МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)

Солдатова О. П.

Нейроинформатика

Учебное пособие

Самара

2013

СОДЕРЖАНИЕ

1. Введение в нейронные сети

1.1. Основные свойства нейронных сетей

Исследования по искусственным нейронным сетям связаны с тем, что способ обработки информации человеческим мозгом принципиально отличается от методов, применяемых обычными цифровыми компьютерами. Мозг представляет собой чрезвычайно *сложный*, *нелинейный, параллельный* компьютер. Он обладает способностью организовывать свои структурные компоненты, называемые *нейронами,* так, чтобы они могли выполнить конкретные задачи (такие как распознавание образов, обработку сигналов органов чувств, моторные функции) во много раз быстрее, чем могут позволить самые быстродействующие компьютеры. Мозг имеет совершенную структуру, позволяющую строить собственные правила на основе опыта. Опыт накапливается с течением времени.

Понятие развития нейронов мозга связано с понятием *пластичности* мозга – способностью настройки нервной системы в соответствии с окружающей средой. Аналогично в искусственных нейронных сетях производится настройка искусственных нейронов и формируется структура нейронной сети. В общем случае нейронная сеть представляет машину, моделирующую способ обработки мозгом конкретной задачи. Эта сеть обычно реализуется с помощью электронных компонентов или моделируется программой.

Таким образом, можно дать следующее определение нейронных сетей, выступающих в роли адаптивной машины [5]:

нейронная сеть – это громадный распределенный параллельный процессор, состоящий из элементарных единиц обработки информации, накапливающих экспериментальные знания и представляющих их для последующей обработки. Нейронная сеть сходна с мозгом с двух точек зрения:

- *знания поступают в нейронную сеть из окружающей среды и используются в процессе обучения;*

- *для накопления знаний применяются связи между нейронами, называемые синаптическими весами.*

Процедура настройки синаптических весов называется *алгоритмом обучения.*

Наиболее существенными свойствами нейронных сетей являются:

1. *Нелинейность*. Поскольку искусственные нейроны могут быть линейными и нелинейными, то нейронные сети позволяют воспроизводить сложные зависимости, как линейные, так и нелинейные. Нейронные сети реализуют нелинейность особого вида, так как она распределена по сети. Кроме того, нейронные сети справляются с "проклятием размерности", которое не позволяет моделировать линейные зависимости в случае большого числа переменных.
2. *Параллельная обработка информации.* Благодаря этой способности при большом количестве межнейронных связей достигается значительное ускорение процесса обработки информации. Во многих ситуациях становится возможной обработка сигналов в реальном масштабе времени.
3. *Обучение на примерах*. Одной из популярных парадигм обучения является *обучение с учителем*. Такой способ обучения предполагает изменение синаптических весов на основе набора *учебных примеров*. Каждый пример состоит из входного сигнала и соответствующего ему *ожидаемого* выходного сигнала. Нейронная сеть модифицирует синаптические веса для минимизации разности ожидаемого выходного сигнала и *реального* выходного сигнала, формируемого нейронной сетью. Таким образом, нейронная сеть обучается на примерах, составляя таблицу соответствий вход-выход для конкретной задачи.
4. *Адаптивность* (adaptivity). Нейронные сети обладают способностью адаптировать свои синаптические веса к изменениям окружающей среды. Нейронные сети могут быть легко переучены для

работы в *нестационарной* среде. Для того, чтобы использовать все достоинства адаптивности, основные параметры системы должны быть достаточно стабильными, чтобы не учитывать внешние помехи, и достаточно гибкими, чтобы обеспечить реакцию на существенные изменения среды.

5. *Нечувствительность к ошибкам* (fault tolerance). Очень большое количество межнейронных соединений приводит к тому, что сеть становится нечувствительной к ошибкам, возникающим в отдельных контактах. Функции поврежденных соединений принимают на себя другие элементы, в результате в деятельности сети не наблюдаются заметные нарушения. Только серьезные повреждения структуры нейронных сети существенно влияют на ее работоспособность.
6. *Способность к обобщению полученных знаний*. Сеть обладает чертами так называемого искусственного интеллекта. Натренированная на ограниченном множестве обучающих примеров, она обобщает накопленную информацию и вырабатывает ожидаемую реакцию применительно к данным, не обрабатывавшимся в процессе обучения.
7. *Единообразие анализа и проектирования.* Нейронные сети являются универсальным механизмом обработки информации. Одно и тоже проектное решение нейронной сети может быть использовано в разных предметных областях. Это свойство проявляется из-за нескольких причин:
    - нейроны являются стандартными составными частями любой нейронной сети;
    - можно использовать одни и те же алгоритмы обучения в различных нейросетевых приложениях;
    - на основе интеграции целых модулей могут быть построены модульные сети.

Наличие перечисленных свойств вызвало в последние годы огромный рост интереса к нейронным сетям и существенный прогресс в их исследовании.

Искусственные нейронные сети используются для аппроксимации функций, классификации и распознавания образов, прогнозирования, идентификации и оценивания и ассоциативного управления.

1.2. Биологические основы нейронных сетей

Искусственные нейронные сети возникли на основе знаний о функционировании нервной системы живых существ.

Мозг представляется сетью нервных клеток. Он получает информацию, анализирует ее и выдает соответствующие решения. Нервная клетка, сокращенно называемая *нейроном,* является основным элементом нервной системы. У нейрона есть тело, называемое *сомой*, внутри которого располагается *ядро*. Из сомы нейрона выходят отростки двух видов: многочисленные тонкие, густо ветвящиеся *дендриты* и более толстый, расщепляющийся на многочисленные нервные окончания – *колатералы, аксон* (рис.1.1).

Выходной сигнал клетки передается через аксон при помощи колатералов. Колатералы контактируют с сомой и дендритами других нейронов, образуя каналы связи выходных сигналов клетки с входами других клеток, которые называются *синапсами*. Синапсы могут находится как на дендритах, так и непосредственно в теле клетки.

Передача сигналов внутри нервной системы – это очень сложный электрохимический процесс. С большим упрощением можно считать, что передача нервного импульса между двумя клетками основана на выделении особых химических субстанций, называемых *нейромедиаторами*, которые формируются под влиянием поступающих от синапсов раздражителей.

Эти субстанции воздействуют на клеточную мембрану, вызывая изменение ее энергетического потенциала, причем величина этого изменения пропорциональна количеству нейромедиатора, попадающего на мембрану.

Синапсы отличаются друг от друга размерами и возможностями концентрации нейромедиатора.

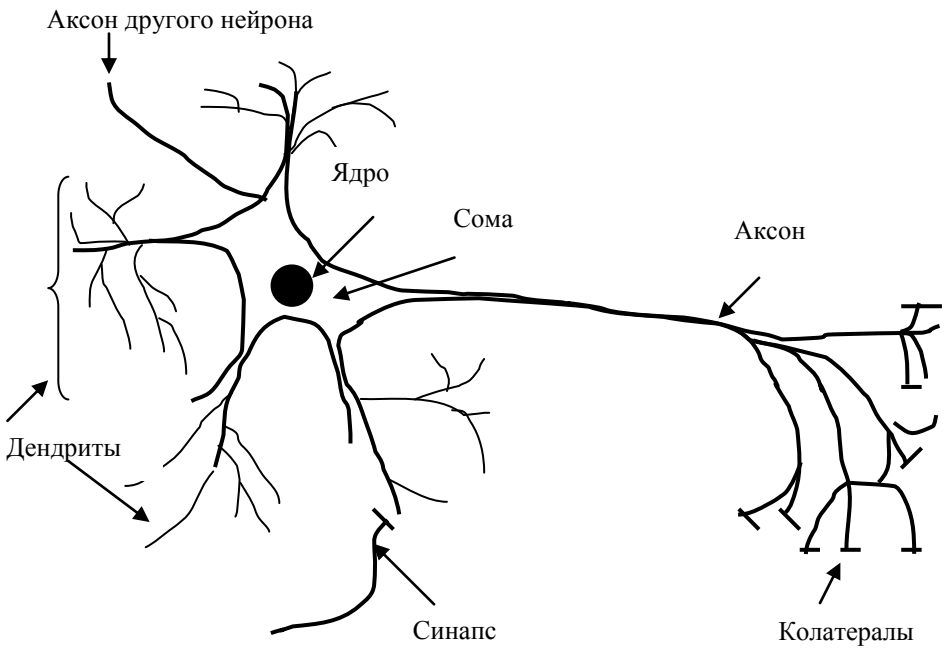


Рис. 1.1 Упрощенная структура биологической нервной клетки

Поэтому импульсы одинаковой величины, поступающие на входы нервной клетки через различные синапсы, могут в разной степени изменять ее энергетический потенциал. Мерой изменения потенциала считается уровень поляризации мембраны, зависящий от суммарного количества нейромедиатора, выделенного на всех синапсах.

В результате поступления входных импульсов на конкретные синапсы происходит изменение электрического потенциала клетки. Если отклонение от состояния электрического равновесия невелико, клетка возвращается в исходное состояние и на ее выходе сигнал не регистрируется. В этом случае считается, что уровень изменения потенциала ниже порога ее срабатывания. Если суммарное изменение потенциала превысила порог активации клетки, значение выходного сигнала начинает нарастать, приобретая характер нервного импульса, пересылаемого аксоном на другие нейроны, подключенные к данной клетке (рис.1.2). Величина этого сигнала не зависит от степени превышения порога срабатывания.

Количество взаимодействующих друг с другом нервных клеток в человеческом мозге оценивается, как 10^{11} . Каждая нервная клетка выполняет функцию суммирования весовых коэффициентов входных сигналов и сравнивает полученную сумму с пороговым значением. Каждый нейрон имеет свои веса и свои пороговые значения. Громадное количество нейронов и межнейронных связей (до 1000 входов в каждый нейрон) приводит к тому, что ошибка в срабатывании отдельного нейрона остается незаметной в общей массе взаимодействующих клеток.

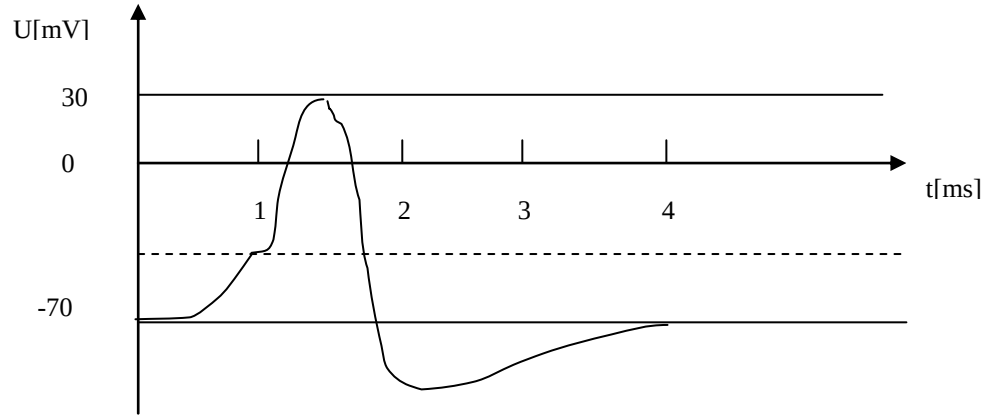


Рис. 1.2 Типичная форма нервного импульса

Следует отметить, что ни одна современная технология не позволяет построить искусственную нейронную сеть, близкую по масштабам к нейронной сети мозга. Однако изучение и копирование биологических нервных систем, позволяет надеяться на создание нового поколения электронных устройств, имеющих аналогичные характеристики.

1.3. Модель МакКаллока - Питса

Из приведенных выше рассуждений следует, что каждый нейрон суммирует с соответствующими весами сигналы, приходящие от других нейронов, выполняет нелинейную решающую функцию и передает результат связанным с ним другим нейронам. В простейших моделях нейронов выходной сигнал принимает двоичные значения: 0 или 1. Значение 1 соответствует превышению порогового уровня, значение 0 – в противном случае. Одна из

первых моделей нейрона была предложена Дж. МакКаллоком и У. Питсом в 1943 году [8]. Структурная схема этой модели представлена на рис. 1.3.

Сигналы $\boldsymbol{x_j}$ на входе синапсов j ($j = 1,2,\ldots,N$), связанные с нейроном $\boldsymbol{i}$, суммируются с учетом соответствующих синаптических весов w_{ij} (первый индекс относится к нейрону, а второй к синапсу), после чего результат сравнивается с пороговым значением w_{i0} . Пороговое значение отражает увеличение или уменьшение входного сигнала, подаваемого на функцию активации. Выходной сигнал нейрона $\boldsymbol{y_i}$ определяется при этом зависимостью

$$y_j = f\left(\sum_{J=1}^{N} w_{ij} x_j(t) + w_{i0}\right) \qquad (1.1)$$

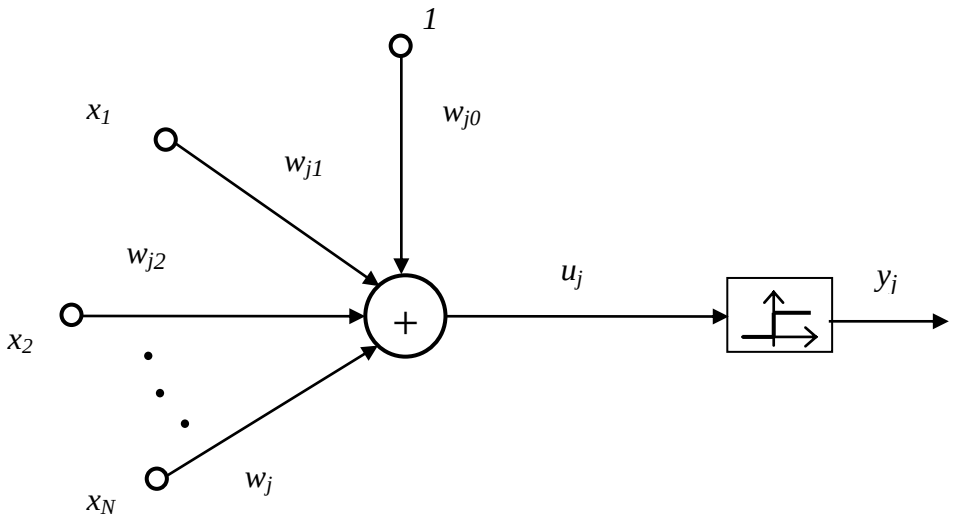


Рис. 1.3 Модель нейрона МакКаллока-Питса

Аргументом функции выступает суммарный сигнал

$$u_i = \sum_{j=1}^{N} w_{ij} x_j(t) + w_{i0}. \qquad (1.2)$$

Коэффициенты w_{ij} в формуле (1.1) представляют веса синапсов. Положительные значения w_{ij} соответствует синапсам, повышающим потенциал, отрицательные значения – синапсам, понижающим потенциал, $w_{ij} = 0$ свидетельствует об отсутствии связи между i-м и j-м нейронами.

Модель МакКаллока – Питса – это дискретная модель, в которой состояние нейрона в момент *(t+1)* рассчитывается по значению его входных сигналов в момент времени *t*.

Функция $f(u_i)$ называется функцией активации. В модели МакКаллока – Питса это пороговая функция вида:

$$f(u) = \begin{cases} 1 \ \ для\, u > 0 \\ 0 \ \ для\, u \le 0 \end{cases}. \qquad (1.3)$$

В общем случае эта функция активации описывается следующим выражением:

$$\mathrm{sgn}(x) = \begin{cases} b, \ если \ x < 0; \\ c, \ если \ x \ge 0, \end{cases} \qquad (1.4)$$

где b и c – некоторые постоянные. На практике чаще всего используют две пары постоянных b и c: первая (-1,1); вторая – (0,1). Первая пара коэффициентов определяет так называемую симметричную пороговую функцию, вторая – смещенную.

1.4. Персептрон

Ф. Розенблатт в 1958 году ввел понятие персептрона как первой модели обучения с учителем [11]. Обучение персептрона требует наличие учителя и состоит в таком подборе весов w_{ij} ,чтобы выходной сигнал y_i был наиболее близок к заданному значению d_i. При таком способе обучения, каждой обучающей выборке, представленной вектором $\boldsymbol{x}$ поставлено в соответствии ожидаемое значение d_i на выходе i-го нейрона.

Наиболее популярный метод обучения персептрона, называемый правилом персептрона, состоит в подборе весовых коэффициентов по следующему алгоритму:

- при первоначально выбранных (как правило, случайным образом) значениях весов w_{ij} на вход нейрона подается обучающий вектор x и рассчитывается значение выходного сигнала y_i. По результатам

сравнения значения y_i с заданным значением d_i уточняются значения весов;

- если y_i совпадает с ожидаемым значением d_i, то весовые коэффициенты w_{ij} не изменяются;
- если $y_i=0$, соответствующее значение $d_i=1$, то значения весов уточняются по формуле $w_{ij}(t+1)=w_{ij}(t)+x_j$, где *(t+1)* – это номер текущего цикла, а t – номер предыдущего цикла;
- если $y_i=1$, соответствующее значение $d_i=0$, то значения весов уточняются по формуле $w_{ij}(t+1)=w_{ij}(t)-x_j$, *где (t+1)* – это номер текущего цикла, а t – номер предыдущего цикла;

По завершении уточнения весов представляются очередной обучающий вектор x и связанное с ним значение d_i, и значения весов уточняются заново. Этот процесс повторяется для всех обучающих выборок, пока не будут минимизированы различия между всеми значениями y_i и соответствующими им значениями d_i.

Правило персептрона представляет собой частный случай (если сигналы принимают только двоичные значения 0 и 1) предложенного позже правила Видроу-Хоффа [12], используемого для подбора весов нейронов разного типа:

$$w_{ij}(t+1)=w_{ij}(t)+\Delta w_{ij}, \qquad (1.5)$$

$$\Delta w_{ij}=x_j(d_i-y_i). \qquad (1.6)$$

Аналогичные соотношения используются при подборе веса порогового элемента w_{i0}, для которого входной сигнал всегда равен 1:

$$\Delta w_{i0}=(d_i-y_i). \qquad (1.7)$$

Минимизация различий между фактическими реакциями нейрона y_i и ожидаемыми значениями d_i может быть представлена как минимизация функции погрешности, чаще всего определяемой как минимум квадратичного отклонения:

$$E = \sum_{t=1}^{p} \left(y_i^{(t)} - d_i^{(t)} \right)^2, \qquad (1.8)$$

где p означает количество обучающих выборок. Такая минимизация для персептрона проводится по методу безградиентной оптимизации. Эффективность метода при большом количестве обучающих выборок невелика, а количество циклов обучения и длительность возрастают быстро, при чем без гарантии достижения минимума целевой функции. Устранить эти недостатки можно только в случае применения непрерывной функции активации, при которой целевая функция E также становится непрерывной, что дает возможность использовать градиентные методы минимизации.

1.5 Сигмоидальный нейрон

Нейрон сигмоидального типа имеет структуру, подобную модели МакКаллока–Питса, с той разницей, что функция активации является непрерывной и может быть выражена в виде сигмоидальной униполярной или биполярной функции [4]. Структура нейрона представлена на рис. 1.4.

Входные сигналы x_j ($j=1,2,\ldots,N$) суммируются с учетом соответствующих весов w_{ij} (сигнал поступает в направлении от узла i к узлу j) в сумматоре, после чего результат сравнивается с пороговым значением w_{i0} Выходной сигнал нейрона y_i определяется при этом зависимостью

$$y_i = f\left(\sum_{j=1}^{N} w_{ij} x_j(t) + w_{i0} \right). \qquad (1.9)$$

Аргументом функции выступает суммарный сигнал $u_i = \sum_{j=1}^{N} w_{ij} x_j(t) + w_{i0}$.

Функция $f(u_i)$, называемая функцией активации, относится к классу непрерывных, монотонно возрастающих и дифференцируемых функций. Нейрон сигмоидального типа использует сигмоидальную униполярную (логистическую) или сигмоидальную биполярную (гиперболический тангенс) функцию активации.

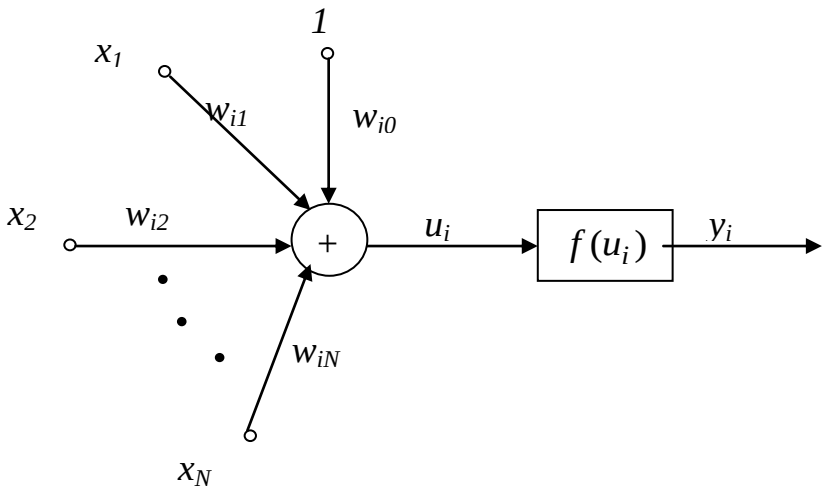


Рис. 1.4 Модель сигмоидального нейрона

Униполярная функция, как правило, представляется формулой

$$f(x) = \frac{1}{1+e^{-kx}}, \tag{1.10}$$

тогда как биполярная функция задается в виде (1.11) или (1.12):

$$f(x) = \tanh(kx). \tag{1.11}$$

$$f(x) = \frac{e^{kx} - e^{-kx}}{e^{kx} + e^{-kx}}. \tag{1.12}$$

Графики сигмоидальных функций при k=1 представлены на рис. 1.5.

Отметим, что, как правило, современные компьютеры вычисляют функцию гиперболического тангенса быстрее, чем логистическую. Другое преимущество функции гиперболического тангенса состоит в том, что она изменяется в диапазоне от –1 до +1. Часто бывает необходимо нормировать обучающий набор данных таким образом, чтобы среднее значение было равно 0 при единичном стандартном отклонении.

Такая нормировка возможна только с функцией активации, которая способна принимать отрицательные значения. И наконец, нечетная функция, такая, как гиперболический тангенс, обеспечивает более быстрое обучение, чем несимметричная логистическая функция.

В этих формулах параметр k подбирается пользователем. Его значение влияет на форму функции активации. При малых значениях k график функции достаточно пологий, по мере роста значения k крутизна графика увеличивается.

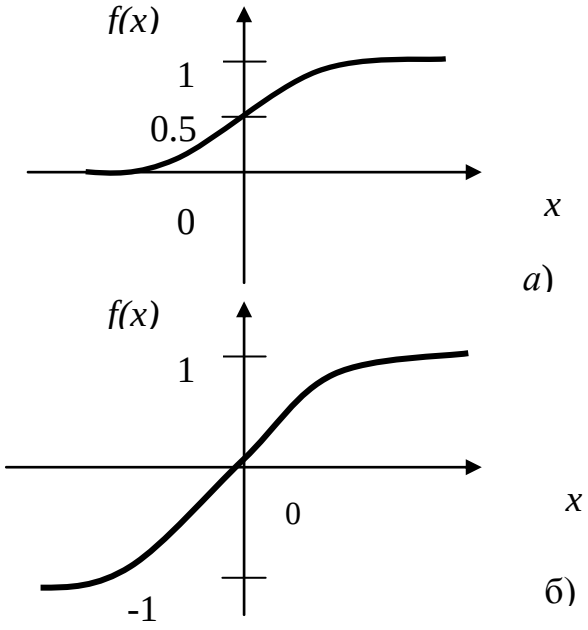


Рис. 1.5 Графики сигмоидальных функций:
а – логистическая; б – гиперболический тангенс

При $k \to \infty$ сигмоидальная функция превращается в пороговую функцию, идентичную функции активации персептрона. На практике чаще всего для упрощения используется значение k =1.

Важным свойством сигмоидальной функции является ее дифференцируемость. Для униполярной функции имеем

$$\frac{df(x)}{dx} = kf(x)(1 - f(x)), \qquad (1.13)$$

тогда как для биполярной функции

$$\frac{df(x)}{dx} = k(1 - f^2(x)). \qquad (1.14)$$

И в первом, и во втором случае график изменения производной относительно переменной *x* имеет колоколообразную форму, а его максимум соответствует значению *x=0*.

Сигмоидальный нейрон, как правило, обучается с учителем.

При обучении с учителем предполагается, что помимо входных сигналов, составляющих вектор *x*, известны также и ожидаемые выходные сигналы нейрона d_i, составляющие вектор *d*. В подобной ситуации подбор весовых коэффициентов должен быть организован так, чтобы фактические выходные сигналы нейрона y_i принимали бы значения, как можно более близкие к ожидаемым значениям d_i. Ключевым элементом процесса обучения с учителем является знание ожидаемых значений d_i выходного сигнала нейрона.

При обучении с учителем производится минимизация целевой функции, которая для единичного обучающего кортежа <*x, d*> *i*-го нейрона определяется в виде

$$E = \frac{1}{2}(y_i - d_i)^2, \tag{1.15}$$

где

$$y_i = f(u_i) = f\left(\sum_{j=0}^{N} w_{ij} x_j\right). \tag{1.16}$$

Применение непрерывной функции активации позволяет использовать при обучении градиентные алгоритмы. Проще всего реализовать метод наискорейшего спуска, в соответствии с которым уточнение вектора весов проводится в направлении отрицательного градиента целевой функции, при этом *j*-я составляющая градиента имеет вид:

$$\nabla_j E = \frac{dE}{dw_{ij}} = e_i x_j \frac{df(u_i)}{du_i} \tag{1.17}$$

$$e_i = (y_i - d_i) \tag{1.18}$$

Если ввести обозначение $\delta_i = e_i \frac{df(u_i)}{du_i}$, то значения весовых коэффициентов могут быть уточнены дискретным способом в соответствии с формулой 1.19.

$$w_{ij}(t+1) = w_{ij}(t) - \dot{\eta}\delta_i x_j \qquad (1.19)$$

В данной формуле коэффициент $\dot{\eta}$ – это коэффициент обучения, значение которого либо выбирают эмпирически из интервала (0,1), либо при помощи решения разностного уравнения, представленного формулой 1.20.

$$\frac{dw_{ij}}{dt} = -\mu\delta_i x_j \qquad (1.20)$$

В данной формуле коэффициент μ выступает в роли аналогичной значению $\dot{\eta}$.

Формулы 1.19 и 1.20 определяют алгоритм обучения.

На эффективность обучения оказывает сильное влияние подбор коэффициентов обучения. В существующих алгоритмах обучения его величина может быть задана константой или переменной, значение которой в процессе обучения изменяется адаптивным способом либо подбирается на каждом шаге по принципу направленной минимизации целевой функции от одной переменной в направлении наискорейшего уменьшения значений этой целевой функции.

Необходимо подчеркнуть, что применение градиентного метода обучения гарантирует достижение только локального минимума. В случае полимодальной целевой функции найденный локальный минимум может быть достаточно далек от глобального минимума. Для таких случаев может оказаться результативным обучение с *моментом* или *разбросом.* В этом методе процесс уточнения весов определяется не только информацией о градиенте функции, но также и фактическим трендом изменений весов. Приращение весов можно задать следующим математическим выражением:

$$\Delta w_{ij}(t+1) = -\dot{\eta}\delta_i x_j + \alpha\Delta w_{ij}(t), \qquad (1.21)$$

в котором первый член соответствует обычному методу наискорейшего спуска, а второй член (*момент*), отражает последнее изменение весов и не зависит от фактического значения градиента. Значение коэффициента момента $\acute{\alpha}$, как правило, выбирается из интервала (0,1). С ростом значения $\acute{\alpha}$ увеличивается влияние момента на подбор весов.

1.6. Нейрон типа WTA

В соответствии с принципами функционирования биологических нейронов созданы различные математические модели, которыми в большей или меньшей степени реализуются свойства природной нервной клетки. Обобщенная схема, составляющая основу большинства таких моделей, восходит к представленной на рисунке 1.3 модели МакКаллока-Питса, содержащий сумматор взвешенных входных сигналов и нелинейный блок выработки выходного сигнала нейрона, функционально зависящего от выходного сигнала сумматора. Однако, существуют и другие модели нейронов существенно отличающиеся от модели МакКаллока–Питса.

Рассмотрим более подробно нейроны типа *WTA* (*winner takes all* – победитель получает все) [4]. Эти нейроны имеют входной модуль в виде стандартного сумматора, рассчитывающего сумму входных сигналов с соответствующими весами w_{ij}. Выходной сигнал i-го сумматора определяется согласно формуле:

$$u_i = \sum_{j=0}^{N} w_{ij} x_j \qquad (1.22)$$

Группа конкурирующих между собой нейронов получает одни и те же входные сигналы x_j. Выходные сигналы нейронов u_i сравниваются между собой, и по результатам сравнения победителем признается нейрон, значение выходного сигнала у которого оказалось наибольшим. Нейрон-победитель вырабатывает на своем выходе состояние 1, а остальные нейроны переходят в состояние 0. для обучения нейронов типа *WTA* не требуется учитель. На начальном этапе случайным образом выбираются весовые коэффициенты

каждого нейрона, нормализуемые относительно 1. После подачи первого входного вектора ***x*** определяется победитель этапа. Победивший нейрон переходит в состояние 1, что позволяет провести уточнение весов его входных линий по следующему правилу:

$$\Delta w_{ij}(t+1) = w_{ij}(t) + \dot{\eta}\left(x_j - \Delta w_{ij}(t)\right) \qquad (1.23)$$

Проигравшие нейроны не изменяют свои весовые коэффициенты.

Схема соединения нейронов типа *WTA* изображена на рис.1.6.

На функционирование нейронов типа *WTA* оказывает существенное влияние нормализация входных векторов и весовых коэффициентов. Выходной сигнал *i*-го нейрона может быть описан векторным отношением:

$$u_i = w^T x = \|w\|\|x\|\cos\varphi_i \qquad (1.24)$$

Поскольку $\|w\| = \|x\| = 1$, значение выходного сигнала определяется углом между векторами *x* и *w*. Поэтому победителем оказывается нейрон, вектор весов которого оказывается наиболее близким текущему обучаемому вектору.

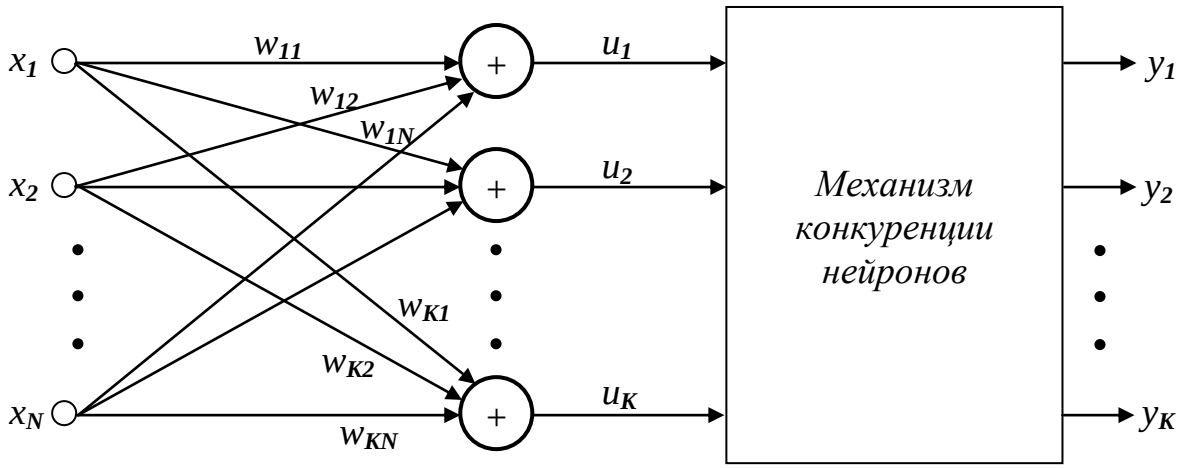


Рис. 1.6 Схема соединения нейронов типа WTA

В результате победы нейрона уточняются его весовые коэффициенты, значения которых приближаются к значениям вектора *x*. Проигравшие нейроны не изменяют свои веса. Следствием такой конкуренции становится самоорганизация процесса обучения.

