Глава 4

Сетевая архитектура

Искусственный интеллект – это в какой-то степени чистая математическая конструкция – набор алгоритмов, эвристических или каких-то иных, но алгоритмов. А любые алгоритмы нуждаются в устройстве для своего исполнения. И вопрос устройства тем более небезразличен для ИИ в силу необычности самой концепции интеллекта, уж больно он не похож на машину в нашем общечеловеческом понимании.

Машину, реализующую искусственный интеллект, можно построить сразу с жесткой структурой, отнеся вопросы гибкости и способности перенастраиваться с задачи на задачу на программное обеспечение. Однако факт существования мыслящего мозга показывает, что такой подход не единственно возможный. Наш мозг не появляется на свет знающим и умеющим все, что можно, это, во-первых, а во-вторых, его системотехника радикально отличается от компьютерной.

Человеческий мозг на нижнем уровне своей организации состоит из одинаковых элементов – нейронов. Один нейрон от другого не отличается, их конструкция очень проста, а функционал довольно примитивен. И все структуры мозга образуются из них в процессе развития. Несколько утрировано можно сказать так: изначально есть неорганизованная масса нейронов, которая, отвечая на внешние потоки информации, начинает процесс самоорганизации, постепенно выстраивая все необходимые структуры.

Выгода такой постановки дела очевидна. Элементы нижнего уровня – нейроны имеют очень простую структуру, а значит, их легко реализовать (если мы, конечно, поймем, какая структура нейронов необходима). Сборка любых частей машины, решающих различные задачи и при этом состоящих из одних и тех же компонентов, – мечта

любого производства. Кроме того, такая конструкторская идея открывает совершенно потрясающие возможности для ремонта. Если есть некоторое количество свободных нейронов, то ремонт мыслящей машины состоит из двух операций: отключения поломавшихся нейронов и подключения запасных. Выгоды очевидны, но и проблем встает немало. Даже поверхностный анализ дает очень сложные вопросы:

1. Очевидно, что простота нейрона должна включать тот минимум, который позволит объединившимся нейронам создать мыслящую машину. А как определить этот минимум?
2. Что может представлять собой механизм самоорганизации? Каким образом поток внешней информации заставляет нейроны собираться в какие-то конструкции?
3. Каким образом из простого функционала можно получить сложный? Конечно, нам известен прецедент человеческой электроники, когда базовый элемент – транзистор – участвует в создании совершенно различных устройств. Но все же транзистор – не единственный компонент элементной базы.

Вопросов много, но тем не менее в целом идея нейронной сети выглядит очень интересно, ясно, что если удастся дать хорошие ответы на эти и многие другие вопросы, то мы получим очень серьезную технологию не только для построения искусственного интеллекта, но и техники вообще. Фантасты, кстати, на такую возможность указали достаточно давно, создав литературные произведения, в которых примитивные организмы, объединяясь в коллектив, приобретают не просто грубую силу, а качественно иные возможности. Примеры такого рода мы можем найти и в нашей земной природе. Пчелиная семья обладает возможностями, не имеющимися у одиночной пчелы, играющей в улье роль нейрона. У пчел даже есть некий символьный язык, благодаря которому они в состоянии передавать друг другу информацию о местонахождении цветов. Еще один интересный пример – это колонии муравьев и термитов. И муравьи, и термиты не только умеют самоорганизовываться на решение рабочих задач. Как утверждают энтомологи, и муравьи, и термиты за счет выбора рациона кормления личинок умеют выращивать специализированных особей: более мелких, рабочих и оснащенных мощной головой солдат. Каким образом на это способны существа, не обладающие даже хорошо развитой нервной системой, – очень большой вопрос. Возможно, ответ на него будет таким же, как и на вопрос, каким образом специально организованные сообщества нейронов человеческого мозга могут рассчитать параметры реактивного самолета.

Но вернемся к нашей частной задаче – построения интеллекта. Любая система обработки информации, а интеллект можно рассматривать и с такой стороны, нуждается в высокой производительности. Способность быстро реагировать упирается в способность быстро обрабатывать информационные потоки. Однако до сих пор многие задачи, которые человек решает практически мгновенно, для современных компьютеров оказываются неимоверно сложными. Такова, например, задача распознавания знакомого лица в толпе людей – задача сложная математически, требующая огромных вычислительных ресурсов, не вызывает затруднений для человека. Почему так?

Ответ на этот частный вопрос получен достаточно давно, и он заключается в простой фразе «параллельные вычисления». Что это такое, можно пояснить на очень простом примере алгебраических квадратных уравнений. Мы помним, что общий вид квадратного уравнения таков:

$$ax^2 + bx + c = 0.$$

Для его решения необходимо вычислить величину, называемую дискриминантом, по формуле:

$$D = b^2 - 4ac.$$

Если полученное значение больше нуля, то корней будет два, расчет которых можно вести независимо друг от друга. Но вот в чем беда, в однопроцессорном компьютере расчеты ведутся последовательно. Нельзя вычислять оба корня одновременно. Но если компьютер будет иметь два вычислительных ядра, то на этапе расчета корней время на вычисления можно сократить в два раза, передав счет первого корня одному ядру, а счет второго – другому.

Это, конечно, примитивный пример, но суть дела он показывает верно. Задач, сводящихся к одному линейному вычислительному процессу, очень мало. Можно даже утверждать, что любая достаточно сложная задача допускает возможность параллельных вычислений. Поэтому уже на заре развития компьютерной техники возникла идея многопроцессорных вычислительных систем. Рост производительности одного процессора упирается в технические ограничения, в то время как увеличивать количество процессоров можно практически бесконечно. Нейронная сеть в каком-то смысле является предельным вариантом развития многопроцессорной системы, в которой процессоров уже миллионы или даже миллиарды, каждый из них умеет мало, а вычислительная сила достигается за счет распараллеливания

вычислений и сосредоточения на одной задаче большого количества таких маленьких вычислителей.

