МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение высшего профессионального образования
Таганрогский государственный радиотехнический университет

Брюхомицкий Ю.А.

НЕЙРОСЕТЕВЫЕ МОДЕЛИ ДЛЯ СИСТЕМ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Учебное пособие

Таганрог 2005

УДК 004.8.032.26
ББК 32.818

Брюхомицкий Ю.А. Нейросетевые модели для систем информационной безопасности. Учебное пособие. – Таганрог: Изд-во ТРТУ, 2005. – 160 с.

Рассмотрены цели, задачи и методология нейронного моделирования, как одного из разделов искусственного интеллекта. С использованием биологических аналогий показаны принципы построения, функционирования и применения искусственных нейронов, ансамблей и сетей. Даны основы построения, обучения, функционирования, а также области применения и характеристики наиболее распространенных специализированных искусственных нейронных сетей (нейронных парадигм), предназначенных для решения различных классов прикладных задач.

Состав материала и форма изложения ориентированы на применение аппарата искусственных нейронных сетей для построения интеллектуальных систем информационной безопасности.

Для студентов ВУЗов, обучающихся по 075-группе специальностей в области информационной безопасности.

Табл. 6. Ил. 78. Библиогр.: 18 назв.

Рецензенты:

Введение в интеллектуальные системы информационной безопасности

В XXI веке мир оказался на пороге новой цивилизации. Формируется новое – постиндустриальное информационное общество, в котором информация становится приоритетным объектом внимания и результатом деятельности человека. Переход информации в разряд важнейших ресурсов человечества порождает стремление к обладанию этим ресурсом и, как следствие – к появлению принципиально новой концепции противостояния и конфликтов – информационным войнам, средствами нападения и защиты в которых становится *информационное оружие*. История вооруженных конфликтов показывает, что эффективность наступательных средств в любой войне и информационной – в том числе, как правило, опережает эффективность систем защиты. В истории информационного противостояния об этом свидетельствует, в частности, обширная статистика успешных информационных атак на компьютерные системы самого различного уровня и назначения. Используемое при этом информационное оружие по существу представляет собой разнообразные средства осуществления несанкционированного доступа к информационным ресурсам (алгоритмические, программные, аппаратные и др.). Поэтому диалектика информационного конфликта выдвигает как актуальную в настоящее время – проблему создания эффективных адекватных средств информационной защиты. Выход этой проблемы на первый план связан еще и с тем, что современные системы управления являются системами критических приложений с высоким уровнем компьютеризации и оказываются весьма уязвимыми с точки зрения воздействия информационного оружия [1].

Разнообразие информационного оружия, форм и способов его воздействия, особенности появления и применения определяют предельно высокую сложность задач информационной защиты. Становится очевидным, что решение этих задач традиционными техническими и организационными средствами оказывается малоэффективным. Требуются принципиально новые идеи и подходы в организации систем информационной безопасности и информационного противодействия. Одним из таких подходов является использование аналогий с организацией жизненно важных поведенческих актов и принятием критических управленческих решений в биологических системах [1]. Современные научные достижения в таких областях информатики как: математическое моделирование состояния внешнего мира, искусственный интеллект, теория принятия решений, обработка изображений, сигналов и сцен, распознавание образов, оптимальное управление, и др. позволяют говорить о реальной возможности перехода к новому поколению средств информационной защиты – *интеллектуальным системам информационной безопасности*.

Из широкого спектра методов и средств искусственного интеллекта в сфере информационной защиты и информационного противодействия в

качестве теоретического базиса могут быть эффективно использованы и уже используются искусственные нейронные сети, М-сети, экспертные системы и системы, основанные на знаниях, эволюционные и генетические технологии, мультиагентные технологии и др.

Стремительно растет и перечень задач информационной безопасности, решаемых с использованием интеллектуальных методов и средств.

Пожалуй, первой актуальной задачей в сфере информационной безопасности, потребовавшей использования мощного арсенала методов и средств искусственного интеллекта, стала задача обнаружения вторжений и атак на автоматизированные информационные системы. Интенсивные исследования и разработки, проводимые в настоящее время в этом направлении, показывают, что достижение приемлемых уровней защиты информационных ресурсов от все более изощренных атак невозможно на основе применения обычных алгоритмических и программно-аппаратных решений. Современные средства обнаружения вторжений неизбежно должны включать в себя интеллектуальные подсистемы, по крайней мере, в качестве одной из своих составных частей. В настоящее время основу таких интеллектуальных подсистем в наибольшей степени составляют экспертные системы и искусственные нейронные сети.

Вместе с тем, задача обнаружения вторжений тесно связана с решением ряда других задач информационной безопасности, таких как, организация соответствующего информационного реагирования и противодействия, проведение периодического активного контроля имеющихся средств защиты, организация автоматизированного аудита событий безопасности и др. Эти задачи также могут эффективно решаться с помощью методов и средств искусственного интеллекта. Например, в качестве меры автоматизированного интеллектуального противодействия, в компьютерных системах могут создаваться ложные объекты атаки, имитирующие результаты работы или процессы функционирования реальных объектов атаки. Такие ложные объекты принимают на себя основные информационные потоки, порождаемые злоумышленниками, снижая их негативные воздействия на собственные информационные ресурсы [1]. Для комплексного решения этих взаимосвязанных задач создан и активно развивается новый интеллектуальный сервис информационной безопасности – *активный аудит*, который направлен на выявление подозрительной (злоумышленной и/или аномальной) активности с целью оперативного (в масштабе реального времени) принятия адекватных ответных мер. С этим сервисом связываются надежды на существенное повышение защищенности корпоративных информационных систем, потому что недостаточность традиционных механизмов, к сожалению, уже доказана практикой [2].

Один из главных рубежей информационной защиты в современных автоматизированных системах обработки информации образуют средства идентификации и аутентификации пользователей. Этот рубеж, являясь одним из

первых в цепи защитных механизмов компьютерных систем, подвержен наибольшему числу атак, значительное число из которых оказываются, к сожалению, успешными. Это обстоятельство привело к появлению и активному использованию в качестве средств идентификации и аутентификации разнообразных парольных систем, замково-ключевых устройств (смарт-карт, магнитных карт, брелков), а также биометрических и криптографических систем. Некоторые из этих систем используют в качестве интеллектуальной подсистемы распознавания образов пользователей – искусственные нейронные сети. В наибольшей степени это относится к биометрическим системам идентификации/аутентификации, в которых используются сложные, трудно поддающиеся обычной компьютерной обработке биометрические признаки личности, такие как, дактилоскопические узоры, сетка кровеносных сосудов, радужная оболочка глаз и др. В классе биометрических систем в последнее время стали интенсивно разрабатываться динамические методы идентификации/аутентификации, основанные на анализе особенностей голоса, рукописного и клавиатурного почерков пользователей компьютерных систем. Природная нестабильность и коррелированность биометрических данных такого типа также обуславливают преимущественное использование для их обработки аппарата искусственных нейронных сетей.

Новым перспективным направлением в развитии современных средств интеллектуального информационного поиска, защиты от вторжений и атак, организации реагирования и противодействия становятся *мультиагентные системы*, которые тоже широко используют средства искусственного интеллекта – искусственные нейронные сети, эволюционные и генетические технологии, нечеткие системы и др. [3].

Еще одним новым примером использования средств искусственного интеллекта – искусственных нейронных сетей – в области информационной безопасности являются системы обнаружения знаний в базах данных (KDD – Knowledge Discovery in Databases). Основное назначение KDD – автоматизированный поиск ранее неизвестных закономерностей в базах данных, хранящих информацию, и использование добытых знаний в процессе принятия решений [4].

Из всего обширного и динамично развивающегося перечня методов и средств искусственного интеллекта, в современных интеллектуальных системах информационной безопасности (как в теоретических исследованиях, так и в практических разработках), пожалуй, наиболее широко представлен аппарат искусственных нейронных сетей. Это обстоятельство и явилось побудительной причиной подготовки настоящего учебного пособия, призванного обогатить стандартные программы подготовки специалистов по информационной безопасности разделами по теории, принципам организации, функционирования и практического использования аппарата искусственных нейронных сетей.

1. Системы искусственного интеллекта

Введение в системы искусственного интеллекта

Людей всегда интересовал механизм их собственного мышления. История накопила множество попыток осмысления природы мышления, от духовных, до анатомических. Множество споров философов и теологов с одной стороны, а также физиологов и анатомов – с другой, принесло мало пользы для понимания вопроса о природе человеческого мышления. Предмет оказался предельно трудным для изучения.

В самом общем виде процесс познания человеком объектов окружающего мира можно отобразить схемой, показанной на рис. 1.1.

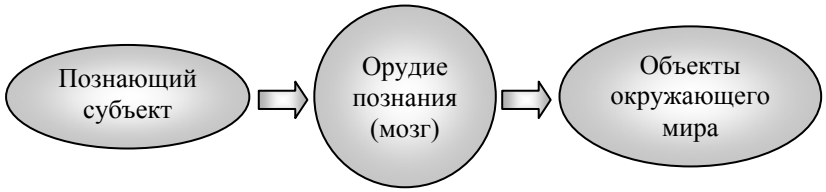


Рис. 1.1. Процесс познания человеком окружающего мира

Если познающий субъект (человек) в качестве объекта познания выбирает орудие познания, которым является его собственный мозг, то схема приобретает вид, показанный на рис. 1.2.

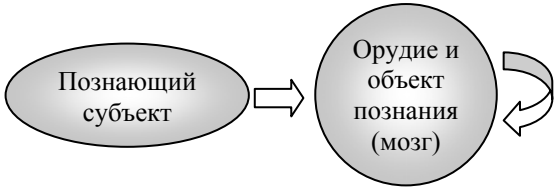


Рис.1.2. Процесс познания человеком собственного мозга

В первом случае процесс познания представляет собой открытую систему, во втором – замкнутую (систему с обратной связью), а из техники хорошо известно, что системы с обратной связью функционально гораздо богаче и сложнее открытых систем. Таким образом, процесс познания человеком собственного мозга не только обнаруживает чрезвычайно высокую сложность самого объекта, но и содержит определенную гносеологическую проблему.

С появлением в 40-х годах вычислительных машин и исследований по кибернетике, вопрос о природе человеческого мышления приобрел

кибернетический аспект. Ученые направили свои усилия на исследование возможности создания машин, обнаруживающих поведение, которое у людей мы называем интеллектуальным. Это направление исследований получило название «Искусственный интеллект» (ИИ). Поскольку и появление, и последующее развитие ИИ связано с вычислительными машинами, это направление обычно относят к области информатики и вычислительной техники.

С момента появления первых исследований по ИИ и по настоящее время целью этих исследований было отыскание некоторого универсального принципа обработки информации, лежащего в основе интеллектуальной деятельности человека и высших животных. Поскольку считается, что носителем интеллекта человека и животных, является мозг, то стремление искусственного воспроизведения интеллектуальной деятельности в системах ИИ должно, очевидно, каким-то образом опираться на изучение деятельности живого мозга.

Определенное представление о строении и принципах функционирования живого мозга дают исследования наук о мозге нейроанатомии и нейрофизиологии. Основываясь на результатах этих исследований, можно пытаться, путем создания искусственных математических и электронных аналогов клеток мозга (нейронов) и последующего их объединения в искусственные нейронные сети, пытаться достичь гипотетической возможности построения думающих машин. Однако в результате этих исследований выяснилось, что мозг имеет ошеломляющую сложность – около 10^{11} нейронов, каждый из которых соединен тысячами связей с другими нейронами. Все это образует невообразимо сложную нейронную паутину, далеко превосходящую самые смелые мечты о суперкомпьютерах. Тем не менее, накапливающиеся знания в этой сокровенной области позволили исследователям создать грубые математические модели для проверки своих теорий. Выяснилось, что эти модели позволяют не только имитировать отдельные функции мозга, но и способны выполнять функции, имеющие собственную прикладную ценность. Это обстоятельство породило две взаимно обогащающие друг друга цели нейронного моделирования, существующие до настоящего времени.

Первая цель – понять функционирование живого мозга как такового, вторая – создать специализированные электронные машины – *нейрокомпьютеры*, выполняющие функции, сходные с функциями живого мозга. В целом же, весь спектр «нейронных» исследований в системах ИИ получил название *бионического* или *структурного подхода*.

При изучении интеллектуальных систем информационной безопасности, как прикладной дисциплины, преследуется исключительно вторая цель, а именно – создание специализированных программно-аппаратных средств, использующих в своей структуре и принципах функционирования нейронные модели.

Другой, альтернативный структурному подход для моделирования мышления основан на попытках – обойти весьма сложные вопросы изучения живого мозга, путем имитации интеллектуальной деятельности по ее результату. Этот подход еще иногда называют подходом *«черного ящика»* или *«совпадением по результату»*. Он заключается в следующем. Постулируется, что исследователь, применяющий такой подход, не знает принципов построения и функционирования естественного интеллекта (живого мозга) и рассматривает его как «черный ящик». Его задача состоит в построении некоторых эвристических компьютерных программ, имитирующих интеллектуальную деятельность по конечному результату, не заботясь при этом об адекватности применяемых методов тем, которые заложены в мозге. Такой подход к моделированию мышления первоначально получил название *эвристического программирования* [5]. В подтверждение этого подхода приводился, в частности, пример, что компьютер прекрасно решает сложные вычислительные задачи лучше и быстрее человека, применяя при этом совсем другие способы и алгоритмы. Такой подход для моделирования мышления впоследствии стали называть *имитационным подходом.*

Эвристическое программирование связано с именами А. Ньюэлла и Г. Саймона из университета Карнеги (США) и основано на убеждении, что мышление человека по конечному результату может быть сведено к сочетанию простых задач манипулирования символами (сравнение, поиск, модификация), т.е. операций, которые могут легко выполняться компьютером. Решение задач при этом сводится к поиску (перебору) в пространстве возможных решений, а применяемые эвристические правила лишь помогают ускорить поиск и направить его к определенной цели [5].

Типичные задачи, которые удовлетворительно решались в рамках эвристического программирования: доказательство теорем, игры (шашки, шахматы, кубики, карты), решение головоломок, геометрические и шахматные задачи, сочинение музыки, определение химических структур и ряд других. Эти успехи воодушевили Г. Саймона в 1957 г. на предсказание, что через 90 лет компьютер станет чемпионом мира по шахматам. Предсказание фактически сбылось гораздо раньше – уже в 1997 г., когда шахматный компьютер Deep Blue обыграл чемпиона мира Гарри Каспарова. Вместе с тем, этот и последующие выдающиеся успехи компьютерного моделирования были обеспечены существенно более широким спектром методов и средств ИИ и вычислительной техники, далеко выходящим за рамки чисто эвристического поиска.

В настоящее время понятие ИИ объединяет в себя довольно широкий спектр научных направлений. Многие из них превратились в самостоятельные области знаний и дисциплины, другие, давшие хорошие прикладные результаты, перешли в практическую область человеческой деятельности.

Исторические аспекты

Официально возникновение исследований и разработок по ИИ относят к середине XX века. Однако теоретико-философские истоки знаний о свойствах человеческого мышления можно отслеживать, начиная еще с 5 века до н.э. от учений древнегреческих философов Сократа, Платона и Аристотеля, заложивших основы формальной логики. Через 2000 лет статическую формальную логику Аристотеля блестяще дополнил своей диалектикой немецкий философ Георг Гегель. Диалектическая логика Гегеля дала определенный ключ к моделированию динамической (творческой) компоненты человеческого мышления [6].

В IXX веке английский математик и логик Джордж Буль установил аналогию между логикой и алгеброй, представив логику как алгебру двоичных переменных, и разработал современный математический аппарат алгебры логики. Эта алгебра впоследствии стала называться булевой.

В 1950 г. английский математик и логик Алан Мэтисон Тьюринг предложил тест, позволяющий установить, является ли машина «думающей». Тест основывался на известной в то время салонной игре «в переговоры». Смысл игры состоял в том, чтобы по ответам на заданные вопросы играющий определил, с кем он имеет дело с мужчиной или женщиной. По тесту Тьюринга играющий должен был определить – с кем он имеет дело с человеком или машиной. Фактически Тьюринг предложил абстрактную универсальную вычислительную машину, копирующую работу человеческого мозга при выполнении им некоторого алгоритмического процесса. Такая машина впоследствии была названа «машиной Тьюринга» [7].

Развивая идеи своих предшественников, математик и логик Джон фон Нейман сформулировал принципы построения электронных вычислительных машин, материализующих абстрактную машину Тьюринга. Эти принципы лежат в основе большинства современных компьютеров, а соответствующая им архитектура называется классической или архитектурой фон Неймана. Эта ветвь исследований породила, по существу, всю вычислительную технику и основанный на ней имитационный подход к моделированию мышления.

Новый толчок в развитии систем ИИ, основанных уже на структурном подходе, связан с появлением науки об управлении – кибернетики. Ее автор – американский математик Норберт Винер постулировал инвариантность законов управления и связи в машине, живом организме и обществе. Иными словами, система управления может рассматриваться и изучаться независимо от своего материального носителя. Из этого следует важный вывод, что информационные механизмы мозга могут быть отделены от своего естественного носителя и реализованы в некоторой искусственной технической среде. Развитие этой идеи нашло свое воплощение в работах «второго отца» кибернетики, друга Н. Винера – американского нейрофизиолога Уоррена Мак-Каллока и его коллеги – математика Уолтера Питтса. В 1943 году в своей работе «Логическое

исчисление идей, относящихся к нервной активности» Мак-Каллок и Питтс показали возможность применения аппарата математической логики к моделированию функций нейронов. Они предложили в качестве упомянутой технической среды использовать искусственные нейронные сети (ИНС), элементами которых были искусственные нейроны, выполненные на бинарных пороговых преобразователях и функционирующие по принципу «все или ничего». ИНС могла выполнять любые математические и логические операции и, самое главное, была способна обучаться распознаванию образов, к обобщению, т.е. обладала свойствами, присущими исключительно живому мозгу, и которыми не обладают обычные компьютеры [8].

Последующий значительный вклад в развитие ИНС внес в начале 60-х годов XX века американский ученый Фрэнк Розенблатт. Основная идея Розенблатта сводилась к попытке заменить жесткие логические схемы Мак-Каллока и Питтса системами со статистическими свойствами. Такие системы Розенблатт назвал перцептронами (от лат. perceptio – перцепция, восприятие). Теорию перцептронов Розенблатт развил в своем фундаментальном труде «Принципы нейродинамики» (1962 г.). Первые успехи в моделировании перцептронов вызвали взрыв активности и оптимизма. На перцептронах удавалось решать довольно широкий класс задач, таких как: предсказание погоды, анализ электрокардиограмм, искусственное зрение и др. [9].

Параллельно с разработкой ИНС психологами были разработаны модели человеческого обучения. Одной из таких моделей, оказавшейся наиболее плодотворной, была модель Д. Хебба, предложенная им в 1949 в процессе исследования ассоциативной памяти. Эта модель послужила отправной точкой для многих алгоритмов обучения ИНС.

В течение некоторого времени казалось, что ключ к моделированию мозга найден, и техническое воспроизведение живого мозга является лишь вопросом конструирования достаточно большой ИНС. Но эта иллюзия скоро рассеялась. Перцептроны почему-то не справлялись со многими задачами, вроде бы аналогичными тем, которые они успешно решали. С этих необъяснимых неудач начался этап кропотливого анализа. В 1969 году в Кембридже появилась работа М. Минского и С. Пайперта «Персептроны», в которой они, используя точные математические методы, строго доказали ряд теорем по функционированию перцептронов [10]. Эти теоремы показали принципиальную ограниченность однослойных перцептронов и их неспособность решать многие простые задачи, в том числе реализовать функцию «исключающее ИЛИ», ставшую впоследствии простейшей тестовой задачей для многих ИНС. Блеск, строгость и неуязвимость аргументации Минского, его высокий авторитет в научной среде в совокупности с пессимистическими выводами относительно будущего ИНС привели к тому, что большинство исследователей и субсидии были переключены на другие более обещающие направления ИИ, а ИНС были фактически преданы забвению. Лишь небольшое число ученых (Кохонен,

Гроссберг, Видроу, Андерсон, Амари, Мальсбург, Фукушима и др.) продолжали исследования и разработки по ИНС.

Постепенное накопление теоретического и экспериментального фундамента в области нейронного моделирования, а также новые технологические возможности по микроэлектронному производству полупроводниковых устройств высокой степени интеграции в 80-х годах вновь привели к резкому возрастанию интереса к средствам параллельной обработки информации. Новый взрыв надежд и оптимизма начался в 1982 году. Поводом послужили несколько работ английского физика Дж. Хопфилда, который, работая со спиновыми стеклами, независимо от исследователей по ИНС показал и доказал большие возможности ИНС определенной конфигурации при решении многих оптимизационных задач и, в частности, задачи коммивояжера. Впоследствии ИНС такой конфигурации стали называть сетями Хопфилда [11].

С 80-х годов в области нейронного моделирования начался следующий виток спирали интенсивного развития. Количество научных центров, занимающихся этой междисциплинарной сферой знаний, непрерывно увеличивается. Разработка новых принципов обучения многослойных ИНС на основе обратного распространения ошибки сняли те ограничения для перцептронов, которые стали главным объектом критики в книге М. Минского и С. Пайперта. Оценка М. Минского оказалось, мягко говоря, излишне пессимистичной, и многие из упомянутых в его книге задач, как не решаемых на перцептронах, достаточно просто решаются сейчас нейронными сетями с помощью стандартных процедур. Урок истории на примере развития ИНС выражается известным законом Артура Кларка. «Если крупный уважаемый ученый говорит, что нечто может быть выполнено, то он почти всегда прав. Если же ученый говорит, что это не может быть выполнено, то он почти всегда не прав».

Масштабное увеличение финансирования предопределило существенный прогресс, как в теории, так и в практических приложениях ИНС. В сочетании с бурным развитием вычислительных систем технологии ИНС получают феноменально широкое распространение во многих прикладных областях.

Современные аппаратно-программные компьютерные системы, в основу функционирования которых положены ИНС, стали называться нейрокомпьютерами, а новая научная дисциплина, связанная с разработкой и исследованием методов их использования в различных практических областях – *«нейрокомпьютингом»*.

Классификация систем искусственного интеллекта

В различных литературных источниках до сих пор нет единого мнения по определению и классификации задач, принципов, методов и систем ИИ. Для ориентировки в этой обширной области, рассмотрим две группы понятий:

- Подходы, методы, концепции, теоретические идеи, т.е. теоретический фундамент, положенный в основу систем ИИ.
- Прикладные области использования методов и систем ИИ.

Классификация систем ИИ в соответствии с указанными группами понятий показана на рис. 1.3 [12].

Рис. 1.3. Классификация систем ИИ

В приведенной классификации направлениям 1 и 2 теоретического фундамента ИИ соответствуют два упомянутых ранее основных подхода к построению систем ИИ – структурный и имитационный. Третий подход также возник в середине XX века и базировался на гипотезе, что человеческий интеллект, в своем развитии эволюционировал, как и все живое, благодаря процессу, включающему мутации и естественный отбор. Суть подхода заключалась в том, что моделируемую на компьютере систему ИИ заставляли эволюционировать, искусственно имитируя процессы мутаций и отбора. При этом компьютерная эволюция проходила существенно быстрее, чем естественная, что позволяло за приемлемое время «сформировать» модель на решение задачи. На этом пути первоначально удалось добиться того, что системы ИИ эволюционировали до уровня, позволяющего решать простейшие задачи. Однако недостаточные (в то время) познания в механизмах естественной эволюции и скромные успехи по ее моделированию, как и в случае с ИНС, обусловили весьма слабый интерес исследователей к этому

направлению ИИ. В большинстве литературных источников по ИИ это направление даже не упоминается, хотя исторически оно было первым.

Начиная с 90-х годов, на новом витке спирали технического прогресса интерес к эволюционным методам ИИ опять резко возрос. Эволюционные модели были органично дополнены так называемыми генетическими алгоритмами и уже позволяли решать достаточно сложные прикладные задачи, суть которых сводилась обычно к многокритериальному поиску и оптимизации. Возникло также новое, перспективное направление ИИ, объединяющее эволюционно-генетический и нейронный подходы. В таких комбинированных системах эволюционные и генетические процедуры берут на себя сложную многокритериальную задачу построения и обучения ИНС [13].

Вторая группа понятий – прикладные области систем ИИ – вбирает в себя достаточно большое число направлений плодотворного использования методов и средств ИИ, которое непрерывно и интенсивно расширяется. Ниже кратко описаны только некоторые из них [12].

Обработка естественного языка. Когда люди общаются между собой на естественном языке (ЕЯ), они практически без всяких усилий используют сложные и пока мало понятные процессы. Оказалось, что построить машины, способные «понимать» хотя бы фрагменты ЕЯ, чрезвычайно трудно. Одной из причин является то обстоятельство, что ЕЯ возник как средство общения *интеллектуальных* существ, которые располагают весьма большими и сходными «умственными структурами». При общении людей между собой эти структуры играют роль колоссальной общедоступной контекстуальной базы знаний, опираясь на которую участники обмена могут создавать и воспринимать чрезвычайно сжатые сообщения.

Техническая система, способная понимать сообщение на ЕЯ, как и человек нуждается в контекстуальных знаниях и механизмах, обеспечивающих логический вывод на основе полученного сообщения и этих контекстуальных знаний, поскольку «генератор сообщения на ЕЯ» – человек, посылая сообщение, уже предполагает наличие этих знаний и механизмов.

К настоящему времени в этом направлении достигнуты реальные результаты в решении задач синтеза и анализа письменных и устных фрагментов речи, а также задач машинного перевода с одного языка на другой с определенными ограничениями. Направление вошло в практическую стадию разработок прикладных систем.

Извлечение знаний из баз данных. Базы данных (БД) представляют собой электронные хранилища, в которых могут находиться большие объемы сведений, относящихся к определенным предметным областям. БД формируется таким образом, чтобы из нее можно было извлекать нужные сведения в виде ответов на запросы по данной предметной области. В настоящее время существует много разнообразных приемов построения БД и способов извлечения информации из них. В рамках ИИ интересна другая постановка вопроса, а именно – как извлечь из БД ответ на вопрос, который,

хотя и содержится в БД в неявном виде на основе имеющихся в ней сведений, но его невозможно сформулировать в рамках стандартного запроса. Т.е. для получения ответа в явном виде необходимо провести некоторые дедуктивные рассуждения со сведениями, хранящимися в БД.

Это направление пока находится в стадии теоретических и экспериментальных исследований. В частности, известны попытки решения подобных задач с помощью ИНС [4].

Экспертные и консультирующие системы. Автоматические экспертные и консультирующие системы (АЭКС) призваны обеспечивать не очень квалифицированных специалистов компетентными заключениями, касающимися определенных предметных областей.

Ключевая проблема при построении АЭКС состоит в том, как представлять и использовать знания, которыми располагают и пользуются люди, являющими экспертами в предметных областях. Эта проблема осложняется тем, что экспертные знания часто являются неточными, неопределенными и плохо формализуемыми, что не мешает, тем не менее, экспертам делать правильные заключения.

Во многих АЭКС применяется метод ИИ, позволяющий строить *логические выводы, основанные на правилах*. В таких системах экспертные знания представлены в виде большого множества простых правил, которые применяются при организации диалога между системой и пользователем, а также для вывода заключений. Дедукция, основанная на правилах, является одним из плодотворных направлений в исследованиях по ИИ.

В настоящее время АЭКС – весьма динамично развивающаяся область. Известны и успешно эксплуатируются коммерческие АЭКС, диагностирующие заболевания, оценивающие потенциальные месторождения полезных ископаемых, предлагающие варианты возможных структур сложных органических соединений и многие другие.

Доказательство теорем. Поиск доказательства (или опровержения) некоторой математической теоремы, несомненно, может рассматриваться как интеллектуальная задача. Во-первых, потому что для доказательства требуется способность провести дедукцию, исходя из гипотез. Во-вторых, для построения доказательства необходимы интуитивные навыки, такие как построение догадки, о том, какие промежуточные леммы следует доказать, чтобы доказать основную теорему. Опытный математик, опираясь на некоторое собственное суждение, основанное на большом объеме специальных знаний, высказывает догадку, какие из ранее доказанных теорем в рассматриваемой предметной области будут полезны для искомого доказательства, а также выделяет в главной проблеме подзадачи, над которыми можно работать независимо друг от друга.

В рамках ИИ был разработан ряд программ, которые в какой-то степени обладали некоторыми из таких способностей. И хотя это не принесло большой практической пользы непосредственно для доказательства теорем, изучение

приемов доказательства теорем сыграло большую роль в развитии методов ИИ. Оказалось, что многие неформальные задачи допускают их формализацию как задачи на доказательство теорем.

Робототехника. В настоящее время это самостоятельная область науки и техники, выделившаяся из ИИ. Ее задачей является решение теоретических и практических вопросов организации целесообразного поведения подвижных кибернетических систем – роботов, снабженных сенсорными и эффекторными (исполнительными) механизмами. Перед роботами обычно ставится некоторая глобальная цель, например – достижение некоторой точки в окружающем пространстве. Однако предполагается, что заранее невозможно полностью предсказать все возможные взаимодействия робота с окружающей средой. Поэтому многие проблемы своего поведения в окружающей среде робот должен решать самостоятельно (методами ИИ), учитывая конкретные условия и используя при этом бортовой компьютер.

Исследования по робототехнике оказали существенное влияние на развитие многих идей ИИ. В частности, они привели к созданию методов описания и моделирования *состояния внешнего мира*, дали лучшее понимание того, каким образом строить *планы* для последовательности действий робота и как управлять выполнением этих планов. Методы планирования действий робота стали строить как многоуровневые системы с высоким уровнем абстракции на верхнем уровне и все более детализированные на последующих уровнях.

Автоматическое программирование. Существующие компиляторы в некотором смысле уже осуществляют автоматическое программирование. Они воспринимают полную спецификацию во входном коде того, что программа должна делать, и пишут программу в объектном коде, которая это делает. Под автоматическим программированием понимается некий «суперкомпилятор», который мог бы воспринимать описание на очень высоком уровне вплоть до ЕЯ того, что требуется от искомой программы. При этом, учитывая высокий уровень входного описания, а, следовательно, и наличие большого числа неоднозначностей в этом описании, автоматическое программирование предполагает, дополнительный диалог между системой и пользователем для исключения этих неоднозначностей.

Задача автоматического написания программы для достижения заданного результата тесно связана с задачей доказательства того, что программа вообще достигнет этого результата. Последняя задача получила название *верификации программы*. Поэтому многие системы автоматического программирования включают верификацию выходной программы в качестве некоторой дополнительной возможности.

Важным вкладом в автоматическое программирование явилось заимствованное из робототехники представление об отладке программ как стратегии решения проблем. Установлено, в частности, что часто более эффективным оказывается создание недорогой, с большим количеством ошибок

программы, с последующей ее модификацией, чем поиск с самого начала идеальной программы, лишенной дефектов.

Комбинаторные задачи и составление расписаний. Многие задачи из этой области исследуются методами ИИ. Классическим примером является *задача коммивояжера,* в которой требуется найти маршрут минимальной длины в пределах нескольких городов, начиная от некоторого исходного города, посещая каждый город один раз и возвращаясь в исходный город.

Такой же характер носят многие головоломки. Например, *задача о восьми ферзях,* которая состоит в том, что восемь ферзей надо разместить на обычной шахматной доске так, чтобы ни один из них не атаковал другого. В задачах такого типа область возможных комбинаций или последовательностей, из которых необходимо выбрать ответ, чрезвычайно велика. Простые попытки решения таких задач вскоре порождает *комбинаторный взрыв* вариантов, который быстро исчерпывает возможности современных компьютеров.

Некоторые из этих задач (включая задачу коммивояжера) математики относят к так называемым *NP-полным (недетерминистки полиномиальным) задачам*. Трудность их решения состоит в высокой степени возрастания числа комбинаций (числа шагов или времени решения), в зависимости от какой-либо характеристики объема задачи. В задаче коммивояжера такой характеристикой является число городов. Степень возрастания комбинаций в задачах может быть линейной, полиномиальной и экспоненциальной. В NP-полных задачах время решения растет экспоненциально с увеличением объема задачи, и пока еще не установлено, существует ли более быстрый, чем экспоненциальный метод решения NP-полных задач. Поэтому усилия специалистов, работающих над решением комбинаторных задач методами ИИ, сводятся к поиску квазиоптимальных решений, удовлетворяющих по точности, при существенно меньших затратах времени. Интересные результаты в решении NP-полных задач (в том числе и задачи коммивояжера) достигнуты, в частности, с помощью рекуррентных ИНС Хопфилда [11].

Зрительное восприятие. Попытки снабдить компьютеры «глазами» показали, что для интеллектуального восприятия и обработки столь сложных входных данных, какими являются зрительные сцены и образы, как и в случае с естественной речью, необходим компонент «понимания». В свою очередь, для моделирования этого компонента, как уже говорилось выше, необходима колоссальная база знаний об окружающей среде.

Смысл процесса машинного зрительного восприятия состоит в создании сжатого представления о реальных входных сценах и образах, с которыми машина не в состоянии работать из-за огромного объема описывающей их информации. При этом характер и качество окончательного представления зависит от целей воспринимающей системы (что является важным: цвета, пространственные соотношения, размеры, наличие определенных объектов, признаков и т.д.).

Основная трудность при восприятии сцен – невообразимое число возможных описаний. Одна из возможных плодотворных стратегий в этих условиях состоит в построении гипотез на различных уровнях описания с последующей их проверкой при помощи набора детекторов. В свою очередь, процесс формирования гипотез также нуждается в большом объеме знаний о сценах, которые ожидается увидеть.

В настоящее время зрительное машинное восприятие традиционно включает в себя следующие направления:

методы анализа трехмерных сцен;

методы представления информации о зрительных образах в базе знаний;

методы перехода от зрительных сцен к их текстовому описанию и обратного перехода;

методы порождения зрительных сцен на основе внутренних представлений в системах ИИ;

методы и средства *когнитивной графики*.

Примечание. *Когнитивная графика это направление ИИ, ориентированное, прежде всего, на порождение качественно нового знания в конкретных предметных областях, и, прежде всего, в фундаментальных науках. Когнитивная графика реализуется в виде интерактивных компьютерных графических систем, в которых с помощью искусственной визуализации и озвучивания различных объектов и понятий предметной области, в том числе абстрактных (например, чисел), синтезируются абстрактные графические и звуковые построения (озвученные мультфильмы). Цель синтеза таких построений – целенаправленная активизация высших творческих метапроцедур образного, интуитивного, правополушарного мышления человека, направленная на повышение его творческих возможностей и генерацию ответных реакций в форме новых идей, гипотез и тому подобное, т.е. – на порождение качественно нового знания.*

Представленный перечень практических приложений ИИ является далеко не полным и имеет тенденцию быстро расширяться. Примером являются интенсивные исследования и разработки по использованию методов и средств ИИ в области информационной безопасности, породившие новое направление – ИСИБ.

Классификация систем ИИ по моделированию способностей разумных организмов. Кроме описанной выше классификации систем ИИ можно предложить и другую, суть которой состоит в следующем. ИИ ставит своей целью – моделирование интеллектуальной деятельности, свойственной разумным организмам (человеку). Но разумный живой организм имеет определенный набор интеллектуальных способностей, каждая из которых может служить объектом моделирования. В этот набор обычно включают: представление знаний, манипулирование знаниями, общение, восприятие и обучение. Тогда классификацию систем ИИ можно провести по принципу, какую из этих способностей (свойств) или их комбинаций моделирует система

ИИ. Классификация систем ИИ в соответствии с этим подходом показана на рис. 1.4 [12].

Рис. 1.4. Классификация систем ИИ

Такой подход можно также интерпретировать, как стремление построить в конечном итоге некую искусственную целостную, интегральную, интеллектуальную систему. Следует, однако, подчеркнуть, что указанный набор свойств следует рассматривать лишь как гипотетический. В различных практических областях применения систем ИИ может потребоваться только определенная часть из этого набора. Примером интеллектуальной системы, в которой в той или иной степени желательно наличие большинства из перечисленных свойств является интегральный робот, предназначенный для выполнения широкого (полностью не определенного) круга задач в естественной внешней среде. Еще более сложный вариант такой системы – коллектив интегральных роботов, взаимодействующих с окружающей средой и между собой. Подобные разработки уже проводятся (роботы-исследователи, микро-роботы и др.).

2. Мозг – как биологический прототип искусственных нейронных сетей

Организация мозга. Нервная система (НС) человека и животных является важнейшей консолидирующей системой организма. Ее основная функция заключается в поддержании внутренней гармонии организма и в организации его приспособительной деятельности в изменяющихся условиях внешней среды. НС имеет клеточную структуру и состоит из нервных клеток – *нейронов*, сгруппированных в нейронные ансамбли и сети. Центральным отделом НС является головной и спинной мозг [6]. Человеческий мозг содержит более 10^{11} нейронов, которые связаны между собой примерно 10^{14} связями. Эта невообразимая паутина отвечает за все наши знания, мысли, эмоции, интуицию, а также – за мириады функций и реакций в нашем организме. Пока мало понятно как все это происходит. Чем больше ученые узнают о мозге, тем больше понимают, сколь сложен и совершенен объект их исследования.

Кроме нейронов мозг содержит густую сеть кровеносных сосудов, обеспечивающих его кислородом и питательными веществами. Весь остальной объем мозга заполнен *глиальными клетками – глиями*, роль и функции которых пока не полностью понятны.

Система кровоснабжения мозга связана с главной системой кровообращения посредством высокоэффективной фильтрующей системы, называемой *гематоэнцефалическим барьером*. Этот барьер предохраняет мозг от возможных токсичных веществ, находящихся в крови. Защита обеспечивается низкой проницаемостью кровеносных сосудов мозга, а также глиальными клетками, окружающими нейроны. Гематоэнцефалический барьер является основой для обеспечения сохранности мозга, но он значительно осложняет лечение терапевтическими лекарствами. Он также препятствует изучению влияния различных химических веществ на функции мозга. Лишь небольшая часть лекарств, созданных специально с целью влияния на мозг, может преодолевать этот барьер. Такие лекарства состоят из небольших молекул, способных проникать через крошечные поры в кровеносных сосудах. Чтобы воздействовать на функции мозга, они должны затем пройти через глиальные клетки или раствориться в их мембране [14].

Мозг является основным потребителем энергии тела. Включая в себя лишь 2% массы тела, в состоянии покоя он использует приблизительно 20% кислорода тела. Потребляя только 20 Вт, мозг с энергетической точки зрения невероятно эффективен [14].

С точки зрения кибернетики мозг представляет собой информационно-управляющую систему, которая при помощи рецепторов (зрительных, слуховых, тактильных, химических, температурных и т.д.) воспринимает информацию о внешней среде, обрабатывает эту информацию и формирует управляющие воздействия на эффекторные (исполнительные) системы организма (мышцы, сосуды, железы внутренней секреции). С кибернетических

позиций структура мозга весьма грубо может быть представлена схемой, показанной на рис. 2.1 [6].

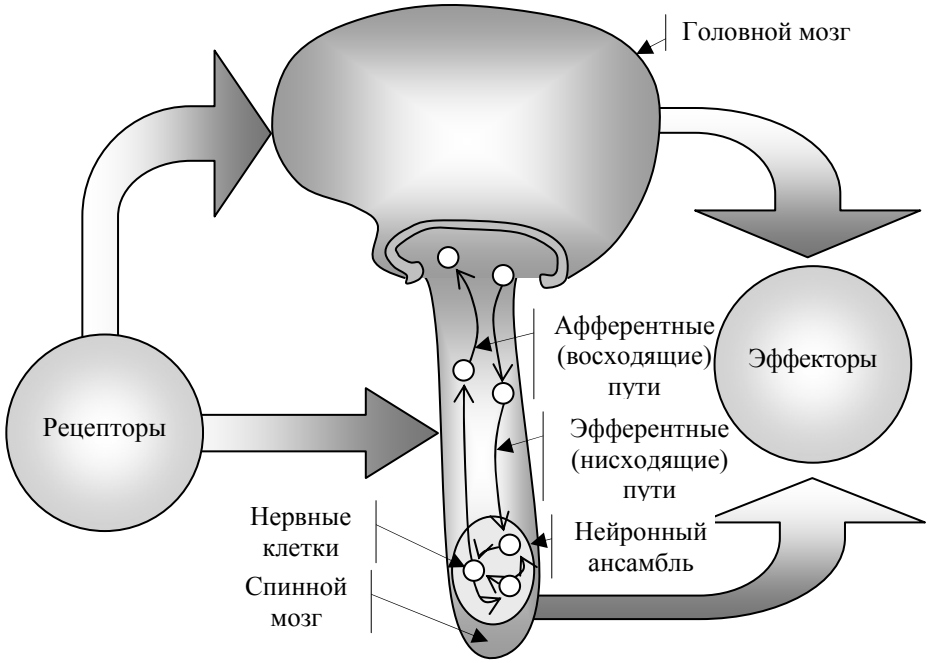


Рис. 2.1. Структура мозга

По месту нахождения и функциям в нервной системе различают сенсорные (рецепторные) нейроны, вставочные (интернейроны) и эффекторные (мотонейроны). Рецепторные нейроны воспринимают энергетические воздействия и сигналы внешней среды (для НС внутренняя среда организма также является внешней). Взаимодействующие друг с другом интернейроны осуществляют внутреннюю обработку информации, а мотонейроны передают результаты этой обработки непосредственно на исполнительные системы организма, в качестве которых выступают мышцы и железы внутренней секреции. Комбинации возможных нейронных связей отражены в табл. 2.1.

Нервные волокна, передающие в головной мозг возбуждения из спинного мозга и периферических отделов головного мозга, называются *афферентными* (восходящими) путями. Возбуждения, направленные обратно из головного мозга называются *эфферентными* (нисходящими) путями.

Таблица 2.1

Комбинации возможных нейронных связей

Входы	**Выходы**	
рецептор	нейрон	←рецепторный нейрон
нейрон	нейрон	← интернейрон
нейрон	мышца, железа	← мотонейрон

Нервные клетки. Нервная клетка или сокращенно нейрон является основным строительным элементом нервной системы (рис. 2.2).

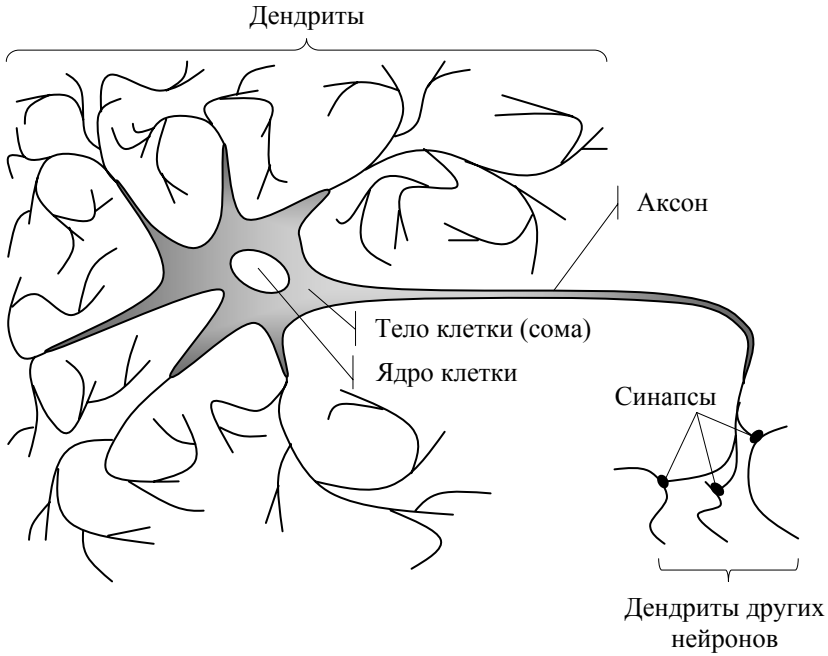


Рис. 2.2. Структура нейрона

Как любая другая клетка, нейрон имеет тело со стандартным набором органелл, называемое *сомой.* Внутри сомы располагается *ядро* клетки, покрытое ядерной оболочкой – мембраной. Ядро, как правило, − одно, хотя в вегетативной НС встречаются и многоядерные нейроны [14]. Из сомы нейрона выходят многочисленные отростки, играющие ключевую роль в его взаимодействии с другими нервными клетками, а также мышцами и железами. С точки зрения нейронного моделирования выделяют два типа отростков:

многочисленные тонкие ветвящиеся *дендриты* и более толстый, расщепляющийся на конце *аксон* (*нейрит*).

