэффициентов. Для этого вводится специальная функция притяжения, определяющая область притяжения нейрона-победителя в конкурентной борьбе. Значение функции притяжения для  $p$ -го нейронного элемента, не являющегося победителем в конкурентной борьбе, можно определить в соответствии с функцией Гаусса:

$$h(t, k, p) = e^{\frac{-|k-p|^2}{2\sigma^2(t)}}, \quad (8.16)$$

где  $p$  – номер нейронного элемента, для которого определяется значение функции притяжения;  $k$  – номер нейронного элемента победителя;  $\sigma(t)$  – среднеквадратичное отклонение (радиус области притяжения), уменьшающееся с течением времени по следующему закону:

$$\sigma(t) = \sigma_0 e^{-\frac{t}{c}}.$$

Значения переменных в последнем выражении можно выбирать следующим образом:

$$c = \frac{1000}{\log_2 \sigma_0},$$

где  $\sigma_0 = m/2$ .

Тогда модификация весовых коэффициентов производится для всех нейронных элементов обрабатывающего слоя в соответствии со значением функции притяжения:

$$w_{ip}(t+1) = w_{ip}(t) + \gamma h(t, k, p)(x_i - w_{ip}).$$

В дискретном варианте вводится область притяжения  $G$ , которая определяет нейронные элементы, принадлежащие классу победителей. В этом случае весовые коэффициенты изменяются для всех нейронов, принадлежащих области  $G$ :

$$\Delta w_{ip} = \begin{cases} \gamma(t)(x_i - w_{ip}), & \text{если } p \in G, \\ 0, & p \notin G. \end{cases} \quad (8.17)$$

### 8.3.3. Контролируемое конкурентное обучение

Если заранее известно соответствие эталонных векторов нейронным элементам, то используется контролируемое конкурентное обучение (Supervised Competitive Learning). Для такого обучения весовые коэффициенты нейрона-победителя усиливаются при корректной классификации, т. е. когда входной образ соответствует заданному номеру нейронного элемента второго слоя и ослабляются в противном случае. Тогда

$$w_{ik}(t+1) = w_{ik}(t) + \gamma(x_i - w_{ik}(t)) \quad (8.18)$$

при корректной классификации, когда  $X$  и  $W_k$  принадлежат к одному классу и

$$w_{ik}(t+1) = w_{ik}(t) - \gamma(x_i - w_{ik}(t)), \quad (8.19)$$

если  $X$  и  $W_k$  принадлежат к различным классам.

Весовые коэффициенты остальных нейронов при этом не изменяются. После обучения нейронная сеть может осуществлять функции векторного квантования. При этом на выходе в каждый квант времени будет активным только один нейрон (его выход равен 1), а остальные нейроны будут иметь нулевые выходные значения.

#### 8.4. САМООРГАНИЗУЮЩИЕСЯ КАРТЫ КОХОНЕНА

Самоорганизующиеся карты Кохонена (self organizing maps) являются дальнейшим расширением нейронных сетей с конкурентным обучением [75], разработанных Кохоненом в 1982 г. Топология нейронной сети Кохонена состоит из двух слоев. Первый слой содержит  $n$  нейронных элементов и выполняет распределительные функции, а нейроны второго слоя расположены на плоскости, образуя матрицу размерностью  $m \times m$  (рис. 8.9).

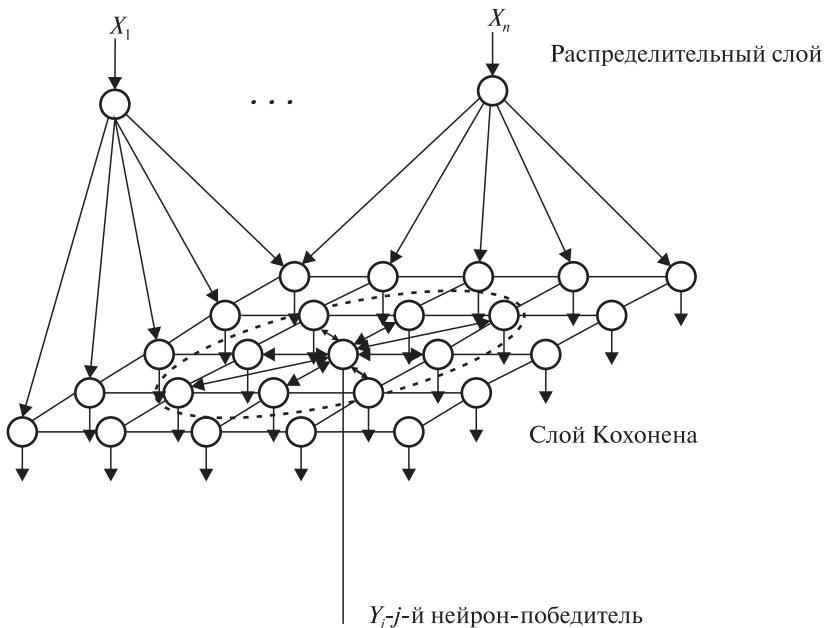


Рис. 8.9. Топология нейронной сети Кохонена

В своем простейшем виде сеть Кохонена функционирует по принципу «победитель берет все». При этом она должна выполнять топологически упорядоченное отображение входных векторов на матрицу нейронов второго слоя. Соседние наиболее похожие входные образы должны отображаться на соседние нейроны матрицы. Для этого применяется конкурентное обучение со многими победителями при помощи введения области притяжения  $G$  для нейрона-победителя, в радиусе действия которой нейроны активно изменяют свои весовые векторы в сторону входного образа. Область притяжения, как уже отмеча-

лось, можно описать функцией притяжения  $h(t, k, p)$ . В дискретном варианте функция притяжения определяется следующим образом:

$$h(t, k, p) = \begin{cases} 1, & \text{если } p \in G, \\ 0, & \text{если } p \notin G. \end{cases} \quad (8.20)$$

В область притяжения нейрона  $k$  входят все нейроны, находящиеся на некотором расстоянии от нейрона-победителя. В непрерывном варианте используется функция Гаусса (рис. 8.10).

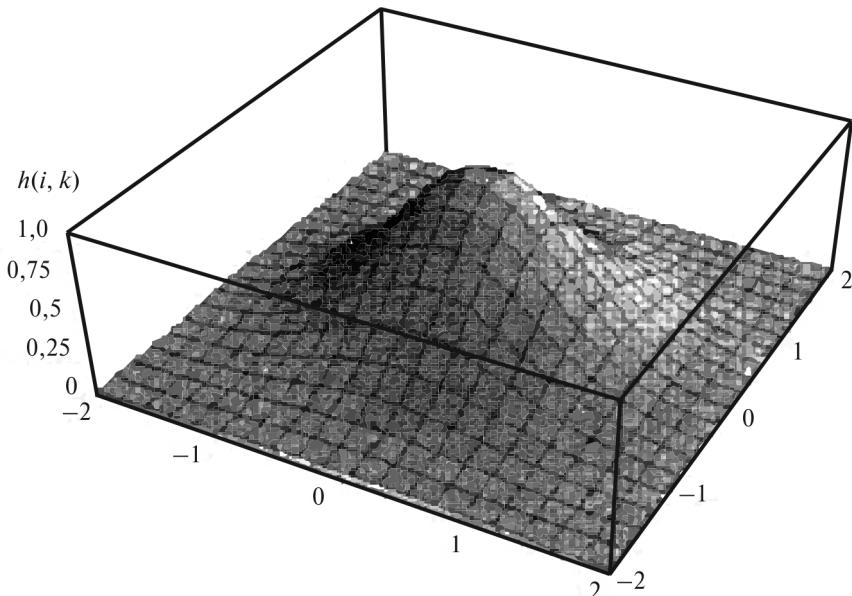


Рис. 8.10. Двумерная функция Гаусса

Она определяется при помощи следующего выражения:

$$h(t, k, p) = e^{\frac{-|u_k - u_p|^2}{2\sigma^2(t)}}, \quad (8.21)$$

где  $|u_k - u_p|$  – евклидово расстояние между нейронами;  $\sigma(t)$  – среднеквадратичное отклонение (радиус области притяжения).

Положение каждого нейрона в матрице характеризуется его координатами

$$u_k = (i_k, j_k), u_p = (i_p, j_p). \quad (8.22)$$

Тогда

$$|u_k - u_p|^2 = (i_k - i_p)^2 + (j_k - j_p)^2. \quad (8.23)$$

В процессе обучения нейронной сети Кохонена изменяются весовые коэффициенты не только нейрона-победителя, но и всех нейронов внутри области притяжения. Так, для нейрона с координатами  $p = (i, j)$  весовые коэффициенты модифицируются по следующему закону:

$$w_{cij}(t+1) = w_{cij}(t) + \gamma(t)h(t, k, p)(x_c - w_{cij}(t)). \quad (8.24)$$

С увеличением времени обучения радиус области притяжения уменьшается. В результате нейронные элементы сжимаются около нейрона-победителя, пока он не останется один. Это схематично изображено на рис. 8.11, где  $G(t)$  – область притяжения в момент времени  $t$ . Введем декартову систему координат так, что каждый нейрон имеет координаты  $(i, j)$ , где  $i = \overline{1, m}$ ;  $j = \overline{1, m}$  (рис. 8.11).

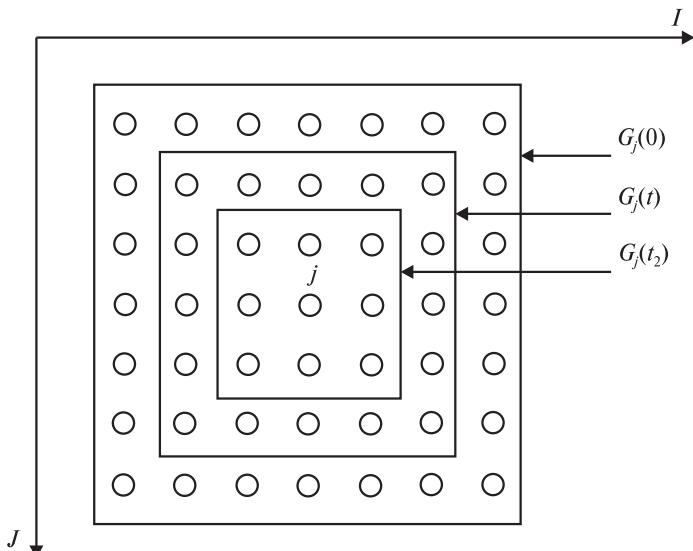


Рис. 8.11. Изменение области притяжения с течением времени

Поставим в соответствие нейрону с координатами  $(i, j)$  весовой вектор  $W_{ij}$ , который определяется следующим образом:

$$W_{ij} = (w_{1ij}, w_{2ij}, \dots, w_{nij}), \quad (8.25)$$

где  $n$  – количество нейронных элементов входного слоя.

Тогда процедуру обучения сети Кохонена для непрерывной функции притяжения можно представить в виде следующих действий:

1. Случайным образом инициализируются весовые коэффициенты  $W$  нейронной сети.

2. Задается начальное значение радиуса притяжения и момент времени  $t = 0$ .

3. Подается входной образ на нейронную сеть, и для каждого нейронного элемента матрицы вычисляется евклидово расстояние между входным образом и весовыми векторами нейронных элементов:

$$D_{ij} = |X - W_{ij}| = \sqrt{(x_1 - w_{1ij})^2 + (x_2 - w_{2ij})^2 + \dots + (x_n - w_{nij})^2},$$

где  $i = \overline{1, m}$ ,  $j = \overline{1, m}$ ;  $W_{ij}$  – весовой вектор нейрона с координатами  $(i, j)$ .

4. Определяется нейрон-победитель:

$$D(i_k, j_k) = \min_{i,j} D_{ij},$$

где  $k = (i_k, j_k)$  – координаты нейрона-победителя.

5. Для каждого нейрона производится вычисление функции притяжения:

$$h(t, k, p) = e^{-\frac{|u_k - u_p|^2}{2\sigma^2(t)}},$$

где  $p = \overline{1, m^2}$ .

6. В соответствии с функцией притяжения осуществляется модификация весовых коэффициентов:

$$w_{cij}(t+1) = w_{cij}(t) + \gamma(t)h(t, k, p)(x_c - w_{cij}(t)),$$

где  $i = \overline{1, m}$ ,  $j = \overline{1, m}$ ,  $c = \overline{1, n}$ .

7. Повторяется процедура, указанная в п. 3, для всех входных образов.

8. Увеличивается на единицу квант времени, уменьшается радиус области притяжения и процесс повторяется, начиная с п. 3.

Обучение производится до получения желаемой степени согласования между весовыми и выходными векторами. Начальное значение области притяжения может охватывать всю матрицу нейронов, а затем последовательно уменьшается, как показано на рис. 8.11. Для дискретной функции притяжения процедура обучения является аналогичной. Функционирование такой сети происходит путем определения нейрона-победителя и присвоения ему единичного значения, а остальным нейронам – нулевого значения.

В процессе обучения происходит упорядочение весовых коэффициентов таким образом, что уменьшается разница между весами соседних нейронов. Покажем это. Пусть весовые векторы соседних нейронов изменяются следующим образом:

$$W_1(t+1) = W_1(t) + \gamma(x(t) - W_1(t)) = (1-\gamma)W_1(t) + \gamma X(t),$$

$$W_2(t+1) = W_2(t) + \gamma(x(t) - W_2(t)) = (1-\gamma)W_2(t) + \gamma X(t).$$

Вычислим расстояние между ними:

$$|W_1(t+1) - W_2(t+1)| = |(1-\gamma)(W_1(t) - W_2(t))|.$$

Поскольку  $0 < \gamma < 1$ , то  $|W_1(t+1) - W_2(t+1)| < |W_1(t) - W_2(t)|$ .

Таким образом, в процессе обучения разница между весовыми векторами топологически близких нейронов уменьшается. Это показано на рис. 8.12 для сети с двумя входными нейронами и  $8 \times 8$  выходными нейронами.