Нейрон

Прежде всего заметим, что дальше разговор пойдет о моделировании нейронных сетей на базе существующей вычислительной техники. Архитектурно мы не отходим от идеи фон Неймана. Речь не идет о создании машин с новой архитектурой, пока нейронные сети – это специально организованные программы для классического компьютера. Конечно, какие-то работы в направлении создания совершенно новых машин идут, но мне, автору этой книги, неизвестны какие-либо яркие прорывные результаты в этой области. И полагаю, и не я только, но и более компетентные в этой области люди, что прорыв будет возможен только с созданием принципиально новой элементной базы.

Итак, мы с вами договорились, что дальше речь пойдет только о математической модели, и в первую очередь определим понятие нейрона. В дальнейшем мы не будем различать искусственный нейрон и естественный (клетка высокоорганизованной нервной системы), будем говорить просто о нейроне.

Единственная функция нейрона заключается в преобразовании сигнала. Есть входной сигнал, который он получает либо от входного устройства, либо от другого нейрона, затем он выполняет с полученным сигналом какие-то действия и выдает его как выходной, например, другому нейрону. И вот здесь кое-что необходимо уточнить.

В любой информационной системе всегда есть шум. Его природа может быть разной. Если система занимается передачей электрических импульсов, то всегда будет некоторое количество паразитных сигналов, не несущих в себе никакой информации и появившихся либо в результате внешнего воздействия (внешние электрические поля и т. д.), либо в результате неуправляемых внутренних процессов. Это означает, что если нейрон станет реагировать на любой входящий сигнал, то он начнет воспроизводить информационный шум, в котором со временем вся система просто утонет.

Заметим, что описанная проблема не новость, она известна и современным цифровым технологиям. Компьютер построен на устройствах, моделирующих триггер – элемент, могущий находиться в двух состояниях: 1 – включен и 0 – выключен. В технической реализации наличие электрического шума приведет к тому, что состояние «вы-

ключен» окажется невозможным в силу гарантированного минимального сигнала. Выход из положения заключается в том, чтобы состояние «выключен» определялось не нулевым сигналом, а меньшим некоего порогового значения, которое определяется так, чтобы порог был выше уровня шума. Ну а состояние «включен» мы определим сигналом выше порога.

Таким же образом можно определить и состояние активности нейрона. Активация означает готовность нейрона принять и обработать поступивший сигнал. Отличие от старого доброго триггера здесь в том, что для нейрона в общем виде определяется не один сигнал, а группа, по которой вычисляется суммарный, или средневзвешенный, сигнал, и если он оказывается выше порога, то нейрон активируется. Надо сказать, что в теории нейронных сетей сказано довольно много о виде таких функций, но наша задача – уяснение общих идей, поэтому углубляться в математические тонкости не будем.

Второе отличие нейрона от триггера еще более интересно. Если задача триггера – лишь изображать собой ноль или единицу, то есть он устройство, в каком-то смысле пассивное, то нейрон соединяет в себе обе функции, он занимается и хранением информации, и ее обработкой. Активизировавшись, нейрон включает функцию обработки сигнала, которая, в принципе, может быть простой. Обработанный сигнал поступает на выход и передается другим нейронам или выходным устройствам.

Активность нейрона может быть угнетена. Предположим, что средневзвешенный входящий сигнал вычисляется по формуле:

$$F = \sum_k w_k x_k,$$

где x_k – это уровень k-го сигнала, а w_k – его вес. Если у некоторого сигнала высокий уровень, но ему приписан отрицательный вес, то этот сигнал будет работать на подавление активности нейрона. То есть даже такая простая функция счета входного сигнала создает интересные возможности для управления активностью нейрона. А это, в свою очередь, означает, что поведение нейронной сети зависимо от конкретной схемы подключения нейронов внутри сети, или, как говорят специалисты, зависимо от топологии сети.

Обучаемость нейронной сети

Нейронные сети, кроме возможности распараллеливания вычислительных процессов, предоставляют качественно новую способность

к обучению. Их обучаемость зашита в самой архитектуре сетей, является их фундаментальным свойством. Сеть – это прежде всего нейроны с определенной функцией преобразования сигнала. Эта функция, очевидно, не изменяема в процессе функционирования сети. Можно придумать сети с группами нейронов, обладающих различными функциями преобразования, можно создать сеть из однотипных элементов. Но сложность сети не объясняет ее способности обучаться.

Что означает фраза «Человек обучен» с точки зрения структур головного мозга? Новые нейроны в результате успешного учебного процесса не появятся. Устройство отдельных нейронов останется прежним. Единственная возможность что-либо изменить – это управление структурой связи между отдельными нейронами. Наверное, идеальным системотехническим решением, эффективным и то же время экономным, были бы блуждающие связи, вроде валентных связей атомов в молекулах. Один и тот же атом может вступать в химическую связь с разными атомами, создавая при этом совершенно разные по своим свойствам вещества. Но это пока из области фантастики. Более реальная схема – это соединение отдельного нейрона большим количеством равнозначных связей с другими нейронами и функцией активации, способной некоторые связи делать активными, а некоторые подавлять. Фактически такая функция будет изменять структуру мыслящей сети, а значит, изменять ее свойства. В общем, обучаемость нейронной сети зашита именно в этой функции.

Для вычислительных сетей есть термин «обучение с учителем». Это означает, что существует входной сигнал, для которого известна правильная реакция. Система, получая сигнал, сравнивает свою реакцию с заданным ответом. Если реакция ответу не соответствует, то это повод для изменения внутри сети

Сейчас, на некоторое время, чтобы дать понимание идей сетевой архитектуры, например то, как сетевое устройство может обучаться решению задач, мы отойдем от нейрона в том смысле, как его понимает современная теория искусственного интеллекта, и будем использовать понятие «вычислительная ячейка». Ячейка умеет выполнять какие-то простые операции, умеет получать данные от других ячеек и, естественно, передавать результаты своей работы. Нейрон – это частный случай такой ячейки. Понятие нейронной сети мы пока заменим понятием «вычислительная сеть».

