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

Курс лекций

Самара

2013

Оглавление

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 [1]:

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.*Эта процедура выстраивает в определённом порядке веса нейронов сети для обеспечения необходимой взаимосвязи между ними.

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 нелинейные 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, представляющих собой таблицу соответствий вход-выход для конкретной задачи. Ранее использованные примеры могут быть использованы для обучения снова в таком же или ином порядке.

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* (faulttolerance). 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 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.

*8. Аналогия с нейробиологией.*Строение нейронных стетй определяется аналогией с живым мозгом, являющимся доказательством возможности существования отказоустойчивых вычислительных параллельных систем, эффективно решающих поставленные задачи.

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, сжатия данных, классификации и распознавания образов, прогнозирования, идентификации, оценивания и ассоциативного управления.

1.2 Биологические основы нейронных сетей

Искусственные нейронные сети возникли на основе знаний о функционировании нервной системы живых существ. Невную систему человека можно представить в виде трёхступенчатой системы.

Центром этой системы является *мозг*, представляемый сетью нервных клеток,то есть нейронной сетью. Он получает информацию, анализирует ее и выдает соответствующие решения. *Рецепторы* получают информацию от тела и окружающей среды и преобразуют её в электрический импульс, передаваемый в мозг. *Эффекторы* преобразовывают электрические импульсы, вырабатываемы мозгом в выходные сигналы.

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.1.3)[2].

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 находиться как на дендритах, так и непосредственно в теле клетки. Самым распространённым типом синапсов является химический синапс, который работает следующим образом.

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. Предсинаптический процесс формирует субстанцию, которая методом диффузии передаётся по синаптическим связям и влияет на

постсинаптический процесс. Синапс преобразует пресинаптический электрический сигнал в химический, а затем в постсинаптический - электрический.

Данные 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.

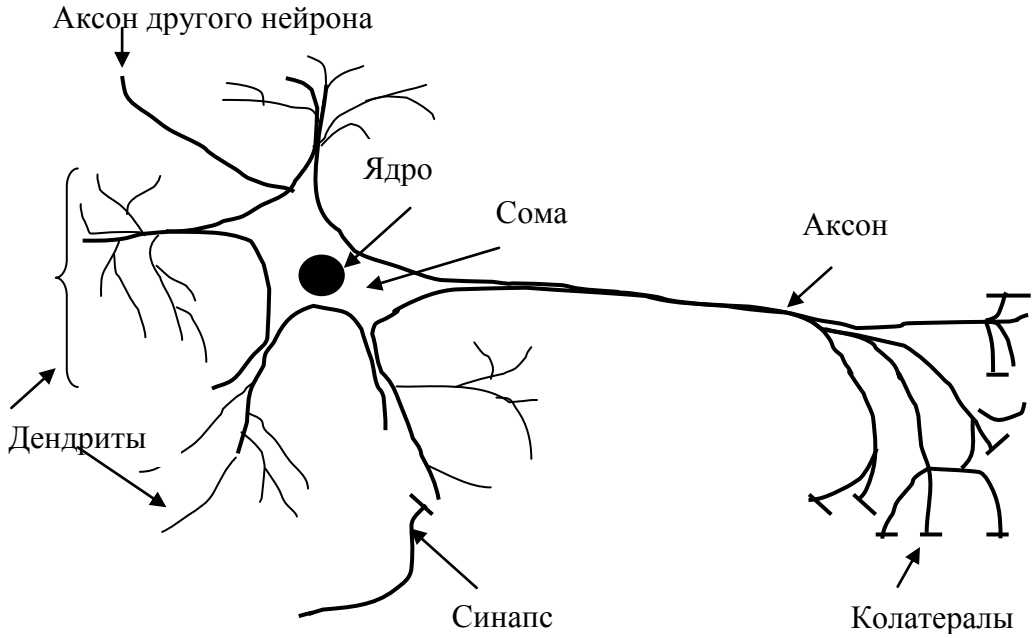


Рис. 1.3 Упрощенная структура биологической нервной клетки

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 превысило порог активации клетки, значение выходного сигнала начинает нарастать, приобретая характер нервного импульса, пересылаемого аксоном на другие нейроны, подключенные к данной клетке (рис.1.4). Величина этого сигнала не зависит от степени превышения порога срабатывания.

Количество взаимодействующих друг с другом нервных клеток в человеческом мозге оценивается, как 10^{11}- 10^{14}. 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.

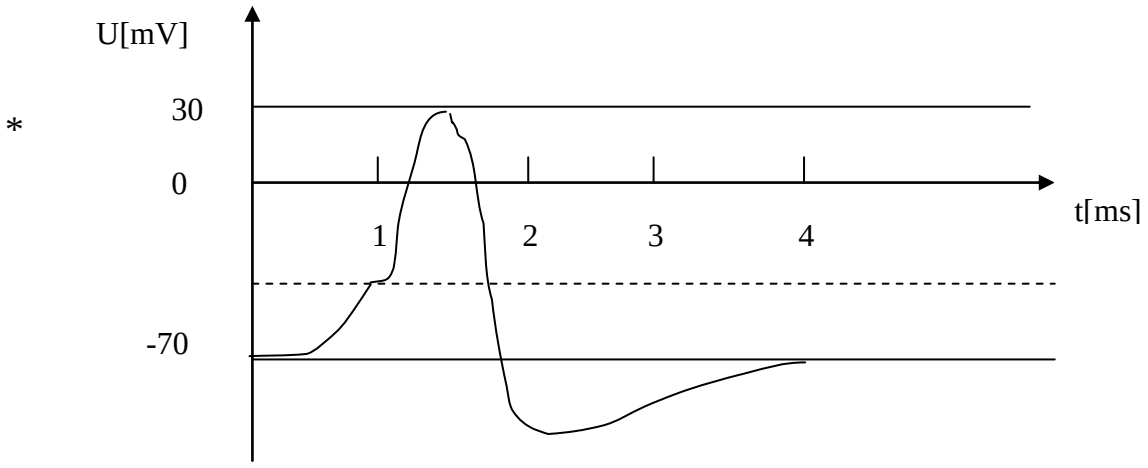


Рис. 1.4 Типичная форма нервного импульса

Существует огромное количество форм и размеров нейронов в зависимости от того, в какой части мозга он находятся.Самыми распространёнными нейронами коры головного мозга являются пирамидальные нейроны.

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

Нейрон является единицей обработки информации в нейронной сети. 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 [2]. Структурная схема этой модели представлена на рис. 1.5.

Сигналы x_j на входе синапсов j $(j = 1,2,...,N)$, связанные с нейроном 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, которая ограничивает амплитуду выходного сигнала. Выходной сигнал нейрона y_i определяется при этом зависимостью

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

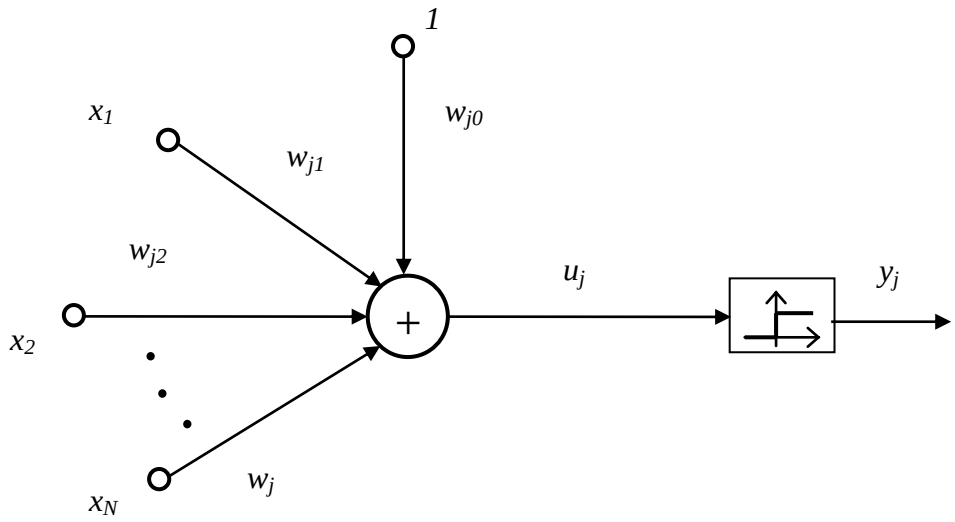


Рис. 1.5 Модель нейрона МакКаллока-Питса

Аргументом функции выступает суммарный сигнал, формируемый сумматором искусственного нейрона.

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

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.

Использование порогового сигнала w_{i0} обеспечивает эффект аффинного преобразования выхода линейного сумматора u_i .

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:

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

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 [2]. 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 $y_i=0$, а 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 $y_i=1$, а 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 предоставляются 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, 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 примеров (выборок). Такая минимизация для персептрона проводится по методу безградиентной оптимизации. Эффективность метода при большом количестве обучающих выборок невелика, а количество циклов обучения и длительность быстро возрастают, причем без гарантии достижения минимума целевой функции. Устранить эти недостатки можно только в случае применения непрерывной функции активации, при которой целевая функция Eтакже 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 [2]. Структура нейрона представлена на рис. 1.6.

Входные сигналы x_j ($j=1,2,\ldots,N$) суммируются с учетом соответствующих весов w_{ij} (сигнал поступает в направлении от узла j к узлу i) 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.

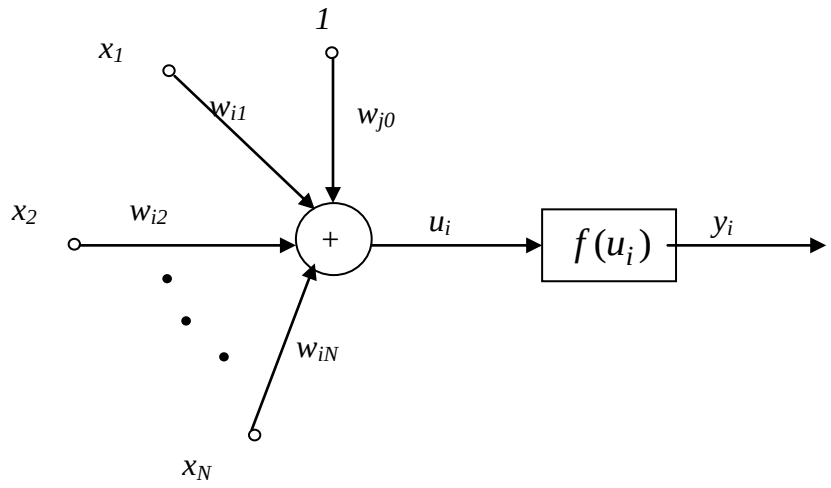


Рис. 1.6 Модель сигмоидального нейрона

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. 1.7.

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.

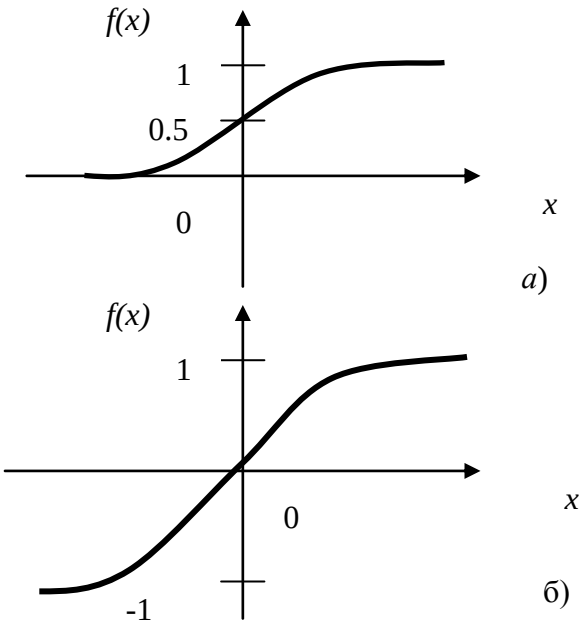


Рис. 2.7 Графики сигмоидальных функций:
а – логистическая; б – гиперболический тангенс

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 *i*-я составляющая градиента имеет вид:

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

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

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.

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

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.

$$\frac{dw_{ij}}{dt} = -\mu\delta_i x_j \tag{1.20}$$

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 1.5 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*(*winnertakesall* – победитель получает все) [2]. 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:

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

Проигравшие нейроны не изменяют свои весовые коэффициенты.

Схема соединения нейронов типа *WTA* изображена на рис.1.10.

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.

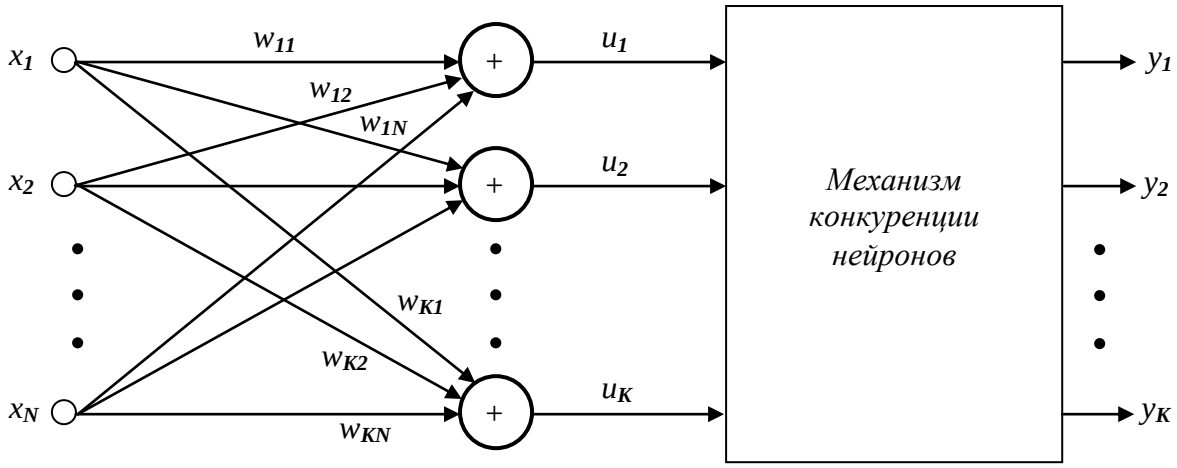


Рис. 1.8 Схема соединения нейронов типа WTA

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 [2, 3]. 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. 1.9). $u_i = \sum_{j=0}^{N} w_{ij} x_j$, $y_i = f(u_i)$. В качестве функции активации часто используется линейная функция, тогда $y_i = u_i$.

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). Если принять $\eta = 1$, то $w_{ij} = x_j$ уже после первой итерации обучения. Ввод нового вектора $\boldsymbol{x}$ вызовет адаптацию весов к его компонентам и «забывание» предыдущих значений. При $\eta < 1$ инстар принимает усреднённое значение обучающих векторов.

Входные данные должны быть нормированы, то есть $\|x\| = 1$. При этом $x_j \leftarrow \dfrac{x_j}{\sqrt{x_1^2 + x_2^2 + \ldots + x_N^2}}$.