Сома нейрона отвечает за управление расходом энергии, питание, обновление ресурсов и множество других процессов. Были идентифицированы сотни типов нейронов, каждый со своей характерной формой тела: клетки Пуркинье, пирамидальные клетки, корзинчатые клетки, клетки Гольджи, звездчатые клетки, зернистые клетки и мн. др. Считается, что форма тела нейрона целиком зависит от его местоположения в соответствующем органе НС. Совершенно одинаковые в функциональном отношении нейроны могут иметь разную форму тела. На форму тела нейрона могут оказывать влияние соседствующие кровеносные сосуды, пучки нервных волокон и даже внешние воздействия. Поэтому, скорее всего, между формой нейрона и его морфологической и функциональной характеристиками нет никакой существенной взаимосвязи [14].

Размеры нейронов колеблются в широких пределах в зависимости от уровня организации животных, местоположения и функционального назначения нейронов в НС и других факторов. Например, клетки мозжечка имеют размер около 5 мкм, моторные клетки головного и спинного мозга – 70 мкм, а тела нейронов брюхоногих моллюсков достигают размеров 500-900 мкм и хорошо различимы невооруженным глазом [14].

Нервная клетка подобно всем другим клеткам покрыта сплошной плазматической мембраной толщиной 6 нм, отделяющей ее от внешней среды. Внешняя мембрана сомы обладает уникальной способностью генерировать нервные импульсы (потенциалы действия), являющейся основой информационной деятельности мозга [14].

Дендриты нейрона имеют диаметр менее 1 мкм, длину до 700 мкм и в совокупности образуют густо ветвящееся дендритное дерево различной формы. Преимущественно на дендритах располагаются контакты, называемые *синапсами*, через которые нейроны получают входные сигналы от других нейронов и рецепторов и передают их в сому [14].

Аксон – это одиночный утолщенный отросток, выходящий из сомы (иногда – из дендрита). Он имеет толщину от 0,3 до 16 мкм и длину от 0,1 мм до более чем 1 м. На конце аксон разветвляется на множество ветвей – *аксонных коллатералей*, каждая из которых завершается синапсом. Аксон передает выходные сигналы нейрона через синапсы на дендриты других нейронов. Такого рода синаптические связи считаются основными и называются *аксо-дендритические* [14].

Синапс. Синапсы делятся на химические и электрические. Подавляющее большинство синапсов человека и животных относятся к химическим синапсам. Синаптический контакт химического синапса состоит из двух контактирующих частей: передающей – *пресинаптической*, представленной окончанием аксона, и приемной (рецепторной) – *постсинаптической*, представленной участком дендрита, сомы или аксона (в

зависимости от типа взаимодействия). Обе части контакта разделены узким пространством шириной около 20 нм, называемым *синоптической щелью*. Пресинаптическая часть аксона содержит маленькие утолщения, содержащие сферические структуры, называемые *синаптическими пузырьками* или *везикулами*, размером от 60 до 200 нм [14].

Когда нервный импульс приходит в аксон, некоторые из этих пузырьков высвобождают свое содержимое в синаптическую щель, тем самым, инициализируя процесс взаимодействия нейронов. Химические вещества, выбрасываемые в синаптическую щель, по своему функциональному назначению (а не химическому содержанию) называются *медиаторами* или *нейротрансмиттерами*. Медиаторы затем улавливаются на постсинаптической части специальными рецепторами дендрита и внедряются в тело клетки (рис. 2.3) [14].

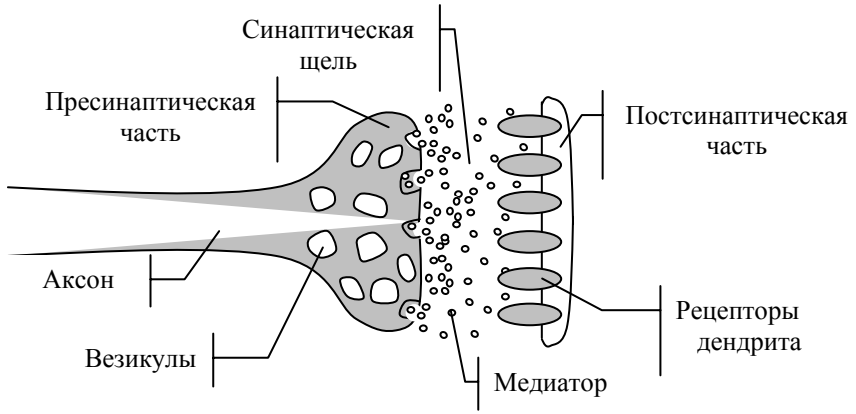


Рис. 2.3. Синапс

Определено более 30 видов медиаторов. Некоторые из них стремятся вызвать возбуждение клетки и называются *возбуждающими*, другие наоборот подавляют возбуждение и называются *тормозными* [14].

Электрические синапсы встречаются значительно реже, чем химические. Они описаны в НС беспозвоночных, низших позвоночных и птиц. Характерной чертой этих синапсов является почти полное слияние пре- и постсинаптических полюсов контакта. Синаптическая щель в них сужена до 2 нм. Химическая передача в электрических синапсах полностью отсутствует, и возбуждение от нейрона к нейрону передается непосредственно на электрическом уровне [14].

У некоторых видов животных обнаружены также редкие случаи синапсов со смешанной химической и электрической передачей [14].

Связи от аксонов к дендритам в нейронах реализуют основной тип – аксо-дендритического взаимодействия. Однако в НС наблюдаются и другие типы

взаимодействия. Довольно часто синаптические связи идут от аксона к аксону. Они называются *аксо-аксональными*. Считается, что такие связи выполняют роль пресинаптического торможения. Наблюдаются синаптические контакты между дендритами одного и того же нейрона или дендритами разных нейронов. Т.е. дендрит может выполнять не только роль рецепторной части нейрона, но и его передающей части – аксона. Такие специализированные контакты называются *дендро-дендритическими* контактами. На ограниченных участках сомы некоторых нейронов были обнаружены зоны, характерные для пресинаптической части синапса, а в непосредственной близости на дендрите – соответственно специализация для постсинаптической части синапса. Такие контакты получили название *сомато-дендритических* синапсов. Позднее были обнаружены также *дендро-соматические* и *сомато-соматические* синапсы [14].

Синапсы отличаются друг от друга размерами и возможностью концентрации медиаторов. По этой причине импульсы одинаковой величины, поступающие на входы нервной клетки через различные синапсы, могут возбуждать ее в разной степени. С точки зрения нейронного моделирования этот эффект можно трактовать как коэффициент синаптической передачи (связи) или просто – *вес связи*. При этом положительные значения веса связи соответствуют возбуждающим синапсам, отрицательные значения – тормозным синапсам, нулевые значения – отсутствию синаптической передачи (связи).

Функционирование нейрона – весьма сложный электрохимический процесс. Синаптическая передача в нейроне имеет, как правило, химическую природу, а передача сигналов внутри нейрона – электрическую.

Специфическая особенность нервных клеток заключается в способности воспринимать, преобразовывать и передавать на другие клетки нервные возбуждения в виде нервных импульсов – *спайков*. Для аксо-дендритического типа взаимодействия нейронов весьма упрощенно это процесс выглядит так. Входные нервные импульсы через синаптические контакты на дендритах поступают в сому, где они суммируется с другими аналогичными сигналами и формируют внутренний *мембранный потенциал нейрона*. Причем одни синапсы являются возбуждающими и увеличивают потенциал нейрона, другие – тормозными, уменьшающими этот потенциал. Если суммарное возбуждение тела клетки – мембранный потенциал нейрона не сильно превосходит уровень электрического равновесия – *потенциал покоя* V_0, либо баланс возбуждений и торможений является отрицательным, мембранный потенциал нейрона возвращается в исходное состояние покоя, и на выходе нейрона никакого сигнала не возникает. Если же суммарное возбуждение тела клетки превысит некоторую характерную для нее критическую величину, называемую *порогом*, потенциал нейрона начинает лавинообразно нарастать, и в области сомы, прилегающей к аксону, возникает спайк или *потенциал действия* $V_п$. Причем величина спайка не зависит от степени превышения порога, т.е. нейрон работает по принципу «все или ничего». Возникнув, спайк без затухания распространяется по аксону и его коллатералям и далее через синапсы

поступает на дендриты других нервных клеток, вызывая их возбуждение или торможение. С позиции моделирования информационной деятельности нейрона спайк можно рассматривать как выходной сигнал нейрона, который по своей природе является импульсным или частотно-модулированным.

Одновременно с генерацией спайка в клетке запускается процесс *рефракции*. Он проявляется как стремительное возрастание порога активации клетки до значения «плюс бесконечность». В результате сразу после генерации импульса нейрон теряет способность вырабатывать очередной спайк даже при сильном возбуждении. Такое состояние сохраняется в течение времени Δt_{r1}, равным примерно 1 мс, и называемым *периодом абсолютной рефрактерности*. По его окончании наступает период *относительной рефрактерности* Δt_{r2}, равный примерно 3 мс, в течение которого порог активации возвращается к первоначальному значению. В это время клетка может быть активирована, но только с приложением более сильных сигналов возбуждения. Процесс возбуждения нейрона и генерации спайка иллюстрируется на рис. 2.4.

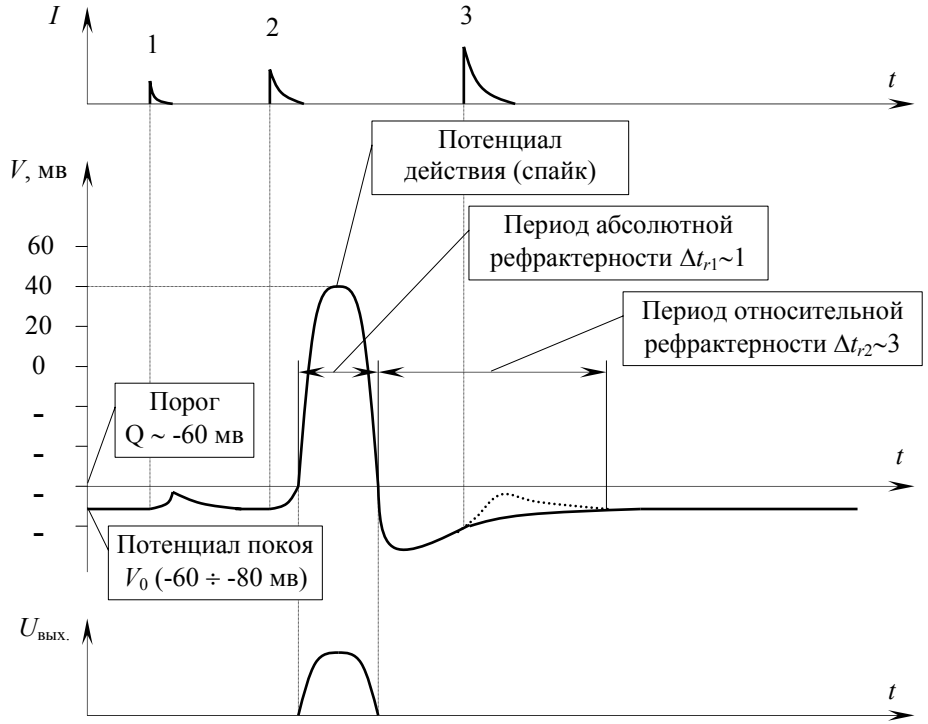


Рис. 2.4. Процесс возбуждения нейрона и генерации спайка

Нервные клетки изучаются нейроанатомией и нейрофизиологией. В нейроанатомических исследованиях используются оптические и сканирующие электронные микроскопы с разрешающей способностью менее 0,1 мкм. В нейрофизиологических исследованиях широко применяется микроэлектродная технология, основанная на вживлении электродов в мозг. Применяется также математическое моделирование. Его суть состоит в разработке математических моделей различных элементов и механизмов НС, помогающих в изучении живого мозга.

Мозг и компьютеры. С точки зрения вычислительных способностей считается, что отдельный нейрон выполняет относительно примитивные функции суммирования взвешенных входных сигналов, сравнения полученной суммы с пороговым значением и генерации выходного сигнала. Вместе с тем, каждый нейрон имеет свои собственные синаптические коэффициенты при суммировании входных сигналов и свой порог возбуждения. Эти параметры определяются местонахождением нейрона и решаемой им задачей и могут интерпретироваться аналогично содержимому локальной памяти процессора. Огромное количество индивидуально настроенных нейронов мозга, взаимодействующих между собой по многочисленным межнейронным связям, образуют колоссальную степень связности. Именно эта связность, а не функциональная сложность отдельного нейрона, по-видимому, обеспечивает общий высокий уровень вычислительной мощности мозга [15].

Скорость выполнения операций отдельным нейроном, измеряемая миллисекундами, существенно меньше в сравнении с наносекундными интервалами работы процессоров современных компьютеров. Однако, высокая степень параллелизма, обеспечиваемая огромным числом параллельно функционирующих нейронов и межнейронных соединений, предопределяют высокую скорость работы мозга в целом. Такие задачи, как распознавание образов, зрительных сцен, принятие решений выполняются мозгом за миллисекунды. Достижение аналогичных результатов при использовании компьютеров на основе полупроводниковых технологий, даже с циклом срабатывания процессоров в несколько наносекунд, пока еще невозможно. Ни одна современная технология не позволяет также построить искусственную нейронную сеть, близкую по масштабам и уровню функциональности к нейронной сети мозга [15, 16].

Специфика обычных компьютеров заключается в том, что они работают безошибочно только при точных входных сигналах и при отсутствии аппаратно-программных сбоев и повреждений. Мозг же способен давать удовлетворительное решение многих жизненно важных для организма задач в условиях недостатка, незавершенности и неточности входных данных, а также при отказах и повреждениях целых своих участков. Огромное число параллельно функционирующих нейронов и межнейронных соединений приводит к тому, что ошибки и отказы в срабатывании отдельных нейронов и связей остаются незаметными в общей массе взаимодействующих клеток.

Нейронная сеть мозга проявляет очень высокую устойчивость к помехам – это «стабильная» сеть, в которой отдельные сбои не оказывают существенного влияния на результаты ее функционирования. Высокая степень избыточности нервных клеток и межнейронных связей мозга позволяет ему при наличии сбоев и повреждений многовариантно реконфигурировать нейронные соединения, обеспечивая высочайшую надежность и робастность всей системы в целом [16].

Таким образом, мозг и компьютер существенно отличаются между собой. Они оптимизированы для решения различных типов проблем, имеют разную структуру, принципы функционирования, способы решения задач, их работа оценивается различными критериями. В этой связи, моделирование нервной деятельности, по крайней мере, пока, не может рассматриваться с позиций построение технического аналога мозга как такового [16].

В настоящее время нейронное моделирование может преследовать две основные цели. Первая состоит в попытке изучения живого прототипа – мозга и может рассматриваться как способ получения новых знаний о строении и деятельности мозга. Вторая – заключается в использовании принципов построения и функционирования мозга для решения практических задач по обработке информации, трудно поддающихся решению другими средствами. В рамках данного пособия описываются нейросетевые системы, ориентированные на использование исключительно в рамках второй упомянутой цели. В этой связи все аналогии описываемых моделей нейронов и нейронных сетей с их биологическими прототипами следует рассматривать с очень большой долей условности. Используемая в таких моделях терминология, часто заимствованная из нейроанатомии и нейрофизиологии, – не более чем исторически сложившаяся традиция.

3. Искусственные нейроны

Структурный (бионический) подход к моделированию мозга реализуется на нескольких уровнях (этапах). Вначале создается информационная модель отдельной нервной клетки – *искусственного нейрона* (ИН), что составляет первый уровень нейронного моделирования. Ограниченное число ИН далее могут структурироваться в жесткие необучаемые конфигурации – *искусственные нейронные ансамбли*, что составляет второй уровень нейронного моделирования. Наконец, создаются конфигурации из большого числа ИН, которые с помощью специальной процедуры обучения могут гибко изменять свои параметры. Такие конфигурации называются *искусственными нейронными сетями*. Они составляют третий уровень нейронного моделирования.

Реализация первого этапа нейронного моделирования – создание ИН – тесно связана с целями такого моделирования. Если целью является изучение живого мозга, то используются те методы и модели, которые позволяют в наибольшей степени отразить изучаемые явления, и, тем самым, – помочь самому процессу познания мозга. Такие нейронные модели обычно очень сложны и создаются, как правило, для перманентного использования в нейроанатомических и нейрофизиологических исследованиях. Если целью нейронного моделирования является создание специализированных вычислительных систем – нейрокомпьютеров для решения практических задач по обработке информации, то модели должны, в первую очередь, отвечать потребительским требованиям, т.е. быть простыми, дешевыми и способными эффективно решать поставленные перед ними практические задачи. Как уже отмечалось, в рамках настоящего пособия преследуется вторая из поставленных целей, поэтому описываемые подходы и методы ограничиваются исключительно потребностями нейроинформатики и нейрокомпьютинга.

В зависимости от степени отражения при моделировании различных элементов, свойств и механизмов биологического нейрона, применения тех или иных способов представления и кодирования сигналов, использования того или иного вида активационной функции предложено множество самых разнообразных моделей нейронов. В настоящем пособии будут рассмотрены только две модели – формального и градуального нейронов, получившие наибольшее распространение в практике нейронного моделирования.

Формальные нейроны. Наиболее простой физически реализуемой информационной моделью нервной клетки, предложенной в 1943 г. Мак-Каллоком и Питтсом, является, так называемый, *формальный нейрон* (ФН) [8]. В основе построения ФН лежит представление о нервной клетке как о логическом элементе, работающем по принципу «все или ничего». Предполагается, что модель воспроизводит только аксо-дендритические синаптические взаимодействия. Входные $x(t_i)$ и выходные $z(t_{i+1})$ сигналы аппроксимируются единичными импульсами прямоугольной формы или единичными потенциалами. Тем самым в модели ФН постулируется, что $x(t_i)$,

$z(t_{i+1})$ являются булевыми переменными, $x(t_i)$, $z(t_{i+1}) \in \{0,1\}$. ФН работает в дискретном времени

$$t_i = t_o + i\delta t,$$

где $\delta t = (t_i - t_{i-1})$ – шаг квантования времени t.

Совокупность всех сигналов, поступающих в дендритное дерево нейрона через синапсы, трактуется в ФН как множество (вектор) входных сигналов в момент времени t_i:

$$\mathbf{X} = [x_1(t_i), x_2(t_i), \ldots, x_j(t_i), \ldots, x_N(t_i)], \quad j = \overline{1, N}.$$

Любой входной сигнал $x_j(t_i)$ нейрона может поступать извне, т.е. являться входным сигналом всей нейронной системы, а также может быть выходным сигналом этого же или другого аналогичного нейрона. Каждый входной сигнал $x_j(t_i)$ умножается на свой синаптический (весовой) коэффициент (вес) w_j, имитирующий силу синаптической передачи по соответствующему входу. Положительные значения весов w_j соответствуют возбуждающим синапсам, отрицательные значения w_j – тормозным синапсам, нулевые значения w_j – отсутствию синаптической связи. Полученные произведения $w_j \cdot x_j(t_i)$ затем суммируются, образуя внутренний мембранный потенциал $V(t_i)$ нейрона. Операция суммирования моделирует накопление мембранного потенциала сомой нейрона:

$$V(t_i) = \sum_{j=1}^{N} w_j \cdot x_j(t_i).$$

Поскольку в ФН входные сигналы $x(t_i)$ являются булевыми переменными $x(t_i) \in \{0,1\}$, реализация операции накопления мембранного потенциала $V(t_i)$ сводится к выборочному суммированию весовых коэффициентов, соответствующих единичным компонентам входного вектора **X**.

Для каждого ФН по аналогии с живым нейроном может быть задан порог возбуждения Q. В этом случае уровень активации нейрона $y(t_i)$ определяется как разность между величиной внутреннего мембранного потенциала $V(t_i)$ и величиной порога Q:

$$y(t_i) = V(t_i) - Q.$$

Если внутренний мембранный потенциал $V(t_i)$ окажется выше величины порога Q, т.е. уровень активации $y(t_i) > 0$, то ФН возбуждается, генерируя выходной сигнал – потенциал действия (спайк). Возникновение спайка в ИН имитируется преобразованием уровня активации $y(t_i)$ в выходной сигнал $z(t_{i+1})$ нейрона с помощью некоторой (в общем случае нелинейной) активационной функции F:

$$z(t) = F[y(t)].$$

ФН – это дискретная модель, поэтому состояние нейрона (его выходной) сигнал) в момент времени t_{i+1} формируется на основе значений входных сигналов в предыдущий момент времени t_i. Тем самым дискретная модель в определенной степени учитывает свойство рефрактерности биологических нейронов, приводящее к тому, что нейрон может изменять свое состояние с конечной частотой.

В ФН в качестве активационной функции F используется пороговая (знаковая) функция:

$$z(t) = F[y(t)] = \text{sign}\,[y(t)] = \begin{cases} 1, \text{ если } [y(t)] > 0; \\ 0, \text{ если } [y(t)] \le 0. \end{cases}$$

График функции $z(t) = \text{sign}[y(t)]$ показан на рис. 3.1.

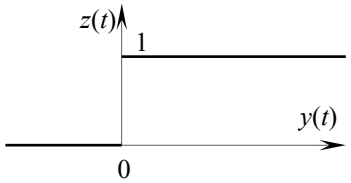


Рис. 3.1. График функции $z(t) = \text{sign}[y(t)]$

Учитывая, что $y(t) = V(t) - Q$ и, соответственно $V(t) = y(t) + Q$, график активационной функции ФН можно представить в виде $z(t) = \text{sign}[V(t) - Q]$, приняв за ось абсцисс величину мембранного потенциала $V(t)$. При этом наглядным становится влияние порога нейрона на выходной сигнал (рис. 3.2).

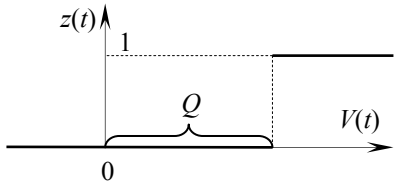


Рис. 3.2. График функции $z(t) = \text{sign}[V(t) - Q]$

Устройство, реализующее знаковую функцию, может быть импульсным или статическим. В первом случае при выполнении условия возбуждения нейрона ($y(t_i) > 0$) на выходе появляется единичный сигнал $z(t_{i+1}) = e(t)$, a priori заданной длительности τ. После окончания этого сигнала ФН переходит в невозбужденное состояние и остается в нем до следующего выполнения

условия возбуждения. Во втором случае при выполнении условия $y(t_i) > 0$ нейрон переходит в единичное состояние и остается в нем до нарушения условия возбуждения.

В итоге, полный алгоритм ФН имеет вид

$$\begin{aligned} &1^0\ V(t_i) = \sum_{j=1}^{N} w_j \cdot x_j(t_i), \\ &2^0\ y(t_i) = V(t_i) - Q, \\ &3^0\ z(t_{i+1}) = \text{sign}[y(t_i)]. \end{aligned} \qquad (3.1)$$

Структурная схема устройства, реализующего ФН на основе алгоритма (3.1), показана на рис. 3.3.

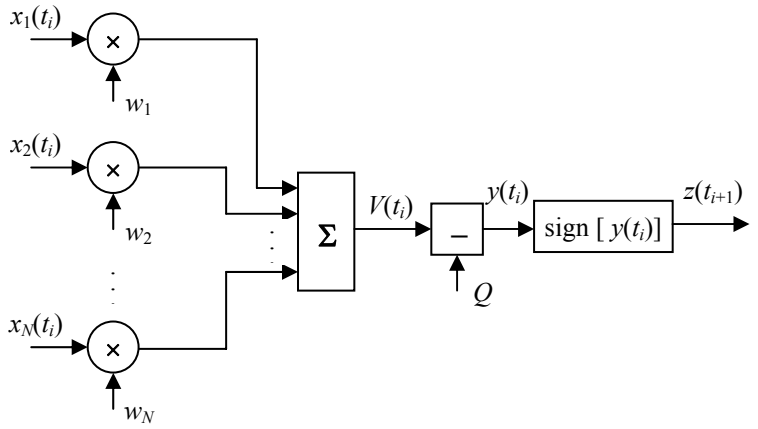


Рис. 3.3. Структурная схема устройства, реализующего ФН

Градуальные нейроны. В отличие от ФН, оперирующего булевыми переменными, в *градуальном нейроне* (ГН) входные и выходные сигналы являются действительными числами, изменяющимися в определенных диапазонах:

$$x_{\min} \le x \le x_{\max},$$
$$z_{\min} \le z \le z_{\max}.$$

В ГН в качестве активационной функции $F[y(t)]$ кроме знаковой могут использоваться и другие линейные и нелинейные зависимости.

В общем случае алгоритм ГН имеет вид:

$$
\begin{aligned}
&1^0\ V(t_i) = \sum_{j=1}^{N} w_j \cdot x_j(t_i), \\
&2^0\ y(t_i) = V(t_i) - Q, \\
&3^0\ z(t_{i+1}) = F[y(t_i)].
\end{aligned}
\qquad (3.2)
$$

Структурная схема устройства, реализующего ГН на основе алгоритма (3.2), показана на рис. 3.4.

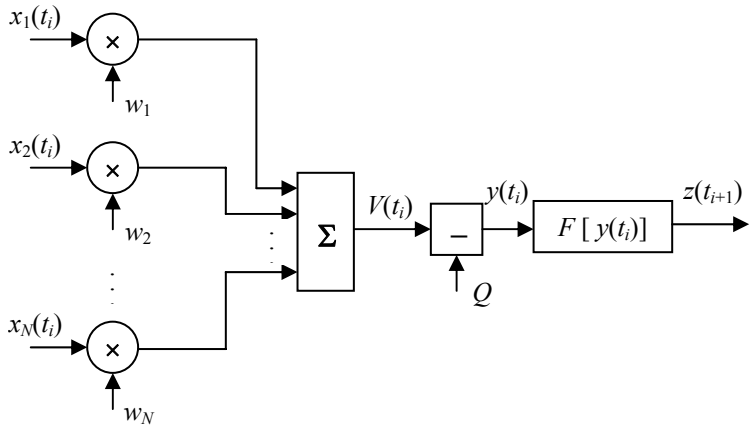


Рис. 3.4. Структурная схема устройства, реализующего ГН

Следует отметить, что в рассмотренных моделях ФН и ГН использованы постоянные значения весов и порога, задаваемые в виде параметров нейрона. Такие модели получили наибольшее распространение на практике. Однако, для некоторых классов задач нейронного моделирования (например, задач управления) может оказаться целесообразным использование переменных значений весов и порога, задаваемых в виде функций времени: $w_j(t_i)$ $Q(t_i)$.

Активационные функции ИН. Кроме пороговой в ИН наиболее часто применяются следующие активационной функции:

1. Кусочно-линейная активационная функция

$$
z(t) = F[y(t)] = \begin{cases} 0, & \text{если } y(t) \le 0, \\ a \cdot y(t), & \text{если } y(t) > 0 \end{cases}
$$

или

$$
\begin{cases} 0, & \text{если } V(t) \le Q, \\ \end{cases}
$$

$$z(t) = F[V(t)] = a[V(t) - Q], \text{ если } V(t) > Q,$$

где a – коэффициент, соответствующий тангенсу угла наклона линейного участка функции.

Графики кусочно-линейной активационной функции показаны на рис. 3.5.

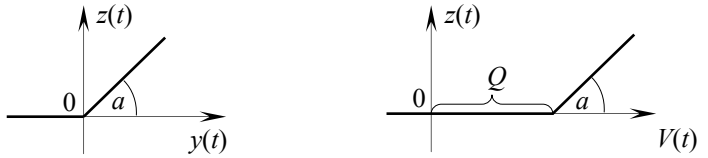


Рис. 3.5. Графики кусочно-линейной активационной функции

2. Кусочно-линейная активационная функция с ограничением (линейный порог, гистерезис)

$$z(t) = F[y(t)] = \begin{cases} 0, & \text{если } y(t) \le 0; \\ a \cdot y(t), & \text{если } 0 < y(t) < z_{\max}/a; \\ z_{\max}, & \text{если } y(t) \ge z_{\max}/a \end{cases}$$

или

$$z(t) = F[V(t)] = \begin{cases} 0, & \text{если } V(t) \le Q; \\ a[V(t) - Q], & \text{если } Q < V(t) < z_{\max}/a; \\ z_{\max}, & \text{если } V(t) \ge z_{max}/a. \end{cases}$$

Графики кусочно-линейной активационной функции с ограничением показаны на рис. 3.6.

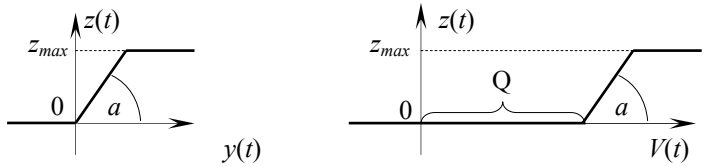


Рис. 3.6. Графики кусочно-линейной активационной функции с ограничением

3. Сигмоидальная активационная функция (нелинейная с насыщением, логистическая, сжимающая, *S*-образная, сигмоид)

$$z(t) = F[y(t)] = \frac{1}{1 + e^{-a \cdot y(t)}}$$

или

$$z(t) = F[V(t)] = \frac{1}{1 + e^{a[Q - v(t)]}}.$$

Коэффициент a характеризует степень крутизны сигмоидальной функции. При уменьшении a, она становится пологой, в пределе, при a=0, вырождаясь в горизонтальную линию на уровне 0,5. При увеличении a, сигмоидальная функция по внешнему виду приближается к пороговой функции с порогом Q в точке $y(t)$=0 (рис. 3.7).

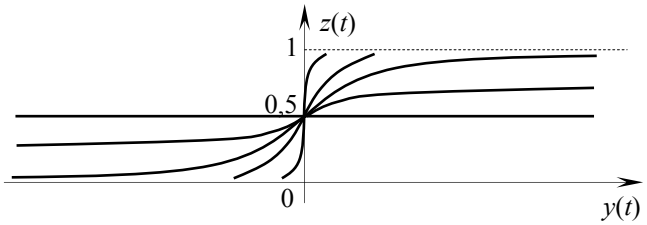


Рис. 3.7. Изменение крутизны логистической функции

Эта нелинейная функция является одной из наиболее распространенной в практике нейронного моделирования и обладает рядом интересных свойств.

Во-первых, сигмоидальная функция изменяется в диапазоне (0,1) и, тем самым, позволяет сжимать произвольный диапазон изменения аргумента $y(t)$ или $V(t)$ в диапазон (0,1) для выходного сигнала $z(t)$, что весьма удобно при моделировании и аппаратно-программной реализации нейронных сетей на ЭВМ (отсюда название – «сжимающая» функция).

Во-вторых, сигмоидальная функция непрерывна и дифференцируема во всем диапазоне изменения аргумента и имеет очень простое выражение для производной:

$$F'[y(t)] = a \cdot F[y(t)]\{1 - F[y(t)]\}.$$

Это свойство позволяет использовать ее в широко распространенных градиентных алгоритмах обучения искусственных нейронных сетей, требующих операций многократного дифференцирования.

В третьих, сигмоидальная функция обладает свойством автоматического регулирования усиления для входных сигналов разного уровня. Центральный участок функции, соответствующий области малых входных сигналов, имеет большой наклон кривой и максимум производной, поэтому коэффициент усиления здесь максимален. По мере удаления от центрального участка функции в область больших по модулю входных сигналов, наклон кривой и ее производная уменьшаются, соответственно уменьшается и коэффициент усиления. Иными словами, один и тот же нейрон или одна и та же сеть нейронов могут эффективно отрабатывать как сильные, так и слабые сигналы.

Графики сигмоидальной функции при a=1 приведены на рис. 3.8.

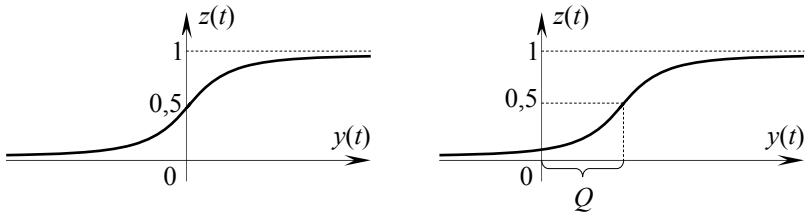


Рис. 3.8. Графики сигмоидальной функции при a=1

График производной сигмоидальной функции имеет колоколообразную форму (рис. 3.9).

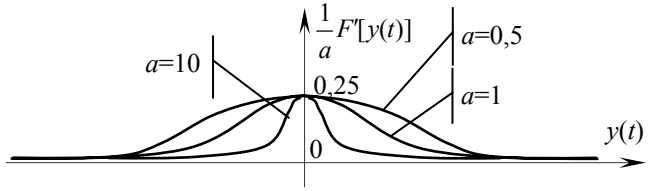


Рис. 3.9. График первой производной сигмоидальной функции

4. Функция гиперболического тангенса (*S*-образная, сигмоид).

$$z(t) = F[y(t)] = \frac{e^{a \cdot y(t)} - e^{-a \cdot y(t)}}{e^{a \cdot y(t)} + e^{-a \cdot y(t)}} = \text{th}[y(t)]$$

или

$$z(t) = F[V(t)] = \frac{e^{a[V(t)-Q]} - e^{a[Q-V(t)]}}{e^{a[V(t)-Q]} + e^{a[Q-V(t)]}} = \text{th}[V(t)]\,.$$

Графики функции гиперболического тангенса при a=1 показаны на рис. 3.10. По форме и свойствам эта функция сходна с предыдущей. Отличие заключается в том, что она симметрична относительно нуля, принимает значения различных знаков и имеет двойной размах по амплитуде, что эффективно используется для ряда нейронных сетей. Коэффициент *a,* как и для предыдущей функции, характеризует степень крутизны функции.

Производная функции гиперболического тангенса равна

$$F'[y(t)] = a \cdot F[y(t)]\{1 - F^2[y(t)]\}.$$

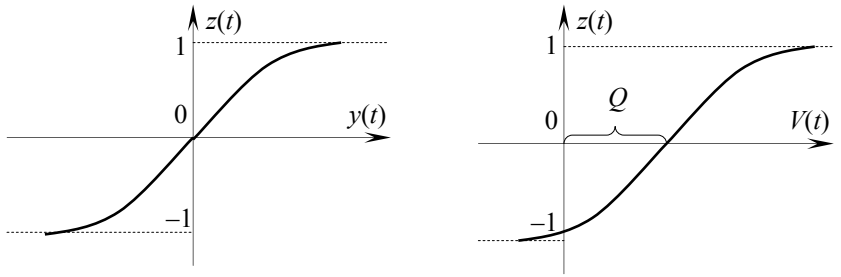


Рис. 3.10. Графики функции гиперболического тангенса

График производной этой функции имеет такой же вид, как и для сигмоидальной функции (см. рис. 3.9).

Другие модели нейронов. Описанные выше модели формального и градуального нейронов являются наиболее простыми и широко применяются на практике. Однако они игнорируют многие свойства своего биологического прототипа, которые некоторые исследователи считают важными или даже решающими: динамические свойства, временные задержки, эффекты синхронизации и частотной модуляции, рефрактерность (временную нечувствительность), аккомодацию (привыкание) и др.

Специалистами по нейронному моделированию предложено большое число моделей нейронов, которые в той или иной степени учитывают различные свойства биологических нейронов. Такие модели, как правило, существенно более сложны и не находят широкого применения для решения практических задач. По-видимому, их ценность в большей степени следует рассматривать в ракурсе первой из названных выше целей нейронного моделирования, а именно – для получения новых знаний о строении и функциях живого мозга.

При решении большинства практических задач немаловажное значение имеют чисто потребительские качества моделей, главное из которых – простота. Поэтому, несмотря на существенные ограничения, принятые в моделях ФН и ГН, искусственные нейронные сети, построенные на основе этих моделей, демонстрируют многие свойства, сильно напоминающие свойства биологических систем, и получили наибольшее распространение.

4. Искусственные нейронные ансамбли

В нервной системе, особенно в ее периферических отделах, существуют устойчивые, генетически предопределенные конфигурации нервных клеток – *нейронные ансамбли* или *ганглии*, функции которых обычно ограничены и предопределены спецификой периферического отдела в организме. В практике нейронного моделирования в ряде случаев также оказывается полезным рассматривать ограниченную совокупность ИН как *искусственный нейронный ансамбль* (ИНА), который имеет жесткую необучаемую структуру, определяемую задачей обработки информации. Понятие ИНА позволяет расширить ограниченный набор вычислительных возможностей одиночного ИН, определяемым его алгоритмом, оставаясь при этом в алгоритмическом базисе используемой модели нейрона.

Переход от одиночного ИН к ИНА можно рассматривать как второй уровень нейронного моделирования. Первоначально этот подход использовался преимущественно для демонстрации возможности реализации операционного базиса вычислительной машины средствами «нейронной логики», в качестве элемента которой выступал ИН [17]. По мнению некоторых исследователей это служило косвенным доказательством сходства принципов функционирования мозга и компьютера.

С точки зрения решения прикладных задач, использование необучаемой «нейронной логики» на основе ИНА вместо традиционной компьютерной логики эквивалентен замене одного функционально полного базиса другим функционально полным базисом. Такая замена не порождает новых уровней функциональности и методов решения задач, и может быть оправдана лишь более эффективной реализацией вычислителя.

В базисе нейронной логики специалистами по нейронному моделированию были предложены решения самых разнообразных задач, которые по эффективности реализации могли конкурировать с вычислителями на обычной логике [17]. Однако стремительный прогресс в развитии компьютерных технологий все более сужает рамки использования «нейронной логики».

В качестве примера, рассмотрим использование ИНА для реализации специальных логических функций: определения модуля, выделения минимума и максимума, установления факта эквивалентности др. [17, 18].

На рис. 4.1. приведена схема ИНА, реализующего функцию получения модуля разности двух сигналов

$$z = z_4 = | x_1 - x_2 |$$

В ИНА (рис. 4.1) в качестве нейронов H_1–H_4 используются ГН с параметрами: $w_{13} = w_{24} = w_{35} = w_{45} = 1$; $w_{14} = w_{23} = -1$; $Q_1 - Q_5 = 0$; $F_1 - F_5 = a \cdot x$ (a=1).

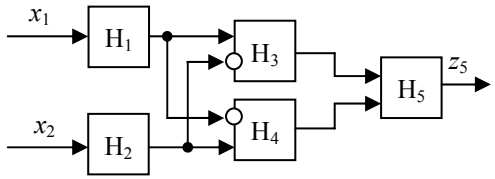


Рис. 4.1. ИНА, реализующий функцию получения модуля разности двух сигналов

На рис. 4.2. приведена схема ИНА, реализующего функцию выбора максимального входного сигнала.

$$z = z_4 = \max(x_1, x_2)$$

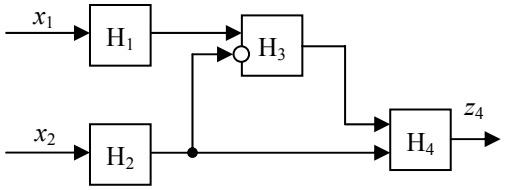
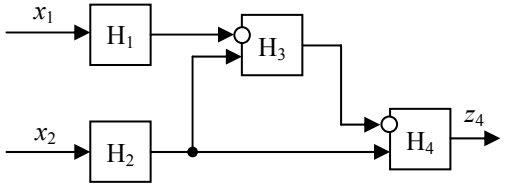


Рис. 4.2. ИНА, реализующий функцию выбора максимума двух сигналов

В ИНА (рис. 4.2) в качестве нейронов H_1–H_4 используются ГН с параметрами: $w_{13} = w_{24} = w_{34} = 1$; $w_{23} = -1$; $Q_1 - Q_4 = 0$; $F_1 - F_4 = a \cdot x$ (a=1).

На рис. 4.3. приведена схема ИНА, реализующего функцию выбора минимального входного сигнала.

$$z = z_4 = \min(x_1, x_2)$$



4.3. ИНА, реализующий функцию выбора минимума двух сигналов

В ИНА (рис. 4.3) в качестве нейронов H_1–H_4 используются ГН с параметрами: $w_{23} = w_{24} = 1$; $w_{13} = w_{34} = -1$; $Q_1 - Q_4 = 0$; $F_1 - F_4 = a \cdot x$ (a=1).

На рис. 4.4. приведена схема ИНА, реализующего функцию эквивалентности (равенства) входных сигналов.

$$z = z_4 = \text{eqv}(x_1, x_2)$$

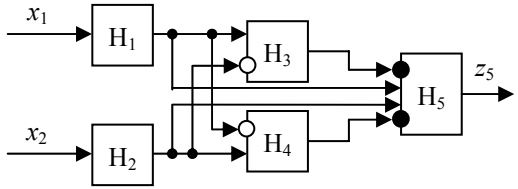


Рис. 4.4. ИНА, реализующий функцию эквивалентности двух сигналов

В ИНА (рис. 4.4) в качестве нейронов H_1–H_5 используются ГН с параметрами: $w_{13} = w_{24} = 1$; $w_{14} = w_{23} = -1$; $w_{15} = w_{25} = 0{,}5$; $w_{35} = w_{45} = -100$; $Q_1 - Q_5 = 0$; $F_1 - F_5 = a \cdot x$ (a=1).

Другим примером применения ИНА является реализация функций классификации и ранжирования (сортировки).

Пусть, например, необходимо отслеживать и классифицировать соотношение между двумя сигналами x_1 и x_2, а точнее – фиксировать факты: $x_1 < x_2$; $x_1 > x_2$; $x_1 = x_2$. В базисе «нейронной логики» такая задача решается с помощью ИНА, показанном на рис. 4.5.

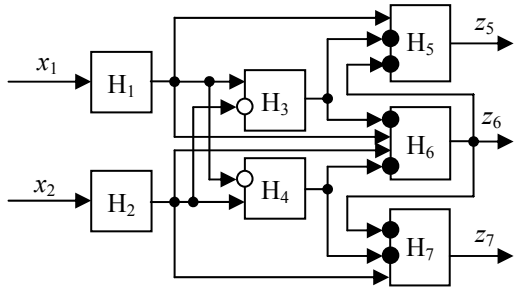


Рис. 4.5. ИНА, реализующий функцию классификации входных сигналов

В ИНА (рис. 4.5) в качестве нейронов H_1–H_7 используются ГН с параметрами: $w_{13} = w_{15} = w_{16} = w_{24} = w_{27} = w_{26} = 1$; $w_{14} = w_{23} = -1$; $w_{35} = w_{36} = w_{47} = w_{46} = w_{65} = w_{67} = -100$; $Q_1 - Q_7 = 0$; $F_1 - F_4 = a \cdot x$; $F_5 - F_7 = \text{sign}\, x$.

Логика работы ИНА (рис. 4.5) описывается выражением

$$\mathbf{Z} = f(x_1, x_2) = \{z_5, z_6, z_7\} = \begin{cases} \{1\ 0\ 0\}, \text{ если } x_1 > x_2; \\ \{0\ 1\ 0\}, \text{ если } x_1 = x_2; \\ \{0\ 0\ 1\}, \text{ если } x_1 < x_2. \end{cases}$$

С помощью ИНА, показанном на рис. 4.6, можно решить задачу ранжирования двух входных сигналов x_1 и x_2 или компонент решетчатой функции $\mathbf{X}=(x_1, x_2)$.

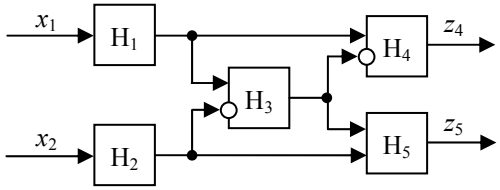


Рис. 4.6. ИНА, реализующий функцию ранжирования входных сигналов

В ИНА (рис. 4.6) в качестве нейронов H_1–H_5 используются ГН с параметрами: $w_{13} = w_{14} = w_{25} = w_{35} = 1$; $w_{34} = w_{23} = -1$; $Q_1 - Q_5 = 0$; $F_1 - F_5 = a \cdot x$.