А теперь вернемся к нашему подзаголовку. Подстройка – это своего рода обучение с «учителем». Роль учителя играет входной сигнал (набор входных данных), для которого известен правильный выходной сигнал (результирующие данные). Для лучшего понимания разберем технику подстройки на примере. Пусть набор входных данных состоит из числовых значений переменных *A*, *B*, *C*. Результирующий сигнал представлен величиной *Q* с известным числовым значением. Повторимся, числовые значения входных и выходных данных известны. Допустим, вычислительная сеть выдала результат, совпадающий с контрольным сигналом, в этом случае подстройка системе не нужна.

Теперь предположим более интересную ситуацию. Выходные данные не соответствуют контрольному сигналу, то есть вычислительная сеть ошиблась. Положим, что в сети есть ячейки, увеличивающие значения выходных величин, и есть – уменьшающие. Если полученная величина *Q* больше исходного значения, то надо в ячейках, увеличивающих *Q*, повышать отрицательные веса. Такая политика начнет торможение ячеек, создающих ошибку, в результате можно ожидать, что величина *Q* начнет уменьшаться, а ошибка между контрольным значением и вычисленным сетью – сокращаться. Спустя некоторое количество учебных запусков сеть может прийти к значению, отличающемуся от контрольного, в допустимых пределах. Покажем на примере, как это возможно.

Сконструируем вычислительную сеть, вычисляющую сумму или разность двух положительных чисел (A и B), и покажем, как она может настраиваться на одну из доступных для нее задач. Построим сеть из трех ячеек: *P* (инициализирующая результат), *Q1* (прибавляющая 1 к результату), *Q2* (вычитающая 1 из результата). Подключим ячейку друг к другу следующим образом (см. рис. 4.1):

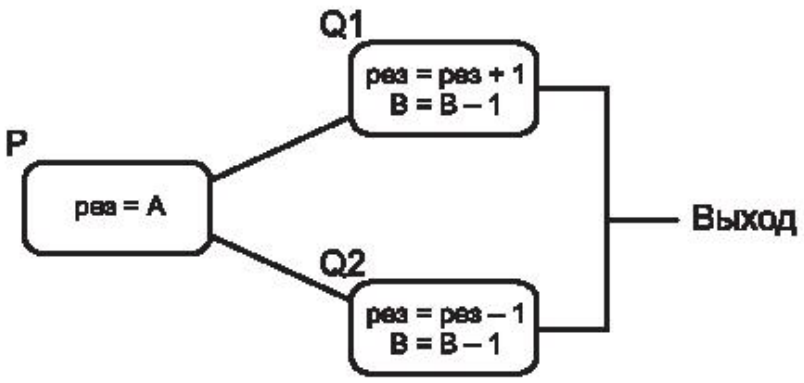

Рис. 4.1 ❖ Суммирующая сеть

Чтобы не загромождать схему, здесь указана только самая общая информация, а работать схема будет так: ячейка *P* получает числа A и B и вычисляет исходное значение величины **рез** (результат). Затем данные передаются ячейкам *Q1* и *Q2*. Они активируются безусловно, только фактом получения информации. *Q1* начинает наращивать величину «рез» и уменьшать величину B. Ячейка *Q2* выполняет обратную работу. Алгоритм работы ячеек таков:

```
Пока B не равно нулю, делать
      Изменять величину «рез»
Конец цикла
```

Пусть теперь сеть получила на вход набор данных (2, 3) и контрольное значение, равное 5. Такой контрольный сигнал будет отражать операцию сложения. *Q1* сможет закончить свою работу и выдать ответ, соответствующий контрольному значению.

Модифицируем сеть, введя функцию активации для *Q1* и *Q2*. Оформим функцию максимально просто. Пусть она определяется одним числом w и ячейка активна, только если это число больше нуля. До начала работы инициализируем значение w единицей. И создадим дополнительный канал между *Q1* и *Q2*, по которому можно передать новое значение w (см. рис. 4.2).

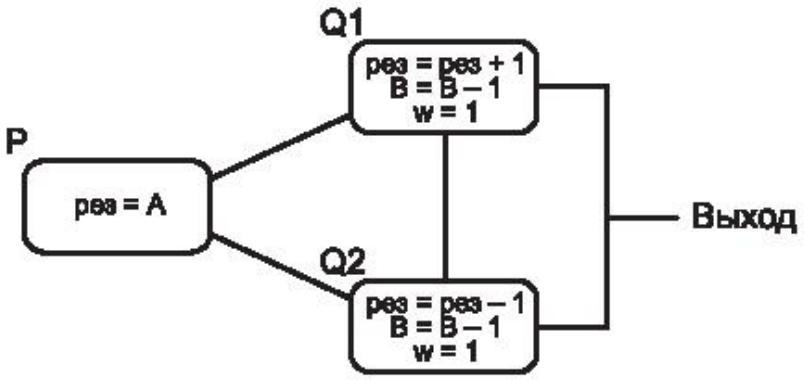

Рис. 4.2 ❖ Подстройка суммирующей сети

Внесем в работу ячеек *Q1* и *Q2* корректировку. Пусть теперь закончивший свою работу первым отошлет по каналу, связывающему его с партнером, отрицательное значение для w. Тогда наш контрольный сигнал (2, 3, 5) приведет к установлению постоянной активной цепочки *P – Q1 – Выход*. А значит, сеть настроится на выполнение сложений двух чисел, а *Q2* будет деактивирована значением, полученным от *Q1*. И можно быть уверенным, что последующие примеры на сложение также будут выполнены правильно. Перенастройка

сети на вычитание двух положительных чисел может быть произведена инициализацией функции активации и выполнением простейшего примера на вычитание, после которого сети можно дать большой пример.