1.7. Звезды Гроссберга

Рассмотрим конфигурации входных и выходных звезд Гроссберга [4]. Нейроны типа инстар и оутстар – это взаимодополняющие элементы. Инстар адаптирует веса сигналов, поступающих на сумматор нейрона, к своим входным сигналам, а оутстар согласовывает веса выходящих из нейронов связей с узлами, в которых формируются значения выходных сигналов.

Нейрон в форме входной звезды (инстар) имеет N входов $x_1, x_2, \ldots, x_N$, которым соответствуют веса $w_{i1}, w_{i2}, \ldots, w_{iN}$, и один выход y_i, являющийся взвешенной суммой входов. Входная звезда обучается выдавать сигнал на выходе всякий раз, когда на входы поступает определенный вектор. Таким образом, входная звезда является детектором совокупного состояния своих входов (рис. 1.7). Процесс обучения представляется в следующей итерационной форме:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\, y_i \left[x_j - w_{ij}(t)\right], \qquad (1.25)$$

где η – коэффициент обучения, значение которого, как правило, выбирается из интервала (0,1).

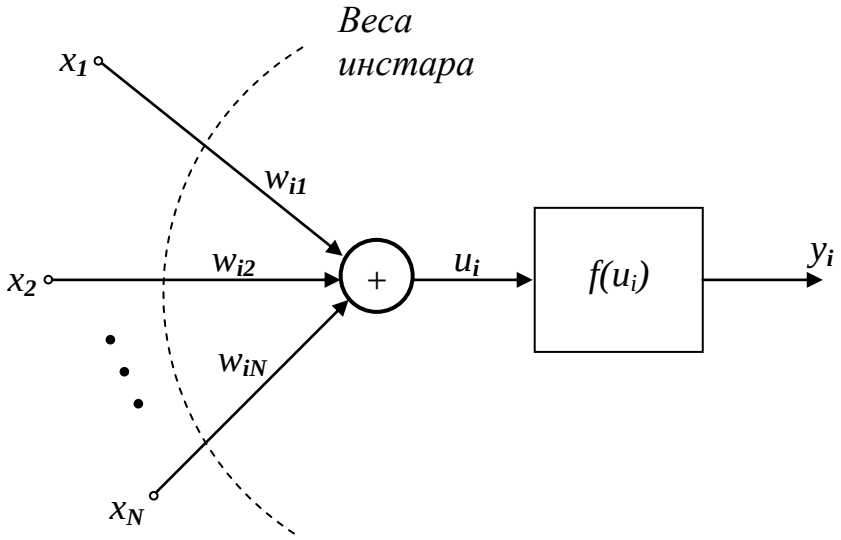


Рис. 1.7 Структурная схема входной звезды (инстара)

Выходная звезда Гроссберга (оутстар) выполняет противоположную функцию – функцию командного нейрона, выдавая на выходах определенный вектор при поступлении сигнала на вход. Структурная схема оутстара представлена на рис. 1.8.

Нейрон этого типа имеет один вход и *M* выходов с весами $w_{1i}, w_{2i}, \ldots, w_{Mi}$. *i*-ый нейрон-источник высылает свой выходной сигнал y_i взаимодействующим с ним нейронам, выходные сигналы которых обозначены y_j (j = 1, 2, …, M). Оутстар, как правило, является линейным нейроном. Обучение состоит в таком подборе его весов w_{ij}, чтобы выходные сигналы оутстара были равны ожидаемым значениям y_j взаимодействующих с ним нейронов.

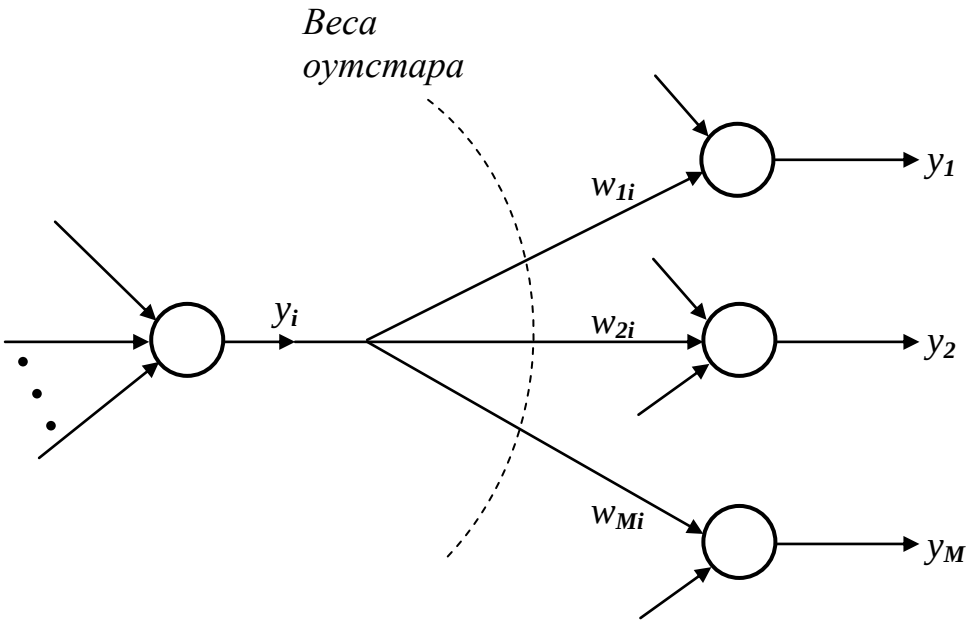


Рис. 1.8 Структурная схема выходной звезды (оутстара)

Обучение оутстара согласно правилу Гроссберга проводится в соответствии с выражением

$$w_{ji}(t+1) = w_{ji}(t) + \eta \cdot y_i \cdot (y_j - w_{ji}(t)), \qquad (1.26)$$

в котором η – это коэффициент обучения, а y_i – выходной сигнал *i*-го нейрона, выступающего в роли источника. В режиме распознавания в момент

активизации нейрона-источника оутстар будет генерировать сигналы, соответствующие ожидаемым значениям y_j.

1.8. Функции активации нейронов

Вид функции активации во многом определяет функциональные возможности нейронной сети и метод обучения этой сети. Перечень наиболее известных функций активации представлен в таблице 1.1.

Табл. 1.1

Примеры функций активации

Название	Формула	Область значений
Линейная	$f(x)=kx$	$(-\infty,\infty)$
Полулинейная	$f(x)=\begin{cases}kx, & x>0\\ 0, & x\le 0\end{cases}$	$(0,\infty)$
Линейная с насыщением	$f(x)=\begin{cases}-1, x\le -1\\ x, -1<x<1\\ 1, x\ge 1\end{cases}$	$(-1,1)$
Логистическая	$f(x)=\frac{1}{1+e^{-kx}}$	(0,1)
Гиперболический тангенс	$f(x)=\frac{e^{kx}-e^{-kx}}{e^{kx}+e^{-kx}}$	$(-1,1)$
Рациональная	$f(x)=\frac{x}{k+\lvert x\rvert}$	$(-1,1)$
Синусоидальная	$f(x)=\sin(x)$	$(-1,1)$
Экспоненциальная	$f(x)=e^{-kx}$	$(0,\infty)$
Пороговая	$f(x)=\begin{cases}0, x<0\\ 1, x\ge 0\end{cases}$	(0,1)
Модульная	$f(x)=\lvert x\rvert$	$(0,\infty)$
Знаковая (сигнатурная)	$f(s)=\begin{cases}1, x>0\\ -1, x\le 0\end{cases}$	$(-1,1)$
Квадратичная	$f(x)=x^2$	$(0,\infty)$

1.9 Контрольные вопросы и упражнения

1. Чем модель персептрона отличается от модели МакКаллока-Питса?

2. Чем модель персептрона отличается от модели сигмоидального нейрона?

3. Для каких целей используется модель нейрона типа WTA?

4. Для каких целей используются звезды Гроссберга?

5. Диапазон значений логистической функции $f(x) = \frac{1}{1+e^{-kx}}$ ограничен нулем и единицей. Покажите, что производная функции $f(x)$ описывается выражением $\frac{df(x)}{dx} = kf(x)(1 - f(x))$.

6. Областью значений гиперболического тангенса $f(x) = \frac{e^{kx} - e^{-kx}}{e^{kx} + e^{-kx}}$ является интервал от -1 до +1. Покажите, что производная функции $f(x)$ описывается выражением $\frac{df(x)}{dx} = k(1 - f^2(x))$. Предположим, что параметр наклона k – бесконечно большой. В функцию какого вида выродится $f(x)$?

7. Рассмотрим линейную с насыщением функцию $f(x)$, представленную на рис. 1.9:

а) Выпишите функциональную зависимость $f(x)$ в аналитическом виде.

б) Как изменится функция $f(x)$, если параметр *a* принять равным нулю?

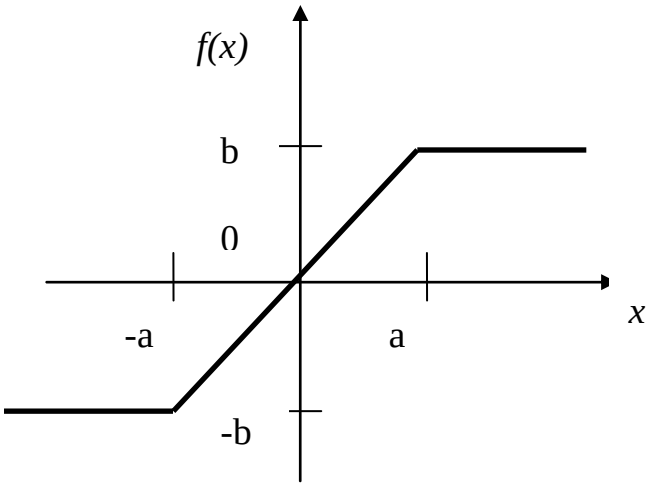


Рис. 1.9 График линейной с насыщением функции активации

2. Многослойный персептрон

2.1. Структура многослойного персептрона

В настоящее время наиболее часто используемой архитектурой нейросети является *многослойный персептрон (MLP),* который представляет собой обобщение однослойного персептрона.

Обычно сеть состоит из множества входных узлов, которые образуют *входной слой*; одного или нескольких *скрытых слоев* вычислительных нейронов и одного *выходного слоя*. Входной сигнал распространяется по сети в прямом направлении от слоя к слою. Многослойные персептроны успешно применяются для решения разнообразных сложных задач. При этом обучение с учителем выполняется с помощью такого популярного алгоритма, как *алгоритм обратного распространения ошибки*.

Многослойный персептрон имеет три отличительных признака:

1. Каждый нейрон имеет *нелинейную функцию активации*. Данная функция должна быть гладкой (то есть всюду *дифференцируемой*). Самой популярной гладкой функцией активации является сигмоидальная функция.
2. Сеть содержит один или несколько слоев *скрытых нейронов*. Эти нейроны позволяют сети обучаться решению сложных задач, последовательно извлекая наиболее важные признаки из входного вектора.
3. Сеть обладает высокой степенью *связности*, реализуемой посредством синаптических соединений.

Структура многослойного персептрона с двумя скрытыми слоями изображена на рис. 2.1 [5]. Показанная на рисунке сеть является *полносвязной*, что характерно для многослойного персептрона. Это значит, что каждый нейрон любого слоя связан со всеми нейронами предыдущего слоя. Сигнал передается по сети в прямом направлении слева направо.

Для многослойного персептрона выделяют два типа сигналов:

1. *Функциональный сигнал* – это входной сигнал сети, передаваемый в прямом направлении по всей сети. Такой сигнал достигает конца сети в виде выходного сигнала. В каждом нейроне, через который передается этот сигнал, вычисляется некоторая функция от взвешенной суммы его входов с поправкой в виде порогового элемента - единичного сигнала с весовым коэффициентом w_{io}.

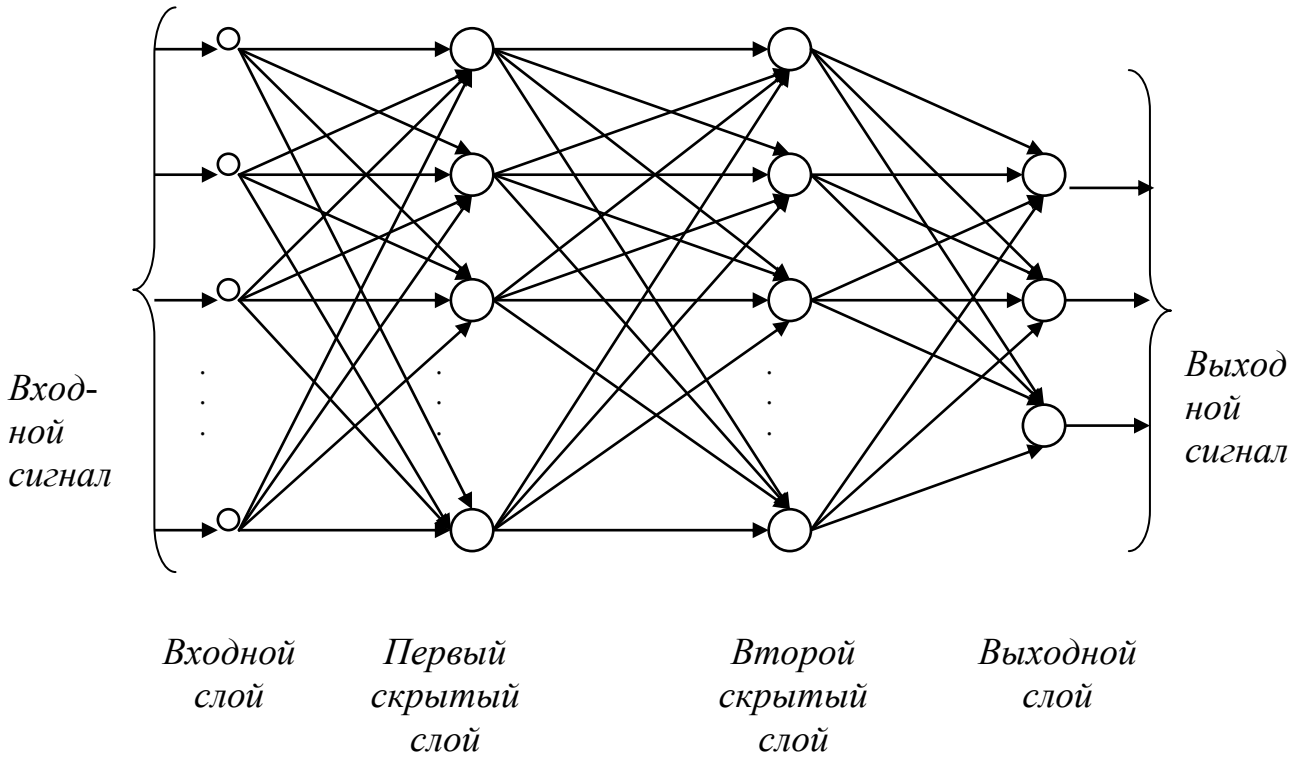


Рис. 2.1 Структура многослойного персептрона с двумя скрытыми слоями

2. *Сигнал ошибки* – берет своё начало на выходе сети и распространяется в обратном направлении от слоя к слою. Вычисляется каждым нейроном на основе функции ошибки. Представленной в той или иной форме.

Выходные нейроны составляют выходной слой сети. Остальные нейроны относятся к скрытым слоям. Первый скрытый слой получает данные из входного слоя. Результирующий сигнал первого скрытого слоя, в свою очередь, поступает на следующий скрытый слой и так далее, до самого конца сети.

В сетях подобного типа используются, в основном, сигмоидальные нейроны. Такую сеть легко можно интерпретировать как модель вход-выход, в которой веса и пороговые значения являются свободными параметрами модели. Такая сеть может моделировать функцию практически любой степени сложности, причем число слоев и число нейронов в каждом слое определяют сложность функции.

В многослойных сетях эталонные значения выходных сигналов известны, как правило, только для нейронов выходного слоя, поэтому сеть невозможно обучить, руководствуясь только величинами ошибок на выходе нейросети.

Один из вариантов решения этой проблемы – разработка учебных примеров для каждого слоя нейросети, что является очень трудоемкой операцией и не всегда осуществимо.

Второй вариант – динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный метод "тыка", несмотря на свою кажущуюся простоту, требует громоздких рутинных вычислений.

И, наконец, третий, более приемлемый вариант – распространение сигналов ошибки от выходов нейросети к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения нейросети получил название процедуры обратного распространения. Разработка алгоритма обратного распространения для определения весов в многослойном персептроне сделала эти сети наиболее популярными у исследователей и пользователей нейронных сетей.

Основными достоинствами многослойного персептрона являются простота в использовании, гарантированное получение ответа после прохода данных по слоям, хорошо апробированные и широко применяемые алгоритмы обучения, способность моделирования функции любой степени сложности.

С другой стороны, существует множество спорных вопросов при проектировании сетей прямого распространения. Например, сколько скрытых слоев необходимо для решения данной задачи, сколько следует выбрать нейронов в каждом слое, как сеть будет реагировать на данные, не включенные в обучающую выборку, и какой размер обучающей выборки необходим для достижения "хорошей" обобщающей способности сети.

2.2. Структура двухслойной сигмоидальной нейронной сети

На рис. 2.2 представлена сеть с одним скрытым слоем. Все последующие рассуждения относятся к сетям именно такого типа. Обозначения сигналов и весов также будут соответствовать этому рисунку. Веса нейронов скрытого слоя пометим верхним индексом (1), а выходного слоя – верхним индексом (2). Выходные сигналы нейронов скрытого слоя обозначим v_i *(i=0, 1, 2,..., K)*, а выходного слоя y_s *(s=1, 2, ..., M)*.

Пусть функция активации нейронов задана в сигмоидальной униполярной или биполярной форме. Для упрощения описания будем использовать расширенное обозначение входного вектора сети в виде $x = [x_0, x_1, ..., x_N]^T$, где $x_0 = 1$ соответствует единичному сигналу порогового элемента.

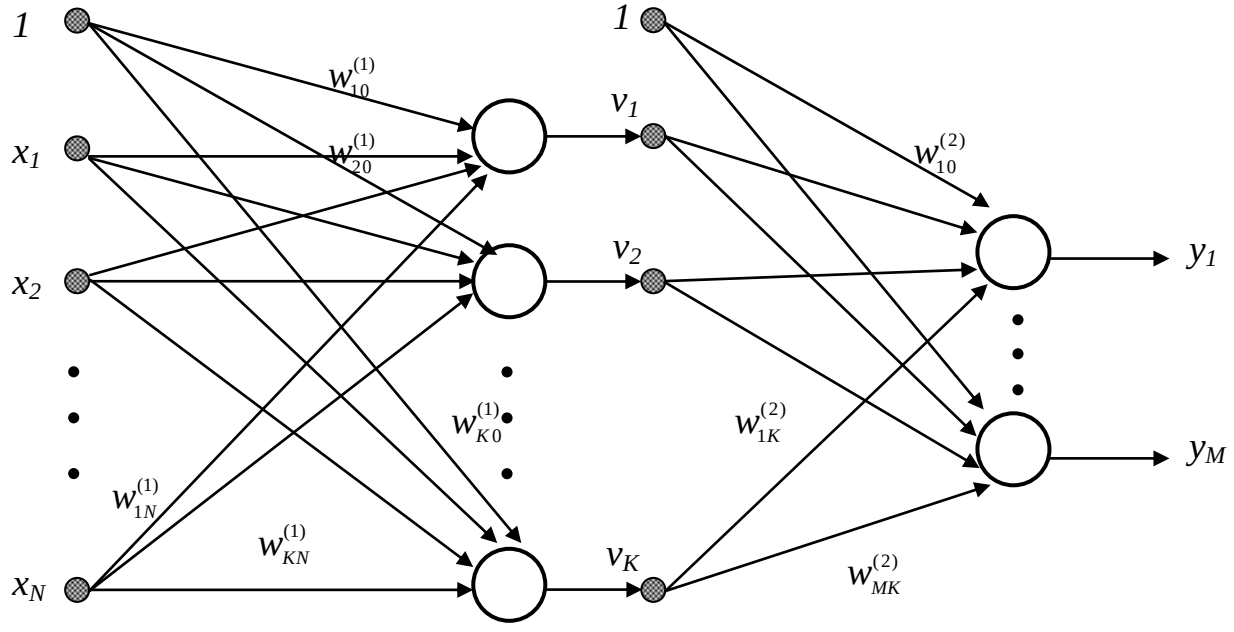


Рис. 2.2 Обобщенная структура двухслойной сигмоидальной нейронной сети

С вектором ***x*** связаны два выходных вектора сети: вектор фактических выходных сигналов $y = [y_{0,} y_1,...,y_M]^T$ и вектор ожидаемых выходных сигналов $d = [d_0, d_1,...,d_M]^T$.

Цель обучения состоит в подборе таких значений весов $w_{ij}^{(1)}$ и $w_{si}^{(2)}$ для двух слоев сети, чтобы при заданном входном векторе *x* получить на выходе значения сигналов y_s, которые с требуемой точностью будут совпадать с ожидаемыми значениями d_s для *s=1, 2, ..., M.*

Если рассматривать единичный сигнал порогового элемента как один из компонентов входного вектора ***x,*** то веса пороговых элементов можно добавить в векторы весов соответствующих нейронов обоих слоев. При таком подходе выходной сигнал *i-го* нейрона скрытого слоя удается описать функцией

$$v_i = f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_j\right), \qquad (2.1)$$

в которой индекс 0 соответствует сигналу и весам пороговых элементов, причем $v_0 \equiv 1,\ x_0 \equiv 1$. В выходном слое *s-ый* нейрон вырабатывает выходной сигнал, определяемый как

$$y_s = f\left(\sum_{i=0}^{K} w_{si}^{(2)} v_i\right) = f\left(\sum_{i=0}^{K} w_{si}^{(2)} f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_j\right)\right). \qquad (2.2)$$

Из формулы (2.2) следует, что на значение выходного сигнала влияют веса обоих слоев, тогда как сигналы, вырабатываемые в скрытом слое, не зависят от весов выходного слоя.

Для того чтобы сеть можно было применять в дальнейшем, ее прежде надо обучить на полученных ранее данных, для которых известны и значения входных параметров, и правильные ответы на них. Это обучение состоит в подборе весов межнейронных связей, обеспечивающих наибольшую близость ответов сети к известным правильным ответам.

Определение числа промежуточных слоев и числа элементов в них является важным вопросом при конструировании многослойного персептрона.

2.3. Методы обучения многослойного персептрона

2.3.1. Основные положения градиентных алгоритмов обучения сети

Задачу обучения нейронной сети будем рассматривать, как требование минимизировать априори определенную целевую функцию *E(w)*. При таком подходе можно применять для обучения алгоритмы, которые в теории оптимизации считаются наиболее эффективными. К ним, без сомнения, относятся градиентные алгоритмы, чью основу составляет выявление градиента целевой функции. Они связаны с разложением целевой функции *E(w)* в ряд Тейлора в ближайшей окрестности точки имеющегося решения *w*. В случае целевой функции от многих переменных ($w=[w_1, w_2, ..., w_n]^T$) такое представление связывается с окрестностью ранее определенной точки (в частности, при старте алгоритма это исходная точка w_0) в направлении *p*. Подобное разложение описывается универсальной формулой вида

$$E(w+p) = E(w) + [g(w)]^T p + \frac{1}{2} p^T H(w) p + \ldots, \qquad (2.3)$$

где $g(w) = \nabla E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n} \right]^T$ - это вектор градиента, а симметричная квадратная матрица

$$H(w) = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial w_n \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_n \partial w_n} \end{bmatrix}$$

является матрицей производных второго порядка, называемой гессианом.

В выражении (2.3) *p* играет роль направляющего вектора, зависящего от фактических значений вектора *w*. На практике чаще всего рассчитываются три первых члена ряда (2.3), а последующие члены ряда просто игнорируются. При этом зависимость (2.3) может считаться квадратичным приближением целевой функции *E(w)* в ближайшей окрестности найденной точки *w* с точностью, равной локальной погрешности отсеченной части $O(h^3)$, где *h*= ||*p*||. Для

упрощения описания значения переменных, полученных в *t-ом* цикле, будем записывать с нижним индексом *t*. Точкой решения $w = w_t$ будем считать точку, в которой достигается минимум целевой функции *E(w)* и $g(w_t) = 0$, а гессиан $H(w_t))$ является положительно определенным.

В процессе поиска минимального значения целевой функции направление поиска *p* и шаг *h* подбираются таким образом, чтобы для каждой очередной точки $w_{t+1} = w_t + \eta_t p_t$ выполнялось условие $E(w_{t+1}) < E(w_t)$. Поиск минимума продолжается, пока норма градиента не упадет ниже априори заданного значения допустимой погрешности либо пока не будет превышено максимальное время вычислений (количество итераций).

Универсальный оптимизационный алгоритм обучения нейронной сети можно представить в следующем виде (будем считать, что начальное значение оптимизируемого вектора известно и составляет $w_t = w_0$) [4]:

1. Проверка сходимости и оптимальности текущего решения w_t. Если точка w_t отвечает градиентным условиям остановки процесса – завершение вычислений. В противном случае перейти к п.2.
2. Определение вектора направления оптимизации p_t для точки w_t.
3. Выбор величины шага η_t в направлении p_t, при котором выполняется условие $E(w_t + \eta_t p_t) < E(w_t)$.
4. Определение нового решения $w_{t+1} = w_t + \eta_t p_t$, а также соответствующих ему значений $E(w_t)$ и $g(w_t)$, а если требуется – то и $H(w_t)$, и возврат к п.1.

2.3.2. Подбор коэффициента обучения

После правильно выбранного направления p_t в градиентных алгоритмах обучения следует определить на нем новую точку решения w_{t+1}, в которой будет выполняться условие $E(w_{t+1}) < E(w_t)$. Необходимо подобрать такое значение η_t, чтобы новое решение $w_{t+1} = w_t + \eta_t p_t$ лежало как можно ближе к минимуму функции *E(w)* в направлении p_t. Грамотный подбор коэффициента η_t оказывает огромное влияние на сходимость алгоритма оптимизации к

минимуму целевой функции. Чем сильнее величина η_t отличается от значения, при котором *E(w)* достигает минимума в выбранном направлении p_t, тем большее количество итераций потребуется для поиска оптимального решения. Слишком малое значение η_t не позволяет минимизировать целевую функцию за один шаг и вызывает необходимость повторно двигаться в том же направлении. Слишком большой шаг приводит к «перепрыгиванию» через минимум функции и фактически заставляет возвращаться к нему.

Существуют различные способы подбора значений η_t, называемого в теории нейронных сетей коэффициентом обучения. Простейших из них основан на фиксации постоянного значения η_t на весь период оптимизации. Этот способ практически используется только совместно с методом наискорейшего спуска. Он имеет низкую эффективность, поскольку значение коэффициента обучения никак не зависит от вектора фактического градиента и, следовательно, от направления *p* на данной итерации. Величина η подбирается, как правило, раздельно для каждого слоя сети с использованием различных эмпирических зависимостей. Один из походов состоит в определении минимального значения η для каждого слоя по формуле

$$\eta \le \min\left(\frac{1}{n_i}\right), \qquad (2.4)$$

где n_i обозначает количество входов *i-го* нейрона в слое.

Наиболее эффективный, хотя и наиболее сложный, метод подбора коэффициента обучения связан с направленной минимизацией целевой функции в выбранном заранее направлении p_t. Необходимо так подобрать скалярное значение η_t, чтобы новое решение $w_{t+1} = w_t + \eta_t p_t$ соответствовало минимуму целевой функции в данном направлении p_t. В действительности получаемое решение w_{t+1} только с определенным приближением может считаться настоящим минимумом. Это результат компромисса между объемом вычислений и влиянием величины η_t на сходимость алгоритма.

Опишем метод аппроксимации целевой функции *E(w)* в предварительно выбранном направлении p_t с последующим расчетом минимума, получаемого таким образом, функции одной переменной η. Выберем для аппроксимации многочлен второго порядка вида

$$E(w) \to P_2(\eta) = a_2\eta^2 + a_1\eta + a_0, \qquad (2.5)$$

где a_2, a_1 и a_0 обозначены коэффициенты, определяемые в каждом цикле оптимизации. Выражение (2.5) – это многочлен P_2 одной скалярной переменной η. Если для расчета входящих в P_2 коэффициентов используются три произвольные точки w_1, w_2 и w_3, лежащие в направлении p_t, т.е. $w_1 = w + \eta_1 p_t$, $w_2 = w + \eta_2 p_t$, $w_3 = w + \eta_3 p_t$ (в этом выражении w обозначено предыдущее решение), а соответствующие этим точкам значения целевой функции *E(w)* обозначены $E_1 = E(w_1)$, $E_2 = E(w_2)$, $E_3 = E(w_3)$, то

$$P_2(\eta_1) = E_1,\ P_2(\eta_2) = E_2,\ P_2(\eta_3) = E_3. \qquad (2.6)$$

Коэффициенты a_2, a_1 и a_0 многочлена P_2 рассчитываются в соответствии с системой линейных уравнений, описываемых в (2.6). Для определения минимума этого многочлена его производная $\frac{dP_2}{d\eta} = 2a_2\eta + a_1$ приравнивается к нулю, что позволяет получить значение η в виде $\eta_{\min} = -\frac{a_1}{2a_2}$. После подстановки выражений для E_1, E_2 и E_3 в формулу расчета $\eta_{\min}$ получаем:

$$\eta_{\min} = \eta_2 - \frac{1}{2}\frac{(\eta_2 - \eta_1)^2(E_2 - E_1) - (\eta_2 - \eta_3)^2(E_2 - E_1)}{(\eta_2 - \eta_1)(E_2 - E_3) - (\eta_2 - \eta_3)(E_2 - E_1)} \qquad (2.7)$$

2.3.3. Алгоритм обратного распространения ошибки

Алгоритм обратного распространения ошибки определяет стратегию подбора весов многослойной сети с применением градиентных методов оптимизации. В настоящее время считается одним из наиболее эффективных алгоритмов обучения многослойной сети. При обучении ставится задача минимизации целевой функции, формируемой, как правило, в виде квадратичной суммы разностей между фактическими и ожидаемыми

значениями выходных сигналов, которая для P обучающих выборок определяется по формуле:

$$E(w) = \frac{1}{2}\sum_{t=1}^{P}\sum_{s=1}^{M}\left(y_s^{(t)} - d_s^{(t)}\right)^2 \qquad (2.8)$$

В случае единичной обучающей выборки *(x,d)* целевая функция имеет вид:

$$E(w) = \frac{1}{2}\sum_{s=1}^{M}\left(y_s - d_s\right)^2 \qquad (2.9)$$

Уточнение весов может проводиться после предъявления каждой обучающей выборки (так называемый режим «онлайн»), при этом используется целевая функция вида (2.9), либо однократно после предъявления всех обучающих выборок (режим «оффлайн»), при этом используется целевая функция вида (2.8). В последующем изложении используется целевая функция вида (2.9).

Для упрощения можно считать, что цель обучения состоит в таком определении значений весов нейронов каждого слоя сети, чтобы при заданном входном векторе получить на выходе значения сигналов y_s, совпадающие с требуемой точностью с ожидаемыми значениями d_s при s= *1, 2,…,M.*

Обучение сети с использованием алгоритма обратного распространения ошибки проводится в несколько этапов.

На первом из них предъявляется обучающая выборка *x* и рассчитываются значения сигналов соответствующих нейронов сети. При заданном векторе *x* определяются вначале значения выходных сигналов v_i скрытого слоя, а затем значения y_s выходного слоя. Для расчета применяются формулы (2.1) и (2.2). После получения значений выходных сигналов y_s становится возможным рассчитать фактическое значение целевой функции ошибки *E(w)***.**

На втором этапе минимизируется значение этой функции.

Так как целевая функция непрерывна, то наиболее эффективными методами обучения оказываются градиентные алгоритмы, согласно которым уточнение вектора весов (обучение) производится по формуле:

$$w(t+1) = w(t) + \Delta w, \tag{2.10}$$

$$\text{где } \Delta w = \eta\, p(w), \tag{2.11}$$

η - коэффициент обучения, а *p(w)* – направление в многомерном пространстве *w*. В алгоритме обратного распространения ошибки *p(w)* определяется как частная производная $\cdot \frac{\partial E}{\partial w_{ij}}$, взятая со знаком минус.