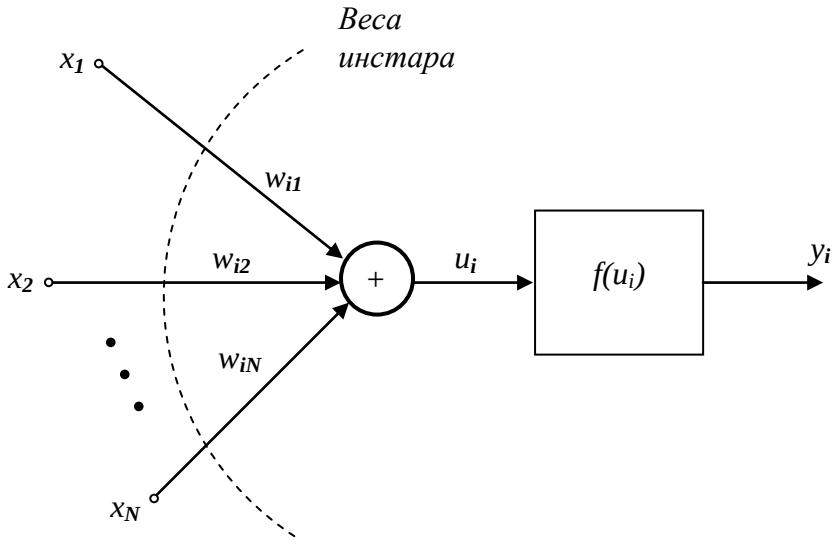


Рис. 1.9 Структурная схема входной звезды (инстара)

Инстар обучается как с учителем, так и без учителя. Если применяется обучение с учителем, то выходные значения y_i являются эталонными значениями d_i.

Выходная звезда Гроссберга (оутстар) выполняет противоположную функцию – функцию командного нейрона, выдавая на выходах определенный вектор, необходимый связанным с ним нейронам, при поступлении сигнала на вход. Структурная схема оутстара представлена на рис. 1.10.

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.

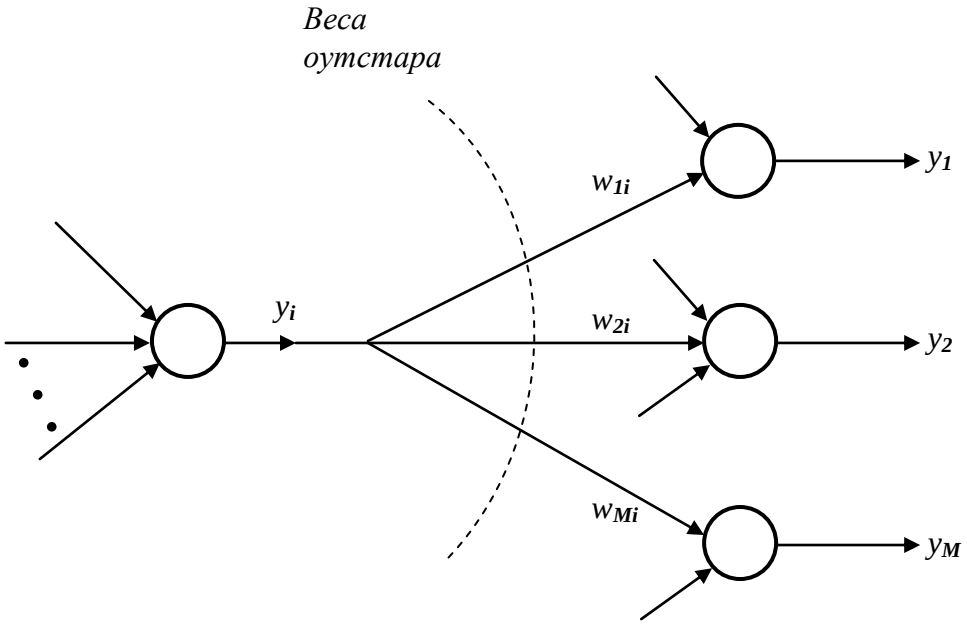


Рис. 1.10 Структурная схема выходной звезды (оутстара)

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 Нейрон Хебба

В процессе исследования нервных клеток Д.Хебб выявил усиление связи между двумя клетками, если они одновременно становятся активными[2]. Если j-ая клетка с выходным сигналом y_j связана с i-ой клеткой с выходным сигналом y_i связью с весом w_{ij}, то на значение веса влияют выходные сигналы y_j и y_i.В соответствии с правилом Хебба вес нейрона изменяется по формуле:

$$\Delta w_{ij} = \eta y_i y_j, \qquad (1.27)$$

где η – это коэффициент обучения, значения которого выбира.тся из интервала (0,1).

Стркутурная схема нейрона Хебба, представленная на рисунке 1.11, соответствует стандартной форме нейрона.

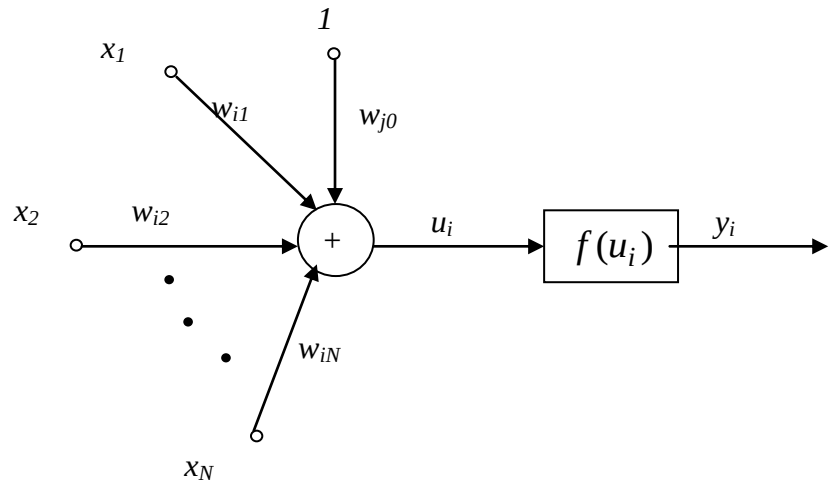


Рис. 1.11 Структурная схема нейрона Хебба

Обучение нейрона по правилу Хебба, может проводиться как с учителем, так и без учителя. При обучении с учителем вместо реального выходного сигнала y_i используется эталонное значение d_i. При этом правило Хебба принимает вид:

$$\Delta w_{ij} = \eta d_i y_j, \tag{1.28}$$

В результате применения правила Хебба веса могут сильно возрастать, так как в каждом цикле обучения происходит суммироваие текущего веса и приращения:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t), \tag{1.29}$$

Одним из способов стабилизации обучения по Хеббу является использовани коэффициента забывания γ. При этом корректирующая формула имеет следующий вид:

$$w_{ij}(t+1) = w_{ij}(t) + (1-\gamma)\Delta w_{ij}(t), \tag{1.30}$$

Значение коэффициента забывания выбирается как правило из интервала (0,1), и составляет некоторый процент от коэффициента обучения η. Рекомендуемые значения коэффициента забывания $\gamma < 0.1$.у

Обучение по Хеббусчитается обучением ассоциативного типа.

При обучении лиейного нейрона стабилизация не наступает даже при введении коэффициента забывания. Выходной сигнал нейрона рассчитывается по формуле:

$$y_i = \sum_{j=0}^{N} w_{ij} x_j = w^T x = x^T w. \tag{1.31}$$

Согласно правилу Хебба:

$$\Delta w = \eta x y, \tag{1.32}$$

И если в выражение (1.32) подставить выражение (1.31) и выбрать для упрощения $\eta = 1$, то получится формула для приращения весов:

$$\Delta w = Cw, \tag{1.33}$$

где $C = xx^T$ - матрица корреляции, которая по определению является симметричной и положительно полуопределённой и имеет собственные натуральные и неотрицательные значения. При этом процесс, определяемый соотношением (1.33), становится расходящимся, а значения компонентов вектора w стремятся к бесконечности.

Е.Ойя модифицировал правило Хебба таким образом, что вектор весов стремится к 1, то есть $\|w\| = 1$. Уточнение весов происходит по формуле:

$$\Delta w = \eta y(x_i - y w_i), \tag{1.34}$$

Доказательство ограниченности весов по правилу Ойя можно получить, представив выражение (1.34) в векторной форме и приняв для упрощения $\eta = 1$:

$$\Delta w = Cw - (w^T C w)w, \tag{1.35}$$

Стабильность процесса достигается, когда при достаточно длительном обучении $\|\Delta w\| = 0$, то есть

$$Cw = (w^T C w)w. \tag{1.36}$$

Если собственное значение матрицы C обозначить через λ, а вектор w подбирать каксвязанный с ней собственный вектор, то по определению собственного значения имеем $Cw = \lambda w$. Подставляя это выражение в (1.35) получим:

$$\lambda = w^T C w = w^T \lambda w = \lambda \|w\|^2. \tag{1.37}$$

Отсюда следует. Что пррименение правила Ойя приводит к ограничению модуля вектора вестов единицей.

1.9 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	$f(x)=e^{-kx}$	$(0,\infty)$
Гаусса	$f(x)=\exp\left(-\frac{\lVert x-c_i\rVert^2}{2\sigma_i^2}\right)$	$(-\infty,\infty)$
Пороговая	$f(x)=\begin{cases}0, x<0\\1, x\geq 0\end{cases}$	$(0,1)$
Модульная	$f(x)=\lvert x\rvert$	$(0,\infty)$
Знаковая (сигнатурная)	$f(s)=\begin{cases}1, x>0\\-1, x\leq 0\end{cases}$	$(-1,1)$
Квадратичная	$f(x)=x^2$	$(0,\infty)$

2 Многослойные сети прямого распространения сигнала

2.1 Однослойный персептрон

Однослойный персептрон образуют нейроны, расположенные в одной плоскости (рисунок 2.1). Каждый нейрон имеет поляризатор, то есть единичный сигнал, который с весом w_{i0} поступает на вход нейрона, а также множество связей с весами w_{ij}, по которым поступают входные сигналы x_j.

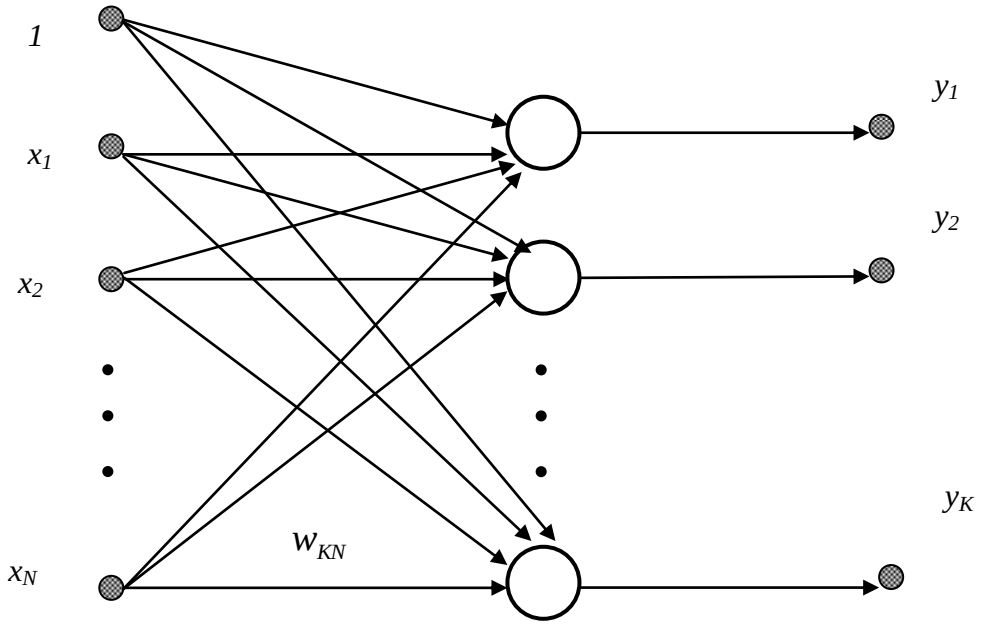


Рисунок2.1 Структура однослойного персептрона

Значения весов подбираются в процессе обучения сети путём приближения реальных выходных значений y_i к эталонным значениям d_i. Мерой близости является значение целевой функции.При использовании p обучающих векторов $\langle x,d \rangle$ для сети из K нейронов, целевую функцию можно определить следующим образом:

$$E = \frac{1}{2}\sum_{m=1}^{p}\sum_{i=1}^{K}\left(y_i^{(m)} - d_i^{(m)}\right)^2 \qquad (2.1)$$

Выходные сигналы y_i являются функциями весов сети w_{ij} и уточняются в процессе обучения в соответствии с критерием минимизации целевой функции (2.1).

Расположенные на одном уровне нейроны функционируют независимо друг от друга, поэтому возможности однослойного персептрона определяются возможностями отдельных нейронов.Каждый нейрон реализует функциональное отображение $y_i = f\left(\sum_{j=0}^{N} w_{ij}x_j\right)$, где f -

сигмоидальная функция, поэтому значение выходного сигнала будет зависеть от знака выражения $\sum_{j=0}^{N} w_{ij} x_j$.

Выходной сигнал y_i при фиксированных значениях весов зависит от входного вектора x, который определяет гиперплоскость, разделяющую многомерное входное пространство на два подпространства.Вследствие этого, задача классификации (приписывание выходному сигналу значения 1 или 0), может быть решена одним нейроном, если это задача линейной разделимости классов. Добавление нейронов в единственный слой сети не позволяет улучшить её функциональные возможности. Вследствие этого, однослойная сеть имеет небольшое практическое значение. Добавление ещё одного слоя нейронов, даже состоящего из единственного нейрона позволяет существенно расширить возможнойти сети.

2.2 Многослойный персептрон

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. 2.2 [2]. 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.

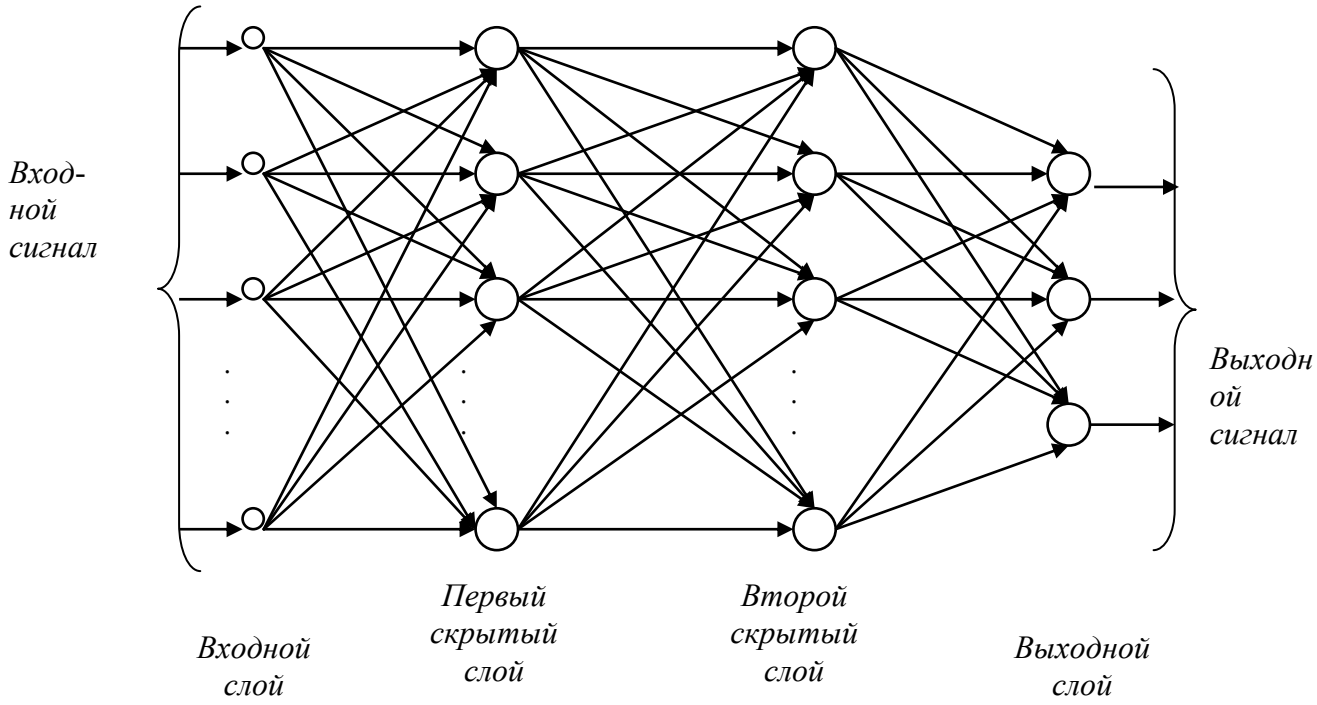


Рисунок2.2 Структура многослойного персептрона с двумя скрытыми слоями

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}.
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.