Построенная нами вычислительная сеть, конечно, игрушечная и не пригодна для реальных задач. И ячейки слишком сложны (умеют выполнять законченный алгоритм счета), но это не более чем иллюстрация, цель которой – показать принцип работы сети, способной настраиваться на выполнение задачи, создавая устойчивые активные связи, которые, в принципе, потом можно разрушать и создавать новые, под новые задачи. При этом сама сеть не изменяется, возможности каждой ячейки не изменяются, модификации подлежат только условия взаимодействия ячеек друг с другом. И чем их больше, чем сеть обширнее, тем больше возможностей по созданию сложных конфигураций. И что очень интересно, активные конфигурации ячеек создаются не программистом, а актом настройки сети, ее обучением на задачу.

Специализированную сеть, умеющую настраиваться на задачи из ограниченного класса, придумывать не так уж сложно. Такие сети могут быть очень просты в реализации. Рассмотрим для примера задачу сортировки в самом общем ее понимании

Если дано множество чисел, то его можно отсортировать по возрастанию, можно по убыванию. А можно распределить числа по какой-то кривой (как, например, на рис. 4.3):

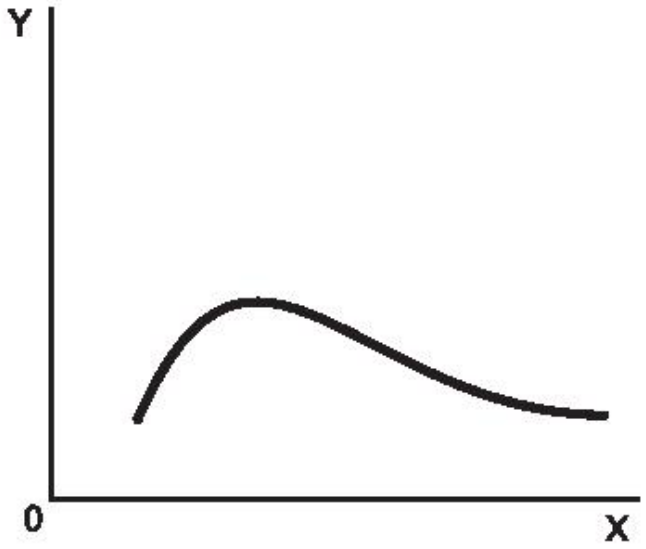

Рис. 4.3 ❖ Произвольная кривая

Всевозможных распределений множества чисел существует сколь угодно много. И алгоритмов сортировки тоже множество. Посмотрим, сможет ли вычислительная сеть справиться с общей задачей сортировки. Предпосылки к этому есть, и очень хорошие. Напомним, что простейший алгоритм сортировки по возрастанию (убыванию), именуемый пузырьком, за одну итерацию сравнивает и переставляет два рядом стоящих элемента. Ясно, что можно придумать линейную сеть, такую, что изначально в каждую ячейку загружается одно число, а затем ячейки начинают обмениваться друг с другом по какому-то правилу. Конечно, не слишком интересно специализированное правило. Хотелось бы придумать самую общую схему, позволяющую настраиваться на любое распределение.

Нам понадобится линейная сеть, способная запоминать шаблон и подгонять любой входной набор данных под этот шаблон. Для упрощения рассуждений положим, что количество ячеек в точности соответствует количеству сортируемых чисел. Иное составляет проблему технического, а не принципиального характера. Построим линейную сеть такого вида (рис. 4.4):

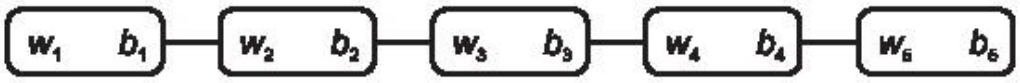

Рис. 4.4 ❖ Линейная сеть для сортировки

Конструктивно устроим так, чтобы ячейки регулярно устанавливали контакт в парах (образованных соседями). Как именно будет это происходить, в каком порядке, для нашей задачи роли не играет. Желательно лишь, чтобы интенсивность контактов между любыми двумя парами была примерно одинаковой. Цель контакта – обмен данными. Но если любая пара, установившая контакт, будет в обязательном порядке обмениваться данными, то толку с этого не будет, необходимо установить какие-то правила взаимодействия. Опишем некоторые желательные свойства нашей сети.

Характеристики w_k используем для хранения обучающего сигнала. Если сеть инициализирована, скажем, нулями, то первый сигнал она воспримет как обучающий и сохранит полученные данные в величинах w.

Введем следующее правило обмена данными. Есть пара ячеек, установившая контакт. Первая из них хранит характеристику w_1 и данное b_1. Вторая имеет характеристику w_2 и данное b_2. Обмен между ними будет осуществляться только в том случае, если выполняется условие:

$$\max(|w_1 - b_1|, |w_2 - b_2|) > \max(|w_1 - b_2|, |w_2 - b_1|).$$

Левая часть неравенства представляет собой максимальную ошибку исходного распределения. Правая часть – это максимальная ошибка после возможного обмена. Естественно, если максимальная ошибка уменьшается, то необходимо запустить процедуру обмена. Небольшой пример. Построим сеть из двух ячеек, моделирующую проверку неравенства A < B и обмен значениям между величинами A и B, в случае если неравенство оказывается истинным (см. рис. 4.5):

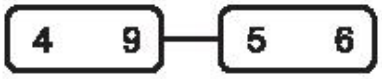

Рис. 4.5 ❖ Пример неравенства

Наша сеть была обучена на примере (4, 5), что видно по весовым значениям, а данные, полученные на обработку, – это (9, 6). Сейчас модуль максимальной ошибки 9 – 4 = 5, после возможного обмена максимальная ошибка окажется равной 9 – 5 = 4. То есть обмен желателен. Наше условие выглядит естественно, и можно ожидать, что оно будет работать на значительном количестве примеров. Но не факт, что везде. Попробуем на этом условии отработать сортировку по возрастанию. Пусть сеть обучена контрольным сигналом (1, 2, 3) и загружены данные для сортировки (5, 4, 1). Естественно ожидать в качестве результата последовательность (1, 4, 5). Посмотрим, получится ли. Исходная картинка такова (см. рис. 4.6):