Обучение многослойной сети с применением градиентных методов требует определения вектора градиента относительно весов всех слоев сети, что необходимо для правильного выбора направления *p(w)*. Эта задача имеет очевидное решение только для весов выходного слоя. Для других слоев используется алгоритм обратного распространения ошибки, который определяется следующим образом [4]:

1. Подать на вход сети вектор ***x*** и рассчитать значения выходных сигналов нейронов скрытых слоев и выходного слоя, а также соответствующие производные $\frac{df(u_i^{(1)})}{du_i^{(1)}}, \frac{df(u_i^{(2)})}{du_i^{(2)}}, \ldots, \frac{df(u_i^{(m)})}{du_i^{(m)}}$ функций активации каждого слоя (m – количество слоев).
2. Создать сеть обратного распространения ошибок путем изменения направления передачи сигналов, замены функций активации их производными и подачи на бывший выход сети в качестве входного сигнала разности между фактическими y_s и ожидаемыми d_s значениями.
3. Уточнить веса по формулам (2.10) и (2.11) на основе результатов, полученных в п.1 и п.2 для исходной сети и для сети обратного распространения ошибки.
4. Пункты 1, 2, 3 повторить для всех обучающих выборок, вплоть до выполнения условия остановки: норма градиента станет меньше заданного значения ε, характеризующего точность обучения.

Рассмотрим основные расчетные формулы для сети с одним скрытым слоем, представленной на рисунке 2.2. Используется сигмоидальная функция

активации, при этом в случае гиперболического тангенса производная функции активации равна

$$\frac{df(u)}{du} = 1 - f^2(u) \qquad (2.12)$$

В случае логистической функции производная равна

$$\frac{df(u)}{du} = f(u) \cdot (1 - f(u)) \qquad (2.13)$$

В формулах (2.12) и (2.13) под переменной *u* будем понимать выходные сигналы сумматоров нейронов скрытого или выходного слоя, представленных формулами (2.14) и (2.15).

$$u_i^{(1)} = \sum_{j=0}^{N} w_{ij}^{(1)} x_j \qquad (2.14)$$

$$u_s^{(2)} = \sum_{i=0}^{K} w_{si}^{(2)} v_i \qquad (2.15)$$

Уточнение весовых коэффициентов будем проводить после предъявления каждой обучающей выборки. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(m)} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \qquad (2.16)$$

Здесь w_{ij} – весовой коэффициент синаптической связи, соединяющей *i-ый* нейрон слоя *m-1* с *j-ым* нейроном слоя *m*, η – коэффициент обучения, $0<\eta<1$.

С учетом принятых на рисунке 2.2 обозначений целевая функция для выходного слоя нейронов определяется следующим образом:

$$E = \frac{1}{2}\sum_{s=1}^{M}\left[f\left(\sum_{i=o}^{K} w_{si}^{(2)} v_i\right) - d_s \right]^2 = \frac{1}{2}\sum_{s=1}^{M}\left[f\left(\sum_{i=0}^{K} w_{si}^{(2)} f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_i\right)\right) - d_s \right]^2 \qquad (2.17)$$

$$\frac{\partial E}{\partial w_{si}^{(2)}} = (y_s - d_s) \cdot \frac{df(u_s^{(2)})}{du_s^{(2)}} \cdot v_i \qquad (2.18)$$

Здесь под y_s, как и раньше, подразумевается выход s –го нейрона.

Если ввести обозначение $\delta_s^{(2)} = (y_s - d_{si}) \frac{df(u_s^{(2)})}{du_{si}^{(2)}}$, то соотношение (2.18) можно представить в виде:

$$\frac{\partial E}{\partial w_{si}^{(2)}} = \delta_s^{(2)} v_i \qquad (2.19)$$

Компоненты градиента относительно нейронов скрытого слоя описываются более сложной зависимостью:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{s=1}^{M} (y_s - d_s) \cdot \frac{dy_s}{dv_i} \cdot \frac{dv_i}{dw_{ij}^{(1)}} \qquad (2.20)$$

В другом виде эта зависимость может быть выражена формулой:

Если ввести обозначение

$$\delta_i^{(1)} = \sum_{s=1}^{M} (y_s - d_s) \cdot \frac{df(u_s^{(2)})}{du_s^{(2)}} \cdot w_{si}^{(2)} \frac{df(u_s^{(1)})}{du_i^{(1)}}, \qquad (2.21)$$

то получим выражение, определяющее компоненты градиента относительно весов нейронов скрытого слоя в виде:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \delta_i^{(1)} x_j \qquad (2.22)$$

2.3.4. Алгоритм наискорейшего спуска

Если при разложении целевой функции *E(w)* в ряд Тейлора ограничиться ее линейным приближением, то мы получим алгоритм наискорейшего спуска. Для выполнения соотношения $E(w_{t+1}) < E(w_t)$ достаточно подобрать $g(w_t)^T p < 0$. Условию уменьшения значения целевой функции отвечает выбор вектора направления

$$p_t = -g(w_t). \qquad (2.23)$$

Именно выражением (2.23) определяется вектор направления *p* в методе наискорейшего спуска. Ограничение слагаемым первого порядка при разложении функции в ряд Тейлора не позволяет использовать информацию о ее кривизне. Это обуславливает медленную сходимость метода (она остается линейной). Указанный недостаток, а также резкое замедление минимизации в

ближайшей окрестности точки оптимального решения, когда градиент принимает очень малые значения, делают алгоритм наискорейшего спуска низкоэффективным. Тем не менее простота, невысокие требования к объему памяти и относительно невысокая вычислительная сложность, обуславливают широкое использование алгоритма. Повысить эффективность удается путем эвристической модификации выражения, определяющего направление градиента.

Одна из модификаций получила название алгоритма обучения с моментом. При этом подходе уточнение весов сети производится по формулам:

$$w_{t+1} = w_t + \Delta w_t \qquad (2.24)$$

$$\Delta w_t = \eta_t p_t + \alpha(w_t - w_{t-1}), \qquad (2.25)$$

где α - это коэффициент момента, принимающий значения в интервале [0, 1].

Первое слагаемое в формуле (2.25) соответствует алгоритму наискорейшего спуска, а второе слагаемое учитывает последнее изменение весов и не зависит от фактического значения градиента. Чем больше значение коэффициента α, тем большее значение оказывает показатель момента на подбор весов. При постоянном значении коэффициента обучения $\eta_t = \eta$ приращение весов остается примерно одинаковым, то есть $\Delta w_t = \eta p_t + \alpha \Delta w_t$, поэтому эффективное приращение весов можно писать формулой:

$$\Delta w_t = \frac{\eta}{1-\alpha} p_t \qquad (2.26)$$

При значении α=*0,9* это соответствует десятикратному увеличению значения коэффициента обучения и, следовательно, десятикратному ускорению процесса обучения. При малых значениях градиента показатель момента начинает доминировать, что приводит к такому приращению весов, которое соответствует увеличению значения целевой функции, позволяющему выйти из зоны локального минимума. Однако показатель момента, не должен доминировать на протяжении всего процесса обучения, поскольку это приводит к нестабильности алгоритма. На практике, увеличение целевой функции не

допускается больше, чем на 4%. В противном случае, $\Delta w_t = 0$. При этом показатель градиента начинает доминировать над показателем момента и процесс развивается в направлении минимизации, заданном вектором градиента [4].

2.3.5. Алгоритм переменной метрики

В алгоритме переменной метрики используется квадратичное приближение целевой функции *E(w)*, представленной формулой (2.3) в окрестности полученного решения w_t.

Для достижения минимуму целевой функции требуется, чтобы $\frac{dE(w_t + p_t)}{dp_t} = 0$. При выполнении соответствующего дифференцирования можно получить условие оптимальности в виде:

$g(w_t) + H(w_t) p_t = 0$, откуда следует

$$p_t = -\left[H(w_t)\right]^{-1} g(w_t) \qquad (2.27)$$

Формула (2.27) однозначно указывает направление p_k, которое гарантирует достижение минимального для данного шага значения целевой функции. Из него следует, что для определения этого направления необходимо в каждом цикле вычислять значение градиента g и гессиана H в точке последнего решения w_t.

Формула (2.27), представляющая собой основу ньютоновского алгоритма оптимизации, является чисто теоретическим выражением, поскольку ее применение требует положительной определенности гессиана на каждом шаге, что практически не осуществимо. Поэтому в реальных алгоритмах вместо точно определенного гессиана $H(w_t)$ используется его приближение $G(w_t)$.

Основная идея метода переменной метрики заключается в том, что на каждом шаге гессиан или обратная ему величина, полученная на предыдущем шаге, модифицируются на величину некоторой поправки для обеспечения условия положительной определенности гессиана. Если прирост вектора w_t и градиента g на двух последовательных шагах итерации обозначить

соответственно s_t и r_t, то есть $s_t = w_t - w_{t-1}$ и $r_t = g(w_t) - g(w_{t-1})$, а матрицу, обратную приближению гессиана $V_t = [G(w_t)]^{-1}$, $V_{t-1} = [G(w_{t-1})]^{-1}$ обозначить *V*, то в соответствии с формулой Бройдена-Флетчера-Гольдфарба-Шенно процесс уточнения матрицы *V* можно описать рекуррентной зависимостью [4]:

$$V_t = V_{t-1} + \left[1 + \frac{r_t^T V_{t-1} r_t}{s_t^T r_t}\right] \frac{s_t s_t^T}{s_t^T r_t} - \frac{s_t r_t^T V_{t-1} r_t s_t^T}{s_t^T r_t} \qquad (2.28)$$

Метод переменной метрики характеризуется более быстрой сходимостью, чем метод наискорейшего спуска. Именно этот метод считается в настоящее время одним из наиболее эффективных способов оптимизации функции нескольких переменных. Применяется для не очень больших сетей, так как требует относительно большой вычислительной сложности, связанной с необходимостью расчета в каждом цикле n^2 элементов гессиана, и значительных объемов памяти.

2.3.6. Алгоритм потоковых графов.

Представляет собой альтернативный метод расчета компонентов градиента на основе анализа чувствительности сети методом сопряженных элементов. Метод потоковых графов предлагает более простые, по сравнению с классическими, правила формирования компонентов градиента, которые не зависят от сложности сети [4]. В теории систем под полной чувствительностью объекта понимается производная любого циркулирующего в нем сигнала относительно значений весов, которая может быть рассчитана на основании знаний о сигналах, распространяющихся по обычному графу (обозначим G) и по сопряженному с ним графу (обозначим $\hat{G}$). Граф $\hat{G}$ определяется как исходный граф G, в котором направленность всех дуг изменена на противоположную. Линейная дуга графа G и соответствующая ей дуга сопряженного графа $\hat{G}$ имеют идентичные описания. В случае нелинейной связи *f(x,K)*, где x - входной сигнал, а K - параметр, соответствующая ей дуга

графа $\hat{G}$ линеаризуется с коэффициентом $\beta = \frac{\partial f(x,K)}{\partial x}$, рассчитанным для фактического входного сигнала x графа G.

Метод расчета чувствительности нейронной сети с использованием потоковых графов основан на анализе исходного графа G и сопряженного с ним графа $\hat{G}$ при возбуждении последнего единичным сигналом, подаваемым на вход $\hat{G}$. Чувствительность графа G относительно параметров дуг этого графа к произвольному входному сигналу ν_0 можно выразить следующим образом:

для линейной дуги w_{ij} графа G:

$$\frac{d\nu_0}{dw_{ij}} = \nu_j \hat{\nu}_i, \qquad (2.29)$$

где w_{ij} - это коэффициент усиления линейной дуги, направленной от j-го узла к i-му, ν_j обозначает сигнал j-го узла графа G, а $\hat{\nu}_i$ - сигнал i-го узла сопряженного графа $\hat{G}$, для которого в качестве сигнала задается значение $\hat{\nu}_0 = 1$;

Для нелинейной дуги графа G, объединяющий l-й и r-й узлы и описываемой функцией $\nu_r = f_{rl}(\nu_l, K)$, чувствительность относительно параметра K определяется выражением

$$\frac{d\nu_0}{dK} = \hat{\nu}_k \frac{\partial f_{rl}(\nu_l, K)}{\partial K}, \qquad (2.30)$$

где $\frac{\partial f_{rl}(\nu_l, K)}{\partial K}$ рассчитывается для сигнала ν_l l-го узла графа G.

Обозначим w вектор оптимизированных параметров (весов w_i) системы, представленной графом G, $w = [w_1, w_2, ..., w_N]^T$, а $E(w)$ - целевую функцию. Тогда градиент $g(w) = \nabla E(w)$ можно определить в виде

$$g(w)=\left[\frac{\partial E(w)}{\partial w_1},\frac{\partial E(w)}{\partial w_2},\ldots,\frac{\partial E(w)}{\partial w_N}\right]^T. \qquad (2.31)$$

Если представить целевую функцию в форме, учитывающей только одну обучающую выборку

$$E(w)=\frac{1}{2}\sum_{s=1}^{M}(y_s-d_s)^2, \qquad (2.32)$$

где d_s обозначено ожидаемое значение s-го выходного нейрона, $s=1,2,\ldots,M$, то градиент целевой функции принимает вид:

$$g(w)=[g_1(w),g_2(w),\ldots,g_N(w)]^T, \qquad (2.33)$$

$$\text{где } g_r(w)=\sum_{s=1}^{M}(y_s-d_s)\frac{\partial y_s}{\partial w_r}. \qquad (2.34)$$

Для задания вектора градиента также необходимы производные выходных сигналов y_s графа относительно весов w_r, умноженные на величину погрешности (y_s-d_s). Способ формирования сопряженного графа $\hat{G}$ и методика его возбуждения для автоматического расчета вектора градиента на основе анализа только двух графов G и $\hat{G}$ представлена на рис. 2.3 и 2.4 соответственно. При замене всех единичных возбуждений в $\hat{G}$ на (y_s-d_s) любой компонент вектора градиента $g_r(w)$ может быть рассчитан по соответствующим сигналам исходного графа и сопряженного с ним точно так же, как и при определении обычной чувствительности.

Для линейной дуги графа G, описываемой весом w_{ij}, формула имеет вид:

$$\frac{\partial E(w)}{\partial w_{ij}}\nu_j\hat{\nu}_i. \qquad (2.35)$$

Для нелинейной дуги графа G, описываемой функцией $f_{rl}(\nu_l,K)$, получаем:

$$\frac{\partial E(w)}{\partial K}=\hat{\nu}_r\frac{\partial f_{rl}(\nu_l,K)}{\partial K}. \qquad (2.36)$$

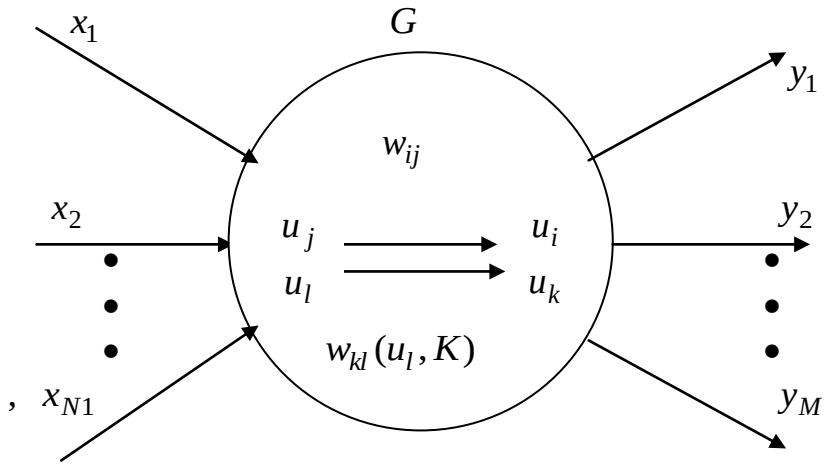


Рис. 2.3 – Исходный граф

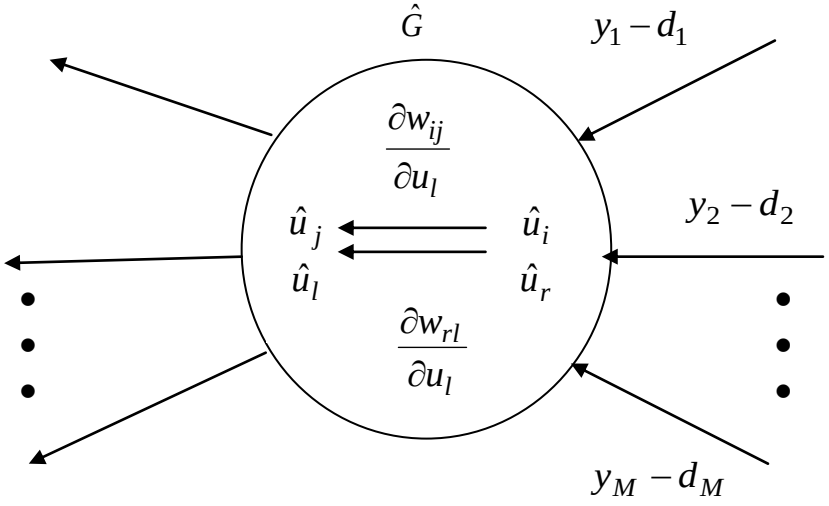


Рис. 2.4 – Сопряженный граф

2.3.7. Алгоритм RPROP.

Простой эвристический алгоритм, демонстрирующий высокую эффективность обучения, - это алгоритм М. Ридмиллера и Х. Брауна, называемый RPROP [9]. В этом алгоритме при уточнении весов учитывается только знак градиентной составляющей, а ее значение игнорируется:

$$\Delta w_{ij}(t) = -\eta_{ij}(t)\,\mathrm{sgn}\left(\frac{\partial E(w(t))}{\partial w_{ij}}\right). \qquad (2.37)$$

Коэффициент обучения подбирается индивидуально для каждого веса w_{ij} с учетом изменения значения градиента:

$$\eta_{ij}(t)=\begin{cases}\min\left(a\eta_{ij}(t-1),\eta_{\max}\right) & для \quad S_{ij}(t)S_{ij}(t-1)>0\\ \max\left(b\eta_{ij}(t-1),\eta_{\min}\right) & для \quad S_{ij}(t)S_{ij}(t-1)<0\\ \eta_{ij}(t-1) & \text{в остальных случаях}\end{cases}, \quad (2.38)$$

где $S_{ij}(t)=\dfrac{\partial E(w(t))}{\partial w_{ij}}$, a и b – константы: $a=1.2;\ b=0.5$. Минимальное и максимальное значения коэффициента обучения составляют $\eta_{\min}=10^{-6}$ и $\eta_{\max}=50$. Функция *sgn( )* принимает значение, равное знаку градиента.

2.3.8. Алгоритм имитации отжига

Все представленные ранее методы обучения нейронных сетей являются локальными. Они ведут к одному из локальных минимумов целевой функции, лежащему в окрестности точки начала обучения. Только в ситуации, когда значение глобального минимума известно, удается оценить, находится ли найденный локальный минимум в достаточной близости от искомого решения. Если локальное решение признается неудовлетворительным, следует повторить процесс обучения при других начальных значениях весов и с другими управляющими параметрами.

При решении реальных задач в общем случае даже приблизительная оценка глобального минимума оказывается неизвестной. По этой причине возникает необходимость применения методов глобальной оптимизации. Рассмотрим метод имитации отжига.

Название данного алгоритма связано с методами имитационного моделирования в статистической физике, основанными на методе Монте-Карло. Исследование кристаллической решетки и поведения атомов при медленном остывании тела привело к появлению на свет вероятностных алгоритмов, которые оказались чрезвычайно эффективными в комбинаторной оптимизации.

Классический алгоритм имитации отжига:

1. Запустить процесс из начальной точки *w* при заданной начальной температуре $T = T_{\max}$.

2. Пока *T>0* повторить *L* раз следующие действия:

2.1. Выбрать новое решение w' из окрестности *w*.

2.2. Рассчитать значение целевой функции $\Delta = E(w') - E(w)$.

2.3. Если $\Delta \le 0$, принять $w = w'$, в противном случае принять, что $w = w'$ с вероятностью $\exp(-\Delta / T)$ путем генерации случайного числа *R* из интервала (0,1) с последующим его сравнением со значением $\exp(-\Delta / T)$; если $\exp(-\Delta / T) > R$, принять новое решение $w = w'$; в противном случае проигнорировать новое решение.

3. Уменьшить температуру *(T← rT)* с использованием коэффициента уменьшения *r*, выбираемого из интервала (0,1), и вернуться к п.2.

4. После снижения температуры до нулевого значения провести обучение сети любым из представленных выше детерминированных методов, вплоть до достижения минимума целевой функции.

2.4. Проектирование архитектуры многослойного персептрона

Для решения какой-либо задачи с применением искусственной нейронной сети следует, прежде всего, спроектировать структуру сети, адекватную поставленной задаче. При проектировании нейронной сети необходимо решить вопрос о числе слоев и нейронов в каждом слое, а также определить необходимые связи между слоями.

Подбор числа нейронов во входном слое обусловлен размерностью входного вектора *x*. Число нейронов выходного слоя принимается равным размерности эталонного вектора *d*. Серьезной проблемой остается подбор числа скрытых слоев и числа нейронов в каждом из них.

Теоретическое решение этой задачи в смысле условия достаточности было предложено математиками, занимающимися аппроксимацией функций нескольких переменных. Следует отметить, что нейронная сеть выступает в роли универсального аппроксиматора обучающих данных *(x,d)* [1, 2, 3].

Определение минимального числа скрытых слоев сети основано на использовании свойств аппроксимирующих функций. Возможность такого обобщения следует из теорем А.Н. Колмогорова [2,3] и из теоремы об универсальной аппроксимации, которую можно рассматривать как естественное расширение теоремы Вейерштрасса [1]. Пусть *N*- число входных узлов многослойного персептрона, а *M*- число выходных нейронов сети. Тогда теорема об универсальной аппроксимации формулируется следующим образом [5]:

Пусть $\varphi(.)$– ограниченная, не постоянная монотонно возрастающая непрерывная функция. Пусть I_N - N- мерный единичный гиперкуб $[0,1]^N$. Пусть пространство непрерывных на I_N функций обозначается символом $C(I_N)$. Тогда для любой функции $f \in C(I_N)$ и $\varepsilon>0$ существует такое целое число K и множество действительных констант w_i, w_0 и w_{ij}, где i=1,…,K, j=1,..,N, что

$$F(x_1, x_2,...,x_N) = \sum_{i=1}^{K} w_i \varphi\left(\sum_{j=1}^{N} w_{ij} x_j + w_0 \right) \qquad (2.39)$$

является реализацией аппроксимации функции f(.), то есть

$$\left|F(x_1, x_2,...,x_N) - f(x_1, x_2,...,x_N)\right| < \varepsilon \qquad (2.40)$$

для всех $x_1, x_2,...,x_N$, принадлежащих входному пространству.

Теорема об универсальной аппроксимации непосредственно применима к многослойному персептрону. Во-первых, в модели многослойного персептрона в качестве функций активации используются ограниченные, монотонно возрастающие сигмоидальные функции, удовлетворяющие условиям, накладываемым теоремой на функцию *$\varphi(.)$*. Во-вторых, выражение (2.39) описывает выходной сигнал сети следующего вида:

1. Сеть содержит *N* входных узлов и один скрытый слой, состоящий из *M* нейронов. Входы обозначены $x_1, x_2,...,x_N$.
2. Скрытый нейрон *i* имеет синаптические веса $w_{i1}, w_{i2},...,w_{iN}$ и w_{i0} – вес порогового элемента.

3. Выход сети представляет собой линейную комбинацию выходных сигналов скрытых нейронов, взвешенных синаптическими весами выходных нейронов - $w_1, w_2,...,w_K$.

Таким образом, теорема утверждает, что многослойного персептрона с одним скрытым слоем достаточно для построения равномерной аппроксимации с точностью ε для любого обучающего множества, представленного набором входов $x_1, x_2,...,x_N$ и ожидаемых выходов $d = f(x_1, x_2,...,x_N)$. Тем не менее из теоремы не следует, что один скрытый слой является оптимальным в смысле времени обучения, простоты реализации и, что более важно, качества обобщения.

Теорема об универсальной аппроксимации обеспечивает необходимый математический базис для доказательства применимости сетей прямого распространения с одним скрытым слоем для решения задач аппроксимации. Однако она не обеспечивает способ конструирования многослойного персептрона, обладающего такими свойствами.

Кроме того, теорема об универсальной аппроксимации предполагает, что аппроксимируемая непрерывная функция известна, и для ее приближения можно использовать скрытый слой неограниченного размера. На практике эти предположения обычно не верны.

Проблема многослойного персептрона с одним скрытым слоем состоит в том, что нейроны могут взаимодействовать друг с другом на глобальном уровне. При наличии двух скрытых слоев, процесс аппроксимации становится более управляемым. В частности, можно утверждать следующее [5]:

1. *Локальные признаки* извлекаются в первом скрытом слое, то есть некоторые нейроны первого скрытого слоя можно использовать для разделения входного пространства на отдельные области, а остальные нейроны слоя обучать локальным признакам, характеризующим эти области.
2. *Глобальные признаки* извлекаются во втором скрытом слое. В частности, нейрон второго скрытого слоя «обобщает» выходные

сигналы нейронов первого скрытого слоя, относящихся к конкретной области входного пространства. Таким образом, он обучается глобальным признакам этой области, а в остальных областях его выходной сигнал равен нулю.

В практических реализациях сетей чаще всего используются сети с одним скрытым слоем, реже – с двумя, причем число нейронов в слое может изменяться (как правило, от N до $3N$) [4].

2.5. Подбор оптимальной архитектуры

Одно из важнейших свойств нейронной сети – это способность к обобщению полученных знаний. Сеть, обученная на некотором множестве входных векторов, генерирует ожидаемые результаты при подаче на ее вход тестовых данных, относящихся к тому же множеству, но не участвовавших непосредственно в процессе обучения. В составе обучающих данных можно выделить определенное подмножество контрольных данных, используемых для определения точности обучения сети. В то же время, в составе обучающих данных не должно быть уникальных данных, свойства которых отличаются от ожидаемых значений. Способность сети распознавать данные из тестового подмножества характеризует ее возможности обобщения знаний.

Имеет место компромисс между точностью и обобщающей способностью сети, который можно оптимизировать посредством выбора количества скрытых нейронов для данной сети. Количество скрытых нейронов, с одной стороны, должно быть достаточным. для того чтобы решить поставленную задачу, а с другой - не должно быть слишком большим, чтобы обеспечить необходимую обобщающую способность.

Не существует простого способа для определения необходимого числа скрытых элементов сети. Ряд противоречивых требований накладывается на количество весовых коэффициентов сети.

Во-первых, необходимо иметь достаточное число данных для обучения сети с выбранным числом весовых коэффициентов. Если бы целью обучения было только запоминание обучающих выборок, то их число могло быть равным

числу весов. Однако такая сеть не будет обладать свойством обобщения, и сможет только восстанавливать данные. Число весов частично обусловлено числом входных и выходных элементов и, следовательно, методом кодировки входных и выходных данных.

Во-вторых, сеть должна обучаться на избыточном множестве данных, чтобы обладать свойством обобщения. При этом веса будут адаптироваться не к отдельным выборкам, а к их статистически усредненным совокупностям.

Обучение сети ведется путем минимизации целевой функции *E(w),* определяемой только на подмножестве обучающих данных, что обеспечивает достаточное соответствие выходных сигналов сети ожидаемым значениям из обучающей выборки.

Истинная цель обучения состоит в таком подборе архитектуры и параметров сети, которые обеспечат минимальную погрешность распознавания тестового подмножества данных, не участвовавших в обучении. Критерием правильности окончательных результатов является погрешность обобщения, вычисленная по тестовой выборке.

Со статистической точки зрения погрешность обобщения E зависит от уровня погрешности обучения H и от доверительного интервала и характеризуется отношением (2.41)[5].

$$E \le H + \varepsilon \qquad (2.41)$$

где $\varepsilon = (\frac{P}{h}, H)$ – доверительный интервал, P – объем обучающей выборки, h – мера Вапника-Червоненкиса .

Мера Вапника-Червоненкиса (*VC*-измерение) отражает уровень сложности сети и тесно связана с количеством содержащихся в ней весов. Чем больше число различных весов, тем больше сложность сети и соответственно значение *VC*-измерения. Метод точного определения этой меры неизвестен, но можно определить верхнюю и нижнюю границу этой меры в виде формулы (2.42).

$$2\left[\frac{K}{2}\right]N \le h \le 2M(1+\lg L) \qquad (2.42)$$

где K – количество нейронов в скрытом слое, N – размерность входного вектора, M – общее количество весов сети, L – общее количество нейронов в сети.

Из формулы (2.42) следует, что нижняя граница диапазона приблизительно равна числу весов, связывающих входной и выходной слои, тогда как верхняя граница превышает двукратное суммарное число всех весов сети. В качестве приближенного значения *VC*–измерения может быть использовано общее число весов нейронной сети.

Таким образом, на погрешность обобщения оказывает влияние отношение количества обучающих выборок к количеству весов сети. Небольшой объем обучающего подмножества при фиксированном количестве весов вызывает хорошую адаптацию сети к его элементам, однако не усиливает способности к обобщению, так как в процессе обучения наблюдается относительное превышение числа подбираемых параметров над количеством пар фактических и ожидаемых выходных сигналов сети. Эти параметры адаптируются с чрезмерной и неконтролируемой точностью к значениям конкретных выборок, а не к диапазонам, которые эти выборки должны представлять. Фактически задача аппроксимации подменяется в этом случае задачей приближенной интерполяции. В результате всякого рода нерегулярности обучающих данных и шумы могут восприниматься как существенные свойства процесса.

На основе опытных данных существуют следующие рекомендации по выбору числа скрытых нейронов:

- Не следует выбирать число скрытых нейронов больше, чем удвоенное число входных элементов.
- Число обучающих данных должно быть по крайней мере в $1/\varepsilon$ раз больше количества весов в сети, где ε - граница ошибки обучения.
- Следует выявить особенности нейросети, так как в этом случае требуется меньшее количество скрытых нейронов, чем входов. Если есть сведения, что размерность данных может быть уменьшена, то следует использовать меньшее количество скрытых нейронов.

- При обучении на бесструктурных входах необходимо, чтобы количество скрытых нейронов было больше, чем количество входов. Если набор данных не имеет общих свойств, необходимо использовать больше скрытых нейронов.
- Имеет место взаимоисключающая связь между обобщающими свойствами (меньше нейронов) и точностью (больше нейронов), которая специфична для каждого приложения.
- Большая сеть требует большего времени для обучения.

Существуют также практические рекомендации по модификации алгоритмов конструирования сети:

- Если ошибка обучения мала, а ошибка тестирования велика, следовательно, сеть содержит слишком много весовых коэффициентов.
- Если и ошибка обучения, и ошибка тестирования велики, следовательно, весовых коэффициентов слишком мало.
- Если все весовые коэффициенты очень большие, следовательно, весовых коэффициентов слишком мало.
- Добавление весов не панацея; если известно, что весовых коэффициентов достаточно, следует подумать о других причинах ошибок, например о недостаточном количестве обучающих данных.
- Не следует добавлять слишком много весовых коэффициентов, чтобы не переступить пределов, установленных ранее.
- И, наконец, что очень важно, начальные весовые коэффициенты должны быть случайными и небольшими по величине (например, между +1 и –1).

2.6. Контрольные вопросы и упражнения

1. Многослойный персептрон относится к типу полносвязных сетей прямого распространения сигналов. Обоснуйте данное утверждение.

2. Какие типы нейронов обычно используют в многослойном персептроне?

3. К какому типу относятся алгоритмы обучения многослойного персептрона: обучение «с учителем» или обучение «без учителя»?

4. Алгоритм обратного распространения ошибки применим к нейронным сетям, у которых функции активации нейронов должны быть непрерывно дифференцируемыми. Объясните причину этого.

5. Сравните скорости сходимости алгоритма наискорейшего спуска и алгоритма переменной метрики. Обоснуйте свой ответ.

6. Алгоритм имитации отжига относится к алгоритмам глобальной оптимизации. Обоснуйте свой ответ.

7. В соответствии с теоремой об универсальной аппроксимации, какое число скрытых слоев в многослойном персептроне достаточно для построения равномерной аппроксимации с точностью ε для любого обучающего множества, представленного набором входов $x_1, x_2, \ldots, x_N$ и ожидаемых выходов $d = f(x_1, x_2, \ldots, x_N)$?

8. Исследуйте аппроксимирующие свойства многослойного персептрона с одним скрытым слоем, обученного методом обратного распространения ошибки, методом компьютерного моделирования следующих отображений:

а) $f(x) = \frac{1}{x}, 1 <= x <= 100$;

б) $f(x) = \lg_{10}(x), 1 <= x <= 10$;

в) $f(x) = \exp(-x), 1 <= x <= 10$;

г) $f(x) = \sin(x), 1 <= x <= \pi / 2$.