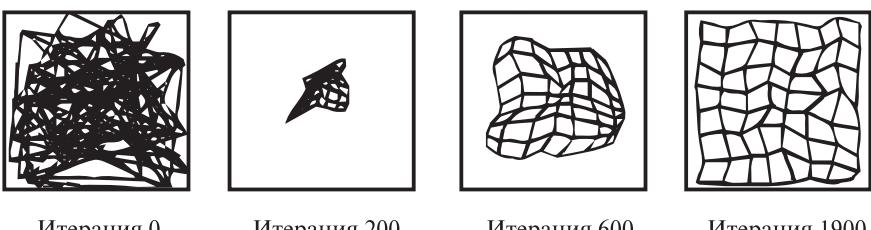


Рис. 8.12. Изменение весовых коэффициентов сети Кохонена

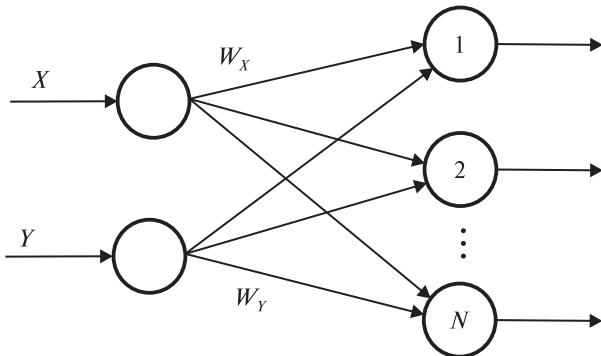
Линии соединяют значения весов нейрона с координатами  $(i, j)$  с весами нейрона  $(i+1, j)$  и  $(i, j+1)$ . При  $t=0$  веса выбраны случайно и беспорядочно распределены на плоскости. Затем веса изменяются, так что их плотность приблизительно соответствует плотности вероятности входных векторов.

## 8.5. РЕШЕНИЕ ЗАДАЧИ КОММИВОЯЖЕРА

Пусть количество городов, которое должен обогнать коммивояжер, будет  $N$ . При использовании полного перебора общее количество вариантов, которое необходимо просмотреть для решения данной задачи, равно  $(N-1)!$ .

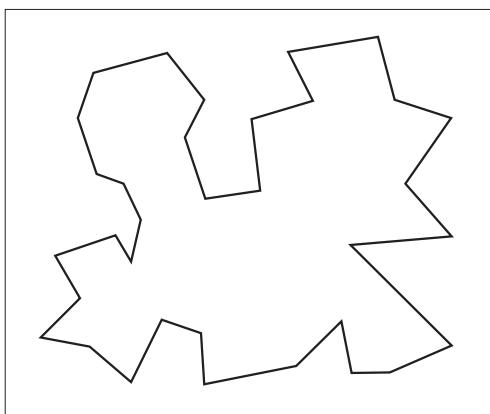
Также известны координаты каждого города  $(x, y)$ , которые являются входными данными для нейронной сети Кохонена. Таким образом,

совокупность координат всех городов образует обучающее множество. Архитектура нейронной сети состоит из двух входных нейронов, которым соответствуют координаты  $x$  и  $y$ , и из  $N$  нейронов слоя Кохонена. Нейроны слоя Кохонена образуют одномерную цепочку (рис. 8.13).



*Рис. 8.13. Архитектура сети Кохонена для решения задачи коммивояжера*

Функционирование такой нейронной сети для решения задачи коммивояжера основывается на том, что при обучении города, расположенные по соседству друг с другом, будут отображаться на соседние нейроны слоя Кохонена. После обучения сети города должны образовывать замкнутую цепочку (рис. 8.14).



*Рис. 8.14. Формирование замкнутого маршрута сетью Кохонена*

В соответствии с этим введем следующую функцию притяжения между нейронами в слое Кохонена:

$$r(i, j) = \exp\left(\frac{-\text{dist}^2(i, j)}{2\sigma^2(t)}\right),$$

где  $\sigma(t)$  – среднеквадратичное отклонение;  $r(i, j)$  – функция притяжения, которая характеризует топологическое расстояние между  $i$ -м и  $j$ -м нейронами;  $\text{dist}(i, j)$  – топологическое расстояние между нейронами  $i$  и  $j$  в замкнутой цепочке (рис. 8.15)

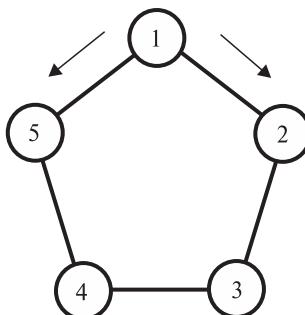


Рис. 8.15. Пример замкнутого маршрута для пяти городов

Для определения  $\text{dist}(i, j)$  необходимо вначале вычислить топологическое расстояние между  $i$ -м и  $j$ -м нейронами, двигаясь по окружности как по часовой, так и против часовой стрелки [27]. Затем необходимо взять наименьшее получающееся число, которое и является искомым топологическим расстоянием между соответствующими нейронами. Топологическое расстояние при движении по окружности в направлении часовой стрелки определяется следующим образом:

$$P \downarrow = |k - i - N| \bmod N, \quad (8.26)$$

при движении против часовой стрелки –

$$P \uparrow = |k - i + N| \bmod N, \quad (8.27)$$

где  $k$  – номер нейрона-победителя.

Топологическое расстояние между  $k$ -м и  $i$ -м нейронами вычислим как

$$\text{dist}(i, k) = \min(P \downarrow, P \uparrow). \quad (8.28)$$

В таблице приведен пример определения топологического расстояния для цепочки нейронов, состоящих из пяти городов, при  $k = 1$  (см. рис. 8.15).

$i$	$P \downarrow$	$P \uparrow$	$\text{dist}(i,k)$
1	0	0	0
2	1	4	1
3	2	3	2
4	3	2	2
5	4	1	1

Алгоритм обучения сети Кохонена для решения задачи коммивояжера состоит из следующих шагов:

1. Случайным образом инициализируются весовые коэффициенты нейронов в слое Кохонена  $W_x$  и  $W_y$ .

2. Из всей совокупности городов выбирается случайным образом один город, координаты которого подаются на вход нейронной сети.

3. Для каждого нейрона слоя Кохонена определяется евклидово расстояние между координатами выбранного города и соответствующими весовыми коэффициентами:

$$D_j = (x - w_{xj})^2 + (y - w_{yj})^2, \quad j = \overline{1, N}.$$

4. Определяется нейрон-победитель, обеспечивающий наименьшее евклидово расстояние:

$$D_k = \min_j \{D_j\},$$

где  $k$  – номер нейрона-победителя.

5. В соответствии с функцией притяжения  $r(j, k)$  модифицируются весовые коэффициенты нейронной сети:

$$w_{xj}(t+1) = w_{xj}(t) + r(j, k)\gamma(t)(x - w_{xj}(t)),$$

$$w_{yj}(t+1) = w_{yj}(t) + r(j, k)\gamma(t)(y - w_{yj}(t)),$$

где  $j = \overline{1, N}$ .

6. Указанная в п. 2 процедура повторяется для всех городов.

7. Уменьшаются значения  $\gamma(t)$ ,  $\sigma(t)$ , и процесс повторяется, начиная с п. 2.

Данная процедура продолжается до тех пор, пока не перестанут изменяться весовые коэффициенты нейронной сети. После обучения каждому нейрону слоя Кохонена будет поставлен в соответствие определенный город, которые образуют замкнутый маршрут. Это и будет являться решением задачи коммивояжера.

## Глава 9

# ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ

В данной главе рассматриваются и анализируются глубокие нейронные сети, которые считаются революционным шагом в области интеллектуальной обработки данных. Данные сети успешно применяются для решения различных проблем в области искусственного интеллекта, таких как обработка и распознавание речи, образов, естественного языка, визуализации данных и т. д. [22–24, 29, 76–90]. В общем случае *глубокие нейронные сети* осуществляют глубокое иерархическое преобразование входного пространства образов и представляют собой нейронные сети с множеством слоев нейронных элементов. Существуют следующие глубокие нейронные сети (DNN):

- нейронные сети глубокого доверия (deep belief neural networks);
- глубокий персептрон (deep perceptron);
- глубокие сверточные нейронные сети различных типов (deep convolutional neural networks): R – CNN, Fast – CNN, Faster – CNN, SSD, ResNet и т. д.;
- глубокая рекуррентная нейронная сеть (deep recurrent neural networks);
- глубокий автоэнкодер (deep autoencoder);
- глубокая рекуррентная-сверточная нейронная сеть (deep RCNN).

Исторически первыми появились нейронные сети глубокого доверия и глубокий персептрон, которые в общем случае представляют собой многослойный персептрон с более чем двумя скрытыми слоями [29]. Основное отличие нейронной сети глубокого доверия от глубокого персептрана заключается в том, что нейронная сеть глубокого доверия в общем случае не является сетью с прямым распространением сигнала (feed forward neural network). До 2006 г. в научной среде была приоритетной парадигма, что многослойный персептрон с одним, максимум двумя скрытыми слоями более эффективен для нелинейного преобразования входного пространства образов в выходное по сравнению с персептраном с большим количеством скрытых слоев. Считалось, что не имеет смысла применять персептрон с более

чем двумя скрытыми слоями. Данная парадигма базировалась на теореме, что персептрон с одним скрытым слоем – универсальный аппроксиматор. Другой аспект этой проблемы заключается в том, что все попытки применять алгоритм обратного распространения ошибки (backpropagation algorithm) для обучения персептрона с тремя и более скрытыми слоями не приводили к улучшению решения различных задач. Это связано с тем, что алгоритм обратного распространения ошибки неэффективен для обучения персептронов с тремя и более скрытыми слоями при использовании сигмоидной функции активации, что происходит из-за проблемы *исчезающего градиента* (vanishing gradient problem). Так, например, максимальное значение производной сигмоидной функции активации  $F(S_j) = 0,25$ . Поэтому использование обобщенного дельта-правила для обучения персептрана с большим количеством скрытых слоев приводит к затуханию градиента при распространении сигнала от последнего к первому слою. В 2006 г. Дж. Хинтон предложил «жадный» алгоритм послойного обучения (greedy layer-wise algorithm) [22], который стал эффективным средством обучения глубоких нейронных сетей. Было показано, что глубокая нейронная сеть имеет большую эффективность нелинейного преобразования и представления данных по сравнению с традиционным персептроном. Такая сеть осуществляет глубокое иерархическое преобразование входного пространства образов. В результате первый скрытый слой выделяет низкоуровневое пространство признаков входных данных, второй слой детектирует пространство признаков более высокого уровня абстракции и т. д. [23].

## 9.1. АРХИТЕКТУРА ГЛУБОКОЙ НЕЙРОННОЙ СЕТИ

Как уже отмечалось, глубокая нейронная сеть содержит множество скрытых слоев нейронных элементов (рис. 9.1) и осуществляет глубокое иерархическое преобразование входного пространства образов.

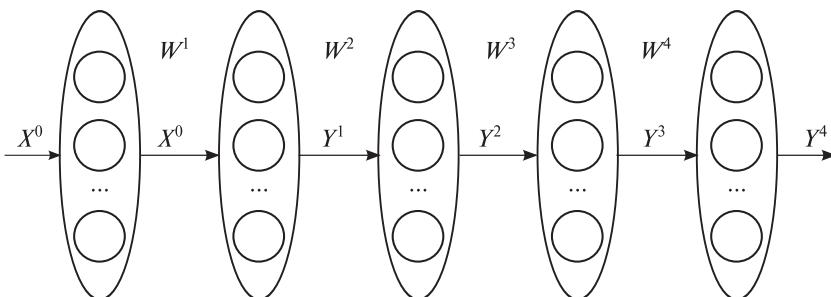


Рис. 9.1. Глубокая нейронная сеть

Выходное значение  $j$ -го нейрона  $k$ -го слоя определяется следующим образом:

$$y_j^k = F(S_j^k), \quad (9.1)$$

$$S_j^k = \sum_{i=1} w_{ij}^k y_i^{k-1} + T_j^k, \quad (9.2)$$

где  $F$  – функция активации нейронного элемента;  $S_j^k$  – взвешенная сумма  $j$ -го нейрона  $k$ -слоя;  $w_{ij}^k$  – весовой коэффициент между  $i$ -м нейроном  $(k-1)$ -го слоя и  $j$ -м нейроном  $k$ -го слоя;  $T_j^k$  – пороговое значение  $j$ -го нейрона  $k$ -го слоя.

Для первого (распределительного) слоя

$$y_i^0 = x_i. \quad (9.3)$$

В матричном виде выходной вектор  $k$ -го слоя

$$Y^k = F(S^k) = F(W^k Y^{k-1} + T^k), \quad (9.4)$$

где  $W$  – матрица весовых коэффициентов;  $Y^{k-1}$  – выходной вектор  $(k-1)$ -го слоя;  $T^k$  – вектор пороговых значений нейронов  $k$ -го слоя.

Если глубокая нейронная сеть используется для классификации образов, то выходные значения сети часто определяются на основе функции активации softmax:

$$y_j^F = \text{softmax}(S_j) = \frac{e^{S_j}}{\sum_l e^{S_l}}.$$

Несмотря на архитектурные различия глубоких нейронных сетей, *принципы их обучения являются идентичными*. Поэтому рассмотрим основные концепции обучения таких сетей на примере глубокого персептрона.

## 9.2. ОБУЧЕНИЕ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

Существуют два основных метода обучения.

1. *Метод с предварительным обучением*, который состоит из двух этапов:

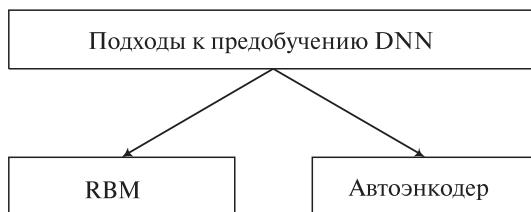
- предобучение нейронной сети методом послойного обучения, начиная с первого слоя (pre-training). Данное обучение осуществляется без учителя и базируется на ограниченной машине Больцмана (RBM);

- настройка синаптических связей всей сети (fine-tuning) при помощи алгоритма обратного распространения ошибки или алгоритма «бодрствования и сна» (wake-sleep algorithm).

2. *Метод стохастического градиента* (SGD) с ректификационной функцией активации (ReLU) нейронных элементов.