2.3 Структура двухслойной сигмоидальной нейронной сети

На рис. 2.3 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.

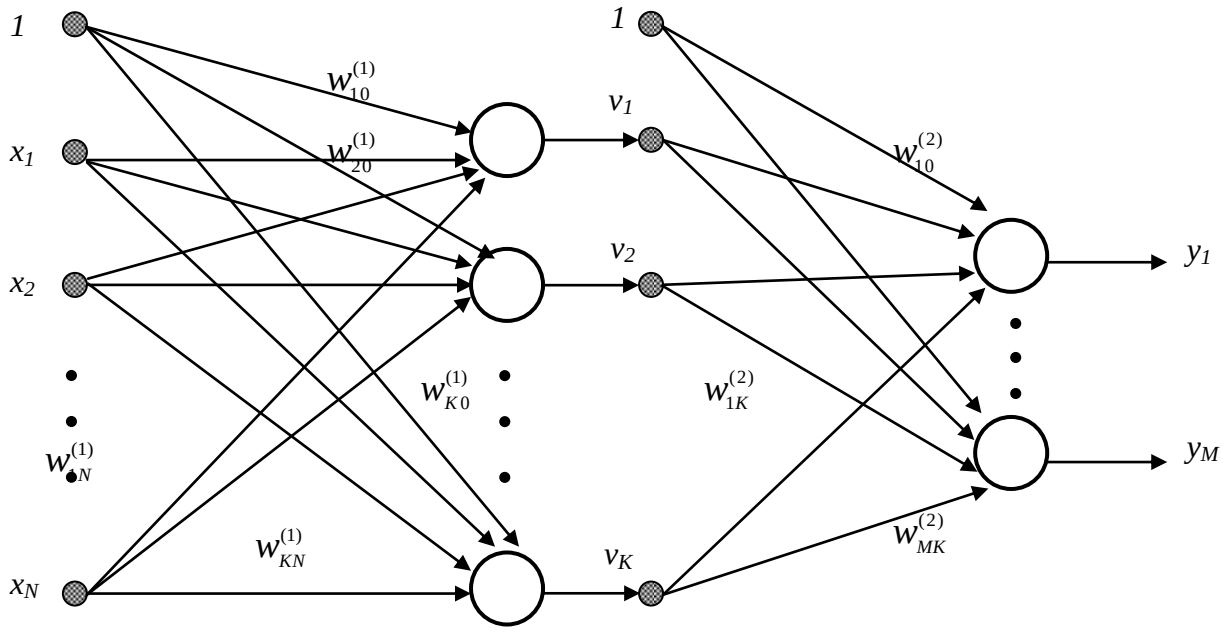


Рисунок2.3 Обобщенная структура двухслойной сигмоидальной нейронной сети

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

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

$$v_i = f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_j\right), \qquad (2.2)$$

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

$$y_s = f\left(\sum_{i=0}^{K} w_{si}^{(2)} v_i\right) = f\left(\sum_{i=0}^{K} w_{si}^{(2)} f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_j\right)\right). \qquad (2.3)$$

Из формулы (2.3) 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.

2.4 Градиентные методы обучения многослойного персептрона

2.4.1 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

$$E(w+p)=E(w)+[g(w)]^T p+\frac{1}{2}p^T H(w)p+..., \qquad (2.4)$$

где $g(w)=\nabla E=\left[\frac{\partial E}{\partial w_1},\frac{\partial E}{\partial w_2},...,\frac{\partial E}{\partial w_n}\right]^T$ - 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 (2.4) *p* играет роль направляющего вектора, зависящего от фактических значений вектора *w*. На практике чаще всего рассчитываются три первых члена ряда (2.4), а последующие члены ряда просто игнорируются. При этом зависимость (2.4) 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 $w_t = w_0$) [2]:

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.

2.4.2 Подбор коэффициента обучения

После правильно выбранного направления p_t, в градиентных алгоритмах обучения, следует определить 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. Простейший 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 ηдля каждого слоя по формуле

$$\eta \le \min\left(\frac{1}{n_i}\right), \tag{2.5}$$

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

В [1] предложена следующая адаптивная формула для корректировки значений коэффициента обучения в зависимости от итерации обучения:

$$\eta(t) = \frac{\eta_0}{1+\left(\frac{t}{\tau}\right)}, \tag{2.6}$$

где η_0 - начальное значение коэффициента обучения, τ - константа времени поиска, задаваемые пользователем, а t - номер итерации обучения.

При достаточно больших значениях η_0 и при большом числе итераций, значительно превосходящих константу τ, алгоритм обучения будет вести себя как стохастический алгоритм аппроксимации, а веса будут сходиться к своим оптимальным значениям.

2.4.3 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:

$$E(w) = \frac{1}{2}\sum_{t=1}^{P}\sum_{s=1}^{M}\left(y_s^{(t)} - d_s^{(t)}\right)^2 \tag{2.7}$$

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

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

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

используется целевая функция вида (2.7). В последующем изложении используется целевая функция вида (2.8).

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_iскрытого 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:

$$w(t+1) = w(t) + \Delta w, \qquad (2.9)$$

где $\Delta w = \eta\, p(w), \qquad (2.10)$

η - коэффициент обучения, а *p(w)* – направление в многомерном пространстве *w*. В алгоритме обратного распространения ошибки *p(w)* определяется как частная производная $\frac{\partial E}{\partial w_{ij}}$, 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 [2]:

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 (2.9) и (2.10) 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 2.3. Используется сигмоидальная функция активации, при этом в случае гиперболического тангенса производная функции активации равна

$$\frac{df(u)}{du} = 1 - f^2(u) \tag{2.11}$$

В случае логистической функции производная равна

$$\frac{df(u)}{du} = f(u) \cdot (1 - f(u)) \tag{2.12}$$

В формулах (2.11) и (2.12) под переменной *u* будем понимать выходные сигналы сумматоров нейронов скрытого или выходного слоя, представленных формулами (2.13) и (2.14).

$$u_i^{(1)} = \sum_{j=0}^{N} w_{ij}{}^{(1)} x_j \tag{2.13}$$

$$u_s^{(2)} = \sum_{i=0}^{K} w_{si}{}^{(2)} v_i \tag{2.14}$$

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

$$\Delta w_{ij}^{(m)} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \tag{2.15}$$

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 2.3 обозначений целевая функция для выходного слоя нейронов определяется следующим образом:

$$E = \frac{1}{2}\sum_{s=1}^{M}\left[f\left(\sum_{i=o}^{K} w_{si}^{(2)} v_i\right) - d_s \right]^2 = \frac{1}{2}\sum_{s=1}^{M}\left[f\left(\sum_{i=0}^{K} w_{si}^{(2)} f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_i\right)\right) - d_s \right]^2 \tag{2.16}$$

$$\frac{\partial E}{\partial w_{si}^{(2)}} = (y_s - d_s) \cdot \frac{df(u_s^{(2)})}{du_s^{(2)}} \cdot v_i \tag{2.17}$$

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 (2.17) можно представить в виде:

$$\frac{\partial E}{\partial w_{si}^{(2)}} = \delta_s^{(2)} v_i \qquad (2.18)$$

Компоненты градиента относительно нейронов скрытого слоя описываются более сложной зависимостью:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{s=1}^{M} (y_s - d_s) \cdot \frac{dy_s}{dv_i} \cdot \frac{dv_i}{dw_{ij}^{(1)}} \qquad (2.19)$$

В другом виде эта зависимость может быть выражена формулой:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{s=1}^{M} (y_s - d_s) \cdot \frac{df(u_s^{(2)})}{du_s^{(2)}} \cdot w_{si}^{(2)} \frac{df(u_s^{(1)})}{du_i^{(1)}} x_j \qquad (2.20)$$

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:

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

2.4.4 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

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

В этом случае коррекция весовых коэффициентов производится по формуле:

$$w_{ij}(t+1) = w_{ij}(t) + \eta p_t \qquad (2.24)$$

В другом виде формулу коррекции весов по методу наискорейшего спуска можно представить следующим образом:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \cdot \frac{\partial E(t)}{\partial w_{ij}(t)} \qquad (2.25)$$

Ограничение слагаемым первого порядка при разложении функции в ряд Тейлора, не позволяет использовать информацию о ее кривизне. Это обуславливает линейную сходимость метода. 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 формуле:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \cdot \frac{\partial E(t)}{\partial w_{ij}(t)} + \alpha(w_{ij}(t) - w_{ij}(t-1)) \qquad (2.26)$$

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

Первое слагаемое в формуле (2.26) 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 $\Delta w_{ij}(t) = \eta p(t) + \alpha \Delta w_{ij}(t)$, поэтому эффективное приращение весов можно писать формулой:

$$\Delta w_{ij}(t) = \frac{\eta}{1-\alpha} p(t) \qquad (2.27)$$

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, $\Delta w_{ij}(t) = 0$. При этом показатель градиента начинает доминировать над показателем момента и процесс развивается в направлении минимизации, заданном вектором градиента [2].

2.4.5 Алгоритм Левенберга-Марквардта

В данном алгоритме 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

$$p(t)=-[H(w(t))]^{-1}g(w(t)) \qquad (2.28)$$

Формула (2.28) однозначно указывает направление $p(t)$, 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 (2.28) представляет собой основу ньютоновского алгоритма оптимизации и 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 $G(w(t))$, которое в алгоритме Левенберга-Марквардта рассчитывается на основе содержащейся в градиенте информации с учётом некоторого регуляризационного фактора [2].

Для описания данного метода представим целевую функцию в виде:

$$E(w)=\frac{1}{2}\sum_{s=1}^{M}[e_s(w)]^2, \qquad (2.29)$$

где $e_s=[y_s(w)-d_s(w)]$. При использовании обозначений

$$e(w)=\begin{bmatrix} e_1(w) \\ e_2(w) \\ \dots \\ e_M(w) \end{bmatrix}, \quad J(w)=\begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \dots & \frac{\partial e_1}{\partial w_p} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \dots & \frac{\partial e_2}{\partial w_p} \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_M}{\partial w_1} & \frac{\partial e_M}{\partial w_2} & \dots & \frac{\partial e_M}{\partial w_p} \end{bmatrix}, w_t=w(t). \qquad (2.30)$$

Вектор градиента и аппроксимированная матрица гессиана, соответствующие целевой функции (2.29) на t-ом шаге алгоритма, определяются в виде:

$$g(w(t))=[J(w(t))]^T e(w), \qquad (2.31)$$

$$G(w(t))=[J(w(t))]^T J(w(t))+R(w(t)), \qquad (2.32)$$

где $R(w(t))$обозначены компоненты гессиана $H(w(t))$, содержащие высшие производные относительно $w(t)$. Аппроксимация $R(w(t))$ осуществляется с помощью регуляризационного фактора $v(t)$, в котором переменная $v(t)$ называется параметром Левенберга-Марквардта и является скалярной величиной, изменяющейся в процессе обучения. Таким образом:

$$G(w(t)) = [J(w(t))]^T J(w(t)) + v(t). \quad (2.33)$$

В начале процесса обучения, когда текущее решение $w(t)$ далеко от искомого решения, следует использовать значение параметра $v(t)$ намного превышающее $[J(w(t))]^T J(w(t))$. В этом случае гессиан фактически заменяется регуляризационным фактором:

$$G(w(t)) \cong v(t), \quad (2.34)$$

а направление минимизации выбирается по методу наискорейшего спуска:

$$p(t) = -\frac{g(w(t))}{v(t)}. \quad (2.35)$$

По мере приближения к искомому решению величина параметра $v(t)$ понижается и первое слагаемое в формуле (2.33) начинает играть более важную роль. Таким образом, на эффективность алгоритма влияет правильный выбор величины $v(t)$. В методике, предложенной Д.Марквардтом, значение $v(t)$ изменяется по следующей схеме:

$$\textit{если } E\left(\frac{v(t-1)}{r}\right) \le E(t), \textit{то принять } v(t) = \frac{v(t)-1}{r};$$

$$\textit{если } E\left(\frac{v(t-1)}{r}\right) > E(t) \textit{ и } E(v(t-1)) < E(t), \textit{то принять } v(t) = v(t-1);$$

$$\textit{если } E\left(\frac{v(t-1)}{r}\right) > E(t) \textit{ и } E(v(t-1)) > E(t), \textit{то увеличить } m \textit{ раз значение } v$$

$$\textit{до достижения } E(v(t-1)r^m) \le E(t), \textit{ принимая } v(t) = v(t-1)r^m,$$

где $E(t), v(t), E(t-1), v(t-1)$ обозначают значения целевой функции и параметра v на t -ом и $(t-1)$ -ом шагах алгоритма, а $r > 1$ - обозначает коэффициент уменьшения v.

Такая процедура изменения $v(t)$ применяется до момента, когда коэффициент верности отображения, рассчитываемый по формуле

$$q = \frac{E(t) - E(t-1)}{[\Delta w(t)]^T g(t) + 0{,}5[\Delta w(t)]^T G(t) \Delta w(t)}, \quad (2.36)$$

достигнет значения, близкого к единице. При этом квадратичная аппроксимация целевой функции имеет высокую степень совпадения с истинными значениями, следовательно, регуляризационный фактор в формуле (2.33) может быть приравнен нулю, а процесс определения гессиана сводится к аппроксимации первого порядка, при этом алгоритм Левенберга-Марквардта превращается в алгоритм Гаусса-Ньютона, имеющему квадратичную сходимость.

2.5 Эвристические алгоритмы обучения многослойного персептрона

2.5.1 Алгоритм RPROP.

Простой эвристический алгоритм, демонстрирующий высокую эффективность обучения, - это алгоритм М. Ридмиллера и Х. Брауна, называемый RPROP [2]. 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:

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

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.

2.5.2 Алгоритм Quickprop

Алгоритм QuickProp содержит элементы, предотвращающие зацикливание в точке неглубокого локального минимума, возникающего в результате работы нейрона на фазе насыщения сигмоидальной кривой, где из-за близости к нулю производной функции активации процесс обучения практически прекращается.

Вес w_{ij} на *k*-ом шаге алгоритма изменяется согласно правилу:

$$\Delta w_{ij}(k) = -\eta_k\left[\frac{\partial E(w(k))}{\partial w_{ij}} + \gamma w_{ij}(k)\right] + \alpha_{ij}^{k}\Delta w_{it}(k-1) \qquad (2.39)$$

Первое слагаемое (2.39) соответствует оригинальному алгоритму наискорейшего спуска, последнее – фактору момента, а средний член предназначен для минимизации абсолютных значений весов. Коэффициент γ, обычно имеющий малую величину (порядка 10^{-4}), приводит к уменьшению весов вплоть до разрыва соответствующих взвешенных связей. Константа η_k - это коэффициент обучения, который в данном алгоритме может иметь ненулевое значение η_0 (как правило, $0{,}01 < \eta_0 < 0{,}6$) на старте процесса обучения, когда $\Delta w_{it}(k-1) = 0$ или когда $\Delta w_{ij}\left[\dfrac{\partial E(w(k))}{\partial w_{ij}} + \gamma w_{ij}(k)\right] > 0$, либо нулевое значение – в противном случае.