Рис. 4.6 ❖ Начало сортировки

Схему образования пар выберем самую простую. Первыми контактируют первая и вторая ячейки, затем вторая и третья, затем опять первая и вторая. Максимальная ошибка первой пары 5 – 1 = 4. В случае обмена максимальная ошибка уменьшается: 4 – 1 = 3 и 5 – 2 = 3. Ячейки обмениваются данными, и мы получаем такую картинку (см. рис. 4.7):

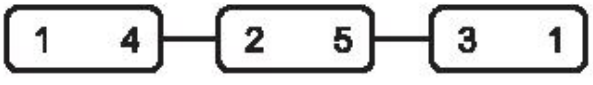

Рис. 4.7 ❖ Первый шаг сортировки

Теперь анализируем пару, составленную из второй и третьей ячеек. Имеем модуль максимальной ошибки 5 – 2 = 3. Если будет произведен обмен, то этот модуль составит 5 – 3 = 2. Обмен желателен, и следующий промежуточный результат таков (см. рис. 4.8):

Рис. 4.8 ❖ Второй шаг сортировки

Возвращаемся к паре, составленной из первой и второй ячеек. В данный момент максимальная ошибка составляет 4 – 1 = 3. После возможного обмена модуль максимальной ошибки уменьшится: 4 – 2 = 2. Выполняем обмен и получаем следующий результат (см. рис. 4.9):

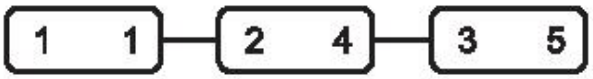

Рис. 4.9 ❖ Последний шаг сортировки

На этом шаге процесс, очевидно, остановится, так как любая попытка обмена в дальнейшем будет приводить к увеличению максимальной ошибки, и в это же время мы видим, что сортировка завершена. В сконструированной сети фактически реализован алгоритм пузырька. Но благодаря архитектуре сети (если, конечно, она реализована не на компьютере с неймановской архитектурой) процесс очень сильно распараллелен. Если, к примеру, в сети 100 ячеек, то на ней могут создаваться 50 одновременно работающих пар, если мы только лишь немного усложним технику работы сети (в нашем примере за раз работает одна пара, но мы можем образовывать много пар одновременно). А это, в свою очередь, означает повышение производительности в 50 раз. И чем сортируемых чисел больше, тем выигрыш в производительности будет выше. И конечно же, такая система должна быть на порядок дешевле, нежели многопроцессорный компьютер, все-таки процессор – это очень сложная и дорогостоящая вещь в сравнении с примитивным устройством ячейки.

Наша сеть обладает довольно значительной степенью универсальности. Точно так же, как мы за один прием обучили ее выполнять сортировку по возрастанию, инициализировав веса нулями и послав контрольный сигнал, представляющий собой массив, упорядоченный по возрастанию, мы обучим ее сортировке по убыванию, отправив

в качестве контрольного сигнала убывающий массив. Если дать ей массив чисел, располагающихся по гауссиане или вдоль другой кривой, она, очевидно, с последующими массивами будет делать то же самое. Но, конечно, придуманное правило уменьшения максимальной ошибки подлежит доказательству, хотя и выглядит разумно. Однако если искать доказательство лень, правило можно принять как полезную эвристику.

Сеть нелинейной геометрии

Заметим, что пока придуманные сети имеют линейную архитектуру. Попробуем на простом примере поиска наибольшего числа в массиве построить сеть с более сложной конфигурацией. Для конструирования сети потребуются ячейки, умеющие хранить одно число, способные его передавать, получать и сравнивать. Такие минимальные навыки заложить в простую схему можно. Конечно, нужна некая выделенная ячейка, в которой в конце процесса окажется найденное наибольшее число. Схема должна работать быстрее обычного алгоритма перебора, учтем, что работа алгоритма сводится к сравнениям и обменам, количество которых сопоставимо с количеством элементов массива. Линейная сеть для такой задачи не подойдет по причине низкой производительности. Если выделенная ячейка, предназначенная для хранения результата, окажется на одном конце сети, а наибольшее число – на другом, то для того, чтобы дотащить наибольшее значение до конца сети, потребуется не меньше операций, чем в обычном алгоритме для машины фон Неймана. Построим сеть немного другой геометрии (см. рис. 4.10):

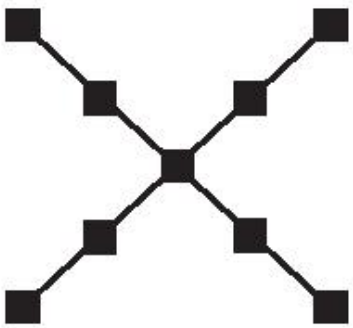

Рис. 4.10 ❖ Сеть для поиска наибольшего

Квадратики – это ячейки с описанными свойствами, соединяющие их линии – связи, по которым можно передавать данные. Придать сети нужные свойства несложно с помощью простой функции

активации. Ячейка может находиться в трех возможных состояниях: неактивна, настроена на передачу, настроена на прием. Пусть в начале процесса крайние настроены на передачу, остальные – на прием. Затем ячейка, принявшая число, перестраивается на передачу, передавшая число становится неактивной. Это условие запустит волну передач, направленную от краев к центру.

Небольшая проблема встанет с центральной ячейкой. После первого успешного сеанса приема она настроится на передачу, несмотря на то что ей необходимо попытаться принять еще три числа. Немного модифицируем функцию активации. Пусть блокирует ячейку значение функции активации, равное –1. 0 настраивает на передачу и число, большее нуля, – на прием. Еще договоримся, что любое действие уменьшает значение функции на 1. Тогда, если центральную ячейку инициализировать большим положительным числом, она долго будет настроена только на прием. Если крайние инициализировать нулями, то они после первой передачи перейдут в неактивное состояние. Остальных достаточно инициализировать 1. В этом случае первое действие (прием) переведет ячейку в состояние готовности к передаче, а второе (передача) переведет его в неактивное состояние. И наконец, в качестве собственного действия ячейки определим следующий простой алгоритм:

```
Если Собственное число меньше Полученного,
          То Собственное число = Полученному
```

Такая сеть сможет обработать массив в четыре раза быстрее. Для ее совершенствования можно увеличить количество ветвей, идущих к центральной ячейке, а можно сеть сделать многослойной, так чтобы центральная ячейка пучка была началом ветки пучка следующего уровня. Картинка ниже (рис. 4.11) иллюстрирует эту идею.