В настоящее время принята следующая парадигма для обучения глубоких нейронных сетей. Если обучающая выборка большая, т. е. размерность обучающей выборки намного больше, чем количество настраиваемых параметров сети, то используется метод стохастического градиента с функцией активации ReLU нейронных элементов. Если размерность обучающей выборки сравнима с количеством настраиваемых параметров сети, то применяется предварительное обучение нейронной сети на основе RBM и алгоритм обратного распространения ошибки для точной настройки синаптических связей сети (fine-tuning).

Важным этапом обучения глубоких нейронных является предобучение слоев нейронной сети. Существует два основных подхода к предварительному обучению слоев глубоких нейронных сетей (рис. 9.2).



*Рис. 9.2. Методы предварительного обучения глубоких нейронных сетей*

Первый подход называется автоэнкодерным и базируется на представлении каждого слоя в виде автоассоциативной нейронной сети. Второй подход основан на представлении каждого слоя нейронной сети в виде ограниченной машины Больцмана.

### 9.3. АВТОЭНКОДЕРНЫЙ МЕТОД ОБУЧЕНИЯ

Данный подход базируется на представлении каждого слоя в виде *автоассоциативной нейронной сети*. В этом случае вначале обучается первый слой как автоассоциативная нейронная сеть в целях минимизации суммарной квадратичной ошибки реконструкции информации,

затем второй и т. д. Для обучения каждого слоя можно применять алгоритм обратного распространения ошибки. После этого осуществляется точная настройка синаптических связей всей сети (fine tuning) с использованием алгоритма обратного распространения ошибки.

Рассмотрим персептрон с тремя скрытыми слоями (рис. 9.3). Тогда в соответствии с автоэнкодерным методом прежде всего берутся первые два слоя нейронной сети (*1* и *2*) и на их основе конструируется автоассоциативная (PCA-сеть) нейронная сеть (*1* – *2* – *1*), т. е. добавляется восстановливающий слой (рис. 9.4, *a*), а затем происходит обучение, например при помощи алгоритма обратного распространения ошибки такой сети в целях минимизации квадратичной ошибки реконструкции информации. Продолжительность обучения обычно составляет не больше 100 эпох.

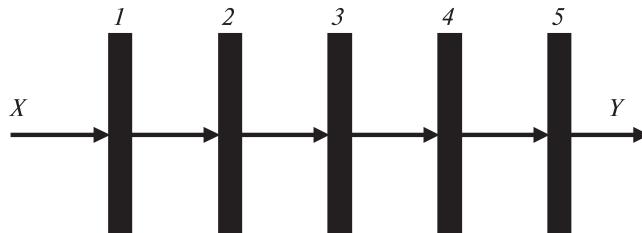


Рис. 9.3. Персептрон с тремя скрытыми слоями

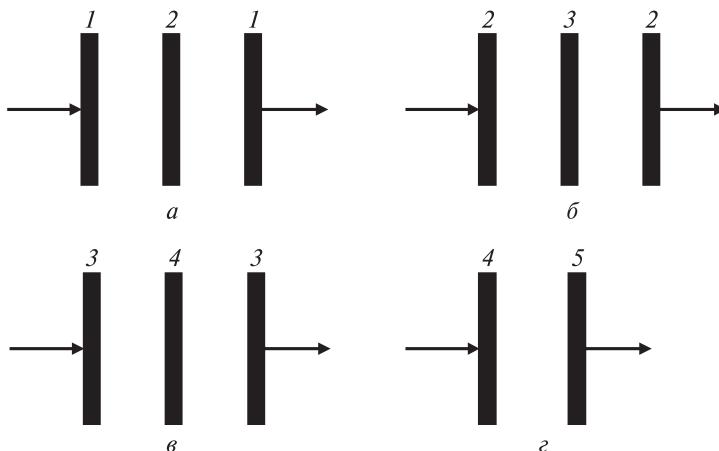


Рис. 9.4. Автоэнкодерный метод обучения:  
*а* – 1-го и 2-го слоя; *б* – 2-го и 3-го слоя; *в* – 3-го и 4-го слоя; *г* – 4-го и 5-го слоя

После этого отбрасывается восстановливающий слой (последний), фиксируются веса скрытого слоя и конструируется автоассоциативная сеть из следующих двух слоев нейронной сети  $2 - 3 - 2$  (рис. 9.4, б), которая обучается на основе данных поступающих с предыдущего (2-го слоя). Процесс продолжается до последнего (рис. 9.4, г) или предпоследнего (рис. 9.4, в) слоя. В результате послойного обучения получается предварительно обученная нейронная сеть. Далее осуществляется точная настройка (fine tuning) посредством, например, алгоритма обратного распространения ошибки с учителем.

Данный процесс можно представить в виде следующего алгоритма:

1. Конструируется автоассоциативная сеть с входным слоем  $X$ , скрытым  $Y$  и выходным слоем  $X$ .
2. Обучается автоассоциативная сеть, например при помощи алгоритма обратного распространения ошибки (как правило, не более 100 эпох), и фиксируются синаптические связи первого слоя  $W^1$ .
3. Берется следующий слой, и аналогичным образом формируется автоассоциативная сеть.
4. С использованием настроенных синаптических связей предыдущего слоя  $W^1$  входные данные подаются на вторую автоассоциативную сеть, и она обучается аналогичным образом. В результате получаются весовые коэффициенты второго слоя  $W^2$ .
5. Процесс продолжается до последнего слоя нейронной сети.
6. Обучается вся сеть для точной настройки параметров при помощи алгоритма обратного распространения ошибки.

## 9.4. ОБУЧЕНИЕ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ НА ОСНОВЕ RBM

Как уже отмечалось, данный подход базируется на представлении каждого слоя нейронной сети в виде *ограниченной машины Больцмана*. Ограниченная машина Больцмана состоит из двух слоев стохастических бинарных нейронных элементов, которые соединены между собой двунаправленными симметричными связями (рис. 9.5). Входной слой нейронных элементов называется видимым (слой  $X$ ), а второй слой — скрытым (слой  $Y$ ). Глубокую нейронную сеть можно представить как совокупность ограниченных машин Больцмана. Ограниченная машина Больцмана может аппроксимировать (генерировать) любое дискретное распределение, если используется достаточное количество нейронов скрытого слоя [80].

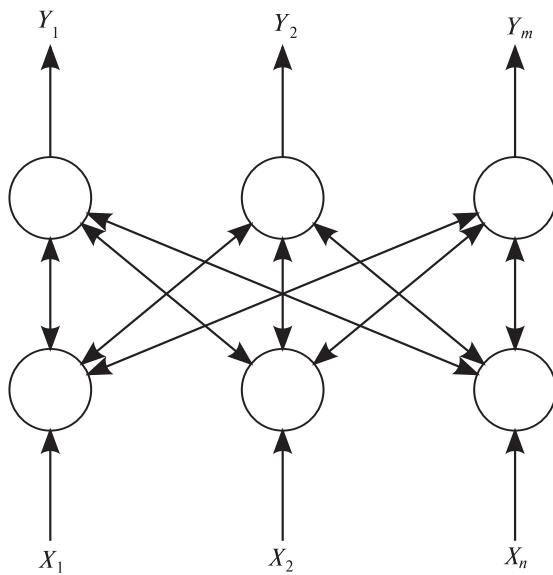


Рис. 9.5. Ограниченнная машина Больцмана

Данная сеть является *стохастической нейронной сетью*, в которой состояния видимых и скрытых нейронов меняются в соответствии с вероятностной версией сигмоидной функции активации:

$$p(y_j | x) = \frac{1}{1 + e^{-S_j}}, \quad S_j = \sum_{i=1}^n w_{ij} x_i + T_j, \quad (9.5)$$

$$p(x_i | x) = \frac{1}{1 + e^{-S_i}}, \quad S_i = \sum_{j=1}^m w_{ij} y_j + T_i. \quad (9.6)$$

Состояния видимых и скрытых нейронных элементов принимаются независимыми:

$$P(x | y) = \prod_{i=1}^n P(x_i | y),$$

$$P(y | x) = \prod_{j=1}^m P(y_j | x).$$

Таким образом, состояния всех нейронных элементов ограниченной машины Больцмана определяются через распределение вероятно-

стей. В RBM нейроны скрытого слоя – это детекторы признаков, которые обуславливают закономерности входных данных. Основная задача обучения состоит в воспроизведении распределения входных данных на основе состояний нейронов скрытого слоя как можно точнее. Это эквивалентно максимизации функции правдоподобия путем модификации синаптических связей нейронной сети. Рассмотрим это подробнее.

Вероятность нахождения видимого и скрытого нейрона в состоянии  $(x, y)$  определяется на основе распределения Гиббса:

$$P(x, y) = \frac{e^{-E(x, y)}}{Z},$$

где  $E(x, y)$  – энергия системы в состоянии  $(x, y)$ ;  $Z$  – параметр условия нормализации вероятностей, сумма вероятностей должна равняться единице. Данный параметр вычисляется следующим образом:

$$Z = \sum_{x, y} e^{-E(x, y)}.$$

Вероятность нахождения видимых нейронов в определенном состоянии равна сумме вероятностей конфигураций  $P(x, y)$  по состояниям скрытых нейронов:

$$P(x) = \sum_y P(x, y) = \sum_y \frac{e^{-E(x, y)}}{Z} = \frac{\sum_y e^{-E(x, y)}}{\sum_{x, y} e^{-E(x, y)}}.$$

Для нахождения правила модификации синаптических связей необходимо максимизировать вероятность воспроизведения состояний видимых нейронов  $P(x)$  ограниченной машиной Больцмана. Для того чтобы определить максимум функции правдоподобия распределения данных  $P(x)$ , применим метод градиентного спуска в пространстве весовых коэффициентов и пороговых значений сети, где в качестве градиента возьмем функцию логарифмического правдоподобия:

$$\ln P(x) = \ln \sum_y e^{-E(x, y)} - \ln \sum_{x, y} e^{-E(x, y)}.$$

Тогда градиент равен

$$\frac{\partial \ln(P(x))}{\partial \omega_{ij}} = \frac{\partial}{\partial \omega_{ij}} \left( \ln \sum_y e^{-E(x, y)} \right) - \frac{\partial}{\partial \omega_{ij}} \left( \ln \sum_{x, y} e^{-E(x, y)} \right).$$

Преобразуя последнее выражение, получим

$$\frac{\partial \ln P(x)}{\partial \omega_{ij}} = -\frac{1}{\sum_y e^{-E(x,y)}} \sum_y e^{-E(x,y)} \frac{\partial E(x,y)}{\partial \omega_{ij}} + \frac{1}{\sum_{x,y} e^{-E(x,y)}} \sum_{x,y} e^{-E(x,y)} \frac{\partial E(x,y)}{\partial \omega_{ij}}.$$

Поскольку

$$P(x,y) = P(y|x)P(x),$$

то

$$P(y|x) = \frac{P(x,y)}{P(x)} = \frac{\frac{1}{Z} e^{-E(x,y)}}{\frac{1}{Z} \sum_y e^{-E(x,y)}} = \frac{e^{-E(x,y)}}{\sum_y e^{-E(x,y)}}.$$

В результате можно получить следующее выражение:

$$\frac{\partial \ln P(x)}{\partial \omega_{ij}} = -\sum_y P(y|x) \frac{\partial E(x,y)}{\partial \omega_{ij}} + \sum_{x,y} P(x,y) \frac{\partial E(x,y)}{\partial \omega_{ij}}.$$

В данном выражении первое слагаемое определяет позитивную fazu работы машины Больцмана, когда сеть работает на основе образов из обучающей выборки. Второе слагаемое характеризует негативную fazu функционирования, когда сеть работает в свободном режиме независимо от окружающей среды.

Рассмотрим энергию сети RBM. С точки зрения энергии сети задача обучения состоит в том, чтобы на основе входных данных найти конфигурацию выходных переменных с минимальной энергией. В результате на обучающем множестве сеть будет иметь меньшую энергию по сравнению с другими состояниями. Функция энергии бинарного состояния  $(x, y)$  определяется аналогично сети Хопфилда:

$$E(x, y) = -\sum_i x_i T_i - \sum_j y_j T_j - \sum_{i,j} x_i y_j \omega_{ij}.$$

В этом случае

$$\frac{\partial E(x, y)}{\partial \omega_{ij}} = -x_i y_j,$$

$$\frac{\partial \ln P(x)}{\partial \omega_{ij}} = \sum_y P(y|x) x_i y_j - \sum_{x,y} P(x,y) x_i y_j.$$

Поскольку математическое ожидание вычисляется как

$$E(x) = \sum_i x_i P_i,$$

то

$$\frac{\partial \ln P(x)}{\partial \omega_{ij}} = E[x_i y_j]_{\text{data}} - E[x_i y_j]_{\text{model}}.$$

Аналогичным образом можно получить градиенты для пороговых значений:

$$\frac{\partial \ln P(x)}{\partial T_i} = E[x_i]_{\text{data}} - E[x_i]_{\text{model}}, \quad (9.7)$$

$$\frac{\partial \ln P(x)}{\partial T_j} = E[y_j]_{\text{data}} - E[y_j]_{\text{model}}. \quad (9.8)$$

Как следует из выражений (9.7), (9.8), первое слагаемое характеризует работу сети на основе данных из обучающей выборки, а второе слагаемое – работу сети на основе данных модели (данные, генерируемые сетью), т. е. в свободном режиме независимо от окружающей среды.

Поскольку вычисление математического ожидания на основе RBM сети является очень сложным, Дж. Хинтон предложил использовать аппроксимацию данных слагаемых, которую он назвал контрастным расхождением (contrastive divergence (CD)) [22].