Логика работы ИНА (рис. 4.6) описывается выражением:

$$\mathbf{Z} = f(x_1, x_2) = \{z_4, z_5\} = \begin{cases} \{x_1, x_2\}, \text{ если } x_1 \le x_2; \\ \{x_2, x_1\}, \text{ если } x_1 \ge x_2. \end{cases}$$

Используя ИНА (рис. 4.6) в качестве структурного элемента, можно решить задачу ранжирования произвольного числа входных сигналов или компонент решетчатой функции $\mathbf{X}=(x_1, x_2, \ldots, x_n)$. В качестве примера на рис. 4.7 показана реализация операции ранжирования (сортировки) трехкомпонентной решетчатой функции $\mathbf{X}=(x_1, x_2, x_3)$ с использованием трех ИНА для двух компонент.

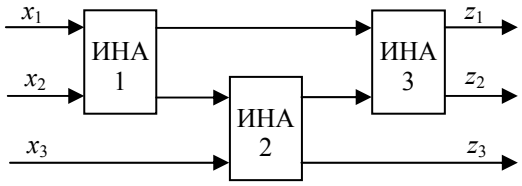


Рис. 4.7. ИНА, реализующий операцию ранжирования трехкомпонентной решетчатой функции

Логика работы ИНА (рис. 4.7) описывается выражением:

$$\mathbf{Z} = f(x_1, x_2, x_3) = \{z_1, z_2, z_3\} = \{x_i^{Ai}, x_j^{Aj}, x_k^{Ak}\}, \quad A_i \le A_j \le A_k,$$

где A_i, A_j, A_k – амплитуды произвольных компонент i, j, k решетчатой функции $\mathbf{Z}= f(x_1, x_2, x_3)$.

5. Искусственные нейронные сети

Выше отмечалось, что одной из главных целей нейронного моделирования является использование принципов построения и функционирования мозга для решения практических задач по обработке информации, трудно поддающихся решению другими средствами. Эта цель реализуется путем создания и использования нейронных конфигураций, которые имитируют некоторые важные свойства, присущие естественному интеллекту, такие как обобщение, обучение, распознавание, принятие решений и др. Объединение ИН в такие конфигурации фактически порождает новый уровень функциональности, отличный от возможностей традиционных компьютеров. Очевидно, что первые два рассмотренных уровня нейронного моделирования – создание ИН и ИНА еще не достигают указанной цели. Их уровень функциональности позволяет выполнять лишь ограниченный набор обычных вычислительных операций и не обнаруживает сколько-нибудь интересных свойств с точки зрения моделирования интеллектуальной деятельности. Новый уровень функциональности нейронных моделей возникает на больших нейронных конфигурациях, в которых воспроизводится свойство обучения. Они называются *искусственными нейронными сетями* (ИНС).

Специалистами по нейронному моделированию предложено множество типов ИНС, отличающихся типом ИН, структурой связей, методами обучения, назначением. Ниже будут рассмотрены основные типы структур ИНС, принципы их обучения, а также ряд конфигураций ИНС, получивших наибольшее распространение для решения определенных классов задач.

Однослойная сеть. ИНС с одним рабочим слоем показана на рис. 5.1 [16].

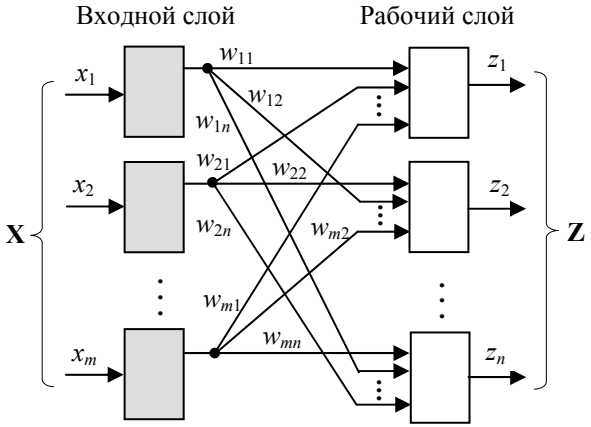


Рис. 5.1. Однослойная ИНС

На входы ИНС поступает множество входных сигналов x_1, x_2, …, x_m, которое удобно интерпретировать как входной вектор $\mathbf{X} = (x_1, x_2, \ldots, x_m)$.

Левый (затемненный) слой нейронов вычислений не производит, а выполняет только вспомогательные функции восприятия и распределения входных сигналов. При подсчете количества слоев ИНС этот слой обычно не учитывается, поэтому при нумерации его удобно считать нулевым слоем.

Синаптические (весовые) коэффициенты всех нейронов (в дальнейшем для краткости будем называть их просто *весами*) правого (рабочего) слоя ИНС можно представить в виде матрицы **W**, имеющей m строк и n столбцов, где m – число входов ИНС, соответствующее размерности вектора **X**, n – число ИН в рабочем слое сети:

$$\mathbf{W} = \begin{Vmatrix} w_{11} & w_{12} & \ldots & w_{1n} \\ w_{21} & w_{22} & \ldots & w_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ w_{m1} & w_{m2} & \ldots & w_{mn} \end{Vmatrix}$$

Множество выходных сигналов z_1, z_2, …, z_n сети по аналогии с входными сигналами также можно интерпретировать как вектор $\mathbf{Z} = (z_1, z_2, \ldots, z_n)$.

Зависимость выходных сигналов ИНС от входных в векторной форме можно представить как произведение вектора **X** на матрицу весов **W**:

$$\mathbf{Z} = \mathbf{X} \cdot \mathbf{W},$$

где **Z** и **X** – векторы-строки.

ИНС (рис. 5.1) имеет полный набор всех возможных соединений. При решении конкретных задач часть из этих соединений может не использоваться (отсутствовать).

Однослойные ИНС наиболее просты в реализации, но они обладают и ограниченными функциональными возможностями.

Многослойная сеть. Более сложные ИНС, как правило, обладают и большими функциональными возможностями. В практике нейронного моделирования получили распространение многослойные ИНС, которые напоминают слоистые структуры определенных отделов мозга. В многослойных ИНС нейроны объединяются в слои. Слой – это совокупность ИН с единым вектором входных сигналов. Многослойная ИНС образуется как каскадное соединение слоев ИН. Внешние входные сигналы поступают на входы нейронов нулевого слоя. Выходы нулевого слоя являются, по существу, входами первого рабочего слоя и сети в целом. Кроме нулевого и выходного рабочего слоя в многослойной ИНС есть один или несколько промежуточных, так называемых *скрытых* слоев. Связи от выходов нейронов некоторого k-слоя к входам нейронов следующего $(k+1)$-слоя называются последовательными.

Следует отметить, что многослойная ИНС обладает большей функциональностью (вычислительной мощностью), чем однослойная только в том случае, если активационные функции нейронов являются нелинейными [16]. В случае использования линейных активационных функций выходной сигнал многослойной сети, содержащей n слоев, образуется следующим образом:

$$\mathbf{Z} = (((\mathbf{X}\cdot\mathbf{W}_1)\cdot\mathbf{W}_2)\cdot\mathbf{W}_3)\ldots\mathbf{W}_n = \mathbf{X}(\mathbf{W}_1\cdot\mathbf{W}_2\cdot\ldots\cdot\mathbf{W}_n).$$

Умножение матриц ассоциативно, поэтому произведение матриц также есть матрица

$$\mathbf{W}_1\cdot\mathbf{W}_2\cdot\ldots\cdot\mathbf{W}_n = \mathbf{W}_\Sigma,$$

а, следовательно

$$\mathbf{Z} = \mathbf{X}\cdot\mathbf{W}_\Sigma. \qquad (5.1)$$

Из выражения (5.1) следует, что любая многослойная ИНС с линейными активационными функциями нейронов эквивалентна однослойной ИНС с весовой матрицей, равной произведению весовых матриц всех слоев.

На рис. 5.2 приведен пример двухслойной ИНС с последовательными связями [16].

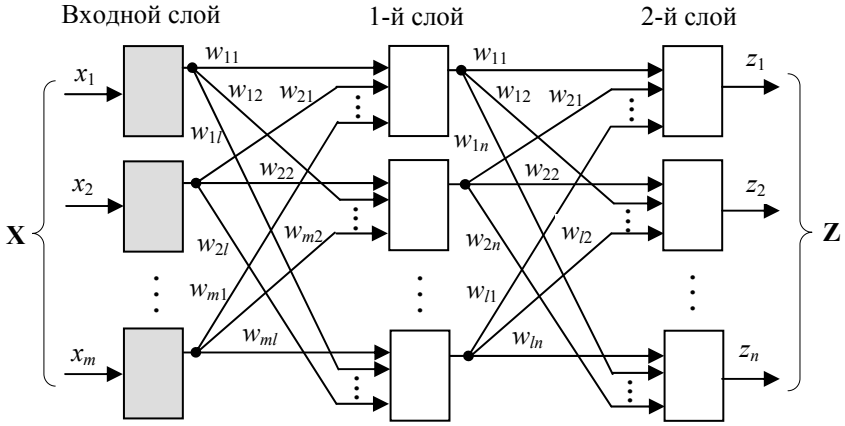


Рис. 5.2. Двухслойная ИНС

Как и в однослойных сетях, при решении конкретных задач часть соединений многослойной сети может не использоваться (отсутствовать).

Разновидностью многослойных ИНС являются сети с прямыми связями. В такой сети связи с нейронов m-слоя могут поступать на входы нейронов

последующих (m+s)-слоев, причем s>1. Фрагмент сети с прямыми связями показан на рис. 5.3.

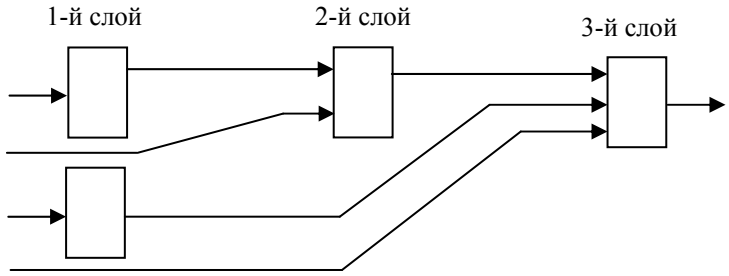


Рис. 5.3. Фрагмент сети с прямыми связями

Рекуррентная сеть. У рассмотренных выше сетей все связи были направлены только с выходов нейронов одного, предшествующего слоя к входам нейронов последующих слоев. Такой класс сетей относится к сетям прямого распространения и широко используется на практике. Сети прямого распространения не обладают внутренней памятью, так как их выходные сигналы полностью определяются текущими значениями входных сигналов и весов. Более общим и сложным типом ИНС являются сети с обратными связями или *рекуррентные* сети (рис. 5.4).

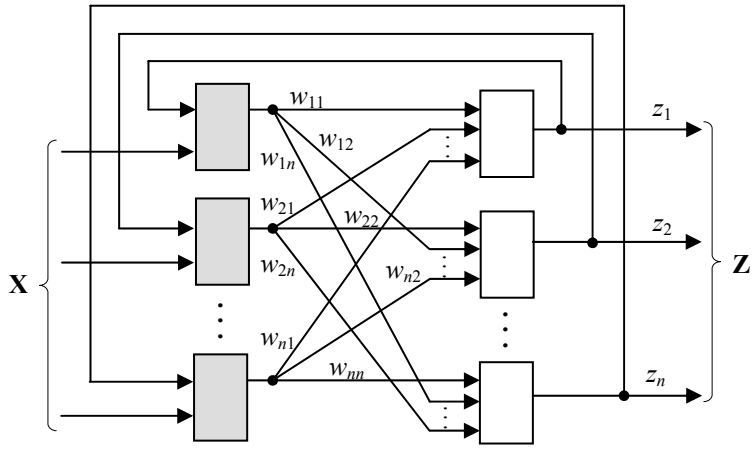


Рис. 5.4. Рекуррентная ИНС

Многослойные и рекуррентные ИНС теоретически можно обобщить в один тип сети, в которой все слои имеют один общий вектор входных сигналов. Для такой сети в качестве частных случаев можно получить ИНС с последовательными, прямыми и обратными связями. Действительно, для слоев с номерами m и $(m+s)$ имеем

- последовательные связи, если $s=1$,
- прямые связи, если $s>1$,
- обратные связи, если $s<0$.

Сеть с латеральными связями. Известны также ИНС, имеющие связи между нейронами одного слоя, которые называются боковыми или *латеральными*. Фрагмент ИНС с латеральными связями показан на рис. 5.5 [17].

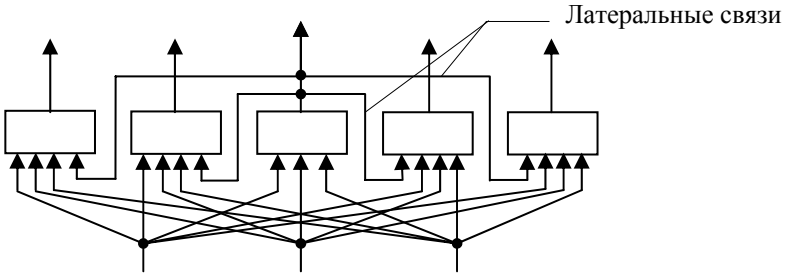


Рис. 5.5. Фрагмент ИНС с латеральными связями

Сеть с локальными связями. Нейроны в сетях с локальными связями располагаются в узлах решетки, чаще – прямоугольной. В этом случае каждый нейрон связан с небольшим числом (обычно 4 или 8) своих топологических соседей, рис. 5.6.

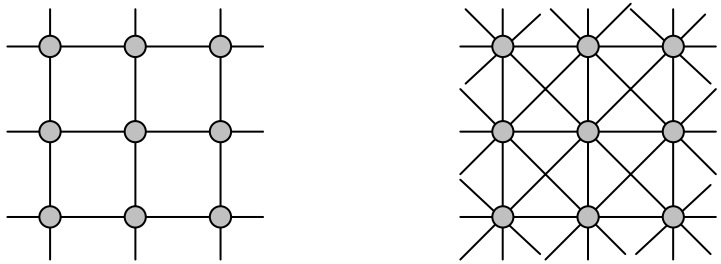


Рис. 5.6. Структура ИНС с локальными связями
на основе прямоугольной решетки

Возможно использование и других типов решетки, например, – шестиугольной (сотовой) или треугольной, рис. 5.7. В этом случае каждый

нейрон будет связан соответственно с тремя или с шестью топологическими соседями.

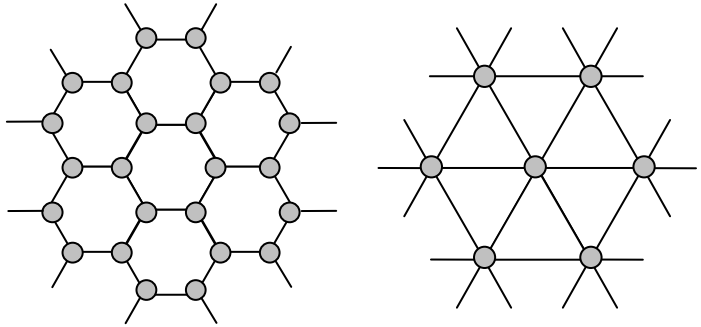


Рис. 5.7. Структура ИНС с локальными связями на основе шестиугольной и треугольной решетки

Все модели ИНС, которые нельзя отнести ни к одной из предыдущих групп называются неструктурированными.

Выбор типа сети. Многолетние исследования в области нейронного моделирования накопили достаточно обширный арсенал ИН и ИНС самых различных типов, ориентированных на решения различных классов задач. Поэтому разработчику в большинстве случаев достаточно выбрать подходящий тип нейронной системы из числа уже имеющихся и приспособить ее для решения своей конкретной задачи.

Для построения ИНС, ориентированной на решение конкретной задачи, используются процедуры (программы) формирования (создания) нейронных сетей. Эти процедуры обеспечивают ввод указанных характеристик моделей ИН и структур ИНС.

Каждая тип ИНС может быть использован для решения лишь некоторого ограниченного класса практических задач, поэтому выбор структуры ИНС осуществляется в соответствии с особенностями и сложностью задачи. Для решения отдельных классов задач уже существуют оптимальные, на сегодняшний день, конфигурации ИНС, которые называют *нейронными парадигмами*. Так, многослойные ИНС с пороговыми и сигмоидальными активационными функциями используются для распознавания образов и классификации. ИНС с локальными связями хорошо подходят для обработки изображений и некоторых других частных задач. Рекуррентные ИНС решают задачи оптимизации и адаптивного управления. Однослойные и многослойные ИНС с кусочно-линейными активационными функциями применяются для воспроизведения различных функциональных зависимостей. Для решения задач линейной алгебры используются многослойные сети с особыми передаточными функциями.

Лишь для небольшого числа ИНС существует строгое математическое обоснование возможностей их применения для решения конкретных практических задач. В наибольшей степени теоретически проработаны однослойные нейронные сети с пороговыми передаточными функциями и двухслойные нейронные сети с сигмоидальными передаточными функциями. Для последних – на основе теоремы Колмогорова-Арнольда доказано, что они могут реализовывать любые отображения входного сигнала в выходной. К построению многопараметрических отображений сводится большинство задач распознавания, управления, идентификации и др.

Если задача не сводится к одному из известных типов, то разработчику приходится решать сложную проблему синтеза новой конфигурации ИНС. При синтезе ИНС руководствуются несколькими основополагающими принципами. Функциональные возможности ИНС возрастают:

- с увеличением числа слоев сети, числа нейронов в слоях и плотности связей между нейронами;
- с увеличением сложности алгоритмов функционирования ИН и ИНС;
- с введением обратных связей, но при этом порождается проблема динамической устойчивости сети.

В целом дать строгие рекомендации по подбору конфигурации ИНС для решения конкретной задачи весьма трудно и оптимальный вариант в большой степени зависит от таланта и интуиции разработчика.

6. Обучение искусственных нейронных сетей

Принципы и методы обучения

Способность к обучению является фундаментальным свойством мозга. Наличие этого свойства в ИНС является наиболее характерной, и привлекательной чертой, сближающей их с мозгом и отличающей от других вычислительных систем.

В ИНС обучение рассматривается как настройка параметров сети для решения поставленной задачи. В качестве таких параметров обычно выступают синаптические коэффициенты (веса связей). Кроме весов связей в параметры сети могут включаться также пороги (смещения). Цель обучения ИНС – достичь желаемой выходной реакции сети на некоторое множество входных сигналов называемое *обучающей выборкой*. Входное и выходное множества сигналов удобно интерпретировать как вектора. Процесс обучения ИНС осуществляется путем последовательного предъявления входных векторов из обучающей выборки с одновременной подстройкой параметров сети в соответствии с некоторой процедурой, называемой *алгоритмом обучения*. Процедура обучения производится до тех пор, пока не будет достигнута желаемая выходная реакция ИНС для всей обучающей выборки.

В математическом смысле обучение ИНС представляет собой итерационную процедуру, направленную на такую подстройку параметров сети, чтобы некоторый функционал качества обращался в оптимум для всей обучающей выборки. В роли такого функционала обычно используется функция ошибки, характеризующая степень близости отображения входного вектора в желаемый выходной. В общем случае функционал качества (функция ошибки) может иметь произвольный вид, поэтому обучение ИНС превращается в задачу многоэкстремальной невыпуклой многомерной оптимизации.

Для формирования процесса обучения необходимо, прежде всего, иметь модель внешней среды, в которой функционирует ИНС, т.е. определить доступную для сети информацию. Эта модель определяет *парадигму обучения*. В рамках определенной парадигмы обучения далее конструируются *правила подстройки параметров*, т.е. конкретный *алгоритм обучения*.

Существуют три парадигмы обучения: *«с учителем», «без учителя» (самообучение)* и смешанная.

Обучение с учителем (supervised learning) предполагает, что ИНС располагает правильными ответами (выходными векторами) на каждый входной образ (входной вектор). При обучении каждому входному вектору обучающей выборки учитель ставит в соответствие целевой вектор, т.е. правильную реакцию сети. Пара входного и целевого вектора называется *обучающей парой*. Параметры сети подстраиваются так, чтобы ответы были максимально близкими к правильным.

Процесс обучения начинается после задания начальных значений весов сети. В общем случае они могут быть произвольными, например, нулевыми. При наличии априорной информации об особенностях процесса обучения, начальные значения весов могут выбираться из каких-либо дополнительных соображений. При предъявлении очередного входного вектора обучающей выборки выходной вектор сравнивается с целевым вектором, и по разности этих векторов алгоритм обучения производит коррекцию весов (а возможно и порогов), с целью минимизировать эту разность (ошибку). Процедура повторяется для всего обучающего множества до тех пор, пока ошибка по всему обучающему множеству не достигнет приемлемо низкого уровня. Функция ошибки численно определяет сходство фактических и целевых выходных векторов сети для всей обучающей выборки. Наиболее распространенной функцией ошибки является среднеквадратичное отклонение:

$$S = \frac{1}{2}\sum_{k=1}^{L}\sum_{i=1}^{N}(z_{\mathrm{A}ik} - z_{\mathrm{T}ik})^2,$$

где $z_{\mathrm{A}ik}$ – фактический выходной сигнал нейрона i;

$z_{\mathrm{T}ik}$ – желаемый (целевой, терминальный) выходной сигнал нейрона i;

$i = \overline{1, N}$ – число нейронов выходного слоя;

$k = \overline{1, L}$ – размер обучающей выборки.

Используются также и другие функции ошибки.

Разновидностью обучения с учителем является критическая оценка учителем правильности выходного сигнала сети без знания самого выходного сигнала.

В зависимости от решаемой задачи в обучающей выборке используются те или иные группы данных с различной размерностью. Входными данными обучающей выборки могут быть, например, таблицы чисел, сигналы, изображения, распределения случайных величин и др. По типу входные и выходные данные могут быть: бинарные (0 и 1), биполярные (-1 и 1), целые и действительные числа из некоторого диапазона. Для решения практических задач часто используются обучающие выборки большого объема и большой размерности. Из-за ограниченного объема оперативной памяти компьютера разместить в ней такие обучающие выборки может оказаться невозможным. В этих случаях обучающая выборка делится на страницы – группы примеров. В каждый отрезок времени лишь одна страница примеров располагается в оперативной памяти компьютера, остальные – во внешней памяти. Страницы последовательно загружаются в оперативную память компьютера. Обучение сети происходит по всей совокупности страниц-примеров, т.е. по всей обучающей выборке.

В настоящее время отсутствует универсальная методика построения обучающих выборок. Они формируются обычно по усмотрению пользователя для каждой конкретной решаемой задачи.

Обучение без учителя (самообучение) (unsupervised learning) по своей природе ближе к биологическому прототипу – мозгу. Самообучение не предполагает наличия правильных ответов ИНС, т.е. целевого вектора. Располагая только информацией из обучающей выборки, алгоритм самообучения «самостоятельно» выявляет внутреннюю структуру входных данных или корреляцию обучающих и выходных данных. Алгоритм самообучения подстраивает веса связей так, чтобы определенные входные сигналы вызывали согласованные с ними выходные сигналы. Другими словами, при предъявлении достаточно близких входных векторов сеть должна выдавать достаточно близкие выходные вектора. Таким образом, процесс самообучения выявляет статистические свойства обучающего множества и группирует сходные входные вектора в классы. Предъявление вектора из данного класса дает определенный выходной вектор, характерный для данного класса.

Характерной чертой процесса самообучения является то, что вид откликов сети на каждый класс входных образов заранее не известен и представляет собой произвольное сочетание возбуждений нейронов выходного слоя, обусловленное случайным распределением начальных значений весов на стадии инициализации и структурой обучающей выборки. Определение топологии классов в картине выходных реакций осуществляется путем тестирования уже обученной сети. Для приведения откликов обученной сети к удобному представлению, сеть обычно дополняют одним выходным слоем, который обучают классическим методом «с учителем». При этом выходные вектора, образованные на стадии самообучения трансформируются в понятную, обусловленную учителем форму.

Детерминированные и стохастические методы обучения. Используя другой принцип классификации, все существующие методы обучения можно разделить на два класса: *детерминированные* и *стохастические*.

Детерминированные методы обучения шаг за шагом осуществляют процедуру коррекции весов сети, основанную на использовании их текущих значений, величин входных сигналов, а также фактических и желаемых выходных сигналов. Преимущество детерминированных методов обучения заключается в высокой скорости обучения. Недостаток – возможность нахождения только локальных минимумов функции ошибки. При попадании процесса обучения в локальный минимум сеть стабилизируется в нем, не имея возможности самостоятельно из него выйти, чтобы достичь глобального минимума.

Стохастические методы обучения выполняют псевдослучайные изменения весов, сохраняя те изменения, которые ведут к улучшениям. Несмотря на то, что в общем случае стохастическое обучение также сводится к многоэкстремальной оптимизации, его преимущество состоит в возможности

выхода из тупиков локальных экстремумов, путем случайного изменения искомых параметров сети (весов связей) в заданном диапазоне. Такую процедуру называют «выбиванием» сети из локального экстремума. Существенный недостаток стохастического обучения состоит в очень низкой скорости, что делает его непригодным для обучения сетей большой размерности.

Правила обучения

Известны четыре основных типа правил обучения (самообучения):

- коррекция по ошибке;
- стохастическое обучение;
- правило Хебба;
- метод соревнований (Кохонена).

Первые два типа правил применяются в алгоритмах обучения с учителем, а вторые два – при самообучении.

Правило коррекции по ошибке. Для каждого входного сигнала обучающей выборки учителем задается правильный (целевой, терминальный, желаемый) выходной сигнал $\mathbf{Z}_T$. После подачи входного сигнала сеть выдает реальный выходной сигнал $\mathbf{Z}_A$, который может совпадать или не совпадать с целевым. Сигнал разности или ошибки ($\mathbf{Z}_T - \mathbf{Z}_A$) используется для модификации весов с целью уменьшения этой ошибки. Известны различные модификации этого правила, на основе которых конструируются градиентные алгоритмы обучения. Все градиентные алгоритмы обучения сетей основаны на вычислении частных производных функций ошибки по параметрам сети (градиентный спуск по поверхности ошибки).

Стохастическое обучение. Целью стохастического обучения является такая подстройка весов связей, при которой состояния нейронов удовлетворяют желаемому распределению вероятностей. Стохастическое обучение может рассматриваться как специальный случай коррекции по ошибке.

К стохастическим алгоритмам обучения относятся: поиск в случайном направлении, имитация отжига (машина Больцмана и машина Коши), методы генетической эволюции.

При поиске в случайном направлении полученный на выходе ИНС вектор $\mathbf{Z}_A$ сравнивается покомпонентно с желаемым выходным вектором сети $\mathbf{Z}_T$. При однократном предъявлении образца (размер обучающей выборки L=1) Сравнение обычно осуществляется по формуле

$$S = \frac{1}{2}\sum_{i=1}^{N}(z_{Ai} - z_{Ti})^2 \ ,$$

где z_{Ai} – фактический выходной сигнал нейрона i;

z_{Ti} – желаемый выходной сигнал нейрона i;

S – целевая функция.

Цель обучения – минимизация целевой функции S. Процедура коррекции весов некоторого нейрона сводится к случайному выбору весового коэффициента этого нейрона и случайному его изменению. Если коррекция помогает (уменьшает S), то она сохраняется, в противном случае происходит возврат к прежнему значению весового коэффициента. Итерации повторяются до тех пор, пока сеть не будет обучена в достаточной степени.

Имитация отжига (машина Больцмана и машина Коши). Отличие этого алгоритма от предыдущего заключается в возможности постепенного уменьшения случайной коррекции весовых коэффициентов, что позволяет выйти сети из тупиков локальных экстремумов и достичь глобального экстремума целевой функции. Этот процесс напоминает отжиг металла, поэтому для его описания и применяется термин «имитация отжига».

Цель обучения, по-прежнему, – минимизация целевой функции S. Процедура коррекции весов некоторого нейрона сводится к случайному выбору весового коэффициента этого нейрона и случайному его изменению. Если коррекция помогает (уменьшает S), то она сохраняется, в противном случае вероятность сохранения изменения веса вычисляется в соответствии с распределением Больцмана

$$P(w) = e^{-\frac{w}{k \cdot T}},$$

где: $P(w)$ – вероятность изменения веса w в целевой функции S;

k – константа, аналогичная константе Больцмана (выбирается в зависимости от задачи);

T – искусственная температура.

Случайное изменение веса и вычисление вероятности $P(w)$ повторяют для каждого веса сети, постепенно уменьшая температуру T, пока не будет достигнуто допустимо низкое значение целевой функции S. Потом предъявляется другой входной вектор из обучающей выборки и процесс обучения повторяется. Обучение последовательно повторяется для всех векторов обучающей выборки, пока целевая функция S не станет допустимой для всех из них.

ИНС, использующую имитацию отжига на основе распределения Больцмана, называют машиной Больцмана.

Чтобы достигнуть сходимости ИНС типа машины Больцмана к глобальному минимуму требуется очень медленная скорость «охлаждения» – уменьшения искусственной температуры T, что приводит соответственно к длительному процессу обучения. Для ускорения обучения вместо

распределения Больцмана часто используют распределение Коши, имеющее более высокую скорость «охлаждения»:

$$P(w) = \frac{T}{T^2 + w^2}.$$

ИНС, использующую имитацию отжига на основе распределения Коши, называют машиной Коши.

Для вычисления вероятности изменения весовых коэффициентов $P(w)$, независимо от вида распределения, используется метод Монте-Карло. В соответствии с этим методом генерируется случайное равномерно распределенное в диапазоне (0,1) число R и сравнивается с величиной $P(w)$. Если $P(w)>R$, то изменение сохраняется, в противном случае величина веса возвращается к предыдущему значению.

Обучение нейронных сетей методами генетического поиска. При использовании генетических алгоритмов параметры ИНС фиксированной архитектуры – синаптические веса (**W**) и смещения (**B**) рассматриваются в виде вектора **H=(W**, **B)**, который трактуется как хромосома. Возможны и другие более сложные способы кодирования информации.

В отличие от большинства других алгоритмов обучения, для работы генетического алгоритма требуется не один набор начальных значений параметров, а несколько наборов, которые составят популяцию хромосом. Популяция затем обрабатывается с помощью генетических операторов. Получившиеся новые особи оцениваются в соответствии с заданной целевой функцией и помещаются в популяцию. В результате на каждой стадии эволюции получаются популяции со все более совершенными индивидами.

Процесс обучения ИНС рассматривается как адаптивный процесс, связанный с минимизацией целевой функции

$$S = \frac{1}{2}\sum_{i=1}^{N}(z_{\mathrm{A}i} - z_{\mathrm{T}i})^2 ,$$

где $z_{\mathrm{A}i}$ – фактический выходной сигнал нейрона i;

$z_{\mathrm{T}i}$ – желаемый выходной сигнал нейрона i.

Структура процесса эволюционной адаптации ИНС показана на (рис. 6.1).

После определения принципов кодирования и декодирования хромосом определяют перечень необходимых генетических операторов и общую структуру процесса генетического поиска. Для разных задач и типов ИНС они могут быть различными. В итоге формируется ГА для подстройки весов и смещений ИНС.

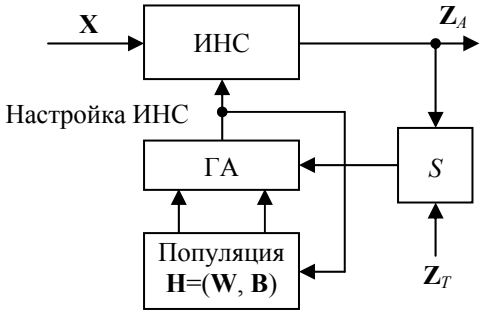


Рис. 6.1. Структура процесса эволюционной адаптации ИНС

В простейшем случае ГА, имитирующий обучение ИНС, работает следующим образом.

1^0. Конструирование начальной популяции.

2^0. Оценка хромосом в популяции и построение целевой функции *S*.

3^0. Выбор пар хромосом из популяции.

4^0. Применение оператора селекции.

5^0. Применение оператора кроссинговера.

6^0. Применение оператора мутации к каждой новой хромосоме с некоторой заданной вероятностью *P*.

7^0. Проверка целевой функции *S*. Если *S* удовлетворительна, то переход к п. 8^0, иначе – переход к п. 3^0.

8^0. Конец работы алгоритма.

Применение механизма случайных мутаций позволяет генетическому алгоритму избежать ловушек локальных минимумов целевой функции *S*.

Правило Хебба является самым старым обучающим правилом, основанным на постулате американского ученого Д. Хебба. В результате нейрофизиологических исследований ассоциативной памяти Хебб в 1949 г. выявил следующую закономерность: если нейроны с обеих сторон синапса активизируются одновременно и регулярно, то сила синаптической связи возрастает. Эта физиологическая закономерность, отражающая феномен обучения через повторение, была сформулирована Хеббом в виде формального правила обучения, которое заключается в усилении весов связей между возбужденными нейронами – источником и приемником:

$$w_{pq}(t_i) = w_{pq}(t_{i-1}) + \eta \cdot z_p^{s-1} z_q^s, \qquad (6.1)$$

где $w_{pq}(t_{i-1})$, $w_{pq}(t_i)$ – вес связи с нейрона *p* слоя *(s-1)* на нейрон *q* слоя *s* до и после коррекции;

η – коэффициент скорости обучения;

z_p^{s-1} – выходной сигнал нейрона *p* слоя *(s-1)*;

z_q^s – выходной сигнал нейрона *q* слоя *s*.

Известно также обратное правило Хебба (antihebbian learning):

$$w_{pq}(t_i) = w_{pq}(t_{i-1}) - \eta \cdot z_p^{s-1} z_q^s .$$

Правило обучения Хебба является исключительно локальным, охватывающим взаимодействие только двух нейронов, находящихся с обеих сторон синапса. По этой причине оно универсально и может применяться для обучения нейрона, как без учителя, так и с учителем. В первом случае используется фактическое значение выходного сигнала нейрона z_q^s. Во втором случае вместо значения z_q^s используется желаемая выходная реакция z_{T}^s.

Особенностью правила (6.1) является то, что в результате его применения веса могут принимать произвольно большие значения, поскольку в каждом цикле обучения происходит суммирование текущего значения веса и его приращения. Один из способов стабилизации процесса обучения по правилу Хебба состоит в учете для уточнения веса последнего значения веса $w_{pq}(t_{i-1})$, уменьшенного с помощью коэффициента забывания γ [15]:

$$w_{pq}(t_i) = (1-\gamma) \cdot w_{pq}(t_{i-1}) + \eta \cdot z_p^{s-1} z_q^s . \qquad (6.2)$$

Значение коэффициента γ в (6.2) выбирается обычно из интервала (0, 1) и чаще всего составляет некоторый процент от коэффициента скорости обучения η. При больших значениях γ нейрон «забывает» значительную часть того, чему он обучился в прошлом. Рекомендуемые значения $\gamma < 0{,}1$, при которых нейрон сохраняет большую часть информации, накопленной в процессе обучения и получает возможность стабилизировать значения весов на определенном уровне.

Правило Хебба может применяться для нейронных сетей различных типов с разнообразными активационными функциями. При использовании линейной активационной функции и правила Хебба стабилизация значений весов нейрона вообще не наступает. Процесс становится расходящимся, и значения весов стремятся к бесконечности даже при использовании правила (6.2). Чтобы стабилизировать процесс для такого случая Е. Ойя предложил модифицировать правило (6.1) следующим образом [15]:

$$w_{pq}(t_i) = w_{pq}(t_{i-1}) + \eta \cdot [z_p^{s-1} - w_{pq}(t_{i-1}) \cdot z_q^s] . \qquad (6.3)$$

Доказано, что применение модифицированного правила (6.3) приводит к сходящемуся процессу, при котором модуль вектора весов нейрона стремится к единице.

Если в сети используется сигмоидальная активационная функция

$$z_q^s = F(y_q^s) = \frac{1}{1+\exp(-a \cdot y_q^s)}$$

и правило обучения Хебба, то метод обучения сети с использованием правила (6.1) принято называть сигнальным методом обучения Хебба. Этот метод предполагает вычисление свертки предыдущих изменений выходных значений z_p^{s-1} для определения изменения весов:

$$y_q^s = \sum_{p=1}^{N} w_{pq} \cdot z_p^{s-1}.$$

Существует также дифференциальный метод обучения Хебба [16], который заключается в усилении весов связей между нейронами, которые увеличили свои выходные сигналы между итерациями t_{i-1} и t_i:

$$w_{pq}(t_i) = w_{pq}(t_{i-1}) + \eta \cdot [z_p^{s-1}(t_i) - z_p^{s-1}(t_{i-1})] \cdot [z_q^s(t_i) - z_q^s(t_{i-1})],$$

где $z_p^{s-1}(t_i)$, $z_p^{s-1}(t_{i-1})$ – выходной сигнал нейрона p слоя $(s-1)$ соответственно на итерациях t_i и t_{i-1};

$z_q^s(t_i)$, $z_q^s(t_{i-1})$ – выходной сигнал нейрона q слоя s соответственно на итерациях t_i и t_{i-1}.

Метод соревнований (Кохонена). В отличие от методов Хебба, в которых в некотором слое s могут возбуждаться одновременно множество нейронов, при обучении методом соревнований нейроны некоторого слоя s соревнуются между собой за право активизации. Это явление известно, как правило «победитель забирает все» (Winner Takes All – WTA) и также имеет нейрофизиологическую аналогию. Обучение этим методом сводится к минимизации разницы между входными сигналами данного нейрона, поступающими с выходов нейронов предыдущего слоя, и весовыми коэффициентами данного нейрона:

$$w_{pq}(t_i) = w_{pq}(t_{i-1}) + \eta \cdot [z_p^{s-1} - w_{pq}(t_{i-1})].$$

Подстройка весов связей по вышеприведенной формуле осуществляется не для всех нейронов слоя, а только для одного «нейрона-победителя», выходной сигнал которого в слое является максимальным. В свою очередь, из формулы обучения следует, что максимальный сигнал будет иметь тот нейрон, веса связей которого максимально близки к входному образу слоя.

Метод соревнований Кохонена впоследствии получил множество различных модификаций, позволяющих для конкретных конфигураций ИНС и конкретных задач существенно улучшить качество самообучения.

7. Перцептроны

Перцептрон Розенблатта. Первая нейросетевая парадигма была предложена американскими учеными – нейрофизиологом Уорреном Мак-Каллоком и математиком Уолтером Питтсом. Это была гомогенная сеть, состоящая из искусственных нейронов в виде бинарных пороговых преобразователей [8]. У. Мак-Каллок и У. Питтс показали, что такая сеть может выполнять любые математические и логические операции, и предположили, что она будет способна обучаться распознаванию образов и обобщению. В современной трактовке в модели У. Мак-Каллока, У. Питтса использовались формальные нейроны, имевшие в качестве активационной функции – пороговую (знаковую) функцию:

$$\mathbf{Z} = F(\mathbf{Y}),$$

где $F = sign\,[y(t_i)] = \begin{cases} 1, \text{ если } [y(t_i)] > Q; \\ 0, \text{ если } [y(t_i)] \le Q, \end{cases}$

Q – порог ИН.

Позднее американский ученый Фрэнк Розенблатт с целью изучения и моделирования физических структур и нейродинамических процессов мозга существенно развил теорию и способы построения модели У. Мак-Каллока и У. Питтса и назвал их *перцептронами* (Perceptrons) [9].

Структура перцептрона Розенблатта показана на рис. 7.1.

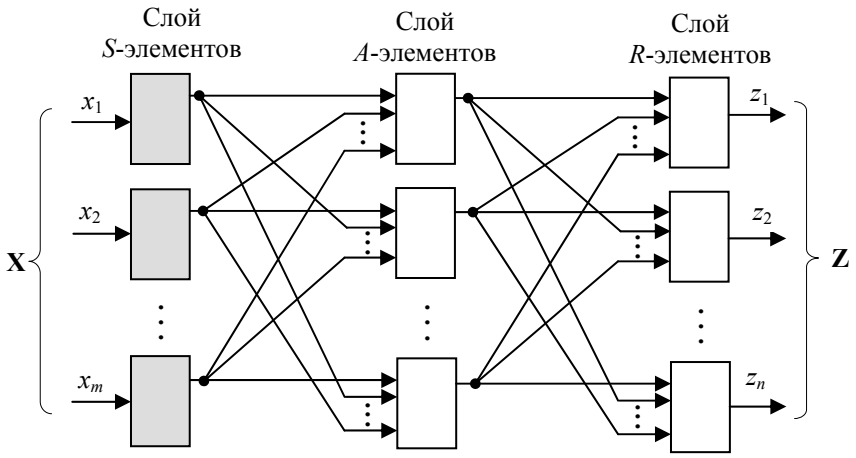


Рис. 7.1. Структура перцептрона Розенблатта

Перцептрон состоит из трех слоев (нумерация слоев соответствует принятой Ф. Розенблаттом) элементов различных типов: слоя *S*-элементов, слоя *A*-элементов и слоя *R*-элементов. *S*-элементы (сенсорные элементы) – это рецепторы, воспринимающие внешние входные сигналы. Рецепторы соединены с входами *A*-элементов с помощью тормозных или возбуждающих связей. Каждый рецептор может находиться в одном из двух состояний – покоя или возбуждения. *A*-элементы (ассоциативные элементы) – это пороговые элементы, которые возбуждаются, если алгебраическая сумма сигналов, приходящих к ним от рецепторов, превышает их порог. При этом сигнал от рецептора, приходящий по возбуждающей связи, считается положительным, а приходящий по тормозной связи – отрицательным. Все связи между *S*-элементами и *A*-элементами, так же как и пороги *A*-элементов выбираются случайным, но фиксированным образом. Сигналы от возбудившихся *A*-элементов с весовыми коэффициентами (весами) передаются в *R*-элементы (реагирующие элементы). Коэффициенты связи между *A*-элементами и *R*-элементами переменные и настраиваются в процессе обучения. Число *R*-элементов определяется количеством классов, на которые необходимо разделить входные образы. Обычно каждому классу соответствует свой *R*-элемент. В простейшем случае, для разделения входных образов на два класса достаточно и одного *R*-элемента, два состояния которого (0,1) кодируют принадлежность образа тому или иному классу.

Обучение перцептрона состоит в таком изменении весовых коэффициентов между *A*- и *R*-элементами, чтобы его реакция на предъявляемые образы была правильной. После обучения перцептрон готов работать в режиме распознавания. На его входы предъявляются «незнакомые» образы, и перцептрон должен установить, к какому классу они принадлежат.

Исследования перцептронов показали, что они способны обучаться распознаванию и обобщению, однако их способности довольно ограничены. Ф. Розенблатт доказал *теорему о сходимости перцептрона*, согласно которой независимо от начальных значений весовых коэффициентов и порядка показа образцов при обучении, перцептрон за конечное число шагов научится различать классы объектов, если только соответствующие значения коэффициентов существуют. Теорема, однако, ничего не говорит о том, какие классы образов могут быть разделены. Кроме того, доказательство конечности числа шагов обучения – малоутешительно, поскольку на практике время обучения может оказаться неприемлемо большим.

Классический перцептрон Ф. Розенблатта в силу случайного характера формирования связей обладает большой избыточностью. Правда, это свойство имеет и положительную сторону, которая заключается в способности удовлетворительного функционирования перцептрона при отказе значительного числа элементов. В дальнейших исследованиях Ф. Розенблатта и его последователей появилось большое число различных модификаций первоначальной схемы перцептрона.

Проблема «ИСКЛЮЧАЮЩЕЕ ИЛИ». Дальнейшие исследования показали, что классический перцептрон с одним слоем настраиваемых весов связей имеет весьма ограниченные функциональные возможности. Имеется много простых задач, которые не могут быть решены таким перцептроном. Одной из них является задача моделирования логической функции от двух переменных ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR)

$$Y = A \oplus B = A \cdot \overline{B} \vee \overline{A} \cdot B$$

Истинность функции Y на входных наборах приведена в табл. 7.1.

Таблица 7.1

Истинность функции ИСКЛЮЧАЮЩЕЕ ИЛИ

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Графически функцию Y можно представить на плоскости в осях координат A-B (рис. 7.2).

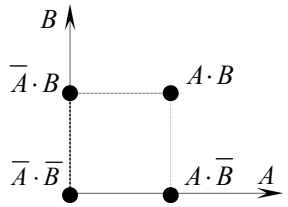


Рис. 7.2. Представление функции ИСКЛЮЧАЮЩЕЕ ИЛИ

Задача классификации при моделировании этой функции сводится к тому, чтобы точки с координатами $A \cdot \overline{B}$ и $\overline{A} \cdot B$ принадлежали к одному классу образов, а точки с координатами $\overline{A} \cdot \overline{B}$ и $A \cdot B$ – к другому классу.