Для проведения исследований выполните следующие действия:

а) Создайте два набора данных: один – для обучения, второй – для тестирования;

б) Оцените сложность сети при помощи *VC*-измерения;

в) Исследуйте, как изменение числа нейронов в скрытом слое влияет на ошибку аппроксимации;

г) Исследуйте, как изменение функции активации влияет на ошибку аппроксимации (используйте логистическую функцию и гиперболический тангенс).

д) Исследуйте, как объем обучающей выборки влияет на ошибку аппроксимации.

3. Радиальные сети

3.1. Математическое обоснование радиально-базисных сетей

Многослойные нейронные сети, с точки зрения математики, выполняют аппроксимацию стохастической функции нескольких переменных путем преобразования множества входных переменных $x \in R^N$ во множество выходных переменных $y \in R^M$. Вследствие характера сигмоидальной функции активации осуществляется аппроксимация глобального типа, так как преобразование значения функции в произвольной точке пространства выполняется объединенными усилиями многих нейронов.

Другой способ отображения входного множества в выходное множество заключается в преобразовании путем адаптации нескольких одиночных аппроксимирующих функций к ожидаемым значениям, причем эта адаптация проводится только в локальной области многомерного пространства. При таком подходе отображение всего множества данных представляет собой сумму локальных преобразований, а скрытые нейроны составляют множество базисных функций локального типа.

Особое семейство образуют радиальные сети, в которых скрытые нейроны реализуют функции, радиально изменяющиеся вокруг выбранного центра и принимающие ненулевые значения только в окрестности этого центра. Подобные функции, определяемые в виде $\varphi(x) = \varphi(\|x - c\|)$, называются *радиальными базисными функциями*. В таких сетях роль скрытого нейрона заключается в отображении радиального пространства вокруг одиночной заданной точки либо вокруг группы таких точек, образующих кластер. Суперпозиция сигналов, поступающих от всех скрытых нейронов, которая выполняется выходным нейроном, позволяет получить отображение всего многомерного пространства.

Сети радиального типа представляют собой естественное дополнение сигмоидальных сетей. Сигмоидальный нейрон представляется в многомерном пространстве гиперплоскостью, которая разделяет это пространство на два класса, в которых выполняется одно из двух условий: либо $\sum_j w_{ij} x_j > 0$, либо $\sum_j w_{ij} x_j < 0$. Такой подход продемонстрирован на рис. 3.1а. В свою очередь радиальный нейрон представляет собой гиперсферу, которая осуществляет шаровое разделение пространства вокруг центральной точки (рис. 3.1б).

Именно с этой точки зрения радиальный нейрон является естественным дополнением сигмоидального нейрона, поскольку в случае круговой симметрии данных позволяет заметно уменьшить количество нейронов, необходимых для разделения различных классов.

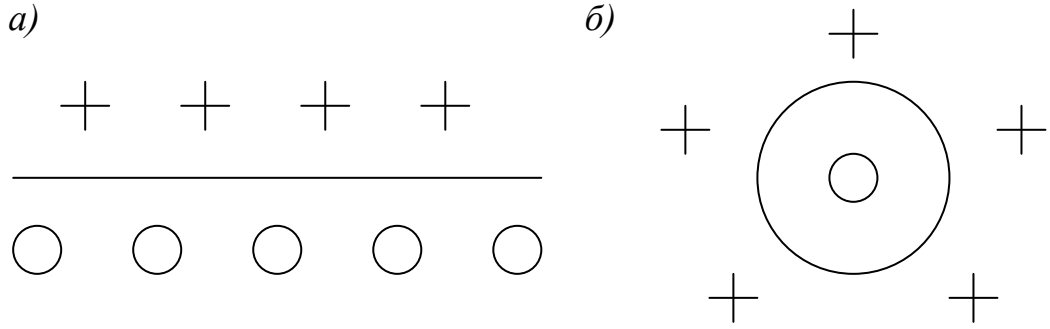


Рис. 3.1 Иллюстрация способов разделения пространства данных:
а) сигмоидальным нейроном; б) радиальным нейроном

Так как нейроны могут выполнять различные базисные функции, в радиальных сетях отсутствует необходимость использования большого количества скрытых слоев. Структура типичной радиальной сети включает входной слой, на который подаются сигналы, описываемые входным вектором *x*, скрытый слой с нейронами радиального типа и выходной слой, состоящий, как правило, из одного или нескольких линейных нейронов. Функция

выходного нейрона сводится исключительно к взвешенному суммированию сигналов, генерируемых скрытыми нейронами.

Математическую основу функционирования радиальных сетей составляет теорема Т. Ковера о разделимости образов, которая утверждает следующее [5]:

Нелинейное преобразование сложной задачи классификации образов в пространство более высокой размерности повышает вероятность линейной разделимости образов.

Теорема Ковера о разделимости образов базируется на двух моментах [4]:

1. Определение нелинейной скрытой функции $\varphi_i(x)$, где x – входной вектор, а $i=1,2,\ldots,K$, K – размерность скрытого пространства.
2. Высокая размерность скрытого пространства по сравнению с размерностью входного. Эта размерность определяется значением, присваиваемым K (то есть количеством скрытых нейронов).

Если вектор радиальных функций $\varphi(x) = [\varphi_1(x), \varphi_2(x), \ldots, \varphi_K(x)]^T$ в N-мерном входном пространстве обозначить $\varphi(x)$, то это пространство является нелинейно φ-разделяемым на два пространственных класса X^+ и X^- тогда, когда существует такой вектор весов w, что

$$\begin{aligned} w^T\varphi(x) > 0 \quad x \in X^+, \\ w^T\varphi(x) < 0 \quad x \in X^-. \end{aligned} \qquad (3.1)$$

Граница между этими классами определяется уравнением $w^T\varphi(x) = 0$.

Ковер доказал, что каждое множество образов, случайным образом размещенных в многомерном пространстве, является φ-разделяемым с вероятностью 1 при условии большой размерности K этого пространства. На практике это означает, что применение достаточно большого количества скрытых нейронов, реализующих радиальные функции $\varphi_i(x)$, гарантирует решение задачи классификации при построении всего лишь двухслойной сети. При этом скрытый слой должен реализовать вектор $\varphi(x)$, а выходной слой может состоять из единственного линейного нейрона, выполняющего

суммирование выходных сигналов от скрытых нейронов с весовыми коэффициентами, заданными вектором *w*.

Простейшая нейронная сеть радиального типа функционирует по принципу многомерной интерполяции, состоящей в отображении *p* различных входных векторов x_t *(t=1,2,…,p)* из входного *N*-мерного пространства во множество из *p* рациональных чисел d_t *(t=1,2,…,p).* Для реализации этого процесса необходимо использовать *p* скрытых нейронов радиального типа и задать такую функцию отображения *F(x)*, для которой выполняется условие интерполяции:

$$F(x_t) = d_t. \qquad (3.2)$$

С практической же точки зрения использование в разложении большого числа *p* базисных функций недопустимо, поскольку число обучающих выборок велико и равно числу радиальных функций, и в результате вычислительная сложность обучающего алгоритма становится чрезмерной, а сама сеть адаптируется к разного рода шумам и нерегулярностям, сопровождающим обучающие выборки. Поэтому необходимо редуцировать количество весов, что приводит к уменьшению количества базисных функций. В этом случае ищется субоптимальное решение в пространстве меньшей размерности, которое с достаточной точностью аппроксимирует точное решение. Если ограничиться *K* базисными функциями, то аппроксимирующее решение можно представить в виде

$$F(x) = \sum_{i=1}^{K} w_i \phi\left(\left\| x - c_i \right\|\right), \qquad (3.3)$$

где $K < p$, а c_i *(i=1,2,…,K)* – множество центров, которые необходимо определить. В особом случае, если принять *K=p*, то можно получить точное решение $c_i = x_t$.

Задача аппроксимации состоит в подборе соответствующего количества радиальных функций $\varphi(\| x - c_i \|)$ и их параметров, а также в таком подборе весов w_i *(i=1,2,…,K)*, чтобы решение уравнения (3.3) было наиболее близким к точному. Поэтому проблему подбора параметров радиальных функций и

значений весов w_i сети можно свести к минимизации целевой функции, которая при использовании метрики Эвклида записывается в форме

$$E = \sum_{t=1}^{p} \left[\sum_{i=1}^{K} w_i \varphi(\| x_t - c_i \|) - d_t \right]^2 . \qquad (3.4)$$

В этом уравнении K представляет количество радиальных нейронов, а p – количество обучающих пар (x_t, d_t), где x_t – это входной вектор, а d_t – соответствующий ему ожидаемый выходной вектор.

3.2. Структура радиально-базисной сети

На рис. 3.2 представлена обобщенная структура радиально-базисной сети. В качестве радиальной функции чаще всего применяется функция Гаусса. При размещении ее центра в точке c_i она может быть определена в сокращенной форме как:

$$\varphi(x) = \varphi(\| x - c_i \|) = \exp\left(- \frac{\| x - c_i \|^2}{2\sigma_i^2} \right). \qquad (3.5)$$

В этом выражении σ_i – параметр, от значения которого зависит ширина функции.

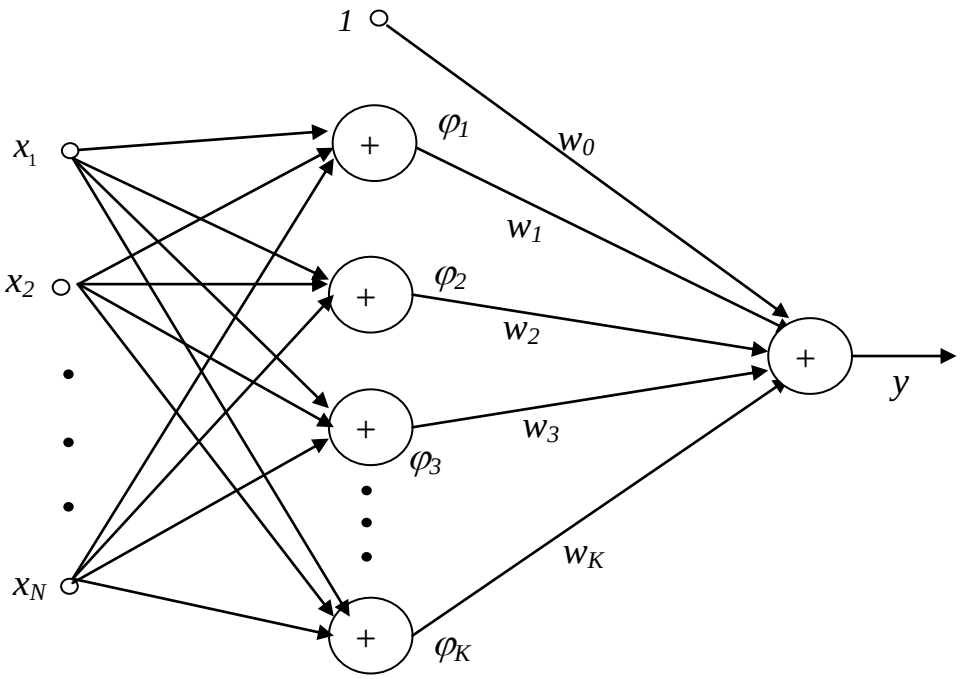


Рис. 3.2 Обобщенная структура радиальной сети RBF

Полученное решение, представляющее аппроксимирующую функцию в многомерном пространстве в виде взвешенной суммы локальных базисных радиальных функций (выражение (3.3)), может быть интерпретировано радиальной нейронной сетью, представленной на рис. 3.2, в которой φ_i определяется зависимостью (3.5).

Это сеть с двухслойной структурой, в которой только скрытый слой выполняет нелинейное отображение, реализуемое нейронами с базисными радиальными функциями, параметры которых (центры c_i и коэффициенты σ_i) уточняются в процессе обучения. Скрытый слой не содержит линейных весов, аналогичных весам сигмоидальной сети. Выходной нейрон, как правило, линеен, а его роль сводится к взвешенному суммированию сигналов, поступающих от нейронов скрытого слоя. Вес w_0, как и при использовании сигмоидальных функций, представляет пороговый элемент, определяющий показатель постоянного смещения функции.

Полученная архитектура радиальных сетей аналогична структуре многослойной сети с одним скрытым слоем. Роль скрытых нейронов в ней играют базисные радиальные функции. Однако, в отличие от сигмоидальной сети, радиальная сеть имеет фиксированную структуру с одним скрытым слоем и линейными выходными нейронами. Используемые радиальные функции скрытых нейронов могут иметь разнообразную структуру. Нелинейная радиальная функция каждого скрытого нейрона имеет свои значения параметров c_i, тогда как в сигмоидальной сети применяются, как правило, стандартные функции активации с одним и тем же параметром. Аргументом радиальной функции является эвклидово расстояние вектора x от центра c_i и σ_i, а в сигмоидальной сети это скалярное произведение векторов $w^T x$.

Структуру *RBF*-сети можно усилить путем применения масштабирования входных сигналов. Если принять во внимание, что многомерная функция может иметь различный масштаб по каждой оси, с практической точки зрения

оказывается полезным уточнить норму масштабирования путем ввода в определение эвклидовой метрики весовых коэффициентов в виде матрицы *Q*.

$$\|x\|_Q^2 = (Qx)^T (Qx) = x^T Q^T Qx \qquad (3.6)$$

Масштабирующая матрица при *N*-мерном векторе *x* имеет вид:

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \dots & Q_{1N} \\ Q_{21} & Q_{22} \dots & Q_{2N} \\ \dots & \dots & \dots & \dots \\ Q_{N1} & Q_{N2} \dots & Q_{NN} \end{bmatrix} \qquad (3.7)$$

При обозначении произведения матриц $Q^T Q$ матрицей корреляции C в общем случае получим:

$$\|x\|_Q^2 = \sum_{j=1}^{N} \sum_{r=1}^{N} C_{jr} x_j x_r \qquad (3.8)$$

Если масштабирующая матрица Q имеет диагональный вид, то получаем $\|x\|_Q^2 = \sum_{j=1}^{N} C_{jj} x_j^2$. Это означает, что норма масштабирования вектора рассчитывается согласно стандартной формуле Эвклида, с использованием индивидуальной шкалы для каждой переменной x_j. При *Q=1* взвешенная метрика Эвклида сводится к классической метрике $\|x\|_Q^2 = \|x\|^2$.

В случае использования функции Гаусса с центром в точке c_i и масштабирующей взвешенной матрицы Q_i, связанной с *i*-й базисной функцией, получим обобщенную форму функции Гаусса:

$$\varphi(x) = \varphi(\|x - c_i\|_{Q_i}) = \exp\left[-(x - c_i)^T Q_i^T Q_i (x - c_i)\right] = \exp\left[\frac{1}{2}(x - c_i)^T C_i (x - c_i)\right] \quad (3.9),$$

где матрица $\frac{1}{2} C_i = Q_i^T Q_i$ играет роль скалярного коэффициента $\frac{1}{2\sigma_i^2}$ стандартной многомерной функции Гаусса, заданной выражением (3.5).

Во многих практических приложениях масштабирующая матрица $Q(i)$ для *i*-го радиального нейрона имеет диагональную форму, в которой только элементы $Q_{jj}^{(i)}$ принимают ненулевые значения. В такой системе отсутствует

круговое перемешивание сигналов, соответствующих различным компонентам вектора *x*, а элемент $Q_{jj}^{(i)}$ играет роль индивидуального масштабирующего коэффициента для *j*-го компонента вектора *x* *i*-го нейрона. На рис. 3.3 представлена структура упрощенной сети *HRBF* с диагональными матрицами $Q(i)$. В сетях *HRBF* роль коэффициентов σ_i^2 выполняют элементы матрицы *Q*, которые уточняются в процессе обучения.

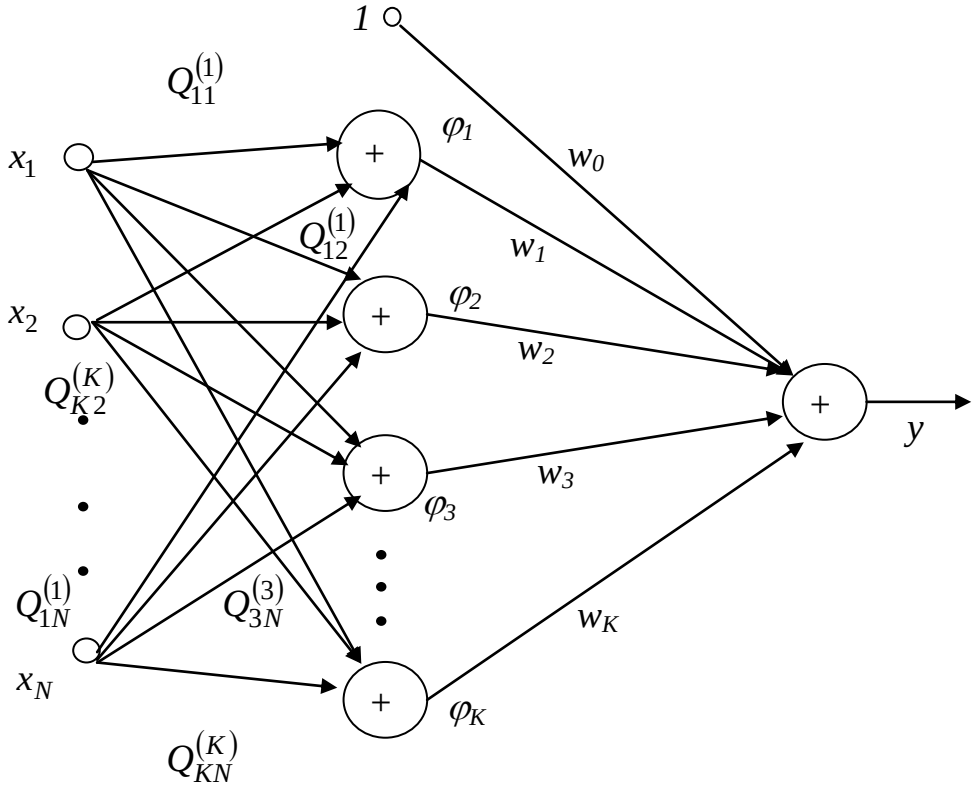


Рис. 3.3 Обобщенная структура радиальной сети HRBF

3.3. Основные алгоритмы обучения радиальных сетей

3.3.1. Алгоритм самоорганизации для уточнения параметров радиальных функций

Процесс обучения сети *RBF* с учетом выбранного типа радиальной базисной функции сводится:

- к подбору центров c_i и параметров σ_i формы базисных функций (часто используются алгоритмы обучения без учителя);

- к подбору весов нейронов выходного слоя (часто используются алгоритмы обучения с учителем).

Подбор количества базисных функций, каждой из которых соответствует один скрытый нейрон, считается основной проблемой, возникающей при корректном решении задачи аппроксимации. Как и при использовании сигмоидальных сетей, слишком малое количество нейронов не позволяет уменьшить в достаточной степени погрешность обобщения множества обучающих данных, тогда как слишком большое их число увеличивает погрешность выводимого решения на множестве тестирующих данных. Подбор необходимого и достаточного количества нейронов зависит от многих факторов. Как правило, количество базисных функций K составляет определенную долю от объема обучающих данных p, причем фактическая величина этой доли зависит от размерности вектора $\boldsymbol{x}$ и от разброса ожидаемых значений d_t, соответствующих входным векторам x_t, для $t=1,2,...,p$.

Процесс самоорганизации обучающих данных автоматически разделяет пространство на так называемые области Вороного, определяющие различающиеся группы данных. Данные, сгруппированные внутри кластера, представляются центральной точкой, определяющей среднее значение всех его элементов. Центр кластера отождествляется с центром соответствующей радиальной функции.

Разделение данных на кластеры можно выполнить с использованием алгоритма *К-усреднений*.

Согласно этому алгоритму центры радиальных базисных функций размещаются только в тех областях входного пространства, в которых имеются информативные данные. Если обучающие данные представляют непрерывную функцию, начальные значения центров в первую очередь размещают в точках, соответствующих всем максимальным и минимальным значениям функции.

Пусть N - число нейронов скрытого слоя, t – номер итерации алгоритма. Тогда алгоритм *К-усреднений* можно описать следующим образом [5]:

1. *Инициализация*. Случайным образом выбираем начальные значения центров $c_i(0)$, которые должны быть различны. При этом значения эвклидовой нормы по возможности должны быть небольшими.
2. *Выборка*. Выбираем вектор x_t из входного пространства.
3. *Определение центра-победителя.* Выбираем центр c_w, ближайший к x_t, для которого выполняется соотношение: $w = \arg\min_i \| x_t - c_i(t) \|, i = 1,2,...,N.$
4. *Уточнение.* Центр-победитель подвергается уточнению в соответствии с формулой (3.6):

$$c_i(t+1) = c_i(t) + \eta(x_t - c_i(t)), \qquad (3.6)$$

где η - коэффициент обучения, имеющий малое значение (обычно η<<1), причем уменьшающееся во времени. Остальные центры не изменяются.

5. *Продолжение.* Увеличиваем на единицу значение *t* и возвращаемся к шагу 2, пока положение центров не стабилизируется.

Также применяется разновидность алгоритма, в соответствии с которой значение центра-победителя уточняется в соответствии с формулой (3.6), а один или несколько ближайших к нему центров отодвигаются в противоположном направлении, и этот процесс реализуется согласно выражению

$$c_i(t+1) = c_i(t) - \eta_1(x_t - c_i(t)). \qquad (3.7)$$

Такая модификация алгоритма позволяет отдалить центры, расположенные близко друг к другу, что обеспечивает лучшее обследование всего пространства данных *($\eta_1 < \eta$)*.

После фиксации местоположения центров проводится подбор значений параметров σ_i, соответствующих конкретным базисным функциям. Параметр σ_i радиальной функции влияет на форму функции и величину области ее охвата, в которой значение этой функции не равно нулю. Подбор σ_i должен проводится таким образом, чтобы области охвата всех радиальных функций

накрывали все пространство входных данных, причем любые две зоны могут перекрываться только в незначительной степени. При такой организации подбора значения σ_i, реализуемое радиальной сетью отображение функции будет относительно монотонным.

Для расчета σ_i может быть применен алгоритм, при котором на значение σ_i влияет на расстояние между *i*-м центром c_i и его *P* ближайшими соседями. В этом случае значение σ_i определяется по формуле (3.8):

$$\sigma_i = \sqrt{\frac{1}{R}\sum_{t=1}^{P} \| c_i - c_t \|^2} \,. \qquad (3.8)$$

На практике значение *R* обычно лежит в интервале [3; 5].

Данный алгоритм обеспечивает только локальную оптимизацию, зависящую от начальных условий и параметров процесса обучения.

При неудачно выбранных начальных условиях, некоторые центры могут застрять в области, где количество обучающих данных ничтожно мало, либо они вообще отсутствуют. Следовательно, процесс модификации центров затормозится или остановится.

Для решения данной проблемы могут быть применены два различных подхода:

1. Задать фиксированные значения η для каждого центра. Центр, наиболее близкий к текущему вектору *x*, модифицируется сильнее, остальные - обратно пропорционально их расстоянию до этого текущего вектора *x*.
2. Использовать взвешенную меру расстояния от каждого центра до вектора *x*. Весовая норма делает «фаворитами» те центры, которые реже всего побеждают.

Оба подхода не гарантируют 100% оптимальность решения.

Подбор коэффициента η тоже является проблемой. Если η имеет постоянное значение, то оно должно быть мало, чтобы обеспечить сходимость алгоритма, следовательно, увеличивается время обучения.

Адаптивные методы позволяют уменьшать значение η по мере роста времени *t*. Наиболее известным адаптивным методом является алгоритм Даркена-Муди:

$$\eta(t) = \frac{\eta_0}{1 + \frac{k}{T}}, \qquad (3.9)$$

где T – постоянная времени, подбираемая для каждой задачи. При $k<T$ η не изменяется, при $k>T$ – уменьшается до нуля.

3.3.2. Гибридный алгоритм обучения радиальных сетей

В гибридном алгоритме процесс обучения разделяется на два этапа [4]:

1) Подбор линейных параметров сети (веса выходного слоя) при использовании метода псевдоинверсии;
2) Адаптация нелинейных параметров радиальных функций (центра c_i и ширины σ_i этих функций).

Оба этапа тесно переплетаются. При фиксации конкретных значений центров и ширины радиальных функций за один шаг, с помощью метода псевдоинверсии подбираются веса выходного слоя. Если обозначить $d = [d_1, d_2, ..., d_t]^T$ вектор ожидаемых значений, $w = [w_1, w_2, ..., w_K]^T$ -вектор весов сети, а G – радиальную матрицу Грина:

$$G = \begin{bmatrix} \varphi(\|x_1 - c_1\|) & \varphi(\|x_1 - c_2\|) & \dots & \varphi(\|x_1 - c_K\|) \\ \varphi(\|x_2 - c_1\|) & \varphi(\|x_2 - c_2\|) & \dots & \varphi(\|x_2 - c_K\|) \\ \dots & \dots & \dots & \dots \\ \varphi(\|x_p - c_1\|) & \varphi(\|x_p - c_2\|) & \dots & \varphi(\|x_p - c_K\|) \end{bmatrix},$$ то задача нахождения вектора весов сводится к решению системы уравнений, линейных относительной весов:

$$G(w)=d \qquad (3.14)$$

Вследствие прямоугольности матрицы G можно определить вектор весов w с использованием операции псевдоинверсии матрицы G, то есть

$$w=G+d \qquad (3.15),$$

где $G^+ = (G^T G)^{-1} G^T$ обозначает псевдоинверсию прямоугольной матрицы G.

На практике псевдоинверсия рассчитывается с применением декомпозиции SVD. Если G – действительная матрица размера pxK, то существуют ортогональные матрицы $U = \lfloor u_1, u_2, ..., u_p \rfloor$ и $V = [v_1, v_2, ..., v_K]$ такие, что $G = USV^T$, где S – псевдодиагональная матрица размера pxK, $K<p$, $s_1 > s_2 > ... > s_K >= 0$.

Пусть только первые r столбцов матрицы S имеют значимые величины, тогда остальными столбцами можно пренебречь. Тогда матрицы U и V будут иметь следующий вид $U = [u_1, u_2, ..., u_r]$ и $V = [v_1, v_2, ..., v_r]$, а матрица S становится полностью диагональной $S = diag[s_1, s_2, ..., s_r]$.

В этом случае матрица G может быть приближенно представлена в виде:

$$G \cong U_r S_r V_r^T . \qquad (3.16)$$

Псевдообратная матрица для матрицы G находится по формуле:

$$G^+ = U_r S_r^{-1} V_r^T \quad , \qquad (3.17)$$

где $S_r^{-1} = [1/s_1, 1/s_2, ..., 1/s_r]$. Тогда вектор весов находится из соотношения:

$$w = U_r S_r^{-1} V_r^T d . \qquad (3.18)$$

На втором этапе при зафиксированных значениях выходных весов возбуждающие сигналы пропускаются по сети до выходного слоя, что позволяет рассчитать величину погрешности для последовательности векторов x_t. Далее происходит возврат к скрытому слою (обратное распространение). По величине погрешности определяется вектор градиента целевой функции относительно конкретных центров c_{ij} и ширины σ_{ij}.

Каждая радиальная функция определяется в общем виде как

$$\varphi_i(x_t) = \exp\left(-\frac{1}{2} u_{it} \right), \qquad (3.19)$$

где суммарный сигнал нейрона u_{it} описывается выражением

$$u_{it} = \sum_{j=1}^{N} \frac{(x_{jt} - c_{ij})^2}{\sigma_{ij}^2} . \qquad (3.20)$$

При существовании p обучающих пар целевую функцию можно задать в виде

$$\frac{\partial E}{\partial c_{ij}} = \sum_{t=1}^{p}\left[(y_t - d_t)w_i \exp\left(-\frac{1}{2}u_{it}\right)\frac{(x_{jt} - c_{ij})}{\sigma_{ij}^2}\right], \qquad (3.21)$$

$$\frac{\partial E}{\partial \sigma_{ij}} = \sum_{t=1}^{p}\left[(y_t - d_t)w_i \exp\left(-\frac{1}{2}u_{it}\right)\frac{(x_{jt} - c_{ij})^2}{\sigma_{ij}^3}\right]. \qquad (3.22)$$

Применение градиентного метода наискорейшего спуска позволяет провести уточнение центров и ширины радиальных функций согласно формулам (3.23) и (3.24):

$$c_{ij}(t+1) = c_{ij}(t) - \eta\frac{\partial E}{\partial c_{ij}}, \qquad (3.23)$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta\frac{\partial E}{\partial \sigma_{ij}}. \qquad (3.24)$$

Уточнение нелинейных параметров радиальной функции завершает очередной цикл обучения. Многократное повторение обоих этапов ведет к полному и быстрому обучению сети, особенно когда начальные значения параметров радиальных функций близки к оптимальным значениям.

3.3.3. Применение метода обратного распространения ошибки для радиальных сетей

Обособленный класс алгоритмов обучения радиальных сетей составляют градиентные алгоритмы обучения с учителем, в которых используется алгоритм обратного распространения ошибки. Их основу составляет целевая функция, которая для одной обучающей выборки имеет вид:

$$E = \frac{1}{2}\left[\sum_{i=1}^{K} w_i\varphi_i(x) - d\right]^2 \qquad (3.25)$$

Предположим, что применяется самая общая форма гауссовской радиальной функции $\varphi_i(x)$, соответствующей сети *HRBF*, в которой

$$\varphi(x) = \exp\left[-\frac{1}{2}\left[Q_i(x - c_i)\right]^T\left[Q_i(x - c_i)\right]\right], \qquad (3.26)$$

а матрица Q_i имеет произвольную структуру.

Обучение сети с использованием алгоритма обратного распространения ошибки проводится в два этапа. На первом этапе предъявляется обучающая выборка и рассчитываются значения сигналов выходных нейронов сети, после чего рассчитывается фактическое значение целевой функции, заданной выражением (3.25). На втором этапе минимизируется значение этой функции.

Подбор значений параметров можно осуществлять, используя градиентные методы оптимизации независимо от объекта обучения – будь то вес или центр. Независимо от выбираемого метода градиентной оптимизации, необходимо, прежде всего, получить вектор градиента целевой функции относительно всех параметров сети. Для этого необходимо решить систему дифференциальных уравнений, представленную формулами:

$$\begin{cases} \dfrac{\partial E}{\partial Q_{ij}^{(1)}} = 0 \\ \dfrac{\partial E}{\partial w_i^{(2)}} = 0 \\ \dfrac{\partial E}{\partial c_i} = 0 \end{cases} \qquad (3.27)$$

Левые части уравнений в системе (3.27) представляют собой частные производные целевой функции по параметрам сети. Аналитические выражения для них сложны и неудобны для практического применения. Поэтому для расчета градиента целесообразно использовать метод потоковых графов, описанный в разделе 2.3.6. Граф сети *HRBF* представлен на рис.3.4.