Такая аппроксимация основывается на *сэмплировании Гиббса* (Gibbs sampling). В этом случае первые слагаемые в выражениях для градиента характеризуют распределение данных в момент времени  $t = 0$ , а вторые – реконструированные или генерируемые моделью состояния в момент времени  $t = k$ . Исходя из этого CD- $k$ -процедура может быть представлена следующим образом:

$$x(0) \rightarrow y(0) \rightarrow x(1) \rightarrow y(1) \rightarrow \dots \rightarrow x(k) \rightarrow y(k). \quad (9.9)$$

В результате можно получить следующие правила для обучения RBM-сети. В случае применения CD-1  $k = 1$  и учетом того, что в соответствии с методом градиентного спуска

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha \frac{\partial \ln P(x)}{\partial \omega_{ij}(t)},$$

для последовательного обучения имеем

$$\begin{aligned} \omega_{ij}(t+1) &= \omega_{ij}(t) + \alpha (x_i(0)y_j(0) - x_i(1)y_j(1)), \\ T_i(t+1) &= T_i(t) + \alpha (x_i(0) - x_i(1)), \\ T_j(t+1) &= T_j(t) + \alpha (y_j(0) - y_j(1)). \end{aligned} \quad (9.10)$$

Аналогичным образом для алгоритма CD- $k$

$$\begin{aligned}\omega_{ij}(t+1) &= \omega_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(k)y_j(k)), \\ T_i(t+1) &= T_i(t) + \alpha(x_i(0) - x_i(k)), \\ T_j(t+1) &= T_j(t) + \alpha(y_j(0) - y_j(k)).\end{aligned}\quad (9.11)$$

В случае группового обучения и CD- $k$

$$\begin{aligned}\omega_{ij}(t+1) &= \omega_{ij}(t) + \alpha \sum_{l=1}^L (x_i^l(0)y_j^l(0) - x_i^l(k)y_j^l(k)), \\ T_j(t+1) &= T_j(t) + \alpha \sum_{l=1}^L (y_j^l(0) - y_j^l(k)), \\ T_i(t+1) &= T_i(t) + \alpha \sum_{l=1}^L (x_i^l(0) - x_i^l(k)).\end{aligned}\quad (9.12)$$

Из последних выражений видно, что правила обучения ограниченной машины Больцмана минимизируют разницу между оригиналными данными и результатами, генерируемыми моделью. Генерируемые моделью значения получаются при помощи сэмплирования Гиббса.

Обучение нейронной сети глубокого доверия происходит на основе «жадного» алгоритма послойного обучения (greedy layer-wise algorithm). В соответствии с ним вначале обучается первый слой сети как RBM-машина. Для этого входные данные поступают на видимый слой нейрон-

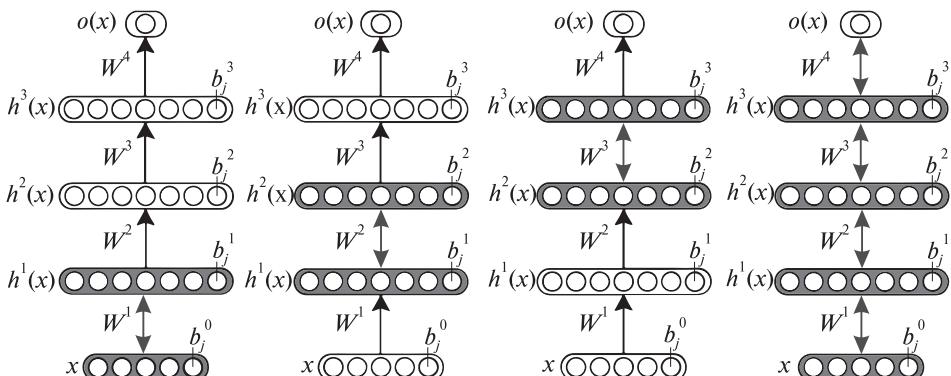


Рис. 9.6. «Жадный» алгоритм послойного обучения (Greedy layer-wise algorithm):  
 $a$  – преобразование 1-го скрытого слоя;  $b$  – преобразование 2-го скрытого слоя;  
 $c$  – преобразование 3-го скрытого слоя;  $\sigma$  – точная настройка всей сети

ных элементов и с использованием CD- $k$  процедуры вычисляются состояния скрытых  $p(y|x)$  и видимых нейронов  $p(x|y)$ . В процессе выполнения данной процедуры (не более 100 эпох) изменяются весовые коэффициенты и пороговые значения RBM-сети, которые затем фиксируются. Затем берется второй слой нейронной сети и конструируется RBM-машина. Входными данными для нее являются данные с предыдущего слоя. Происходит обучение, и процесс продолжается для всех слоев нейронной сети, как показано на рис. 9.6 [82].

В результате такого обучения без учителя можно получить подходящую начальную инициализацию настраиваемых параметров глубокой нейронной сети. На заключительном этапе осуществляется точная настройка параметров всей сети при помощи алгоритма обратного распространения ошибки или алгоритма «бодрствования и сна» (wake-sleep algorithm).

## 9.5. МЕТОД СТОХАСТИЧЕСКОГО ГРАДИЕНТА С ИСПОЛЬЗОВАНИЕМ RELU

Ректификационная функция активации применяется, как правило, во всех слоях глубокой нейронной сети, за исключением последнего слоя. В случае применения данной функции активации (рис. 9.7) для обучения глубокой нейронной сети необязательно использовать предобучение слоев нейронных элементов. Можно взять стандартный алгоритм обратного распространения ошибки для обучения сети. Метод градиентного спуска для последовательного или группового обучения, если применяются группы образов небольшого размера (minibatch) и происходит случайный выбор образов из обучающей выборки, называется методом *стохастического градиентного спуска* (stochastic gradient descent, SGD).

В случае использования функции активации ReLU выходное значение нейронного элемента можно вычислить как

$$y_j = F(S_j) = \begin{cases} S_j, & S_j > 0, \\ kS_j, & S_j \leq 0, \end{cases} \quad (9.13)$$

где  $k = 0$  либо принимает небольшое значение, например,  $k = 0,01$  или  $k = 0,001$ .

Тогда

$$\frac{\partial y_j}{\partial S_j} = F(S_j) = \begin{cases} 1, & S_j > 0, \\ k, & S_j \leq 0. \end{cases} \quad (9.14)$$

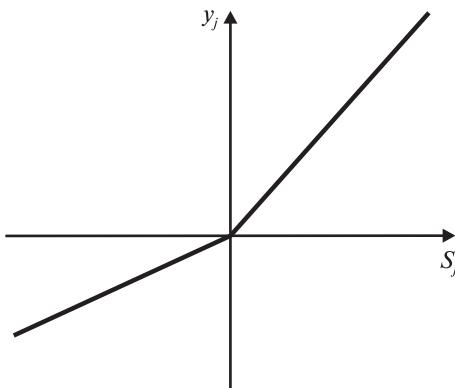


Рис. 9.7. Функция активации ReLU

Обобщенное дельта-правило (гл. 3) позволяет осуществить настройку синаптических связей глубокой нейронной сети. Почему при использовании ректификационной функции активации эффективно применение алгоритма обратного распространения ошибки для обучения глубоких нейронных сетей? Как следует из выражения (9.14), в области положительных значений взвешенной суммы производная ректификационной функции активации равна единице, что позволяет нейтрализовать проблему исчезающего градиента.

## 9.6. АЛЬТЕРНАТИВНЫЙ ВЗГЛЯД НА ОГРАНИЧЕННУЮ МАШИНУ БОЛЬЦМАНА

В данном разделе рассматривается альтернативный взгляд на ограниченную машину Больцмана как автоассоциативную нейронную сеть, которая может функционировать с любыми данными, как бинарными, так и числовыми, а также предлагаются новые методы для получения правила обучения ограниченной машины Больцмана [29, 86-90]. Первый метод базируется на минимизации ошибки реконструкции видимых и скрытых образов, которую можно получить, используя простые итерации сэмплирования Гиббса. По сравнению с традиционным подходом, основанным на энергии методе (energy-based method), который базируется на линейном представлении нейронных элементов, предложенный метод позволяет учитывать нелинейную природу нейронных элементов. Второй метод базируется на минимизации кросс-энтропии видимых и скрытых образов.

Рассмотрим ограниченную машину Больцмана, которую будем представлять в виде трех слоев нейронных элементов [88]: видимый, скрытый и видимый (рис. 9.8). Такое представление RBM эквивалентно автоэнкодерной нейронной сети (гл. 6), где скрытый и последний видимый слой являются сжимающим и восстанавливающим слоями соответственно.

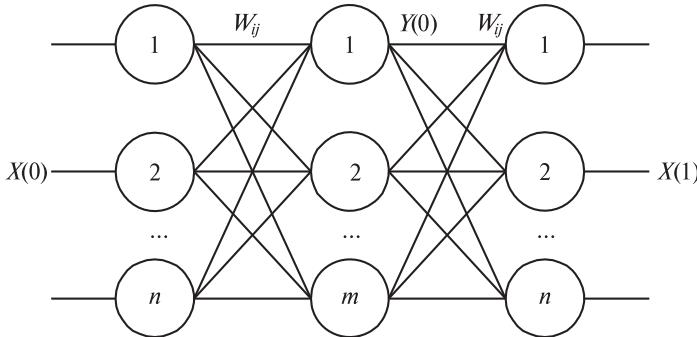


Рис. 9.8. Представление RBM в виде автоэнкодерной сети

Сэмплирование Гиббса для данного представления RBM изображено на рис. 9.9.

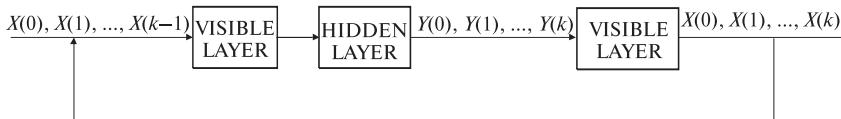


Рис. 9.9. Сэмплирование Гиббса

Процесс сэмплирования Гиббса заключается в следующей процедуре. Пусть  $x(0)$  – входной вектор, который поступает на видимый слой в момент времени 0. Тогда выходные значения нейронов скрытого слоя

$$y_j(0) = F(S_j(0)), \quad (9.15)$$

$$S_j(0) = \sum_i \omega_{ij} x_i(0) + T_j. \quad (9.16)$$

Инверсный (последний) слой реконструирует входной вектор на основе данных со скрытого слоя. В результате получается восстановленный вектор  $x(1)$  в момент времени 1:

$$x_i(1) = F(S_i(1)), \quad (9.17)$$

$$S_i(1) = \sum_j \omega_{ij} y_j(0) + T_i. \quad (9.18)$$

Затем вектор  $x(1)$  поступает на видимый слой и вычисляются выходные значения нейронов скрытого слоя:

$$y_j(1) = F(S_j(1)), \quad (9.19)$$

$$S_j(1) = \sum_i \omega_{ij} x_i(1) + T_j. \quad (9.20)$$

Продолжая данный процесс, на шаге  $k$  можно получить следующее:

$$x_i(k) = F(S_i(k)),$$

$$S_i(k) = \sum_j \omega_{ij} y_j(k-1) + T_i,$$

$$y_j(k) = F(S_j(k)),$$

$$S_j(k) = \sum_i \omega_{ij} x_i(k) + T_j.$$

Существуют различные методы обучения RBM-сети, которые базируются на использовании разных целевых функций обучения. Как отмечалось ранее, Дж. Хинтон предложил модель, основанную на энергии максимизации функции логарифмического правдоподобия распределения входных данных  $P(x)$ . В этом разделе предлагается использовать другие целевые функции. Первый критерий базируется на *минимизации суммарной квадратичной ошибки сети* (MSE), а второй — на *минимизации кросс-энтропии сети* (CE). Покажем, что применение различных критериев обучения приводит к идентичным правилам обучения.

### 9.6.1. Минимизация суммарной квадратичной ошибки

Целью обучения ограниченной машины Больцмана является минимизация суммарной квадратичной ошибки реконструкции данных на скрытом и восстанавливающем слое. Суммарная квадратичная ошибка на скрытом слое пропорциональна разнице между выходными значениями нейронов скрытого слоя в различные моменты времени и в случае CD- $k$  определяется следующим образом:

$$E_h(k) = \frac{1}{2} \sum_{l=1}^L \sum_{j=1}^m \sum_{p=1}^k (y_j^l(p) - y_j^l(p-1))^2. \quad (9.21)$$

Аналогично квадратичная ошибка инверсного слоя пропорциональна разнице между выходными значениями нейронов выходного слоя в разные моменты времени:

$$E_v(k) = \frac{1}{2} \sum_{l=1}^L \sum_{i=1}^n \sum_{p=1}^k (x_i^l(p) - x_i^l(p-1))^2, \quad (9.22)$$

где  $L$  – размерность обучающей выборки.

Тогда общая квадратичная ошибка реконструкции данных на скрытом и восстанавливающем слое определяется как сумма соответствующих ошибок:

$$E_s(k) = E_h(k) + E_v(k). \quad (9.23)$$

Отсюда

$$E_s(k) = \frac{1}{2} \sum_{l=1}^L \sum_{j=1}^m \sum_{p=1}^k (y_j^l(p) - y_j^l(p-1))^2 + \frac{1}{2} \sum_{l=1}^L \sum_{i=1}^n \sum_{p=1}^k (x_i^l(p) - x_i^l(p-1))^2. \quad (9.24)$$

Аналогичным образом для CD-1 суммарная квадратичная ошибка вычисляется как

$$E_s(1) = \frac{1}{2} \sum_{l=1}^L \sum_{j=1}^m (y_j^l(1) - y_j^l(0))^2 + \frac{1}{2} \sum_{l=1}^L \sum_{i=1}^n (x_i^l(1) - x_i^l(0))^2. \quad (9.25)$$

**Теорема 9.1.** Максимизация функции правдоподобия распределения данных  $P(x)$  в пространстве синаптических связей ограниченной машины Больцмана эквивалентна минимизации суммарной квадратичной ошибки сети в том же пространстве при использовании линейных нейронов.