Для моделирования такой задачи на перцептроне в решающем слое потребуется один двухвходовой ФН (рис. 7.3).

Работа схемы будет описываться соотношениями:

$$Y = A \cdot w_1 + B \cdot w_2,$$
$$Z = F_{\text{п}}(Y),$$

где $F_п = \begin{cases} 1, \text{ если } Y > Q; \\ 0, \text{ если } Y \le Q. \end{cases}$

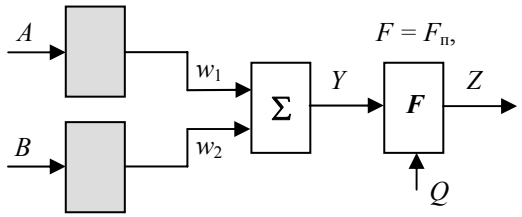


Рис. 7.3. Схема перцептрона для моделирования функции ИСКЛЮЧАЮЩЕЕ ИЛИ

Положим $Y = Q$, тогда перцептрон будет описываться уравнением прямой на плоскости P в координатах A-B [16]:

$$w_1 \cdot A + w_2 \cdot B = Q.$$

Прямая будет разделять плоскость P на две полуплоскости P_1 и P_2. Изменение весов w_1 и w_2 и порога Q будут менять положение этой прямой на плоскости P относительно координат A и B. Любые значения A и B, лежащие на самой прямой, будут давать пороговое значение Q для выхода Y: Y=Q. Значения A и B, лежащие в верхней полуплоскости P_1, будут давать для выхода Y значения больше порога Q: Y>Q. Значения A и B, лежащие в нижней полуплоскости P_2, будут давать для выхода Y значения меньше порога Q: $Y \le Q$ (рис. 7.4).

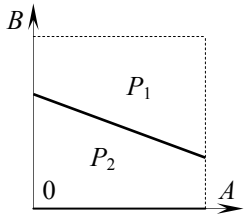


Рис. 7.4. Уравнение прямой $w_1 \cdot A + w_2 \cdot B = Q$

Для решения поставленной задачи классификации прямая должна быть так расположена в плоскости P, чтобы точки с координатами $A \cdot \overline{B}$ и $\overline{A} \cdot B$ находились по одну сторону прямой, а точки с координатами $\overline{A} \cdot \overline{B}$ и $A \cdot B$ – по другую сторону прямой (рис. 7.5). Очевидно, что при любых изменениях параметров w_1, w_2 и Q это невыполнимо. Следовательно, классический

перцептрон в принципе не способен решить поставленную задачу классификации для функции ИСКЛЮЧАЮЩЕЕ ИЛИ.

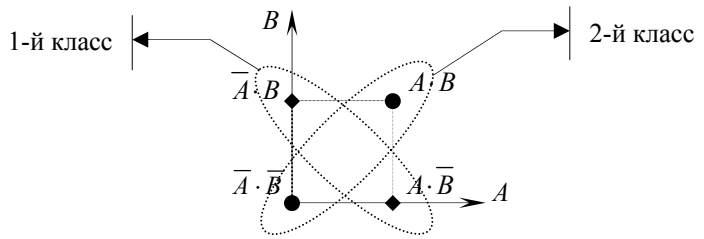


Рис. 7.5. Решение задачи классификации для функции ИСКЛЮЧАЮЩЕЕ ИЛИ

Линейная разделимость при распознавании образов двух классов. Приведенный пример далеко не единственный. Имеется обширный класс функций, не реализуемых перцептроном с одним слоем настраиваемых весов связей. Об этих функциях говорят, что они являются линейно неразделимыми.

Проблему линейной разделимости рассмотрим для наиболее распространенного частного случая перцептрона, предназначенного для распознавания образов только двух классов. Такой элементарный перцептрон (в терминологии Ф. Розенблатта – α-перцептрон), показан на рис. 7.6.

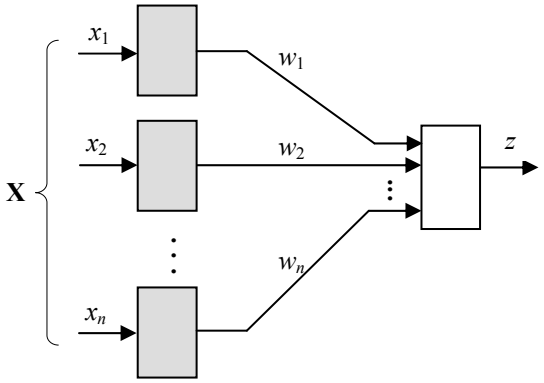


Рис. 7.6. α-перцептрон для распознавания образов двух классов

Рабочий слой сети представлен единственным формальным нейроном, алгоритм работы которого имеет вид

$$1^0.\ V(t_i) = \sum_{j=1}^{n} w_j x_j(t_i),$$

$$2^0.\ y(t_i) = V(t_i) - Q,$$

$$3^0.\ z(t_{i+1}) = \text{sign}[y(t_i)],$$

где sign $[y(t_i)]$ – функция знака

$$\text{sign}\,[y(t_i)] = \begin{cases} 1, \text{ если } [y(t_i)] > 0; \\ 0, \text{ если } [y(t_i)] \le 0. \end{cases}$$

Из алгоритма α-перцептрона следует, что его выходной сигнал формируется следующим образом

$$z = \text{sign}[\sum_{j=1}^{n} w_j x_j - Q]\ . \qquad (7.1)$$

Значение выходного сигнала $z \in (0, 1)$ указывает на принадлежность входного вектора **X** к одному из двух классов.

Пусть R_1 и R_2 – множества точек **X** в n-мерном пространстве E_n, соответствующие объектам (образам) из первого и второго классов. Геометрически задача классификации, очевидно, будет заключаться в построении между этими множествами некоторой n-мерной разделяющей поверхности. Частным случаем такой поверхности является линейная n-мерная поверхность, или n-мерная плоскость, или *гиперплоскость*. Если такая гиперплоскость существует, т.е. имеется функция вида

$$G(\mathbf{X}) = \sum_{i=1}^{n} c_i x_i + c_0, \quad c_i = \text{const}\ (i = 1,2,...,n)$$

такая, что

$$G(\mathbf{X}) > 0, \text{ если } \mathbf{X} \in R_1,$$
$$G(\mathbf{X}) < 0, \text{ если } \mathbf{X} \in R_2,$$

то множества R_1 и R_2 называются линейно разделимыми. Причем разделяющей поверхностью будет функция $G(\mathbf{X})$.

Очевидно, что выражение для разделяющей гиперплоскости можно записать также в виде

$$G(\mathbf{X}) = \text{sign}[\sum_{i=1}^{n} c_i x_i + c_0]. \qquad (7.2)$$

Сравнивая выражение (7.1) для выходного сигнала α-перцептрона и выражение (7.2) для разделяющей гиперплоскости, видим, что математически они идентичны. Это свидетельствует о том, что перцептрон в процессе обучения путем подстройки весовых коэффициентов и порогов в математическом смысле формирует в пространстве входных данных не что иное, как гиперплоскость, линейно разделяющую это пространство на два подпространства. При этом каждому подпространству соответствует множество входных образов одного из двух классов. Из этого следует важный вывод: перцептрон способен обучиться разделять множества входных образов на классы только в том случае, если между этими множествами можно построить разделяющую линейную поверхность (гиперплоскость). Такая задача классификации относится к задачам *линейной сепарации*.

Рис. 7.7. иллюстрирует линейное разделение двух множеств R_1 и R_2 для случая n=2.

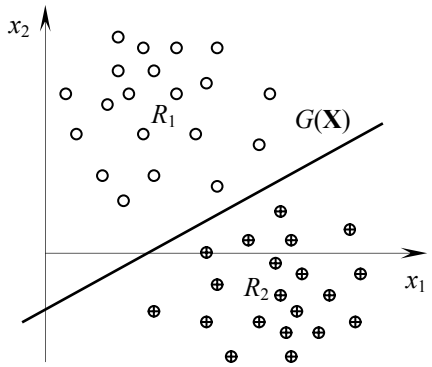


Рис. 7.7. Линейное разделение двух множеств R_1 и R_2 для случая n=2

Таким образом, линейная разделимость для α-перцептрона означает возможность геометрического разделения множества точек n-мерного пространства линейной гиперповерхностью (гиперплоскостью). При n=2 – это линия (см. рис. 7.7), при n=3 – плоскость, при n>3 – гиперплоскость.

Линейная разделимость при распознавании образов *m* классов. При решении на перцептроне более общей задачи – классификации входных образов на m классов (при m>2), он будет иметь три слоя нейронов: входной, рабочий и выходной, рис. 7.8. Входной слой – вспомогательный, выполняет функции приема и распределения входных сигналов. Рабочий – это основной слой с переменными весами нейронов, настраиваемыми в процессе обучения. Рабочий

и выходной слой в общем случае должны содержать по m нейронов – по числу классов входных образов. Каждый нейрон рабочего слоя связан с n нейронами входного слоя, образуя нейронный ансамбль (α-перцептрон). Таким образом, в рабочем слое будет m нейронных ансамблей по (n+1) нейронов в каждом. Нейрон рабочего слоя, имеющий максимальный выходной сигнал определяет класс, к которому принадлежит предъявляемый входной образ.

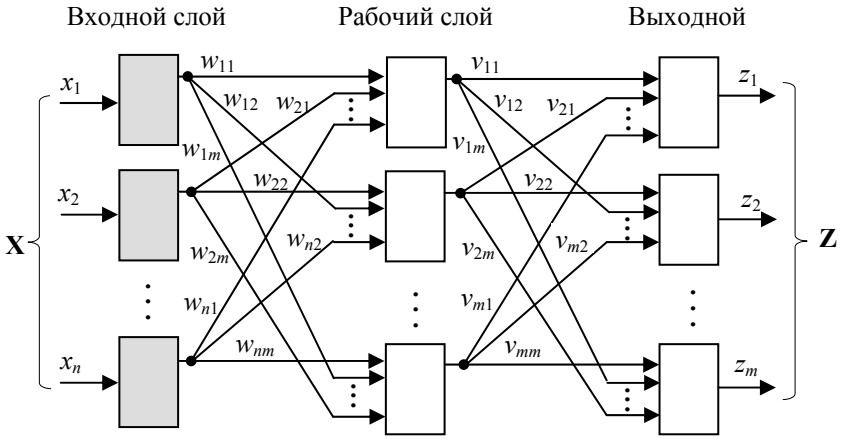


Рис. 7.8. Перцептрон для распознавания образов m классов

Выходной слой выполняет вспомогательную функцию желаемого кодирования различных классов в терминах выходных сигналов. Т.е. каждому нейрону рабочего слоя, а, следовательно, и каждому классу образов, ставится в соответствие определенное значение выходного вектора. Коэффициенты связи между рабочим и выходным слоями являются не обучаемыми и определяются, исходя из способа кодирования выходных сигналов. Эквивалентной реализацией выходного слоя является мажоритарный элемент, выбирающий максимальное значение z_{max}, из выходных сигналов рабочего слоя z_l, l=1, 2, …, m, которое соответствует определенному номеру класса.

В отличие от α-перцептрона, перцептрон, реализующий классификацию на m классов, в процессе обучения в математическом смысле формирует не одну, а множество (m) разделяющих поверхностей в виде гиперплоскостей G_{ij}, сегменты которых и определяют границы между всеми m классами образов. Во многих случаях, некоторые из гиперплоскостей G_{ij} в действительности не используются как разделяющие поверхности. Не используемые гиперплоскости G_{ij} возникают в том случае, если области R_i и R_j не соприкасаются. Такие гиперплоскости называются лишними.

Рис. 7.9. иллюстрирует пример линейного разделения перцептроном четырех множеств R_1; R_2; R_3; R_4 для случая n=2. Разделяющими поверхностями

в этом случае являются отрезки линий G_{ij}. Линия G_{12}, предназначенная для разделения областей R_1 и R_2, фактически не используется и является лишней, поскольку эти области не соприкасаются.

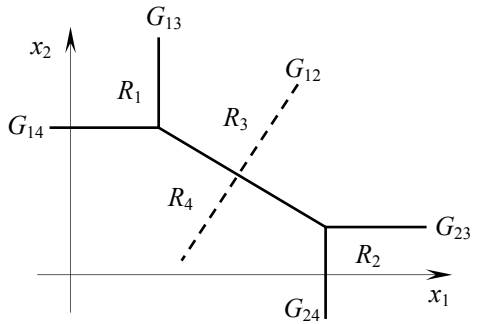


Рис. 7.9. Линейное разделение четырех множеств R_1; R_2; R_3; R_4

Таким образом, для перцептрона, решающего задачу классификации входных образов на m классов (m>2), линейная разделимость означает возможность геометрического разделения множества точек n-мерного пространства сегментами гиперплоскостей. При n=2 – это отрезки линий (см. рис. 7.9), при n=3 – сегменты плоскостей, при n>3 – сегменты гиперплоскостей.

Линейная разделимость и конфигурация областей. Ограничения по представимости задач в перцептронах с одним рабочим слоем можно сформулировать с геометрических позиций в виде требований, которым должны удовлетворять конфигурации областей разделяемых множеств. Однако прежде чем это сделать, дадим два определения областей разделяемых множеств [16].

1. Область называется выпуклой, если для двух любых точек соединяющий их отрезок целиком лежит в этой области.

2. Область называется ограниченной в пространстве C, если она содержится в некотором замкнутом шаре этого пространства.

Примеры выпуклых ограниченных, неограниченных, закрытых и открытых областей для двумерного пространства C (n=2) приведены на рис. 7.10.

Теперь ограничения по представимости задач в перцептронах с одним рабочим слоем иначе можно сформулировать следующим образом. Перцептрон с одним рабочим слоем способен за конечное число шагов решить задачу разделения m произвольных множеств R_i, $i = \overline{1,m}$ точек n-мерного пространства лишь в том случае, если эти множества представлены выпуклыми ограниченными областями. Причем ошибка такого разделения зависит от степени пересечения множеств. Безошибочное разделение возможно только в случае полностью непересекающихся множеств.

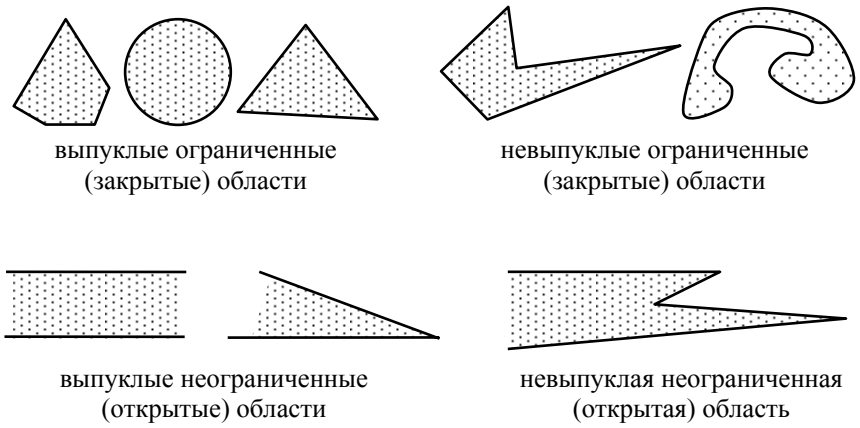


Рис. 7.10. Примеры областей

Для рассмотренной выше функции ИСКЛЮЧАЮЩЕЕ ИЛИ одна область представлена точками $A \cdot \overline{B}$ и $\overline{A} \cdot B$, а другая – точками $\overline{A} \cdot \overline{B}$ и $A \cdot B$ (рис. 7.11). Причем одна из областей всегда будет невыпуклой (на рис. 7.11 невыпуклой является область, образуемая точками $A \cdot \overline{B}$ и $\overline{A} \cdot B$). Следовательно, согласно сформулированным выше ограничениям, задача разделения указанных областей перцептроном с одним рабочим слоем решена быть не может.

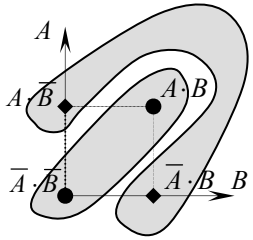


Рис. 7.11. Конфигурация областей в задаче ИСКЛЮЧАЮЩЕЕ ИЛИ

Преодоление ограничений линейной разделимости. Ограничения, накладываемые на перцептронную представимость задач, преодолеваются путем увеличения числа рабочих слоев перцептрона (слоев с настраиваемыми весами).

Рассмотрим перцептрон с двумя рабочими слоями (рис. 7.12).

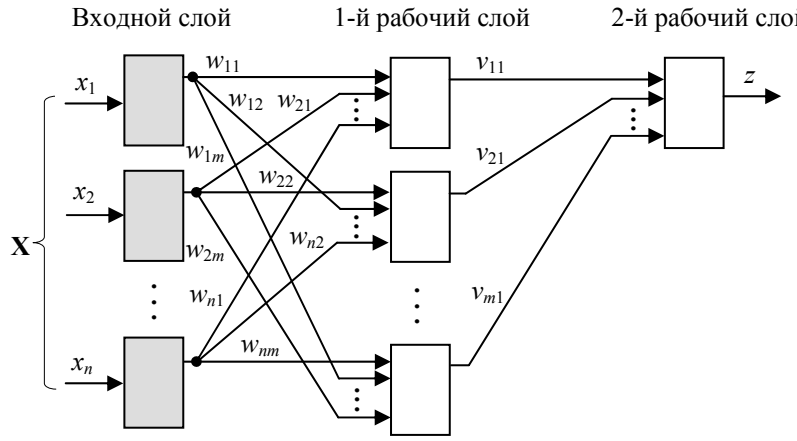


Рис. 7.12. Перцептрон с двумя рабочими слоями

Входной слой содержит n (в общем случае – градуальных) нейронов – по числу компонент входного сигнала **X**. Это вспомогательный слой, который выполняет функции приема и распределения входных сигналов. Первый рабочий слой содержит m формальных нейронов. Каждый нейрон этого слоя связан с n нейронами входного слоя, образуя нейронный ансамбль (α-перцептрон). Всего образуется m нейронных ансамблей по (n+1) нейронов в каждом. Второй рабочий слой содержит единственный формальный нейрон, связанный со всеми m нейронами первого рабочего слоя. Веса связей обоих рабочих слоев настраиваются в процессе обучения.

Пусть веса и порог нейрона второго рабочего слоя выбраны таким образом, чтобы его выходной сигнал обращается в единицу только при всех единичных выходных сигналах нейронов первого рабочего слоя (функция логичекое И).

Перцептрон, содержащий m нейронных ансамблей, в процессе обучения формирует m гиперплоскостей в пространстве входных данных, каждая из которых делит исходное пространство на два подпространства. При этом каждый нейрон первого рабочего слоя обеспечивает единичный выходной сигнал для одного из этих подпространств. В результате, при обучении сети возникает многообразие комбинаций по разделению исходного пространства входных данных на отдельные части (подобласти), ограниченные секущими гиперплоскостями. Нейрон второго рабочего слоя выполняет операцию сборки (логического суммирования) необходимых частей (подобластей) для формирования желаемой конфигурации области, соответствующей определенному классу входных сигналов. Единичный выходной сигнал этого нейрона соответствует попаданию входного сигнала в сформированную область, т.е. – определяет его принадлежность определенному классу.

Перцептрон с двумя рабочими слоями обладает существенно более широкими функциональными возможностями, чем однослойный. Например, он позволяет очень просто решить задачу ИСКЛЮЧАЮЩЕЕ ИЛИ. На рис. 7.13 показана структура сети для решения этой задачи [15].

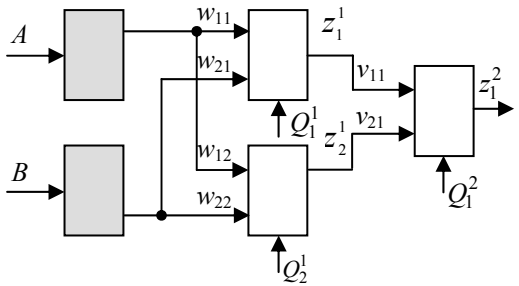


Рис. 7.13 Структура перцептрона для решения задачи ИСКЛЮЧАЮЩЕЕ ИЛИ

Нейроны первого рабочего слоя реализуют операции

$$z_1^1 = \text{sign}[y_1^1] = [w_{11} \cdot A + w_{21} \cdot B - Q_1^1] = \begin{Bmatrix} 1, \text{ если } [y_1^1] > 0 \\ 0, \text{ если } [y_1^1] \le 0 \end{Bmatrix};$$

$$z_2^1 = \text{sign}[y_2^1] = [w_{12} \cdot A + w_{22} \cdot B - Q_2^1] = \begin{Bmatrix} 1, \text{ если } [y_2^1] > 0 \\ 0, \text{ если } [y_2^1] \le 0 \end{Bmatrix}.$$

Подбор весов и порогов должны обеспечить разделение пространства входных данных, показанное на рис. 7.14. Общая часть подмножеств, соответствующая условиям $z_1^1 = 1$, $z_2^1 = 1$ определяет область, отделенную от остального пространства, соответствующего условиям $z_1^1 = 0$, $z_2^1 = 0$. Единственный нейрон второго рабочего слоя, путем соответствующего задания его весов и порога, реализует функцию логического суммирования, выделяющую общую часть подмножеств $z_1^1 = 1$, $z_2^1 = 1$:

$$z_1^2 = z_1^1 \vee z_2^1.$$

Нейрон второго рабочего слоя, путем соответствующего задания его весов и порога, кроме операции логического суммирования может реализовывать также и другие функции линейной комбинации подобластей.

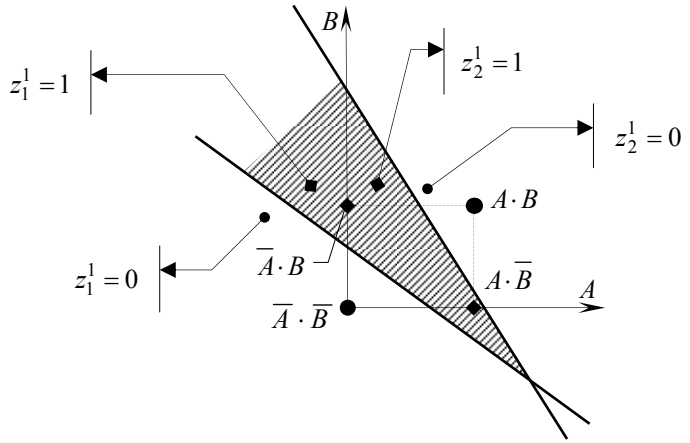


Рис. 7.14 Принцип решения задачи классификации для функции ИСКЛЮЧАЮЩЕЕ ИЛИ

При необходимости решения более общей задачи − классификации входных образов на m классов (при m>2), второй рабочий слой должен содержать достаточное число нейронов для кодирования сигналов всех m классов. В простейшем случае число нейронов этого слоя выбирают равным количеству классов образов m.

В трехслойных перцептронах (с тремя рабочими слоями) ограничения по линейной разделимости полностью снимаются. Их классифицирующие возможности ограничены лишь числом нейронов и весов.

Следует отметить, что перцептронный принцип решения задач классификации полностью применим как для дискретных, так и для непрерывных переменных и областей.

Обучение перцептрона. Перцептрон обучают, подавая множество образов обучающей выборки по одному на его входы и подстраивая веса до тех пор, пока для всех образов не будет достигнут желаемая выходная реакция. Контроль правильности реакции осуществляет «внешний» учитель, поэтому обучение перцептрона относится к классу обучения с учителем.

Для уяснения принципа обучения рассмотрим вначале частный случай, когда число классов образов m=2 (см. рис. 7.6).

Пусть для обучения перцептрона используется обучающее множество Ψ, состоящее из L образов (векторов) обоих классов:

$$\Psi = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L).$$

Множество Ψ содержит два подмножества Ψ_1 и Ψ_2, соответствующих обучающим выборкам образов 1-го и 2-го классов. Будем полагать, что между

подмножествами Ψ_1 и Ψ_2 может быть построена разделяющая гиперплоскость, т.е. Ψ_1 и Ψ_2 линейно разделимы. Цель обучения состоит в том, чтобы выходной сигнал перцептрона обращался в 1 при предъявлении ему образа 1-го класса (вектора из Ψ_1) и обращался в 0 при предъявлении ему образа 2-го класса (вектора из Ψ_2).

Процедура обучения состоит в следующем. На входы перцептрона предъявляется очередной образ **X** из обучающего множества Ψ и определяется выходная реакция, т.е. значение функции z. Если ответ перцептрона правильный (т.е. образ отнесен к нужному классу), то ничего не меняется и предъявляется следующий образ обучающего множества Ψ. Если ответ неправильный (т.е. образ отнесен к другому классу), то веса, связанные с компонентами данного входного образа, ответственные за ошибочный результат, модифицируются, чтобы уменьшить ошибку. Модификация весов может осуществляться в сторону их увеличения или уменьшения, в зависимости от знака отклонения реального выходного сигнала z от правильного (желаемого).

Вербальное описание алгоритма обучения α-перцептрона [16]:

1^0. Подать очередной входной образ $\mathbf{X}_k$ и вычислить z.

2^0. *а*) Если ответ z правильный, то перейти к шагу 1^0;

б) Если ответ z неправильный и равен нулю (образ первого класса неправильно отнесен ко второму классу), то добавить каждую компоненту x_j образа $\mathbf{X}_k$ к соответствующему ей весу w_j;

в) Если ответ z неправильный и равен единице (образ второго класса неправильно отнесен к первому классу), то вычесть каждую компоненту x_j образа $\mathbf{X}_k$ из соответствующего ей веса w_j.

3^0. Если обучающее множество Ψ исчерпано, то конец, иначе перейти к шагу 1^0.

Исходя из сделанного допущения, что обучающие подмножества Ψ_1 и Ψ_2 линейно разделимы, перцептрон за конечное число шагов (что было доказано Ф. Розенблаттом) научится правильно классифицировать все L образов обучающего множества Ψ.

Дельта-правило для α-перцептрона. Современное математическое обобщение алгоритма обучения перцептрона носит название *дельта-правила* или правила Видроу-Хоффа (Widrow-Hoff rule). Оно справедливо для любой формы представления входных и выходных сигналов перцептрона: бинарной, цифровой или непрерывной.

Для определения величины и знака отклонения реального выходного сигнала α-перцептрона z от желаемого правильного (терминального) z_T, определяется их разность

$$\delta = z_T - z,$$

которая является исходной величиной для последующей процедуры коррекции весов. Действительно, случай $\delta=0$ соответствует шагу 2^0*а* алгоритма, случай $\delta>0$ – шагу 2^0*б*, а случай $\delta<0$ – шагу 2^0*в*.

Не меняя сути вербального описания алгоритма, величину коррекции веса можно представить в виде [16]

$$\Delta_j = \eta \cdot \delta \cdot x_j \,, \qquad (7.3)$$

где Δ_j – величина коррекции каждой компоненты w_j весовой матрицы **W**;

x_j – *j*-я компонента входного образа $\mathbf{X}_k$;

η – коэффициент скорости обучения.

Таким образом, величина коррекции Δ_j каждой компоненты w_j весовой матрицы **W** определяется знаком и величиной отклонения δ реального выходного сигнала перцептрона z от желаемого правильного $z_\text{т}$, величиной самой компоненты x_j входного образа $\mathbf{X}_k$ и коэффициентом скорости обучения η, который задает среднюю величину шага изменения весов (темп обучения).

Полученная величина коррекции каждой весовой компоненты позволяет определить скорректированную величину самой весовой компоненты:

$$w_j^\text{н} = w_j^\text{с} + \Delta_j \,,$$

где $w_j^\text{н}$ – новое значение весовой компоненты w_j (после коррекции);

$w_j^\text{с}$ – старое значение весовой компоненты w_j (до коррекции).

В итоге, математическая запись алгоритма обучения α-перцептрона (дельта-правило) будет иметь вид

$$1^0.\ \delta = z_\text{т} - z;$$
$$2^0.\ \Delta_j = \eta \cdot \delta \cdot x_j \,;$$
$$3^0.\ w_j^\text{н} = w_j^\text{с} + \Delta_j \,.$$

Дельта-правило для перцептрона при числе классов больше двух. При решении перцептронной задачи распознавания образов *m* классов (*m*>2), (см. рис. 7.8) используется модифицированное дельта-правило.

Допустим для обучения перцептрона используется обучающее множество Ψ, состоящее из *L* образов (векторов) *m* классов:

$$\Psi = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L).$$

Множество Ψ теперь состоит из *m* подмножеств Ψ_1, Ψ_2, …, Ψ_m, соответствующих обучающим выборкам образов каждого из *m* классов. Будем

полагать, что между подмножествами Ψ_1, Ψ_2, …, Ψ_m могут быть построены разделяющие гиперплоскости, т.е. Ψ_1, Ψ_2, …, Ψ_m линейно разделимы.

Цель обучения состоит в том, чтобы при предъявлении на входы перцептрона образа k-класса – $\mathbf{X}_k$ (вектор из Ψ_k) выходной сигнал перцептрона (вектор $\mathbf{Z}$) соответствовал кодировке этого k-класса Ψ_k. Если вместо выходного слоя перцептрона используется мажоритарный элемент, то он должен при этом выбрать из выходных сигналов рабочего слоя z_l, l=1, 2, …, m максимальное значение z_{max}, которое будет соответствовать k-классу Ψ_k.

Для определенности положим, что кодирование выходного вектора $\mathbf{Z}$ перцептрона заключается в обращении некоторой компоненты z_l этого вектора в 1. Причем порядковый номер l этой единичной компоненты в векторе $\mathbf{Z}$ будет определять номер класса, а, значит, и – соответствовать максимальному значению z_{max} выходных сигналов рабочего слоя. Т.е. обученный перцептрон при предъявлении образа k-класса – $\mathbf{X}_k$ (вектор из Ψ_k) должен выдавать в единственной – k-й позиции вектора $\mathbf{Z}$ единичное значение (z_k=1). Остальные компоненты этого вектора должны быть нулевыми.

Модификация дельта-правила для m классов состоит в том, что при обучении в случае неправильной реакции перцептрона корректируются веса, связанные с компонентами данного входного образа сразу в двух нейронных ансамблях рабочего слоя: в том, который фактически имеет максимальный выходной сигнал на предъявленный образ и в том, который должен иметь такой сигнал. Причем соответствующие веса обоих ансамблей корректируются на одни и те же величины, но с разным знаком. В ансамбле, который фактически имеет максимальный выходной сигнал, они уменьшаются, а в том, который должен иметь такой сигнал, – увеличиваются. Результатом применения этого правила будет уменьшение ошибки.

Пусть при обучении на входы перцептрона поступил образ k-класса – $\mathbf{X}_k$ (вектор из Ψ_k), а в выходном векторе $\mathbf{Z}$ перцептрона отличной от нуля оказалась l-компонента – z_l. Т.е. перцептрон ошибочно отнес образ k-класса к l-классу. При этом, k-классу соответствует выходной сигнал z_k нейрона рабочего слоя, а l-классу– выходной сигнал z_l другого нейрона этого слоя. Тогда разность этих сигналов можно интерпретировать как величину и знак отклонения реального выходного сигнала перцептрона от желаемого правильного:

$$\delta = z_k - z_l,$$

где z_k – выходной сигнал нейрона рабочего слоя, соответствующий правильной классификации;

z_l – выходной сигнал нейрона рабочего слоя, соответствующий фактической классификации.

Коррекция соответствующих весов в обоих ансамблях рабочего слоя осуществляется на одинаковые величины

$$\Delta_j = \eta \cdot \delta \cdot x_j ,$$

но с противоположными знаками:

$$w_{jk}^{н} = w_{jk}^{с} - \Delta_j;$$
$$w_{jl}^{н} = w_{jl}^{с} + \Delta_j,$$

где $w_{jk}^{н}$, $w_{jk}^{с}$ – новое и старое значения j-весовой компоненты w_j в k-нейронном ансамбле, соответствующем правильной классификации;

$w_{jl}^{н}$, $w_{jl}^{с}$ – новое и старое значения j-весовой компоненты w_j в l-нейронном ансамбле, соответствующем фактической классификации.

Очевидно, что в случае правильной реакции перцептрона $\Delta_j = 0$ и коррекция производиться не будет.

Для нашего примера $\delta<0$, следовательно, веса в k-нейронном ансамбле увеличатся на величину Δ_j, а в l-нейронном ансамбле уменьшатся на ту же величину Δ_j.

В итоге, математическая запись алгоритма обучения перцептрона (дельта-правило) для m классов будет иметь вид

$$1^0.\ \delta = z_k - z_l,$$
$$2^0.\ \Delta_j = \eta \cdot \delta \cdot x_j ,$$
$$3^0.\ w_{jk}^{н} = w_{jk}^{с} - \Delta_j,$$
$$4^0.\ w_{jl}^{н} = w_{jl}^{с} + \Delta_j.$$

Исходя из сделанного допущения, что обучающие подмножества Ψ_1, Ψ_2, …, Ψ_m линейно разделимы, перцептрон за конечное число шагов научится правильно классифицировать на m классов все L образов обучающего множества Ψ. Заметим, что при m=2 модифицированное дельта-правило совпадает с описанным ранее для α-перцептрона.

При обучении перцептронов образы из обучающего множества Ψ могут поступать в любом порядке. Важно, чтобы каждый образ предъявлялся несколько раз. Образы можно предъявлять либо циклически, либо в случайном порядке до тех пор, пока несколько раз не повторится предъявление каждого из них. При циклическом предъявлении, каждая совокупность однократных предъявлений всех образов называется *итерацией*.

Начальные значения весовых коэффициентов могут быть любыми. На

практике весам обычно придают небольшие начальные значения.

Описанные выше правила обучения применяются для перцептронов, имеющих один рабочий слой ИН. Для многослойных перцептронов, имеющих два и более рабочих слоев ИН с настраиваемыми весами связей, вопросы обучения становятся существенно более сложными, поскольку в рамках дельта-правила невозможно обучать слои, предшествующие последнему.

8. Нейронные сети обратного распространения ошибки (многослойный перцептрон)

Ограниченные возможности однослойных перцептронов, с одной стороны, и отсутствие теоретически обоснованных алгоритмов обучения многослойных перцептронов, с другой – были одной из причин долгого охлаждения к ИНС. Сети обратного распространения ошибки (ОРО, error backpropagation) по существу открыли новую страницу в теории многослойных ИНС и, главное, получили широкое практическое распространение и коммерческое использование. Название сети обусловлено алгоритмом обучения, в котором ошибка распространяется от выходного слоя к входному, т.е. в направлении, противоположном направлению распространения сигналов при нормальном функционировании сети. По существу в ИНС ОРО была эффективно решена задача обучения многослойных сетей (многослойных перцептронов). По этой причине в литературе по ИНС сеть ОРО относят также к *многослойным перцептронам.* В то же время, для многослойных перцептронов было предложено также множество других алгоритмов обучения, отличных от ОРО, поэтому термин многослойная ИНС (многослойный перцептрон) существенно шире, чем термин ИНС ОРО.

Структура искусственного нейрона. Математическая модель ИН в сети ОРО отличается от других использованием сигмоидальной активационной функцией:

$$1^0.\ V(t_i) = \sum_{j=1}^{N} w_j x_j(t_i),$$

$$2^0.\ y(t_i) = V(t_i) - Q, \qquad (8.1)$$

$$3^0.\ z(t_{i+1}) = F[y(t_i)] = \frac{1}{1+e^{-ay(t_i)}},$$

Использование сигмоидальной активационной функции обусловлено тем, что эта функция непрерывна и дифференцируема во всем диапазоне изменения аргумента и имеет очень простое выражение для производной

$$F'[y(t_i)] = aF[y(t_i)]\{1 - F[y(t_i)]\},$$

которое непосредственно используется в алгоритме обучения сети ОРО.

Структура сети. Сеть ОРО состоит из нескольких слоев, причем каждый нейрон данного слоя связан с каждым нейроном последующего слоя. Т.е. сеть ОРО относится к нейронным сетям с последовательными связями. Как и другие многослойные сети, она имеет входной (нулевой) слой, выполняющий функцию распределения входных сигналов, один или несколько промежуточных

(скрытых) слоев и выходной слой. Конфигурация ИНС ОРО с двумя скрытыми слоями показана на рис. 8.1.

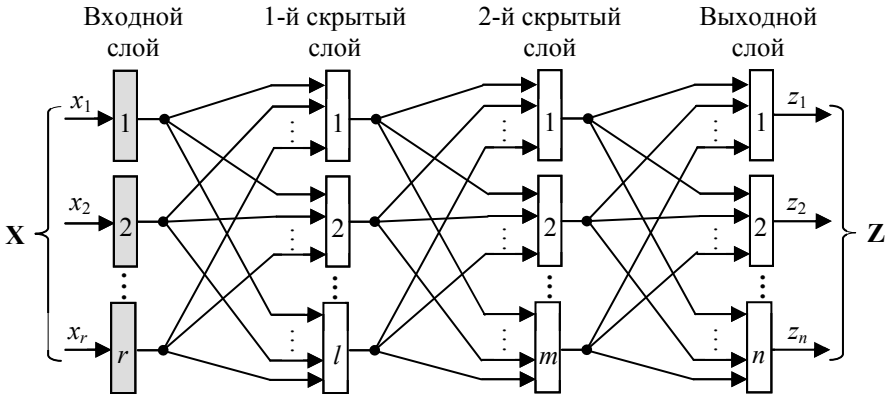


Рис. 8.1. Трехслойная ИНС ОРО

Алгоритм обучения. Проблема обучения многослойных ИНС состоит в том, что оптимальные значения выходных сигналов всех слоев, кроме последнего, как правило, не известны. Поэтому перцептрон с двумя и более слоями невозможно обучить, руководствуясь только величинами ошибок на его выходах. Наиболее эффективный способ решения данной проблемы заключается в распространении сигналов ошибки от выходов ИНС к ее входам, т.е. в направлении, обратном прямому распространению сигналов в обычном режиме работы сети. Этот алгоритм обучения получил название процедуры ОРО (error backpropagation). Он определяет стратегию подбора весов многослойной сети с применением градиентных методов оптимизации. «Изобретенный заново» несколько раз, он в настоящее время считается одним из наиболее эффективных алгоритмов обучения многослойных сетей.

Обучение сети ОРО – это обучение с учителем. Цель обучения состоит в такой подстройке весов, чтобы заданному входному вектору **X** сеть ставила в соответствие целевой выходной вектор $\mathbf{Z}_\text{Т}$. Вместе эти вектора составляют обучающую пару, а группа обучающих пар составляет обучающее множество.

Изменение весов в сети ОРО может производиться после предъявления каждой обучающей пары (режим «on-line»), либо однократно после предъявления всех обучающих пар (всего обучающего множества), составляющих цикл обучения (режим «off-line»).

Общее вербальное описание алгоритма обучения сети в режиме «on-line» формулируется следующим образом:

1^0. Выбрать очередную обучающую пару векторов из обучающего множества и подать входной вектор на входы сети.

2^0. Вычислить выходной вектор сети.

3^0. Вычислить разность между выходным и целевым векторами.

4^0. Подкорректировать веса сети так, чтобы минимизировать ошибку.

5^0. Повторять шаги с 1^0 по 4^0 для каждой пары обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

Вычисления выполняются послойно от входного до выходного слоя. Шаги 1^0 и 2^0 трактуются как «проход вперед». Шаги 3^0 и 4^0 составляют «обратный проход», во время которого вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов.

Число нейронов в каждом слое сети, в общем случае, может быть различным. Для вывода алгоритма обучения обозначим количество слоев сети ОРО через $\sigma = 1, 2, \ldots, s$ и положим, что входной слой содержит r нейронов, выходной s-слой содержит $k = \overline{1,n}$ нейронов, последний скрытый, т.е. (s-1)-слой содержит $j = \overline{1,m}$ нейронов, а предшествующий ему скрытый, т.е. (s-2)-слой содержат $i = \overline{1,l}$ нейронов.

Для обучения ИНС ОРО используем обучающее множество Ψ, состоящее из $p = \overline{1,L}$ обучающих пар:

$$\Psi(\Psi_1, \Psi_2, \ldots, \Psi_L),\ \Psi_p = (\mathbf{X}_p, \mathbf{Z}_p),\ p = \overline{1,L}\ .$$

При проходе вперед выходные вектора **Z** вычисляются последовательно для всех слоев, начиная с первого (σ=1) и кончая последним (σ=s). Выходной вектор каждого слоя вычисляется по формуле:

$$\mathbf{Z}_\sigma = F(\mathbf{X}_\sigma \cdot \mathbf{W}_\sigma).$$

После получения при проходе вперед выходного вектора сети осуществляется обратный проход. При этом ставится задача минимизации целевой функции ошибки, которая находится методом наименьших квадратов.

Для одной обучающей пары целевая функция имеет вид:

$$E(w) = \frac{1}{2}\sum_{k=1}^{n}(z_{k,s} - z_{\mathrm{T},k,s})^2\ ,$$

где $z_{k,s}$, $z_{\mathrm{T},k,s}$ – реальный и желаемый выходные сигналы k-нейрона в s-слое при обработке одной обучающей пары.

Для всего обучающего множества Ψ, содержащего $p = \overline{1,L}$ обучающих пар Ψ_p, целевая функция определяется в виде:

$$E(w) = \frac{1}{2}\sum_{p=1}^{L}\sum_{k=1}^{n}(z_{k,s}^{(p)} - z_{\Gamma,k,s}^{(p)})^2 . \qquad (8.2)$$

Рассмотрим процесс коррекции весов между двумя скрытыми слоями: (*s*-2)-слоем и (*s*-1)-слоем. Минимизация целевой функции ошибки производится методом градиентного спуска. Это означает, что на каждой итерации подстройка весов осуществляется по формуле

$$\Delta w_{ij,(s-1)} = -\eta \cdot \frac{\partial E}{\partial w_{ij,(s-1)}}, \qquad (8.3)$$

где $\Delta w_{ij,(s-1)}$ – приращение веса от *i*-нейрона (*s*-2)-слоя к *j*-нейрону (*s*-1)-слоя;

η – коэффициент скорости обучения, $0<\eta<1$.

Авторами ИНС ОРО было показано, что

$$\frac{\partial E}{\partial w_{ij,(s-1)}} = \frac{\partial E}{\partial z_j} \cdot \frac{dz_j}{dV_j} \cdot \frac{\partial V_j}{\partial w_{ij,(s-1)}}, \qquad (8.4)$$

где V_j – взвешенная сумма входных сигналов *j*-нейрона (*s*-1)-слоя.

Из алгоритма ИН (8.1) следует, что V_j с учетом порога является аргументом *y* активационной функции *F* этого нейрона.

Второй множитель в (8.4) – производная выходного сигнала *j*-нейрона, а следовательно – производная активационной функции *F* по ее аргументу *y*:

$$\frac{dF(y_j)}{dy_j}. \qquad (8.5)$$

Из (8.5) следует, что производная активационной функции *F* должна быть определена на всей оси абсцисс. Именно этим и обусловлено использование в качестве активационной функции *F* – сигмоида.

Третий множитель в (8.4), очевидно, равен входному сигналу *j*-нейрона, а значит – выходному сигналу $z_{i,(s-2)}$ *i*-нейрона предшествующего слоя.