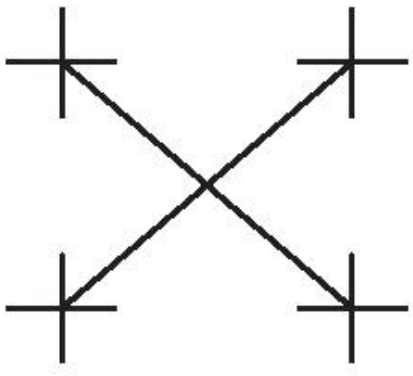

Рис. 4.11 ❖ Многослойная сеть для наибольшего

Это двухуровневая сеть. А так как количество уровней можно сделать сколь угодно большим, то и скорость поиска наибольшего можно увеличивать почти безгранично. Заметим также, что задача поиска наибольшего числа допускает естественное обобщение: дано множество объектов, описывающихся определенной характеристикой. Необходимо найти объект, для которого эта характеристика: принимает наибольшее значение, менее всего отличается от заданного, попадает в некоторый интервал. В общем, можно придумать много задач, для решения которых такая конструкция сети будет полезна. Конечно, функцию активации для иерархической сети придется усложнить, но это уже проблема технического, а не принципиального характера.

Персептрон Розенблатта

Проблема теории нейронных сетей, а может быть, ее достоинство, – в том, что она еще очень молода, как и вообще теория искусственного интеллекта. Поэтому в изложении разных авторов можно встретить нюансы понимания тех или иных концепций, разночтения если не в базовых терминах, то в их использовании. Еще один важный момент – необычность идей искусственного интеллекта, ограничивающая их понимание. Конечно, есть элита специалистов, для которых проблем восприятия идей ИИ не существует, однако эта книга не для них. Моя цель – дать представление основных идей в виде, доступном для неспециалиста, для чего, в частности, в этой главе допущены некоторые вольности. Например, мои вычислительные ячейки, использованные выше, довольно сильно отличаются по своим возможностям от математических моделей, предложенных отцами – основателями теории нейронных сетей. Но это общая проблема поиска золотой середины между популярным изложением и научным. Если первое рискует впасть в профанацию, то второе может оказаться слишком академичным и непонятным. Надеюсь, мы смогли уйти от обеих угроз, и базовая идея – конструирование устройства, состоящего из простых элементов, способных настраиваться на решение сложных задач, изменяя собственные свойства, вами понята. А если это так, то давайте посмотрим, что понимал под нейронной сетью американский нейрофизиолог Ф. Розенблатт. Персептрон (в некоторых транскрипциях «перцептрон») Розенблатта (рис. 4.12) – это модель нейронной сети, которая по его идее должна моделировать принципы работы человеческого мозга в деле распознавания изображений.

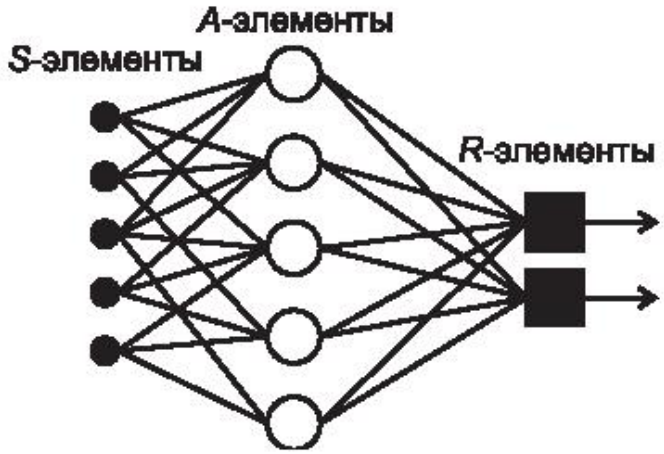

Рис. 4.12 ❖ Персептрон Розенблатта

Персептрон состоит из трех слоев устройств разной функциональности. Поэтому его разумно называть однослойной нейронной сетью, хотя в некоторых источниках о ней говорят как о трехслойной. Первый слой: *S*-элементы, получают входной сигнал и преобразуют его в вид, понимаемый *A*-элементами, собственно, и являющимися нейронами Розенблатта. *A*-элементы воспринимают сигнал и изменяют собственный вес по некоторому правилу. Умение изменять вес является основой обучаемости персептрона. И наконец, *R*-элементы выдают результат работы всего устройства. Сейчас попробуем разобраться, как это работает.

В режиме обучения схема сравнивает результат обработки получаемой картинки с эталоном. Получаемые извне картинки, будем их далее называть обучающими примерами, должны чем-то отличаться друг от друга, в противном случае обучения не будет. Эталон, конечно же, всегда один и тот же. Результат работы персептрона будет давать некоторую ошибку в сравнении с эталоном. Можно описать функцию, минимизирующую ошибку посредством изменения весов нейронов, определяющих вклад нейрона в результирующую картину. После достаточного количества исходных примеров можно ожидать, что персептрон научится выделять эталонную картинку из входного примера, после чего ему можно дать пример, не используемый в обучении.

Поясним, как это работает на геометрическом примере. Внизу три картинки: две учебные и эталон (см. рис. 4.13).