*Доказательство.* Рассмотрим последовательное обучение RBM, когда модификация синаптических связей происходит после подачи каждого входного образа на сеть. В соответствии с методом градиентного спуска для минимизации суммарной квадратичной ошибки сети синаптические связи должны изменяться следующим образом:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial E}{\partial \omega_{ij}(t)},$$

$$T_i(t+1) = T_i(t) - \alpha \frac{\partial E}{\partial T_i(t)},$$

$$T_j(t+1) = T_j(t) - \alpha \frac{\partial E}{\partial T_j(t)}.$$

В случае CD- $k$  для одного образа квадратичная ошибка

$$E = \frac{1}{2} \sum_{j=1}^m \sum_{p=1}^k (y_j(p) - y_j(p-1))^2 + \frac{1}{2} \sum_{i=1}^n \sum_{p=1}^k (x_i(p) - x_i(p-1))^2.$$

Тогда

$$\begin{aligned}\frac{\partial E}{\partial \omega_{ij}} &= \frac{\partial E}{\partial y_j(p)} \frac{\partial y_j(p)}{\partial S_j(p)} \frac{\partial S_j(p)}{\partial \omega_{ij}} + \frac{\partial E}{\partial x_i(p)} \frac{\partial x_i(p)}{\partial S_i(p)} \frac{\partial S_i(p)}{\partial \omega_{ij}} = \\ &= \sum_{p=1}^k (y_j(p) - y_j(p-1)) x_i(p) F'(S_j(p)) + \\ &\quad + \sum_{p=1}^k (x_i(p) - x_i(p-1)) y_j(p-1) F'(S_i(p)).\end{aligned}$$

Если ограниченная машина Больцмана использует линейные нейроны (линейная функция активации), то

$$F'(S_j(p)) = \frac{\partial S_j(p)}{\partial w_{ij}} = F'(S_i(p)) = \frac{\partial S_i(p)}{\partial w_{ij}} = 1.$$

Тогда

$$\frac{\partial E}{\partial w_{ij}} = \sum_{p=1}^k y_j(p) x_i(p) - y_j(p-1) x_i(p-1) = y_j(k) x_i(k) - y_j(0) x_i(0).$$

В результате можно получить CD- $k$ -правило обучения RBM:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(k)y_j(k)).$$

Аналогичным образом для пороговых значений CD- $k$ -правило обучения:

$$T_j(t+1) = T_j(t) + \alpha(y_j(0) - y_j(k)),$$

$$T_i(t+1) = T_i(t) + \alpha(x_i(0) - x_i(k)).$$

Как видно, последние выражения совпадают с классическим правилом обучения ограниченной машины Больцмана для CD- $k$ . Отсюда следует, что для линейной RBM максимизация функции правдоподобия распределения данных  $P(x)$  эквивалентна минимизации суммарной квадратичной ошибки сети. Теорема доказана.

Таким образом, природа классического правила обучения RBM-сети является линейной с точки зрения минимизации MSE. Поэтому назовем такую машину линейной RBM.

**Следствие 9.1.** Для линейной ограниченной машины Больцмана правило модификации синаптических связей в случае CD-1 будет следующим:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(1)y_j(1)),$$

$$T_i(t+1) = T_i(t) + \alpha(x_i(0) - x_i(1)),$$

$$T_j(t+1) = T_j(t) + \alpha(y_j(0) - y_j(1)).$$

**Следствие 9.2.** Линейная ограниченная машина Больцмана с точки зрения обучения эквивалентна автоэнкодерной нейронной сети при использовании в ней при обучении сэмплирования Гиббса.

**Следствие 9.3.** Для нелинейной ограниченной машины Больцмана правило модификации синаптических связей в случае CD- $k$  имеет следующий вид:

$$\begin{aligned}\omega_{ij}(t+1) &= \omega_{ij}(t) - \\ -\alpha \left( \sum_{p=1}^k (y_j(p) - y_j(p-1)) x_i(p) F'(S_j(p)) + x_i(p) - x_i(p-1) \right) y_j(p-1) F'(S_i(p)) \Bigg), \\ T_j(t+1) &= T_j(t) - \alpha \left( \sum_{p=1}^k (y_j(p) - y_j(p-1)) F'(S_j(p)) \right), \\ T_i(t+1) &= T_i(t) - \alpha \left( \sum_{p=1}^k (x_i(p) - x_i(p-1)) F'(S_i(p)) \right).\end{aligned}$$

**Следствие 9.4.** Для нелинейной ограниченной машины Больцмана правило модификации синаптических связей в случае CD-1 будет следующим:

$$\begin{aligned}\omega_{ij}(t+1) &= \omega_{ij}(t) - \\ -\alpha \left( (y_j(1) - y_j(0)) F'(S_j(1)) x_i(1) + (x_i(1) - x_i(0)) F'(S_i(1)) y_j(0) \right), \\ T_i(t+1) &= T_i(t) - \alpha (x_i(1) - x_i(0)) F'(S_i(1)), \\ T_j(t+1) &= T_j(t) - \alpha (y_j(1) - y_j(0)) F'(S_j(1)).\end{aligned}$$

Если используется групповое обучение (batch learning), то в этом случае метод градиентного спуска записывается как

$$\begin{aligned}\omega_{ij}(t+1) &= \omega_{ij}(t) - \alpha \frac{\partial E_s(k)}{\partial \omega_{ij}(t)}, \\ T_i(t+1) &= T_i(t) - \alpha \frac{\partial E_s(k)}{\partial T_i(t)}, \\ T_j(t+1) &= T_j(t) - \alpha \frac{\partial E_s(k)}{\partial T_j(t)}.\end{aligned}$$

**Теорема 9.2.** При CD- $k$  для нелинейной ограниченной машины Больцмана в случае группового обучения правило модификации синаптических связей определяется на основе выражений

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \sum_{l=1}^L \sum_{p=1}^k \left( (y_j^l(p) - y_j^l(p-1)) x_i^l(p) F'(S_j^l(p)) + \right. \\ \left. + (x_i^l(p) - x_i^l(p-1)) y_j^l(p-1) F'(S_i^l(p)) \right),$$

$$T_j(t+1) = T_j(t) - \alpha \sum_{l=1}^L \sum_{p=1}^k (y_j^l(p) - y_j^l(p-1)) F'(S_j^l(p)),$$

$$T_i(t+1) = T_i(t) - \alpha \sum_{l=1}^L \sum_{p=1}^k (x_i^l(p) - x_i^l(p-1)) F'(S_i^l(p)).$$

Здесь  $L$  – размер группы образов. Процесс доказательства данной теоремы является аналогичным доказательству теоремы 9.1.

**Следствие 9.5.** При использовании CD-1 для нелинейной ограниченной машины Больцмана в случае группового обучения правило модификации синаптических связей имеет вид

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \\ - \alpha \sum_{l=1}^L \left( (y_j^l(1) - y_j^l(0)) x_i^l(1) F'(S_j^l(1)) + (x_i^l(1) - x_i^l(0)) y_j^l(0) F'(S_i^l(1)) \right),$$

$$T_j(t+1) = T_j(t) - \alpha \sum_{l=1}^L (y_j^l(1) - y_j^l(0)) F'(S_j^l(1)),$$

$$T_i(t+1) = T_i(t) - \alpha \sum_{l=1}^L (x_i^l(1) - x_i^l(0)) F'(S_i^l(1)).$$

**Следствие 9.6.** При CD- $k$  для линейной ограниченной машины Больцмана в случае группового обучения правило модификации синаптических связей можно записать как

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha \sum_{l=1}^L (x_i^l(0) y_j^l(0) - x_i^l(k) y_j^l(k)),$$

$$T_j(t+1) = T_j(t) + \alpha \sum_{l=1}^L (y_j^l(0) - y_j^l(k)),$$

$$T_i(t+1) = T_i(t) + \alpha \sum_{l=1}^L (x_i^l(0) - x_i^l(k)).$$

**Следствие 9.7.** При CD-1 для линейной ограниченной машины Больцмана в случае группового обучения правило модификации синаптических связей определяется на основе следующих выражений:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha \sum_{l=1}^L (x_i^l(0)y_j^l(0) - x_i^l(1)y_j^l(1)),$$

$$T_j(t+1) = T_j(t) + \alpha \sum_{l=1}^L (y_j^l(0) - y_j^l(1)),$$

$$T_i(t+1) = T_i(t) + \alpha \sum_{l=1}^L (x_i^l(0) - x_i^l(1)).$$

**Примечание.** При групповом обучении обычно вторые слагаемые в выражениях для модификации синаптических связей делятся на количество образов  $L$ .

В данном разделе получены правила обучения для ограниченной машины Больцмана, которые базируются на минимизации квадратичной ошибки восстановления информации в скрытом и видимом слоях. Предложенный метод обучения позволяет учитывать нелинейную природу нейронных элементов с точки зрения минимизации суммарной квадратичной ошибки сети. Назовем его REBA (reconstruction error-based approach) [88]. Показано, что классические выражения для обучения ограниченной машины являются частным случаем предложенного метода. Доказана теорема об эквивалентности максимизации функции правдоподобия распределения входных данных  $P(x)$  в пространстве синаптических связей и минимизации суммарной квадратичной ошибки сети в том же пространстве для линейной ограниченной машины Больцмана. Впервые приведенные выше выражения были получены в работах [29, 86, 87].

### 9.6.2. Минимизация кросс-энтропии

**Кросс-энтропия** может применяться в качестве альтернативы целевой функции квадратичной ошибки. Рассмотрим ограниченную машину Больцмана, нейронные элементы которой используют сигмоидную функцию активации. Тогда целью обучения RBM является минимизация кросс-энтропии в скрытом и видимом слоях. В случае CD- $k$  кросс-энтропия функции ошибки для инверсного (восстанавливающего) слоя определяется как

$$CE_v(k) = -\sum_{l=1}^L \left[ \sum_{p=1}^k \sum_{i=1}^n (x_i^l(p-1) \log(x_i^l(p)) + (1 - x_i^l(p-1)) \log(1 - x_i^l(p))) \right]. \quad (9.24)$$

Кросс-энтропия функции ошибки для скрытого слоя имеет следующий вид:

$$CE_h(k) = -\sum_{l=1}^L \left[ \sum_{p=1}^k \sum_{j=1}^m \left( y_j^l(p-1) \log(y_j^l(p)) + (1-y_j^l(p-1)) \log(1-y_j^l(p)) \right) \right]. \quad (9.25)$$

Общая кросс-энтропия функции ошибки для RBM в случае CD- $k$  вычисляется как сумма соответствующих ошибок:

$$CE_s(k) = CE_h(k) + CE_v(k). \quad (9.26)$$

**Теорема 9.3.** Максимизация функции правдоподобия распределения входных данных  $P(x)$  в пространстве синаптических связей ограниченной машины Больцмана эквивалентна минимизации кросс-энтропии функции ошибки сети  $CE_s(k)$  в том же пространстве.

*Доказательство.* Для упрощения доказательства теоремы рассмотрим кросс-энтропию для CD-1. Тогда найдем кросс-энтропию функции ошибки сети для одного образа:

$$\begin{aligned} CE(l) = & -\sum_{i=1}^n \left( x_i(0) \log(x_i(1)) + (1-x_i(0)) \log(1-x_i(1)) \right) - \\ & -\sum_{j=1}^m \left( y_j(0) \log(y_j(1)) + (1-y_j(0)) \log(1-y_j(1)) \right). \end{aligned}$$

Для последовательного обучения

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial CE(l)}{\partial \omega_{ij}(t)},$$

$$T_i(t+1) = T_i(t) - \alpha \frac{\partial CE(l)}{\partial T_i(t)},$$

$$T_j(t+1) = T_j(t) - \alpha \frac{\partial CE(l)}{\partial T_j(t)}.$$

Отсюда можно получить, что

$$\begin{aligned} \frac{\partial CE(l)}{\partial \omega_{ij}} = & -\frac{x_i(0)}{x_i(1)} x_i(1) (1-x_i(1)) y_j(0) + \\ & + (1-y_j(0)) y_j(1) x_i(1) = -x_i(0) (1-x_i(1)) y_j(0) + \\ & + (1-x_i(0)) x_i(1) y_j(0) - y_j(0) (1-y_j(1)) x_i(1) + \end{aligned}$$

$$\begin{aligned}
 & + (1 - y_j(0))y_j(1)x_i(1) = -x_i(0)y_j(0) + x_i(0)x_i(1)y_j(0) + \\
 & + x_i(1)y_j(0) - x_i(0)x_i(1)y_j(0) - y_j(0)x_i(1) + \\
 & + y_j(0)y_j(1)x_i(1) + y_j(1)x_i(1) - y_j(0)y_j(1)x_i(1) = x_i(1)y_j(1) - x_i(0)y_j(0).
 \end{aligned}$$

Аналогично для пороговых значений

$$\begin{aligned}
 \frac{\partial CE(l)}{\partial T_i} &= x_i(1) - x_i(0), \\
 \frac{\partial CE(l)}{\partial T_j} &= y_j(1) - y_j(0).
 \end{aligned}$$

Теорема доказана.

Как следует из теоремы, классические правила обучения RBM-сети могут быть получены более простым путем по сравнению с конвенциальным методом, основанным на энергии. Таким образом, используя минимизацию функции кросс-энтропии и сэмплирование Гиббса, можно получить классические выражения для обучения RBM.

Полученные результаты могут быть обобщены в виде следующей теоремы.

**Теорема 9.4.** Максимизация функции правдоподобия распределения входных данных  $P(x)$  в пространстве синаптических связей ограниченной машины Больцмана эквивалентна минимизации кросс-энтропии функции ошибки сети и минимизации суммарной квадратичной ошибки сети в том же пространстве при использовании линейных нейронов:

$$\max(\ln P(x)) = \min(CE_s) = \min(E_s).$$

Из теоремы следует, что применение различных критериев обучения приводит к одинаковым правилам обучения. Поэтому природа неконтролируемого обучения (обучение без учителя) в RBM-сети является идентичной при использовании различных целевых функций. Максимизация функции правдоподобия распределения входных данных и минимизация кросс-энтропии функции ошибки приводят к линейному представлению нейронных элементов с точки зрения минимизации MSE. Применение MSE в качестве целевой функции позволяет получить как линейные, так и нелинейные правила обучения, но не наоборот. Поэтому метод, основанный на минимизации суммарной квадратичной ошибки, более универсальный и открывает новые возможности для неконтролируемого обучения в глубоких нейронных сетях. Впервые приведенный выше метод был предложен в работе [88].

## 9.7. ПРИМЕНЕНИЕ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

Глубокие нейронные сети используются для сжатия и визуализации данных, распознавания образов, обработки речи и т. д. Рассмотрим применение автоэнкодерных и персептронных глубоких нейронных сетей для решения различных задач обработки информации.