Первый множитель в (8.4) раскладывается следующим образом:

$$\frac{\partial E}{\partial z_j} = \sum_{k=1}^{n} \frac{\partial E}{\partial z_k} \cdot \frac{dz_k}{dV_k} \cdot \frac{\partial V_k}{\partial z_j} = \sum_{k=1}^{n} \frac{\partial E}{\partial z_k} \cdot \frac{dz_k}{dV_k} \cdot w_{jk,s} . \qquad (8.6)$$

Введем новую переменную

$$\delta_{j,(s-1)} = \frac{\partial E}{\partial z_j} \cdot \frac{dz_j}{dV_j} . \qquad (8.7)$$

С учетом введенного обозначения (8.7) выражение (8.6) примет вид

$$\frac{\partial E}{\partial z_j} = \sum_{k=1}^{n} \delta_{k,s} \cdot w_{jk,s} . \qquad (8.8)$$

Теперь, подставляя (8.8) в (8.7), получим рекурсивную формулу для расчета величин $\delta_{j,(s\text{-}1)}$ (s-1)-слоя, исходя из известных величин $\delta_{k,s}$ последующего s-слоя.

$$\delta_{j,(s-1)} = \left[\sum_{k=1}^{n} \delta_{k,s} \cdot w_{jk,s}\right] \cdot \frac{\partial z_j}{\partial V_j} . \qquad (8.9)$$

Для выходного слоя формула (8.9) с учетом (8.2) упрощается и приобретает следующий вид

$$\delta_{k,s} = (z_{k,s} - z_{\mathrm{T},k,s}) \cdot \frac{\partial z_k}{\partial V_k} . \qquad (8.10)$$

Выражение (8.3) для коррекции весов теперь можно представить в раскрытом виде

$$\Delta w_{ij,(s-1)} = -\eta \cdot \delta_{j,(s-1)} \cdot z_{i,(s-2)} . \qquad (8.11)$$

Используя теперь выражение (8.11) для коррекции весов, можно получить скорректированные значения всех весов (s-1)-слоя:

$$w^{\mathrm{н}}_{ij,(s-1)} = w^{\mathrm{с}}_{ij,(s-1)} + \Delta w_{ij,(s-1)} , \qquad (8.12)$$

где $w^{\mathrm{н}}_{ij,(s-1)}$ – новое значение весовой компоненты $w_{ij,(s\text{-}1)}$ (после коррекции);

$w^{\mathrm{с}}_{ij,(s-1)}$ – старое значение весовой компоненты $w_{ij,(s\text{-}1)}$ (до коррекции).

Математическую форму алгоритма обучения ИНС ОРО в режиме «on-line» теперь можно представить в следующем виде:

1^0. Выбрать очередную обучающую пару векторов $\Psi_p = (\mathbf{X}_p, \mathbf{Z}_p)$, $p = \overline{1, L}$ из обучающего множества Ψ. Подать входной вектор $\mathbf{X}_p$ на входы сети.

2^0. Вычислить выходной вектор $\mathbf{Z}_p$ сети.

3^0. Вычислить величину $\delta_{k,s}$ для выходного слоя по формуле

$$\delta_{k,s} = (z_{k,s} - z_{\mathrm{T},k,s}) \cdot \frac{\partial z_k}{\partial V_k}.$$

4^0. Вычислить величину коррекции весов s-слоя по формуле

$$\Delta w_{jk,s} = -\eta \cdot \delta_{k,s} \cdot z_{j,(s-1)}.$$

5^0. Вычислить величину $\delta_{j,(s\text{-}1)}$ (s-1)-слоя по формуле

$$\delta_{j,(s-1)} = \Big[\sum_{k=1}^{n} \delta_{k,s} \cdot w_{jk,s}\Big] \cdot \frac{\partial z_j}{\partial V_j}.$$

6^0. Вычислить величину коррекции весов (s-1)-слоя по формуле

$$\Delta w_{ij,(s-1)} = -\eta \cdot \delta_{j,(s-1)} \cdot z_{i,(s-2)}.$$

7^0. Повторить шаги 5^0, 6^0 для всех остальных скрытых слоев $\sigma = (s\text{-}2), (s\text{-}3), \ldots, 1$.

8^0. Произвести коррекцию весов всех слоев σ по формуле

$$w^{\text{н}}_{ij,\sigma} = w^{\text{с}}_{ij,\sigma} + \Delta w_{ij,\sigma}.$$

9^0. Вычислить ошибку сети по формуле

$$E(w) = \frac{1}{2} \sum_{k=1}^{n} (z_{k,s} - z_{\mathrm{T},k,s})^2$$

Если ошибка $E(w)$ приемлема, то конец, в противном случае – переход на шаг 1^0.

Следует отметить, что метод обратного распространения применим и к сетям с обратными связями, причем обучение происходит достаточно быстро без нарушения критериев устойчивости.

Проблемы. ИНС ОРО в настоящее время являются наиболее распространенными. Однако они также имеют ряд проблем, которые следует учитывать [16].

Проблема 1, которая может возникнуть – это неопределенно долгий процесс обучения. В сложных задачах для этого могут потребоваться часы, а то и дни. Для его сокращения существует ряд методов.

Во многих случаях процесс обучения можно ускорить путем введения в каждый ИН *обучаемого смещения.* Это позволяет сдвигать начало отсчета логистической функции, давая эффект, аналогичный подстройке порога перцептронного нейрона. Реализуется смещение путем добавления в каждый ИН дополнительного взвешенного входа, постоянно подключенного к константе (обычно +1). Вес этого входа обучается, как и остальные.

Коррекция весов по формуле (8.11) для произвольного слоя σ

$$\Delta w_{ij,\sigma} = -\eta \cdot \delta_{j,\sigma} \cdot z_{k,(\sigma-1)}$$

может давать значительные скачки при перемещении по поверхности целевой функции $\Delta w_{ij,\sigma}$, что помимо замедления может привести к неустойчивости процесса обучения. Для придания процессу коррекции весов некоторой инерционности можно запоминать величину предыдущей коррекции и использовать ее с некоторым коэффициентом для сглаживания последующей модификации весов:

$$\Delta w_{ij,\sigma}(t) = -\eta \cdot \delta_{j,\sigma} \cdot z_{k,(\sigma-1)} + \mu[\Delta w_{ij,\sigma}(t-1)],$$

где μ - коэффициент импульса, задаваемый обычно около 0,9.

Этот метод называется *импульсом*. Он позволяет идти процессу коррекции по дну узких оврагов поверхности ошибки (если таковые имеются), а не перемещаться резко от склона к склону. Для одних задач метод работает хорошо, для других – дает слабый или даже отрицательный эффект.

Для коррекции весов вместо формулы (8.11) применяется также метод *экспоненциального сглаживания*. Этот метод сходен с предыдущим и может давать преимущество в некоторых задачах. Коррекция весов для произвольного слоя σ в этом методе осуществляется по формуле

$$\Delta w_{ij,\sigma}(t) = -\eta[\mu \cdot \Delta w_{ij,\sigma}(t-1) + (1-\mu)\delta_{j,\sigma} \cdot z_{k,(\sigma-1)}],$$

где μ - коэффициент сглаживания, варьируемый в диапазоне от 0 до 1. При μ=0 коррекция весов осуществляется по обычной формуле (8.11). При μ=1 новая коррекция игнорируется и повторяется предыдущая. Внутри области (0…1) коррекция сглаживается величиной, пропорциональной μ.

Существуют и другие методы ускорения обучения таких сетей (метод вторых производных, изменение диапазона 0-1 входных значений на ±1/2 и др.). С помощью этих методов удается сократить время обучения сети от 30 до 50%.

Проблема 2. В процессе обучения сети значения весовых коэффициентов могут стать очень большими. Следствием этого будет смещение рабочих точек на сигмоидах в область насыщения, где значения выходной функции нейронов очень велики, а значения производной этой функции – очень малы. Так как посылаемая обратно ошибка и величина коррекции весов пропорциональна этой производной, то процесс обучения может практически замереть. Такое явление получило название *паралича сети*. Избежать этого можно уменьшением коэффициента скорости обучения, но при этом возрастает время обучения.

Проблема 3. Сеть ОРО по существу использует разновидность метода градиентного спуска по поверхности ошибки, непрерывно подстраивая веса в направлении к минимуму ошибки. Поверхность ошибки сложной сети сильно изрезана и состоит из холмов, долин, складок и оврагов в пространстве высокой размерности. Сеть может попасть в локальный минимум (неглубокую долину), в то время как рядом есть более глубокий минимум. В точке локального минимума все направления (так же как и в глобальном минимуме) ведут вверх, и сеть не способна из него самостоятельно выбраться.

В таких случаях обычно выручают стохастические методы. Сеть «выбивают» из ловушки локального минимума путем присвоения параметрам сети случайных значений из заданного диапазона, после чего продолжается обычная процедура градиентного спуска.

Другой метод исключения попаданий в локальные минимумы заключается в следующем. Как только значения весов стабилизируются, коэффициент скорости обучения η кратковременно сильно увеличивают, чтобы начать градиентный спуск с новой точки. Если повторение этой процедуры несколько раз приводит сеть в одно и то же состояние, то с большой долей уверенности можно считать, что найден глобальный минимум, а не локальный. Применяют также комбинации различных методов.

Проблема 4. Важнейшим параметром, влияющим на скорость обучения сети, является размер шага (коэффициент скорости обучения). Дело в том, что доказательство сходимости процесса обучения для сети ОРО основано на производных функции ошибки, т.е. бесконечно малых приращениях весов. Но тогда и скорость обучения будет бесконечно малой, что, конечно, на практике не приемлемо. Если размер шага будет очень велик, то может возникнуть паралич сети или постоянная неустойчивость. Поэтому η выбирают меньшим, чем 1, но и не сильно малым, примерно 0,1. Причем в процессе обучения его желательно постепенно уменьшать. В некоторых работах предложены адаптивные алгоритмы автоматического выбора шага в процессе обучения.

Проблема 5. Процесс обучения сети какому-либо множеству образов желательно проводить так, чтобы сети предъявлялись все векторы обучающего множества, прежде чем выполняется коррекция весов (режим «off-line»). Т.е.

необходимые изменения весов должны вычисляться на всем обучающем множестве, что требует дополнительной памяти. Однако если сеть находится в постоянно изменяющейся внешней среде, так что один и тот же вектор второй раз может уже не повториться, процесс обучения может никогда не сойтись, бесцельно блуждая или сильно осциллируя.

9. Сети встречного распространения

Общие сведения. ИНС встречного распространения (ВР) предложены известным американским специалистом по ИНС Робертом Хехт-Нильсеном в 1987 г.

ИНС ВР относится к так называемым *модульным* сетям, в которых комбинируются различные нейронные парадигмы. В сети ВР Хехт-Нильсен весьма удачно объединил две известные нейронные парадигмы: самоорганизующуюся карту Кохонена и выходную звезду Гроссберга. Получившаяся сеть приобрела новые свойства, которые отсутствовали у родительских сетей.

С методической точки зрения рассмотрение ИНС ВР полезно еще и по той причине, что позволяет – познакомится сразу с тремя типами ИНС: картой Кохонена, выходной звездой Гроссберга и комбинированной сетью на их основе.

Возможности сетей ВР превосходят возможности однослойных сетей, но уступают по общности сетям ОРО. Однако время обучения у них может уменьшаться в 100 раз по сравнению с ИНС ОРО. Кроме того, сеть ВР обладает некоторыми собственными интересными и полезными свойствами.

Сеть ВР способна к обобщению. В процессе обучения входные вектора ассоциируются с соответствующими выходными векторами. Обобщающая способность сети позволяет получать правильный выход даже при приложении входного вектора, который является неполным или искаженным. Это обстоятельство позволяет применять сети ВР для распознавания образов, восстановления образов и усиления сигналов.

Структура сети. Упрощенная версия сети ВР показана на рис. 9.1 [16].

Слой 0 вычислений не выполняет. Слой 1 называется слоем Кохонена. Каждый нейрон слоя 0 соединен с каждым нейроном слоя 1 с весом связи w_{mn}. Совокупность весов w_{mn} может рассматриваться как матрица весов **W**. Слой 2 называется слоем Гроссберга. Каждый нейрон слоя 1 аналогично соединен с каждым нейроном слоя 2 с весом связи v_{np}. Совокупность весов v_{np} также может рассматриваться как матрица весов **V**.

Как и другие сети, сеть ВР функционирует в двух режимах: рабочем режиме (режим распознавания) и режиме обучения.

Слой 1 функционирует на основе процесса самоорганизации и в простейшем случае реализует алгоритм «победитель получает все» (Winner Takes All – WTA). Суть этого алгоритма состоит в том, что для данного входного вектора только один нейрон слоя 1 выдает на выходе логическую единицу («нейрон-победитель»), все остальные нейроны выдают ноль.

Слой 2 обучается с учителем и служит для преобразования полученных в результате самоорганизации выходных сигналов слоя 1 в желаемую, обусловленную учителем форму.

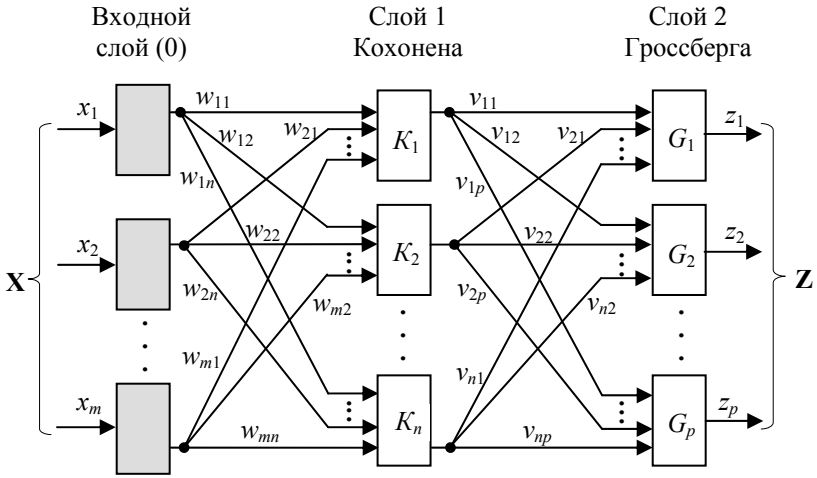


Рис. 9.1. Упрощенная версия сети встречного распространения

Латеральное торможение. В нейронах карты Кохонена использовалась сигмоидальная активационная функция:

$$z_i = F(y_i) = 1/(1 + e^{-y}),$$

а выявление нейрона победителя осуществлялось с помощью механизма *латерального торможения.*

Латеральное торможение имеет глубокую нейробиологическую природу и является общей закономерностью для многих участков нервной системы. Впервые оно было обнаружено Хартлайном в сетчатке глаза мечехвоста, имеющего фасеточное строение. Феномен заключался в том, что при возбуждении одного нейрона, соседние в слое нейроны тормозятся, причем степень торможения монотонно убывает с увеличением нейронного расстояния. Латеральное торможение позволяет зрительной системе обострить световые контрасты, что увеличивает четкость контуров изображений. Позднее латеральное торможение было выявлено в сетчатке позвоночных, в слуховой системе, в системе осязания. Выяснилось также, что оно свойственно не только периферическим отделам нервной системы, но и ее верхним уровням.

По типу латеральных связей модели нейронных слоев можно разделить на два типа: слои с прямыми связями и слои с обратными связями [17]. Структура модели одномерного нейронного слоя с прямыми латеральными связями показана на рис. 9.2.

Слой с прямыми латеральными связями состоит из двух подслоев: возбуждающих нейронов (В) и тормозных нейронов (Т). Тормозные нейроны собственно и реализуют механизм латерального торможения (см. рис. 9.2).

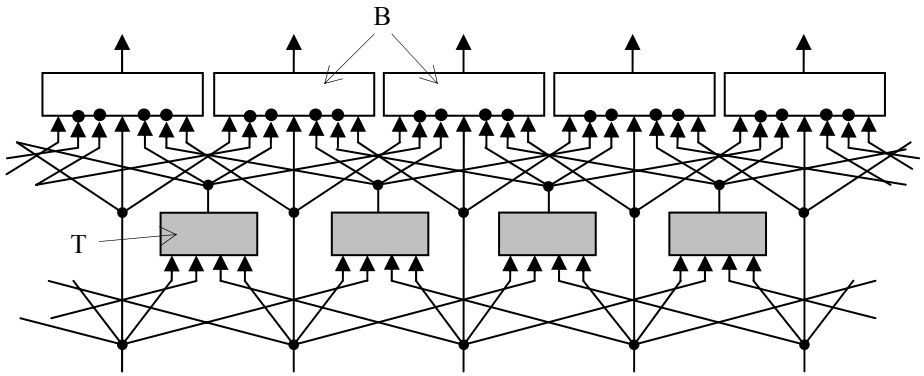


Рис. 9.2. Структура модели одномерного нейронного слоя
с латеральными связями.

Слой нейронов с латеральными связями может осуществлять различные преобразования входных сигналов. Характер преобразования зависит от параметров слоя, вида и ширины фронта входного сигнала, а также радиуса действия возбуждающих и тормозных связей в слое. На рис. 9.3 показаны основные операции, которые могут выполняться слоем с латеральными связями для колоколообразного входного сигнала.

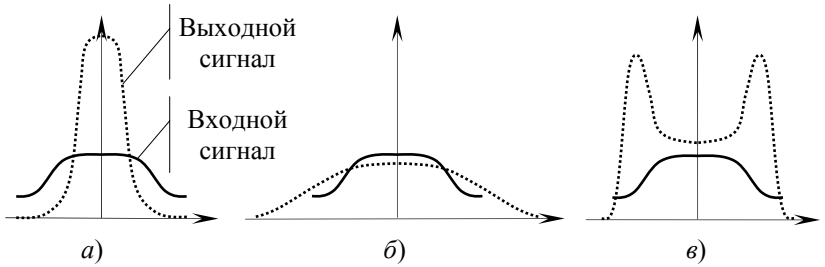


Рис. 9.3. Основные операции, которые могут выполняться
слоем с латеральными связями
а) обострение, *б*) расширение, *в*) выделение краев

Структура модели одномерного нейронного слоя с обратными латеральными связями показана на рис. 9.4. Для простоты на рис 9.4. показаны латеральные тормозные связи только для одного центрального нейрона.

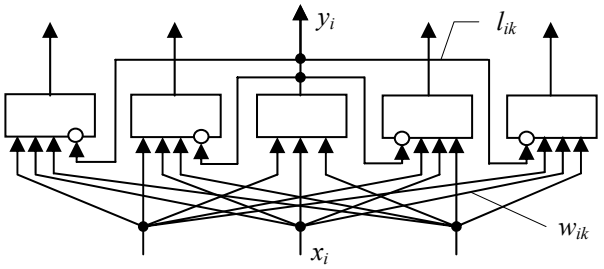


Рис. 9.4. Структура модели одномерного нейронного слоя
с обратными латеральными связями

Процесс самоорганизации Кохонена. В карте Кохонена используется эффект обострения входного сигнала (см. рис. 9.3, *а*), причем радиус возбуждения вокруг некоторого нейрона ограничен одним этим нейроном, остальные нейроны слоя, расположенные слева и справа, равномерно вытормаживаются с максимальным радиусом, ограниченным только размерами слоя.

Эффект наличия латеральных связей с радиусом действия, порядка размеров сети, проявляется в следующем. Если на слой нейронов, содержащий латеральные связи, подать входной вектор, имеющий небольшой максимум, то в процессе релаксации сети осуществляется повышение его контрастности (обострение). В результате вокруг первоначального максимума образуется «пузырек» выходной активности, рис. 9.5.

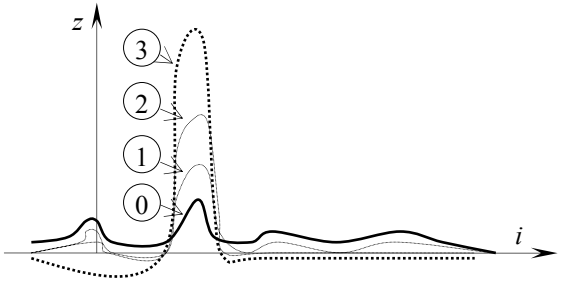


Рис. 9.5. Выходная активность сети с латеральными связями
(Цифры в кружках показывают эволюцию выходного сигнала)

Входные сигналы слоя с латеральными связями полностью определяют процесс самоорганизации этого слоя, что соответствует алгоритму обучения без учителя, т.е. самообучению. Процесс самоорганизации заключается в следующем. Пусть весам связей w_{mn} первоначально приданы некоторые небольшие случайные значения. При подаче на входы слоя некоторого входного вектора

$$\mathbf{X} = (x_1, x_2, \ldots, x_m),$$

он умножается на матрицу весов **W**, образуя вектор уровня активации этого слоя:

$$\mathbf{Y}_K = \mathbf{X} \cdot \mathbf{W}.$$

Покомпонентное применение активационной функции к вектору $\mathbf{Y}_K$ дает выходной вектор слоя

$$\mathbf{Z}_K = F(\mathbf{Y}_K) = F(\mathbf{X} \cdot \mathbf{W}).$$

Очевидно, что за счет случайных начальных значений **W**, выходной сигнал некоторого *i*-нейрона (*i*-компонента z_i вектора **Z**) окажется несколько большим, чем остальные выходные сигналы нейронов этого слоя. Наличие в слое латеральных связей с радиусом действия, равным размеру слоя, приведет к тому, что в процессе релаксации вокруг первоначального максимума на выходе *i*-нейрона образуется «пузырек» выходной активности, и сигнал z_i еще более увеличится, в то время как остальные выходные сигналы этого слоя уменьшатся. По окончании процесса релаксации *i*-нейрон будет иметь уже четко выраженный максимум выходного сигнала z_i и станет «нейроном-победителем».

Рабочий режим сети ВР [16]. Слой 1 в сети ВР, как и в классической карте Кохонена, реализует алгоритм «победитель получает все» (WTA). Отличие заключается только в способе выявления нейрона-победителя.

Если Кохонен в своей карте при выявлении нейрона-победителя ставил как основную – задачу нейробионического подобия, то Хехт-Нильсен преследовал другую цель – получение нейронной парадигмы, способной эффективно решать прикладные задачи. Поэтому использованный Кохоненом способ выявления нейрона-победителя с помощью латерального торможения он упростил, применив обычную вычислительную процедуру. Функция y_{Kj}, соответствующая уровню активности каждого *j*-нейрона в слое 1 сети ВР формируется традиционно:

$$y_{K\,j} = \sum_{i=1}^{m} x_i \cdot w_{ij} \ ,$$

но вместо вычисления выходного сигнала z_{Kj} каждого нейрона с помощью сигмоидальной активационной функции с последующим выявлением «победителя» на основе механизма латерального торможения, применяется простая процедура отыскания максимума среди всех функций y_{Kj} в слое 1:

$$z_{Kj} = F(y_{Kj}) = \begin{cases} 1, \text{ если } \forall i = \overline{1,n}\ \ y_{Kj} > y_{Ki},\ i \neq j; \\ \\ 0, \text{ в противном случае.} \end{cases} \qquad (9.1)$$

Таким образом, для данного входного вектора **X** один и только один нейрон слоя 1 выдает на выходе логическую единицу, все остальные нейроны этого слоя выдают 0.

В результате выходной вектор $\mathbf{Z}_K$ слоя 1 сети ВР

$$\mathbf{Z}_K = (z_{K1},\ z_{K2},\ \ldots,\ z_{Kn}) = \ F(\mathbf{Y}_K) = F(\mathbf{X}\cdot\mathbf{W})$$

формируется на основе покомпонентного применения функции $F(\mathbf{Y}_K)$, вычисляемой согласно (9.1), к вектору $\mathbf{Y}_K$.

Второй слой сети – слой Гроссберга получает выходные сигналы $\mathbf{Z}_K$ слоя. Веса слоя 2 представлены матрицей весов **V**. Каждый j-нейрон слоя 2 получает выходные сигналы $\mathbf{Z}_K$ слоя 1, которые умножаются на вектор-столбец (v_{1j}, v_{2j}, . . . , v_{nj}) матрицы **V**, образуя функцию уровня активации y_{Gj}:

$$y_{G\,j} = \sum_{i=1}^{n} z_i \cdot v_{ij}\ .$$

Вектор слоя 2, имеющий в качестве своих компонент, функции y_{Gj}, $j = \overline{1,p}$, будет иметь вид

$$\mathbf{Y}_G = \mathbf{Z}_K \cdot \mathbf{V}.$$

Функция активации для нейронов в слое 2 в простейшем случае может быть линейной:

$$F(y_{Gj}) = \begin{cases} 0, & \text{если } y_{Gj} \le 0, \\ \\ a \cdot y_{Gj}\ , & \text{если } y_{Gj} > 0. \end{cases}$$

Покомпонентное применение активационной функции к вектору $\mathbf{Y}_K$ дает выходной вектор $\mathbf{Z}_G$ слоя 2

$$\mathbf{Z}_G = (z_{G1},\ z_{G2},\ \ldots,\ z_{Gp}) = \ F(\mathbf{Y}_G) = F(\mathbf{Z}_K \cdot \mathbf{V}).$$

Поскольку только один нейрон слоя 1 имеет выходной сигнал, равный 1, то, при a=1, каждый нейрон слоя 2 будет выдавать величину веса v, который связывает этот нейрон с нейроном-победителем слоя 1.

Предварительная обработка входных векторов. В сети ВР желательно провести предварительную нормализацию входных векторов. Это выполняется путем деления каждой компоненты входного вектора на его длину [16]:

$$x_i^н = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + ... + x_m^2}}.$$

Такая операция превращает входной вектор **X** в вектор единичной длины $\mathbf{X}^н$ в m-мерном пространстве. Операция нормализации для двумерного вектора **X** поясняется рис. 9.6.

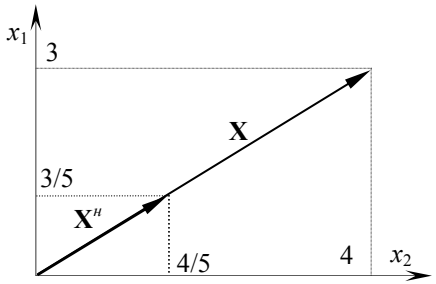


Рис. 9.6. Формирование единичного входного вектора

Для сети с двумя входами двухкомпонентные векторы **X** будут оканчиваться на единичной окружности (окружности единичного радиуса), рис. 9.7. Для сети с тремя входами трехкомпонентные векторы **X** будут оканчиваться на поверхности единичной сферы. В общем случае, для сети с m входами m-компонентные входные векторы **X** будут оканчиваться на поверхности единичной m-мерной гиперсферы.

Обучение слоя 1. Отличием карты Кохонена от других нейронных парадигм является то, что в ней используется обучение без учителя, т.е. самообучение. При самообучении сеть не получает инструкций учителя о правильности ее ответов на предъявляемые входные образы. Она самостоятельно решает задачу сопоставления входных образов между собой и разделяет их в группы схожих, называемые *кластерами*. Все образы одного кластера должны иметь что-то общее и классифицироваться как подобные (принадлежащие одному классу).

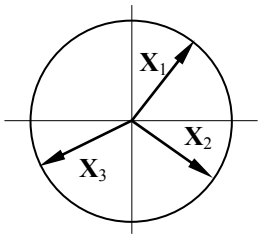


Рис. 9.7. Двумерные входные векторы на единичной окружности

Допустим, для обучения сети ВР используется обучающее множество Ψ, состоящее из L обучающих входных векторов:

$$\Psi = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L).$$

Каждый входной вектор **X** размерности m из множества Ψ можно рассматривать как точку с координатами (x_1, x_2, …, x_m) в m-мерном пространстве. На рис. 9.8. показан пример размещения на двумерной плоскости входных образов, которые естественным образом формируют три кластера.

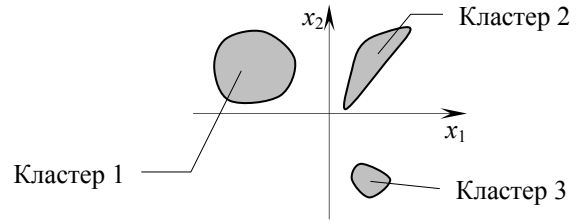


Рис. 9.8. Входные образы, формирующие три кластера

В качестве меры близости (подобия) двух точек p и q обычно используется квадрат евклидова расстояния между ними, вычисляемый по формуле

$$d_{pq} = \sum_{i=1}^{m} (x_{pi} - x_{qi})^2 \ .$$

Для j-кластера можно найти некоторый вектор $\mathbf{P}_j$, определяемый как центр данного кластера (точка с усредненными координатами всех точек в кластере). Векторы $\mathbf{P}_j$ могут рассматриваться как прототипы кластеров, представляющие

их ключевые признаки. Тогда решение о принадлежности произвольного вектора **X** к какому-либо кластеру определится значением

$$\text{index}(\mathbf{X}) = \min d(\mathbf{P}_j, \mathbf{X}) \text{ для всех } j.$$

В процессе обучения сети ВР нейроны слоя 1 можно рассматривать как претендентов на роль победителя в конкурсе за право захвата предъявляемых входных образов. Победителем становится тот нейрон, до которого евклидово расстояние от предъявленного образа наименьшее. При этом осуществляется корректировка весов нейрона-победителя с тем, чтобы еще более сократить это расстояние. В результате обучения близкие входные вектора (принадлежащие одному кластеру) будут активировать один и тот же нейрон слоя 1. В конечном итоге все L образов обучающей выборки Ψ будут классифицированы в группы схожих, т.е. распределены по кластерам.

Для нормированных входных векторов мерой близости вместо евклидова расстояния может служить угол между ними. Тогда решение о принадлежности произвольного вектора **X** к какому-либо кластеру будет определяться максимальным значением скалярного произведения векторов $\mathbf{P}_j$ и **X**:

$$\text{index}(\mathbf{X}) = \max(\mathbf{P}_j, \mathbf{X}) \text{ для всех } j. \qquad (9.2)$$

Вектор $\mathbf{P}_j$ определяет центр j-кластера и эквивалентен вектору-столбцу $\mathbf{W}_j = (w_{1j}, w_{2j}, \ldots, w_{mj})$ матрицы **W** для обученного слоя 1. Поэтому, формулу (9.2) можно представить в виде

$$\text{index}(\mathbf{X}) = \max(\mathbf{W}_j, \mathbf{X}) \text{ для всех } j. \qquad (9.3)$$

Процедура обучения слоя 1 с использованием формулы (9.3) будет следующей. На входы слоя подается входной вектор **X** из обучающей выборки и вычисляются его скалярные произведения с векторами весов, всех нейронов слоя 1. Нейрон с максимальным значением скалярного произведения объявляется победителем, и его веса подстраиваются. Так как скалярное произведение является мерой сходства между входным вектором и вектором весов, то процесс обучения состоит в выборе нейрона слоя 1 с весовым вектором, наиболее близким к входному вектору, и дальнейшем его приближении к входному. При этом каждый вес выигравшего нейрона слоя 1, изменяется пропорционально разности между его величиной и величиной сигнала на входе, с которым он связан. Направление изменения минимизирует разность между весом и величиной сигнала на входе, с которым он связан.

Уравнение, описывающее самообучение слоя 1, имеет вид

$$w_i^{\text{н}} = w_i^{\text{с}} + \eta \cdot (x_i - w_i^{\text{с}}), \qquad (9.4)$$

где $w_i^{\text{н}}$, $w_i^{\text{с}}$ – соответственно новое и старое (предыдущее) значения веса w_i, соединяющего входную компоненту x_i с выигравшим нейроном;

η – коэффициент скорости обучения, который может варьироваться в процессе обучения.

На рис. 9.9 геометрически иллюстрируется двумерный случай коррекции веса. Из рисунка видно, что эффект обучения состоит во вращении весового вектора в направлении входного вектора без существенного изменения его длины.

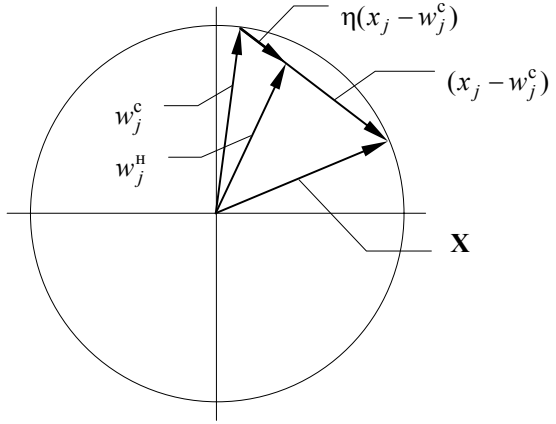


Рис. 9.9. Вращение весового вектора в процессе обучения

Коэффициент η обычно является переменной величиной и в начале обучения принимается равным 0,7, а затем плавно уменьшается в процессе обучения. Это позволяет быстро и грубо подойти к искомой величине веса и затем, более плавно его уточнять.

Если бы с одним нейроном слоя 1 ассоциировался только один входной вектор (т.е. соответствующий кластер содержал бы только один образ), то слой 1 мог бы быть обучен одним вычислением на вес. Действительно, принимая в (9.4) η=1, получим $w_j^{\text{н}} = x_j$. Однако обучающая выборка содержит, как правило, много сходных между собой векторов, сгруппированных в соответствующие кластеры, и сеть должна обучиться активировать один и тот же нейрон слоя 1 для каждого входного вектора из одного кластера. Поэтому веса каждого нейрона слоя 1 должны быть получены усреднением всех сходных входных векторов одного кластера, активирующих данный нейрон. Постепенное уменьшение η позволяет снижать воздействие каждого последующего

обучающего шага, так что окончательное значение будет средней величиной от всех входных векторов, на которых происходит обучение. Геометрически это некоторый центр соответствующего кластера. Таким образом, в результате самообучения слой 1 формирует кластеры в пространстве входных образов.

После обучения слоя 1 решение задачи в рабочем режиме сводится, по существу, к отнесению каждого предъявляемого входного образа к одному из сформированных кластеров, т.е. определению его принадлежности определенному классу.

Выбор начальных значений весовых векторов [16]. Перед началом обучения всем весам сети следует придать некоторые начальные значения. Общепринятой практикой является присваивание весам небольших случайных значений, равномерно распределенных в некотором ограниченном диапазоне, например (0-1). Нормализация начальных значений весов сокращает процесс обучения, так как приближает их значения к нормализованным входным векторам. Что же касается случайного равномерного распределения начальных значений весов (их рандомизация), то это может серьезно помешать процессу самообучения слоя 1. Действительно, равномерное распределение весовых векторов по поверхности гиперсферы в сочетании с неравномерным распределением входных векторов приведет к тому, что лишь небольшая часть весовых векторов будет использоваться для подстройки под входные вектора. Т.е. большинство нейронов слоя 1 не будут задействовано, а используемых нейронов может не хватить для разделения близко расположенных входных векторов на классы.

Здесь могут быть две неприятные крайности.

1. Кластеры входных векторов сконцентрированы на малой части гиперсферы и, тем не менее, их необходимо правильно классифицировать. Если начальная плотность весовых векторов в области кластеров слишком мала, то их может не хватить для правильной классификации всех кластеров. Ситуацию иллюстрирует рис. 9.10, *а,* где для простоты каждый входной вектор **X** представляет собой самостоятельный кластер и должен быть отнесен к отдельному классу.

2. В области какого-либо кластера входных векторов начальная плотность весовых векторов оказывается слишком высокой. Тогда образы этого кластера могут быть неправильно разнесены в разные классы. В принципе, в последующем слое 2 можно исправить эту ошибку и сгруппировать несколько полученных подклассов образов в один класс, но нейроны слоя 1 будут использованы в этом случае не эффективно. Более того, для классификации других кластеров оставшихся нейронов слоя 1 может не хватить. Ситуацию иллюстрирует рис. 9.10, *б*, где два кластера содержат группы из 4-х и 3-х входных векторов, и все векторы одной группы должны быть отнесены к одному классу.

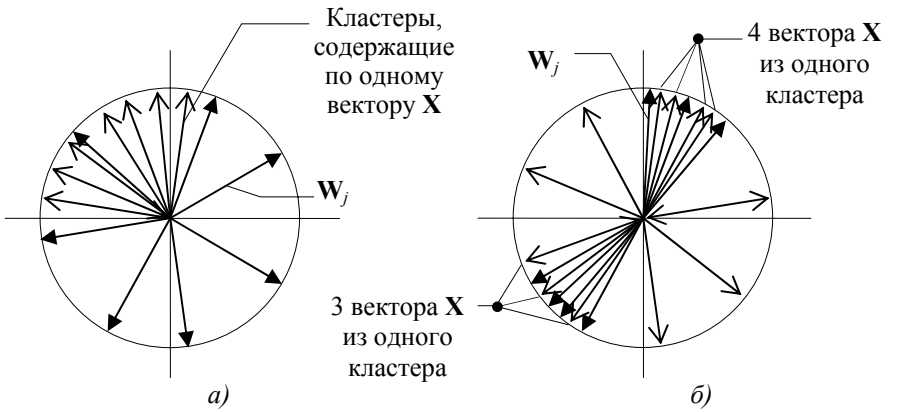


Рис. 9.10. Примеры несогласованной плотности распределения входных и весовых векторов

Очевидно, что оптимальное решение будет состоять в распределении начальных значений весовых векторов, соответствующем плотности распределения входных векторов. На практике это не выполнимо, но существует ряд методов приближения к достижению этой цели.

Модифицированные методы и алгоритмы самоорганизации. Главная проблема алгоритма самоорганизации WTA – это наличие «мертвых» нейронов, которые после инициализации ни разу не побеждают в конкурентной борьбе. Тем самым уменьшается эффективное число нейронов, принимающих участие в классификации входных образов, а, следовательно, увеличивается общая погрешность распознавания. Для разрешения этой проблемы применяются модифицированные методы и алгоритмы самообучения, дающие обычно существенно лучшие результаты, чем алгоритм WTA.

Метод выпуклой комбинации (convex combination method) [16]. На этапе инициализации сети все начальные веса берутся равными $1/\sqrt{m}$, где m – число входов сети (а, следовательно, и компонент весового вектора). Благодаря этому, все весовые векторы совпадают и имеют единичную длину. Каждой компоненте вектора **X** вначале придается значение

$$x_i(0) = \alpha x_i + \frac{1-\alpha}{\sqrt{m}}.$$

Вначале α очень мало, входные векторы имеют длину, близкую к $1/\sqrt{m}$ и почти совпадают с векторами весов. В процессе обучения α постепенно

увеличивают, приближая к 1. Т.е. компоненты вектора **X** постепенно приближаются к своим истинным значениям. Весовые векторы отслеживают один или небольшую группу входных векторов и в конце обучения дают требуемую картину выходов. Метод хорошо работает, но замедляет процесс обучения, так как весовые векторы подстраиваются к изменяющимся входным векторам.

Метод добавления шума к входным векторам [16]. В начале обучения к компонентам входных векторов искусственно добавляют определенный уровень шума. В результате случайные изменения входного вектора, в конце концов, схватывают нужный весовой вектор. В процессе обучения уровень шума во входных векторах постепенно снижают, приближая их к своим истинным значениям. Метод работоспособен, но еще более медленен, чем предыдущий.

Метод интерполяции [16]. В алгоритме WTA в слое самоорганизации активируется только один нейрон. Этот метод называется еще *методом аккредитации.* Его точность ограничена, т.к. выходной сигнал слоя полностью определяется лишь состоянием одного единственного нейрона. В слоях самоорганизации применяется и другой метод – *метод интерполяции*, когда ненулевые выходные сигналы имеет целая группа нейронов. Состав группы формируется по принципу наибольших значений выходных сигналов в слое. Оптимальный размер группы зависит от задачи и общего решения для выбора этого размера не существует. После определения группы нейронов с наибольшими выходными сигналами, их выходы рассматриваются как вектор $\mathbf{Z}_{iK}$, длина которого нормализуется на единицу:

$$z_{iK}^{н} = \frac{z_{iK}}{\sqrt{z_{1K}^2 + z_{2K}^2 + ... + z_{qK}^2}},$$

где q – число компонент вектора $\mathbf{Z}_{iK}$.

Все нейроны вне группы имеют нулевые выходы. Метод потенциально может давать более точные результаты, чем метод WTA, однако убедительных данных, позволяющих объективно сравнить методы аккредитации и интерполяции, нет.

Алгоритм «Победитель справедливо получает все» (Conscience Winner Takes All – CWTA) [15]. Является улучшенной модификацией алгоритма WTA. Он позволяет учитывать усталость нейронов путем подсчета количества побед каждого из них и поощрять нейроны с наименьшей активностью для выравнивания их шансов. Для этого подсчитывается среднее значение случаев выигрыша каждым нейроном слоя самоорганизации. Если он становится победителем чаще своей законной доли времени (примерно $1/n$, где n – число нейронов слоя), то он временно увеличивает свой порог, что автоматически уменьшает его шансы на выигрыш, давая, тем самым, шансы обучаться и

другим нейронам. Этот алгоритм считается одним из наилучших и наиболее быстрых в классе алгоритмов самоорганизации.

Алгоритм «Победитель получает больше» (Winner Takes Most – WTM) [15]. В слое самоорганизации кроме нейрона-победителя обучаются также нейроны из его ближайшего окружения. При этом, чем дальше какой-либо нейрон находится от победителя, тем меньше изменяются его веса.

Уравнение, описывающее самообучение слоя 1, имеет вид

$$w_{ik}^{\text{н}} = w_{ik}^{\text{с}} + \eta_k \cdot G(k,x) \cdot (x_i - w_{ik}^{\text{с}}), \qquad (9.5)$$

где k – номер нейрона в окрестности нейрона-победителя j;

η_k – коэффициент скорости обучения k-нейрона;

$G(k, x)$ – функция соседства.

В выражении (9.5) η_k в общем случае есть функция, значения которой уменьшаются с увеличением расстояния между k-нейроном и j-нейроном-победителем, а также – с течением времени.

Если функция соседства $G(k, x)$ в (9.5) определяется в форме

$$G(k,x) = \begin{cases} 1, \text{для} & k = j; \\ 0, \text{для} & k \neq j, \end{cases}$$

то получим классический алгоритм обучения (9.4).

Алгоритм нейронного газа [15]. Наилучшая самоорганизация достигается при использовании алгоритма нейронного газа. Для этого алгоритма функция соседства выбирается в виде

$$G(k,x) = \exp\left(-\frac{\alpha(k)}{\lambda}\right), \qquad (9.6)$$

где $\alpha(k)$ – номер позиции в очередности нейронов при их сортировке в зависимости от степени удаленности d до вектора $\mathbf{X}$:

$$d_0 < d_1 < d_2 < \ldots < d_{n-1}.$$

Степень удаленности d_0 соответствует нейрону-победителю;

λ - уровень соседства, измеряемый количеством нейронов от данного нейрона до нейрона-победителя.

В начале обучения параметру λ приписываются большие значения, постепенно снижая его до нуля. Снижение λ может быть линейным или показательным. В одной из работ предложено изменять $\lambda(k)$ так:

$$\lambda(k) = \lambda_{\max} \cdot \left(\frac{\lambda_{\min}}{\lambda_{\max}} \right)^{k / k_{\max}},$$

где k – номер итерации;

$\lambda_{\max}$, $\lambda_{\min}$ – принятые максимальное и минимальное значения λ;

$k_{\max}$ – заданное максимальное число итераций.

Коэффициент скорости обучения η также может изменяться линейно или показательно:

$$\eta(k) = \eta(0) \cdot \left(\frac{\eta_{\min}}{\eta(0)} \right)^{k / k_{\max}}.$$

На практике лучшие результаты дает линейное изменение η.

С целью сокращения объема вычислений при сортировке можно учитывать только наиболее значимые величины функции $G(k, x)$. Для этого в выражении (9.6) при $\alpha(k) >> 1$ берут $G(k, x) = 0$.

Сравнение алгоритмов [15]. Наихудшие результаты дает классический алгоритм **WTA** (Кохонена). Значительная часть нейронов размещается в областях, свободных от данных, т.е. вообще не используются сетью. Алгоритмы **WTM** – это целый класс алгоритмов. Эффективность конкретного алгоритма этого класса зависит от вида функции $G(k, x)$. Существенно лучшие результаты, чем **WTA**, дает алгоритм **CWTA**. Алгоритм нейронного газа несколько лучше, чем **CWTA** по результатам самоорганизации, но значительно медленнее за счет операции сортировки. В усеченном виде (см. выше) он примерно эквивалентен **CWTA**, как по эффективности самоорганизации, так и по скорости обучения.

В целом, следует отметить, что для разных приложений сети ВР, применение указанных методов дают разный эффект, который в общем случае заранее не известен.

Статистические свойства обученного слоя самоорганизации. Метод самообучения Кохонена обладает интересной и полезной способностью выявлять статистические свойства входных данных. Обученный слой самоорганизации предоставляет, по существу, карту распределения входных образов, т.е. решает задачу нахождения кластеров в пространстве входных образов. Кохоненом было показано, что для полностью обученной сети вероятность того, что случайно выбранный входной вектор будет ближайшим к любому заданному весовому вектору, равна $1/n$, где n – число нейронов в слое 1. Это является оптимальным распределением весов на гиперсфере. (Предполагается, что используются все весовые векторы, т.е. используется один из обсуждавшихся методов распределения весов) [16].