Тогда аналитические выражения для частных производных можно записать в более простом виде:

$$\frac{\partial E}{\partial w_0} = y - d \qquad (3.28)$$

$$\frac{\partial E}{\partial w_i} = \exp\left(-\frac{1}{2}u_i\right)(y-d) \qquad (3.29)$$

$$\frac{\partial E}{\partial Q_{jr}^{(i)}} = v_t^{(i)}\hat{z}_t^{(i)} = -\exp\left(-\frac{1}{2}u_i\right)w_i(y-d)\left(x_j - c_j^{(i)}\right)z_r^{(i)} \qquad (3.30)$$

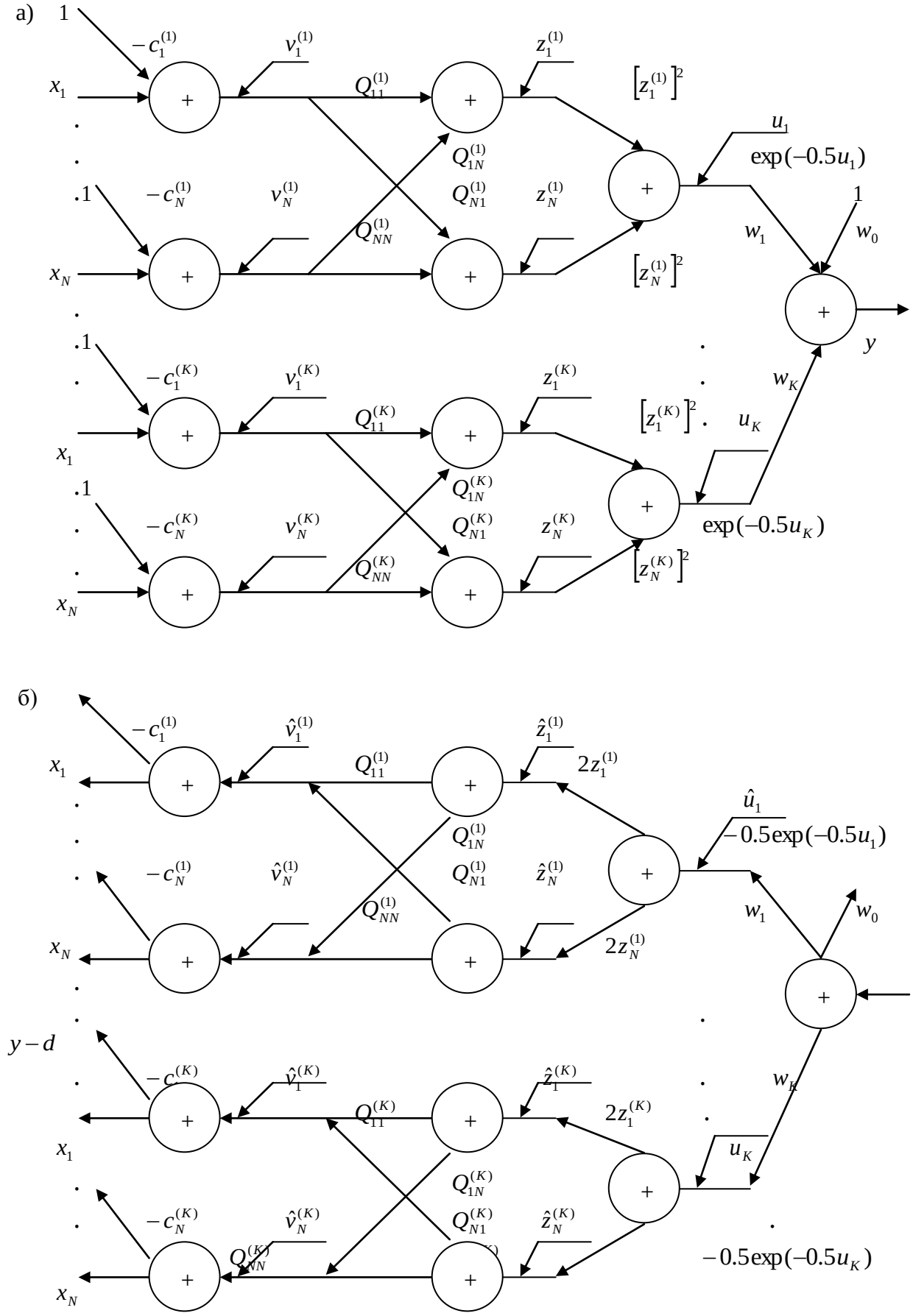


Рис. 3.4. Структура HRBF-сети, используемая для расчета градиента

а) исходная сеть; б) сопряженная сеть

где $z_t^{(i)} = \sum_{j=1}^{N} Q_{tj}^{(i)} (x_j - c_j^{(i)})$ (3.31)

i- индекс нейрона скрытого слоя, *i=1,2,…,K*;

$$u_i = \sum_{j=1}^{N} \left[z_j^{(i)} \right]^2 \quad (3.32)$$

j- индекс компонента входного вектора *x*, *j=1,2,…,N*;

r- индекс переменной в компоненте входного вектора x_r, *r=1,2,…,N*;

Q_{rj} – элементы масштабирующей матрицы *Q*.

3.4. Методы подбора числа базисных функций

Подбор числа базисных функций, каждой из которых соответствует один скрытый нейрон, считается основной проблемой, возникающей при корректном решении задачи аппроксимации, решаемой радиально-базисной сетью. Семейство сетей *RBF* является достаточно широким, чтобы равномерно аппроксимировать любую непрерывную функцию на компактном множестве. Теоретический базис построения нейронных сетей на основе радиальных базисных функций дает универсальная теорема об аппроксимации [5].

Пусть $G : \Re^N \to \Re$ - ограниченная, непрерывная и интегрируемая функция, такая, что

$$\int_{\Re^N} G(x)dx \neq 0. \quad (3.33)$$

Пусть G_G - семейство сетей *RBF*, включающих функцию $F : \Re^N \to \Re$ следующего вида:

$F(x) = \sum_{i=1}^{K} w_i G\left(\frac{x - c_i}{\sigma} \right)$, где $\sigma > 0, w_i \in \Re, c_i \in \Re^N$ для $i = 1, 2, \ldots, K$. Тогда выполняется следующая *теорема об универсальной аппроксимации* для сетей *RBF*.

Для любой непрерывной функции $f(x)$ *найдется сеть RBF с множеством центров* $\{c_i\}_{i=1}^{K}$ *и общей шириной* $\sigma > 0$, *такая, что функция* $F(x)$, *реализуемая сетью, будет близка к* $f(x)$ *по норме* $L_p, p \in [1, \infty]$.

Теорема является более строгой, чем необходимо для сетей *RBF*, так как ядро $G:\Re^N \to \Re$ не обязательно должно удовлетворять условию симметрии.

При подборе числа нейронов в скрытом слое приходится учитывать следующие факторы. Слишком малое число нейронов не позволяет сильно уменьшить погрешность обобщения множества обучающих данных, слишком большое число – увеличивает погрешность выводимого решения на множестве тестирующих данных. Как правило, число базисных функций K составляет определенную долю от объема обучающих данных p.

Наиболее эффективным алгоритмом подбора числа скрытых нейронов считается метод ортогонализации наименьших квадратов, использующий алгоритм ортогонализации Грэма-Шмидта [4].

Отправная точка этого метода – представление задачи обучения в виде линейной адаптации вектора весов сети $w=[w_0, w_1,...,w_K]^T$, направленной на минимизацию значения вектора погрешности e. Для p обучающих выборок вектор ожидаемых значений имеет вид: $d=[d_0, d_1,...,d_p]^T$. При использовании K базисных функций и p обучающих пар реакции скрытых нейронов образуют матрицу G вида (3.15)

$$G=\begin{bmatrix} \varphi_{11} & \varphi_{21} & \cdots & \varphi_{K1} \\ \varphi_{12} & \varphi_{22} & \cdots & \varphi_{K2} \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_{1p} & \varphi_{2p} & \cdots & \varphi_{Kp} \end{bmatrix}, \qquad (3.34)$$

в которой φ_{it} обозначает реакцию i-й радиальной функции на t-ю обучающую выборку, $\varphi_{it}=\varphi(\| x_t - c_i \|)$. Если вектор реакций i-й радиальной функции на все обучающие выборки обозначить $g_i=[\varphi_{i1}, \varphi_{i2},...,\varphi_{ip}]^T$, то матрицу G можно представить в форме $G=[g_1, g_2,...,g_K]$.

При таких обозначениях на каждом этапе обучения будет выполняться линейное равенство

$$d=Gw+e, \qquad (3.35)$$

где w – вектор весов, а $e=[e_1,e_2,...,e_p]^T$ - вектор фактической погрешности обучения. Квадрат произведения Gw соответствует ожидаемой энергии, исходящей от сигналов, задаваемых вектором d, которая и подвергается максимизации в процессе обучения.

Метод ортогонализации наименьших квадратов основан на преобразовании векторов g_i во множество базисных ортогональных векторов. В процессе обучения матрица $G \in R^{p\times K}$ раскладывается на произведение матрицы $Q \in R^{p\times K}$ с ортогональными столбцами q_i на верхнетреугольную матрицу $A \in R^{K\times K}$ с единичными диагональными значениями:

$$G = QA, \tag{3.36}$$

где $A=\begin{bmatrix} 1 & a_{12} & \dots & a_{1K} \\ 0 & 1 & \dots & a_{2K} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}$, а матрица Q соответствует условию $Q^TQ=H$. При этом H – диагональная матрица с элементами $H_{ii}=q_i^Tq_i=\sum_{t=1}^{p}q_{it}^2$. Решение зависимости (3.35) методом наименьших квадратов может быть спроецировано в пространство, образуемое ортогональными векторами q_i Если ввести новую векторную переменную b, определенную как

$$b = Aw, \tag{3.37}$$

то из уравнения (3.35) получим:

$$d = QB + e. \tag{3.38}$$

Приближенное решение уравнения (3.38) методом наименьших квадратов имеет вид:

$$\hat{b}=[Q^TQ]^{-1}Q^Td=H^{-1}Q^Td. \tag{3.39}$$

Принимая во внимание диагональный характер матрицы H, можно получить формулу, описывающую i-й компонент вектора $\hat{b}$:

$$\hat{b}_i=\frac{q_i^Td}{q_i^Tq_i}. \tag{3.40}$$

Решение, определяющее вектор весов *w*, находится непосредственно из уравнения (3.37), которое можно переписать в форме $\hat{w} = A^{-1}\hat{b}$.

3.5. Метод ортогонализации Грэма-Шмидта

Ортогонализация матрицы *Q*, описанная выражением (3.37), может быть проведена различными методами, наиболее эффективным из которых является алгоритм Грэма-Шмидта. В соответствии с этим методом матрица ***A*** формируется последовательно, столбец за столбцом с одновременным формированием очередных столбцов ортогональной матрицы *Q*. На *r-м* шаге создается столбец q_r**,** ортогональный ко всем созданным ранее (*r-1*) столбцам q_i (*i =1,2,...,r-1*). Процедура повторяется для значений *r=2,3,...,K*. Математическая модель этой операции имеет вид:

$$q_1 = g_1, \tag{3.41}$$

$$a_{ir} = \frac{q_i^T g_r}{q_i^T q_i}, \tag{3.42}$$

$$q_r = g_r - \sum_{i=1}^{r-1} a_{ir} q_i \ , \tag{3.43}$$

для $1 \le i \le r$, *r=2,3,...,K*. Многократно повторенная процедура ортогонализации позволяет сформировать все ортогональные векторы q_r и матрицу *A*, на основе которых можно найти вектор весов $\hat{w}$.

Однако важнейшим достоинством описываемого метода ортогонализации считается возможность селекции векторов q_i с учетом их важности для отображения обучающих данных. В случае априорно определенного количества *K* радиальных функций, задача заключается в такой расстановке векторов q_i, чтобы отобрать из них первые K_r наиболее значимые в энергетическом плане, при этом, как правило, $K_r << K$. Использование в дальнейших вычислениях только K_r радиальных функций означает сокращение количества скрытых нейронов с начального их числа *K* до K_r.

В качестве начального значения берется *K=p*.

Алгоритм отбора наиболее значимых базисных функций выглядит следующим образом:

1. На первом этапе (r=1) для $1 \le i \le K$ рассчитать

$$q_i(1) = g_i, \tag{3.44}$$

$$b_i(1) = \frac{[q_i(1)]^T d}{[q_i(1)]^T q_i(1)}, \qquad \varepsilon_i(1) = \frac{[b_i(1)]^2 [q_i(1)]^T q_i(1)}{d^T d}. \tag{3.45}$$

Предполагается, что $\varepsilon_{i_1}(1) = \max\{\varepsilon_i(1)\}$ для $1 \le i \le K$, а вектор $q_1 = q_{i_1} = g_{i_1}$.

2. На следующих этапах ($r \ge 2$) для $1 \le i \le K$, $i \ne i_1 \ne \ldots \ne i_{r-1}$ следует провести очередные циклы ортогонализации:

$$a_{ir}^{(i)} = \frac{q_i^T g_r}{q_i^T q_i}, \tag{3.46}$$

$$q_i(r) = g_i - \sum_{i=1}^{r-1} a_{ir}^{(i)} q_i, \tag{3.47}$$

а также оценить влияние очередных радиальных функций на суммарное значение энергетической функции путем расчета:

$$b_i(r) = \frac{[q_i(r)]^T d}{[q_i(r)]^T [q_i(r)]}, \qquad \varepsilon_i(r) = \frac{[b_i(r)]^T [q_i(r)]^T [q_i(r)]}{d^T d}. \tag{3.48}$$

Если наибольший вклад радиальной функции в общую энергию обозначить $\varepsilon_{i_r(r)}$, т.е. $\varepsilon_{i_r(r)} = \max\{\varepsilon_i(r)\}$ для $1 \le i \le K$, $i \ne i_1 \ne \ldots \ne i_{r-1}$, тогда очередной выделенный вектор q_r будет соответствовать радиальной функции со следующим по важности вкладом в общую энергию. Этот вектор определяется выражением:

$$q_r = q_{i_r}(r) = q_{i_r} - \sum_{i=1}^{r-1} a_{ir} q_i, \tag{3.49}$$

в котором коэффициент $a_{ir} = a_{ir}^{(i_r)}$ для $1 \le i \le r-1$.

3. Процедура выявления наиболее значимых для отображения радиальных функций завершается на этапе $r = K_r$, в момент выполнения условия

$$1 - \sum_{i=1}^{K_r} \varepsilon_i < \rho, \qquad (3.50)$$

где $0 < \rho < 1$ – это заранее установленный порог толерантности.

В результате выполнения процедуры в сети остается только K_r наиболее значимых радиальных функций, расположенных в ранее определенных центрах. Одновременно вычисляются конкретные составляющие вектора *b*, на основе которых по формуле (3.28) находятся значения весов *w* выходного слоя сети.

Еще одно достоинство процесса ортогонализации – возможность избежать неудачной комбинации параметров процесса обучения. Выполнение условия $q_r^T q_r = 0$, означает, что соответствующий вектор g_r является линейной комбинацией векторов $g_1, g_2, ..., g_{r-1}$. Поэтому если в процессе ортогонализации произведение $q_r^T q_r$ меньше, чем заданное значение, то функцию g_r можно не включать во множество базисных функций.

3.6. Сравнение радиально-базисной сети и многослойного персептрона

Сети *RBF* и *MLP* являются примерами нелинейных многослойных сетей прямого распространения. И те и другие являются универсальными аппроксиматорами, однако эти два типа сетей отличаются по некоторым важным аспектам.

1. Сети *RBF* (в своей основной форме) имеют один скрытый слой, в то время как многослойный персептрон может иметь большее число скрытых слоев.
2. Обычно скрытые и выходные нейроны сети *MLP* используют одну и ту же модель нейрона. Нейроны скрытого слоя сети *RBF* могут отличаться друг от друга и от нейронов выходного слоя.

3. Скрытый слой сети *RBF* является нелинейным, а выходной – линейным. В то же время скрытые и выходной слой сети *MLP* являются нелинейными. Если сеть *MLP* используется для решения задач нелинейной регрессии, в качестве выходных нейронов выбираются линейные нейроны.
4. Аргумент функции активации каждого скрытого нейрона сети *RBF* представляет собой *эвклидову меру* между входным вектором и центром радиальной функции. Аргументом функции активации каждого скрытого нейрона сети *MLP* является *скалярное произведение* входного вектора и вектора синаптических весов данного нейрона.
5. Сеть *MLP* обеспечивает *глобальную* аппроксимацию нелинейного отображения. Сеть *RBF* создает *локальную* аппроксимацию нелинейного отображения.

Аппроксимация нелинейного отображения с помощью многослойного персептрона может потребовать меньшее число параметров, чем для радиально-базисной сети при одинаковой точности вычислений.

3.7. Контрольные вопросы и упражнения

1. Сравните структуру и характеристики радиально-базисной сети и многослойного персептрона.

2. Сколько скрытых слоев в радиально-базисных сетях?

3. Можно использовать разные типы функций активации для различных скрытых нейронов?

4. Объясните, чем *HRBF* сети отличаются от *RBF* сетей. Обоснуйте свой ответ.

5. Какие типы алгоритмов обучения можно использовать в радиально-базисных сетях: алгоритмы обучения «с учителем»; алгоритмы обучения «без учителя».

6. Исследуйте аппроксимирующие свойства радиально-базисных сетей методом компьютерного моделирования для построения следующих отображений:

а) $f(x) = \frac{1}{x}, 1 <= x <= 100$;

б) $f(x) = \lg_{10}(x), 1 <= x <= 10$;

в) $f(x) = \exp(-x), 1 <= x <= 10$;

г) $f(x) = \sin(x), 1 <= x <= \pi / 2$.

Для проведения исследований выполните следующие действия:

а) Создайте два набора данных: один – для обучения, второй – для тестирования;

б) Оцените сложность сети при помощи *VC*-измерения;

в) Исследуйте, как изменение числа нейронов в скрытом слое влияет на ошибку аппроксимации;

г) Исследуйте, как изменение функции активации влияет на ошибку аппроксимации (используйте функцию Гаусса и мультиквадратичную функцию $(x^2 + c_1^2)^{0.5}$);

д) Исследуйте, как объем обучающей выборки влияет на ошибку аппроксимации.

4. Сети с самоорганизацией на основе конкуренции

4.1. Сеть Кохонена

Основу самоорганизации нейронных сетей составляет подмеченная закономерность, что глобальное упорядочение сети становится возможным в результате *самоорганизующих* операций, независимо друг от друга проводящихся в различных локальных сегментах сети. В соответствии с поданными сигналами осуществляется *активация* нейронов, в результате чего активным оказывается один нейрон в сети (или в группе). Выходной нейрон, который выиграл соревнование, называется *нейроном-победителем*.

Нейроны в ходе конкурентного процесса, вследствие изменения значений синаптических весов, избирательно настраиваются на различные входные векторы или классы входных векторов. В процессе обучения наблюдается тенденция к росту значений весов, из-за которой создается своеобразная

положительная обратная связь: более мощные возбуждающие импульсы – более высокие значения весов – большая активность нейронов.

При этом происходит естественное расслоение нейронов на различные группы, отдельные нейроны или их группы сотрудничают между собой и активизируются в ответ на возбуждение, создаваемое конкретными обучающими векторами, подавляя своей активностью другие нейроны. Можно говорить как о сотрудничестве между нейронами внутри группы, так и о конкуренции между нейронами внутри группы и между различными группами.

Среди механизмов самоорганизации можно выделить два основных класса: самоорганизация, основанная на ассоциативном правиле Хебба, и механизм конкуренции нейронов на базе обобщенного правила Кохонена. В дальнейшем будем рассматривать механизм конкуренции нейронов.

Нейроны помещаются в узлах решетки, обычно одно- или двумерной. Сети более высокой размерности также возможны, но используются достаточно редко. Как правило, это однослойные сети прямого распространения, в которых нейрон соединен со всеми компонентами N-мерного входного вектора x так, как это схематично изображено для $N = 2$ на рис. 4.1 [4].

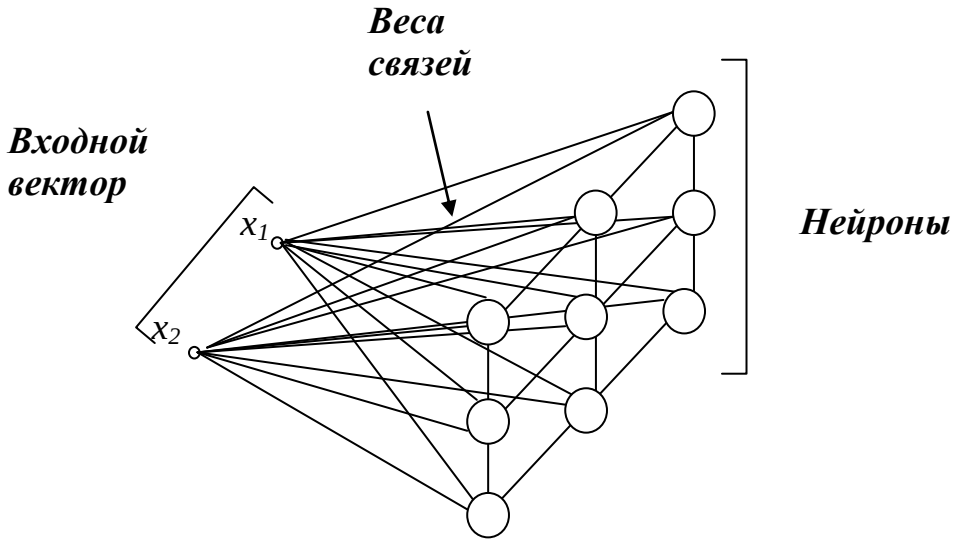


Рис. 4.1 Структура самоорганизующейся сети Кохонена.

Формирование самоорганизующихся сетей начинается с инициализации синаптических весов сети. Обычно синаптическим весам присваиваются малые значения, которые формируются генератором случайных чисел. При такой инициализации сеть изначально не имеет какого-либо порядка признаков входных векторов. После инициализации сети реализуются три основных процесса [5]:

1. *Конкуренция.* Для каждого входного вектора нейроны сети вычисляют относительные значения дискриминантной функции.
2. *Кооперация.* Победивший нейрон определяет топологическую окрестность группы нейронов, обеспечивая базис для кооперации между ними.
3. *Синаптическая адаптация*. Корректировка синаптических весов возбужденных нейронов позволяет увеличить их собственные значения дискриминантных функций по отношению к входным векторам. Корректировка производится таким образом, чтобы выходной сигнал нейрона-победителя усиливался при последующем применении аналогичных входных векторов.

Веса синаптических связей нейронов образуют вектор $w_i = [w_{i1}, w_{i2}, ..., w_{iN}]^T$. После нормализации входных векторов при активации сети вектором x в конкурентной борьбе побеждает тот нейрон, веса которого в наименьшей степени отличаются от соответствующих компонентов этого вектора. Для w-го нейрона-победителя выполняется соотношение

$$d(x, w_w) = \min_{1 \le i \le K} d(x, w_i) \qquad (4.1)$$

где $d(x, w)$ обозначает расстояние между векторами x и w, а K - количество нейронов. Вокруг нейрона-победителя образуется топологическая окрестность $S_w(t)$ с определенной энергетикой, уменьшающейся с течением времени. Нейрон-победитель и все нейроны, лежащие в пределах его окрестности, подвергаются адаптации, в ходе которой их векторы весов изменяются в направлении вектора x по правилу Кохонена:

$$w_i(t+1) = w_i(t) + \eta_i(t)(x - w_i(t)) \qquad (4.2)$$

для $i \in S_w(t)$, где $\eta_i(t)$ - коэффициент обучения i-го нейрона на окрестности $S_w(t)$ в t-й момент времени. Значение $\eta_i(t)$ уменьшается с увеличением расстояния между i-м нейроном и победителем. Веса нейронов, находящихся вне окрестности $S_w(t)$, не изменяются. Размер окрестности и коэффициенты обучения нейронов являются функциями, значения которых уменьшаются с течением времени. В [10] доказано, что адаптация по формуле (4.2) эквивалентна градиентному методу обучения, основанному на минимизации целевой функции

$$E(w) = \frac{1}{2}\sum_{i,j,t} S_i(x(t))[x_j(t) - w_{ij}(t)]^2 \quad , \qquad (4.3)$$

а $S_i(x(t))$ представляет собой функцию определения окрестности, изменяющуюся в процессе обучения.

После предъявления двух различных векторов x_1 и x_2 активизируются два нейрона сети, веса которых наиболее близки к координатам соответствующих векторов. Эти веса, обозначенные w_1 и w_2, могут отображаться в пространстве как две точки. Сближение векторов x_1 и x_2 вызывает соответствующее изменение в расположении векторов w_1 и w_2 В пределе равенство $w_1 = w_2$ выполняется тогда и только тогда, когда x_1 и x_2 совпадают или практически неотличимы друг от друга. Сеть, в которой эти условия выполняются, называется топографической картой или картой Кохонена.

4.2. Меры расстояния между векторами и нормализация векторов

Процесс самоорганизации предполагает определение победителя каждого этапа, то есть нейрона, вектор весов которого в наименьшей степени отличается от поданного на вход сети вектора x. В этой ситуации важной проблемой становится выбор метрики, в которой будет измеряться расстояние между векторами x и w_i. Чаще всего в качестве меры расстояния используются:

- эвклидова мера

$$d(x, w_i) = \|x - w_i\| = \sqrt{\sum_{j=1}^{N} (x_j - w_{ij})^2}\ ; \qquad (4.4)$$

- скалярное произведение

$$d(x, w_i) = 1 - x \cdot w = 1 - \|x\| \cdot \|w_i\| \cdot \cos(x, w_i)\ ; \qquad (4.5)$$

- мера относительно нормы L1 (Манхэттен)

$$d(x, w_i) = \sqrt{\sum_{j=1}^{N} \left| x_j - w_{ij} \right|}\ ; \qquad (4.6)$$

- мера относительно нормы L_∞

$$d(x, w_i) = \max_j \left| x_j - w_{ij} \right|. \qquad (4.7)$$

При использовании эвклидовой меры разбиение пространства на зоны доминирования нейронов равносильно разбиению на области Вороного. При использовании другой меры формируется другое разделение областей влияния нейронов. Использование скалярного произведения требует нормализации входных векторов, так как в противном случае возможно неравномерное распределение нейронов: в одной области может находиться несколько нейронов, а в другой – ни одного нейрона.

При нормализованных входных обучающих векторах стремящиеся к ним векторы весов нормализуются автоматически. Однако нормализация векторов приводит к тому, что если $\|w_i\| = const$, то для всех нейронов при фиксированном значении x произведение $\|x\|\|w_i\|$ также становится постоянной величиной. Поэтому активация нейрона определяется значением $\cos(x, w_i)$, которое становится общей мерой для всей сети. Следует отметить, что при нормализации вектора весов эвклидова мера и скалярное произведение равнозначны друг другу, так как $\|x - w_i\|^2 = \|x\|^2 + \|w_i\|^2 - 2x^T w_i$. Поэтому $\min\|x - w\|^2 = \max\left(x^T w_i\right)$ при $\|w_i\| = const$. Экспериментальные исследования подтвердили необходимость применения нормализации векторов при малой

размерности пространства [4]. Такая нормализация проводится двумя способами:

1. Переопределение компонентов вектора в соответствии с формулой

$$x_i \leftarrow \frac{x_i}{\sqrt{\sum_{i=1}^{N} x_i^2}}. \qquad (4.8)$$

2. Увеличением размерности пространства на одну координату с таким выбором $(N+1)$-го компонента вектора, чтобы выполнялось условие:

$$\sum_{i=1}^{N+1} x_i^2 = 1. \qquad (4.9)$$

При использовании второго способа возникает необходимость предварительного масштабирования компонентов вектора x.

С увеличением размерности входного вектора эффект нормализации становится менее заметным, а при размерности $N > 200$ вообще сходит на нет.

4.3. Проблема мертвых нейронов

При инициализации весов сети случайным способом часть нейронов может оказаться в области пространства, в которой отсутствуют данные или их количество ничтожно мало. Эти нейроны имеют мало шансов на победу и адаптацию своих весов, поэтому они остаются мертвыми. Таким образом, входные данные будут интерпретироваться меньшим количеством нейронов, а погрешность интерпретации данных увеличится. Поэтому важной проблемой становится активация всех нейронов сети, которую можно осуществить, если в алгоритме обучения предусмотреть учет количества побед каждого нейрона, а процесс обучения организовать так, чтобы дать шанс победить и менее активным нейронам.

Существуют различные механизмы учета активности нейронов в процессе обучения [4]. Часто используется метод подсчета потенциала p_i каждого нейрона, значение которого модифицируется всякий раз после предъявления

очередной реализации входного вектора x в соответствии со следующей формулой (в ней предполагается, что победителем стал w-й нейрон):

$$p_i(k+1) = \begin{cases} p_i(k) + \dfrac{1}{n}; (i \neq w) \\ p_i(k) - p_{\min}; (i = w) \end{cases}. \qquad (4.10)$$

Значение коэффициента $p_{\min}$ определяет минимальный потенциал, разрешающий участие в конкурентной борьбе. Максимальное значение потенциала ограничивается на уровне, равном 1. Выбор конкретного значения $p_{\min}$ позволяет установить порог готовности нейрона к конкурентной борьбе. При $p_{\min}$=0 утомляемость нейронов не возникает, и каждый из них сразу после победы будет готов к продолжению соперничества. При $p_{\min}$=1 возникает другая крайность, вследствие которой нейроны побеждают по очереди, так как в каждый момент только один из них оказывается готовым к соперничеству. На практике хорошие результаты достигаются при $p_{\min} \approx 0{,}75$.

В другом очень удачном алгоритме обучения количество побед нейрона учитывается при подсчете эффективного расстояния между вектором весов и реализацией обучающего вектора x. Это расстояние модифицируется пропорционально количеству побед данного нейрона в прошлом. Если обозначить количество побед i-го нейрона N_i, такую модификацию можно представить в виде

$$d(x, w_i) \leftarrow N_i d(x, w_i). \qquad (4.11)$$

Активные нейроны с большим значением N_i штрафуются искусственным завышением этого расстояния. Отметим, что модификация расстояния производится только при выявлении победителя. В момент уточнения весов учитывается фактическое расстояние. Обычно после двух или трех циклов обучения модификация прекращается, что позволяет продолжить «честную» конкуренцию нейронов.

4.4. Алгоритмы обучения без учителя

4.4.1. Алгоритм WTA

Алгоритмы обучения, используемые для обучения нейронных сетей Кохонена, называются алгоритмами обучения без учителя. Подобные алгоритмы применяются в тех случаях, когда нет эталонных выходных значений для входных векторов.

Целью обучения сети с самоорганизацией на основе конкуренции, считается такое упорядочение нейронов, которое минимизирует значение отклонения вектора весов от входного вектора *x*. При *p* входных векторах *x* эта погрешность в эвклидовой метрике может быть выражена в виде:

$$E_q = \frac{1}{2}\sum_{t=1}^{p}\left\| x_i - w_{w(t)} \right\|^2, \qquad (4.12)$$

где $w_{w(t)}$ - это вес нейрона-победителя при предъявлении вектора x_i.

Этот подход также называется *векторным квантованием* (*VQ*). Номера нейронов-победителей при последовательном предъявлении векторов *x* образуют так называемую кодовую таблицу. При классическом решении задачи кодирования применяется алгоритм *K-усреднений*, носящий имя обобщенного алгоритма Ллойда.

Для нейронных сетей аналогом алгоритма Ллойда считается алгоритм *WTA (Winner Takes All* – победитель получает все). В соответствии с ним после предъявления вектора *x* рассчитывается активность каждого нейрона. Победителем признается нейрон с самым сильным выходным сигналом, то есть тот, для которого скалярное произведение $\left(x^T w\right)$ оказывается наибольшим, что для нормализованных векторов равнозначно наименьшему эвклидову расстоянию между входным вектором и вектором весов нейронов. Победитель получает право уточнить свои веса в направлении вектора *x* согласно правилу

$$w_w \leftarrow w_w + \eta(x - w_w). \qquad (4.13)$$

Веса остальных нейронов уточнению не подлежат. Алгоритм позволяет учитывать усталость нейронов путем подсчета количества побед каждого из

них и поощрять элементы с наименьшей активностью для выравнивания их шансов.

Помимо алгоритмов *WTA,* в которых в каждой итерации может обучаться только один нейрон, для обучения сетей с самоорганизацией широко применяется алгоритмы типа *WTM* (*Winner Takes Most* – победитель получает больше), в которых, кроме победителя, уточняют значения своих весов и нейроны из его ближайшего окружения. При этом, чем дальше какой-либо нейрон находится от победителя, тем меньше изменяются его веса. Процесс уточнения вектора весов может быть определен в виде

$$w_i \leftarrow w_i + \eta_i G(i,x)(x - w_i) \qquad (4.14)$$

для всех i нейронов, расположенных в окрестности победителя. В приведенной формуле коэффициент обучения η_i каждого нейрона отделен от его расстояния до предъявленного вектора x функцией $G(i,x)$. Если $G(i,x)$ определяется в форме

$$G(i,x) = \begin{cases} 1, & для\ \ i = w \\ 0, & для\ \ i \neq w \end{cases}, \qquad (4.15)$$

для w обозначает номер победителя, то мы получаем классический алгоритм *WTA.* Существует множество вариантов алгоритма *WTM*, отличающихся ,прежде всего формой функции $G(i,x)$. Для дальнейшего обсуждения выберем классический алгоритм Кохонена и алгоритм нейронного газа.