### 9.7.1. Сжатие данных

Пусть дана система трех динамических уравнений [29], где параметр времени  $t$  генерируется в диапазоне  $[-1,1]$ :

$$\begin{cases} x_1 = \sin(\pi t) + \mu, \\ x_2 = \cos(\pi t) + \mu, \\ x_3 = t + \mu. \end{cases}$$

Здесь  $\mu$  – гауссовский шум с нулевым средним и квадратичным отклонением, равным 0,05. Рассмотрим отображение входного трехмерного пространства данных в одномерное при помощи глубокого автоэнкодера, который состоит из семи слоев нейронных элементов (рис. 9.10). Для обучения сети возьмем обучающую выборку, состоящую из 1000 тренировочных наборов. Тестирование сети проведем на данных, не входящих в обучающую выборку, количество которых равняется 1000 образов. В качестве функции активации нейронных элементов для всех слоев, кроме сжимающего, использовалась сигмоидная функция. Для сжимающего нейрона – линейная функция активации. Для обучения каждого слоя нейронной сети применялось 50 эпох, а для точной настройки параметров сети при помощи алгоритма обратного распространения ошибки – 200 эпох.

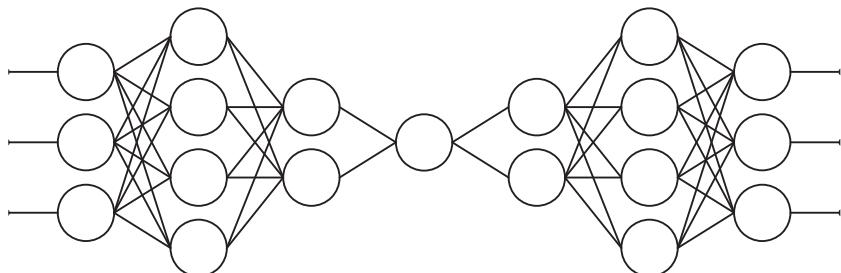


Рис. 9.10. Глубокий автоэнкодер

Результаты экспериментов приведены в табл. 9.1. Здесь MSE – суммарная квадратичная ошибка на обучающей выборке, MS – квадратичная ошибка на тестовой выборке (ошибка обобщения), RBM – метод обучения на основе ограниченной машины Больцмана, REBA – предложенный метод.

Таблица 9.1

Метод обучения	CD- <i>k</i>	MSE	MS
RBM	1	0,699	0,886
	5	0,710	0,932
	10	0,689	0,916
	15	0,688	0,873
REBA	1	0,673	0,851
	5	0,719	0,966
	10	0,677	0,907
	15	0,700	0,895

Как следует из табл. 9.1, метод REBA более эффективен по сравнению с традиционным подходом для CD-1 и CD-10.

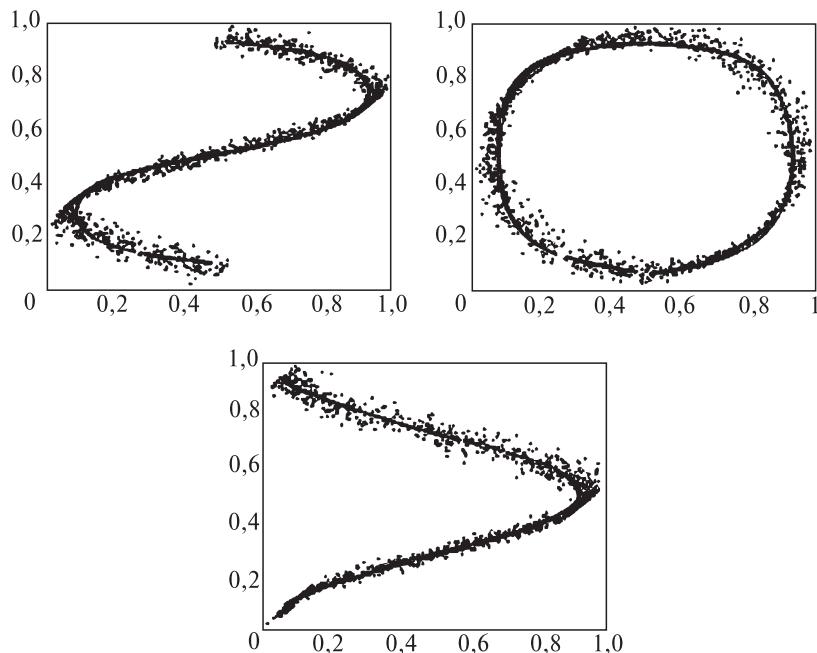
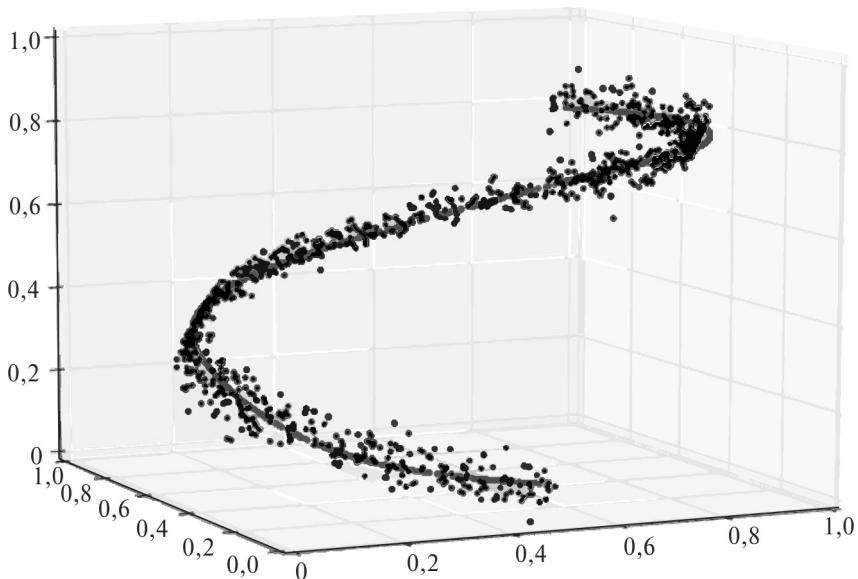


Рис. 9.11. Нелинейная ось первой главной компоненты в двумерном пространстве

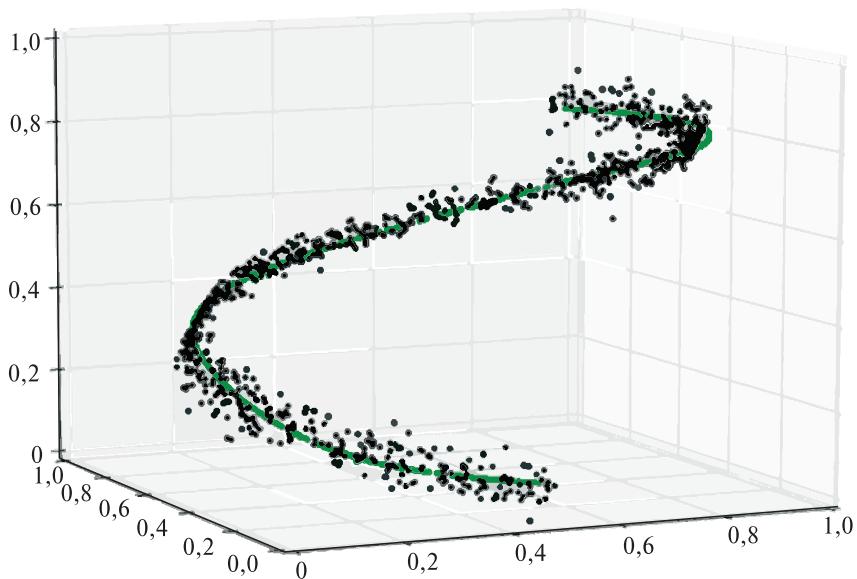


*Рис. 9.12.* Нелинейная ось первой главной компоненты в трехмерном пространстве

На рис. 9.11 и 9.12 изображена нелинейная ось первой главной компоненты, на которую проецируется входное пространство образов.

### 9.7.2. Визуализация данных

Рассмотрим визуализацию рукописных цифр с применением глубокого автоэнкодера на основе базы данных MNIST. Она содержит 60 000 образов рукописных цифр для обучения и 10 000 образов для тестирования. Каждый образ представляет собой изображение  $28 \times 28$  пикселей в градациях серого. Для отображения 784-мерных образов в двумерное пространство признаков будем использовать глубокий автоэнкодер с архитектурой 784-1000-500-250-2-250-500-1000-784. В среднем слое нейронной сети, который состоит из двух нейронов, применяется линейная функция активации, в остальных слоях – сигмоидная функция активации. Для предварительного обучения глубокого автоэнкодера рассмотрим алгоритм послойного обучения на основе RBM- и REBA-методов. Данная процедура начинается с первого слоя и выполняется без учителя. После этого выполняется обучение всей нейронной сети с алгоритмом обратного распространения ошибки. Для



настройки синаптических связей сети применялись следующие параметры: скорость предварительного обучения, равная 0,2 для REBA и 0,05 для классического RBM-метода для всех слоев, за исключением среднего. Скорость обучения для среднего слоя равна 0,001. Сравнительный анализ обоих методов представлен в табл. 9.2, из которой следует, что предложенный подход REBA показывает лучшую обобщающую способность.

Визуализация рукописных цифр, выполненная на основе REBA, представлена на рис. 9.13 для первых 500 тестовых изображений каждого класса.

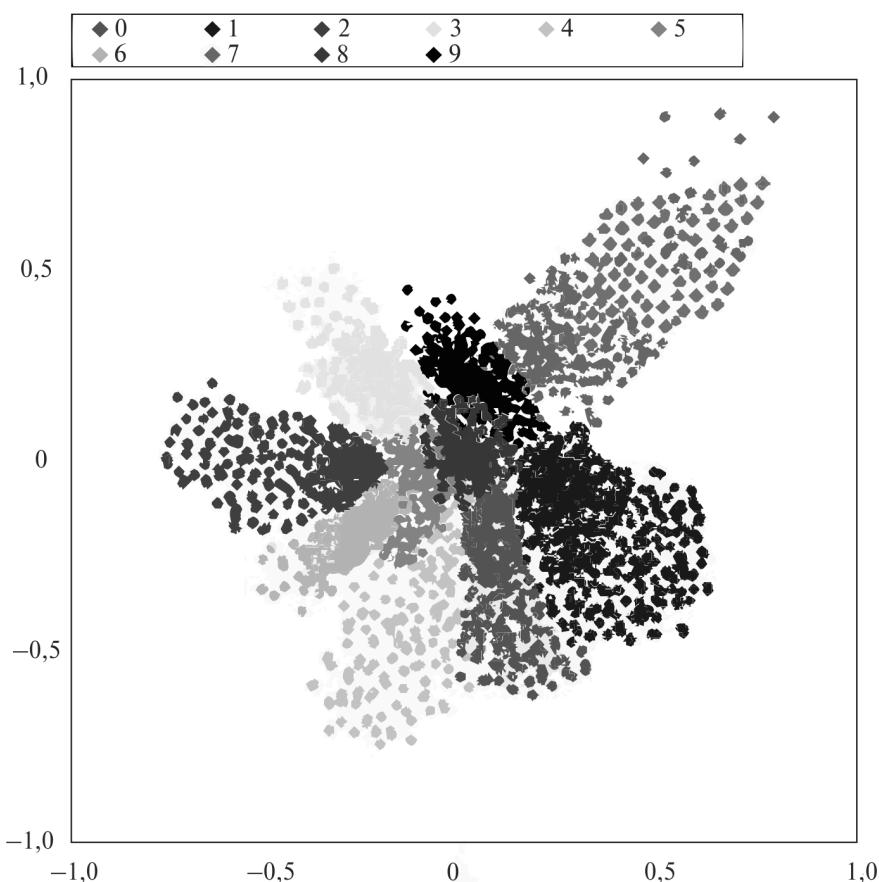


Рис. 9.13. Визуализация рукописных цифр

Таблица 9.2

Метод обучения	MSE	MS
RBM	3,7801	4,0115
REBA	3,6490	3,8726

На рис. 9.14 изображен пример визуализации документов при помощи сети 2000-500-250-2-250-500-2000, предложенный Дж. Хинтоном.

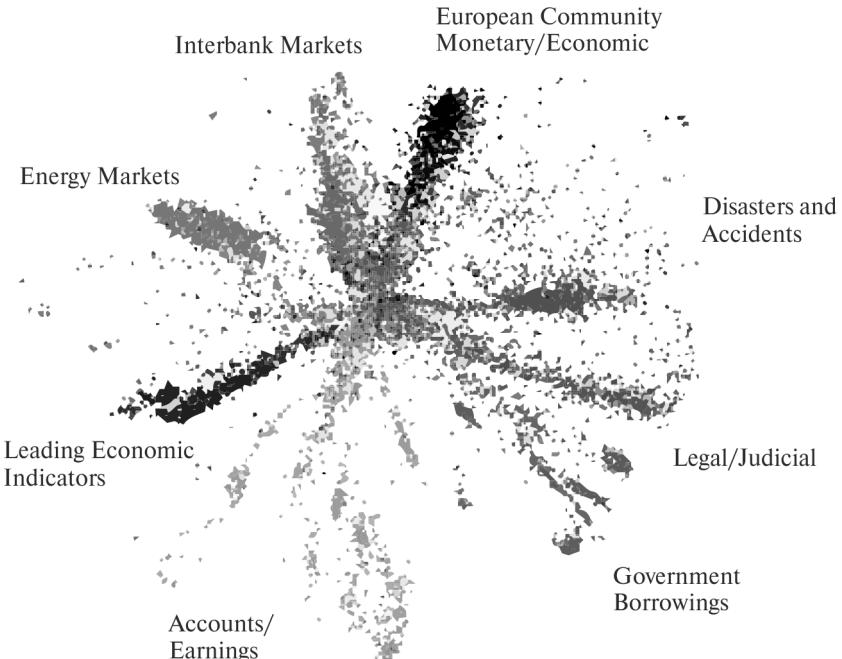


Рис. 9.14. Визуализация документов

Обучающая выборка состояла из 400 000 документов [23]. Как видно из рис. 9.14, глубокая нейронная сеть довольно точно осуществляет разделение документов на различные классы.