Обучение слоя Гроссберга [16]**.** Обучение слоя 1 является самообучением, протекающим без учителя. Поэтому заранее неизвестно, какой нейрон этого слоя будет активирован для заданного входного вектора, т.е. слой 1 производит классификацию кластеров входных образов в

недетерминированных позициях своих выходных сигналов. Для того чтобы преобразовать полученную случайную картину выходов слоя 1 в желаемую картину выходных сигналов классификатора, необходим слой 2. Кроме этого, слой 2 может быть использован для исправления неверно проведенной классификации кластеров в слое 1: объединения единичных выходных сигналов слоя 1, соответствующих одному кластеру (см. пример на рис.9.10, б).

Выходной вектор $\mathbf{Z}_K$ слоя 1 поступает на входы слоя 2, и выходные сигналы слоя 2 вычисляются точно так же, как и в рабочем режиме:

$$\mathbf{Y}_G = \mathbf{Z}_K \cdot \mathbf{V};$$
$$\mathbf{Z}_G = F(\mathbf{Y}_G) = F(\mathbf{Z}_K \cdot \mathbf{V}).$$

Обучение слоя 2 осуществляется с учителем. Цель обучения – минимизировать ошибку между реально полученным выходным вектором $\mathbf{Z}_G$ и желаемым выходным вектором $\mathbf{Z}_G^T$.

В слое 1 только один нейрон имеет выходной сигнал, равный 1. При использовании линейной активационной функции в нейронах слоя 2 с коэффициентом a=1 это приводит к тому, что каждый нейрон слоя 2 будет выдавать величину веса v, который связывает этот нейрон с нейроном-победителем слоя 1:

$$z_{Gj} = (z_{Ki}=1) \cdot v_{ij} = v_{ij}. \qquad (9.7)$$

В слое 2 корректируются не все веса, а только те, которые связаны с нейроном-победителем слоя 1. Коррекция осуществляется на основе классического дельта-правила:

$$v_{ij}^{\text{н}} = v_{ij}^{\text{с}} + \eta \cdot (z_{\text{Т},Gj} - z_{Gj}) \cdot z_{Ki}\,, \qquad (9.8)$$

где: $v_{ij}^{\text{н}}$ $v_{ij}^{\text{с}}$ – новое (после коррекции) и старое (до коррекции) значения веса, соединяющего i-нейрон слоя 1 с j-нейроном слоя 2;

$z_{\text{Т},Gj}$, z_{Gj} – желаемое (терминальное) и реальное значение выходного сигнала j-нейрона слоя 2;

z_{Ki} – выходной сигнал i-нейрона слоя 1 (только для одного нейрона слоя 1 он отличен от нуля);

η – коэффициент скорости обучения.

Согласно (9.7) формулу коррекции весов слоя 2 можно записать в рабочем виде

$$v_{ij}^{\text{н}} = v_{ij}^{\text{с}} + \eta \cdot (z_{\text{Т},Gj} - v_{ij}^{\text{с}}) \cdot z_{Ki}\,. \qquad (9.9)$$

Полная сеть встречного распространения. Рассмотренная выше сеть ВР является упрощенной версией сети прямого действия. Полная двунаправленная сеть ВР показана на рис. 9.11 [16].

В рабочем режиме сети предъявляются нормализованные единичные входные векторы **X** и **Y**, а на выходе сети формируются ответы в виде векторов **X***'* и **Y***'*, являющихся аппроксимациями векторов **X** и **Y**.

В режиме обучения на сеть подаются одновременно векторы **X** и **Y**, которые выполняют двоякую роль: с одной стороны это входные векторы, с другой – векторы желаемой выходной реакции сети. При этом слой 1 воспринимает входные сигналы **X** и **Y**, как если бы это был один большой вектор, составленный из **X** и **Y**. Вектор **X** используется для обучения выходов **X***'*, вектор **Y** – для обучения выходов **Y***'*. Сеть обучается по описанной выше схеме. При этом в слое 1 вначале обучаются веса всех нейронов в окрестности нейрона-победителя с постепенным сужением радиуса обучения до одного нейрона-победителя. Темпы обучения для слоев 1 и 2 должны быть согласованными.

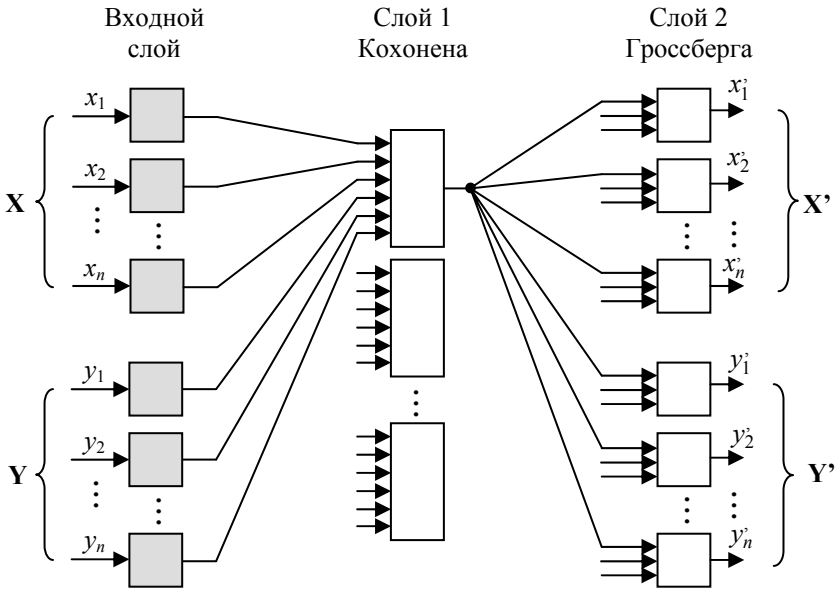


Рис. 9.11. Полная двунаправленная сеть встречного распространения

Интересным в этой сети является то, что предъявление ей только вектора **X** (с вектором **Y**, равным 0) порождает как выходы **X***'*, так и выходы **Y***'*. Т.е. сеть аппроксимирует некоторую функцию *F*, отображающую **X** в **Y***'*. Если функция *F* обратима, то предъявление сети только вектора **Y** порождает и выходы **X***'*.

Кроме этого сеть способна решать гибридную задачу по восстановлению отдельных недостающих компонент векторов **X** и **Y**. Свойства обобщения и восстановления недостающих сигналов отличают сеть ВР от всех других нейронных парадигм, превращая ее в уникальный нейросетевой инструмент.

Сеть ВР получила свое название из-за встречного противотока. На рис. 9.11 он явно не виден, однако Хехт-Нильсен, с учетом описанного принципа ее работы, изображал ее несколько иначе.

10. Стохастические нейронные сети

Стохастическое обучение в ИНС. Целью стохастического обучения является такая подстройка весов связей, при которой состояния нейронов удовлетворяют желаемому распределению вероятностей.

Простейшим принципом стохастического обучения ИНС с прямыми связями является поиск в случайном направлении. Обучение осуществляется с учителем. Перед обучением весам сети придаются небольшие случайные значения, равномерно распределенные в заданном диапазоне (обычно 0-1). Сети поочередно предъявляются входные образы из обучающей выборки и вычисляются выходные векторы. Реально полученный выходной вектор $\mathbf{Z}_\mathrm{A}$ покомпонентно сравнивается с желаемым выходным вектором сети $\mathbf{Z}_\mathrm{T}$ по формуле

$$S = \sum_{i=1}^{n} (z_{\mathrm{A}i} - z_{\mathrm{T}i})^2 \ ,$$

где $z_{\mathrm{A}i}$ – фактический выходной сигнал;

$z_{\mathrm{T}i}$ – желаемый (терминальный) выходной сигнал;

S – целевая функция.

Цель обучения – минимизация целевой функции S. Процедура коррекции весов некоторого нейрона сводится к случайному выбору весового коэффициента этого нейрона и случайному его изменению. Если коррекция помогает (уменьшает S), то она сохраняется, в противном случае происходит возврат к прежнему значению весового коэффициента. Итерации повторяются до тех пор, пока сеть не будет обучена в достаточной степени.

Ловушки локальных минимумов. Сложность обучения заключается в том, что процесс минимизации целевой функции может попасть в ловушку локального минимума. Рис. 10.1 иллюстрирует проблему в системе с единственным весом [16].

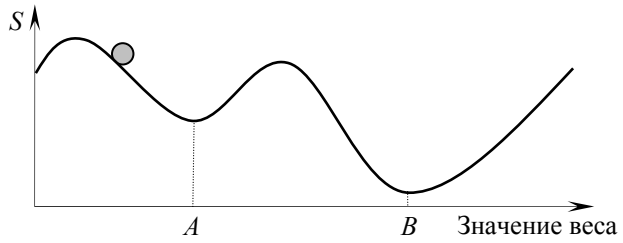


Рис. 10.1. Проблема локальных минимумов

Допустим, в результате случайных коррекций вес принял значение, близкое к точке *A*. Если дальнейшие случайные изменения веса будут малы, то любые отклонения от точки *A* приводят к увеличению целевой функции *S,* и, следовательно, будут отвергнуты. Система окажется в ловушке локального минимума (точка *A*) и никогда не сможет достичь абсолютного минимума (точка *B*). Если случайные коррекции веса в области точки *A* будут очень велики, система будет сильно колебаться во всем диапазоне весов и может также никогда не остановиться в точке абсолютного минимума *B*.

Проблема попадания нейронной сети в ловушки локальных минимумов свойственна всем алгоритмам обучения и может быть решена с помощью стохастических методов. Полезная стратегия в такой ситуации заключается в больших начальных шагах и постепенном уменьшении размера среднего случайного шага. Это позволяет сети вырваться из локальных минимумов и в то же время гарантирует ее окончательную стабилизацию в глобальном минимуме.

Введем горизонтальную и вертикальную границы двумерной системы, изображенной на рис. 10.1, с тем, чтобы шарик мог перемещаться по кривой поверхности в этих границах. Если теперь всю систему сильно трясти в горизонтальной плоскости, то шарик будет быстро перекатываться от одной вертикальной границы к другой, нигде не задерживаясь. В каждый момент времени шарик будет с равной вероятностью находиться в любой точке поверхности. Если постепенно уменьшать силу встряхивания, то будет достигнуто условие, при котором шарик будет на короткое время «застревать» в точке *B*. Если постепенно уменьшать силу встряхивания, возникнет условие, когда шарик будет на короткое время «застревать» как в точке *B*, так и в точке *A*. При дальнейшем уменьшении силы встряхивания будет достигнута критическая точка, когда сила встряхивания будет достаточна для перемещения шарика из точки *A* в точку *B*, но недостаточна для обратного подъема шарика в точку *A*. В результате, когда амплитуда встряхивания уменьшится до нуля, шарик окончательно остановится в точке глобального минимума *B*. Этот процесс иллюстрирует рис. 10.2.

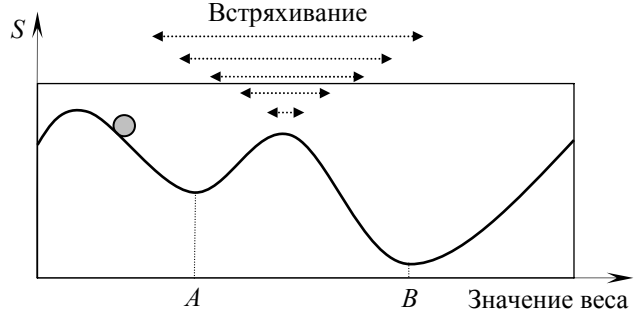


Рис. 10.2. Преодоление проблемы локальных минимумов

Аналогичным образом можно обучать ИНС. Очевидно, что сила встряхивания системы весов эквивалентна амплитуде их среднего случайного изменения. Поэтому при обучении сети вначале делают большие случайные коррекции весов с сохранением только тех изменений весов, которые уменьшают целевую функцию. Затем средний размер коррекции постепенно уменьшают, что позволяет в итоге достичь глобального минимума целевой функции *S*.

Отжиг металла и его имитация [16]. Описанный выше процесс напоминает отжиг металла. В металле, нагретом до температуры, превышающей точку плавления, атомы находятся в сильно беспорядочном движении. Как и во всех физических системах, атомы стремятся к состоянию минимума энергии. В процессе постепенного охлаждения возникают все более низкоэнергетические состояния, пока не будет достигнуто состояние с минимумом энергии – глобальный минимум. Распределение энергетических уровней в процессе отжига описывается законом Больцмана:

$$P(E) = e^{-\frac{E}{kT}},$$

где: $P(E)$ – вероятность того, что система находится в состоянии с энергией E;
k – постоянная Больцмана;
T – температура по шкале Кельвина.

При высоких температурах $P(E)$ близко к 1 для всех энергетических состояний, т.е. разноэнергетические состояния равновероятны. По мере уменьшения температуры вероятность высокоэнергетических состояний уменьшается, а низкоэнергетических – возрастает. При температуре, близкой к 0, вероятность высокоэнергетического состояния близка к 0.

Процесс отжига металла может быть искусственно воспроизведен при обучении ИНС, поэтому для его описания применяется термин «имитация отжига».

Вербальное описание стохастического алгоритма обучения для одного слоя сети:

1^0. Придать всем весам слоя сети $w_{ij}, i = \overline{1,m};\ j = \overline{1,n}$ небольшие случайные значения, равномерно распределенные в заданном диапазоне.

2^0. Определить переменную T как искусственную температуру и придать ей большое начальное значение.

3^0. Предъявить сети входной вектор $\mathbf{X}_p$ из обучающего множества $\Psi = \{\Psi_1, \Psi_2, \ldots, \Psi_L\}$, $p = \overline{1,L}$ и вычислить выходной вектор $\mathbf{Z}_\text{A}$.

4^0. Вычислить целевую функцию S по формуле

$$S = \sum_{i=1}^{n} (z_{Ai} - z_{Ti})^2$$

5^0. Выбрать детерминированным или случайным образом вес w_{ij} и изменить его случайным образом.

6^0. Пересчитать выходной вектор $\mathbf{Z}_A$ сети и изменение целевой функции S.

7^0. Если S уменьшилась, то сохранить изменение веса, в противном случае вероятность сохранения изменения веса w_{ij} вычисляется в соответствии с распределением Больцмана:

$$P(w_{ij}) = e^{-\frac{w_{ij}}{kT}},$$

где: k – константа, аналогичная константе Больцмана (выбирается в зависимости от задачи;

T – искусственная температура.

8^0. Если $i=m$ и $j=n$ (все веса исчерпаны), то следующий шаг, иначе переход на шаг 5^0.

9^0. Если обучающее множество Ψ исчерпано ($p=L$), то конец алгоритма, иначе – переход на шаг 2^0.

Вычисление случайного изменения веса w на шаге 5^0 можно делать с помощью метода Монте-Карло. Например, подобно тепловой системе весовое изменение может выбираться в соответствии с гауссовым распределением

$$P(w) = e^{-\frac{w^2}{T^2}},$$

где $P(w)$ – вероятность изменения веса w.

Для определения самой величины изменения веса Δw, а не вероятности изменения веса, имеющего величину w, необходимо численно проинтегрировать $P(w)$ от 0 до w и результат представить в виде таблицы значений Δw. Затем выбрать случайное число, равномерно распределенное в диапазоне (0,1) и, используя его в качестве $P(w)$, определить из таблицы Δw.

Вычисление на шаге 7^0 также делается с помощью метода Монте-Карло следующим образом. Выбирается случайное число r, равномерно распределенное в диапазоне (0,1) и сравнивается с величиной $P(w)$. Если $P(w)>r$, то изменение веса сохраняется, в противном случае величина веса возвращается к предыдущему значению. Такая процедура позволяет делать случайные шаги в направлении увеличения целевой функции S, обеспечивая тем самым выход сети из локальных минимумов S.

Машина Больцмана [16]. ИНС, использующую при обучении имитацию отжига называют *машиной Больцмана* (МБ).

При изучении свойств МБ было показано, что для достижения сходимости сети к глобальному минимуму, скорость уменьшения температуры T должна быть обратно пропорциональна логарифму времени:

$$T(t) = \frac{T_0}{\log(1+t)},$$

где: $T(t)$ – искусственная температура как функция времени;
T_0 – начальное значение искусственной температуры;
t – искусственное время.

Такая оценка предсказывает очень медленную скорость охлаждения и, следовательно, очень большое время обучение МБ, что и подтверждается экспериментально.

Обучение Коши, машина Коши [16]. Существует метод ускорения обучения стохастической сети, который состоит в использовании при вычислении размера шага вместо распределения Больцмана распределение Коши.

Распределение Коши имеет более длинные «хвосты», что повышает вероятность больших шагов, рис. 10.3.

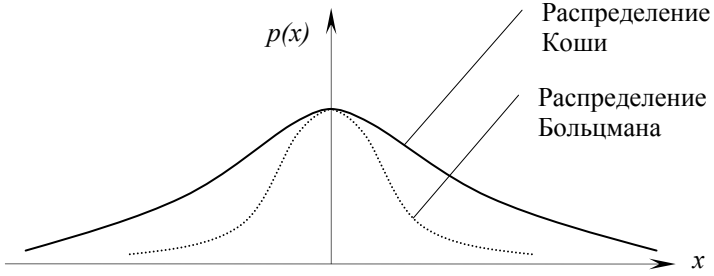


Рис. 10.3. Распределение Коши и распределение Больцмана

Для распределения Коши максимальная скорость уменьшения температуры становится обратно пропорциональной линейной величине, а не логарифму:

$$T(t) = \frac{T_0}{1+t}$$

Распределение Коши имеет вид

$$P(w) = \frac{T(t)}{T(t)^2 + w^2},$$

где $P(w)$ есть вероятность изменения веса w.

Это уравнение интегрируется стандартными методами. Решая относительно w, получим

$$\Delta w = \eta \cdot \{T(t) \cdot \mathrm{tg}[P(w)]\},$$

где: η – коэффициент скорости обучения;

Δw – изменение веса.

Нахождение Δw осуществляется методом Монте-Карло. Для этого выбирается случайное число, равномерно распределенное на открытом интервале $(-\pi/2, \pi/2)$, соответствующем изменению тангенса, и подставляется в формулу Δw в качестве $P(w)$.

ИНС, использующая распределение Коши, называют машиной Коши (МК). Скорость обучения МК по сравнению с МБ возрастает. Однако время сходимости остается все же достаточно большим и примерно в 100 раз превышает время обучения ИНС ОРО. Кроме того, для МК вероятным и весьма опасным явлением может быть паралич сети. Это связано с тем, что бесконечная дисперсия распределения Коши может привести к неограниченному увеличению значений весов, особенно, если в качестве нелинейности используется логистическая функция.

Комбинирование сетей ОРО и МК. В комбинированном алгоритме коррекция весов состоит из двух компонент: направленной компоненты, вычисляемой с использованием алгоритма ОРО, и случайной компоненты, определяемой распределением Коши. Обе компоненты вычисляются для каждого веса, и их сумма является величиной, на которую изменяется вес. Комбинированная сеть обычно находит глобальный минимум быстрее, чем каждая из ее составляющих отдельно.

В настоящее время существует множество модификаций стохастических сетей, использующих имитацию отжига, а также различные комбинации сетей, включающие стохастические методы обучения. Главным преимуществом этих сетей является возможность достичь глобального минимума целевой функции, избежав ловушек локальных минимумов. Стохастические и комбинированные нейронные сети применяются для решения задач распознавания образов, классификации, адаптивного управления, многопараметрической идентификации и оптимизации, прогнозирования и диагностики.

11. Рекуррентные нейронные сети. Нейронная сеть Хопфилда

Рекуррентные сети. Большинство рассмотренных ранее ИНС не имели обратных связей, т.е. связей с выходов нейронов на их входы. Отсутствие обратных связей гарантирует абсолютную устойчивость сетей. Это означает, что, при подаче на сеть входных сигналов, через короткий промежуток времени, соответствующий переходному процессу, сеть формирует выходные сигналы, которые остаются неизменными до тех пор, пока на сеть вновь не будут поданы входные сигналы. Это, весьма положительное качество сетей с прямыми связями, ограничивает, однако, их функциональные возможности.

Функционально более мощными, чем сети с прямыми связями, являются сети с обратными связями между различными слоями нейронов, − это, так называемые, *рекуррентные* сети (РС). Структура типичной однослойной РС показана на рис. 11.1.

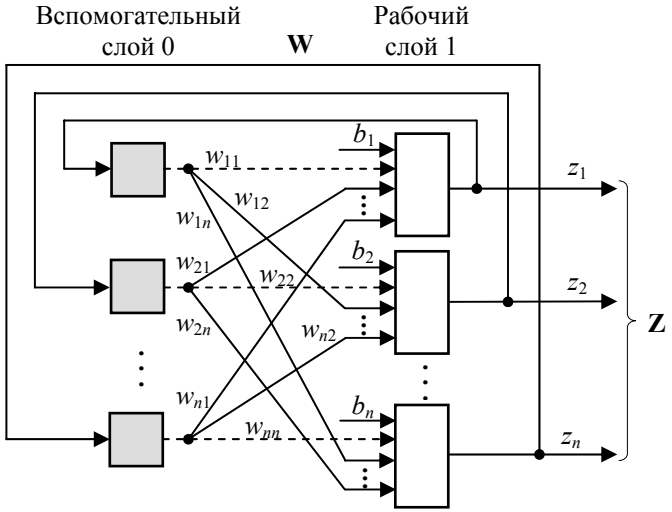


Рис. 11.1. Однослойная рекуррентная сеть

С целью единообразного представления в РС, как и в другие сети, введен вспомогательный слой 0, который вычислений не выполняет и служит для распределения выходных сигналов сети обратно на ее входы. Сеть имеет один рабочий слой 1, совокупность весов которого можно представить матрицей **W**. В слое 1 кроме связей с нейронов слоя 0 имеются внешние входы b_1, b_2, …, b_n, называемые *смещениями*.

Характерная особенность РС состоит в том, что выходные сигналы нейронов рабочего слоя в некоторый момент времени t_k являются одновременно входными сигналами сети в следующий момент времени t_{k+1}: $\mathbf{X}(t_{k+1})=\mathbf{Z}(t_k)$. Это приводит к тому, что отклик РС получается динамическим. Приложение входного сигнала порождает выходной сигнал, который, передаваясь по обратным связям, модифицирует входной сигнал. Изменившийся входной сигнал снова порождает новый выходной сигнал, и т.д. Т.е. изменение состояния одного нейрона отражается на всей сети вследствие обратной связи типа «один ко многим».

Вектор входных сигналов **X** в РС особо не выделяется, а начальное возбуждение сети осуществляется кратковременным приложением $\mathbf{X}(t_0)$ к выходам сети.

Уровень активации нейронов сети вычисляется следующим образом:

$$y_j = \sum_{i \neq j} w_{ij} z_i + b_j, \qquad (11.1)$$

где: b_j – внешний входной сигнал j-нейрона.

В качестве активационной функции F для нейронов сети может быть использована пороговая функция с тремя уровнями:

$$z_j = F(y_j) = \begin{cases} 1, & \text{если } y_j > Q_j, \\ 0, & \text{если } y_j < Q_j, \\ z_j, & \text{если } y_j = Q_j. \end{cases} \qquad (11.2)$$

Если последовательные итерации в РС приводят к все меньшим изменениям выходного сигнала, то спустя некоторое время выходной сигнал перестанет изменяться совсем и сеть придет в устойчивое состояние. Для других же РС этот процесс никогда не заканчивается. Такие сети называются неустойчивыми или *осциллирующими*. Как пример хаотических систем, они сами по себе обладают рядом интересных свойств, однако для большинства приложений систем ИИ необходимы именно устойчивые РС. Поэтому использование РС неизбежно порождает как первоочередную – проблему предсказания ее устойчивости.

Устойчивость РС [16]. В 1983 году Коэном и Гроссбергом было показано, что РС является устойчивой, если ее матрица связей **W**

$$\mathbf{W} = \begin{Vmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & \dots & w_{nn} \end{Vmatrix}$$

симметрична ($w_{ij} = w_{ji}$) и имеет нули на главной диагонали ($w_{ii} = 0$):

$$\mathbf{W} = \begin{Vmatrix} 0 & w_{12} & \dots & w_{1n} \\ w_{21} & 0 & \dots & w_{2n} \\ \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & \dots & 0 \end{Vmatrix}, \quad w_{ij} = w_{ji} \qquad (11.3)$$

В структуре РС симметрия матрицы **W** означает, что вес связи с *i*-элемента на *j*-элемент должен быть равен связи с *j*-элемента на *i*-элемент. Наличие нулей на главной диагонали вводит запрет на наличие обратных связей с *i*-элемента на этот же *i*-элемент. На рис. 11.1. это условие отображено штриховыми связями $w_{11}, w_{22}, \dots, w_{nn}$.

Для доказательства устойчивости РС ей сопоставляется некоторая функция, которая всегда убывает при любых изменениях состояния сети. В пределе эта функция должна достичь минимума и прекратить изменение, гарантируя тем самым устойчивость сети. Такая функция называется *энергетической* функцией (функция энергии, функция Ляпунова, гамильтониан). Для рассматриваемых РС она может быть введена следующим образом:

$$E = -\frac{1}{2}\sum_{\substack{i \\ i \neq j}}\sum_{j} w_{ij} z_i z_j - \sum_j b_j z_j + \sum_j Q_j z_j, \qquad (11.4)$$

где: E – искусственная энергия сети;
w_{ij} – вес связи с выхода нейрона i к входу нейрона j;
z_j – выходной сигнал нейрона j;
b_j – внешний входной сигнал нейрона j;
Q_j – порог нейрона j.

Изменение энергии E, вызванное изменением состояния нейрона j, есть

$$\delta E = -[\sum_{i \neq j} w_{ij} z_i + b_j - Q_j] \cdot \delta z_j = -[y_j - Q_j] \cdot \delta z_j, \qquad (11.5)$$

где δz_j – изменение выходного сигнала нейрона j.

Рассмотрим три возможных случая: $y_j > Q_j$; $y_j < Q_j$; $y_j = Q_j$.

1) Допустим величина y_j больше порога Q_j. Тогда выражение в квадратных скобках будет положительным, а из выражения для выходной функции нейрона (11.2) следует, что выход нейрона j должен измениться в положительную сторону или остаться без изменения (был 0, стала 1 или была 1 и осталась 1). Это значит, что δz_j может быть только положительным или нулем, и,

следовательно, δE должно быть отрицательным. Т.е. энергия сети должна либо уменьшиться, либо остаться без изменения.

2) Допустим, что величина y_j меньше порога Q_j. Тогда выражение в квадратных скобках будет отрицательным, а из выражения для выходной функции нейрона (11.2) следует, что выход нейрона j должен измениться в отрицательную сторону или остаться без изменения (была 1, стал 0 или был 0 и остался 0). Это значит, что δz_j может быть только отрицательным или нулем и, следовательно, δE опять должно быть отрицательным. Т.е. энергия сети опять должна либо уменьшиться, либо остаться без изменения.

3) Наконец, если величина y_j равна порогу Q_j. Тогда выражение в квадратных скобках будет равно 0 и приращение энергии δE также будет равно 0. Т.е. энергия сети останется без изменения.

Из этого анализа следует вывод, что любое изменение состояния нейрона либо уменьшит энергию, либо оставит ее без изменения. Благодаря такой закономерности, энергия, в конце концов, достигнет некоторого минимального уровня и прекратит дальнейшее изменение. По определению такая сеть является устойчивой.

Обычно в РС имеется множество локальных минимумов, каждый из которых представляет одно из возможных состояний системы, сформированных на этапе обучения сети. В пространстве состояний локальные энергетические минимумы E представлены точками стабильности, называемыми *аттракторами* из-за тяготения к ним ближайшего окружения.

Условие устойчивости (11.3) сети является достаточным, но не необходимым. Т.е. имеется много устойчивых сетей, которые этому условию не удовлетворяют. Например, устойчивы все сети прямого действия (без обратных связей). Существуют примеры сетей, в которых небольшие отклонения от условия (11.3) приводят к неустойчивости в виде непрерывных осцилляций, но в большинстве случаев даже приближенной симметрии оказывается вполне достаточно для устойчивости сети.

Сеть Хопфилда. Среди конфигураций ИНС встречаются такие, при классификации которых по принципу обучения, строго говоря, не подходят ни обучение с учителем, ни обучение без учителя. В таких сетях весовые коэффициенты рассчитываются только однажды перед началом функционирования сети на основе информации о задаче, и обучение сети сводится именно к этому расчету. С одной стороны, предъявление априорной информации о задаче можно расценивать, как помощь учителя, но с другой – при поступлении на ее входы реальных данных, она не может изменять свое поведение, поэтому говорить о звене обратной связи с учителем не приходится.

Одна из сетей с подобной логикой работы была предложена в 1984 г. английским физиком Хопфилдом. Впоследствии эта сеть получила широкое распространение и стала называться его именем – *сеть Хопфилда*. Появилось множество ее модификаций. Другие названия: ассоциативная память, автоассоциативная память, модель Хопфилда.

В первоначальном виде сеть Хопфилда представляла собой автоассоциативную сеть, структура которой показана на рис. 11.2 [16].

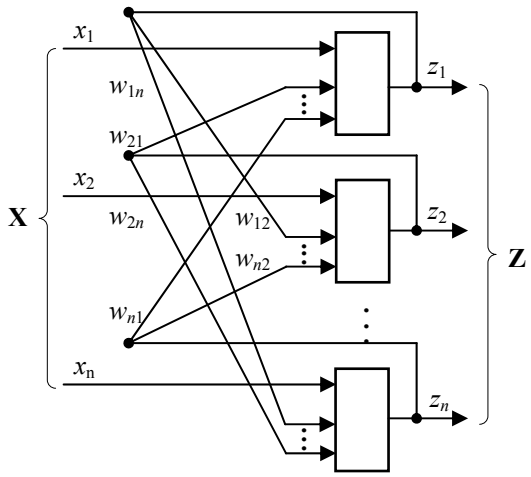


Рис. 11.2. Нейронная сеть Хопфилда

Сеть состоит из единственного рабочего слоя нейронов, число n которых определяет одновременно количество входов и выходов сети. Каждый нейрон имеет отдельный вход для компоненты $x_i,\ i=\overline{1,n}$ вектора **X** входных сигналов. Выход $z_i,\ i=\overline{1,n}$ каждого нейрона обратными связями связан с входами всех остальных нейронов кроме самого себя (соблюдение условия устойчивости). Совокупность выходных сигналов $z_i,\ i=\overline{1,n}$ образует выходной вектор **Z** сети.

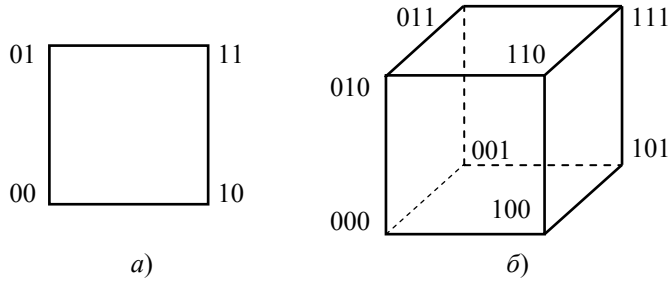
Состояние сети это множество текущих сигналов $z_i,\ i=\overline{1,n}$ всех нейронов. В первоначальном варианте сети Хопфилда состояние каждого нейрона менялось в дискретные случайные моменты времени, в последующих модификациях состояния нейронов менялись одновременно.

В сети Хопфилда могут использоваться формальные нейроны, принимающие выходные значения 0 или 1. В этом случае текущее состояние сети является двоичным числом, каждый бит которого является сигналом z_i, $i=\overline{1,n}$ некоторого нейрона.

Геометрическая интерпретация принципа функционирования сети из двух и трех нейронов показана на рис. 11.3, *а, б* соответственно.

В двухнейронной сети каждой вершине квадрата соответствует одно из четырех состояний системы. Трехнейронная сеть представлена кубом в трехмерном пространстве. Каждой вершине куба соответствует одно из восьми

состояний системы. В общем случае сеть из n нейронов будет иметь 2^n различных состояний и будет представляться n-мерным гиперкубом.



а) *б*)

Рис. 11.3. Система из четырех (*а*) и восьми (*б*) устойчивых состояний

При подаче каждого нового входного вектора сеть переходит из вершины в вершину, пока не стабилизируется. Устойчивая вершина определяется весами, текущими входными сигналами и величиной порога. Если входной вектор искажен или неполон, то сеть стабилизируется в вершине, ближайшей к желаемой.

Ассоциативная память. Обычная компьютерная память является локально адресуемой, т.е. по конкретному адресу из памяти извлекается совершенно конкретная информация. Человеческая же память обладает ассоциативными свойствами. Например, несколько музыкальных тактов могут вызвать целую гамму воспоминаний, включая пейзажи, звуки, запахи, какие-либо сопутствовавшие этому события.

Нейронная сеть Хопфилда также обладает ассоциативными свойствами. По заданной части какого-либо образа из памяти извлекается весь этот образ. Исходя из выше приведенной геометрической интерпретации, ассоциативная память в сети Хопфилда может быть реализована таким подбором весов, при котором в нужных вершинах единичного гиперкуба образуются энергетические минимумы. Такие минимумы энергии E можно соотнести с определенными образами. В этом случае предъявление части одного из таких образов приведет к сползанию сети в устойчивое состояние, соответствующему одному из аттракторов в вершине гиперкуба, которое будет соответствовать полному образу.

Задача, решаемая сетью в качестве ассоциативной памяти, как правило, формулируется следующим образом. Известен некоторый набор двоичных сигналов, которые считаются образцовыми. Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить («вспомнить» по частичной информации) соответствующий образец (если такой есть) или дать заключение о том, что входные данные не соответствуют ни одному из образцов.

В общем случае, любой входной сигнал может быть описан вектором $\mathbf{X} = (x_1, x_2, \ldots, x_n)$, $i = \overline{1,n}$ с двоичными компонентами x_i. Для сети Хопфилда двоичные компоненты x_i входного вектора $\mathbf{X}$ удобнее кодировать биполярными сигналами, т.е. состояниями +1 («включено») и –1 («выключено»). Соответственно, в качестве активационной функции F для нейронов сети вместо (11.2) используется так называемая *signum*-функция [19]:

$$z_j = F(y_j) = \begin{cases} +1, \text{ если } y_j > Q_j; \\ -1, \text{ если } y_j < Q_j, \\ z_j, \text{ если } y_j = Q_j. \end{cases} \qquad (11.6)$$

Третье состояние функции в (11.6) иногда присоединяют ко второму состоянию:

$$z_j = F(y_j) = \begin{cases} +1, \text{ если } y_j > Q_j; \\ \\ -1, \text{ если } y_j \le Q_j. \end{cases}$$

В компактной записи *signum*-функция имеет вид

$$z_j = F(y_j) = \operatorname{sgn} y_j.$$

Пусть в системе необходимо запомнить $k = \overline{1,L}$ образцов. Обозначим вектор, описывающий k-образец, через $\mathbf{X}^k$. На входы сети предъявим некоторый произвольный вектор в метрике $\mathbf{X}^k$. Этот вектор может совпадать с одним из образцов $\mathbf{X}^k$, $k = \overline{1,L}$ или – не соответствовать ни одному из них. Сеть на основе предъявленных ей данных может выделить («вспомнить») некоторый k-образец, как самый ближайший к предъявленным данным. Тогда ее выходы будут содержать именно его, т.е. $\mathbf{Z}=\mathbf{X}^k$, где $\mathbf{Z}=(z_1, z_2, \ldots, z_n)$, $i = \overline{1,n}$ – выходной вектор. В противном случае, выходной вектор $\mathbf{Z}$ будет представлять собой некоторый ложный образ, не совпадающий ни с одним из образцов.

На стадии инициализации (обучения) сети весовые коэффициенты устанавливаются следующим образом [19]:

$$w_{ij} = \begin{cases} \sum_{k}^{L} (x_i^k)^{\mathrm{T}} \cdot x_j^k, & i \ne j \\ 0, & i = j \end{cases}, \qquad (11.7)$$

где x_j^k – j-элемент вектора $\mathbf{X}^k$;

$\left(x_i^k\right)^{\mathrm{T}}$ – i-элемент транспонированного вектора $\mathbf{X}^k$.

В матричной форме выражение (11.7) имеет вид

$$\mathbf{W} = \sum_{k=1}^{L} (\mathbf{X}^k)^{\mathrm{T}} \mathbf{X}^k .$$

Принцип формирования матрицы коэффициентов **W** поясним на примере [19]. Пусть, в сети должны быть запомнены два образца:

$$\mathbf{X}^1 = [-1 \quad 1 \quad -1],$$
$$\mathbf{X}^2 = [\ 1 \quad -1 \quad 1].$$

Весовая матрица **W** сети вычисляется следующим образом:

$$\mathbf{W} = (\mathbf{X}^1)^{\mathrm{T}} \cdot \mathbf{X}^1 + (\mathbf{X}^2)^{\mathrm{T}} \cdot \mathbf{X}^2 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 1 & -1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} =$$
$$= \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{bmatrix}.$$

Алгоритм функционирования сети следующий.

1^0. На сеть подается неизвестный входной сигнал. Фактически его ввод осуществляется непосредственной кратковременной установкой значений компонент выходного вектора **Z** (обозначение на схеме сети входных сигналов в явном виде носит чисто условный характер):

$$z_i(0) = x_i ,\ i = \overline{1,n} .$$

Цифра в скобке означает номер итерации в цикле работы сети.

2^0. Рассчитываются новые значения состояний нейронов:

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

и новые значения выходных сигналов

$$z_j(t+1) = F[s_j(t+1)] ,$$

где F – активационная функция нейрона, вычисляемая согласно (11.6).

3^0. Выполняется проверка, изменились ли выходные сигналы сети за последнюю итерацию:

$$z_j(t+1) = z_j(t).$$

Если да, – переход к пункту 2^0, иначе (если выходные сигналы остались прежними) – конец. При этом выходной вектор **Z** представляет собой образец, наилучшим образом сочетающийся с входными данными.

Энергетическая функция сети Хопфилда, доказывающая ее способность сходится к устойчивому набору своих состояний, соответствует виду (11.4).

Число запоминаемых образцов. Как говорилось выше, иногда сеть Хопфилда не может провести распознавание входного образа и выдает на выходе ложный (несуществующий) образ. Это связано с проблемой ограниченной возможности сети по запоминанию образцов. Очевидно, что для сети Хопфилда число запоминаемых образцов L не должно превышать некоторой величины $L_{\max}$, зависящей от числа n нейронов сети. Точный расчет $L_{\max}$ затруднен тем обстоятельством, что запоминаемые сетью образцы обычно коррелированы между собой. Это означает, что часть нейронов, необходимых для запоминания одного образца обычно используется сетью для запоминания другого образца. Для слабо коррелированных образцов известна приближенная оценка [16]

$$L_{\max} = \frac{n}{2\ln n}.$$

На практике чаще используют приближенную оценку $L_{\max} = 0{,}15n$.

Вместе с тем, польский специалист по ИНС Станислав Осовский утверждает, что созданная в Варшавском политехническом университете программа, использующая метод обучения на основе псевдоинверсии, позволяет получить $L_{\max} = n$-1 [15].

Сеть Хопфилда с непрерывной активационной функцией. В работах Хопфилда рассмотрены также сети, использующие модели нейронов с непрерывной активационной функцией F, которая точнее моделирует биологический прототип [16]. В общем случае это S-образная или логистическая функция

$$z(x) = F[y(x)] = \frac{1}{1+e^{-\lambda y(x)}}, \qquad (11.8)$$

где λ – коэффициент, определяющий крутизну сигмоидальной функции.

Если λ велико, F приближается к пороговой функции и сеть функционирует подобно бинарной системе, стабилизируясь в вершинах

единичного гиперкуба. Меньшие значения λ дают более пологий наклон характеристики, устойчивые точки удаляются от вершин гиперкуба, последовательно исчезая по мере приближения λ к нулю, рис. 11.4.

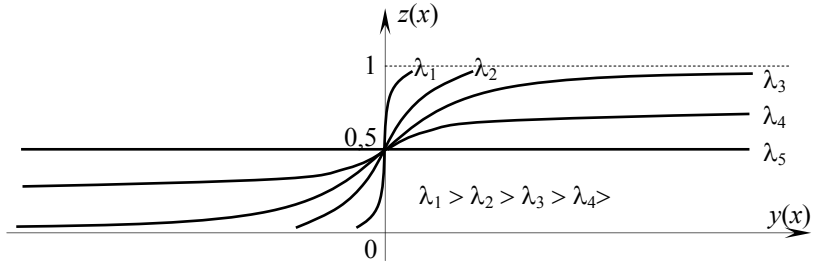


Рис. 11.4. Вид логистической функции для различных значений λ

Устойчивость таких сетей, как и бинарных, гарантируется свойствами (11.3) матрицы коэффициентов. Доказательство устойчивости также основано на использовании соответствующей энергетической функции, которая концептуально не отличается от (11.4).

В целом, ассоциативная память, реализованная в сети Хопфилда как на бинарных, так и на непрерывных элементах строится по одному принципу. Конструируется энергетическая функция, минимумы которой совпадают с устойчивыми состояниями сети, соответствующими заданному набору образцов, которые необходимо запомнить. После обучения сеть самопроизвольно «скатывается» к тому образцу, в чей «бассейн притяжения» попадает входной образ.

Стохастическая сеть Хопфилда. Сеть Хопфилда, как и другие детерминированные сети, может стабилизироваться в локальных минимумах, не достигая глобального минимума. Эта проблема, как и в других сетях, решается применением стохастических методов, рассмотренных ранее. Если правила изменения состояний для бинарной сети Хопфилда задать стохастистически, а не детерминированно, как на шаге 2^0 алгоритма, то можно построить систему, имитирующую отжиг металла [16]. Для ее реализации вводится вероятность изменения веса как функция от величины E, на которую выход k-нейрона y_k превышает его порог Q:

$$E_k = y_k - Q_k,$$

$$p_k = \frac{1}{1 + e^{\frac{-\delta E_k}{T}}},$$

где T – искусственная температура.

Вначале T приписывается большое значение, нейроны устанавливаются в начальном состоянии, определяемом входным вектором, и сети предоставляется возможность искать минимум энергии в соответствии со следующей процедурой:

1^0. Приписать состоянию каждого нейрона с вероятностью p_k значение 1, а с вероятностью $(1 - p_k)$ – значение 0.

2^0. Постепенно уменьшать T и повторять шаг 1^0, пока не будет достигнуто равновесие.

Результатом такой процедуры будет стабилизация сети в глобальном минимуме. Недостаток этого метода состоит в увеличении времени релаксации сети к устойчивому состоянию.

Области применения сетей Хопфилда. Выше было рассмотрено использование сети Хопфилда в качестве ассоциативной памяти. Известны и другие прикладные области для сетей такого типа: решение задач классификации, оптимизационных задач, аналого-цифрового преобразования и ряда других. Появились интересные работы по использованию сетей Хопфилда в качестве интеллектуальных средств анализа знаний и обеспечения информационной безопасности компьютерных систем.

Оптимизационные задачи. Хопфилд предложил использовать свойство своей сети минимизировать энергетическую функцию для решения оптимизационных задач. Принцип решения состоит в следующем. Вначале выбирается нейронное представление для конкретной задачи, т.е., исходя из ее условий, состоянию нейронов приписывается определенный смысл. Затем с учетом ограничений, налагаемых на задачу, конструируется энергетическая функция таким образом, чтобы в состояниях, представляющих возможные решения, она была пропорциональна стоимостной функции задачи. После этого по энергетической функции определяются веса связей, пороги нейронов и конструируется соответствующая сеть. Т.к. в процессе функционирования сеть минимизирует свою энергетическую функцию, динамика в пространстве состояний направлена на минимизацию соответствующей стоимостной функции. Поиск минимума энергии занимает всего несколько постоянных времени нейронов, поэтому сеть быстро достигает стабильного состояния, из которого декодируется решение задачи. Поскольку в таких задачах, в отличие от ассоциативной памяти, требуется нахождение не локального, а глобального энергетического минимума, в сети используются не бинарные, а аналоговые нейроны с сигмоидальными активационными функциями. Решения оптимизационных задач с помощью нейронной сети не самые лучшие, но достаточно близки к ним. При этом существенно экономится время для получения квазиоптимального решения. Примером сложной оптимизационной задачи, сетевое решение которой предложил Хопфилд, является задача коммивояжера.