Важную роль в алгоритме QuickProp играет фактор момента, который приспосабливается к текущим результатам процесса обучения. Этот коэффициент подбирается индивидуально для каждого веса по правилу:

$$\alpha_{ij}^{k} = \begin{cases} \alpha_{\max}, \text{где } \beta_{ij}(k) > \alpha_{\max}, \text{где } S_{ij}(k)\Delta w_{ij}(k-1)\beta_{ij}(k) < 0; \\ \beta_{ij}(k) \end{cases} \quad (2.40)$$

причем:

$$S_{ij}(k) = \frac{\partial E(w(k))}{\partial w_{ij}} + \gamma w_{ij}(k) \quad (2.41)$$

$$\beta_{ij}(k) = \frac{S_{ij}(k)}{S_{ij}(k-1) - S_{ij}(k)} \quad (2.42)$$

Константа $\alpha_{\max}$ - это максимальное значение коэффициента момента, которое принимается равным 1,75.

Также известна упрощенная версия алгоритма QuickProp, в которой значения весов изменяются в соответствии с правилом:

$$\Delta w_{ij}^{k} = \begin{cases} \alpha_{ij}(k)\Delta w_{ij}(k-1) \text{ для } \Delta w_{ij}(k-1) \neq 0 \\ \eta_0(k)\dfrac{\partial E}{\partial w_{ij}} \end{cases} \quad (2.43)$$

$$\alpha_{ij}(k) = \min\left(\frac{S_{ij}(k)}{S_{ij}(k) - S_{ij}(k)}, a_{\max}\right) \quad (2.44)$$

где $S_{ij}(k) = \dfrac{\partial E(w(k))}{\partial w_{ij}}$. (2.45)

В нем уменьшено количество управляющих параметров и упрощена формула уточнения значений весов.

2.6 Алгоритмы глобальной оптимизации

2.6.1 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.

2.7 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].

Определение минимального числа скрытых слоев сети основано на использовании свойств аппроксимирующих функций. Возможность такого обобщения следует из теорем А.Н. Колмогорова [1,2] и из теоремы об универсальной аппроксимации, которую можно рассматривать как естественное расширение теоремы Вейерштрасса.

Пусть *N*- число входных узлов многослойного персептрона, а *M*- число выходных нейронов сети. Тогда теорема об универсальной аппроксимации формулируется следующим образом [1]:

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 ε>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

$$F(x_1, x_2,...,x_N) = \sum_{i=1}^{K} w_i \varphi\left(\sum_{j=1}^{N} w_{ij} x_j + w_0 \right) \quad (2.46)$$

является реализацией аппроксимации функции f(.), то есть

$$\left|F(x_1, x_2,...,x_N) - f(x_1, x_2,...,x_N)\right| < \varepsilon \quad (2.47)$$

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 (2.46) описывает выходной сигнал сети следующего вида:

Сеть содержит N входных узлов и один скрытый слой, состоящий из M нейронов. Входы обозначены $x_1, x_2,...,x_N$.

Скрытый нейрон i имеет синаптические веса $w_{i1}, w_{i2},...,w_{iN}$ и w_{i0} – вес порогового элемента.

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 [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.

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) [2].

2.8 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 (2.55)[5].

$$E \le H + \varepsilon \qquad (2.48)$$

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 (2.56).

$$2\left[\frac{K}{2}\right]N \le h \le 2M(1+\lg L) \qquad (2.49)$$

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

Из формулы (2.49) 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:

1. Не следует выбирать число скрытых нейронов больше, чем удвоенное число входных элементов.
2. Число обучающих данных должно быть по крайней мере в $1/\varepsilon$ раз больше количества весов в сети, где ε - граница ошибки обучения.
3. 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.
4. 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.
5. Имеет место взаимоисключающая связь между обобщающими свойствами (меньше нейронов) и точностью (больше нейронов), которая специфична для каждого приложения.
6. Большая сеть требует большего времени для обучения.

Существуют также практические рекомендации по модификации алгоритмов конструирования сети:

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

3 Сети на основе радиально-базисных функций

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.

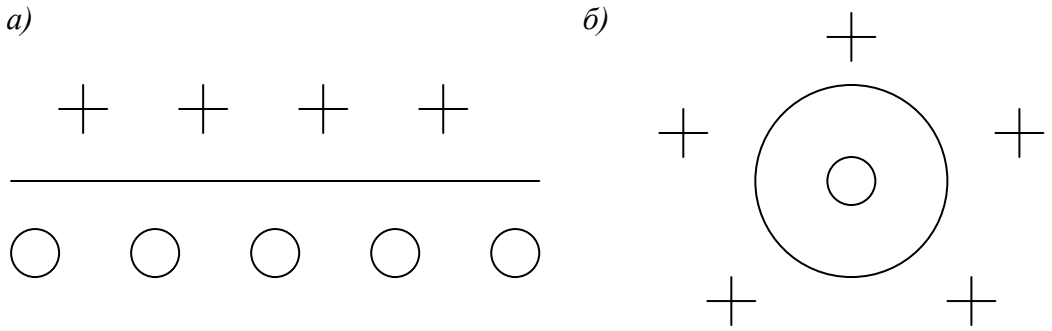


Рис. 3.1 Иллюстрация способов разделения пространства данных:
а) сигмоидальным нейроном; б) радиальным нейроном

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 [1]:

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 [1]:

1. Определение нелинейной скрытой функции $\varphi_i(x)$, где *x*– входной вектор, а *i=1,2,...,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 $\varphi(x) = [\varphi_1(x), \varphi_2(x), ..., \varphi_K(x)]^T$ в N-мер-ном входном пространстве обозначить *φ(x)*, то это пространство является нелинейно *φ*-разделяемым на два пространственных класса X^+и X^- тогда, когда существует такой вектор весов *w*, что

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

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 $(t=1,2,\ldots,p)$ из входного N-мерного пространства во множество из pрациональных чисел d_t $(t=1,2,\ldots,p)$. Для реализации этого процесса необходимо использовать pскрытых 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 pбазисных функций недопустимо, поскольку если число обучающих выборок велико и равно числу радиальных функций, то в результате вычислительная сложность обучающего алгоритма становится чрезмерной, а сама сеть адаптируется кразного 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 Kбазисными 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 $(i=1,2,\ldots,K)$ – 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 $(i=1,2,\ldots,K)$, 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.

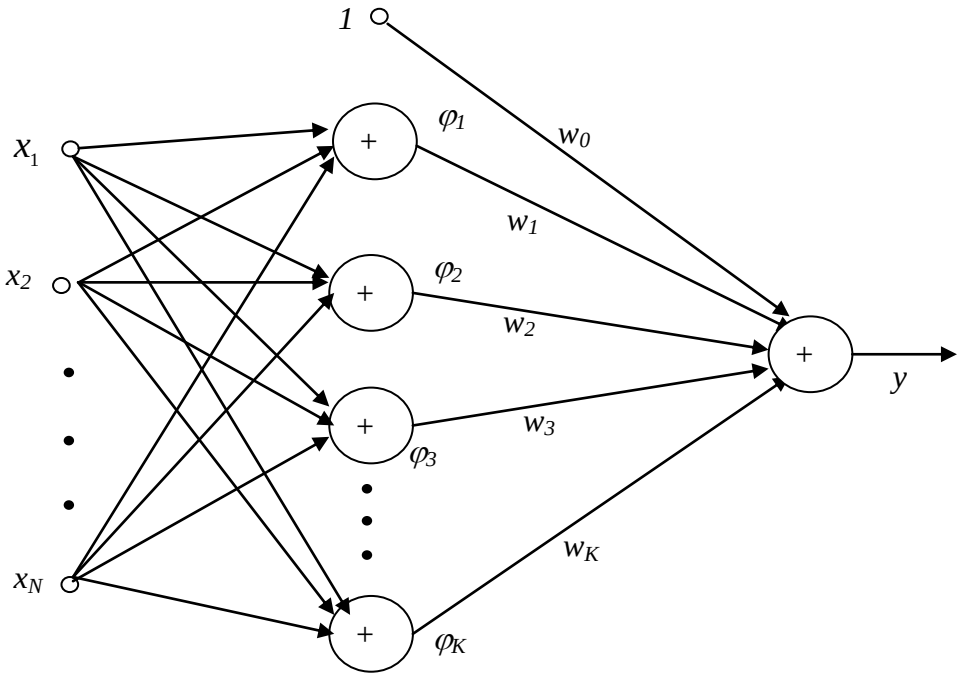


Рис. 3.2 Обобщенная структура радиальной сети RBF

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 $w^T x$.На рисунке 3.3 приведена детальная структура *RBF*-сети с радиальной функцией вида (3.5).

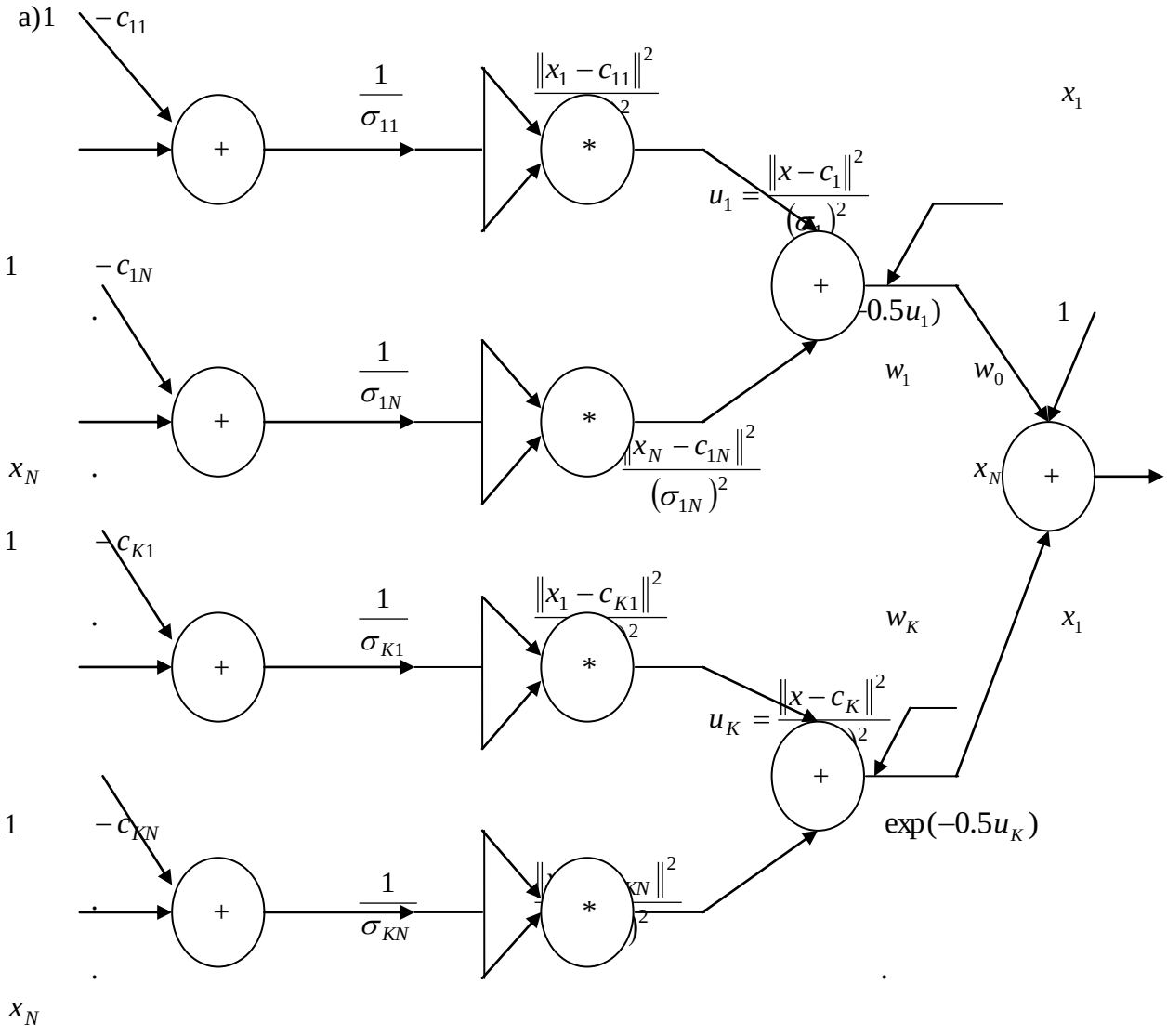


Рис. 3.3 Детальная структура *RBF*-сети.

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 (обычно используются алгоритмы обучения без учителя);
- к подбору весов нейронов выходного слоя (обычно 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 $t=1,2,\ldots,p$.

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 [2]:

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

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

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 *R* ближайшими соседями. В этом случае значение σ_i определяется по формуле (3.8):

$$\sigma_i = \sqrt{\frac{1}{R}\sum_{k=1}^{R} \| c_i - c_k \|^2}\ . \quad (3.8)$$

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 ηтоже является проблемой. Если ηимеет 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:

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

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

3.3.2. Применение метода обратного распространения ошибки для радиально-базисных 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 одного обучающегопримера имеет вид:

$$E = \frac{1}{2}\left[\sum_{i=1}^{K} w_i \varphi_i(x) - d\right]^2 \tag{3.10}$$

Предположим, что применяется гауссовская радиальная функциявида:

$$\varphi_i(x(t)) = \exp\left(-\frac{1}{2}u_i(t)\right) \tag{3.11}$$

$$u_i(t) = \sum_{j=1}^{N} \frac{(x_j(t) - c_{ij}(t))^2}{\sigma_{ij}^2(t)} \tag{3.12},$$

где i– индекс нейрона скрытого слоя, j–индекс компонента входного вектора, t – индекс обучающего примера в выборке.

Обучение сети с использованием алгоритма обратного распространения ошибки проводится в два этапа. На первом этапе предъявляется

обучающийпример и рассчитываются значения сигналов выходных нейронов сети и значение целевой функции, заданной выражением (3.10). 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. В результате дифференцирования этой функции получим:

$$\frac{\partial E(t)}{\partial w_0(t)} = y(t) - d(t) \tag{3.13}$$

$$\frac{\partial E(t)}{\partial w_i(t)} = \exp\left(-\frac{1}{2}u_i(t)\right)(y(t) - d(t)) \tag{3.14}$$

$$\frac{\partial E(t)}{\partial c_{ij}(t)} = (y(t) - d(t))w_i(t)\exp\left(-\frac{1}{2}u_i(t)\right)\frac{(x_j(t) - c_{ij}(t))}{\sigma_{ij}^2(t)} \tag{3.15}$$

$$\frac{\partial E(t)}{\partial \sigma_{ij}(t)} = (y(t) - d(t))w_i(t)\exp\left(-\frac{1}{2}u_i(t)\right)\frac{(x_j(t) - c_{ij}(t))}{\sigma_{ij}^3(t)} \tag{3.16}$$

Если в выходном слое содержится несколько нейронов, то формулы (3.13), (3.14), (3.15), (3.16) соответственно примут следующий вид:

$$\frac{\partial E(t)}{\partial w_{0s}(t)} = y_s(t) - d_s(t) \tag{3.13a}$$

$$\frac{\partial E(t)}{\partial w_{is}(t)} = \exp\left(-\frac{1}{2}u_i(t)\right)(y_s(t) - d_s(t)) \tag{3.14a}$$

$$\frac{\partial E(t)}{\partial c_{ij}(t)} = (y(t) - d(t))w_{is}(t)\exp\left(-\frac{1}{2}u_i(t)\right)\frac{(x_j(t) - c_{ij}(t))}{\sigma_{ij}^2(t)} \tag{3.15a}$$

$$\frac{\partial E(t)}{\partial \sigma_{ij}(t)} = (y(t) - d(t))w_{is}(t)\exp\left(-\frac{1}{2}u_i(t)\right)\frac{(x_j(t) - c_{ij}(t))}{\sigma_{ij}^3(t)} \tag{3.16a}$$

где s – номер нейрона выходного слоя.