### 9.7.3. Классификация образов

Рассмотрим классификацию рукописных цифр с базой данных MNIST. В качестве классификатора возьмем глубокий персепtron с архитектурой 784-500-500-2000-10 и сигмоидной функцией активации. Будем использовать кодировку выходных классов на основе номера

нейрона последнего слоя. Для этого необходимо определить номер  $k$  нейронного элемента, который имеет максимальное выходное значение:

$$y_k = \arg \max y_j.$$

Выходному значению нейрона с номером  $k$  присваивается единичное значение, а выходные значения остальных нейронных элементов равно нулю:

$$y_j = \begin{cases} 1, & j = k, \\ 0, & \text{иначе.} \end{cases}$$

Для обучения применялся групповой метод со следующими параметрами: скорость обучения равна 0,1 для REBA и 0,2 для классического RBM-метода; размер группы (mini batch) – 100 образов; количество эпох предварительного обучения – 10; количество эпох алгоритма обратного распространения ошибки равняется 100; параметр регуляризации  $\lambda = 0,00001$ . Лучшие результаты были получены при гибридном подходе, который представляет собой комбинацию REBA- и RBM-метода. Результаты экспериментов представлены в табл. 9.3.

Таблица 9.3

Метод обучения	MSE	MS	Ошибка тестирования (%)	NIT
RBM	6,178e-6	0,0235	1,23	10
Гибридный RBM+REBA (9+1)	5,962e-6	0,0224	1,09	9+1

Здесь NIT обозначает количество эпох предварительного обучения каждого слоя сети. Для гибридного подхода в процессе предобучения использовалось 9 эпох RBM- и 1 эпоха REBA-метода. Как следует из табл. 9.3, ошибка тестирования в этом случае составляет 1,09 %.

---

## БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1. Советский энциклопедический словарь / гл. ред. А. М. Прохоров. М., 1983.
2. *Фишбах Д.* Психика и мозг // В мире науки. – № 11–12. – С. 10–20.
3. *Хьюбелл Д.* Глаз, мозг, зрение : пер. с англ. – М., 1990.
4. *Гольдман-Ракич П.* Оперативная память и разум // В мире науки. – № 11–12. – С. 63–70.
5. *Кендел Э., Хокинг Р.* Биологические основы обучения и индивидуальности // В мире науки. – 1992. – № 11–12. – С. 43–51.
6. *Зеки С.* Зрительный образ в сознании и мозге // В мире науки. – 1992. – № 11–12. – С. 33–41.
7. *Шатц К.* Развивающийся мозг // В мире науки. – 1992. – № 11–12. – С. 23–30.
8. *Сенко Д.* Стареющий мозг // В мире науки. – 1992. – № 11–12. – С. 93–100.
9. *Фрейд З.* Психология бессознательного : сб. произведений / науч. ред. М. Г. Ярошевский. – М., 1989.
10. *Крик Ф., Кох К.* Проблема сознания // В мире науки. – 1992. – № 11–12. – С. 113–120.
11. *Невв Д.* The Organization of behaviour. – N. Y., 1949.
12. *McCulloh W., Pitts W.* A logical calculus of the ideas immanent in nervous activity // Bulletin of mathematical biophysics. – 1943. – № 5. – P. 115–133.
13. *Розенблattт Ф.* Принципы нейродинамики: персептрон и теория механизмов мозга : пер. с англ. – М., 1965.
14. *Widrow B., Hoff M.* Adaptive switching circuits // In 1960 IRE WES-CON convention record. – N. Y., 1960. – P. 96–104.
15. *Minsky M., Papert S.* Perceptrons: an introduction to computational geometry. – Cambridge, 1969.

16. *Anderson J.* Neural models with cognitive implications // Basic processes in reading perception and comprehension models. — Hillsdale, 1977. — P. 27–90.
17. *Kohonen T.* Associative memory: a system – theoretical approach. — N. Y., 1977.
18. *Grossberg S.* Adaptive pattern classification and universal recoding // Biological cybernetics. — 1976. — № 23. — P. 121–134.
19. *Hopfield J.* Neural networks and physical systems with emergent collective computational abilities // Proc. of the Nat. acad. of sciences USA. — Vol. 79, № 8. — 1982. — P. 2554–2558.
20. *Kohonen T.* Self-organised formation of topologically correct feature maps // Biological cybernetics. — 1982. — № 43. — P. 59–69.
21. *Rumelhart D., Hinton G., Williams R.* Learning representation by backpropagation errors // Nature. — 1986. — № 323. — P. 533–536.
22. *Hinton G. E., Osindero S., Teh Y.* A fast learning algorithm for deep belief nets // Neural computation. — 2006. — № 18. — P. 1527–1554.
23. *Hinton G., Salakhutdinov R.* Reducing the dimensionality of data with neural networks // Science. — 2006. — № 313 (5786). — P. 504–507.
24. *Hinton G. E.* A practical guide to training restricted Boltzmann machines. — Toronto, 2010.
25. *Головко В. А.* Нейроинтеллект: теория и применение : в 2 кн. — Брест, 1999. — Кн. 1 : Организация и обучение нейронных сетей с прямыми и обратными связями.
26. *Головко В. А.* Нейроинтеллект: теория и применение : в 2 кн. — Брест, 1999. — Кн. 2 : Самоорганизация, отказоустойчивость и применение нейронных сетей.
27. *Головко В. А.* Нейронные сети: обучение, организация и применение : учеб. пособие / общ. ред. А. И. Галушкина. — М., 2001.
28. *Гантмахер Ф.* Теория матриц. — М., 1988.
29. *Головко В. А.* От многослойных персепtronов к нейронным системам глубокого доверия: парадигмы обучения и применение : лекции по нейроинформатике. — М., 2015.
30. *Lipmann R.* An introduction to computing with neural nets // IEEE acoustic, speech and signal processing magazine. — 1987. — № 2 — P. 4–22.
31. *Колмогоров А. Н.* Представление непрерывных функций многих переменных суперпозицией функций одной переменной и сложением // Дагест. АН. — 1958. — № 5. — С. 953–956.
32. *Cybenko G.* Approximations by superpositions of a sigmoidal function // Mathematics of control, signals, systems. — 1989. — Vol. 2. — P. 303–314.
33. *Hornik K., Stinchcombe M., White H.* Multilayer feedforward networks are universal approximators // Neural networks. — 1989. — № 2 — P. 359–366.

34. Rumelhart D., Hinton G., Williams R. Learning internal representations by errors propagation // Parallel distributed processing. – Cambridge, 1986.
35. Поляк Б. Т. Введение в оптимизацию. – М., 1983.
36. Головко В. А. Нейросетевые методы обработки хаотических процессов : лекции по нейроинформатике. – М., 2005. – С. 43–88.
37. Jochem T., Pomerlau D., Thorpe C. MANIAC: a next generation neurally based autonomous road follower : Proc. of the intern. conf. on intelligent autonomous systems. – Pittsburgh, 1993. – Р. 34–39.
38. Golovko V., Savitsky Yu., Maniakov N. Neural networks for signal processing in measurement analysis and industrial applications: the case of chaotic signal processing // Neural networks for instrumentation, measurement analysis and related industrial applications. – Amsterdam, 2003. – Р. 119–143.
39. Jordan M. Attractor dynamics and parallelism in a connectionist sequential machine : Proc. of the eighth annual conf. of the cognitive science soc. – Hillsdale, 1986. – Р. 531–546.
40. Elman J. Finding structure in time // Cognitive science. – 1990. – № 14. – Р. 179–211.
41. Haykin S. Neural networks: a comprehensive foundation. – N. Y., 1994.
42. Takens F. Detecting strange attractors in fluid turbulence // In dynamical systems and turbulence. – B., 1981.
43. Backpropagation applied to handwritten zip code recognition / Y. Le Cun [et al.] // Neural computation. – 1989. – № 1(4). – Р. 541–551.
44. Object recognition with gradient-based learning / Y. Le Cun [et al.] // In shape, contour and grouping in computer vision. – B.; Heidelberg, 1999. – Р. 319–345.
45. Gradient-based learning applied to document recognition / Y. Le Cun [et al.] // Proc. of the IEEE. – 1998. – № 86(11). – Р. 2278–2324.
46. Golovko V., Mikhno E., Brich A. A simple shallow convolutional neural network for accurate handwritten digits classification : Proc. of the 13th on pattern recognition and inform. processing (PRIP'2016, 3–5 oct., 2016). – Minsk, 2016. – Р. 209–212.
47. Oja E. A simplified neuron model as a principal component analyzer // J. of mathematical biology. – 1982. – № 15. – Р. 267–273.
48. Прикладная статистика. Классификация и снижение размерности : справ. изд. / С. А. Айвазян [и др.]; под ред. С. А. Айвазяна. – М., 1989. – 607 с.
49. Афиши А., Эйзен С. Статистический анализ. Подход с использованием ЭВМ : пер. с англ. – М., 1982.

50. Малиновский Л. Г. Классификация объектов средствами дискриминантного анализа. – М., 1978.
51. Unsupervised learning for dimensionality reductions / L. Grandinetti [et al.] : Proc. of second intern. ICSC symposium on engineering of intelligent systems EIS'2000. – Paisley, 2000. – P. 140–144.
52. Neural network model for transient ischemic attacks diagnostics / V. Golovko [et al.] // Optical memory and neural networks (Springer Link). – 2012. – Vol. 21, № 3. – P. 166–176.
53. Hinton G., McClelland J. Learning representation by recirculation : Proc. of IEEE conf. on neural inform. processing systems. – N. Y., 1989. – P. 358–366.
54. Dimensionality reduction and attack recognition using neural network approaches/ V. Golovko [et al.] : Proc. of the IEEE intern. joint conf. on neural networks (IJCNN 2007), Orlando, FL, USA, INNS. – Orlando, 2007. – P. 2734–2739.
55. Neural network and artificial immune systems for malware and network intrusion detection / V. Golovko [et al.] // Studies in computational intelligence. – Heidelberg, 2010. – Vol. 263 : Advances in machine learning II. – P. 485–513.
56. Hyvaerinen A., Oja E. Independent component analysis: algorithms and applications // Neural networks. – 2000. – Vol. 13, № 4–5. – P. 411–430.
57. Hopfield J. Neurons with graded response have collective computational properties like those of two-state neurons // Proc. of the Nat. acad. of science USA. – 1984. –Vol. 81. – P. 3088–3092.
58. Hopfield J., Tank D. Neural computation of decisions in optimization problems // Biological cybernetics. – 1985. – Vol. 52. – P. 141–152.
59. Тэнк Д., Хопфилд Дж. Коллективные вычисления в нейроподобных электронных схемах // В мире науки. – 1988. – № 2. – С. 45–53.
60. Kosko B. Competitive adaptive bidirectional associative memories : Proc. of the IEEE first intern. conf. on neural networks. – San-Diego, 1987. – Vol. 2.
61. Kosko B. Feedback stability and unsupervised learning : Proc. of the IEEE second intern. conf. on neural networks. – San-Diego, 1988.
62. Ляпунов А. М. Общая задача об устойчивости движения. – М., 1952.
63. Воронов А. А., Титов В. К., Новогранов Б. Н. Основы теории автоматического регулирования и управления : учеб. пособие. – М., 1977.
64. Теория автоматического управления : учеб. пособие / под. ред. А. С. Шаталова. – М., 1977.

65. Справочник по теории автоматического управления / под ред. А. А. Красковского. – М., 1987.
66. Меламед И. И. Нейронные сети и комбинаторная оптимизация // Автоматика и телемеханика. – 1994. – № 11. – С. 3–40.
67. Гусак А. А., Гусак Г. М. Справочник по высшей математике. – Минск, 1991.
68. Nijhuis J., Spaanenburg L. Fault tolerance of neural associative memories // Comput. and digital techn. – 1989. – № 5. – P. 389–394.
69. Lam D., Carroli J. Pattern recognition using a double layer associative memory : Proc. intern. conf. on real-time signal process. – San-Diego, 1989. – P. 177–188.
70. Aiyer V., Nirajan M., Fallside F. A. A theoretical investigation into the performance of the Hopfield model // IEEE transactions on neural networks. – 1990. – Vol. 2, № 2. – P. 204–215.
71. Wilson G., Pawley G. On the stability of the travelling salesman problem algorithm of Hopfield and Tank // Biological cybernetics. – 1988. – № 58. – P. 63–70.
72. Васильев Ф. П. Численные методы решения экстремальных задач : учеб. пособие. – М., 1988.
73. Lipmann R. An introduction to computing with neural nets // IEEE acoustic: speech and signal processing magazine. – 1987. – № 2. – P. 4–22.
74. Kohonen T. Self-organizing and associative memory. – Б., 1984.
75. Kohonen T. Self-organizing maps. – Heidelberg, 1995.
76. Krizhevsky A., Sutskever L., Hinton G. ImageNet classification with deep convolutional neural networks : Proc. of the intern. conf. advances in neural inform. processing systems (NIPS-2012). – Lake Tahoe, 2012. – № 25. – P. 1090–1098.
77. Le Cun Y., Bengio Y., Hinton G. Deep learning // Nature. – 2015. – № 521 (7553). – P. 436–444.
78. Strategies for training large scale neural network language models / T. Mikolov [et al.] : Proc. 2011 IEEE Workshop automatic speech recognition and understanding. – Waikoloa, 2011. – P. 195–201.
79. Deep neural network for acoustic modeling in speech recognition / G. Hinton [et al.] // IEEE signal processing magazine. – 2012. – № 29. – P. 82–97.
80. Bengio Y. Learning deep architectures for AI // Foundations and trends in machine learning. – 2009. – № 2(1). – P. 1–127.
81. Greedy layer-wise training of deep networks / Y. Bengio [et al.] ; ed : B. Scholkopf, J. C. Platt, T. Hoffman // Advances in neural information processing systems. – Cambridge, 2007. – № 11. – P. 153–160.