Задача коммивояжера. Кратко эта задача может быть сформулирована так: требуется найти маршрут минимальной длины в пределах нескольких

городов, начиная от некоторого исходного города, посещая каждый город один раз и возвращаясь в исходный город. Задача коммивояжера (ЗК) относится к классу задач, называемых математиками *NP-полными (недетерминистки полиномиальные)*. Трудность решения таких задач состоит в высокой степени возрастания числа комбинаций при увеличении параметра, характеризующего объем задачи. Степень возрастания комбинаций в *NP*-полных задачах растет экспоненциально с увеличением объема задачи. Попытки прямого решения таких задач методом простого перебора комбинаций вскоре порождает *комбинаторный взрыв*, который быстро исчерпывает возможности обычных компьютеров. В ЗК параметром, задающим объем задачи, является число городов. При большом числе городов простой перебор из-за колоссальных временных затрат становится просто невозможным, и решение обычно сводится к поиску приемлемых, хотя и неоптимальных, эвристических решений. Решение ЗК на сети Хопфилда также относится к эвристическим и потому является квазиоптимальным. Вместе с тем, решение получается так быстро, что метод оказывается весьма привлекательным.

Пусть схема городов выглядит так, как показано на рис. 11.5 [16].

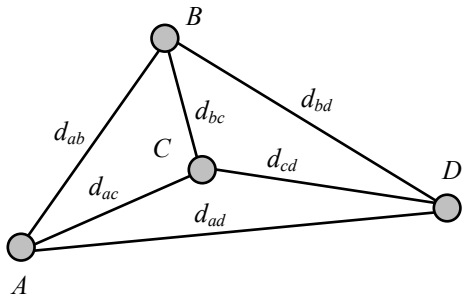


Рис. 11.5. Схема городов в задаче коммивояжера

Решением является упорядоченное множество из n городов (на схеме n=4). В сети будем использовать нейроны с большой крутизной выходной характеристики, близкой к пороговой. Каждый город представлен строкой из n нейронов. Выход только одного нейрона в строке равен 1 (остальные равны 0). Позиция единицы в строке показывает порядковый номер, в котором город посещается при обходе. В нашем примере для n=4 городов потребуется n^2 нейронов, т.е. 16. Порядок посещения городов может быть, например, таким, как показан в табл. (11.1), т.е: *C, A, D, B*. Длина такого маршрута $L = d_{ca} + d_{ad} + d_{db} + d_{bc}$.

ЗК с n городами соответствует $n!$ различных маршрутов. Для нашего простого примера число возможных маршрутов равно 24. Для большего числа городов рост числа возможных маршрутов несет взрывной характер. Например,

при n=60 число возможных маршрутов становится равным $6934155 \cdot 10^{78}$ (для сравнения в нашей галактике «лишь» 10^{11} звезд). Время решения такой задачи простым перебором даже на самом быстром компьютере будет соразмерно с геологическими эпохами.

Таблица 11.1

Задание порядка посещения городов

Город	Порядок следования			
	1	2	3	4
A	0	1	0	0
B	0	0	0	1
C	1	0	0	0
D	0	0	1	0

Построим сеть для решения ЗК. Снабдим каждый нейрон сети двумя индексами: первый соответствует городу; второй – порядковому номеру его посещения в маршруте. Энергетическая функция должна удовлетворять двум требованиям:

– должна быть малой только для тех решений, которые имеют по одной единице в каждой строке и в каждом столбце;

– должна оказывать предпочтение решениям с более короткой длиной маршрута.

Первое требование удовлетворяется, если энергетическая функция будет иметь вид:

$$E = \frac{K_1}{2} \sum_x \sum_i \sum_{j \neq i} z_{xi} z_{xj} + \frac{K_2}{2} \sum_i \sum_x \sum_{y \neq x} z_{xi} z_{yi} + \frac{K_3}{2} \left[\left(\sum_{\mathrm{x}} \sum_{\mathrm{i}} z_{xi} \right) - n \right]^2, \quad (11.9)$$

где K_1, K_2, K_3 – некоторые константы.

В выражении (11.9):

– первый член равен нулю только в том случае, если каждая строка (город) содержит не более одной единицы.

– второй член равен нулю только в том случае, если каждый столбец (порядковый номер посещения) содержит не более одной единицы.

– третий член равен нулю только в том случае, если матрица содержит ровно n единиц.

Второе требование – предпочтение коротким маршрутам – удовлетворяется добавлением в энергетическую функцию (11.9) четвертого члена:

$$E = \frac{1}{2K_4} \sum_x \sum_{y \neq x} \sum_i d_{xy} z_{xi} (z_{y,\,i+1} + z_{y,\,i-1}), \qquad (11.10)$$

где K_4 – некоторая константа.

Этот член представляет собой длину любого допустимого маршрута. Для удобства индексы определяются по модулю n, т.е. $z_{n+j} = z_j$.

При достаточно больших значениях K_1, K_2 и K_3 низкоэнергетические состояния будут соответствовать допустимым маршрутам, а большие значения K_4 гарантируют, что будет найден короткий маршрут.

Теперь устанавливается соответствие между членами полученной энергетической функции (11.9), (11.10) и членами общей формы этой функции (11.4). В результате находятся элементы весовой матрицы:

$$w_{xi,\,yj} = -K_1 \delta_{xy} (1 - \delta_{ij}) - K_2 \delta_{ij} (1 - \delta_{xy}) - K_3 - K_4 d_{xy} (\delta_{j,\,i+1} + \delta_{j,\,i-1}), \qquad (11.11)$$

где $\delta_{ij} = 1$, если $i=j$ и $\delta_{ij} = 0$ в противном случае.

Кроме того, каждый нейрон имеет дополнительный вход со смещающим весом $b_i = K_3 \cdot n$, соединенный с константой +1.

Сеть, динамика которой управляется энергетической функцией (11.9), (11.10), будет искать решение, стремясь к некоторому конечному состоянию, имеющему вид матрицы перестановок (см. табл. 11.1), с выбором «лучшего» маршрута из множества $n!$ конечных состояний, удовлетворяющих наложенным ограничениям.

Хопфилд моделировал, в частности, ЗК с десятью городами. Расположение городов выбиралось случайно с постоянной плотностью вероятности. Параметры были следующими: $K_1 = K_2 = 500$; $K_3 = 200$; $K_4 = 500$. Для того чтобы оценить оптимальность выбранных сетью маршрутов, на компьютере путем простого перебора был сделан подсчет всех возможных маршрутов, которых было 181440. В результате моделирования сеть из 20 прогонов выдала 11 хороших (квазиоптимальных) решения и около 50% решений оказались одним из двух кратчайших маршрутов.

Одной из проблем при решении Хопфилдом ЗК было отсутствие систематического метода определения параметров K_1, K_2, K_3, K_4, от которых в значительной степени зависит сходимость решения. В этой связи позднее Боутом и Миллером была предложена другая энергетическая функция для ЗК с единственным параметром, который легко определяется.

Опыт решения ЗК с помощью сети Хопфилда показал, что поверхность энергии для таких задач очень сильно изрезана, поэтому нет никакой гарантии нахождения глобального оптимального решения. Вместе с тем, нахождение глобального минимума для *NP*-полных задач другими методами вообще проблематично и если и возможно, то занимает существенно больше времени и

не дает лучших результатов. Поэтому применение нейронных сетей в этом случае можно считать вполне оправданным. Для аналоговой реализации сети время решения занимает всего несколько постоянных времени сети. Более того, время решения мало зависит от размерности задачи, что резко отличает этот метод от других численных методов.

При решении других оптимизационных задач с помощью сети Хопфилда определение функции энергии для каждой конкретной задачи не является тривиальным и зависит, как правило, от математических способностей, изобретательности и таланта исполнителя.

Другие типы рекуррентных сетей. Рассмотренная выше сеть Хопфилда – не единственный представитель РС. Существуют РС, представляющие собой развитие однонаправленных многослойных сетей перцептронного типа, за счет введения в них соответствующих обратных связей. Обратные связи могут исходить из выходного или из скрытого слоя. В каждом контуре такой связи есть элемент единичной задержки, благодаря которому поток сигналов может считаться однонаправленным (выходной сигнал предыдущего временного цикла рассматривается как априори заданный, который просто увеличивает размерность входного вектора сети). Алгоритм обучения таких сетей оказывается более сложным, чем классических однонаправленных сетей. Наиболее известными из таких рекуррентных сетей являются [15]:

- «Рекуррентный многослойный перцептрон» (РМП) (англ.: Recurrent MultiLayer Perceptron – RMLP);
- «Рекуррентная сеть реального времени» (РСРВ) Вильямса-Зипсера (англ.: Real Time Recurrent Network – RTRN);
- Рекуррентная сеть Эльмана.

12. Двунаправленная ассоциативная память

Общие сведения. Память человека демонстрирует широкие ассоциативные возможности. Если наше сознание фокусируется на каком-либо образе, то он может вызвать целую гамму различных ассоциаций. В памяти всплывают множество других образов, связанных с первым, которые, в свою очередь, напоминают о чем-то третьем и т.д. Создаются многочисленные разветвленные цепочки умственных ассоциаций. Кроме того, такие ассоциации позволяют восстанавливать забытые образы. Распространенный пример: Вы что-то делали в комнате, и Вам для продолжения работы понадобился определенный предмет, за которым Вы пошли в другую комнату. Но по пути Вас что-то отвлекло и, придя в другую комнату, Вы совершенно забыли, зачем сюда пришли. Для того чтобы восстановить «утраченный» образ необходимо мысленно отследить всю возникшую цепочку ассоциаций, дойдя в итоге, до причины, вызвавшей необходимость посещения другой комнаты.

Рассмотренная выше классическая сеть Хопфилда моделирует, строго говоря, лишь *автоассоциативную* память. В ответ на предъявленный искаженный или неполный образ, сеть по ассоциации восстанавливает этот же образ, запомненный в ней ранее в качестве образца. Такая сеть не может ассоциировать один образ с другим, поскольку имеет лишь одноуровневую ассоциативную структуру, когда выходной образ формируется в тех же нейронах, на которые поступил входной образ.

Используя идею Хопфилда, в 1987 г. Б. Коско предложил построить двухзвенную ассоциативную память, которая сохраняет пары образцов и может восстановить один образец, когда ассоциированный с ним другой образец предлагается ей в качестве подсказки. Как и в сети Хопфилда, сеть способна выдавать правильные реакции на зашумленные входные образы. Нейронная сеть с такими свойствами является *гетероассоциативной*, а в литературе по ИНС называется двунаправленной ассоциативной памятью (ДАП) (Bidirectional Associative Memory – BAM).

Структура сети. На рис. 12.1 приведена базовая конфигурация ДАП [19]. В общем случае размерности слоя 1 и 2 могут быть различными. Входной вектор $\mathbf{A} = (a_1, a_2, \ldots, a_m)$ обрабатывается матрицей весов **W** сети, в результате чего вырабатывается выходной вектор $\mathbf{B} = (b_1, b_2, \ldots, b_n)$. Вектор **B** затем обрабатывается транспонированной матрицей $\mathbf{W}^T$ весов сети, которая вырабатывает новые выходные сигналы, представляющие собой новый входной вектор **A**. Этот процесс повторяется до тех пор, пока сеть не достигнет стабильного состояния, в котором ни вектор **A**, ни вектор **B** не изменяются.

Нейроны в слоях 1 и 2 функционируют, как и в других сетях, вычисляя сумму взвешенных входных сигналов и по ней – значение функции активации *F*. Этот процесс может быть выражен следующим образом:

$$b_i = F[\sum_{j=1}^{m} a_j \cdot w_{ij}] \qquad (12.1)$$

$$a_j = F[\sum_{i=1}^{n} b_i \cdot w_{ji}] \qquad (12.2)$$

или в векторной форме:

$$\mathbf{B} = F\,[\mathbf{A}\cdot\mathbf{W}], \qquad (12.3)$$

$$\mathbf{A} = F\,[\mathbf{B}\cdot\mathbf{W}^{\mathrm{T}}], \qquad (12.4)$$

где **B** – вектор выходных сигналов слоя 2;
A – вектор выходных сигналов слоя 1;
W – матрица весов связей между слоями 1 и 2;
$\mathbf{W}^{\mathrm{T}}$ – транспонированная матрица весов связей между слоями 2 и 1;
F – активационная функция.

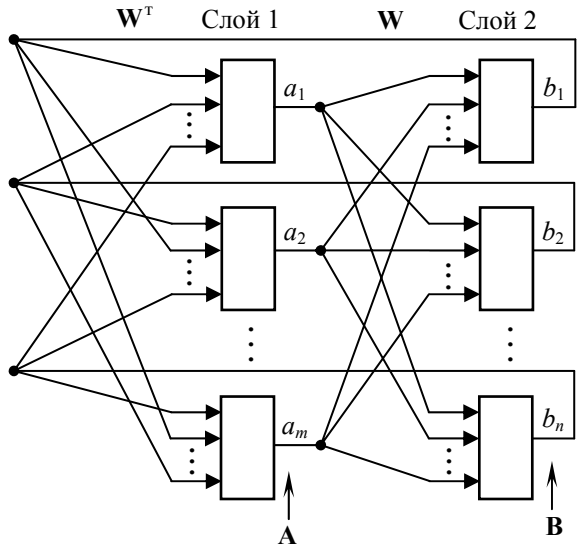


Рис. 12.1. Базовая конфигурация ДАП

Каждый слой сети функционирует также как и сеть Хопфилда, но в отличие от нее на вид матриц **W** и $\mathbf{W}^{\mathrm{T}}$ снимаются ограничения, связанные с

устойчивостью, т.е. они не обязаны быть квадратными и симметричными с нулями на главных диагоналях. Это объясняется отношением транспонирования между двумя матрицами. Т.е. ДАП остается устойчивой при любых значениях ее весовых матриц. Более того, необходимо, чтобы матрица **W** была несимметричной. Если в ДАП матрица **W** будет квадратной и симметричной, то $\mathbf{W} = \mathbf{W}^{\mathrm{T}}$, оба слоя становятся идентичными, и ДАП превращается в автоассоциативную сеть Хопфилда.

Так же как и в сети Хопфилда, векторы **A** и **B** обычно представляются двоичными компонентами с биполярным кодированием на основе signum-функции:

$$z_j = F(y_j) = \operatorname{sgn} y_j = \begin{cases} +1, \text{ если } y_j > Q_j; \\ -1, \text{ если } y_j < Q_j, \\ z_j, \text{ если } y_j = Q_j. \end{cases} \qquad (12.5)$$

Возможно также использование непрерывной сигмоидальной функции

$$z(x) = F[y(x)] = \frac{1}{1 + e^{-\lambda y(x)}}, \qquad (12.6)$$

где λ – коэффициент, определяющий крутизну сигмоидальной функции.

В простейших версиях ДАП значение константы λ обычно выбирается большим, в результате чего активационная функция (12.6) приближается к пороговой (12.5). Поэтому далее будем предполагать, что в качестве F используется пороговая функция (12.5). Примем также, что существует память внутри каждого нейрона в слоях 1 и 2, и что выходные сигналы нейронов изменяются одновременно с каждым тактом синхронизации, оставаясь постоянными между этими тактами.

Функционирование. ДАП функционирует следующим образом [19]. Для восстановления ассоциированного с образом **A** образа **B**, вектор **A** или его часть кратковременно устанавливаются на выходах слоя 1. Затем вектор **A** удаляется, и сеть приводится в стабильное состояние, вырабатывая ассоциированный вектор **B** на выходе слоя 2. Затем вектор **B** воздействует через транспонированную матрицу $\mathbf{W}^{\mathrm{T}}$, воспроизводя воздействие исходного входного вектора **A** на выходе слоя 1. Каждый такой цикл вызывает уточнение выходных векторов слоя 1 и 2 до тех пор, пока не будет достигнута стабилизация сети. Последовательность циклов образует переходной процесс двунаправленной обработки сигналов:

$$F(\mathbf{A}_0 \cdot \mathbf{W}) = \mathbf{B}_1 \to F(\mathbf{B}_1 \cdot \mathbf{W}^{\mathrm{T}}) = \mathbf{A}_1 \to F(\mathbf{A}_1 \cdot \mathbf{W}) = \mathbf{B}_2 \to F(\mathbf{B}_2 \cdot \mathbf{W}^{\mathrm{T}}) =$$
$$= \mathbf{A}_2 \to F(\mathbf{A}_2 \cdot \mathbf{W}) = \mathbf{B}_3 \to \ldots \to F(\mathbf{B}_F \cdot \mathbf{W}^{\mathrm{T}}) = \mathbf{A}_F \to F(\mathbf{A}_F \cdot \mathbf{W}) = \mathbf{B}_F.$$

ДАП функционирует в направлении минимизации функции энергии Ляпунова подобно сети Хопфилда. Каждый цикл модифицирует систему в направлении энергетического минимума, расположение которого определяется значениями весов. Каждой промежуточной точке процесса соответствует энергетическая функция

$$E_k = -\mathbf{A}_k \cdot \mathbf{W} \cdot \mathbf{B}_k^{\mathrm{T}} .$$

Б. Коско доказал, что каждое очередное изменение состояния переходного процесса ведет к уменьшению значения энергетической функции вплоть до достижения локального минимума. Этот минимум достигается за конечное число итераций, принимая значение

$$E_{\min} = -\mathbf{A}_F \cdot \mathbf{W} \cdot \mathbf{B}_F^{\mathrm{T}} .$$

Состояние нейронов ДАП можно трактовать как кратковременную память, так как оно может быстро изменяться при появлении другого входного вектора. Значения коэффициентов весовых матриц, напротив, образуют долговременную память, поскольку могут изменяться только на более длительном отрезке времени, связанном со сменой запоминаемых образцов (кодирование ассоциаций).

Кодирование ассоциаций. Как и сеть Хопфилда, ДАП способна к запоминанию множества образов (образцов). Обучение производится с использованием обучающего набора, состоящего из L пар векторов **A** и **B**. Процесс обучения реализуется в форме вычислений. Это означает, что весовая матрица формируется как сумма произведений всех векторных пар обучающего набора:

$$\mathbf{W} = \sum_{i}^{L} \mathbf{A}_i^{\mathrm{T}} \cdot \mathbf{B}_i , \qquad (12.7)$$

где L – число обучающих пар.

Пусть, в качестве примера, в сети необходимо запомнить три пары образов (L=3) [19]. Для простоты положим, что все образы представляют собой двоичные векторы с биполярным кодированием (12.5), причем векторы $\mathbf{A}_i$ и $\mathbf{B}_i$ имеют одинаковую размерность, равную трем битам (в общем случае это не обязательно).

Исходные данные примера приведены в табл. 12.1.

Исходные данные примера для запоминания в ДАП

Таблица 12.1

Исходный вектор	Ассоциированный вектор	Биполярная версия	
$\mathbf{A}_1 = 001$ (1)	$\mathbf{B}_1 = 101$ (5)	$\mathbf{A}_1' = -1\,-1\;\;1$	$\mathbf{B}_1' = 1\,-1\;\;1$
$\mathbf{A}_2 = 010$ (2)	$\mathbf{B}_2 = 110$ (6)	$\mathbf{A}_2' = -1\;\;1\,-1$	$\mathbf{B}_2' = 1\;\;1\,-1$
$\mathbf{A}_3 = 011$ (3)	$\mathbf{B}_3 = 111$ (7)	$\mathbf{A}_3' = -1\;\;1\;\;1$	$\mathbf{B}_3' = 1\;\;1\;\;1$

Вычисляем весовую матрицу

$$\mathbf{W} = \left(\mathbf{A}_1'\right)^{\mathrm{T}} \cdot \mathbf{B}_1' + \left(\mathbf{A}_2'\right)^{\mathrm{T}} \cdot \mathbf{B}_2' + \left(\mathbf{A}_3'\right)^{\mathrm{T}} \cdot \mathbf{B}_3'.$$

$$\mathbf{W} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & -1 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -3 & -1 & -1 \\ 1 & 3 & -1 \\ 1 & -1 & 3 \end{bmatrix}.$$

Приложим первый входной вектор $\mathbf{A}_1 = 001$ и вычислим выходной вектор $\mathbf{B}_1$ по правилу:

$$\mathbf{B}_1 = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} = \mathbf{A}_1^{\mathrm{T}} \cdot \mathbf{W} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \cdot \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} = \begin{bmatrix} b_1 = a_1 w_{11} + & a_2 w_{21} + & a_3 w_{31} \\ b_2 = a_1 w_{12} + & a_2 w_{22} + & a_3 w_{32} \\ b_3 = a_1 w_{13} + & a_2 w_{23} + & a_3 w_{33} \end{bmatrix}.$$

Подставляя численные биполярные значения, получим

$$\mathbf{B}_1' = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} -3 & -1 & -1 \\ 1 & 3 & -1 \\ 1 & -1 & 3 \end{bmatrix} = \begin{bmatrix} b_1 = 3 & -1 & 1 \\ b_2 = 1 & -3 & -1 \\ b_3 = 1 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & -3 & 5 \end{bmatrix}.$$

Используя пороговое правило signum-функции

$$b_i = 1, \text{ если } b_i > 0,$$
$$b_i = 0, \text{ если } b_i < 0,$$
$$b_i \text{ не изменяется, если } b_i = 0,$$

получаем $\mathbf{B}_1 = 101$, что является требуемой ассоциацией.

Затем, подавая вектор $\mathbf{B}_1$ во вновь образованной биполярной форме через обратную связь на вход первого слоя к матрице $\mathbf{W}^{\mathrm{T}}$, вычисляем вектор $\mathbf{A}_1$ по правилу:

$$\mathbf{A}_1^{\mathrm{T}} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \mathbf{B}_1 \cdot \mathbf{W}^{\mathrm{T}} = [b_1\ b_2\ b_3] \cdot \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \\ w_{13} & w_{23} & w_{33} \end{bmatrix} = \begin{bmatrix} a_1 = b_1 w_{11} + & b_2 w_{12} + & b_3 w_{13} \\ a_2 = b_1 w_{21} + & b_2 w_{22} + & b_3 w_{23} \\ a_3 = b_1 w_{31} + & b_2 w_{32} + & b_3 w_{33} \end{bmatrix}.$$

$$\mathbf{A}_1^{\mathrm{T}} = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -3 & 1 & 1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix} = \begin{bmatrix} a_1 = -3 & 1 & -1 \\ a_2 = 1 & -3 & -1 \\ a_3 = 1 & -1 & 3 \end{bmatrix} = \begin{bmatrix} -3 \\ -3 \\ 3 \end{bmatrix}.$$

После применения порогового правила получим $\mathbf{A}_1 = 001$.

Аналогично можно получить ассоциации для двух остальных пар векторов $\mathbf{A}_2$, $\mathbf{B}_2$ и $\mathbf{A}_3$, $\mathbf{B}_3$. Этот пример демонстрирует, как входной вектор $\mathbf{A}$ с использованием матрицы $\mathbf{W}$ производит выходной вектор $\mathbf{B}$, который, в свою очередь, с использованием матрицы $\mathbf{W}^{\mathrm{T}}$ производит вектор $\mathbf{A}$.

ДАП обладает способностью к обобщению. Например, если незавершенный или частично искаженный вектор подается в качестве $\mathbf{A}$, сеть имеет тенденцию к выработке запомненного вектора $\mathbf{B}$, который в свою очередь стремится исправить ошибки в $\mathbf{A}$. Для этого может потребоваться несколько проходов, но сеть сходится к воспроизведению ближайшего запомненного образа.

Емкость памяти. Как и сети Хопфилда, ДАП имеет ограничения на максимальное количество ассоциаций, которые она может точно воспроизвести. Если этот лимит превышен, сеть может выработать неверный выходной сигнал, воспроизводя ассоциации, которым сеть не обучена (ложные образы – фантомы).

Существуют разные оценки числа L запоминаемых в ДАП ассоциаций [16].

В соответствии с данными Б. Коско L не может превышать количества нейронов в меньшем слое:

$$L < \sqrt{\min(n, m)}\ .$$

Эта оценка справедлива при условии применения специального кодирования, при котором количество компонент со значениями +1 равно количеству компонент со значениями –1 в каждом биполярном векторе. Емкость памяти в этом случае оказывается максимальной. Такую оценку можно считать наиболее оптимистичной.

Данные другой работы, посвященной оценке емкости сетей Хопфилда, расширенные Ф. Уоссерменом для ДАП, приводят к гораздо более пессимистической оценке. Для случайно выбранных L векторов, представленных в указанной выше форме, $L < n/2\log_2 n$. Например, при n=1024, L < 51. При условии гарантированного восстановления оценка еще хуже: $L < n/4\log_2 n$, что для того же примера дает $L < 25$.

Хейнс и Хехт-Нильсен приводят описание так называемой *негомогенной* ДАП, в которой в отличие от классической *гомогенной* ДАП пороги нейронов вместо нулевых подбираются строго индивидуально. Ими показано, что такая ДАП может иметь до 2^n стабильных состояний. К сожалению, эти состояния не могут быть выбраны случайно, и определяются жесткой геометрической процедурой. Если выбор L состояний осуществляется случайным образом, причем L меньше $0{,}68n^2/(\log_2 n+4)^2$, и если каждый вектор имеет $\log_2 n+4$ компонент, равных +1, и остальные, равные –1, то можно сконструировать негомогенную ДАП, имеющую 98% этих векторов в качестве стабильных состояний. Например, если n = 1024, L должно быть меньше 3637, что является существенным улучшением по сравнению с гомогенными ДАП, но намного меньше 2^{1024} возможных состояний.

Непрерывная асинхронная ДАП. В описанной выше ДАП нейроны в слоях 1 и 2 функционировали синхронно, т.е. изменяли состояния одновременно под воздействием тактирующих импульсов. Кроме того, в качестве активационной функции нейронов использовался простой порог, образующий разрывность передаточной функции нейронов. Синхронность функционирования и разрывность передаточных функций делают ДАП биологически неправдоподобной.

Более близка к биологическому прототипу *непрерывная асинхронная* ДАП [16], в которой любой нейрон может изменять состояние в любое время, когда его входные сигналы предписывают это сделать, а в качестве активационной функции нейронов используется гладкая сигмоидальная функция (обычно с величиной λ близкой к 1). Непрерывные асинхронные ДАП также являются стабильными, однако имеют те же ограничения на емкость памяти. Они могут быть реализованы в виде аналоговых схем из резисторов и усилителей, а также с помощью оптических средств.

Адаптивная ДАП. В версиях ДАП, рассмотренных выше, весовая матрица вычисляется в виде суммы произведений пар векторов. Этот способ также отдаляет эти версии ДАП от нейронных сетей живого мозга. *Адаптивная* ДАП [16] изменяет свои веса в процессе функционирования. Это означает, что

подача на вход сети обучающего набора входных векторов заставляет ее изменять энергетическое состояние до получения резонанса. Постепенно кратковременная память превращается в долговременную память, настраивая сеть в результате ее функционирования. Один или оба вектора, подаваемые на адаптивную ДАП, могут быть зашумленными версиями эталона, и сеть обучается векторам, свободным от шума. Так как непрерывная ДАП является стабильной независимо от значения весов, то и медленное изменение ее весов не нарушает этой стабильности. Б. Коско доказал стабильность и адаптивной ДАП.

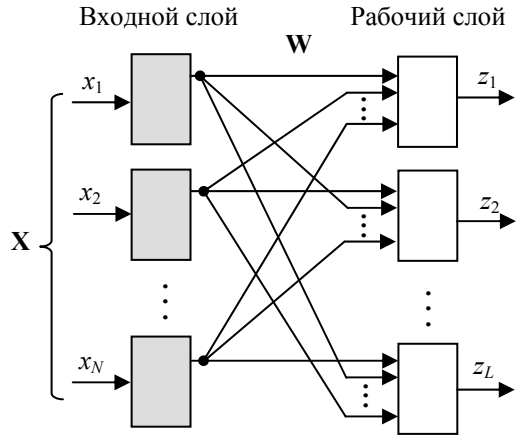
Конкурирующая ДАП. Во многих конкурирующих нейронных системах наблюдаются некоторые виды конкуренции между нейронами. В нейронах, обрабатывающих сигналы от сетчатки, латеральное торможение приводит к увеличению выходных сигналов наиболее высокоактивных нейронов за счет соседних. Такие системы увеличивают контрастность, поднимая уровень активности нейронов, подсоединенных к яркой области сетчатки, в то же время еще более ослабляя выходы нейронов, подсоединенных к темным областям.

ДАП, в которой вводится конкуренция между нейронами внутри каждого слоя, называется *конкурирующей* [16]. Конкуренция реализуется с помощью дополнительных соединения нейронов внутри каждого слоя по типу латерального торможения. Веса этих связей формируют другую весовую матрицу с положительными значениями элементов главной диагонали и отрицательными значениями остальных элементов. Теорема Кохонена-Гроссберга показывает, что такая сеть является безусловно стабильной, если весовые матрицы симметричны.

13. Нейронная сеть Хэмминга

Нейронная сеть Хэмминга была предложена Р.П. Липпманом в 1987 г. Ее можно рассматривать как развитие сети Хопфилда [20]. Основная идея функционирования этой сети состоит в параллельном вычислении расстояния Хэмминга между предъявляемым на входы сети неизвестным образцом и учебными образцами, закодированными в структуре сети.

Однослойная сеть Хэмминга. В своем простейшем варианте ИНС Хэмминга реализуется в виде однослойной структуры (рис. 13.1) [21].



13.1. Структура однослойной ИНС Хэмминга

Входной слой вычислений не выполняет и служит для приема и распределения входных сигналов. Он содержит N нейронов, по числу компонент входного сигнала $\mathbf{X} = (x_1, x_2, \ldots, x_N)$, $i = \overline{1, N}$.

В сети Хэмминга, как и в сети Хопфилда, входные сигналы $\mathbf{X} = (x_1, x_2, \ldots, x_N)$, $i = \overline{1, N}$ могут быть только бинарными векторами, которые удобнее представлять с помощью биполярного кодирования двоичных компонент: $x_i = +1$ («включено»); $x_i = -1$ («выключено»).

Рабочий слой сети содержит L нейронов, по числу закодированных в структуре сети учебных образцов $\mathbf{X}_j$, $j = \overline{1, L}$, и вычисляет расстояния Хэмминга (число несовпадающих бит) между предъявленным вектором $\mathbf{X}$ и каждым из L учебных образцов. Выходными сигналами рабочего слоя являются целые числа $z_1, z_2, \ldots, z_L$, показывающие расстояния Хэмминга для соответствующих учебных образцов $j = \overline{1, L}$.

Матрица весов **W** рабочего слоя сети формируется на основе предъявленных обучающих данных следующим образом:

$$\mathbf{W} = \begin{Vmatrix} w_{11} & w_{12} & \dots & w_{1L} \\ w_{21} & w_{22} & \dots & w_{2L} \\ \dots & \dots & \dots & \dots \\ w_{N1} & w_{N2} & \dots & w_{NL} \end{Vmatrix} = \begin{Vmatrix} -x_{11} & -x_{12} & \dots & -x_{1L} \\ -x_{21} & -x_{22} & \dots & -x_{2L} \\ \dots & \dots & \dots & \dots \\ -x_{N1} & -x_{N2} & \dots & -x_{NL} \end{Vmatrix}, \qquad (13.1)$$

$$i = \overline{1, N}\ ,\ \ j = \overline{1, L}\ .$$

Т.е. $w_{ij} = -x_{ij}$.

Нейроны рабочего слоя производят вычисления по обычной формуле

$$z_j = F(y_j) = F(\sum_{i=1}^{N} w_{ij} \cdot x_i - Q_j)\ ,\ i = \overline{1, N}\ ,\ \ j = \overline{1, L}\ , \qquad (13.2)$$

где Q_j – порог нейрона;

$F(y_j)$ – активационная функция нейрона.

Для того чтобы нейроны рабочего слоя могли реализовать вычисление расстояния Хэмминга $E(\mathbf{X}_j, \mathbf{X})$ между предъявленным вектором **X** и каждым из L учебных образцов X_j, $j = \overline{1, L}$, для них устанавливаются параметры: $w_{ij} = -x_{ij}$, $Q_j = -N$, а активационная функция $F(y_j)$ выбирается линейная с насыщением:

$$F(y_j) = \begin{cases} 0, & \text{если } y_j \le 0; \\ y_j, & \text{если } 0 < y_j \le N; \\ N, & \text{если } y_j > N. \end{cases}$$

График активационной функции нейронов рабочего слоя имеет вид, показанный на рис. 13.2.

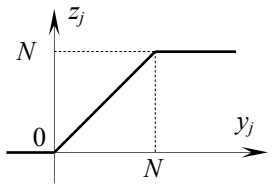
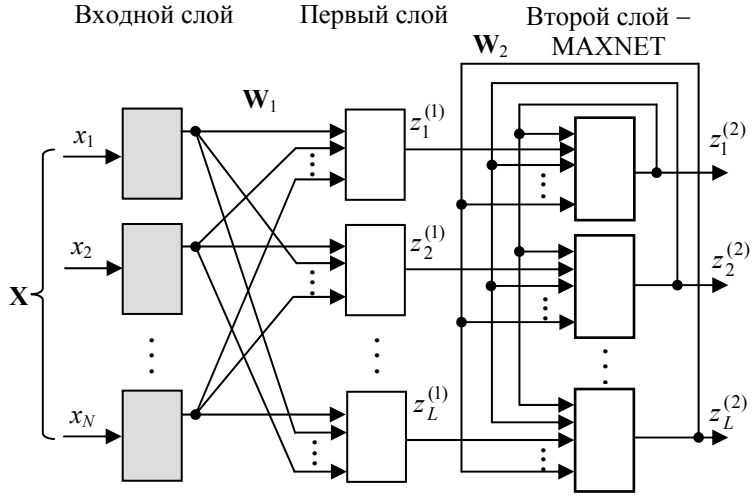


Рис. 13.2. График активационной функции нейронов рабочего слоя

В литературе часто рассматривается модель, в которой параметры нейронов рабочего слоя выбираются равными: $w_{ij} = -x_{ij}/2$, $Q_j = -N/2$.

Двухслойная сеть Хэмминга. В том случае, если среди выходных сигналов z_1, z_2, …, z_L рабочего слоя ИНС Хэмминга средствами нейронного моделирования необходимо выделить минимальное значение расстояния Хэмминга $E_{\min}(\mathbf{X}_j, \mathbf{X})$, $j = \overline{1, L}$, определяющее соответствующий класс неизвестного входного сигнала $\mathbf{X}$, сеть дополняют вторым рабочим слоем, называемым MAXNET (рис. 13.3) [22].



13.3. Структура двухслойной ИНС Хэмминга

В отличие от однослойной сети Хэмминга, первый рабочий слой в двухслойной сети выполняет инверсную операцию, т.е. минимальному расстоянию Хэмминга $E_{\min}(\mathbf{X}_j, \mathbf{X})$, $j = \overline{1, L}$ в первом слое соответствует максимальный выходной сигнал $z^1_{j\max}$. Для этого в первом слое устанавливаются параметры: $w_{ij} = x_{ij}$, $Q_j = 0$ или, чаще, $w_{ij} = x_{ij}/2$, $Q_j = -N/2$.

Второй слой – MAXNET, как и первый слой, содержит L нейронов, по числу закодированных в структуре сети учебных образцов $\mathbf{X}_j$, $j = \overline{1, L}$. Все нейроны второго слоя связаны между собой обратными связями по принципу «каждый с каждым». Причем, в отличие от сети Хопфилда, все нейроны второго слоя имеют ненулевые обратные связи на самих себя. Веса всех связей постоянны, причем разноименные нейроны связаны отрицательной (тормозной) обратной связью с весом $-\varepsilon$, а на самих себя нейроны имеют положительную (возбуждающую) обратную связь с весом, равным +1.

Матрица весов $\mathbf{W}_2$ слоя MAXNET имеет вид

$$\mathbf{W}_2 = \begin{Vmatrix} w_{11} & w_{12} & \dots & w_{1L} \\ w_{21} & w_{22} & \dots & w_{2L} \\ \dots & \dots & \dots & \dots \\ w_{L1} & w_{L2} & \dots & w_{LL} \end{Vmatrix} = \begin{Vmatrix} 1 & -\varepsilon & \dots & -\varepsilon \\ -\varepsilon & 1 & \dots & -\varepsilon \\ \dots & \dots & \dots & \dots \\ -\varepsilon & -\varepsilon & \dots & 1 \end{Vmatrix}, \quad j,k = \overline{1,L}. \qquad (13.3)$$

Слой MAXNET функционирует в режиме WTA (Winner Takes All – Победитель получает все), когда в установившемся режиме в слое остается активированным только один нейрон, а остальные находятся в невозбужденном состоянии. Отличное от нуля значение выходного сигнала единственного активированного нейрона $z_k^{(2)}$, указывает на принадлежность входного образа к соответствующему классу $k = \overline{1,L}$ учебных образцов.

Сеть имеет две фазы функционирования. В первой фазе на входы сети подается неизвестный входной вектор $\mathbf{X} = (x_1, x_2, \dots, x_N)$, $i = \overline{1,N}$. На выходах первого рабочего слоя формируются выходные сигналы

$$z_j^{(1)} = F(y_j) = F(\sum_{i=1}^{N} w_{ij} \cdot x_i - Q_j), \; i = \overline{1,N}, \; j = \overline{1,L}, \qquad (13.4)$$

которые задают начальные состояния нейронов второго слоя – MAXNET.

Во второй фазе инициировавшие слой MAXNET сигналы удаляются, и из сформированного ими начального состояния запускается итерационный процесс внутри второго слоя. Итерационный процесс завершается в момент, когда все нейроны, кроме одного (победителя с выходным сигналом, отличным от нуля), перейдут в нулевое состояние. Нейрон-победитель с ненулевым выходным сигналом становится представителем класса данных, которому принадлежит входной вектор **X**.

Процесс определения нейрона-победителя – это рекуррентный процесс, выполняемый в соответствии с выражением

$$z_k^{(2)}(p) = F(y_k) = [\sum_{j=1}^{L} w_{jk} \cdot z_j^{(2)}(p-1)], \quad j,k = \overline{1,L}, \qquad (13.5)$$

при начальном значении $z_j^{(2)}(0) = z_j^{(1)}$.

Активационная функция $F(y_k)$, как и для первого слоя, выбирается линейная с насыщением:

$$F(y_k) = \begin{cases} 0, \text{ если } y_k \le 0; \\ y_j, \text{ если } 0 < y_j \le A; \\ A, \text{ если } y_j > A, \end{cases}$$

причем величина A должна быть достаточно большой, чтобы любые возможные значения аргумента не приводили к насыщению.

График активационной функции нейронов слоя MAXNET имеет вид, показанный на рис. 13.4.

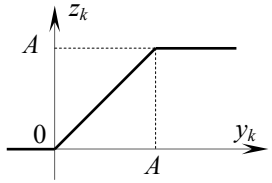


Рис. 13.4. График активационной функции нейронов слоя MAXNET

Учитывая, что веса второго слоя выбираются, исходя из условия:

$$w_{jk} = 1, \quad \text{при } j=k;$$
$$w_{jk} = -\varepsilon, \text{ при } j \neq k,$$

выражение (13.5) можно представить в окончательном виде

$$z_k^{(2)}(p) = F[z_k^{(2)}(p-1) + \varepsilon \sum_{\substack{j=1 \\ j \neq k}}^{L} z_j^{(2)}(p-1)], \quad j,k = \overline{1,L}$$

Значение ε весов тормозных связей w_{jk} ($j \neq k$) обычно выбираются в диапазоне

$$0 < \varepsilon < -\frac{1}{L-1}.$$

Для обеспечения абсолютной сходимости процесса в слое MAXNET веса тормозных связей должны отличаться друг от друга. Чтобы удовлетворить этому условию, Р.П. Липпманн предложил следующую формулу для расчета весов тормозных связей:

$$w_{jk} = -\frac{1}{L-1} + \xi,$$

где ξ - малая случайная величина.

Трехслойная сеть Хэмминга. Сеть Хэмминга может быть использована для построения гетероассоциативного запоминающего устройства, когда неизвестный входной вектор **X** вначале соотносится с одним из классов учебных образцов $\mathbf{X}_j$, $j = \overline{1, L}$, а затем ему в соответствие ставится заданный выходной вектор $\mathbf{Z}_j$. В этом случае двухслойная сеть Хэмминга (рис. 13.3) дополняется третьим однонаправленным выходным слоем (рис. 13.5) [15].

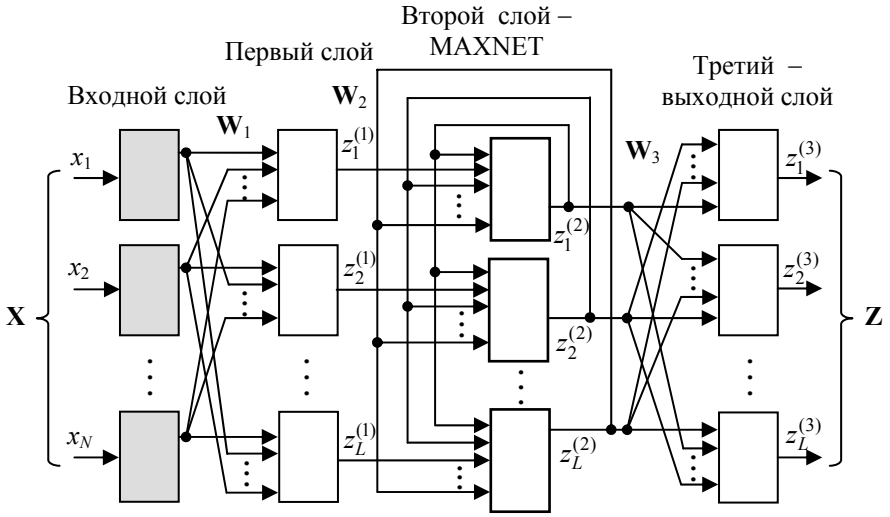


Рис. 13.5. Трехслойная ИНС Хэмминга

Матрица весов $\mathbf{W}_3$ третьего выходного слоя подбирается в зависимости от обучающих данных с таким расчетом, чтобы каждому учебному образцу (вектору) $\mathbf{X}_j$, $j = \overline{1, L}$ соответствовал желаемый выходной вектор сети $\mathbf{Z}_j$, $j = \overline{1, L}$.

В процессе функционирования трехслойной сети Хэмминга по сравнению с двухслойной появляется третья фаза. В этой фазе нейрон-победитель во втором слое посредством весов w_{ij}^3, связывающих его с нейронами третьего выходного слоя, формирует на выходе отклик сети в виде вектора $\mathbf{Z}_j$, соответствующий возбуждающему вектору **X**.

Преимущества и недостатки. Важным достоинством сети Хэмминга считается небольшое количество взвешенных связей между нейронами. Например, сеть Хопфилда с размерностью входного сигнала 100 может запомнить примерно 10 образцов, при этом она будет содержать 10000

взвешенных связей с подбираемыми значениями весов. Однослойная сеть Хэмминга с такой же емкостью будет содержать 1000 взвешенных связей, двухслойная – 1100 связей, из которых 100 связей – в слое MAXNET.

В отличие от сети Хопфилда, емкость сети Хэмминга не зависит от размерности входного сигнала и в точности равна числу нейронов первого рабочего слоя. Это позволяет легко рассчитать конфигурацию сети для заданного числа запоминаемых образцов.

Однослойная сеть Хэмминга работает существенно быстрее сети Хопфилда, поскольку решение задачи формируется в результате однократного прохода только через один слой нейронов.

Сеть Хэмминга имеет один из самых простых алгоритмов формирования весов и смещений.

В результате многочисленных экспериментов доказано, что двухслойная рекуррентная сеть Хэмминга дает лучшие результаты, чем сеть Хопфилда, особенно в ситуациях, когда взаимосвязанные векторы **X** и **Z** являются случайными.

Недостатком сети Хэмминга является слабая классифицирующая способность для сильно зашумленных входных сигналов. Если зашумленные входные сигналы находятся на одинаковом (в смысле Хэмминга) расстоянии от двух и более эталонов, то выбор сетью одного из этих эталонов становится совершенно случайным. Другим недостатком сети Хэмминга является использование только бинарных входных сигналов, что существенно ограничивает ее применение.

Области применения. Сеть Хэмминга может успешно использоваться для решения задач распознавания образов и классификации, для надежной передачи сигналов в условиях помех, в качестве ассоциативной и гетероассоциативной памяти.