При использовании метода наискорейшего спуска формулы для корректировки параметров радиально-базисной сети примут следующий вид:

$$w_i(t+1) = w_i(t) - \eta\frac{\partial E(t)}{\partial w_i(t)}, \tag{3.17}$$

$$c_{ij}(t+1) = c_{ij}(t) - \eta\frac{\partial E(t)}{\partial c_{ij}(t)}, \tag{3.18}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta\frac{\partial E(t)}{\partial \sigma_{ij}(t)}. \tag{3.19}$$

Если в выходном слое содержится несколько нейронов, то формула (3.17) примет следующий вид:

$$w_{is}(t+1) = w_{is}(t) - \eta_1 \frac{\partial E(t)}{\partial w_{is}(t)} \qquad (3.17a)$$

3.3.3 Гибридный алгоритм обучения радиальных сетей

В гибридном алгоритме процесс обучения разделяется на два этапа [2]:

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:

$$G(w)=d \qquad (3.20)$$

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

$$w=G^{+}d \qquad (3.21),$$

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 *p*x*K*, то существуют ортогональные матрицы $U = [u_1, u_2, ..., u_p]$ и $V = [v_1, v_2, ..., v_K]$ такие, что $G = USV^T$, где *S* – псевдодиагональная матрица размера *p*x*K*, *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:

$$G \cong U_r S_r V_r^T . \qquad (3.22)$$

Псевдообратная матрица для матрицы *G* находится по формуле:

$$G^{+} = U_r S_r^{-1} V_r^T \quad , \qquad (3.23)$$

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

$$w = U_r S_r^{-1} V_r^T d. \quad (3.24)$$

На втором этапе при зафиксированных значениях выходных весов рассчитываетсяреальные выходы сети и величина погрешности для последовательности векторов x_t, $t=1,2,..,p$. По величине погрешности определяется вектор градиента целевой функции относительно конкретных центров c_{ij} и ширины σ_{ij} (по методу обратного распространения ошибки).

Каждая радиальная функция определяется в общем виде как

$$\varphi_i(x_t) = \exp\left(-\frac{1}{2}u_{it}\right), \quad (3.25)$$

где суммарный сигнал нейрона u_{it} описывается выражением

$$u_{it} = \sum_{j=1}^{N} \frac{(x_{jt} - c_{ij})^2}{\sigma_{ij}^2}. \quad (3.26)$$

При существовании p обучающих пар целевую функцию можно задать в виде:

$$E = \frac{1}{2}\sum_{t=1}^{P}(y_t - d_t)^2 = \frac{1}{2}\sum_{t=1}^{P}\left[\sum_{i=1}^{K} w_i \varphi(x_k) - d_k\right]^2 \quad (3.27)$$

В результате дифференцирования этой функции получим:

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

$$\frac{\partial E}{\partial \sigma_{ij}} = \sum_{t=1}^{p}\left[(y_t - d_t) w_i \exp\left(-\frac{1}{2}u_{it}\right)\frac{(x_{jt} - c_{ij})^2}{\sigma_{ij}^3}\right]. \quad (3.28)$$

Применение градиентного метода наискорейшего спуска позволяет провести уточнение центров и ширины радиальных функций согласно формулам (3.29) и (3.30):

$$c_{ij}(t+1) = c_{ij}(t) - \eta\frac{\partial E}{\partial c_{ij}}, \quad (3.29)$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta\frac{\partial E}{\partial \sigma_{ij}}. \quad (3.30)$$

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.

3.4 Структура гипер радиально-базисной сети

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*.

$$\|x\|_Q^2 = (Qx)^T (Qx) = x^T Q^T Qx \tag{3.31}$$

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

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \dots & Q_{1N} \\ Q_{21} & Q_{22} \dots & Q_{2N} \\ \dots & \dots & \dots & \dots \\ Q_{N1} & Q_{N2} \dots & Q_{NN} \end{bmatrix} \tag{3.32}$$

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

$$\|x\|_Q^2 = \sum_{j=1}^{N} \sum_{r=1}^{N} C_{jr} x_j x_r \tag{3.33}$$

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:

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

(3.34),

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 *xi*-го нейрона. На рис. 3.4 представлена детальная структура сети *HRBF*с

произвольнойматрицей Q. В сетях *HRBF* роль коэффициентов σ_i^2 выполняют элементы матрицы *Q*, которые уточняются в процессе обучения.

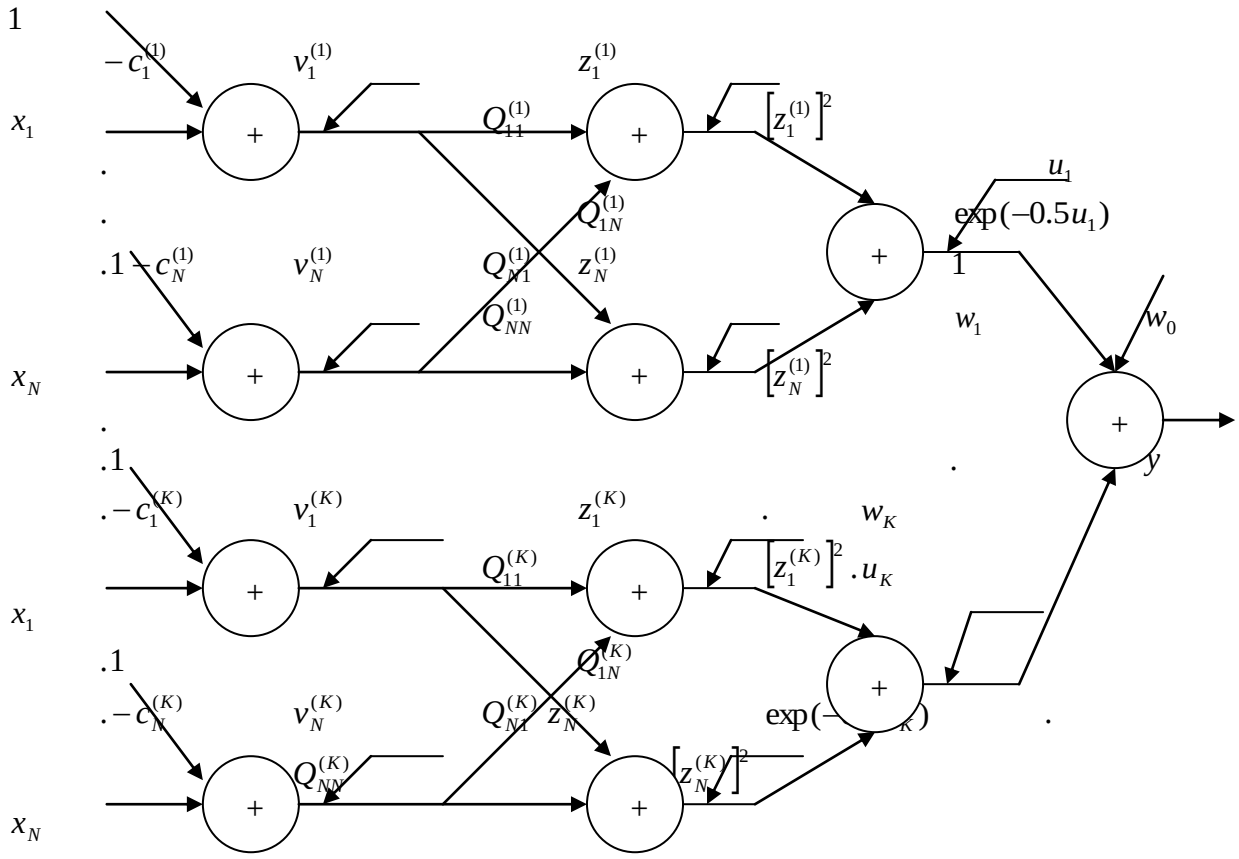


Рис. 3.4Детальная структура радиальной сети *HRBF* с произвольной масшабирующей матрицей *Q*

3.5 Основные алгоритмы обучения гипер радиально-базисных сетей

3.5.1 Применение метода обратного распространения ошибки для гипер радиально-базисных 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 одного обучающегопримера имеет вид:

$$E = \frac{1}{2}\left[\sum_{i=1}^{K} w_i \varphi_i(x) - d\right]^2 \qquad (3.35)$$

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

$$\varphi(x) = \exp\left[-\frac{1}{2}\left[Q_i(x-c_i)\right]^T\left[Q_i(x-c_i)\right]\right], \qquad (3.36)$$

а матрица Q_i имеет произвольную структуру.

Обучение сети с использованием алгоритма обратного распространения ошибки проводится в два этапа. На первом этапе предъявляется обучающий вектор и рассчитываются значения сигналов выходных нейронов сети и фактическое значение целевой функции, заданной выражением (3.35). 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 расчета градиента целевой функции по параметрам сети целесообразно использовать метод потоковых графов, описанный в [2]. Граф сети *HRBF* представлен на рис.3.4. Сопряжённая сеть, используемая для расчёта градиента, представлена на рис.3.5.

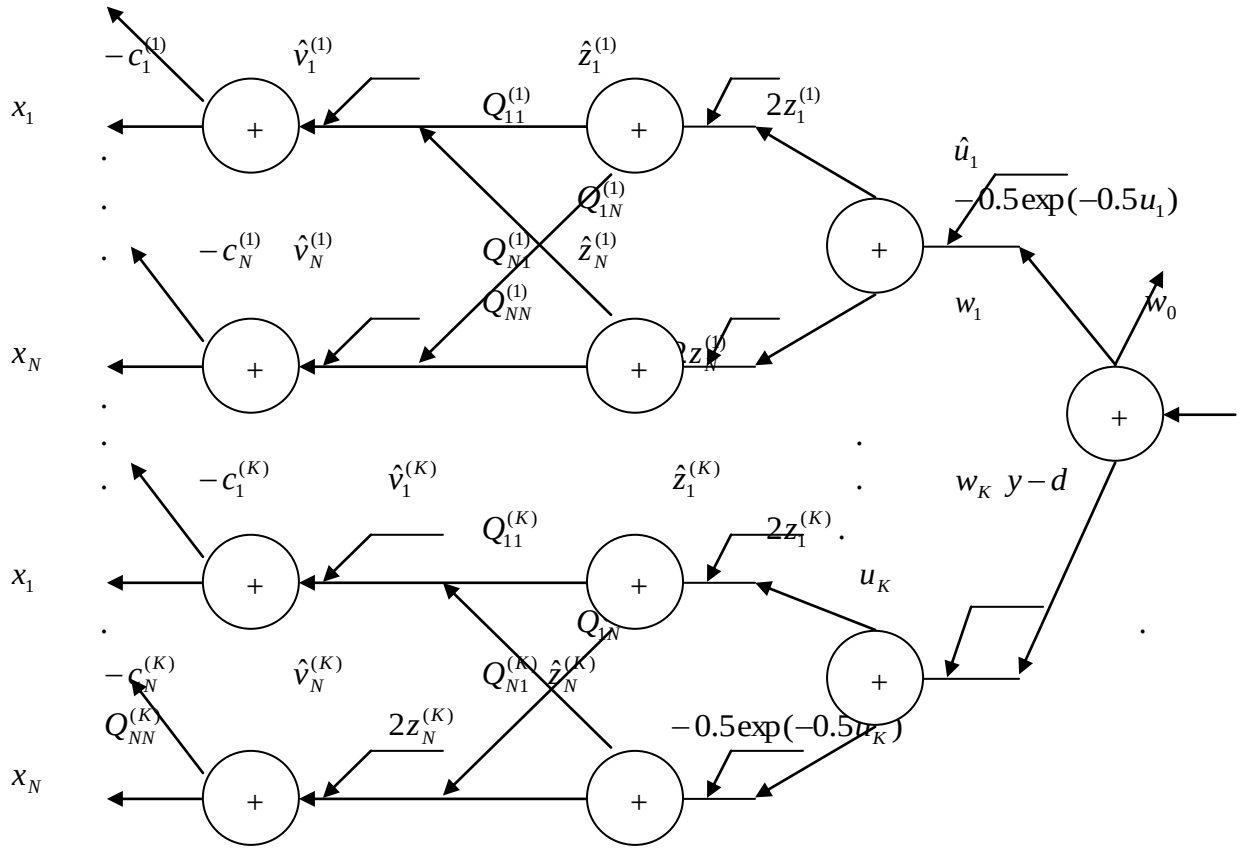


Рис. 3.5. Структура сопряженнойHRBF-сети, используемая для расчета градиента

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

$$\frac{\partial E}{\partial w_0} = y - d \tag{3.38}$$

$$\frac{\partial E}{\partial w_i} = \exp\left(-\frac{1}{2}u_i\right)(y - d) \tag{3.39}$$

$$\frac{\partial E}{\partial c_j^{(i)}} = \hat{v}_j^{(i)} = -\exp\left(-\frac{1}{2}u_i\right) w_i (y-d) \sum_{j=1}^{N} Q_{jr}^{(i)} z_j^{(i)} \qquad (3.40)$$

$$\frac{\partial E}{\partial Q_{jr}^{(i)}} = v_t^{(i)} \hat{z}_t^{(i)} = -\exp\left(-\frac{1}{2}u_i\right) w_i \left(y-d\right)\left(x_j - c_j^{(i)}\right) z_r^{(i)} \qquad (3.41)$$

где $z_r^{(i)} = \sum_{j=1}^{N} Q_{jr}^{(i)} (x_j - c_j^{(i)})$ (3.42)

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

$$u_i = \sum_{j=1}^{N} \left[z_j^{(i)}\right]^2 \qquad (3.43)$$

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*.

3.6 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. На практике, часто используют два простых алгоритма определения оптимального числа нейронов в скрытом слое. В первом из них происходит увеличение числа нейронов в скрытом слое в процессе обучения сети, начиная с одного.Процесс продолжается до тех пор, пока погрешность обучения не перестанет уменьшаться. Во втором, происходит уменьшение числа нейронов в скрытом слое в процессе обучения сети, начиная с заведомого большого числа нейронов. Процесс также продолжается до тех пор, пока погрешность обучения не перестанет уменьшаться.

Однако, существуют и другие подходы для определения оптимального числа нейронов скрытого слоя.

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 [2].

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

$\int_{\Re^N} G(x)dx \neq 0.$ (3.44)

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 [2].

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)

$$G = \begin{bmatrix} \varphi_{11} & \varphi_{21} & \cdots & \varphi_{K1} \\ \varphi_{12} & \varphi_{22} & \cdots & \varphi_{K2} \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_{1p} & \varphi_{2p} & \cdots & \varphi_{Kp} \end{bmatrix}, \qquad (3.45)$$

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

$$d = Gw + e, \qquad (3.46)$$

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:

$$G = QA, \tag{3.47}$$

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 $Q^T Q = H$.

При этом H – диагональная матрица с элементами $H_{ii} = q_i^T q_i = \sum_{t=1}^{p} q_{it}^2$.