Все три рисунка содержат квадрат с эллипсами в вершинах. Функция, изменяющая активность нейронов, должна обеспечить небольшую корректировку формы и величины эллипсов и небольшую корректировку их местоположения. Если же персептрону подадут

примеры, в которых появится пятая фигура, то сигнал, за нее ответственный, должен быть максимально ослаблен. Если качественно понятно, что должно происходить, то посмотрим, каким образом деятельность персептрона Розенблатта обеспечена количественными соотношениями.

Рис. 4.13 ❖ Пример для распознавания

Введем обозначения: x^k – вектор входящего сигнала, y^k – вектор выходящего сигнала.

Шаг первый. Начальные веса всех нейронов в нулевой момент времени полагаются случайными:

$W(t = 0)$ = случайное число.

Шаг второй. Вычисляется вектор ошибки, каждая составляющая которого – это локальная ошибка в матрице входного сигнала. Вычисляется ошибка как разность между входным сигналом и эталонным:

$$\Delta_k = y^k_{\text{оригинал}} - y^k_{\text{эталон}}.$$

Шаг третий. Вектор весов (все веса, всех нейронов) для следующего момента времени изменяется по формуле:

$$W(t + \Delta t) = W(t) + \eta x^k (\Delta^k)^T.$$

Большая T за последней скобкой означает транспонирование вектора ошибки для перемножения его с вектором входного сигнала. С правилами и операциями векторной алгебры можно ознакомиться по любому учебнику векторной алгебры. Величина η обеспечивает скорость обучения персептрона, и поэтому она так и называется – темп обучения. Ее величина варьируется от 0 до 1. Принципиально функция работает очень просто. Если сигнал, выданный нейроном, недобирает до эталона, то он усиливается. Если нейрон выдал больше, чем нужно, то его активность подавляется.

Для иллюстрации работы персептрона Розенблата есть интересная геометрическая интерпретация. Заметим, что функция преобразования весов имеет линейный характер. Геометрически это означает, что персептрон пытается отделить полезную информацию от шума прямой линией. Это означает, что персептрон может распознать любой объект, отделимый прямой. Однако легко привести контрпример. На рисунке ниже два типа кружков. Черные и светлые. Допустим, что черные – это шум, а светлые представляют собой полезную информацию (см. рис. 4.14):

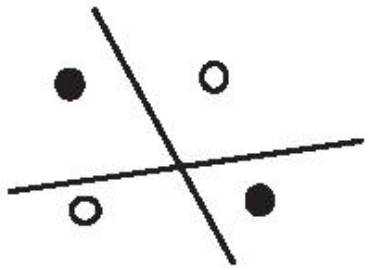

Рис. 4.14 ❖ Неустранимая проблема

Легко видеть, что нет такой прямой, которая отделила бы черные кружки от светлых. А значит, возможности линейного персептрона по распознаванию образов сильно ограничены. Впрочем, это означает лишь необходимость усложнения устройства. И направление усложнения вполне понятно. Выше было сказано, что персептрон Розенблатта однослойный, в том смысле что слой нейронов только один. Здесь зарыт колоссальный резерв усложнения, заключающийся в добавлении слоев и усложнении механизмов, передающих информацию между слоями.

Сети Кохонена

Теуво Калеви Кохонен – финский ученый, специалист в области искусственного интеллекта, предложил вариант архитектуры нейронных сетей, решающих задачу кластеризации объектов. Не перепутайте два различных понятия: классификация и кластеризация. Если первое из них означает анализ входных данных и отнесение объекта к какому-либо из уже известных классов, то второе означает отнесение объекта к одному из еще неизвестных классов. Звучит определение кластеризации несколько парадоксально, поэтому давайте разберемся с термином. Какой смысл относить объект к классу, который сам еще нуждается в определении?

А смысл есть, и достаточно солидный. Кластеризация, по большому счету, представляет собой начало любого исследования незнакомого окружения. Прежде всего, получив на анализ множество объектов, мы должны выделить из них похожие между собой, то есть, возможно, являющиеся модификацией одного объекта. Тем самым мы отделяем этот объект от других, принципиально от него отличных. Выполнив такую работу, мы уже можем дать объекту имя и начать исследовать его свойства.

Сеть Кохонена предполагает, что количество кластеров изначально известно. Это не такое уж сильное предположение. В каком-то смысле ошибиться в оценке их количества невозможно. Просто процесс кластеризации может провести свою работу более тонко или более грубо. Например, сеть может выделить в один кластер стулья, кресла, табуретки. Такой кластер вполне возможен, так как функционально эти три типа объектов похожи. Но при более тонком делении возможно образование не одного, а трех кластеров.

Разберемся, как сеть Кохонена может отделить стул от табуретки, или, в более общем виде, как она может разнести объекты A и B по разным кластерам. Договоримся, что для каждого объекта существует описание, представляющее собой упорядоченный набор признаков. Даже положим, что этот набор представляет собой множество чисел, или, как говорят математики, вектор (множество чисел, в котором роль играет не только значение, но и местоположение числа во множестве – его номер).

Договоримся, что описание любого объекта представляет собой одинаковые по длине наборы чисел, или, иначе говоря, векторы одной размерности. Это не слишком страшное ограничение. Так как нулевые значения не несут в себе никакой информации, то можно составить вектор, в котором некоторые величины будут нулями, то есть добавить нулей до определенной длины вектора. Такой вектор сеть воспринимает первым слоем нейронов, а значит, в первом слое количество нейронов равно размерности вектора описания объекта.

Второй слой состоит уже из любого количества нейронов, называемых линейными сумматорами. Сумматор называется линейным по причине принципа своей работы, он воспринимает входящий вектор и преобразует его в число следующей линейной функцией:

$$F = \sum_k w_k x_k,$$

где величины x_k – значения входящего вектора, а w_k – набор весов, свой, особенный для каждого нейрона. Такой линейный сумматор

выдает для каждого нейрона число. Конечно, в зависимости от выбора весов некоторые нейроны могут выдать одинаковое значение (не отличить стул от табуретки). Далее выбирается нейрон, выдавший наиболее сильный сигнал, а все остальные обнуляются. Такой метод формирования выходного сигнала обозначается определением «Победитель забирает все».