4.4.2. Алгоритм Кохонена

Алгоритм Кохонена относится к наиболее старым алгоритмам обучения сетей с самоорганизацией на основе конкуренции, и в настоящее время существуют различные его версии [4]. В классическом алгоритме Кохонена сеть инициализируется путем приписывания нейронам определенных позиций в пространстве и связывании их с соседями на постоянной основе. В момент выбора победителя уточняются не только его веса, но также веса и его соседей, находящихся в ближайшей окрестности. Таким образом, нейрон-победитель подвергается адаптации вместе со своими соседями.

$$G(i,x)=\begin{cases}1, для\ -K<d(i,w)<K\\ 0, иначе\end{cases}, \qquad (4.16)$$

В этом выражении $d(i,w)$ может обозначать как эвклидово расстояние между векторами весов нейрона-победителя w и i-го нейрона, так и расстояние, измеряемое количеством нейронов.

Другой тип соседства в картах Кохонена- это соседство гауссовского типа, при котором функция $G(i,x)$ определяется формулой

$$G(i,x)=\exp\left(-\frac{d^2(i,w)}{2\lambda^2}\right). \qquad (4.17)$$

Уточнение весов нейронов происходит по правилу:

$$w_i(t)=w_i(t-1)+\eta(t)\cdot G(i,w)\cdot(x-w_i). \qquad (4.18)$$

Степень адаптации нейронов-соседей определяется не только эвклидовым расстоянием между i-м нейроном и нейроном-победителем (w-м нейроном) $d(i,w)$, но также уровнем соседства λ. Как правило, гауссовское соседство дает лучшие результаты обучения и обеспечивает лучшую организацию сети, чем прямоугольное соседство.

4.4.3. Алгоритм нейронного газа

Значительно лучшую самоорганизацию сети и ускорение сходимости алгоритма *WTM* можно получить применением метода, предложенного в [7] и названного авторами алгоритмом нейронного газа из-за подобия его динамики движению молекул газа.

В этом алгоритме на каждой итерации все нейроны сортируются в зависимости от их расстояния до вектора x. После сортировки нейроны размечаются в последовательности, соответствующей увеличению удаленности

$$d_0<d_1<d_2<\ldots<d_{K-1}, \qquad (4.19)$$

где $d_i=\left\|x-w_{m(i)}\right\|$ обозначает удаленность i-го нейрона, занимающего в результате сортировки m-ю позицию в последовательности, возглавляемой

нейроном-победителем, которому сопоставлена удаленность d_0. Значение функции соседства для i-го нейрона $G(i,x)$ определяется по формуле

$$G(i,x) = \exp\left(-\frac{m(i)}{\lambda}\right), \qquad (4.20)$$

в которой $m(i)$ обозначает очередность, полученную в результате сортировки ($m(i)$=0,1,2,…,K-1), а λ – параметр, аналогичный уровню соседства в алгоритме Кохонена, уменьшающийся с течением времени. При $\lambda = 0$ адаптации подвергается только нейрон-победитель, и алгоритм превращается в обычный алгоритм *WTA*, но при $\lambda \neq 0$ уточнению подлежат веса многих нейронов, причем уровень уточнения зависит от величины $G(i,x)$. Алгоритм нейронного газа напоминает стратегию нечетких множеств, в соответствии с которой каждому нейрону приписывается значение функции принадлежности к окрестности, определенной отношением (4.19).

Для достижения хороших результатов самоорганизации процесс обучения должен начинаться с большого значения λ, однако с течением времени его величина уменьшается до нуля. Изменение $\lambda(t)$ может быть линейным или показательным. Предложено изменять значение $\lambda(t)$ в соответствии с выражением

$$\lambda(t) = \lambda_{\max}\left(\frac{\lambda_{\min}}{\lambda_{\max}}\right)^{t/t_{\max}}, \qquad (4.21)$$

где $\lambda(t)$ обозначает значение λ на t-й итерации, а $\lambda_{\min}$ и $\lambda_{\max}$ – принятые минимальное и максимальное значения λ соответственно. Коэффициент $t_{\max}$ определяет максимальное заданное количество итераций.

Коэффициент обучения i-го нейрона $\eta_i(t)$ тоже может изменяться как линейно, так и показательно, причем его степенная изменчивость определяется формулой

$$\eta_i(t) = \eta_i(0)\left(\frac{\eta_{\min}}{\eta_i(0)}\right)^{t/t_{\max}}, \qquad (4.22)$$

в которой $\eta_i(0)$ обозначает начальное значение коэффициента обучения, $\eta_{\min}$ – априорно заданное минимальное значение, соответствующее $t = t_{\max}$. На практике наилучшие результаты самоорганизации достигаются при линейном изменении $\eta_i(t)$.

Для сокращения объема вычислений, необходимых для реализации алгоритма нейронного газа, можно применить определенное упрощение, состоящее в учете при сортировке только нейронов с наиболее значимой величиной функции $G(i,x)$. При этом используется зависимость (4.20), в соответствии с которой если $m(i) >> 1$, то значение $G(i,x) \cong 0$.

Алгоритм нейронного газа наряду с алгоритмом *WTA*, учитывающим активность нейронов, считается одним из наиболее эффективных средств самоорганизации нейронов в сети Кохонена. При соответствующем подборе параметров управления процессом можно добиться очень хорошей организации сети при скорости функционирования, превышающей скорость, достижимую в классическом алгоритме Кохонена.

4.5. Сети обратного распространения

4.5.1. Структура сети

Недостатком сетей с самоорганизацией на основе конкуренции считается сложность отображения пар обучающих данных (x,d), поскольку сеть не обладает свойством хорошего аппроксиматора, присущему многослойному персептрону или радиально-базисной сети. Хорошие результаты удается получить при объединении самоорганизующегося слоя и персептронного слоя или слоя Гроссберга. Объединение сети Кохонена и звезды Гроссберга называется сетью обратного распространения [17].

На рис. 4.2 показана упрощенная версия прямого действия сети обратного распространения. На нем иллюстрируются функциональные свойства данной парадигмы. Полная двунаправленная сеть основана на тех же принципах.

Нейроны слоя (0) служат лишь точками разветвления и не выполняют вычислений. Каждый нейрон слоя (0) соединен с каждым нейроном слоя (1),

называемого слоем Кохонена, отдельным весом w_{KN}. Эти веса в целом рассматриваются как матрица весов *w*. Аналогично, каждый нейрон в слое Кохонена соединен с каждым нейроном в слоя (2), называемого слоем Гроссберга, весом v_{MK}. Эти веса образуют матрицу весов *v*.

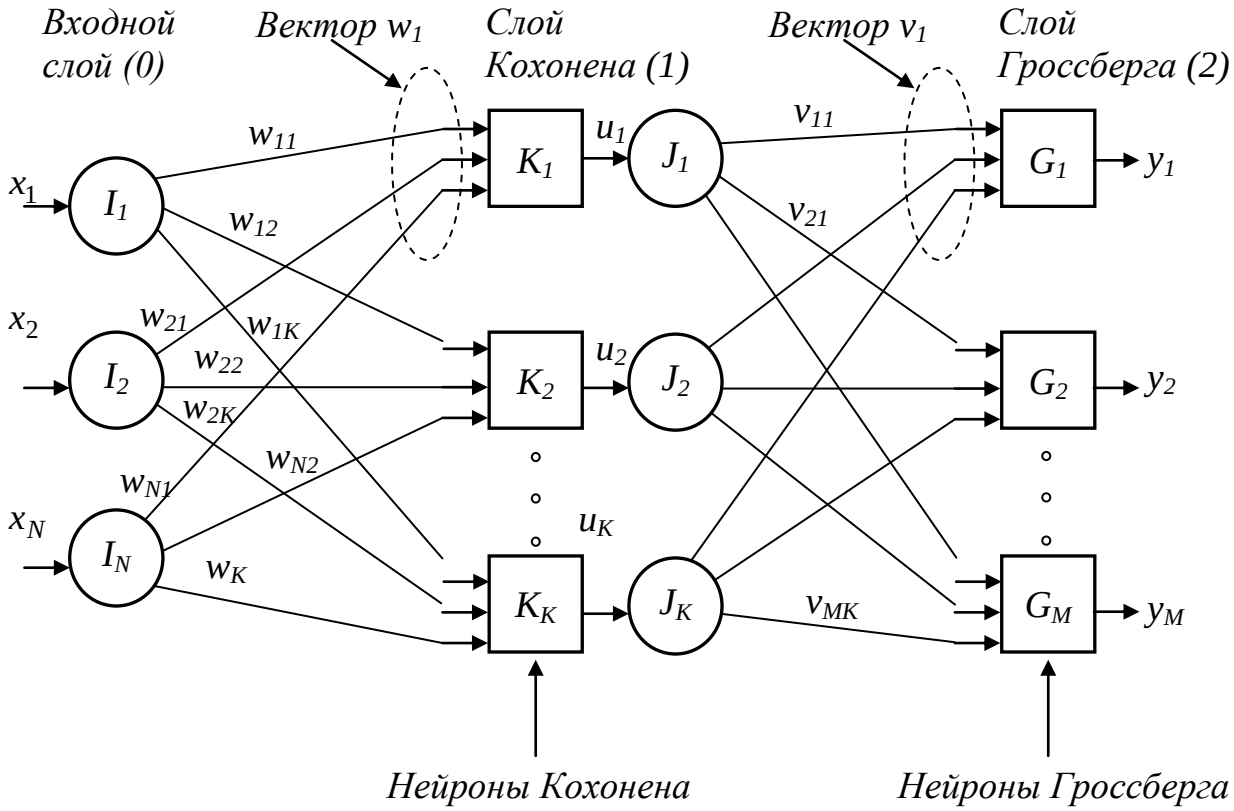


Рис. 4.2 Сеть обратного распространения без обратных связей

Сети обратного распространения функционируют в двух режимах: в нормальном режиме, при котором принимается входной вектор *x* и выдается выходной вектор *y*; и в режиме обучения, при котором подается входной вектор *x* и корректируются веса, чтобы получить требуемый эталонный вектор.

4.5.2. Нормальное функционирование сети обратного распространения

В своей простейшей форме слой Кохонена функционирует по принципу «победитель получает все», то есть для данного входного вектора только один

нейрон Кохонена выдает на выходе логическую единицу, все остальные выдают ноль.

Каждый нейрон Кохонена соединён с каждой компонентой входного вектора. Подобно нейронам большинства сетей выход u каждого нейрона Кохонена является просто суммой взвешенных входов. Это может быть выражено следующей формулой:

$$u_i = w_{i1} \cdot x_1 + w_{i2} \cdot x_2 + \ldots + w_{iN} \cdot x_N , \qquad (4.23)$$

где u_i – это выход u i –го нейрона Кохонена,

$$u_i = \sum_{j=1}^{N} x_j \cdot w_{ij} \qquad (4.24)$$

или в векторной записи

$$u = x \cdot w. \qquad (4.25)$$

Нейрон Кохонена с максимальным значением u является нейроном-победителем.

Слой Гроссберга функционирует в сходной манере. Его выход y является взвешенной суммой выходов $u_1, u_2, \ldots, u_K$ слоя Кохонена, образующих вектор u. Вектор соединяющих весов, обозначенный через v, состоит из весов $v_{11}, v_{21}, \ldots, v_{MK}$. Тогда выход y каждого нейрона Гроссберга есть

$$y_s = \sum_{i=1}^{M} u_{s_i} \cdot v_{si} , \qquad (4.26)$$

где y_s – выход s-го нейрона Гроссберга, или в векторной форме

$$y = u \cdot v , \qquad (4.27)$$

где y – выходной вектор слоя Гроссберга, u – выходной вектор слоя Кохонена, v – матрица весов слоя Гроссберга.

Если слой Кохонена функционирует таким образом, что лишь один элемент вектора u отличен от нуля, то вычисления очень просты. Фактически каждый нейрон слоя Гроссберга лишь выдает величину веса, который связывает этот нейрон с единственным ненулевым нейроном Кохонена.

4.5.3. Структура полной сети обратного распространения

На рис. 4.3 показана структура полной сети обратного распространения. В режиме нормального функционирования предъявляются входные векторы X_1 и X_2, и обученная сеть дает на выходе векторы Y_1 и Y_2, являющиеся аппроксимациями соответственно для X_1 и X_2. Предполагается, что векторы X_1 и X_2 являются нормализованными единичными векторами, следовательно, выходные векторы также будут нормализованными.

В процессе обучения векторы X_1 и X_2 подаются одновременно и как входные векторы сети, и как эталонные выходные сигналы.

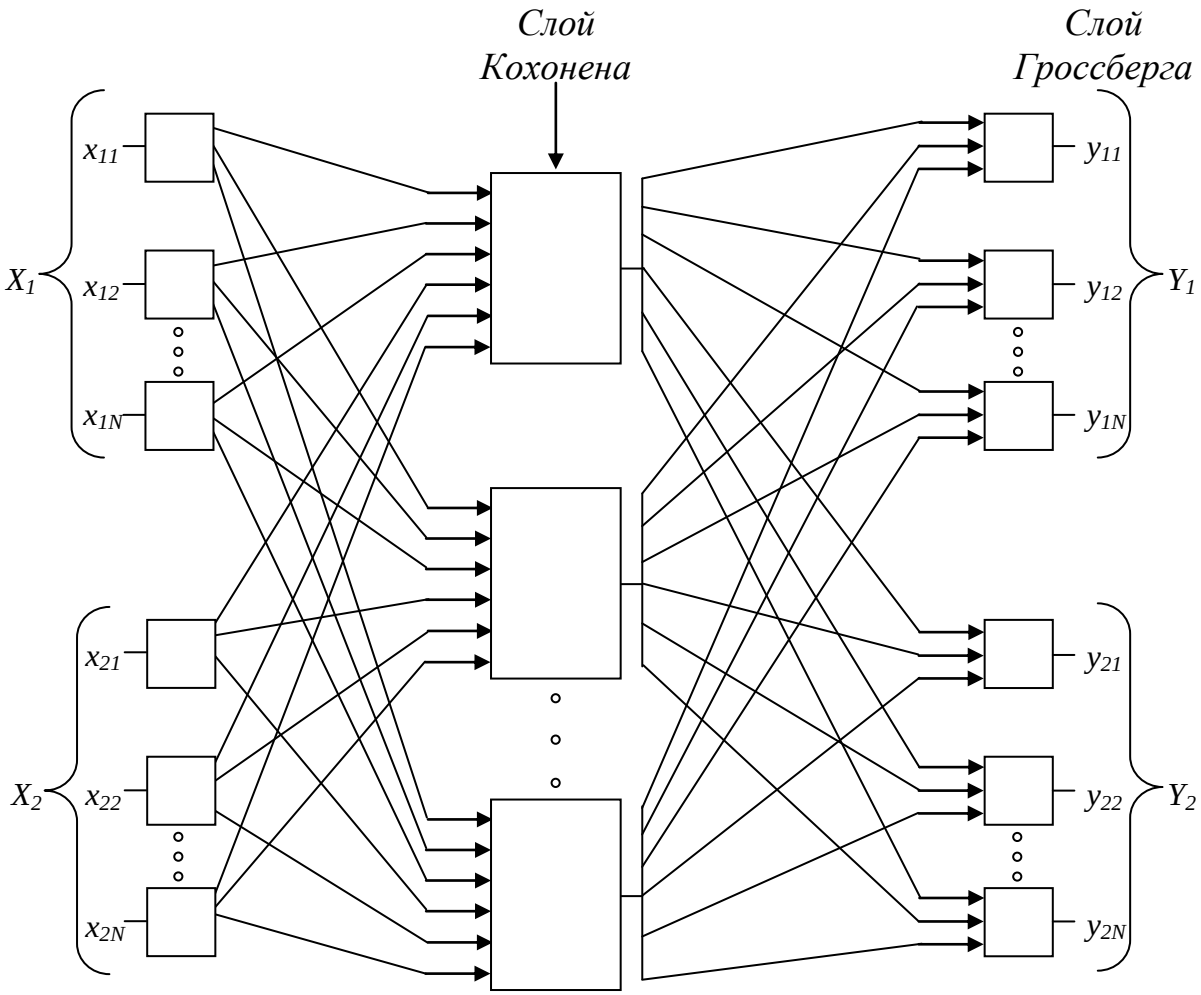


Рис. 4.3 Общая структура гибридной сети

Вектор X_1 используется для обучения выходов Y_1, а вектор X_2 – для обучения выходов Y_2 слоя Гроссберга. Гибридная сеть обучается с использованием того же самого метода, который описывался для сети прямого действия. Нейроны Кохонена принимают входные сигналы как от вектора X_1 так и от вектора X_2, что аналогично ситуации, когда имеется один общий вектор, составленный из векторов X_1 и X_2.

В качестве результирующего получается единичное отображение, при котором предъявление пары входных векторов порождает их копии на выходе. Это не представляется особенно интересным, если не заметить, что предъявление только вектора X_1 порождает как выходы Y_1 так и выходы Y_2 Если F – функция, отображающая X_1 в Y_2, то сеть аппроксимирует ее. Также, если F обратима, то предъявление только вектора X_2 порождает Y_1. Уникальная способность порождать функцию и обратную к ней делает сеть встречного распространения полезной в ряде приложений.

Сеть обратного распространения быстро обучается и при правильном использовании может сэкономить значительное количество машинного времени. Она полезна также для быстрого моделирования систем, где большая точность обратного распространения вынуждает отдать ему предпочтение в окончательном варианте, но важна быстрая начальная аппроксимация. Возможность порождать функцию и обратную к ней также нашло применение в ряде систем.

4.5.4. Анализ методов обучения сети обратного распространения

Обучение слоя Кохонена реализуется по одному из приведенных выше алгоритмов обучения сетей Кохонена.

Слой Гроссберга обучается относительно просто. Входной вектор, являющийся выходом слоя Кохонена, подается на слой нейронов Гроссберга, при этом выходы слоя Гроссберга вычисляются так же, как при нормальном функционировании. Далее, каждый вес корректируется лишь в том случае, если он соединен с нейроном Кохонена, имеющим ненулевой выход. Величина

коррекции веса пропорциональна разности между весом и требуемым выходом нейрона Гроссберга, с которым он соединен. В символьной записи:

$$v_{si}(t+1) = v_{si}(t) + \beta \cdot (y_s - v_{si}(t)) \cdot u_i, \qquad (4.28)$$

где u_i – выход i-го нейрона Кохонена (только для одного нейрона Кохонена он отличен от нуля); y_s – j-я компонента эталонного вектора выходов, t – номер входного вектора.

Первоначально β берется равным ~0,1 и затем постепенно уменьшается в процессе обучения. Отсюда видно, что веса слоя Гроссберга будут сходиться к средним величинам эталонных значений выходов, тогда как веса слоя Кохонена обучаются на средних значениях входов. Обучение слоя Гроссберга – это обучение с учителем, алгоритм располагает эталонным выходом, по которому он обучается. Обучающийся без учителя, самоорганизующийся слой Кохонена дает выходы в недетерминированных позициях. Они отображаются в эталонные выходы слоем Гроссберга.

Для сети обратного распространения рекомендуется нормализовать входные векторы перед тем, как подавать их на вход сети. Нормализация выполняется с помощью деления каждой компоненты входного вектора на длину вектора по следующей формуле (4.8).

Всем весам сети перед началом обучения следует придать начальные значения. Общепринятой практикой при работе с нейронными сетями является присваивание весам небольших случайных значений.

Рандомизация весов слоя Кохонена может породить проблеме мертвых нейронов. Более того, оставшихся весов, дающих наилучшие соответствия, может оказаться слишком мало, чтобы разделить входные векторы на классы, которые расположены близко друг к другу.

Допустим, что имеется несколько множеств входных векторов, все множества сходные, но должны быть разделены на различные классы. Сеть должна быть обучена активировать отдельный нейрон Кохонена для каждого класса. Если начальная плотность весовых векторов в окрестности обучающих

векторов слишком мала, то может оказаться невозможным разделить сходные классы.

Наоборот, если несколько входных векторов получены незначительными изменениями из одного и того же образца и должны быть объединены в один класс, то они должны включать один и тот же нейрон Кохонена. Если же плотность весовых векторов очень высока вблизи группы слегка различных входных векторов, то каждый входной вектор может активировать отдельный нейрон Кохонена. Это не является катастрофой, так как слой Гроссберга может отобразить различные нейроны Кохонена в один и тот же выход, однако при этом нейроны Кохонена будут использоваться не рационально.

Наиболее рациональное решение состоит в том, чтобы распределять весовые векторы в соответствии с плотностью входных векторов, которые должны быть разделены. На практике это невыполнимо, однако существует несколько методов приближенного достижения тех же целей.

Одно из решений, известное под названием метода выпуклой комбинации, состоит в том, что все веса приравниваются одной и той же величине:

$$w_i = \frac{1}{\sqrt{N}}, \qquad (4.29)$$

где N – число входов и, следовательно, число компонент каждого вектора весов. Благодаря этому все векторы весов совпадают и имеют единичную длину. Каждой же компоненте входа X придается значение

$$x_j = \alpha \cdot x_j + \frac{1-\alpha}{\sqrt{N}}, \qquad (4.30)$$

где N – число входов. В начале α очень мало, вследствие чего все входные векторы имеют длину, близкую к $1/\sqrt{\mathrm{N}}$, и почти совпадают с векторами весов. В процессе обучения сети α постепенно возрастает, приближаясь к единице. Это позволяет разделять входные векторы и окончательно приписывает им их истинные значения. Векторы весов отслеживают один или небольшую группу входных векторов и в конце обучения дают требуемую картину выходов.

Метод выпуклой комбинации хорошо работает, но замедляет процесс обучения, так как весовые векторы подстраиваются к изменяющейся цели.

Другой подход состоит в добавлении шума к входным векторам. При этом они подвергаются случайным изменениям, приближаясь в конце концов к вектору весов. Этот метод также работоспособен, но работает еще медленнее, чем метод выпуклой комбинации.

Третий метод предлагает начинать работу с присвоения случайных значений весам, но на начальной стадии обучающего процесса подстраивать веса всех нейронов сети. Тем самым векторы весов перемещаются ближе к области входных векторов. В процессе обучения коррекция весов начинает производиться лишь для ближайших к победителю нейронов Кохонена. При этом радиус коррекции постепенно уменьшается, так что в конце концов корректируется только вес нейрона-победителя Кохонена.

Еще один метод наделяет каждый нейрон Кохонена «чувством справедливости». Если он становится победителем чаще своей законной доли времени (примерно $1/K$, где K – число нейронов Кохонена), он временно увеличивает свой порог, что уменьшает его шансы на выигрыш, давая тем самым возможность обучаться и другим нейронам.

Во многих приложениях точность результата существенно зависит от распределения весов. К сожалению, эффективность различных решений исчерпывающим образом не оценена и остается проблемой.

До сих пор мы обсуждали алгоритм обучения, в котором для каждого входного вектора активировался лишь один нейрон Кохонена. Это называется методом аккредитации. Его точность ограничена, так как выход полностью является функцией лишь одного нейрона Кохонена.

В методе интерполяции целая группа нейронов Кохонена, имеющих наибольшие выходы, может передавать свои выходные сигналы в слой Гроссберга. Число нейронов в такой группе должно выбираться в зависимости от задачи, и убедительных данных относительно оптимального размера группы не имеется. Как только группа определена, ее множество выходов Y

рассматривается как вектор, длина которого нормализуется на единицу делением каждого значения *Y* на корень квадратный из суммы квадратов значений *Y* в группе. Все нейроны вне группы имеют нулевые выходы.

Метод интерполяции способен устанавливать более сложные соответствия и может давать более точные результаты. По-прежнему, однако, нет убедительных данных, позволяющих сравнить режимы интерполяции и аккредитации.

4.6. Контрольные вопросы и упражнения

1. Какими особенностями обладают самоорганизующие сети?

2. Какой тип алгоритмов обучения используется в сетях Кохонена: «с учителем» или «без учителя»?

3. Объясните суть проблемы мертвых нейронов.

4. Какие функции используются для определения топологической окрестности нейрона-победителя?

5. Сравните эффективность следующих алгоритмов обучения сетей Кохонена: алгоритма WTA, классического алгоритма Кохонена, алгоритма нейронного газа.

6. Для решения каких задач, могут быть использованы сети Кохонена?

7. Сравните характеристики сетей обратного распространения и сетей Кохонена. Какие преимущества дает сединение слоя Кохонена и слоя Гроссберга?

8. Исследуйте классифицирующие свойства сети Кохонена методом компьютерного моделирования для идентификации следующих типов законов распределения:

а) Равномерный закон распределения $F(x)=\begin{cases}0, & x<0\\ \dfrac{x}{b}, & 0\le x\le b\\ 1, & x>b\end{cases}$;

б) Экспоненциальный закон распределения $F(x)=\begin{cases}0, & x<0\\ 1-e^{\lambda x}, & x\ge 0\end{cases}$;

Для проведения исследований выполните следующие действия:

а) Создайте два набора данных: один – для обучения, второй – для тестирования;

б) Исследуйте, как изменение алгоритма обучения влияет на ошибку классификации (используйте алгоритм WTA и алгоритм нейронного газа);

в) Исследуйте, как объем входного вектора влияет на ошибку классификации;

г) Исследуйте, как изменение коэффициента скорости обучения на ошибку классификации.

5. Рекуррентные сети

5.1. Общие положения

Отдельную группу нейронных сетей составляют сети с обратной связью между различными слоями нейронов. Это так называемые *рекуррентные* сети. Их общая черта состоит в передаче сигналов с выходного либо скрытого слоя во входной слой.

Главная особенность таких сетей – динамическая зависимость на каждом этапе функционирования. Изменение состояния одного нейрона отражается на всей сети вследствие обратной связи типа «один ко многим». В сети возникает переходный процесс, который завершается формированием нового устойчивого состояния, отличающегося в общем случае от предыдущего.

Другой особенностью рекуррентных сетей является тот факт, что для них не подходит ни обучение с учителем, ни обучение без учителя. В таких сетях весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных, и все обучение сети сводится именно к этому расчету.

С одной стороны, предъявление априорной информации можно расценивать, как помощь учителя, но с другой – сеть фактически просто запоминает образцы до того, как на ее вход поступают реальные данные, и не

может изменять свое поведение, поэтому говорить о звене обратной связи с учителем не приходится.

Из сетей с подобной логикой работы наиболее известны сеть Хопфилда и сеть Хемминга, которые обычно используются для организации ассоциативной памяти. Ассоциативная память играет роль системы, определяющей взаимную зависимость векторов. В случае, когда на взаимозависимость исследуются компоненты одного и того же вектора, говорят об ассоциативной памяти. Если же взаимозависимыми оказываются два различных вектора, можно говорить о памяти гетероассоциативного типа. Типичным представителем первого класса является сеть Хопфилда, а второго – сеть Хемминга.

Главная задача ассоциативной памяти сводится к запоминанию входных обучающих выборок таким образом, чтобы при представлении новой выборки система могла сгенерировать ответ – какая из запомненных ранее выборок наиболее близка к вновь поступившему образу. Наиболее часто в качестве меры близости отдельных множеств применяется мера Хемминга.

При использовании двоичных значений $(0,1)$ расстояние Хемминга между двумя векторами $y=[y_1, y_2,...,y_N]^T$ и $d=[d_1, d_2,...,d_N]^T$ определяется в виде [4]:

$$d_H(y,d)=\sum_{i=1}^{N}[d_i(1-y_i)+(1-d_i)y_i]. \qquad (5.1)$$

При биполярных (± 1) значениях элементов обоих векторов расстояние Хемминга рассчитывается по формуле:

$$d_H(y,d)=\frac{1}{2}\left[N-\sum_{i=1}^{N}y_i d_i\right]. \qquad (5.2)$$

Мера Хемминга равна нулю только тогда, когда $y=d$. В противном случае она равна количеству битов, на которое различаются оба вектора.

5.2. Сеть Хопфилда

Обобщенная структура сети Хопфилда представляется, как правило, в виде системы с непосредственной обратной связью выхода с входом (рис. 5.1).

Характерная особенность такой системы состоит в том, что выходные сигналы нейронов являются одновременно входными сигналами сети:

$$x_i(t) = y_i(t-1) \qquad (5.3)$$

В классической системе Хопфилда отсутствует связь нейрона с собственным выходом, что соответствует $w_{ii} = 0$, а матрица весов является симметричной $W = W^T$.

Процесс обучения сети формирует зоны притяжения некоторых точек равновесия, соответствующих обучающим данным. При использовании ассоциативной памяти мы имеем дело с обучающим вектором *x* либо с множеством этих векторов, которые в результате проводимого обучения определяют расположение конкретных точек притяжения.

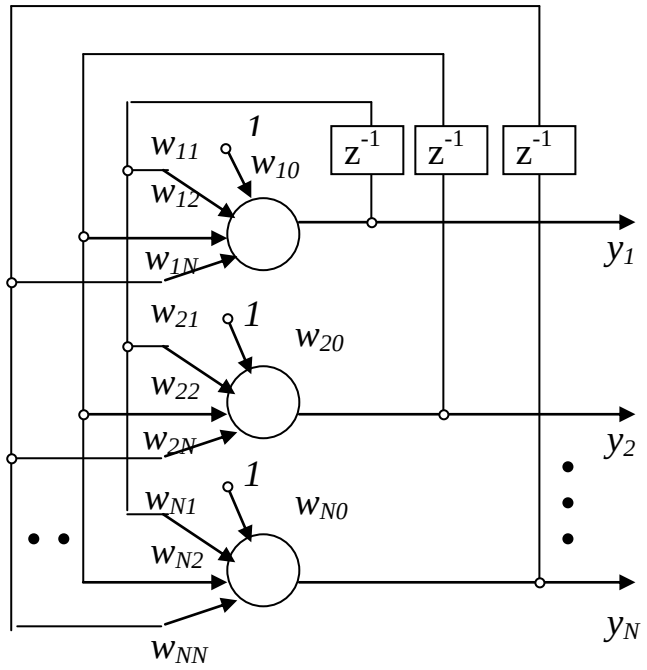


Рис. 5.1 Обощенная структура сети Хопфилда

Сеть Хопфилда состоит из единственного слоя нейронов, число которых является одновременно числом входов и выходов сети. Каждый нейрон связан синапсами со всеми остальными нейронами, а также имеет один входной синапс, через который осуществляется ввод сигнала. В качестве функции активации нейронов сети Хопфилда будем использовать знаковую функцию,

хотя для сетей Хопфилда также можно использовать пороговую функцию, линейную функцию с насыщением или сигмоидальные функции активации.

Это означает, что выходной сигнал $i-го$ нейрона определяется функцией:

$$y_i = \mathrm{sgn}\left(\sum_{j=1}^{N} w_{ij} x_j + b_i \right), \qquad (5.4)$$

где N обозначает число нейронов.

Будем считать, что пороговые элементы являются компонентами вектора x. Без учета единичных задержек сети, представляющих собой способ синхронизации процесса передачи сигналов, основные зависимости, определяющие сеть Хопфилда, можно представить в виде:

$$y_i(t) = \mathrm{sgn}\left(\sum_{j=0, i \neq j}^{N} w_{ij} y_j(t-1) \right), \qquad (5.5)$$

с начальным условием $y_i(0) = x_j$. В процессе функционирования сети Хопфилда можно выделить два режима: обучения и классификации. В режиме обучения на основе известных обучающих выборок x подбираются весовые коэффициенты w_{ij}. В режиме классификации при зафиксированных значениях весов и вводе конкретного начального состояния нейронов $y(0) = x$ возникает переходной процесс, протекающий в соответствии с выражением (5.5) и завершающийся в одном из локальных минимумов, для которого $y(t) = y(t-1)$.

При вводе только одной обучающей выборки x процесс изменений продолжается до тех пор, пока зависимость (5.5) не начнет соблюдаться для всех N нейронов. Это условие автоматически выполняется в случае выбора значений весов, соответствующих отношению

$$w_{ij} = \frac{1}{N} x_i x_j, \qquad (5.6)$$

поскольку только тогда $\frac{1}{N}\left(\sum_{j=1}^{N} x_i x_j x_j \right) = x_i$ (вследствие биполярных значений элементов вектора x всегда выполняется соотношение $x_j^2 = (\pm 1)^2 = 1$).

Следует отметить, что зависимость (5.6) представляет собой правило обучения Хебба. При вводе большого числа обучающих выборок $x(t)$ для $t = 1, 2, \ldots, p$ веса w_{ij} подбираются согласно обобщенному правилу Хебба, в соответствии с которым

$$w_{ij} = \frac{1}{N} \sum_{t=1}^{p} x_i^{(t)} x_j^{(t)}. \qquad (5.7)$$

Благодаря такому режиму обучения веса принимают значения, определяемые усреднением множества обучающих выборок.