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

$$b = Aw, \tag{3.48}$$

то из уравнения (3.35) получим:

$$d = QB + e. \tag{3.49}$$

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

$$\hat{b} = [Q^T Q]^{-1} Q^T d = H^{-1} Q^T d. \tag{3.50}$$

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

$$\hat{b}_i = \frac{q_i^T d}{q_i^T q_i}. \tag{3.51}$$

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

3.7 Метод ортогонализации Грэма-Шмидта

Ортогонализация матрицы Q, описанная выражением (3.47), может быть проведена различными методами, наиболее эффективным из которых является алгоритм Грэма-Шмидта. В соответствии с этим методом матрицаAформируется 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:

$$q_1 = g_1, \tag{3.52}$$

$$a_{ir} = \frac{q_i^T g_r}{q_i^T q_i}, \tag{3.53}$$

$$q_r = g_r - \sum_{i=1}^{r-1} a_{ir} q_i, \tag{3.54}$$

для $1 \le i \le r$, $r=2,3,\ldots,K$. 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

$$q_i(1) = g_i, \tag{3.55}$$

$$b_i(1) = \frac{[q_i(1)]^T d}{[q_i(1)]^T q_i(1)}, \qquad \varepsilon_i(1) = \frac{[b_i(1)]^2 [q_i(1)]^T q_i(1)}{d^T d}. \tag{3.56}$$

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:

$$a_{ir}^{(i)} = \frac{q_i^T g_r}{q_i^T q_i}, \tag{3.57}$$

$$q_i(r) = g_i - \sum_{i=1}^{r-1} a_{ir}^{(i)} q_i, \tag{3.58}$$

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

$$b_i(r) = \frac{[q_i(r)]^T d}{[q_i(r)]^T [q_i(r)]}, \qquad \varepsilon_i(r) = \frac{[b_i(r)]^T [q_i(r)]^T [q_i(r)]}{d^T d}. \tag{3.59}$$

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:

$$q_r = q_{i_r}(r) = q_{i_r} - \sum_{i=1}^{r-1} a_{ir} q_i \,, \tag{3.60}$$

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

$$1 - \sum_{i=1}^{K_r} \varepsilon_i < \rho \,, \tag{3.61}$$

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.

3.8 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.

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 [2].

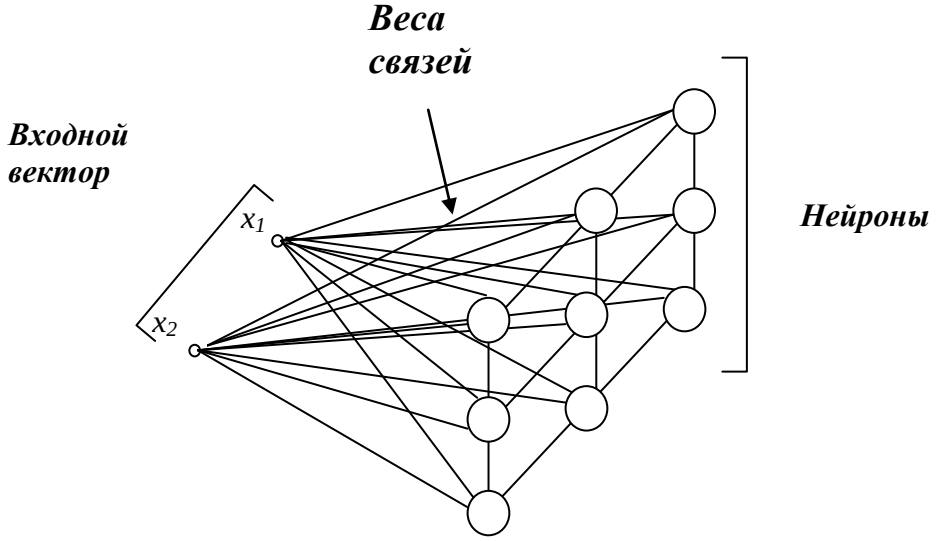


Рис. 4.1 Структура самоорганизующейся сети Кохонена.

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 [2]:

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 $w_1 = w_2$ выполняетсятогда 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 $\min\|x - w\|^2 = \max\left(x^T w_i\right) \text{при} \|w_i\| = const$. Экспериментальные исследования подтвердили необходимость применения нормализации векторов при малой размерности пространства [2]. 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 [2]. 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):

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

Значение коэффициента $p_{\min}$ определяет минимальный потенциал, разрешающий участие в конкурентной борьбе. Если фактическое значение потенциала p_i падает ниже $p_{\min}$, то *i-й* нейрон «отдыхает», а победитель ищется среди нейронов, для которых выполняется соотношение

$$d(x, w_w) = \min\{d(x, w_i)\}, для 1 \le i \le K \; и \; p_i \ge p_{\min} \qquad (4.11)$$

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 $p_{\min} \approx 0{,}74$.

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

$$d(x, w_i) \leftarrow N_i d(x, w_i). \qquad (4.12)$$

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:

$$E_q = \frac{1}{2}\sum_{t=1}^{p}\left\| x_i - w_{w(t)} \right\|^2, \qquad (4.13)$$

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(WinnerTakesAll* – 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

$$w_w(t+1) = w_w(t) + \eta(x - w_w(t)) \qquad (4.14)$$

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* (*WinnerTakesMost* – 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

$$w_i(t+1) = w_i(t) + \eta_i S(i,x)(x - w_i(t)) \qquad (4.15)$$

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 $S(i,x)$. Если $S(i,x)$ определяется в форме

$$S(i,x)=\begin{cases}1, \textit{для } i=w\\ 0, \textit{для } i\neq w\end{cases}, \tag{4.16}$$

для wобозначает номер победителя, то мы получаем классический алгоритм *WTA*. Существует множество вариантов алгоритма *WTM*, отличающихся ,прежде всего формой функции $S(i,x)$. 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.

$$S(i,x)=\begin{cases}1, \textit{для } -K<d(i,w)<K\\ 0, \textit{иначе}\end{cases}, \tag{4.17}$$

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 $S(i,x)$ определяется формулой

$$S(i,x)=\exp\left(-\frac{d^2(i,w)}{2\lambda^2}\right). \tag{4.18}$$

Уточнение весов нейронов происходит по правилу:

$$w_i(t)=w_i(t-1)+\eta(t)\cdot S(i,w)\cdot(x-w_i). \tag{4.19}$$

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

$$d_0 < d_1 < d_2 < \ldots < d_{K-1}, \tag{4.20}$$

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 $S(i,x)$ определяется по формуле

$$S(i,x) = \exp\left(-\frac{m(i)}{\lambda}\right), \tag{4.21}$$

вкоторой $m(i)$ 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 $S(i,x)$. 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

$$\lambda(t) = \lambda_{\max}\left(\frac{\lambda_{\min}}{\lambda_{\max}}\right)^{t/t_{\max}}, \tag{4.22}$$

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

$$\eta_i(t) = \eta_i(0)\left(\frac{\eta_{\min}}{\eta_i(0)}\right)^{t/t_{\max}}, \tag{4.23}$$

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 $S(i,x)$. При этом используется зависимость (4.20), в соответствии с которой если $m(i) >> 1$, то значение $S(i,x) \cong 0$.

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.

5 Гибридные сети

5.1 Сетьвстречного распространения

5.1.1 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 встречного распространения [3].

На рис. 5.1 показана упрощенная версия прямого действия сети встречного 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 слоя (2), называемого слоем Гроссберга, весом v_{MK}. Эти веса образуют матрицу весов *v*.

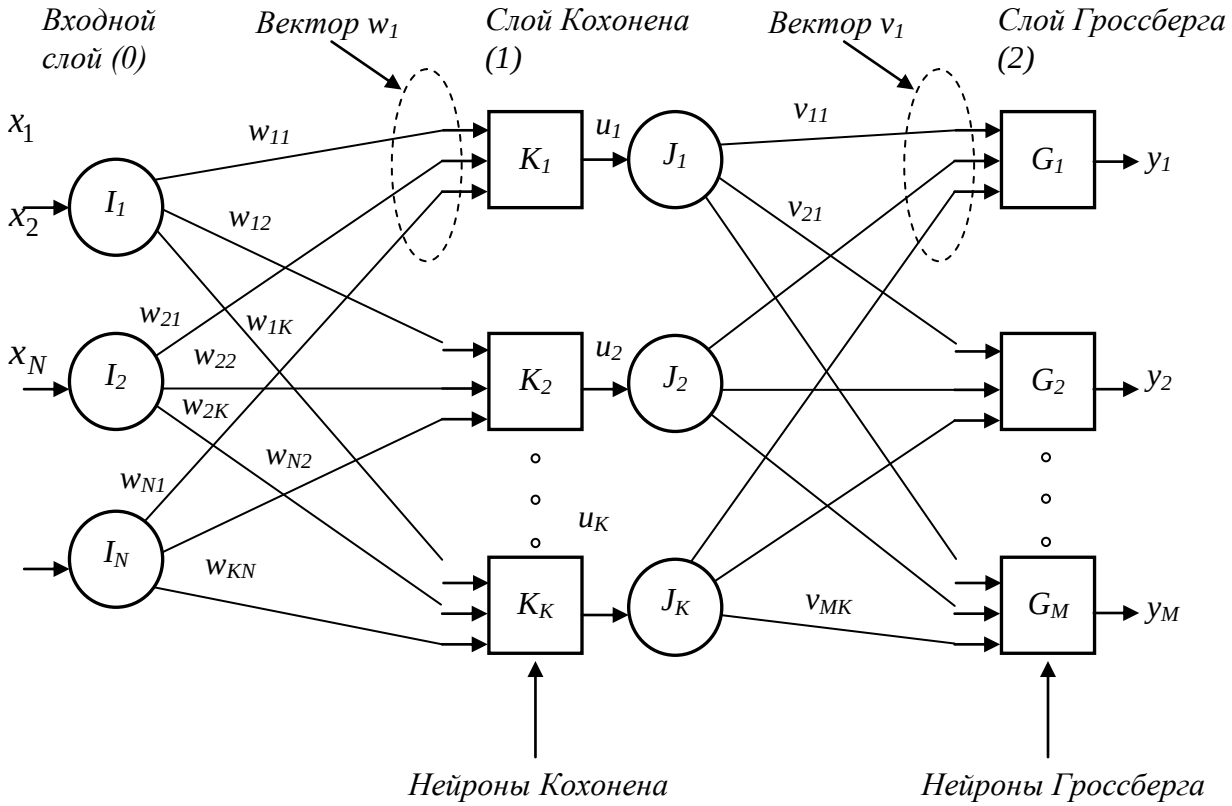


Рисунок5.1 Сеть встречного распространения без обратных связей

Сети встречного 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.

5.1.2. Нормальное функционированиесети встречного 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:

$$u_i = w_{i1} \cdot x_1 + w_{i2} \cdot x_2 + \ldots + w_{iN} \cdot x_N , \qquad (5.1)$$

где u_i – это выход *ui* –го нейрона Кохонена,

$$u_i = \sum_{j=1}^{N} x_j \cdot w_{ij} \qquad (5.2)$$

или в векторной записи

$$u = x \cdot w. \qquad (5.3)$$

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

$$y_s = \sum_{i=1}^{M} u_{s_i} \cdot v_{si} , \qquad (5.4)$$

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

$$y = u \cdot v, \qquad (5.5)$$

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.

5.1.3 Структура полной сети встречного распространения

На рис. 5.2 показана структура полной сети встречного 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 X_1 и X_2 соответственно. 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.

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 результата 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 встречного распространения быстро обучается и при правильном использовании может сэкономить значительное количество машинного времени. Она полезна также для быстрого моделирования систем, где большая точность встречного 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.

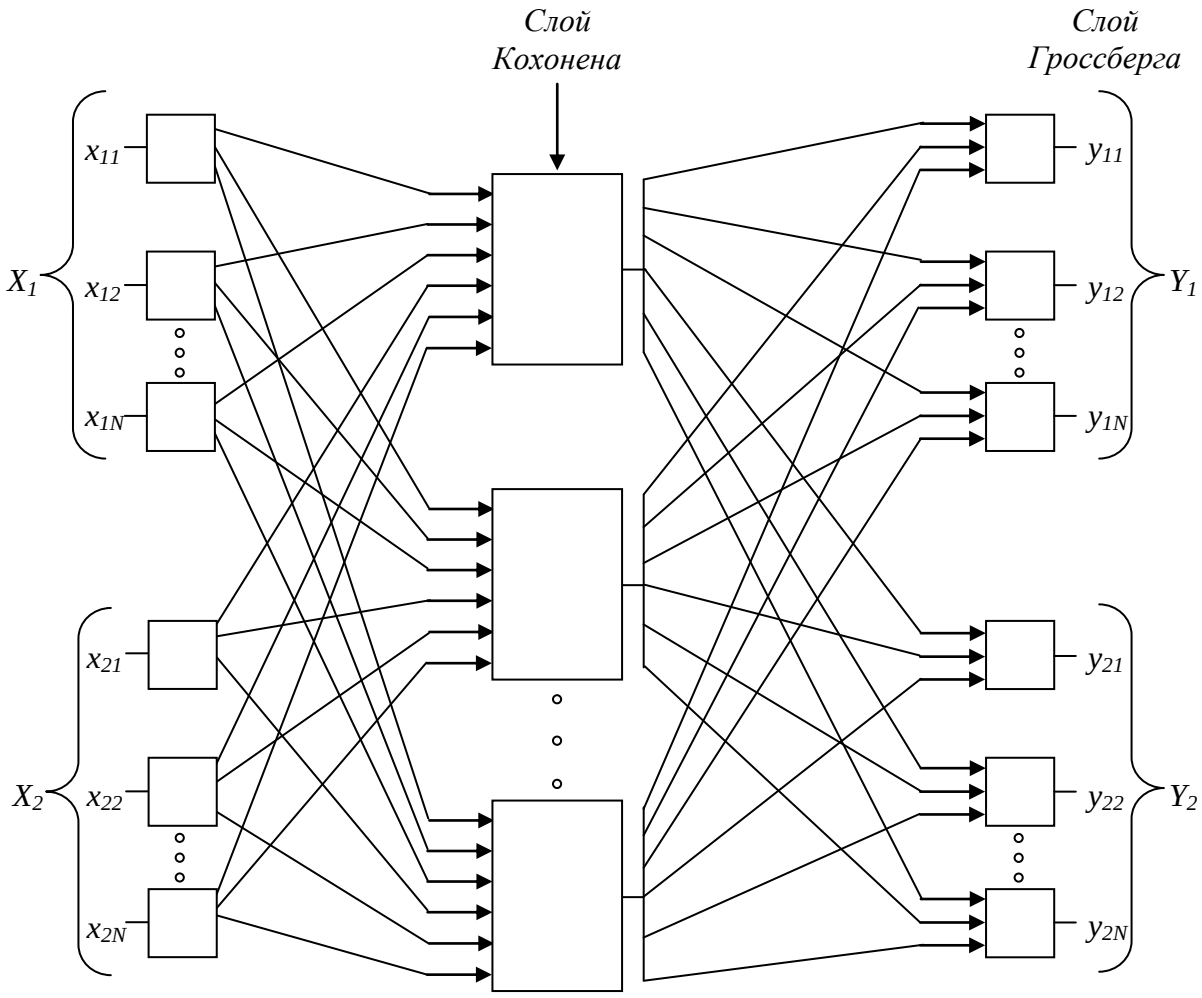


Рисунок5.2 Общая структура гибридной сети встречного распространения

5.1.4 Анализ методов обучения сети встречного 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:

$$v_{si}(t+1) = v_{si}(t) + \beta \cdot (y_s - v_{si}(t)) \cdot u_i, \qquad (5.6)$$

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 встречного 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 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 проблему мертвых нейронов. Более того, оставшихся весов, дающих наилучшие соответствия, может оказаться слишком мало, для того, 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:

$$w_i = \frac{1}{\sqrt{N}}, \qquad (5.7)$$

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

$$x_j = \alpha \cdot x_j + \frac{1-\alpha}{\sqrt{N}}, \qquad (5.8)$$

где N – число входов. В начале αочень 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.

5.2 Гибридная сеть Кохонена

5.2.1 Структура сети

Главной особенностью сетей с самоорганизацией на основе конкуренции является очень высокая скорость обучения, много больше, чем у сетей, обучающихся с учителем. Так как сеть с самоорганизацией выполняет обработку только входного вектора x, то основным недостатком такой сети является отсутствие аппроксимирующих свойств, которыми обладают многослойный персептрон и радиально-базисные сети.

Очень хороший результат достигается при объединении самоорганизующегося слоя Кохонена и многослойного персептрона. Подобная структура, называемая гибридной сетью Кохонена, приведена на рисунке 5.3 [2].