14. Радиальные нейронные сети

Многослойные нейронные сети, преобразуя входной вектор **X** в выходной вектор **Z**, с математической точки зрения выполняют в многомерном пространстве *глобальную аппроксимацию* некоторой стохастической функции. Причем всякий раз преобразование этой функции осуществляется объединенными усилиями многих нейронов.

Существует другой способ построения нейронной сети для преобразования **X** в **Z**, который заключается в *локальной аппроксимации* нескольких базисных стохастических функций в ограниченной области многомерного пространства. Условно такие сети можно назвать нейронными сетями с локальной аппроксимацией.

Наибольший интерес представляет частный случай задания базисных функций φ, когда они радиально изменяются вокруг некоторого выбранного центра **C** и принимают ненулевые значения только в окрестности этого центра:

$$\varphi(x) = \varphi(\|\mathbf{X} - \mathbf{C}\|) .$$

ИНС с локальной аппроксимацией на основе радиальных базисных функций φ(x) составляют особое семейство и называются *радиальными нейронными сетями* (РНС) [15].

Главное отличие РНС от обычных многослойных сетей (многослойного перцептрона) состоит в функции нейронов рабочего (скрытого) слоя. В обычной многослойной сети каждый нейрон рабочего слоя реализует в многомерном пространстве гиперплоскость, которая разделяет это пространство на два подпространства (рис. 14.1 *а*):

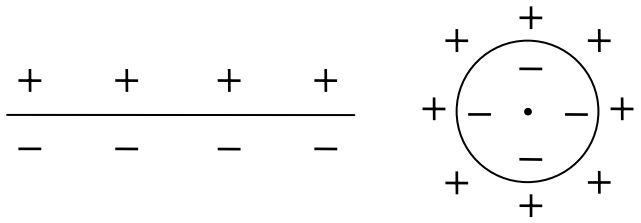
$$V_i = \sum_{j=1}^{N} w_{ij} x_j > 0, \qquad V_i = \sum_{j=1}^{N} w_{ij} x_j < 0 ,$$

где V_i – выходной сигнал i-нейрона.

В РНС каждый нейрон рабочего слоя, называемый *радиальным нейроном*, реализует в многомерном пространстве гиперсферу, которая разделяет пространство на два подпространства вокруг некоторой центральной точки (рис. 14.1 *б*):

$$\varphi_i = \varphi(|\mathbf{X}_i - \mathbf{C}_i|) > 0, \qquad \varphi_i = \varphi(|\mathbf{X}_i - \mathbf{C}_i|) < 0 .$$

Если решаемая задача имеет круговую симметрию данных, то использование РНС позволяет заметно уменьшить количество нейронов, необходимых для разделения различных классов.



а) *б)*

Рис. 14.1. Иллюстрация способов разделения пространства данных:
а) обычным нейроном; *б*) радиальным нейроном

Структура РНС. Обобщенная структура РНС показана на рис. 14.2 [15].

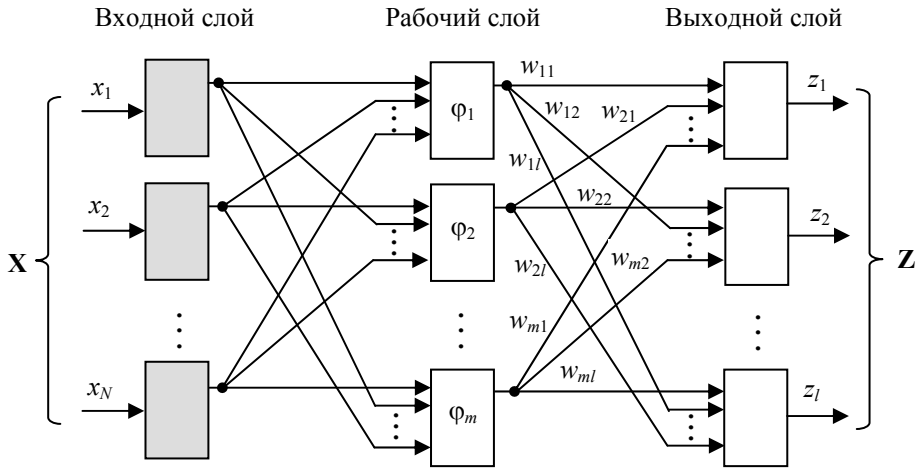


Рис. 14.2. Обобщенная структура РНС

Входной слой РНС как и в других сетях вычислительных функций не выполняет и служит для приема и распределения компонент *N*-мерного входного вектора **X**. В простейшем случае, когда входной многомерный вектор **X** имеет одинаковый масштаб по каждой оси, связи между входным и рабочим слоем не взвешиваются, т.е. каждый нейрон рабочего слоя получает один и тот же входной вектор **X**.

Рабочий слой РНС содержит *m* нейронов, каждый из которых реализует радиальную базисную функцию $\varphi(x)$. Нейроны рабочего слоя в общем случае могут выполнять различные базисные функции и иметь различные параметры,

поэтому в РНС отсутствует необходимость использования нескольких рабочих (скрытых) слоев.

В качестве радиальной функции чаще всего применяется функция Гаусса. При размещении ее центра в точке $\mathbf{C}_i$, она определяется в виде

$$\varphi(\mathbf{X}) = \exp\left(-\frac{\left\|\mathbf{X}_i - \mathbf{C}_i\right\|^2}{2\sigma_i^2}\right),$$

где σ_i - параметр, от значения которого зависит ширина функции.

Нейроны выходного слоя имеют линейную активационную функцию. Их роль сводится исключительно к взвешенному суммированию сигналов, генерируемых нейронами рабочего слоя. Число нейронов выходного слоя определяется характером представления выходных данных. В большинстве случаев в выходном слое достаточно иметь один или несколько нейронов.

Математическое обоснование функционирования РНС базируется на теореме Т. Ковера. Обозначим вектор радиальных функций в *N*-мерном входном пространстве

$$\mathbf{\Phi}(\mathbf{X}) = [\varphi_1(\mathbf{X}), \varphi_2(\mathbf{X}), \ldots, \varphi_m(\mathbf{X})]^{\mathrm{T}}.$$

Входное пространство является нелинейно **Φ**-разделяемым на два пространственных класса $\mathbf{X}^+$ и $\mathbf{X}^-$ тогда, когда существует такой вектор весов **W**, что

$$\mathbf{W}^{\mathrm{T}} \cdot \mathbf{\Phi}(\mathbf{X}) > 0 \text{ для } \mathbf{X} \in \mathbf{X}^+,$$
$$\mathbf{W}^{\mathrm{T}} \cdot \mathbf{\Phi}(\mathbf{X}) < 0 \text{ для } \mathbf{X} \in \mathbf{X}^-.$$

Граница между этими классами определяется уравнением

$$\mathbf{W}^{\mathrm{T}} \cdot \mathbf{\Phi}(\mathbf{X}) = 0.$$

Т. Ковером доказано, что каждое множество образов, случайным образом размещенных в многомерном пространстве, являются **Φ**-разделяемым с вероятностью 1 при условии соответственно большой размерности *m* этого пространства. На практике это означает, что применение достаточно большого количества нейронов для реализации функций $\varphi_i(\mathbf{X})$ гарантирует решение задачи классификации на двухслойной РНС, в которой рабочий слой должен реализовать вектор **Φ**(**X**), а выходной слой может состоять из единственного

линейного нейрона, выполняющего суммирование выходных сигналов нейронов рабочего слоя с весами, заданными вектором **W**.

Простейшая РНС реализует многомерную интерполяцию, состоящую в отображении L различных входных векторов $\mathbf{X}_i$, (i = 1, 2, …, L) N-мерного пространства в выходной вектор $\mathbf{Z}_i$, (i = 1, 2, …, L). Для реализации этого процесса число нейронов в рабочем слое необходимо выбрать равным L и задать такую функцию отображения $F(\mathbf{X})$, для которой выполняется условие интерполяции

$$F(\mathbf{X}_i) = z_i.$$

Рассмотрим РНС с L обучающими парами ($\mathbf{X}_i$, $\mathbf{Z}_i$) и примем, что координаты каждого из L центров узлов сети определяются одним из векторов $\mathbf{X}_i$, т.е. $\mathbf{C}_i = \mathbf{X}_i$. В этом случае взаимосвязь между входными и выходными сигналами сети может быть определена системой уравнений, линейных относительно весов $\mathbf{W}_i$, которая в матричной форме имеет вид

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \dots & \varphi_{1L} \\ \varphi_{21} & \varphi_{22} & \dots & \varphi_{2L} \\ \dots & \dots & \dots & \dots \\ \varphi_{L1} & \varphi_{L2} & \dots & \varphi_{LL} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_L \end{bmatrix}, \qquad (14.1)$$

где $\varphi_{ij} = (\|\mathbf{X}_j - \mathbf{X}_i\|)$ определяет радиальную функцию с центром в точке $\mathbf{X}_i$ с вынужденным вектором $\mathbf{X}_j$.

В редуцированной матричной форме выражение (14.1) будет иметь вид

$$\mathbf{\Phi} \cdot \mathbf{W} = \mathbf{Z}. \qquad (14.2)$$

Т. Ковером доказано, что для ряда радиальных функций, в случае

$$\mathbf{X}_1 \neq \mathbf{X}_2 \neq \dots, \neq \mathbf{X}_L,$$

существует решение уравнения (14.2) в виде

$$\mathbf{W} = \mathbf{\Phi}^{-1} \cdot \mathbf{Z},$$

что позволяет получить вектор весов выходного слоя.

На практике число обучающих векторов $\mathbf{X}_i$, (i = 1, 2, …, L) часто бывает очень большим, и при выборе равного ему числа базисных функций система с математической точки зрения становится бесконечной (плохо структурированной), поскольку количество описывающих ее уравнений

начинает превышать число степеней свободы физического процесса. Качество работы сети при этом снижается. Кроме того, существенно возрастает вычислительная сложность обучающего алгоритма. Поэтому в РНС ищется субоптимальное решение в пространстве меньшей размерности, которое, тем не менее, с достаточной точностью аппроксимирует точное решение.

Если ограничиться m базисными функциями ($m<L$). То аппроксимирующее решение можно представить в виде

$$F(\mathbf{X}) = \sum_{i=1}^{m} w_i \cdot \varphi(\|\mathbf{X} - \mathbf{C}_i\|)$$

Подбор параметров радиальных функций и значений весов сводится к минимизации целевой функции, которая при использовании метрики Эвклида записывается в форме

$$S = \sum_{i=1}^{L} [\sum_{j=1}^{m} w_j \cdot \varphi(\|\mathbf{X}_i - \mathbf{C}_i\|) - z_i]^2. \qquad (14.3)$$

Совокупность радиальных функций в (14.3) представляется в виде матрицы Грина

$$\mathbf{G} = \begin{bmatrix} \varphi(\|\mathbf{X}_1 - \mathbf{C}_1\|) & \varphi(\|\mathbf{X}_1 - \mathbf{C}_2\|) & \dots & \varphi(\|\mathbf{X}_1 - \mathbf{C}_m\|) \\ \varphi(\|\mathbf{X}_2 - \mathbf{C}_1\|) & \varphi(\|\mathbf{X}_2 - \mathbf{C}_2\|) & \dots & \varphi(\|\mathbf{X}_2 - \mathbf{C}_m\|) \\ \dots & \dots & \dots & \dots \\ \varphi(\|\mathbf{X}_L - \mathbf{C}_1\|) & \varphi(\|\mathbf{X}_L - \mathbf{C}_2\|) & \dots & \varphi(\|\mathbf{X}_L - \mathbf{C}_m\|) \end{bmatrix}.$$

Если параметры радиальных функций известны, то оптимизация (14.3) сводится к решению системы уравнений, линейных относительно весов:

$$\mathbf{G}(\mathbf{W}) = \mathbf{Z}. \qquad (14.4)$$

Вектор весов определяется в результате следующих матричных преобразований

$$\mathbf{W} = (\mathbf{G}^{\mathrm{T}} \cdot \mathbf{G})^{-1} \cdot \mathbf{G}^{\mathrm{T}} \cdot \mathbf{Z}. \qquad (14.5)$$

В том случае, если входной многомерный вектор $\mathbf{X}$ имеет различный масштаб по каждой оси, то необходимо ввести масштабирующие коэффициенты в виде матрицы $\mathbf{Q}$:

$$\mathbf{Q} = \begin{bmatrix} Q_{11} & Q_{12} & \dots & Q_{1N} \\ Q_{21} & Q_{22} & \dots & Q_{2N} \\ \dots & \dots & \dots & \dots \\ Q_{N1} & Q_{N2} & \dots & Q_{NN} \end{bmatrix}.$$

При этом

$$\|\mathbf{X}\|_Q^2 = (\mathbf{Q}\cdot\mathbf{X})^{\mathrm{T}}\cdot(\mathbf{Q}\cdot\mathbf{X}) = \mathbf{X}^{\mathrm{T}}\cdot\mathbf{Q}^{\mathrm{T}}\cdot\mathbf{Q}\cdot\mathbf{X}.$$

Если обозначить произведение матриц $\mathbf{Q}^{\mathrm{T}}\cdot\mathbf{Q}$ матрицей корреляции $\mathbf{K}$, то в общем случае получим

$$\|\mathbf{X}\|_Q^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} K_{ij}\cdot x_i\cdot x_j.$$

При использовании в качестве радиальной функции – функции Гаусса, с учетом масштабирования она будет иметь

$$\varphi(\mathbf{X}) = \varphi(\|\mathbf{X}_i - \mathbf{C}_i\|_{Qi}) = \exp[-(\mathbf{X}_i - \mathbf{C}_i)^{\mathrm{T}}\cdot\mathbf{Q}_i^{\mathrm{T}}\cdot\mathbf{Q}_i(\mathbf{X} - \mathbf{C}_i)] =$$

$$= -\exp[\frac{1}{2}(\mathbf{X} - \mathbf{C}_i)^{\mathrm{T}}\cdot\mathbf{K}_i(\mathbf{X} - \mathbf{C}_i)],$$

где матрица $\frac{1}{2}\mathbf{C}_i$ играет роль скалярного коэффициента $\frac{1}{2\sigma_i^2}$ стандартной многомерной функции Гаусса.

Обучение РНС. В том случае если $m=L$, то центры $\mathbf{C}_i$ радиальных функций известны заранее ($\mathbf{C}_i=\mathbf{X}_i$), и задача обучения РНС сводится к решению избыточной системы линейных уравнений (14.4). Значение параметра σ_i радиальных функций можно легко подобрать экспериментальным путем при соблюдении определенного компромисса между монотонностью и точностью отображения.

Если $m < L$, то вектор весов выходного слоя вычисляется за один шаг по формуле (14.5) и остается подобрать параметры радиальных функций. Одним из простейших способов, хотя и не самым эффективным, считается их случайный выбор параметров. При этом центры $\mathbf{C}_i$ базисных функций выбираются случайным образом на основе равномерного распределения. Такой подход

допустим, если равномерное распределение обучающих данных хорошо соответствует специфике задачи. Параметры стандартного отклонения в функции Гаусса в свою очередь задаются в соответствии с разбросом случайно выбранных центров $\mathbf{C}_i$.

Среди многих специализированных методов подбора центров наиболее применяемыми являются: самоорганизующийся процесс разделения на кластеры, гибридный алгоритм и обучение с учителем [15].

Процесс самоорганизации основан на известном алгоритме самообучения Кохонена и производных от него. Параметры базисных функций определяются после автоматического разбиения пространства на кластеры с последующим определением центров кластеров геометрическими методами.

Гибридный алгоритм обучения РНС содержит два этапа: подбор весов выходного слоя и подбор параметров $\mathbf{C}_i$ и σ_i базисных радиальных функций. Эти этапы циклически многократно чередуются, что позволяет достаточно быстро обучать РНС, особенно при задании начальных значений параметров, близких к оптимальным.

Алгоритмы обучения РНС с учителем составляют обособленный класс градиентных алгоритмов, в которых используется широко известная процедура обратного распространения ошибки. Как и в классическом алгоритме backpropagation основной проблемой, влияющей на качество обучения, является выбор начальных значений параметров. При случайном начальном выборе параметров высока вероятность попадания процесса в ловушки локальных минимумов. По этой причине случайный выбор часто заменяется специальной процедурой инициализации, основанной на анализе информации, содержащейся во множестве входных данных. Эта процедура обычно использует уже упомянутый выше процесс самоорганизации данных путем их кластеризации.

Сравнение радиальных и многослойных сетей. РНС отличаются от многослойных сетей с сигмоидальными функциями активации некоторыми специфическими свойствами. Многослойная сеть, в которой ненулевое значение сигмоидальной функции распространяется от некоторой точки в пространстве до бесконечности, решает задачу глобальной аппроксимации заданной функции. РНС, основанная на функциях, имеющих ненулевые значения только в определенной области вокруг их центров, реализует аппроксимацию локального типа, сфера которой, как правило, более ограничена. Поэтому обобщающие способности РНС несколько хуже, чем многослойных, особенно на границах области обучающих данных. Вследствие глобального характера сигмоидальной функции многослойные сети не обладают встроенным механизмом идентификации области данных, на которые сильнее всего реагирует конкретный нейрон. Из-за физической невозможности связать зону активности нейрона с соответствующей областью обучающих данных для многослойных сетей сложно определить исходную позицию процесса обучения. Принимая во внимание полимодальность целевой функции,

достижение глобального минимума в такой ситуации становится чрезвычайно трудным даже при самых совершенных методах обучения [15].

РНС решают эту проблему гораздо лучше. Наиболее часто применяемые на практике радиальные функции гауссовского типа по своей природе имеют локальный характер и принимают ненулевые значения только в зоне вокруг определенного центра. Это позволяет легко установить зависимость между параметрами базисных функций и физическим размещением обучающих данных в многомерном пространстве. Поэтому удается относительно просто найти удовлетворительные начальные условия процесса обучения с учителем. Применение подобных алгоритмов обучения при начальных условиях, близких к оптимальным, многократно увеличивает вероятность достижения успеха с помощью РНС.

Важное достоинство РНС – значительно упрощенный алгоритм обучения. При наличии только одного скрытого слоя и тесной связи активности нейрона с соответствующей областью пространства обучающих данных точка начала обучения оказывается гораздо ближе к оптимальному решению, чем в многослойных сетях. Кроме того, можно отделить этап подбора параметров базисных функций от подбора весов сети (гибридный алгоритм), что сильно упрощает и ускоряет процесс обучения. Выигрыш во времени становится еще более ощутимым, если принять во внимание процедуру формирования оптимальной структуры сети. Для многослойных сетей это очень трудоемкая задача, требующая, как правило, многократного повторения обучения и дообучения.

Считается, что РНС лучше, чем многослойные сети, решают задачи классификации и распознавания, в которых выражена локализация обучающих данных (определение повреждений в различных системах, идентификация биометрических данных и т.п.). Хорошие результаты получены при использовании РНС для задач прогнозирования временных процессов (колебания занятости трудоспособного населения, экономические тренды и т.п.).

Исходя из перечисленных особенностей организации, функционирования и обучения многослойных и радиальных сетей, следует, что выбор того или иного типа сети для решения конкретной задачи определяется, в первую очередь, особенностями распределения обучающих данных. При наличии выраженной локализации данных в определенных областях пространства, предпочтительнее использовать РНС. И, наоборот, при отсутствии выраженной локализации данных, а также в условиях отсутствия достоверных сведений о распределении данных, больших результатов можно ожидать от применения многослойных сетей с сигмоидальными функциями активации.

15. Вероятностные нейронные сети

Вероятностная нейронная сеть (ВНС или PNN – Probabilistic Neural Network) представляет собой параллельную реализацию статистических методов Байеса [19]. В ВНС образцы классифицируются на основе оценок их близости к соседним образцам. При этом используется ряд критериев статистических методов, на основе которых принимается решение о том к какому классу отнести еще не классифицированный образец. Формальным правилом при классификации является то, что класс с наиболее плотным распределением в области неизвестного образца, а также – с более высокой априорной вероятностью, а также – с более высокой ценой ошибки классификации, будет иметь преимущество по сравнению с другими классами. В соответствии с этим правилом для двух классов A и B выбирается класс A, если:

$$h_A \cdot c_A \cdot f_A(x) > h_B \cdot c_B \cdot f_B(x),$$

где h– априорная вероятность;

c – цена ошибки классификации;

$f(x)$ – функция плотности вероятностей.

Оценка стоимости ошибки классификации и априорной вероятности предполагает хорошее знание решаемой задачи и во многих приложениях выбираются одинаковыми.

Для оценки функции плотности распределения вероятностей применяют метод Парцена (Parzen), в соответствии с которым для каждого учебного образца рассматривается некоторая весовая функция, называемая *функцией потенциала* или *ядром*. Чаще всего в качестве ядра используется упрощенная форма *функции Гаусса*

$$\varphi(\mathbf{X}) = \exp\left(-\frac{\left\|\mathbf{X}-\mathbf{X}_i\right\|^2}{2\sigma^2}\right), \qquad (15.1)$$

где $\mathbf{X}_i$ – i-й образец вектора $\mathbf{X}_i$, $i = \overline{1, L}$;

$\mathbf{X}$ – неизвестный образец;

σ – параметр, задающий ширину функции и определяющий ее влияние.

Классическая функция Гаусса для нормального распределения имеет вид

$$\varphi(\mathbf{X}) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{\left\|\mathbf{X}-\mathbf{X}_i\right\|^2}{2\sigma^2}\right). \qquad (15.2)$$

Используемая функция (15.1) отличается от классической (15.2) отсутствием коэффициента перед экспонентой. Это позволяет получить максимальное значение функции плотности распределения вероятностей (15.1), равное единице, а не величине указанного коэффициента.

Чтобы определить функцию плотности распределения вероятностей для всего k-класса, функции Гаусса для всех учебных векторов суммируются:

$$\varphi(\mathbf{X}) = \sum_{i=1}^{L_k} \exp\left(-\frac{\left\| \mathbf{X} - \mathbf{X}_i \right\|^2}{2\sigma^2} \right), \qquad (15.3)$$

где L_k – объем обучающей выборки k-класса.

Структура ВНС. Пример структуры РНС для решения простой задачи разделения 4-х компонентных входных векторов **X** на два класса показан на рис. 15.1 [19].

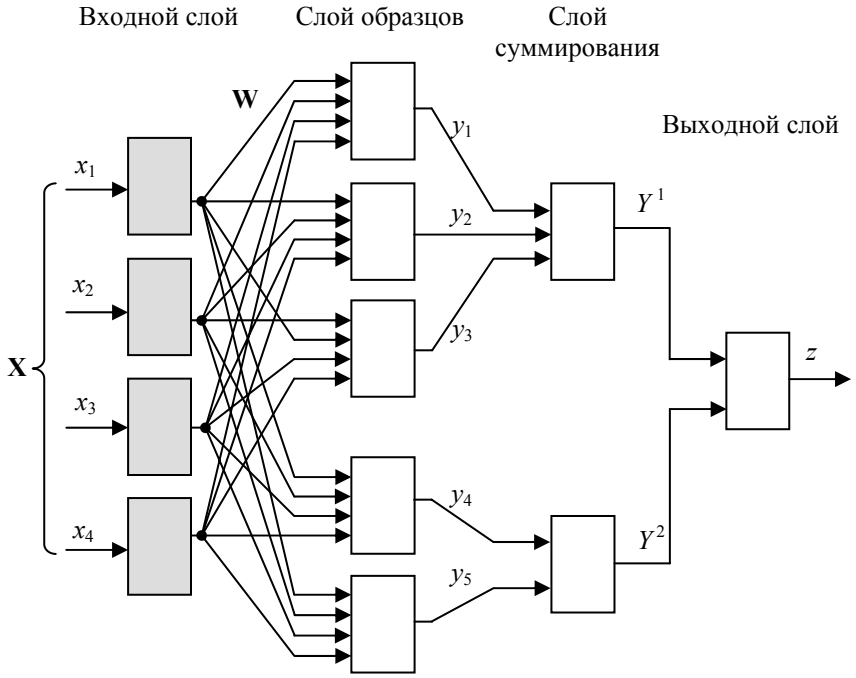


Рис. 15.1. Пример структуры ВНС для разделения 4-х компонентных входных векторов на два класса

Входной слой ВНС как и в других сетях вычислительных функций не выполняет и служит для приема и распределения компонент входного вектора **X**. Число нейронов входного слоя определяется числом компонент входного вектора **X**. Слой образцов содержит по одному нейрону для каждого образца входного вектора **X** из обучающей выборки. Т.е. при общем объеме обучающей выборки, равном L образцов, слой образцов должен иметь L нейронов. В приведенном на рис. 15.1 примере L=5, L_1=3, L_2=2, поэтому слой образцов имеет 5 нейронов: 3 для первого класса и 2 для второго класса. Входной слой и слой образцов образуют полносвязную структуру. Веса матрицы связей **W** слоя образцов определяются значениями компонент соответствующего образца входного вектора **X**. Для входных векторов **X**, содержащих N компонент и представленных L своими образцами, матрица весов **W** имеет вид

$$\mathbf{W} = \begin{Vmatrix} w_{11} & w_{12} & \dots & w_{1L} \\ w_{21} & w_{22} & \dots & w_{2L} \\ \dots & \dots & \dots & \dots \\ w_{N1} & w_{N2} & \dots & w_{NL} \end{Vmatrix} = \begin{Vmatrix} x_{11} & x_{12} & \dots & x_{1L} \\ x_{21} & x_{22} & \dots & x_{2L} \\ \dots & \dots & \dots & \dots \\ x_{N1} & x_{N2} & \dots & x_{NL} \end{Vmatrix}. \qquad (15.4)$$

Т.е. i-нейрон слоя образцов имеет набор из N весов, соответствующих N компонентам входного вектора, представленного своим i-образцом (i-столбец матрицы **W**).

Слой суммирования содержит число нейронов, равное числу классов, на которые разбиваются входные образы. Каждый нейрон слоя суммирования имеет связи только с нейронами слоя образцов, которые относятся к соответствующему классу. Все веса связей нейронов слоя суммирования равны 1.

Выходной нейрон выполняет функции дискриминатора пороговой величины. Он указывает, какой нейрон слоя суммирования имеет максимальный выходной сигнал. Тем самым определяется класс, к которому принадлежит предъявленный входной образ. Веса связей нейрона выходного слоя устанавливаются так, чтобы на его выходе идентифицировался нейрон слоя суммирования с наибольшим значением активности.

Функционирование сети. Принцип обучения ВНС несколько отличается от принципов обучения других типов сетей. Особенностью является то, что число нейронов в слое образцов определяется числом самих образцов, а следовательно, параметрами сети, определяемыми непосредственно в ходе обучения, являются и число нейронов в слое образцов, и значения их весов. Другими словами, в ходе обучения формируется сама структура ВНС.

Активность i-нейрона слоя образцов определяется на основе функции Гаусса (15.1). Для этого в соответствии с (15.4) заменим в (15.1) векторы

образцов $\mathbf{X}_i$ на соответствующие им векторы весов $\mathbf{W}_i^{\mathrm{T}}$. Тогда функция активности *i*-нейрона слоя образцов приобретает вид

$$y_i(\mathbf{X}) = \exp\left(-\frac{\left\|\mathbf{X} - \mathbf{W}_i^{\mathrm{T}}\right\|^2}{2\sigma^2}\right),$$

или, в покомпонентном представлении

$$y_i(\mathbf{X}) = \exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{N}(x_j - w_{ij})^2\right). \qquad (15.5)$$

В ВНС необходимо провести предварительную нормализацию входных векторов. Это выполняется путем деления каждой компоненты входного вектора на его длину:

$$x_j^{\mathrm{н}} = \frac{x_j}{\sqrt{x_1^2 + x_2^2 + \ldots + x_N^2}} = \frac{x_j}{\sqrt{\sum_{j=1}^{N} x_j^2}}. \qquad (15.6)$$

Такая операция превращает входной вектор $\mathbf{X}$ в вектор единичной длины $\mathbf{X}^{\mathrm{н}}$ в *N*-мерном пространстве.

Исходя из соответствия между векторами весов $\mathbf{W}_i^{\mathrm{T}}$ и векторами образцов $\mathbf{X}_i$, нормализацию следует провести также и для весов

$$w_{ij}^{\mathrm{н}} = \frac{w_{ij}}{\sqrt{w_{i1}^2 + w_{i2}^2 + \ldots + w_{iN}^2}} = \frac{w_{ij}}{\sqrt{\sum_{j=1}^{N} w_{ij}^2}}. \qquad (15.7)$$

Введем в выражение (15.5) для функции активности *i*-нейрона нормализованные значения компонент x_j и w_{ij} и преобразуем его к более простой для вычислений форме:

$$y_i(\mathbf{X}) = \exp\left(-\frac{1}{2\sigma^2} \sum_{j=1}^{N} (x_j - w_{ij})^2 \right) =$$

$$= \exp\left(-\frac{1}{2\sigma^2} \sum_{j=1}^{N} \frac{2x_j \cdot w_{ij}}{\sqrt{\sum_{j=1}^{N} x_j^2} \cdot \sqrt{\sum_{j=1}^{N} w_{ij}^2}} \right) = \exp\left(\frac{1}{\sigma^2} \sum_{j=1}^{N} x_j^{\text{н}} \cdot w_{ij}^{\text{н}} - 1 \right). \qquad (15.8)$$

Функция активности *k*-нейрона слоя суммирования определяет значение плотности распределения вероятностей для всего *k*-класса. После нормализации она вычисляется по формуле

$$Y^k(\mathbf{X}) = \sum_{i-1}^{L_k} \exp\left(\frac{1}{\sigma^2} \sum_{j=1}^{N} x_j^{\text{н}} \cdot w_{ij}^{\text{н}} - 1 \right), \quad k = \overline{1, m}, \qquad (15.9)$$

где m – число классов образов.

Процедура обучения ВНС производится следующим образом. Векторы образцов $\mathbf{X}_i,\ i = \overline{1, L}$ предварительно нормализуются и уже в нормализованном виде $\mathbf{X}_i^{\text{н}}$ последовательно предъявляются на входы сети.

Как уже отмечалось выше, в ходе обучения формируется сама структура ВНС. Размерность N векторов обучающей выборки $\mathbf{X}_i,\ i = \overline{1, L}$ определяет число нейронов и структуру входного слоя ВНС. Общий размер L обучающей выборки $\mathbf{X}_i,\ i = \overline{1, L}$ соответствует общему числу нейронов слоя образцов. Число классов m входных образов соответствует числу нейронов слоя суммирования. Число образцов каждого *k*-класса в общем числе L образцов определяют структуру связей слоя суммирования: каждый *k*-нейрон слоя суммирования имеет связи только с нейронами слоя образцов, которые относятся к *k*-классу.

Предъявление сети каждого из L векторов $\mathbf{X}_i^{\text{н}}$ сопровождается указанием от учителя номера *k*-класса, которому принадлежит входной образец. Последовательность предъявления обучающих векторов может быть любой. После предъявления всех L векторов $\mathbf{X}_i^{\text{н}}$ обучающей выборки, формируется структура сети, и становятся определенными параметры сети в виде матрицы $\mathbf{W}_i^{\text{н}}$. На этом процесс обучения ВНС заканчивается, и сеть готова к классификации неизвестных образцов.

В рабочем режиме сети предъявляется входной образ $\mathbf{X}$ неизвестного класса, который вначале нормализуется (приводится к виду $\mathbf{X}^{\text{н}}$), затем умножается на матрицу весов $\mathbf{W}_i^{\text{н}}$ и соответствующим образом активирует нейроны слоя образцов. Каждый нейрон слоя образцов выдает на своем выходе некоторый уровень активности $y_i(\mathbf{X})$. Каждый k-нейрон слоя суммирования суммирует уровни активности $y_i(\mathbf{X})$ всех нейронов слоя образцов своего k-класса и выдает на своем выходе общий уровень активности данного k-класса $\mathbf{Y}^k(\mathbf{X})$. Выходной нейрон на основании вычисленных сетью уровней активности по каждому классу $\mathbf{Y}^k(\mathbf{X})$, $k=\overline{1,m}$ определяет какой нейрон слоя суммирования имеет максимальный выходной сигнал $\mathbf{Y}^k(\mathbf{X})$. Тем самым (по номеру k-нейрона), определяется номер класса k, к которому с большей вероятностью принадлежит предъявленный входной образ $\mathbf{X}$.

Пример. В табл. 15.1 (1-й и 2-й столбцы) приведены исходные учебные данные для классификатора в виде обучающей выборки, состоящей из 16 двухкомпонентных (N=2) векторов $\mathbf{X}_i$, $i=1,2,...,16$ трех классов: A, B и C. Причем класс A представлен шестью образцами, а классы B и C – пятью образцами каждый. Классификации подлежит неизвестный входной вектор $\mathbf{X}$, имеющий компоненты: x_1=5,8; x_2=4,4. Расположение исходных учебных данных и неизвестного образца иллюстрирует рис. 15.2.

Этап 1. Проведем нормализацию всех обучающих данных, а также – неизвестного образца $\mathbf{X}$. Результаты нормализации учебных данных приведены в табл. 15.1 (3-й и 4-й столбцы). Нормализация неизвестного вектора $\mathbf{X}$ дает $\mathbf{X}^{\text{н}}$: $x_1^{\text{н}}=0,7967$, $x_2^{\text{н}}=0,6044$. Расположение нормализованных учебных данных и нормализованного неизвестного образца иллюстрирует рис. 15.3.

Этап 2. Проведем обучение сети. Предъявление сети каждого из 16 векторов $\mathbf{X}_i^{\text{н}}$ сопровождается указанием (учителя) номера k-класса, которому принадлежит входной образец. После предъявления всех 16 векторов формируется структура сети и матрица $\mathbf{W}_i^{\text{н}}$. На этом процесс обучения ВНС заканчивается, и сеть готова к классификации неизвестных образцов.

Этап 3. В рабочем режиме сети предъявляется входной нормализованный образ $\mathbf{X}^{\text{н}}$ неизвестного класса, который умножается на матрицу весов $\mathbf{W}_i^{\text{н}}$ и соответствующим образом активирует нейроны слоя образцов. Тем самым вычисляются уровни активности $y_i(\mathbf{X})$ всех 16 нейронов слоя образцов. Далее вычисляются общие уровни активности $\mathbf{Y}^k(\mathbf{X})$ каждого из трех нейронов слоя суммирования. Из трех уровней активности $\mathbf{Y}^k(\mathbf{X})$ выбирается максимальное (0,5459), которое и определяет, что предъявленный входной образ $\mathbf{X}$ с большей вероятностью принадлежит к первому из трех классов, т.е. – к классу А.

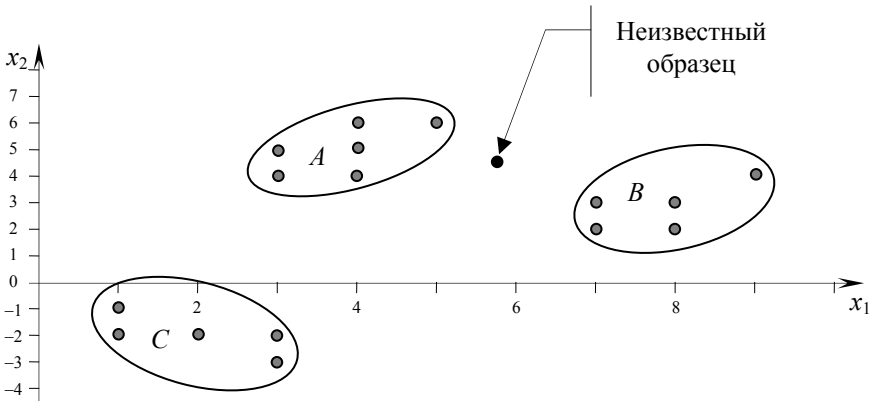


Рис. 15.2. Расположение исходных учебных данных и неизвестного образца

Таблица 15.1

Учебные данные классификатора

Класс	Исходные обучающие векторы		Нормализованные обучающие векторы	
	x_1	x_2	$x_1^{\text{н}}$	$x_2^{\text{н}}$
A	3,0	5,0	0,5145	0,8575
	4,0	4,0	0,7071	0,7071
	3,0	4,0	0,6000	0,8000
	5,0	6,0	0,6402	0,7682
	4,0	6,0	0,5547	0,8321
	4,0	5,0	0,6247	0,7809
B	7,0	2,0	0,9615	0,2747
	7,0	3,0	0,9191	0,3939
	8,0	2,0	0,9701	0,2425
	8,0	3,0	0,9363	0,3511
	9,0	4,0	0,9138	0,4061
C	1,0	–1,0	0,7071	–0,7071
	1,0	–2,0	0,4472	–0,8944
	2,0	–2,0	0,7071	–0,7071
	3,0	–2,0	0,8321	–0,5547
	3,0	–3,0	0,7071	–0,7071

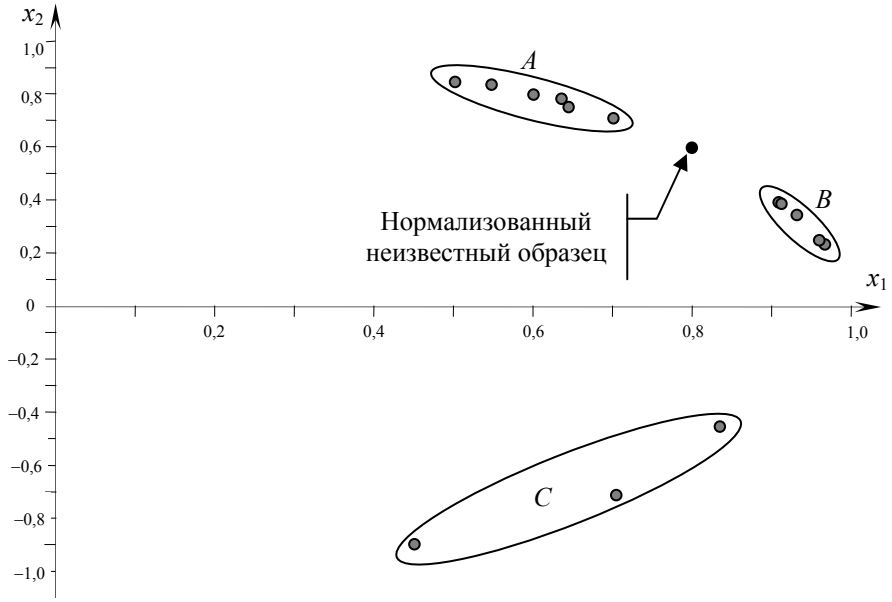


Рис. 15.3. Расположение нормализованных учебных данных и нормализованного неизвестного образца

Вычисления, производимые сетью отображены в табл. 15.2.

Таблица 15.2

Вычисления, производимые ВНС

Класс	№ нейрона слоя образцов	w_1	w_2	$y_i(\mathbf{X})$	$\mathbf{Y}^k(\mathbf{X})$
A	1	0,5145	0,8575	0,0008	0,5459
	2	0,7071	0,7071	0,3950	
	3	0,6000	0,8000	0,0213	
	4	0,6402	0,7682	0,0768	
	5	0,5547	0,8321	0,0040	
	6	0,6247	0,7809	0,0480	
B	7	0,9615	0,2747	0,0011	0,1389
	8	0,9191	0,3939	0,0516	
	9	0,9701	0,2425	0,0003	
	10	0,9363	0,3511	0,0153	
	11	0,9138	0,4061	0,0706	

C	12	0,7071	–0,7071	0,0000	0,0000
	13	0,4472	–0,8944	0,0000	
	14	0,7071	–0,7071	0,0000	
	15	0,8321	–0,5547	0,0000	
	16	0,7071	–0,7071	0,0000	

Применение ВНС. ВНС не является столь общей, как некоторые другие многослойные сети. Например, в отличие от многослойных сетей с обратным распространением ошибки, способных моделировать отображение общего вида, ВНС в своей базовой архитектуре ограничивается только задачами классификации. Вместе с тем, практическая потребность в решении задач классификации достаточно велика, а классификацию ВНС выполняет очень хорошо. Они быстро обучаются, допускают наличие ошибочных данных и обеспечивают полезные результаты даже на малых выборках учебных данных. Однако ВНС оказываются весьма требовательными в отношении ресурсов. Для задач, требующих большого объема обучающих данных (сотни и тысячи примеров), классификация на основе ВНС может потребовать значительного времени. В таких случаях становится целесообразной аппаратная поддержка вычислений на основе ВНС.

ВНС могут быть и более сложными, чем сеть, описанная выше. Например, для каждого входного признака можно использовать разные значения σ – ширины функции Гаусса, определяющей ее влияние. Можно также строить ВНС для классификации многомерных векторов **X**, имеющих различный масштаб по каждой оси (коррелированные входные данные) Существуют и другие усложненные модификации ВНС.

Библиографический список

1. Гриняев С.Н. Интеллектуальное противодействие информационному оружию. – М.: СИНТЕГ, 1999. – 232 с.
2. Галатенко А. Активный аудит. – Информационный бюллетень Jet Info, 1999. № 8(75). – http://www.jetinfo.ru/1999/8/1/article1.8.1999.html.
3. Котенко И.В., Карсаев О.И. Использование многоагентных технологий для комплексной защиты информационных ресурсов в компьютерных сетях / Электронный журнал «Перспективные информационные технологии и интеллектуальные системы», 2001. № 2. – http://pitis.tsure.ru/files6/12.htm.
4. Гаврилов А. В. , Канглер В. М. Использование искусственных нейронных сетей для анализа данных. – Сборник научных трудов НГТУ. – 1999. №3(16). – С. 56-63.
5. Слэйгл Дж. Искусственный интеллект: Подход на основе эвристического программирования: Пер. с англ./Под ред. Г.Е. Поздняка. – М.: Мир, 1973. – 320 с.
6. Чернухин Ю.В. Искусственный интеллект и нейрокомпьютеры. – Таганрог: Изд-во ТРТУ, 1997. – 273 с.
7. Тьюринг А. Может ли машина мыслить? – М., 1960.
8. Мак-Каллок У., Питтс У. Логические исчисления идей, относящихся к нервной активности. Сб. «Автоматы». – М.: ИЛ, 1956.
9. Розенблатт Ф. Принципы нейродинамики (Перцептроны и теория механизмов мозга). – М.: Мир, 1965. – 480 с.
10. Минский М., Пейперт С. Персептроны. – М: Мир, 1971.
11. Тэнк Д., Хопфилд Д. Коллективные вычисления в нейроподобных электронных схемах. / В мире науки. – 1988. № 2.
12. Искусственный интеллект. – В 3-х кн. Кн. 2. Модели и методы / Под ред. Д.А. Поспелова. – М.: Радио и связь, 1990. – 304 с.
13. Курейчик В. М. Генетические алгоритмы. Монография. – Таганрог: Изд-во ТРТУ, 1998. – 242 с.
14. Общая физиология нервной системы. В серии: Руководство по физиологии. Л., Наука, 1979. – 717 с.
15. Осовский С. Нейронные сти для обработки информации / Пер. с польского И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.
16. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика: Пер. с англ. – М.: Мир, 1992. – 240 с.
17. Позин Н.В. Моделирование нейронных структур. – М.: Наука, 1970. – 264 с.
18. Брюхомицкий Ю.А., Письмиченко Д.В., Фадеев Р.В. Лабораторный практикум «Исследование цифровых моделей нейронов, нейронных ансамблей и сетей» по курсу: «Системы искусственного интеллекта». Таганрог: Изд-во ТРТУ, № 2965, 2001. – 43 с.

Оглавление

Брюхомицкий Юрий Анатольевич

Учебное пособие

НЕЙРОСЕТЕВЫЕ МОДЕЛИ ДЛЯ СИСТЕМ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Ответственный за выпуск	Брюхомицкий Ю.А.
Редактор	Надточий З.И.
Корректор	Чиканенко Л.В.

ЛР № 02565 от 23 июня 1997 г.	Подписано к печати
Формат 60 × 84 $^{1}/_{16}$	Бумага офсетная
Офсетная печать.	Усл. п.л. – 10 Уч.-изд. – 9.7
Заказ №	Тираж 100 экз.

«С»

Издательство Таганрогского государственного радиотехнического университета
ГСП 17А, Таганрог, 28, пер. Некрасовский, 44
Типография Таганрогского государственного радиотехнического университета
ГСП 17А, Таганрог, 28, Энгельса, 1