В случае множества обучающих выборок становится актуальным фактор стабильности ассоциативной памяти. Для стабильного функционирования сети необходимо, чтобы реакция i-го нейрона $y_i^{(l)}$ на l-ю обучающую выборку $x^{(l)}$ совпадала с ее i-й составляющей $x_i^{(l)}$. Это означает, что с учетом выражения (5.7) получим

$$y_i^{(l)} = \operatorname{sgn}\left(\sum_{j=0}^{N} w_{ij} x_j^{(l)} \right) = \operatorname{sgn}\left(\frac{1}{N} \sum_{j=0}^{N} \sum_{t=1}^{p} x_i^{(t)} x_j^{(t)} x_j^{(l)} \right) = x_i^{(l)}. \quad (5.8)$$

Если взвешенную сумму входных сигналов i-го нейрона обозначить $u_i^{(l)}$, то можно выделить в ней ожидаемое значение $x_i^{(l)}$ и остаток, называемый *диафонией* [4]:

$$u_i^{(l)} = x_i^{(l)} + \frac{1}{N} \sum_{j=0}^{N} \sum_{t \neq l} x_i^{(t)} x_j^{(t)} x_j^{(l)}. \qquad (5.9)$$

Вследствие применения знаковой функции активации, выполнение условия (5.8) возможно при малых значениях диафонии, не способных изменить знак $x_i^{(l)}$. Это означает, что, несмотря на определенное несовпадение битов, переходный процесс завершается в нужной точке притяжения. При предоставлении тестовой выборки, отличающейся некоторым количеством битов, нейронная сеть может откорректировать эти биты и завершить процесс классификации в нужной точке притяжения.

Тем не менее правило Хебба обладает невысокой продуктивностью. Максимальная емкость ассоциативной памяти (число запомненных образцов) при обучении по правилу Хебба с допустимой погрешностью 1%, составляет примерно 14% от числа нейронов сети [4]. Кроме того, при наличии шума, применение правила Хебба приводит к различным неточностям в виде локальных минимумов, далеких от исходного решения. Поэтому в качестве альтернативы используют методы обучения, основанные на псевдоинверсии.

Идея этого метода состоит в том, что при правильно подобранных весах, каждая поданная на вход выборка x генерирует на выходе саму себя, мгновенно приводя к исходному состоянию (зависимость (5.5)) [4]. В матричной форме это можно представить в виде:

$$WX = X, \tag{5.10}$$

где W - матрица весов сети размерностью $N \times N$, а X - прямоугольная матрица размерностью $N \times p$, составленная из p последовательных обучающих векторов $x^{(t)}$, то есть $X = \left\lfloor x^{(1)}, x^{(2)}, \ldots, x^{(p)} \right\rfloor$. Решение такой линейной системы уравнений имеет вид:

$$W = X\ X^{+}, \tag{5.11}$$

где знак + обозначает псевдоинверсию. Если обучающие векторы линейно независимы, последнее выражение можно представить в форме:

$$W = X\left(X^{T}\ X\right)^{-1} X^{T}. \tag{5.12}$$

Псевдоинверсия матрицы размерностью $N \times p$ в этом выражении заменена обычной инверсией квадратной матрицы $X^{T}\ X$ размерностью $p \times p$. Дополнительное достоинство выражения (5.12) – возможность записать его в итерационной форме, не требующей расчета обратной матрицы. В этом случае выражение (5.12) принимает вид функциональной зависимости от последовательности обучающих векторов $x^{(t)}$ для $t = 1, 2, \ldots, p$:

$$W^{(t)} = W^{(t-1)} + \frac{1}{\left[x^{(t)}\right]^{T} x^{(t)} - \left[x^{(t)}\right]^{T} W^{(t-1)} x^{(t)}} \times \left[W^{(t-1)} x^{(t)} - x^{(t)}\right] \times \left[W^{(t-1)} x^{(t)} - x^{(t)}\right]^{T}$$

(5.13)

при начальных условиях $W^{(0)} = 0$. Такая форма предполагает однократное предъявление всех обучающих выборок, в результате чего матрица весов принимает значение $W = W^{(p)}$. Зависимости (5.12) и (5.13) называются *методом проекций.* Применение метода псевдоинверсии увеличивает максимальную емкость сети Хопфилда, которая становится равной $N - 1$.

Модифицированный вариант метода проекций – метод $\Delta -$проекций – это градиентная форма алгоритма минимизации целевой функции. В соответствии с этим способом веса подбираются рекуррентно с помощью циклической процедуры, повторяемой для всех обучающих выборок:

$$W \leftarrow W + \frac{\eta}{N}\left[x^{(t)} - W x^{(t)}\right]\left[x^{(t)}\right]^T. \qquad (5.14)$$

Коэффициент η - это коэффициент обучения, выбираемый обычно из интервала [0.7 – 0.9]. Процесс обучения завершается, когда изменение вектора весов становится меньше априорно принятого значения ε.

По завершении подбора весов сети их значения «замораживаются», и сеть можно использовать в режиме распознавания. В этой фазе на вход сети подается тестовый вектор x и рассчитывается ее отклик в виде:

$$y(t) = \mathrm{sgn}(W y(t-1)) \qquad (5.15)$$

(в начальный момент $y(0) = x$), причем итерационный процесс повторяется для последовательных значений $y(t)$ вплоть до стабилизации отклика.

В процессе распознавания образа по зашумленным сигналам, образующим начальное состояние нейронов, возникают проблемы с определением конечного состояния, соответствующего одному из запомненных образов. Возможны ошибочные решения. Одной из причин нахождения ошибочных решений является возможность перемешивания различных компонентов запомненных образов и формирования стабильного состояния, воспринимаемого как локальный минимум.

5.3. Сеть Хемминга

Сеть Хемминга – это трехслойная рекуррентная структура, которую можно считать развитием сети Хопфилда, была предложена Р. Липпманом [6]. Она позиционируется как специализированное гетероассоциативное запоминающее устройство. Основная идея функционирования сети состоит в минимизации расстояния Хемминга между тестовым вектором, подаваемым на вход сети, и векторами обучающих выборок, закодированными в структуре сети. Обобщенная структура сети Хемминга представлена на рис. 5.2. [4].

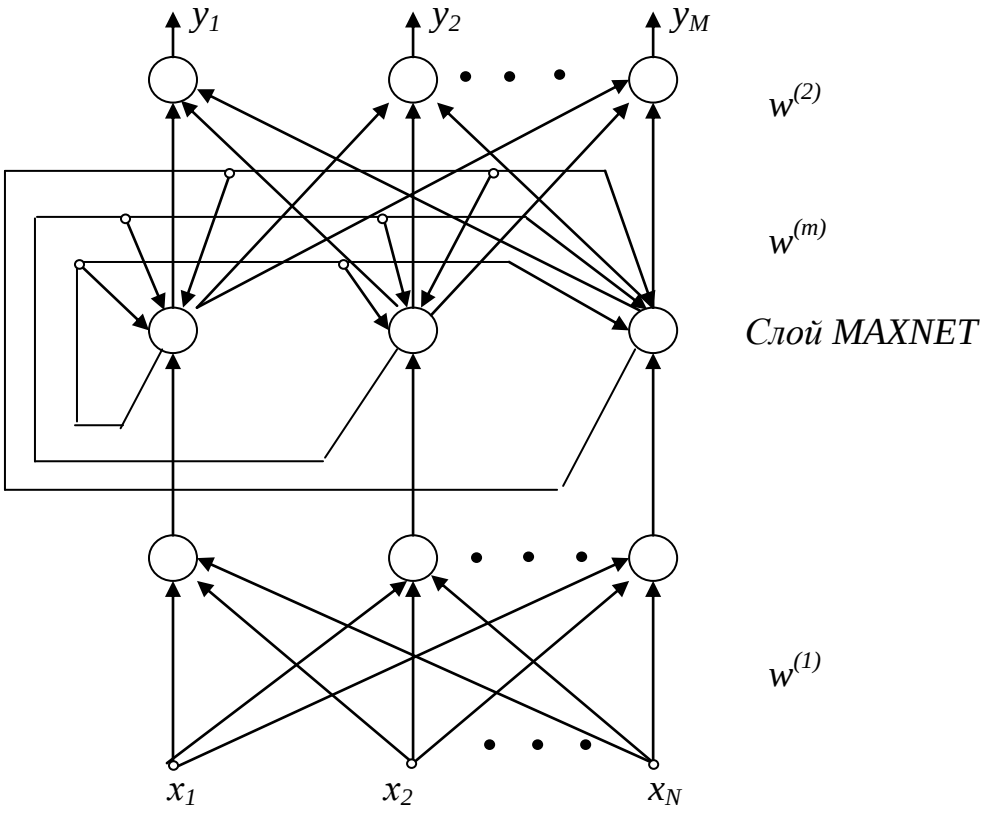


Рис. 5.2 Обобщенная структура сети Хемминга

Первый ее слой имеет однонаправленное распространение сигналов от входа к выходу и фиксированные значения весов. Второй слой, MAXNET, состоит из нейронов, связанных обратными связями по принципу «каждый с каждым», при этом в отличие от структуры Хопфилда существует ненулевая связь входа нейрона со своим собственным выходом. Веса нейронов в слое MAXNET постоянны. Разные нейроны связаны отрицательной обратной связью

с весом $-\varepsilon$, при этом обычно величина ε обратно пропорциональна числу образов.

С собственным выходом нейрон связан положительной обратной связью с весом +1. Веса пороговых элементов равны нулю. Нейроны этого слоя функционируют в режиме WTA, при котором в каждой фиксированной ситуации активизируется только один нейрон. Выходной однонаправленный слой формирует выходной вектор, соответствующий входному вектору. Веса нейронов этого слоя подбираются в зависимости от входных обучающих выборок.

В процессе функционирования сети можно выделить три фазы. В первой из них на вход подается *N*-элементный вектор *x*. После предъявления этого вектора на выходах нейронов первого слоя генерируются сигналы, задающие начальные состояния нейронов второго слоя.

Во второй фазе инициировавшие MAXNET сигналы удаляются, и из сформированного ими начального состояния запускается итерационный процесс. Итерационный процесс завершается в момент, когда все нейроны, кроме нейрона-победителя с выходным сигналом равным 1, перейдут в нулевое состояние. Нейрон-победитель становится представителем класса данных, к которому принадлежит входной вектор.

В третьей фазе этот нейрон посредством весов, связывающих его с нейронами выходного слоя, формирует на выходе сети отклик в виде вектора *y*, соответствующего возбуждающему вектору *x*, где *x* и *y* биполярные векторы сети со значениями элементов ± 1. Входные узлы сети принимают значения, задаваемые аналогичными компонентами вектора *x*. Нейроны первого слоя рассчитывают расстояние Хемминга между входным вектором *x* и каждым из p закодированных векторов-образцов $x^{(t)}$, образующих веса нейронов первого слоя. Нейроны в слое MAXNET выбирают вектор с наименьшим расстоянием Хемминга, определяя, таким образом, класс, к которому принадлежит предъявленный входной вектор *x*. Веса нейронов выходного слоя формируют вектор, соответствующий предъявленному входному вектору. При p нейронах

первого слоя, емкость запоминающего устройства Хемминга также равна p, так как каждый нейрон представляет единственный класс.

Веса первого слоя соответствуют очередным векторам-образцам $x^{(t)}$, поэтому $w_{ij}^{(1)} = x_j^{(t)}$ для $t = 1, 2, \ldots, p$. Аналогично веса выходного слоя соответствуют очередным векторам образов $y^{(t)}$, $w_{li}^{(2)} = y_l^{(t)}$.

В случае нейронов слоя MAXNET, функционирующих в режиме WTA, веса сети должны усиливать собственный сигнал нейрона и ослаблять остальные сигналы. Для достижения этого принимается $w_{ii}^{(m)} = 1$, а также

$$-\frac{1}{p-1} < w_{is}^{(m)} < 0 \qquad (5.16)$$

для $i \neq s$. Для обеспечения абсолютной сходимости алгоритма веса $w_{is}^{(m)}$ должны отличать друг от друга. Р. Липпман в своей работе принял

$$w_{is}^{(m)} = -\frac{1}{p-1} + \xi, \qquad (5.17)$$

где ξ - случайная величина с достаточно малой амплитудой.

Нейроны различных слоев сети Хемминга функционируют по-разному. Нейроны первого слоя рассчитывают расстояния Хемминга между поданными на вход сети вектором x и векторами весов $w^{(t)} = x^{(t)}$ отдельных нейронов этого слоя. Значения выходных сигналов этих нейронов определяются по формуле:

$$\hat{y}_i = 1 - \frac{d_H\left(x^{(t)}, x\right)}{N}, \qquad (5.18)$$

где $d_H\left(x^{(t)}, x\right)$ обозначает расстояние Хемминга между входными векторами $x^{(t)}$ и x, то есть число битов, на которое различаются эти два вектора. Значение $\hat{y}_i = 1$, если $x = x^{(t)}$, и $\hat{y}_i = 0$, если $x = -x^{(t)}$. В остальных случаях значения $\hat{y}_i$ лежат в интервале [0, 1].

Сигналы $\hat{y}_i$ нейронов первого слоя становятся начальными состояниями нейронов слоя MAXNET на второй фазе функционирования сети. Задача нейронов этого слоя состоит в определении победителя, то есть нейрона, у

которого выходной сигнал наиболее близок к 1. Процесс определения победителя выполняется согласно формуле:

$$y_l(t) = f\left(\sum_s w_{is}^{(m)} y_s(t-1)\right) = f\left(y_l(t-1) + \sum_{s \neq i} w_{is}^{(m)} y_s(t-1)\right), \qquad (5.19)$$

при начальном значении $y_s(0) = \hat{y}_i$. Функция активации $f(y)$ нейронов слоя MAXNET задается выражением:

$$f(y) = \begin{cases} y \text{ для } y \geq 0 \\ 0 \text{ для } y < 0 \end{cases}. \qquad (5.20)$$

Итерационный процесс (5.19) завершается в момент, когда состояние нейронов стабилизируется, и активность продолжает проявлять только один нейрон, тогда как остальные пребывают в нулевом состоянии. Активный нейрон становится победителем и через веса $w_{is}^{(2)}$ линейных нейронов выходного слоя представляет вектор $y^{(t)}$, который соответствует вектору $x^{(t)}$, признанному слоем MAXNET в качестве ближайшего к входному вектору x.

Важным достоинством сети Хемминга считается небольшое, по сравнению с сетью Хопфилда, число взвешенных связей между нейронами. Так, например, 100-входовая сеть Хопфилда, кодирующая 10 различных векторных классов, должна содержать 10000 взвешенных связей, тогда как аналогичная сеть Хемминга содержит 1100 связей, из которых 1000 весов находятся в первом слое, а 100 – в слое MAXNET [4].

5.4. Рекуррентная сеть Эльмана

Данная рекуррентная сеть представляет собой развитие сетей персептронного типа за счет добавления в них обратных связей. Сеть Эльмана характеризуется частичной рекуррентностью в форме обратной связи между входным и скрытым слоем, реализуемой с помощью единичных элементов запаздывания z^{-1}. Обобщенная структура этой сети представлена на рис. 5.3.

Каждый скрытый нейрон имеет свой аналог в контекстном слое, образующем совместно с внешними входами сети входной слой. Выходной слой состоит из нейронов, однонаправлено связанных только с нейронами

скрытого слоя. Обозначим внутренний вектор возбуждения сети x (в его состав входит пороговый элемент), состояния скрытых нейронов - $\nu \in R^K$, а выходные сигналы сети - $y \in R^M$. Тогда входной вектор сети в момент времени t имеет форму:

$$x(t) = \left[x_0(t), x_1(t), \ldots, x_N(t), \nu_1(t-1), \nu_2(t-1), \ldots, \nu_K(t-1)\right] \quad (5.21)$$

Веса синаптических связей первого (скрытого) слоя сети обозначим $w_{ij}^{(1)}$, а второго (выходного) слоя $w_{si}^{(2)}$. Если взвешенную сумму *i*-го нейрона скрытого слоя обозначить u_i, а его выходной сигнал - ν_i, то

$$u_i(t) = \sum_{j=0}^{N+K} w_{ij}^{(1)} x_j(t); \quad (5.22)$$

$$\nu_i(t) = f_1(u_i(t)); \quad (5.23)$$

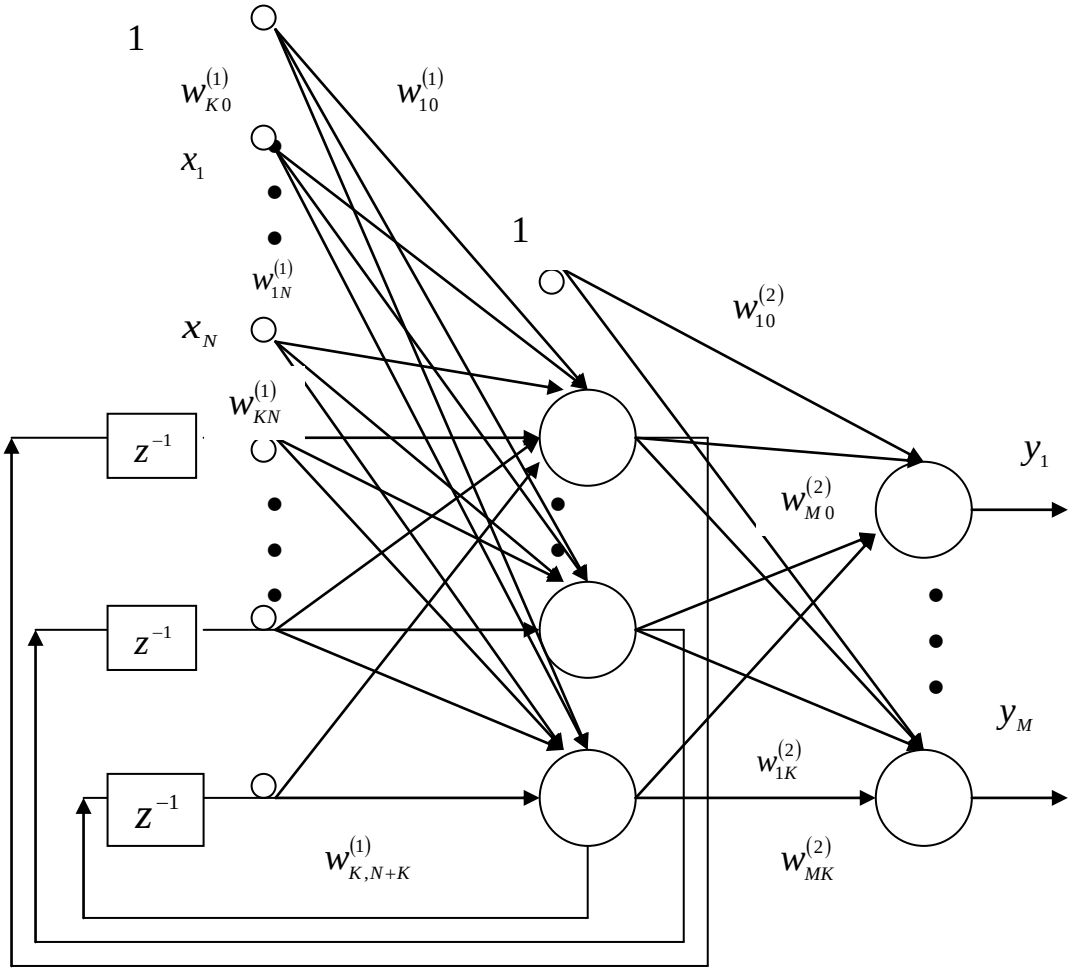


Рис. 5.3 Структура сети Эльмана

Веса $w_{ij}^{(1)}$ образуют матрицу $W^{(1)}$ синаптических связей скрытого слоя, а $f_1(u_i)$ - функция активации i-го нейрона скрытого слоя. Аналогично можно обозначить взвешенную сумму s-го нейрона выходного слоя g_s, а соответствующий ему выходной сигнал сети - y_s. Эти сигналы описываются формулами:

$$g_s(t) = \sum_{i=0}^{K} w_{si}^{(2)} v_s(t); \qquad (5.24)$$

$$y_s(t) = f_2(g_s(t)). \qquad (5.25)$$

В свою очередь, веса $w_{si}^{(2)}$ образуют матрицу $W^{(2)}$, описывающую синаптические связи нейронов выходного слоя, а $f_2(u_s)$ - функция активации s-го нейрона выходного слоя.

5.5. Алгоритм обучения рекуррентной сети Эльмана

Для обучения сети Эльмана будем использовать градиентный метод наискорейшего спуска.

Для этого метода необходимо задать формулы, позволяющие рассчитывать градиент целевой функции в текущий момент времени. Целевая функция в момент времени t определяется как сумма квадратов разностей между значениями выходных сигналов сети и их ожидаемыми значениями для всех M выходных нейронов:

$$E(k) = \frac{1}{2}\sum_{s=1}^{M}\left[y_s(t) - d_s(t)\right]^2 = \frac{1}{2}\sum_{s=1}^{M} e_s(t)^2 \qquad (5.26)$$

При дифференцировании целевой функции относительно весов выходного слоя получаем:

$$\nabla_{\alpha\beta}^{(2)} E(t) = \frac{\partial E(t)}{\partial w_{\alpha\beta}^{(2)}} = \sum_{s=1}^{M} e_s(t) \frac{df_2(g_s(t))}{dg_s(t)} \frac{dg_s(t)}{dw_{\alpha\beta}^{(2)}} = \sum_{s=1}^{M} e_s(t) \frac{df_2(g_s(t))}{dg_s(t)} \sum_{i=0}^{K} \frac{d\left(w_{si}^{(2)} v_i(t)\right)}{dw_{\alpha\beta}^{(2)}} =$$

$$\sum_{s=1}^{M} e_s(t) \frac{df_2(g_s(t))}{dg_s(t)} \sum_{i=0}^{K}\left(\frac{dv_i}{dw_{\alpha\beta}^{(2)}} w_{si}^{(2)} + \frac{dw_{si}^{(2)}}{dw_{\alpha\beta}^{(2)}} v_i(t) \right) \qquad (5.27)$$

Связи между скрытым и выходным слоем однонаправленные, поэтому:

$$\frac{d\nu_i}{dw_{\alpha\beta}^{(2)}} = 0 \qquad (5.28)$$

С учетом этого факта получим:

$$\nabla_{\alpha\beta}^{(2)} E(t) = \sum_{s=1}^{M} e_s(t) \frac{df_2(g_s(t))}{dg_s(t)} \sum_{i=0}^{K} \frac{dw_{si}^{(2)}}{dw_{\alpha\beta}^{(2)}} \nu_i(t) = e_\alpha(t) \frac{df_2(g_\alpha(t))}{dg_\alpha(t)} \nu_\beta(t) \qquad (5.29)$$

При использовании метода наискорейшего спуска веса $w_{\alpha\beta}^{(2)}$ уточняются по формуле:

$$w_{\alpha\beta}^{(2)}(t+1) = w_{\alpha\beta}^{(2)}(t) + \Delta w_{\alpha\beta}^{(2)}, \qquad (5.30)$$

$$\text{где } \Delta w_{\alpha\beta}^{(2)} = -\eta \nabla_{\alpha\beta}^{(2)} E(t) \qquad (5.31)$$

Формулы уточнения весов скрытого слоя сети Эльмана более сложные по сравнению с персептронной сетью из-за наличия обратных связей между скрытым и контекстным слоями. Расчет компонентов вектора градиента целевой функции относительно весов скрытого слоя реализуется по формулам:

$$\nabla_{\alpha\beta}^{(1)} E(t) = \sum_{s=1}^{M} e_s(t) \frac{df_2(g_s(t))}{dg_s(t)} \sum_{i=1}^{K} \frac{d(\nu_i w_{si}^{(2)})}{dw_{\alpha\beta}^{(1)}} = \sum_{s=1}^{M} e_s(t) \frac{df_2(g_s(t))}{dg_s(t)} \sum_{i=1}^{K} \frac{d\nu_i(t)}{dw_{\alpha\beta}^{(1)}} w_{si}^{(2)} \qquad (5.32)$$

$$\frac{d\nu_i(t)}{dw_{\alpha\beta}^{(1)}} = \frac{df_1(u_i)}{du_i} \sum_{k=0}^{N+K} \frac{d(x_k(t) w_{ik}^{(1)})}{dw_{\alpha\beta}^{(1)}} = \frac{df_1(u_i)}{du_i} \left[\delta_{i\alpha} x_\beta + \sum_{k=N+1}^{N+K} \frac{dx_k(t)}{dw_{\alpha\beta}^{(1)}} w_{ik}^{(1)} \right] \qquad (5.33)$$

Из определения входного вектора x (формула (5.21)) в момент времени t следует выражение (5.34):

$$\frac{d\nu_i(t)}{dw_{\alpha\beta}^{(1)}} = \frac{df_1(u_i)}{du_i} \left[\delta_{i\alpha} x_\beta + \sum_{k=N+1}^{N+K} \frac{d\nu_k(t-1)}{dw_{\alpha\beta}^{(1)}} w_{ik}^{(1)} \right] = \frac{df_1(u_i)}{du_i} \left[\delta_{i\alpha} x_\beta + \sum_{k=1}^{K} \frac{d\nu_k(t-1)}{dw_{\alpha\beta}^{(1)}} w_{i,k+N}^{(1)} \right]. \qquad (5.34)$$

Это выражение позволяет рассчитать производные целевой функции относительно весов скрытого слоя в момент времени t. Следует отметить, что это рекуррентная формула, определяющая производную в момент времени t в

зависимости от ее значения в предыдущий момент $t-1$. Начальные значения производных в момент $t=0$ считаются нулевыми:

$$\frac{d\nu_1(0)}{dw_{\alpha\beta}^{(1)}} = \frac{d\nu_2(0)}{dw_{\alpha\beta}^{(1)}} = \ldots = \frac{d\nu_K(0)}{dw_{\alpha\beta}^{(1)}} = 0. \qquad (5.35)$$

Таким образом, алгоритм обучения сети Эльмана можно представить в следующем виде:

1. Присвоить весам случайные начальные значения, имеющие, как правило, равномерное распределение в определенном интервале (например, между -1 и 1).
2. Для очередного момента t $(t=0,\ 1,\ 2,\ldots)$ определить состояние всех нейронов сети (сигналы ν_l и y_l). На этой основе можно сформировать входной вектор $x(t)$ для произвольного момента t.
3. Определить вектор погрешности обучения $e(t)$ для нейронов выходного слоя как разность между фактическим и ожидаемым значениями сигналов выходных нейронов.
4. Сформировать вектор градиента целевой функции относительно весов выходного и скрытого слоя с использованием формул (5.29), (5.32) и (5.34).
5. Уточнить значения весов сети согласно правилам метода наискорейшего спуска:
   - для нейронов выходного слоя сети по формуле
   $$w_{\alpha\beta}^{(2)}(t) = w_{\alpha\beta}^{(2)}(t-1) - \eta\nabla_{\alpha\beta}^{(2)}E(t) \qquad (5.36)$$
   - для нейронов скрытого слоя сети по формуле
   $$w_{\alpha\beta}^{(1)}(t) = w_{\alpha\beta}^{(1)}(t-1) - \eta\nabla_{\alpha\beta}^{(1)}E(t) \qquad (5.37)$$

После уточнения значений весов перейти к пункту 2 алгоритма для расчета в очередной момент времени t.

Практические реализации алгоритма обучения сети Эльмана строятся на методе наискорейшего спуска, усиленном моментом. Это значительно

повышает эффективность обучения и вероятность достижения глобального минимума целевой функции. При использовании такого подхода уточнение вектора весов в момент времени t выполняется в соответствии с формулой:

$$\Delta w(t) = -\eta \nabla E(t) + \alpha(t) \Delta w(t-1), \qquad (5.38)$$

где $\alpha(t)$ - коэффициент момента, выбираемый из интервала (0, 1). Первое слагаемое этого выражения соответствует обычному методу обучения, второе – учитывает фактор момента, отражающий последнее изменение весов и не зависящий от фактического значения градиента. Чем больше величина α, тем большее влияние на подбор весов оказывает слагаемое момента. Его значение существенно возрастает на плоских участках целевой функции и около локального минимума, где значение градиента близко к нулю.

В окрестностях локального минимума фактор момента может вызвать изменение весов, ведущее к росту целевой функции и к выходу из зоны локального минимума с возобновлением поиска глобального минимума.

5.6. Контрольные вопросы и упражнения

1. Что такое рекуррентные сети?

2. Объясните, в чем различия сетей Хопфилда и Хемминга?

3. Объясните, чем сеть Эльмана принципиально отличается от сетей Хопфилда и Хемминга?

4. Объясните, почему метод псевдоинверсии эффективнее правила Хебба?

5. Как рассчитать расстояние Хемминга между векторами?

6. Что такое ассоциативная и гетероассоциативная память?

7. Для решения, каких задач, могут быть использованы рекуррентные сети?

8. Исследуйте классифицирующие свойства сети Хемминга методом компьютерного моделирования для идентификации следующих типов законов распределения:

а) Равномерный закон распределения $F(x) = \begin{cases} 0, & x < 0 \\ \dfrac{x}{b}, & 0 \le x \le b \\ 1, & x > b \end{cases}$;

б) Экспоненциальный закон распределения $F(x) = \begin{cases} 0, & x < 0 \\ 1 - e^{\lambda x}, & x \geq 0 \end{cases}$;

Для проведения исследований выполните следующие действия:

а) Создайте два набора данных: один – для обучения, второй – для тестирования;

в) Исследуйте, как объем входного вектора влияет на ошибку классификации.

9. Методом компьютерного моделирования исследуйте возможности сети Эльмана для построения прогноза следующих отображений:

а) $f(x) = \frac{1}{x}, 1 <= x <= 100$;

б) $f(x) = \lg_{10}(x), 1 <= x <= 10$;

в) $f(x) = \exp(-x), 1 <= x <= 10$;

г) $f(x) = \sin(x), 1 <= x <= \pi / 2$.

Для проведения исследований выполните следующие действия:

а) Создайте два набора данных: один – для обучения, второй – для тестирования;

в) Исследуйте, как изменение числа нейронов в скрытом слое влияет на ошибку прогнозирования;

г) Исследуйте, как изменение коэффициента обучения и коэффициента момента влияют на ошибку прогнозирования;

д) Исследуйте, как объем обучающей выборки влияет на ошибку прогнозирования.

6. Применение нейронных сетей для решения задач

6.1. Пример использования многослойного персептрона для решения задачи прогнозирования

В качестве примера приведем исследование зависимости качества прогноза MLP-сети от параметров алгоритма обратного распространения ошибки [13].

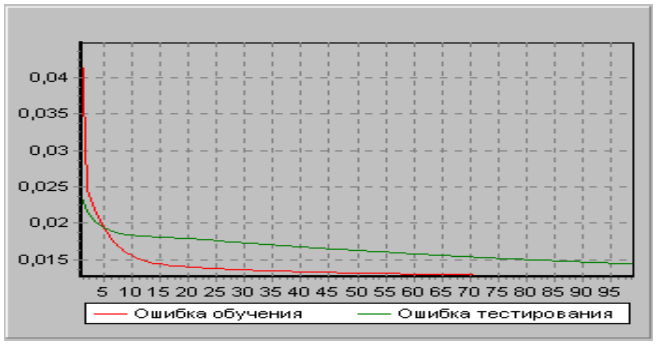
Для проведения исследований использован временной ряд температуры воздуха из 320 значений. При обучении нейронной сети значения отсчетов разбивались на обучающее, тестовое и контрольное множества следующим образом: обучающее множество с 0 до 200 отсчета; тестирующее множество с 200 до 250 отсчета; контрольное множество с 250 до 290 отсчета.

Была исследована зависимость качества обучения нейронной сети от параметров алгоритма обучения: η (коэффициента скорости обучения) и α (коэффициента момента обучения). В ходе исследований максимальное количество итераций было принято равным 100, а достижимая ошибка обучения – равной 0,001. Результаты исследований приведены в табл. 6.1 и представлены на рис. 6.1 – 6.3.