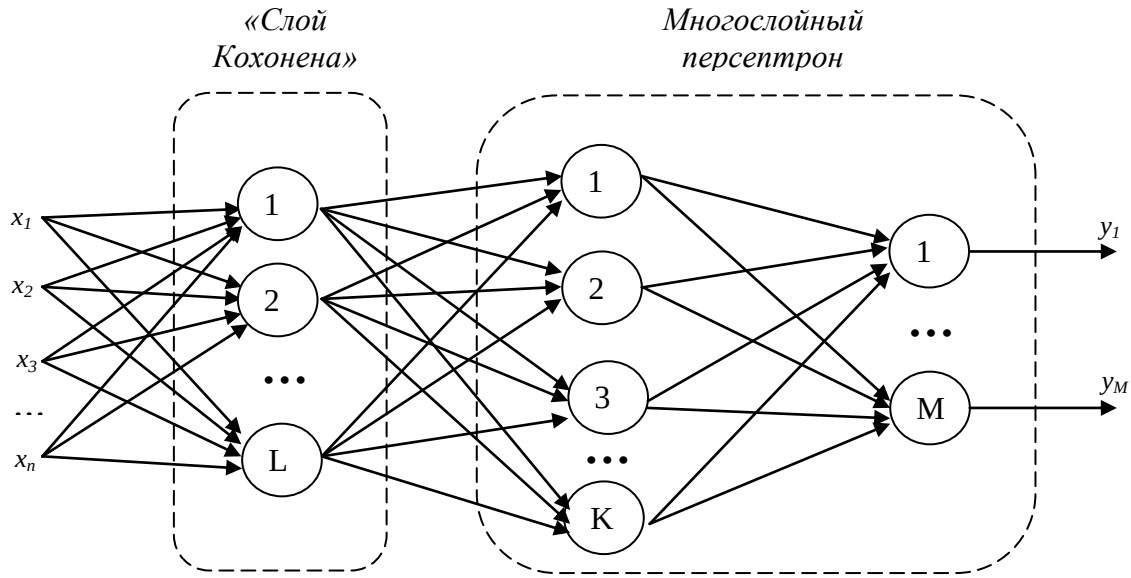


Рисунок5.3 Общая структура гибридной сети Кохонена

Слой Кохонена локализует значимые признаки процесса на основе анализа входного вектора x, после чего формируется входной вектор персептронного слоя. Вследствие хорошей локализации признаков первым слоем, обычно достаточно использовать персептронную сеть с одним скрытым слоем нейронов (зачастую линейных).

5.2.1 Обучение сети

Обучение гибридной сети состоит из двух последовательных этапов. Сначала обучается слой Кохонена. В результате обучения векторы весов нейронов данного слоя с минимальной погрешностью отображают распределение данных обучающих векторов x. Обучение слоя Кохонена

проводится по одному из алгоритмов, описанных в главе 4. После завершения обучения, значения весов нейронов слоя Кохонена замораживаются, и производится анализ выходных сигналов. Выход нейрона-победителя переводится в состояние 1, а выходы остальных нейронов переводятся в состояния из интервала (0-1). Переход от фактических выходных сигналов к нормализованным сигналам может производиться по различным формулам. Хорошие результаты можно получить, используя формулу:

$$y_r = \exp\left(-\frac{|u_r - u_{\max}|^2}{\sigma^2}\right), \qquad (5.9)$$

где u_r - реальный выход r-го нейрона слоя Кохонена, y_r - нормализованный выход r-го нейрона слоя Кохонена, $u_{\max}$- выход нейрона-победителя, σ - параметр, подбираемый для конкретной задачи.

Затем обучается персептронная сеть по одному из алгоритмов обучения с учителем. Обучающими сигналами является множество пар (y_t, d_t), где y_t - нормализованный выходной вектор слоя Кохонена, а d_t - это вектор эталонных значений выходных нейронов персептрона, соответствующих входному вектору x_t, $t = 1,2,\ldots,p$.

Чаще всего используются градиентные алгоритмы и метод обратного распространения ошибки. Процесс обучения будет протекать гораздо быстрее, чем для обычного многослойного персептрона, благодаря хорошей локализации данных в слое Кохонена. Кроме того, при этом, как правило, достигается глобальный минимум функции погрешности.

Следует отметить, что данная сеть считается обобщением сети встречного распространения, в отличии от которой в сети Кохонена допускается дробная активность нейронов от значения 1 для победителя и от 0 до 1 для остальных нейронов. Учёт активности многих нейронов позволяет лучше локализовать входной вектор x в многомерном пространстве и получить лучшее отображение данных нейронной сетью в целом.

6 Рекуррентные сети

6.1 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 векторов 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 [2]:

$$d_H(y,d)=\sum_{i=1}^{N}[d_i(1-y_i)+(1-d_i)y_i]. \qquad (6.1)$$

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

$$d_H(y,d) = \frac{1}{2}\left[N - \sum_{i=1}^{N} y_i d_i \right]. \qquad (6.2)$$

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

6.2 Сеть Хопфилда

Обобщенная структура сети Хопфилда представляется, как правило, в виде системы с непосредственной обратной связью выхода с входом (рис. 6.1). Характерная особенность такой системы состоит в том, что выходные сигналы нейронов являются одновременно входными сигналами сети:

$$x_i(t) = y_i(t-1) \qquad (6.3)$$

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.

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.

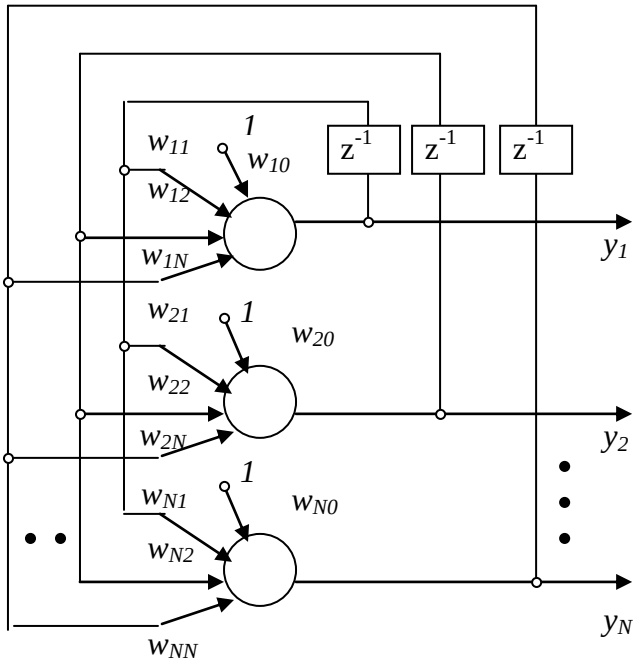


Рисунок6.1 Обобщенная структура сети Хопфилда

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

$$y_i = \mathrm{sgn}\left(\sum_{j=1}^{N} w_{ij} x_j + b_i\right), \tag{6.4}$$

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:

$$y_i(t) = \mathrm{sgn}\left(\sum_{j=0, i\neq j}^{N} w_{ij} y_j(t-1)\right), \tag{6.5}$$

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 (6.5) 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 (6.5) не начнет соблюдаться для всех N нейронов. Это условие автоматически выполняется в случае выбора значений весов, соответствующих отношению

$$w_{ij} = \frac{1}{N} x_i x_j, \tag{6.6}$$

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 (6.6) 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

$$w_{ij} = \frac{1}{N} \sum_{t=1}^{p} x_i^{(t)} x_j^{(t)}. \tag{6.7}$$

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 (6.7) получим

$$y_i^{(l)} = \mathrm{sgn}\left(\sum_{j=0}^{N} w_{ij} x_j^{(l)}\right) = \mathrm{sgn}\left(\frac{1}{N}\sum_{j=0}^{N}\sum_{t=1}^{p} x_i^{(t)} x_j^{(t)} x_j^{(l)}\right) = x_i^{(l)}. \qquad (6.8)$$

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 [2]:

$$u_i^{(l)} = x_i^{(l)} + \frac{1}{N}\sum_{j=0}^{N}\sum_{t\neq l} x_i^{(t)} x_j^{(t)} x_j^{(l)}. \qquad (6.9)$$

Вследствие применения знаковой функции активации, выполнение условия (6.8) 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 (6.5)) [2]. В матричной форме это можно представить в виде:

$$WX = X, \qquad (6.10)$$

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 $X = \left[x^{(1)}, x^{(2)}, \ldots, x^{(p)}\right]$. Решение такой линейной системы уравнений имеет вид:

$$W = X\ X^{+}, \qquad (6.11)$$

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

$$W = X\left(X^T X\right)^{-1} X^T. \qquad (6.12)$$

Псевдоинверсия матрицы размерностью $N \times p$ в этом выражении заменена обычной инверсией квадратной матрицы $X^T X$ размерностью $p \times p$. Дополнительное достоинство выражения (6.12) – возможность записать его в итерационной форме, не требующей расчета обратной

матрицы. В этом случае выражение (6.12) принимает вид функциональной зависимости от последовательности обучающих векторов $x^{(t)}$ для $t = 1, 2, \ldots, p$:

$$W^{(t)} = W^{(t-1)} + \frac{1}{\left[x^{(t)}\right]^T x^{(t)} - \left[x^{(t)}\right]^T W^{(t-1)} x^{(t)}} \times \left[W^{(t-1)} x^{(t)} - x^{(t)}\right] \times \left[W^{(t-1)} x^{(t)} - x^{(t)}\right]^T \tag{6.13}$$

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 (6.12) и (6.13) 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:

$$W \leftarrow W + \frac{\eta}{N}\left[x^{(t)} - W x^{(t)}\right]\left[x^{(t)}\right]^T. \tag{6.14}$$

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:

$$y(t) = \mathrm{sgn}(W y(t-1)) \tag{6.15}$$

(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.

6.3 Сеть Хемминга

Сеть Хемминга – это трехслойная рекуррентная структура, которую можно считать развитием сети Хопфилда, была предложена Р. Липпманом. 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 рисунке6.2. [2].

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, в котором только один нейрон имеет выходное значение, равное 1, а все остальные – равные 0.

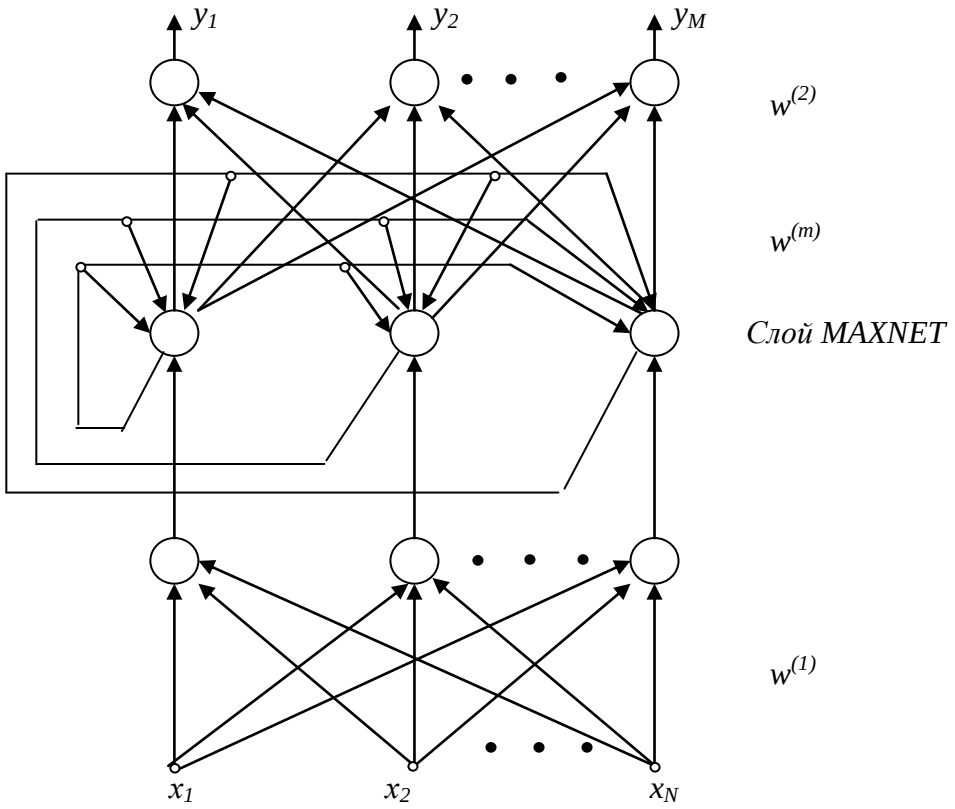


Рисунок6.2 Обобщенная структура сети Хемминга

Веса первого слоя соответствуют входным векторам-образцам $x^{(t)}$, поэтому $w_{ij}^{(1)} = x_j^{(t)}$ для $t = 1,2,...,p$, $i=t$, $j = 1,2,...,N$, то есть веса первого нейрона запоминают компоненты первого входного вектора. Веса второго нейрона – компоненты второго вектора и т. д.

Аналогично веса выходного слоя соответствуют очередным векторам образов $y^{(t)}$, $w_{li}^{(2)} = y_l^{(t)}$, *i,t,l=1,2,...p*.

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

$$-\frac{1}{p-1} < w_{is}^{(m)} < 0 \qquad (6.16)$$

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

$$w_{is}^{(m)} = -\frac{1}{p-1} + \xi, \qquad (6.17)$$

где ξ - случайная величина с достаточно малой амплитудой.

В процессе функционирования сети в режиме распознавания 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. 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:

$$\hat{y}_i = 1 - \frac{d_H\left(x^{(t)}, x\right)}{N}, \qquad (6.18)$$

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.

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 не равным 0, перейдут в нулевое состояние.

Задача нейронов этого слоя состоит в определении победителя, то есть нейрона, у которого выходной сигнал отличен от 0. Процесс определения победителя выполняется согласно формуле:

$$y_l(t) = f\left(\sum_s w_{is}^{(m)} y_s(t-1)\right) = f\left(y_l(t-1) + \sum_{s \neq i} w_{is}^{(m)} y_s(t-1)\right), \qquad (6.19)$$

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

$$f(y)=\begin{cases} y \text{ для } y \geq 0 \\ 0 \text{ для } y < 0 \end{cases}. \qquad (6.20)$$

Итерационный процесс (6.19) 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 $w_{is}^{(2)}$*(s= 1,2,..,N)*линейных нейронов выходного слоя представляет вектор $y^{(t)}$, который соответствует номеру вектора $x^{(t)}$, признанному слоем MAXNET в качестве ближайшего к входному вектору x.

В третьей фазе этот нейрон посредством весов, связывающих его с нейронами выходного слоя, формирует на выходе сигнал, равный 1, его номер является номер входного образца, к которому принадлежит входной вектор.

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 классу входного вектора. При p нейронах первого слоя, емкость запоминающего устройства Хемминга также равна p, так как каждый нейрон представляет единственный класс.

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 [2].

6.4 Рекуррентная сеть Эльмана

Данная рекуррентная сеть представляет собой развитие сетей персептронного типа за счет добавления в них обратных связей. Сеть Эльмана является одним из представителей типа сетей, названных рекуррентными многослойными персептронами (RMLP)[1].