82. Why does unsupervised pre-training help deep learning? / D. Erhan [et al.] // J. of machine learning research. – 2010. – № 11. – P. 625–660.
83. Exploring strategies for training deep neural networks / H. Larochelle [et al.] // J. of machine learning research. – 2009. – № 1. – P. 1–40.
84. Glorot X., Bordes A., Bengio Y. Deep sparse rectifier networks : Proc. of the 14th intern. conf. on artificial intelligence and statistics. – 2011. – Vol. 15. – P. 315–323.
85. Le Cun Y., Bengio Y., Hinton G. Deep learning. – P. 436–444.
86. A learning technique for deep belief neural networks / V. Golovko [et al.] // Neural networks and artificial intelligence. – Cham, 2014. – Vol. 440 : Communication in computer and information science. – P. 136–146.
87. A new technique for restricted boltzmann machine learning / V. Golovko [et al.] : Proc. of the 8th IEEE intern. conf. IDAACS – 2015, Warsaw, 24–26 sept. 2015. – Warsaw, 2015 – P. 182–186.
88. The nature of unsupervised learning in deep neural networks: a new understanding and novel approach / V. Golovko [et al.] // Optical memory and neural networks (Springer Link). – 2016. – Vol. 25 – № 3. – P. 127–141.
89. Deep neural networks: a theory, application and new trends / V. Golovko [et al.] : Proc. of the 13th intern. conf. on pattern recognition and inform. processing, Minsk, 3–5 oct. 2016. – Minsk, 2016. – P. 33 – 37.
90. Deep learning classifier based on NPCA and orthogonal feature selection / S. Jankowski [et al.] : Proc. of the intern. conf. on photonics applications in astronomy, communications, industry, and high-energy physics experiments, Wilga, Poland, may 29 2016. – Wilga, 2016. – P. 5–9.

---

## ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ .....	7
Г л а в а 1. БИОЛОГИЧЕСКИЕ ОСНОВЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА .....	
1.1. НА СТЫКЕ НАУК .....	9
1.2. БИОЛОГИЧЕСКИЙ НЕЙРОН .....	9
1.3. НЕЙРОННАЯ ОРГАНИЗАЦИЯ МОЗГА .....	12
1.4. МОРФОГЕНЕЗ МОЗГА .....	16
1.5. МЕХАНИЗМЫ ОБУЧЕНИЯ .....	18
1.6. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СИСТЕМЫ .....	21
1.7. КЛАССИФИКАЦИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ .....	23
Г л а в а 2. ОДНОСЛОЙНЫЕ ПЕРСЕПТРОНЫ .....	
2.1. ИСКУССТВЕННЫЙ НЕЙРОН .....	26
2.2. ФУНКЦИИ АКТИВАЦИИ НЕЙРОННЫХ ЭЛЕМЕНТОВ .....	28
2.2.1. Линейная функция .....	28
2.2.2. Пороговая функция .....	28
2.2.3. Линейная ограниченная функция .....	29
2.2.4. Модифицированная пороговая функция .....	30
2.2.5. Сигмоидная функция .....	30
2.2.6. Биполярная сигмоидная функция .....	31
2.2.7. Гиперболический тангенс .....	31
2.2.8. Радиально-базисная функция .....	32
2.2.9. Ректификационная функция активации .....	33
2.2.10. Функция активации softmax .....	33
2.3. НЕЙРОННЫЕ СЕТИ С ОДНИМ ОБРАБАТЫВАЮЩИМ СЛОЕМ .....	34
2.4. ВОЗМОЖНОСТИ ОДНОСЛОЙНЫХ ПЕРСЕПТРОНОВ .....	35
2.5. ПРАВИЛО ОБУЧЕНИЯ РОЗЕНБЛАТТА .....	39
2.6. ГЕОМЕТРИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ ПРОЦЕДУРЫ ОБУЧЕНИЯ ПЕРСЕПТРОНА .....	41
2.7. ПРИМЕРЫ РЕШЕНИЯ ЗАДАЧ ОДНОСЛОЙНЫМ ПЕРСЕПТРОНОМ .....	42
2.8. ПРАВИЛО ОБУЧЕНИЯ ВИДРОУ – ХОФФА (ДЕЛЬТА-ПРАВИЛО) .....	46
2.8.1. Последовательное обучение .....	48
2.8.2. Групповое обучение .....	50
2.9. АДАПТИВНЫЙ ШАГ ОБУЧЕНИЯ .....	51

2.10. ИСПОЛЬЗОВАНИЕ ПСЕВДООБРАТНОЙ МАТРИЦЫ ДЛЯ ОБУЧЕНИЯ ЛИНЕЙНЫХ НЕЙРОННЫХ СЕТЕЙ .....	54
2.11. АНАЛИЗ ЛИНЕЙНЫХ НЕЙРОННЫХ СЕТЕЙ .....	55
2.12. ОДНОСЛОЙНЫЙ ПЕРСЕПТРОН ДЛЯ РЕШЕНИЯ ЗАДАЧИ «ИСКЛЮЧАЮЩЕЕ ИЛИ» .....	57
2.13. ПРИМЕНЕНИЕ ОДНОСЛОЙНЫХ ПЕРСЕПТРОНОВ .....	59
2.13.1. Решение системы линейных уравнений .....	60
2.13.2. Использование линейной нейронной сети для прогнозирования .....	61
 Г л а в а 3. МНОГОСЛОЙНЫЕ ПЕРСЕПТРОНЫ.....	
3.1. ТОПОЛОГИЯ МНОГОСЛОЙНЫХ ПЕРСЕПТРОНОВ .....	66
3.2. АНАЛИЗ МНОГОСЛОЙНЫХ ПЕРСЕПТРОНОВ.....	67
3.2.1. Персептрон с одним скрытым слоем.....	67
3.2.2. Персептрон с двумя скрытыми слоями .....	74
3.3. НЕЙРОННЫЕ СЕТИ ВЫСОКОГО ПОРЯДКА.....	76
3.4. МАТЕМАТИЧЕСКИЕ ОСНОВЫ АЛГОРИТМА ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ .....	77
3.4.1. Последовательное обучение .....	79
3.4.2. Групповое обучение.....	82
3.5. ОБОБЩЕННОЕ ДЕЛЬТА-ПРАВИЛО ДЛЯ РАЗЛИЧНЫХ ФУНКЦИЙ АКТИВАЦИИ НЕЙРОННЫХ ЭЛЕМЕНТОВ .....	83
3.5.1. Сигмоидная функция .....	84
3.5.2. Биполярная сигмоидная функция .....	85
3.5.3. Гиперболический тангенс .....	86
3.5.4. Функция активации softmax.....	86
3.5.5. Ректификационная функция активации ReLU .....	88
3.6. АЛГОРИТМ ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ .....	88
3.7. НЕДОСТАТКИ АЛГОРИТМА ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ .....	90
3.8. РЕКОМЕНДАЦИИ ПО ОБУЧЕНИЮ И АРХИТЕКТУРЕ МНОГОСЛОЙНЫХ НЕЙРОННЫХ СЕТЕЙ .....	92
3.9. ГЕТЕРОГЕННЫЕ ПЕРСЕПТРОНЫ .....	97
3.10. АЛГОРИТМ МНОГОКРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ ....	99
3.11. ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ВХОДНЫХ ДАННЫХ.....	100
3.12. ПРИМЕНЕНИЕ МНОГОСЛОЙНЫХ ПЕРСЕПТРОНОВ .....	101
3.12.1. Классификация образов .....	102
3.12.2. Экспертные системы .....	103
3.12.3. Прогнозирование.....	104
3.12.4. Автономное управление автомобилем .....	106
3.12.5. Автономное управление роботом.....	109
 Г л а в а 4. РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ .....	
4.1. ОБЩАЯ АРХИТЕКТУРА РЕКУРРЕНТНОЙ НЕЙРОННОЙ СЕТИ .....	114
4.2. РЕКУРРЕНТНАЯ СЕТЬ ДЖОРДАНА .....	116
4.3. РЕКУРРЕНТНАЯ СЕТЬ ЭЛМАНА .....	117

4.4. ОБУЧЕНИЕ РЕКУРРЕНТНОЙ СЕТИ .....	119	
4.5. ПРИМЕНЕНИЕ РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЕЙ.....	122	
 Г л а в а 5. СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ .....		127
5.1. ПОСТРОЕНИЕ СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ.....	127	
5.2. АРХИТЕКТУРА СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ LENET-5.....	130	
5.3. ОБУЧЕНИЕ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ .....	132	
5.4. РЕДУЦИРОВАННАЯ АРХИТЕКТУРА СВЕРТОЧНОЙ СЕТИ ДЛЯ РАСПОЗНАВАНИЯ РУКОПИСНЫХ ЦИФР .....	134	
 Г л а в а 6. АВТОЭНКОДЕРНЫЕ НЕЙРОННЫЕ СЕТИ .....		136
6.1. МЕТОД ГЛАВНЫХ КОМПОНЕНТ.....	137	
6.1.1. Сжатие информации .....	139	
6.1.2. Восстановление информации.....	142	
6.1.3. Пример расчета по методу главных компонент .....	144	
6.2. АРХИТЕКТУРА АВТОЭНКОДЕРНОЙ НЕЙРОННОЙ СЕТИ.....	149	
6.3. ПРАВИЛО ОБУЧЕНИЯ ОЙЯ .....	151	
6.4. ОБОБЩЕННОЕ ДЕЛЬТА-ПРАВИЛО.....	155	
6.5. КУМУЛЯТИВНОЕ ДЕЛЬТА-ПРАВИЛО .....	157	
6.6. МЕТОД ПОСЛОЙНОГО ОБУЧЕНИЯ .....	159	
6.6.1. Теоретические основы метода .....	160	
6.6.2. Алгоритм послойного обучения .....	162	
6.7. АНАЛИЗ АВТОЭНКОДЕРНЫХ НЕЙРОННЫХ СЕТЕЙ .....	162	
6.8. ПРИМЕНЕНИЕ АВТОЭНКОДЕРНЫХ НЕЙРОННЫХ СЕТЕЙ .....	164	
6.8.1. Предварительная обработка данных .....	165	
6.8.2. Классификация и визуализация данных.....	165	
6.8.3. Обнаружение аномалий .....	166	
6.8.4. Разделение гауссовых сигналов.....	167	
6.8.5. Сжатие изображений .....	168	
 Г л а в а 7. РЕЛАКСАЦИОННЫЕ НЕЙРОННЫЕ СЕТИ .....		170
7.1. УСТОЙЧИВОСТЬ ДИНАМИЧЕСКИХ СИСТЕМ .....	170	
7.2. НЕЙРОННАЯ СЕТЬ ХОПФИЛДА .....	173	
7.2.1. Архитектура нейронной сети Хопфилда .....	173	
7.2.2. Нейронная сеть Хопфилда как динамическая система.....	175	
7.2.3. Энергия сети Хопфилда .....	177	
7.2.4. Анализ аттракторов .....	181	
7.2.5. Правило обучения Хебба .....	184	
7.2.6. Ассоциативная память .....	185	
7.2.7. Функционирование сети Хопфилда.....	190	
7.2.8. Решение задач оптимизации .....	193	
7.3. НЕЙРОННАЯ СЕТЬ ХЭММИНГА .....	197	
7.3.1. Архитектура сети .....	197	
7.3.2. Обучение.....	198	
7.3.3. Алгоритм функционирования .....	200	

7.4. ДВУНАПРАВЛЕННАЯ АССОЦИАТИВНАЯ ПАМЯТЬ .....	200
7.4.1. Архитектура.....	201
7.4.2. Обучение и функционирование.....	202
7.4.3. Алгоритм функционирования .....	203
 Г л а в а 8. САМООРГАНИЗУЮЩИЕСЯ НЕЙРОННЫЕ СЕТИ КОХОНЕНА .....	
8.1. ОБЩАЯ ХАРАКТЕРИСТИКА СЕТЕЙ КОХОНЕНА .....	205
8.2. КОНКУРЕНТНОЕ ОБУЧЕНИЕ.....	207
8.2.1. Определение нейрона-победителя по взвешенной сумме.....	208
8.2.2. Определение нейрона-победителя по евклидовому расстоянию.....	212
8.3. ВЕКТОРНЫЙ КВАНТОВАТЕЛЬ.....	214
8.3.1. Конкурентное обучение с одним победителем .....	215
8.3.2. Конкурентное обучение со многими победителями.....	216
8.3.3. Контролируемое конкурентное обучение .....	217
8.4. САМООРГАНИЗУЮЩИЕСЯ КАРТЫ КОХОНЕНА .....	218
8.5. РЕШЕНИЕ ЗАДАЧИ КОММИВОЯЖЕРА.....	222
 Г л а в а 9. ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ .....	
9.1. АРХИТЕКТУРА ГЛУБОКОЙ НЕЙРОННОЙ СЕТИ.....	227
9.2. ОБУЧЕНИЕ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ.....	228
9.3. АВТОЭНКОДЕРНЫЙ МЕТОД ОБУЧЕНИЯ .....	229
9.4. ОБУЧЕНИЕ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ НА ОСНОВЕ RBM.....	231
9.5. МЕТОД СТОХАСТИЧЕСКОГО ГРАДИЕНТА С ИСПОЛЬЗОВАНИЕМ RELU .....	238
9.6. АЛЬТЕРНАТИВНЫЙ ВЗГЛЯД НА ОГРАНИЧЕННУЮ МАШИНУ БОЛЬЦМАНА .....	239
9.6.1. Минимизация суммарной квадратичной ошибки .....	240
9.6.2. Минимизация кросс-энтропии .....	245
9.7. ПРИМЕНЕНИЕ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ.....	248
9.7.1. Сжатие данных.....	248
9.7.2. Визуализация данных .....	250
9.7.3. Классификация образов.....	252
БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ .....	254

Учебное издание

*Классическое университетское издание*

**Головко Владимир Адамович  
Краснопрошин Виктор Владимирович**

**НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ  
ОБРАБОТКИ ДАННЫХ**

**Учебное пособие**

Редактор *Н. Ф. Акулич*  
Художник обложки *Т. Ю. Таран*  
Технический редактор *Т. К. Раманович*  
Компьютерная верстка *А. А. Микулевича*  
Корректор *Е. В. Гордейко*

Подписано в печать 27.09.2017. Формат 60×90/16. Бумага офсетная.  
Печать офсетная. Усл. печ. л. 16,5. Уч.-изд. л. 17,78. Тираж 100 экз. Заказ 673.

Белорусский государственный университет.  
Свидетельство о государственной регистрации издателя, изготовителя,  
распространителя печатных изданий № 1/270 от 03.04.2014.  
Пр. Независимости, 4, 220030, Минск.

Республиканская унитарное предприятие  
«Издательский центр Белорусского государственного университета».  
Свидетельство о государственной регистрации издателя, изготовителя,  
распространителя печатных изданий № 2/63 от 19.03.2014.  
Ул. Красноармейская, 6, 220030, Минск.