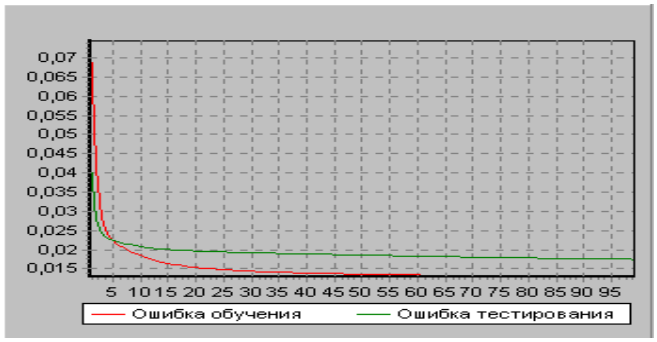
Таблица 6.1

Зависимость итоговых статистик обучения MLP-сети от параметров алгоритма обучения

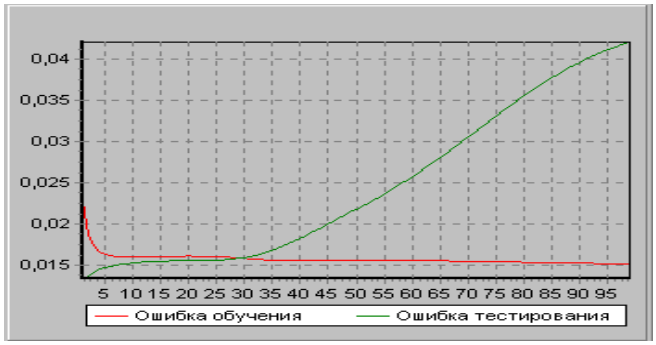
Итоговые статистики	Параметры метода обучения			
	$\eta = 0{,}25$ $\alpha = 0{,}9$	$\eta = 0{,}1$ $\alpha = 0{,}9$	$\eta = 0{,}25$ $\alpha = 0{,}5$	$\eta = 0{,}5$ $\alpha = 0{,}9$
Математическое ожидание ошибки обучения	0,01412	0,01543	0,01570	0,01379
Среднеквадратическое отклонение ошибки обучения	0,02656	0,02817	0,03878	0,02408
Математическое ожидание ошибки тестирования	0,01639	0,01900	0,02432	0,01492
Среднеквадратическое отклонение ошибки тестирования	0,00232	0,00341	0,00962	0,00159
Математическое ожидание ошибки контрольного тестирования	0,02686	0,02579	0,13362	0,03056
СКО ошибки контрольного тестирования	0,03688	0,03500	0,14359	0,03898



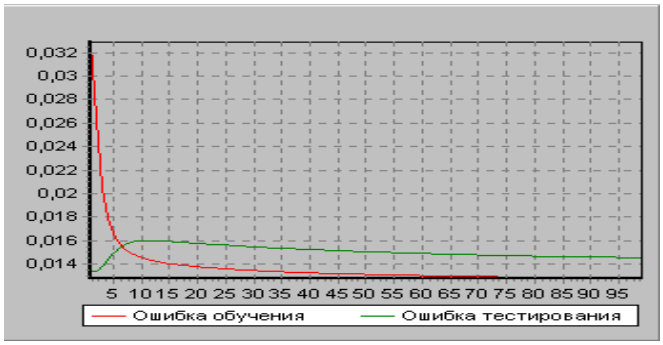
η = 0,25 α = 0,9



η = 0,1 α = 0,9



η = 0,25 α =0,5



η=0,5 α =0,9

Рис. 6.1 Ошибки обучения и тестирования

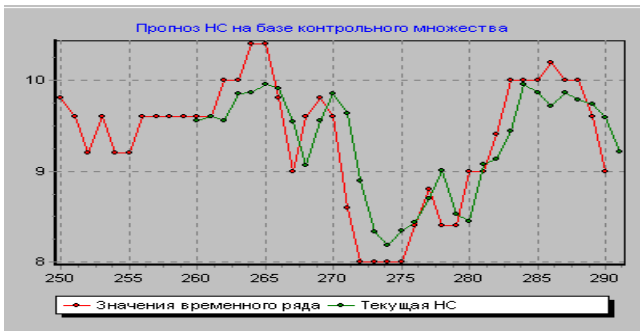
η = 0,25 α = 0,9

η = 0,1 α = 0,9

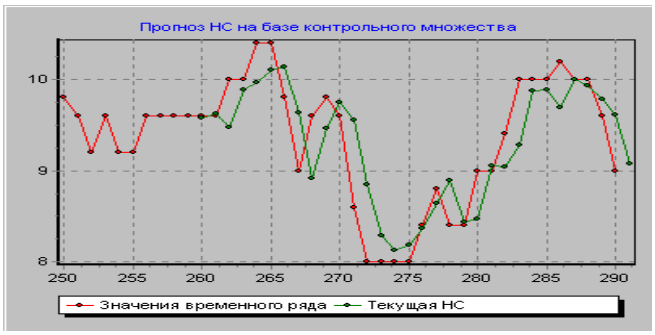
η = 0,25 α=0,5

η=0,5 α = 0,9

Рис.6.2 Ошибки контрольного тестирования



η = 0,25 α = 0,9



η = 0,1 α = 0,9

η = 0,25 α=0,5

η=0,5 α= 0,9

Рис. 6.3 Зависимость прогноза на контрольном множестве от параметров алгоритма обучения

Сравнение значений итоговых статистик проводилось по следующим критериям:

сравнение статистик контрольного тестирования;

сравнение статистик, связанных с ошибками тестирования и обучения.

Чем меньше математическое ожидание и среднеквадратическое отклонение ошибки, тем лучше параметры обучения, при которых достигаются данные показатели.

В результате сравнения значений итоговых статистик обучения, в качестве оптимальных параметров алгоритма обучения обратного распространения ошибки, были выбраны следующие значения: η = 0,25, α = 0,9.

6.2. Пример исползования радиально-базисной сети для аппроксимации функций

Исследование аппроксимативных возможностей радиально-базисной нейронной сети покажем на примере восстановления значений рациональной функции одного переменного в соответствии с методикой, изложенной в [15].

В качестве теста будем использовать функцию $f(x) = e^{-0,5x}$, заданную в табличной форме с равномерным распределением своих значений в точках из диапазона [0, 9,9] с шагом 0,1. Для тестирования используется та же функция, заданная на интервале [0, 0,09] с шагом 0,01.

Эксперимент проведен для сети HRBF, у которой нейроны скрытого слоя могут иметь различные функции активации:

1. Функция Гаусса $\exp(\frac{(x-\alpha_1)^2}{2\alpha_2^2})$;
2. Мульти-квадратичная функция $(x^2+\alpha_1^2)^{0.5}$.

Зафиксируем также удовлетворительный минимум функционала качества обучения на уровне 0,1. Длительность обучения «нейрона» примем равной 50 итерациям. Были проведены эксперименты для объемов обучающей выборки 25, 50 и 100 примеров. Результаты исследований приведены на рис. 6.4 -6.7.

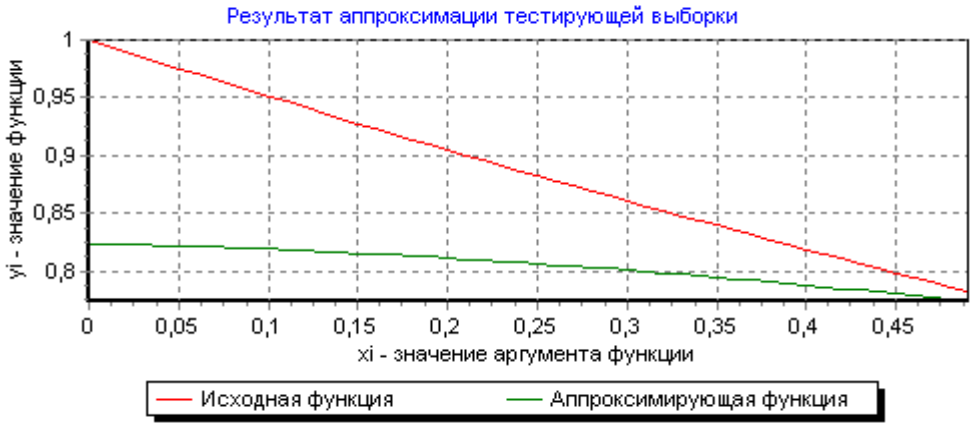


Рис. 6.4 Результат аппроксимации тестовых данных на 25 примерах

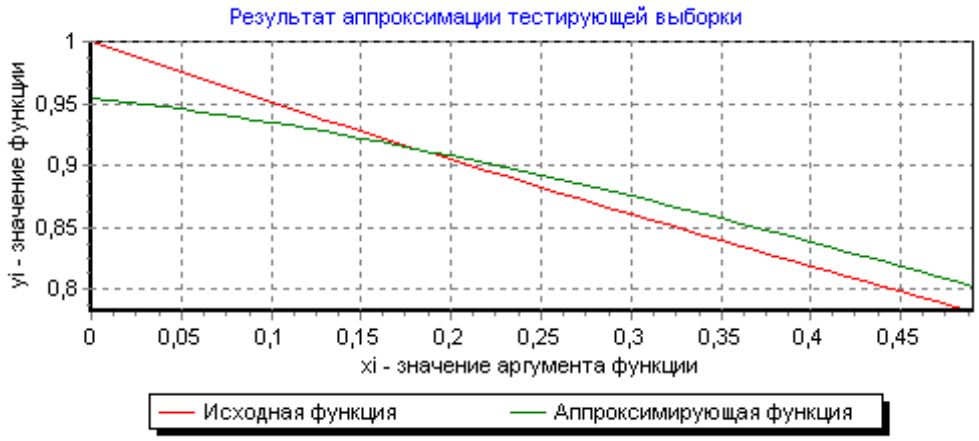


Рис. 6.5 Результат аппроксимации тестовых данных на 50 примерах

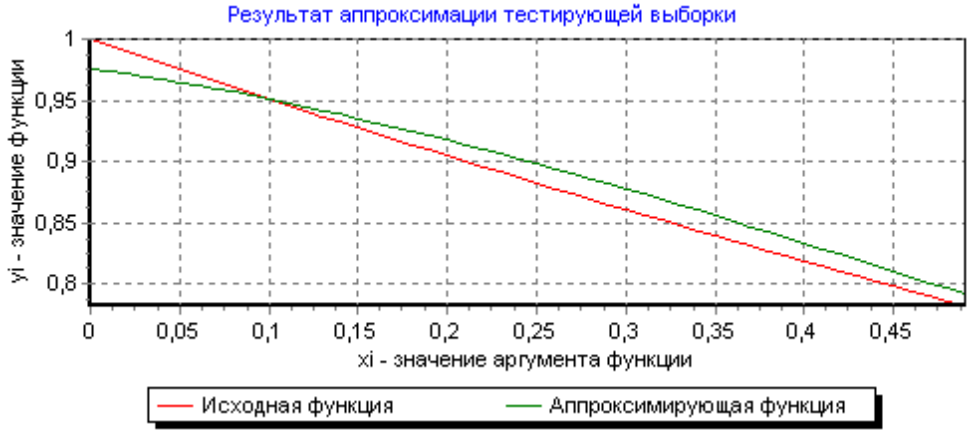


Рис. 6.6 Результат аппроксимации тестовых данных на 100 примерах

Качество аппроксимации растет при увеличении объема выборки, однако эта зависимость имеет нелинейный характер (рис. 6.7).

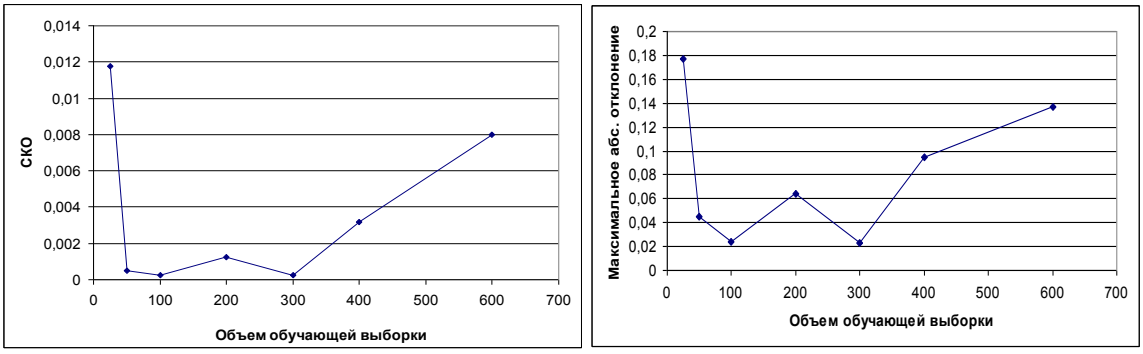


Рис. 6.7 Зависимость СКО и максимального абсолютного отклонения от объема выборки.

Для установления более точной зависимости между объемом выборки и качеством аппроксимации примем меру сложности сети: число весов в конфигурации сети. Значения сложности сети приведены в табл. 6.2.

Таблица 6.2

Значения сложности конфигураций сети

Объем выборки	25	50	100	200	300	400	600
Сложность сети	4	6	6	4	6	4	4
Отношение объема выборки к сложности сети	6,25	8,33	16,67	50	50	100	150

Исходя из данных табл. 6.2, сети разделены на две группы и для каждой группы построены графики зависимости числовых характеристик аппроксимации тестовой выборки от объема выборки (рис. 6.8).

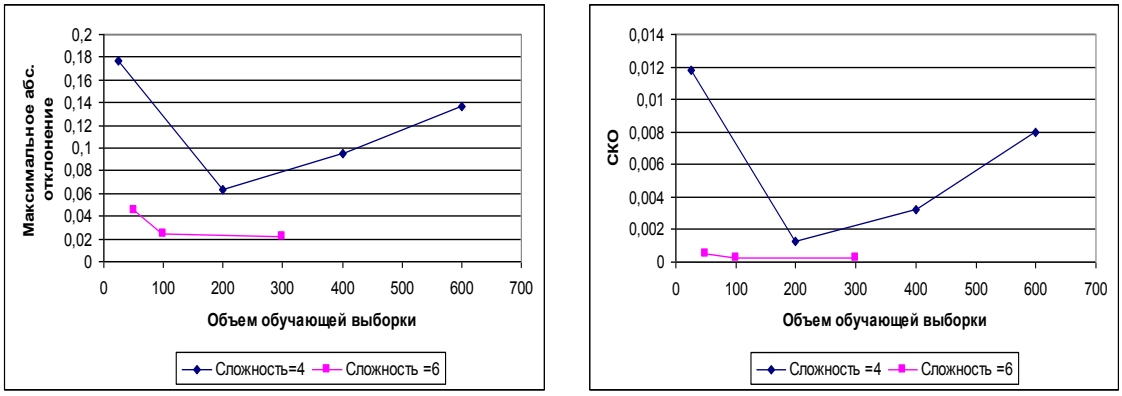


Рис. 6.8 Зависимость СКО и максимального абсолютного отклонения от объема выборки, соотнесенного со сложностью сети

6.3. Пример использования сети обратного распространения для решения задачи идентификации

При заданном числе отсчетов временного ряда, генерировали случайный процесс методом обратных функций, и затем строится функция распределения на интервале [0,M] с шагом 1. По полученной функции распределения

производится идентификация. Были проведены исследования зависимости качества идентификации от объема тестирующей выборки V[14]. В ходе исследований идентифицировались типовые законы распределения, перечисленные в табл. 6.3.

Результаты исследований показали, что чем больше отсчетов случайного процесса сгенерировано, тем он ближе будет к стандартному, следовательно, качество идентификации улучшается.

При 10000 отсчетах качество идентификации для некоторых законов распределения не очень хорошее. Ошибки возникают при идентификации законов распределения Лапласа и (Sech x)^2 при некоторых значениях их параметров. Уже при 50000 отсчетах результаты гораздо лучше. Дальнейшее увеличение числа отсчетов не приводит к значительному улучшению качества идентификации.

Для того чтобы построить качественный классификатор, необходимо иметь качественные данные. Никакой из методов построения классификаторов, никогда не даст нужного качества, если имеющийся набор примеров не будет достаточно полным и представительным для той задачи, с которой придется работать системе.

Результаты идентификации законов распределения при M=30, E=0.0000001, α=0.9 приведены в табл. 6.4 и 6.5.

Таблица 6.3

Типовые законы распределения

Название закона распределения	Функция распределения
Равномерный	$F(x)=\begin{cases}0,(-\infty<x<a)\\ \dfrac{x-a}{b-a},(a<x<b)\\ 1,(b<x<\infty)\end{cases}$
Симпсона	$F(x)=\begin{cases}0,(-\infty<x<a)\\ \dfrac{2(x-a)^2}{(b-a)^2},(a<x<(a+b)/2)\\ 1-\dfrac{2(b-x)^2}{(b-a)^2},((a+b)/2<x<b)\\ 1,(b<x<\infty)\end{cases}$
Арксинуса	$F(x)=\begin{cases}0,(-\infty<x<-a)\\ \dfrac{1}{2}+\dfrac{1}{\pi}\arcsin\dfrac{x}{a},(-a<x<a)\\ 1,(a<x<\infty)\end{cases}$
Коши	$F(x)=\dfrac{1}{\pi}arctg\dfrac{x-\mu}{a}+\dfrac{1}{2}$
Лапласа	$F(x)=\begin{cases}\dfrac{1}{2}e^{\lambda(x-\mu)},(-\infty<x<\mu)\\ 1-\dfrac{1}{2}e^{-\lambda(x-\mu)},(\mu<x<\infty)\end{cases}$
Вейбулла	$F(x)=\begin{cases}0,(-\infty<x<0)\\ 1-\exp(-\beta x^{\alpha}),(0<x<\infty)\end{cases}$
Рэлея	$F(x)=\begin{cases}0,(-\infty<x<0)\\ 1-\exp(-\dfrac{x^2}{2\sigma^2}),(0<x<\infty)\end{cases}$
Экспоненциальный	$F(x)=\begin{cases}0,(-\infty<x<0)\\ 1-e^{-\alpha x},(0<x<\infty)\end{cases}$
(Sech x)^2	$F(x)=\dfrac{1}{2}+\dfrac{1}{2}\tanh(ax)$

Таблица 6.4

Результаты идентификации

Название закона распределения и значение параметра	% правильного определения, V =			Название закона распределения в случаях ошибочной идентификации		
	10000	50000	70000	V=10000	V=50000	V=70000
Равномерн. ЗР, b=1	100	100	100			
Равномерн. ЗР, b=2	100	100	100			
Экспоненц. ЗР, λ=1	100	100	100			
Экспоненц. ЗР, λ=3	100	100	100			
ЗР Коши, a=1	100	100	100			
ЗР Коши, a=3	100	100	100			
ЗР Коши, a=5	100	100	100			
ЗР арксинуса, a=1	100	100	100			
ЗР арксинуса, a=3	100	100	100			
ЗР арксинуса, a=4	100	100	100			
ЗР Рэлея, σ=1	100	100	100			
ЗР Рэлея, σ=3	100	100	100			
ЗР (Sech x)^2, a=2	100	100	100			

Таблица 6.5

Результаты идентификации

Название закона распределения и значение параметра	% правильного определения, V =			Название закона распределения в случаях ошибочной идентификации		
	10000	50000	70000	V=10000	V=50000	V=70000
3P (Sech x)^2, a=3	86	98	99	Лапласа, λ=5	Лапласа, λ=5	Лапласа, λ=5
3P (Sech x)^2, a=4	91	96	99	(Sech x)^2, a=5 (6%). Арксинуса, a=1 (3%)	(Sech x)^2, a=5	(Sech x)^2, a=5
3P (Sech x)^2, a=5	28	61	84	Арксинуса, a=1.(64%).(Sech x)^2, a=4.(8%)	Арксинуса, a=1	Арксинуса, a=1
3P Симпсона, b=3	100	100	100			
3P Симпсона, b=5	100	100	100			
3P Симпсона, b=7	100	100	100			
3P Лапласа, λ=1	100	100	100			
3P Лапласа, λ=3	100	100	100			
3P Лапласа, λ=5	69	93	99	(Sech x)^2, a=3	(Sech x)^2, a=3	(Sech x)^2, a=3

6.4. Пример использования сети рекуррентной Эльмана для решения задачи прогнозирования

При решении задач прогнозирования роль нейронной сети состоит в предсказании будущей реакции системы по ее предшествующему поведению. Обладая информацией о значениях переменной x в моменты, предшествующие прогнозирования $x(t-1), x(t-2),\ldots,x(t-N)$, сеть вырабатывает решение, каким будет наиболее вероятное значение последовательности $\hat{x}(t)$ в текущий момент t [16].

Алгоритм получения прогноза на основе рекуррентной сети Эльмана:

1. Обнулить контекстный слой сеть (элементы задержки).
2. Подать на внешние входы сети N последних значений временного ряда.
3. Рассчитать цикл сети. Полученное значение y является прогнозом.
4. Добавить спрогнозированное значение y к временному ряду в качестве нового отсчета.
5. Повторить пункты 2-4 до достижения заданной глубины прогноза.

При исследовании оценивались следующие параметры, характеризующие качество прогноза нейросети:

математическое ожидание ошибки тестирования;

дисперсия ошибки тестирования;

СКО ошибки тестирования;

$СКО_{ош}/СКО_{в.р.}$ - отношение стандартного отклонения ошибки прогноза к стандартному отклонению обучающих данных; если оно существенно меньше единицы (например, ниже 0,1), то это говорит о хорошем качестве прогнозирования;

показатель погрешности $MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|P_t - \hat{P}_t\right|}{P_t}$;

время обучения нейронной сети.

При исследовании проводилось тестирование системы на различных временных рядах, как стохастическом, так и на реальном ряде отказов АТС. Анализ производился и использованием рядов, представленных на рис. 6.9 и 6.10.

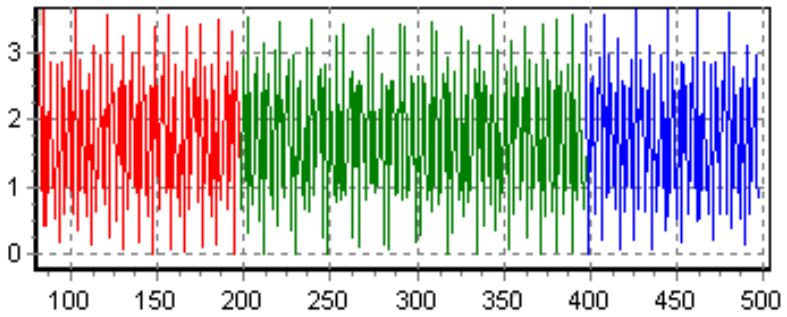


Рис. 6.9 Вид используемого стохастического временного ряда

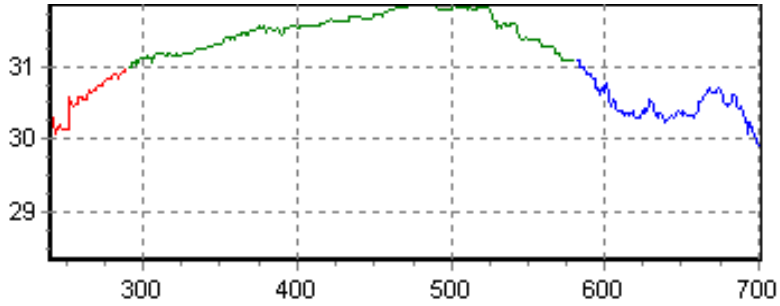


Рис. 6.10 Вид используемого реального временного ряда

Для приведенных выше временных рядов проводились следующие виды исследований:

1. Исследование зависимости качества прогноза от параметров алгоритма обучения (в процессе проведения тестов изменялись коэффициенты обучения и момента. Фиксированной оставалась структура сети: число внешних входов нейросети – 3, число нейронов скрытого слоя – 10, функция активации для скрытых и выходного нейронов – гиперболический тангенс).
2. Исследование зависимости качества прогноза от количества нейронов в скрытом слое (при исследовании влияния числа скрытых нейронов на качество прогноза фиксировались следующие значения: число входов – 3, функции активации для скрытых и выходного нейрона

– гиперболический тангенс, коэффициент обучения $\eta = 0{,}05$, коэффициент момента $\alpha = 0{,}2$).

3. Исследование зависимости качества прогноза сети Эльмана от числа входов нейросети (при подборе оптимального числа входов для рекуррентной нейронной сети Эльмана фиксировались следующие значения: число скрытых нейронов – 6, функции активации для скрытых и выходного нейрона – гиперболический тангенс, коэффициент обучения $\eta = 0{,}05$, коэффициент момента $\alpha = 0{,}2$.).

4. Исследование зависимости качества прогноза сети Эльмана от вида функций активации (вид функции активации во многом определяет функциональные возможности нейронной сети. Для выбранного ряда фиксировались следующие значения: число входов – 3, число скрытых нейронов – 6, коэффициент обучения $\eta = 0{,}05$, коэффициент момента $\alpha = 0{,}2$).

Проведенные исследования показали, что для выбранного стохастического временного ряда оптимальными являются следующие значения параметров обучения и структуры нейросети:

- коэффициент обучения $\eta = 0{,}01$; коэффициент момента $\alpha = 0$;
- число входов сети $N = 10$; число нейронов в скрытом слое $K = 6$;
- функция активации для скрытых нейронов – синусоидальная;
- функция активации для выходных нейронов – синусоидальная.

Результаты обучения, тестирования и прогноза на основе обученной сети представлены на рис. 6.11, 6.12, 6.13.

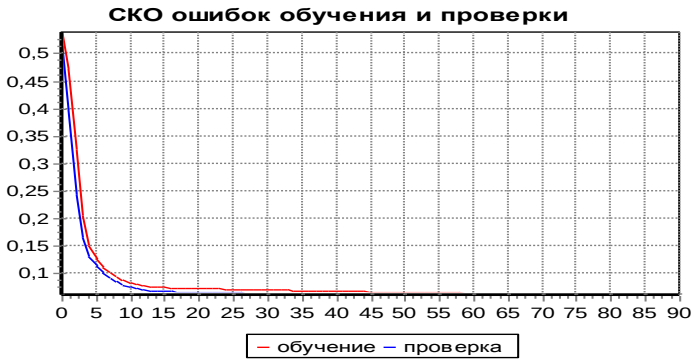


Рис. 6.11 Графики изменения СКО обучения и проверки в процессе обучения нейросети

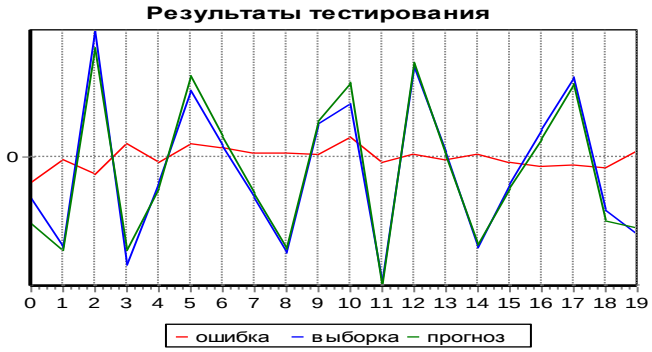


Рис. 6.12 Графики ошибки на этапе тестирования сети, тестирующей выборки и значения нейросети на этапе тестирования

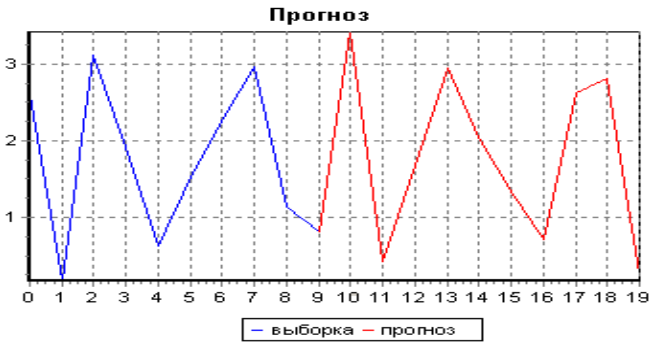


Рис. 6.13 Прогноз обученной сети

Проведенные испытания показали, что для выбранного временного ряда, оптимальными являются следующие значения:

- коэффициент обучения $\eta = 0{,}03$; коэффициент момента $\alpha = 0{,}5$;
- число входов сети $N = 10$; число нейронов в скрытом слое $K = 8$;

- функция активации для скрытых нейронов – сигмоидальная;
- функция активации для выходных нейронов – синусоидальная.

На рис. 6.14, 6.15, 6.16 представлены результаты обучения, тестирования и прогноз обученной нейронной сети.

Рис. 6.14 Графики изменения СКО обучения и проверки в процессе обучения нейросети

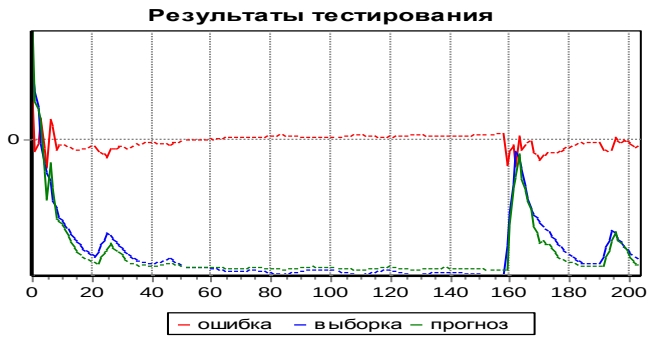


Рис. 6.15 Графики ошибки на этапе тестирования сети, тестирующей выборки и значения нейросети на этапе тестирования

Рис. 6.16 Прогноз обученной сети

Предметный указатель

Список литературы

1. Горбань А.Н. Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей // Сибирский журнал вычислительной математики. – 1998. – Т. 1, № 1. – С. 12–24.
2. Колмогоров А.Н. О представлении непрерывных функций нескольких переменных суперпозициями непрерывных функций меньшего числа переменных. Избранные труды: В 6 т. Т 1. Математика и механика.– М.: Наука, 2005. – 519 с.
3. Колмогоров А.Н. О представлении непрерывных функций нескольких переменных в виде суперпозиции непрерывных функций одного переменного и сложения. Избранные труды: В 6 т. Т 1. Математика и механика.– М.: Наука, 2005. – 519 с.
4. Оссовский С. Нейронные сети для обработки информации / Пер. с пол. И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.: ил.
5. Хайкин С. Нейронные сети: Полный курс: Пер. с англ. - 2-е изд. – М.: Вильямс, 2006. – 1104 с.: ил.
6. Lippman R. An introduction to computing with neural nets // IEEE ASSP Magazine. 1987. – April. – P. 4–22.
7. Martinetz M., Bercovich S., Schulten K. “Neural-gas” network for quantization and its application to time series prediction // Trans. Neural Networks. – 1993. – Vol. 4. – P. 558–569.
8. McCulloch W.S., Pitts W.H. A logical calculus of ideas immanent in nervous activity// Bull. Math. Biophysics. – 1943. – Vol. 5. – P. 115–119.
9. Riedmiller M., Brawn H. RPROP – a fast adaptive learning algorithm. Technical Report, Karlsruhe: University Karlsruhe, 1992.
10. Ritter H., Schulten K. On the stationary state of the Kohonen self-organizing sensory mapping // Biological Cybernetics. –1986. – Vol. 54. – P. 234–249.
11. Rosenblatt F. The Perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review. –1958. – Vol. 65. – P. 386–408.

12. Widrow B., Hoff M. Adaptive switching circuits // Proc. IRE WESCON Convention Record, 1960. – P. 107–115.
13. Солдатова О.П. Использование многослойного персептрона и радиально-базисных сетей для прогнозирования временных рядов. Компьютерные технологии в науке, практике и образовании // Тр. Всерос. межвуз. научн.-практ. конф.// ГОУ ВПО Самар. гос. техн. ун-т. Самара, 2005. – С. 20–22.
14. Солдатова О.П., Байков С.С. Практическое применение нейронных сетей для решения задач классификации и идентификации // Исследовано в России: электрон. журн. – 135/060201. – С. 1261–1269. – Режим доступа к журн.: http://zhurnal.ape.relarn.ru/articles/2006/135.pdf.
15. Солдатова О.П., Каюкова А.С. Аппроксимация функций RBF-сетями с произвольным выбором функций активации. Перспективные информационные технологии в научных исследованиях, проектировании и обучении «ПИТ-2006» // Тр. научн.-техн. конф. с междунар. участием. – Самара, 2006. – Т. 1. – С. 159–169.
16. Солдатова О.П., Сталькина У.М. Нейросетевые модели прогнозирования временных рядов // Материалы междунар. научн.-техн. конф., посвящ. 110-летию изобретения радио и 75-летию Сарат. гос. техн. ун – та. Сарат. гос. техн. ун – т. Саратов, 2005. – С.17–21.
17. Уоссерман Ф. Нейрокомпьютерная техника: теория и практика / Пер. с англ. Ю.А. Зуев. – М.: Мир, 1992.