Из вышесказанного следует, что количество нейронов второго слоя равно количеству кластеров, на которые сеть может разделить входные объекты. Заметим также, что сеть Кохонена не нуждается в предварительном обучении с контрольным выходным сигналом. Графически сеть можно представить так (рис. 4.15):

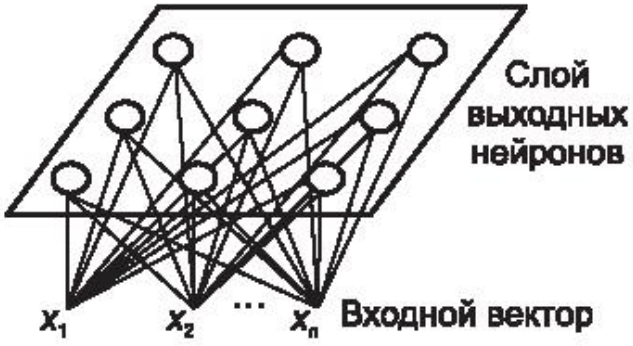

Рис. 4.15 ❖ Сеть Кохонена

Заметим, что конструкция обрабатывающих нейронов очень проста, вся смысловая нагрузка ложится на функции суммирования, на подбор весов. Но это уже отдельная математическая проблема, решаемая с учетом природы задачи. Задача кластеризации, заметим, легко преобразуется в задачу классификации, если о кластерах, на которые мы будем разделять объекты, что-то известно. Многослойные сети вполне могут использоваться для задач обобщения. При нескольких вторых слоях результат их работы можно интерпретировать как входной вектор для третьего слоя. Пусть, например, первый слой выделяет: стулья, табуретки, кресла, тарелки, блюдца, чашки. Сумматоры третьего слоя можно подобрать так, что первые три типа предметов будут выделены в кластер, который условно назовем мебель для сидения, а следующие три – в кластер посуда.

Многослойные сети Кохонена можно использовать для любого уровня обобщения, так как количество слоев может быть любым. Область использования сетей практически ничем не ограничена, точнее, ее функционал задается лишь нашей способностью представить входной набор данных вектором: числовым или символьным. Кстати,

числовое представление совсем не обязательно. Может быть, даже более интересно будет рассмотреть возможности сети на символьном представлении. Не таким уж невероятным выглядит и сумматор на символах. Как вариант рассмотрим следующее представление. Назовем базовым множеством сети множество символов $A = \{x_k\}$. Поставим каждому символу в соответствие числовой код по какому-либо правилу, например код может быть равен номеру символа в базовом множестве. Тогда работа сумматора ничем не отличается от описанной выше, сумматор символьной сети обрабатывает кодовые представления и находит их сумму.

Затем в базовом множестве ищется символ, чей код равен сумме. На тот случай, если сумма окажется больше максимально возможного кода, ее можно преобразовать к остатку от деления суммы на максимально возможный код. Возможных символьных функций можно придумывать очень много, вопрос лишь в том, будут ли они соответствовать какой-либо реальности. Но это вопрос, приложимый к любому математическому результату.

Звезды Гроссберга

Моделей нейронных сетей с различными свойствами, ориентированных на ту или иную задачу, довольно много, и для описания даже базовых идей, если делать это детально и обстоятельно, можно написать отдельную книгу. Здесь и сейчас задача создания энциклопедии не стоит, однако, как говорится, «бог любит троицу», поэтому завершим перечень примеров моделью, возникшей на заре теории, так называемыми «звездами Гроссберга». В каком-то смысле сети Кохонена можно считать обобщением идеи звезд. Обе конфигурации настраиваются на определенные образы, но если сеть Кохонена может быть настроена на N образов, то звезда Гроссберга – на один.

Звезда – это сеть со множеством входов и одним выходом. На вход подаются сигналы X_i, на выходе – один сигнал Y. Входные нейроны снабжены весами W_i. Выходной сигнал считается как среднезвешенная сумма. Формула такой суммы в тексте уже использовалась.

На вход звезды поступают немного отличающиеся сигналы. Отличающиеся в силу того, что они представляют собой описание различных экземпляров объекта, а отличаются они немного в силу того, что это разные экземпляры одного и того же объекта. Настройка звезды на образ объекта происходит подстройкой весов по формуле:

$$W_t = W_i + \alpha(X_i - W_i).$$

Здесь величина α – коэффициент обучаемости звезды, со значением от нуля до единицы. После нескольких сеансов звезда настраивается на свой объект, и если отключить возможность перестройки весов, то «свой объект» звезда будет распознавать достаточно уверенно, естественно, если сообщество входных нейронов было достаточно велико. Если включить режим обучения (опять разрешить звезде изменять веса), то она после нескольких сеансов перестроится на другой объект. Но, конечно, если звезда на входе будет получать образы различных объектов, то никакого обучения не получится.

В заключение

Мы рассмотрели примеры нейронных сетей на задачах распознавания объектов. И сегодня теория распознавания представляет собой главное поле применения сетей. Наверное, так будет не всегда. Человеческий мозг являет нам пример нейронной сети, обладающей универсальными навыками к самообучению. И будем надеяться, что человеческой науке с развитием элементной базы удастся повторить успех эволюции. Но сама по себе возможность создавать совершенные нейроны еще не будет означать появления совершенного искусственного интеллекта. В этом вопросе, пожалуй, более важную роль играют вопросы нейронной организации. Понять, каким образом группа нейронов может соответствовать интеллектуальной задаче, может быть еще более сложно, чем создать нейрон. Но возможности открываются действительно колоссальные. Если природа за миллиарды лет создала один тип нейронов, то мы, люди, уловив принцип, сможем создавать функционально разные нейроны, с заданными свойствами, скоростью обработки информации, намного превышающей возможности человеческого мозга. Но пока самые лучшие нейронные сети, созданные человеческим мозгом, даже близко не приближаются по возможностям к своему создателю.