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 рисунке6.4.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:

$$x(t) = [x_0(t), x_1(t), ..., x_N(t), \nu_1(t-1), \nu_2(t-1), ..., \nu_K(t-1)] \quad (6.21)$$

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

$$u_i(t) = \sum_{j=0}^{N+K} w_{ij}^{(1)} x_j(t); \quad (6.22)$$

$$\nu_i(t) = f_1(u_i(t)); \quad (6.23)$$

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:

$$g_s(t) = \sum_{i=0}^{K} w_{si}^{(2)} \nu_s(t); \quad (6.24)$$

$$y_s(t) = f_2(g_s(t)). \quad (6.25)$$

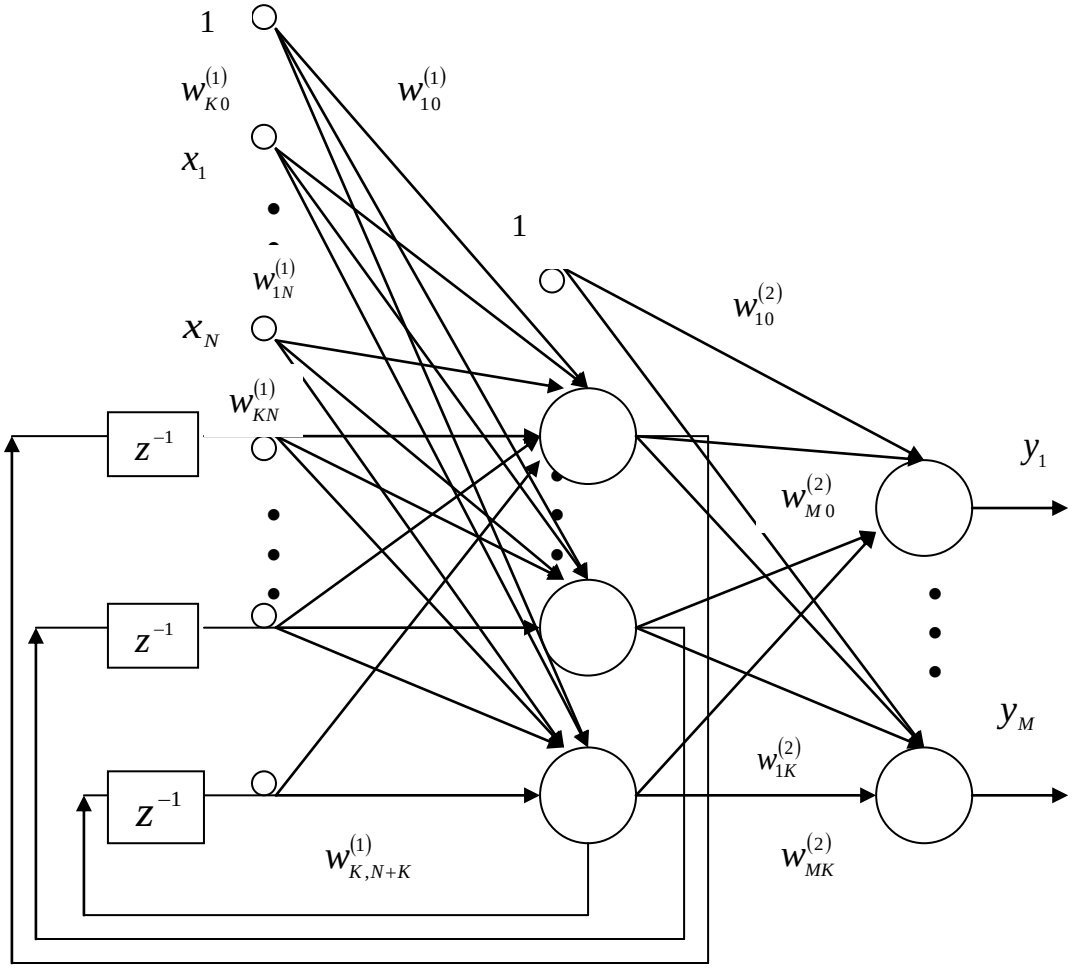


Рисунок6.3 Обобщенная структура сети Эльмана

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.

6.5 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:

$$E(t)=\frac{1}{2}\sum_{s=1}^{M}\left[y_s(t)-d_s(t)\right]^2=\frac{1}{2}\sum_{s=1}^{M}e_s(t)^2 \qquad (6.26)$$

При дифференцировании целевой функции относительно весов выходного слоя получаем:

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

$$\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\sum_{i=0}^{K}\left(\frac{dv_i}{dw_{\alpha\beta}^{(2)}}w_{si}^{(2)}+\frac{dw_{si}^{(2)}}{dw_{\alpha\beta}^{(2)}}v_i(t)\right) \qquad (6.27)$$

Связи между скрытым и выходным слоем однонаправленные, поэтому:

$$\frac{dv_i}{dw_{\alpha\beta}^{(2)}}=0 \qquad (6.28)$$

С учетом этого факта получим:

$$\nabla_{\alpha\beta}^{(2)}E(t)=\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\sum_{i=0}^{K}\frac{dw_{si}^{(2)}}{dw_{\alpha\beta}^{(2)}}v_i(t)=e_\alpha(t)\frac{df_2(g_\alpha(t))}{dg_\alpha(t)}v_\beta(t) \qquad (6.29)$$

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

$$w_{\alpha\beta}^{(2)}(t+1)=w_{\alpha\beta}^{(2)}(t)+\Delta w_{\alpha\beta}^{(2)}, \qquad (6.30)$$

где $\Delta w_{\alpha\beta}^{(2)}=-\eta\nabla_{\alpha\beta}^{(2)}E(t)$ (6.31)

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:

$$\nabla^{(1)}_{\alpha\beta}E(t)=\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\sum_{i=1}^{K}\frac{d(\nu_i w^{(2)}_{si})}{dw^{(1)}_{\alpha\beta}}=\sum_{s=1}^{M}e_s(t)\frac{df_2(g_s(t))}{dg_s(t)}\sum_{i=1}^{K}\frac{d\nu_i(t)}{dw^{(1)}_{\alpha\beta}}w^{(2)}_{si} \tag{6.32}$$

$$\frac{d\nu_i(t)}{dw^{(1)}_{\alpha\beta}}=\frac{df_1(u_i)}{du_i}\sum_{k=0}^{N+K}\frac{d(x_k(t)w^{(1)}_{ik})}{dw^{(1)}_{\alpha\beta}}=\frac{df_1(u_i)}{du_i}\left[\delta_{i\alpha}x_\beta+\sum_{k=N+1}^{N+K}\frac{dx_k(t)}{dw^{(1)}_{\alpha\beta}}w^{(1)}_{ik}\right] \tag{6.33}$$

где $\delta_{i\alpha}$ - дельта Кронекера, то есть:

$$\delta_{i\alpha}=\begin{cases}1, i=\alpha\\0, i\neq\alpha\end{cases}.$$

Из определения входного вектора x (формула (6.21)) в момент времени t следует выражение (6.34):

$$\frac{d\nu_i(t)}{dw^{(1)}_{\alpha\beta}}=\frac{df_1(u_i)}{du_i}\left[\delta_{i\alpha}x_\beta+\sum_{k=N+1}^{N+K}\frac{d\nu_k(t-1)}{dw^{(1)}_{\alpha\beta}}w^{(1)}_{ik}\right]=\frac{df_1(u_i)}{du_i}\left[\delta_{i\alpha}x_\beta+\sum_{k=1}^{K}\frac{d\nu_k(t-1)}{dw^{(1)}_{\alpha\beta}}w^{(1)}_{i,k+N}\right]$$
. (6.34)

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:

$$\frac{d\nu_1(0)}{dw^{(1)}_{\alpha\beta}}=\frac{d\nu_2(0)}{dw^{(1)}_{\alpha\beta}}=\ldots=\frac{d\nu_K(0)}{dw^{(1)}_{\alpha\beta}}=0. \tag{6.35}$$

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 (6.29), (6.32) и (6.34).
5. Уточнить значения весов сети согласно правилам метода наискорейшего спуска:

- для нейронов выходного слоя сети по формуле
$$w_{\alpha\beta}^{(2)}(t) = w_{\alpha\beta}^{(2)}(t-1) - \eta\nabla_{\alpha\beta}^{(2)}E(t) \quad (6.36)$$
- для нейронов скрытого слоя сети по формуле
$$w_{\alpha\beta}^{(1)}(t) = w_{\alpha\beta}^{(1)}(t-1) - \eta\nabla_{\alpha\beta}^{(1)}E(t) \quad (6.37)$$

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:

$$\Delta w(t) = -\eta\nabla E(t) + \alpha(t)\Delta w(t-1), \quad (6.38)$$

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.

7 Специализированные сети

7.1 Сеть Вольтерри

Сеть Вольтерри относится к сетям со специализированной структурой. Это динамическая сеть для нелинейной обработки последовательности сигналов, задержанных относительно друг друга. Возбуждением для сети в момент t служит вектор $x=[x_t, x_{t-1},...,x_{t-L}]^T$, где L - количество единичных задержек, а $(L+1)$ означает длину вектора. В соответствии с определением ряда Вольтерри выходной сигнал y генерируется по формуле:

$$\begin{aligned} y(t) &= \sum_{i_1=1}^{L} w_{i_1} x(t-i_1) + \sum_{i_1=1}^{L}\sum_{i_2=1}^{L} w_{i_1 i_2} x(t-i_1)x(t-i_2) + \dots \\ &+ \sum_{i_1=1}^{L}\dots\sum_{i_K=1}^{L} w_{i_1 i_2 \dots i_K} x(t-i_1)x(t-i_2)\dots x(t-i_K) \end{aligned} \tag{7.1}$$

где x обозначает входной сигнал, а веса $w_{i_1}, w_{i_2},...,w_{i_1 i_2 .. i_K}$, называемые ядрами Вольтерри, соответствуют реакциям высших порядков. Порядок полинома Вольтерри K также называется степенью ряда Вольтерри.

Нелинейная функциональная зависимость Вольтерри является полиномиальным обобщением описания линейного фильтра *FIR*. Порядок этого полинома *K* называется степенью ряда Вольтерри. Пусть целевая функция для одного обучающего вектора выражается формулой:

$$E = \frac{1}{2}\big(y(t) - d(t)\big)^2 \tag{7.2}$$

В этом случае можно минимизировать значение целевой функции градиентными методами, которые сводятся к решению системы дифференциальных уравнений вида:

$$\begin{aligned} \frac{dw_{i_1}}{dt} &= -\mu(y(t)-d(t))x(t-i_1), \\ \frac{dw_{i_1 i_2}}{dt} &= -\mu(y(t)-d(t))x(t-i_1)x(t-i_2), \\ \frac{dw_{i_1 i_2 i_3}}{dt} &= -\mu(y(t)-d(t))x(t-i_1)x(t-i_2)x(t-x_3), \\ \frac{dw_{i_1 i_2 \dots i_{K-1} i_K}}{dt} &= -\mu(y(t)-d(t))x(t-i_1)x(t-i_2)\dots x(t-i_k). \end{aligned} \tag{7.3}$$

Для упрощения структуры сети представленное разложение Вольтерри можно записать в следующей форме:

$$y_t = \sum_{i_1=0}^{L} x_{t-1}\left[w_{i_1} + \sum_{i_2=0}^{L} x_{t-i_2}\left[w_{i_1 i_2} + \sum_{i_3=0}^{L} x_{t-i_3}(w_{i_1 i_2 i_3} + \dots) \right] \right], \tag{7.4}$$

где используются обозначения $y_t = y(t), x_{t-i_1} = x(t-i_1)$ и т.д. Каждое слагаемое в квадратных скобках представляет собой линейный фильтр

первого порядка, в котором соответствующие веса представляют импульсную реакцию другого линейного фильтра следующего уровня. Количество уровней, на которых создаются фильтры, равно порядку K. На рисунке 7.1 показано распространение сигналов по сети Вольтерри, реализующей зависимость (7.4) при ограничении $K = 3$.

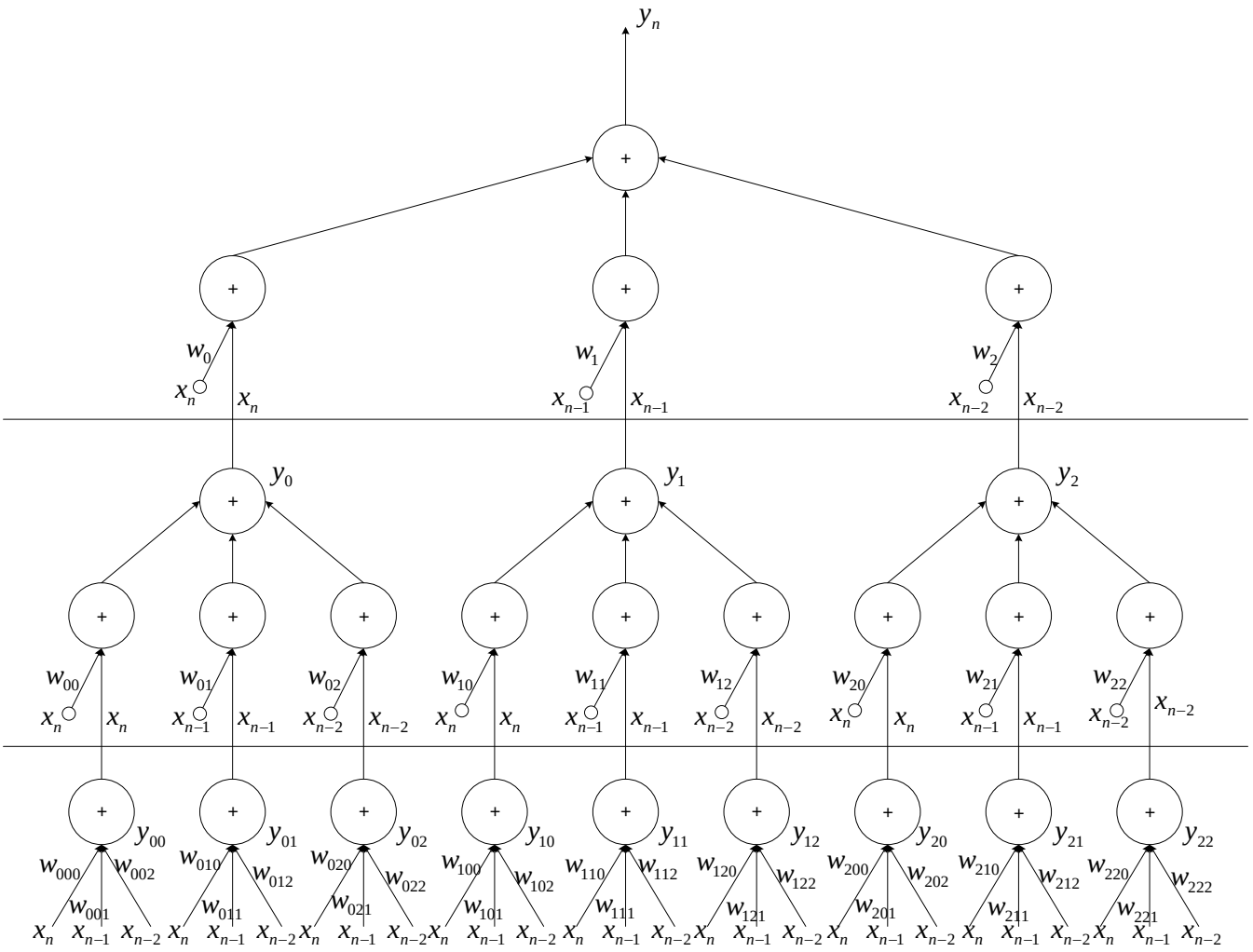


Рисунок 7.1 – Граф сети Вольтерри

Система представляет собой структуру типичной многослойной однонаправленной динамической нейронной сети. Это сеть с полиномиальной нелинейностью. Подбор весов производится последовательно слой за слоем, причём эти процессы независимы друг от друга, и, увеличение числа весов в слое и числа самих слоёв в незначительной степени сказывается на обусловленности задачи. Это даёт возможность существенно увеличить длину L и порядок K системы при её практической реализации. Обучение нейронной сети Вольтерри лучше всего приводить с использованием технологии сопряжённых графов.

Для сети, представленной на рисунке 7.1, сопряжённый граф строится без особого труда (рисунок 7.2).

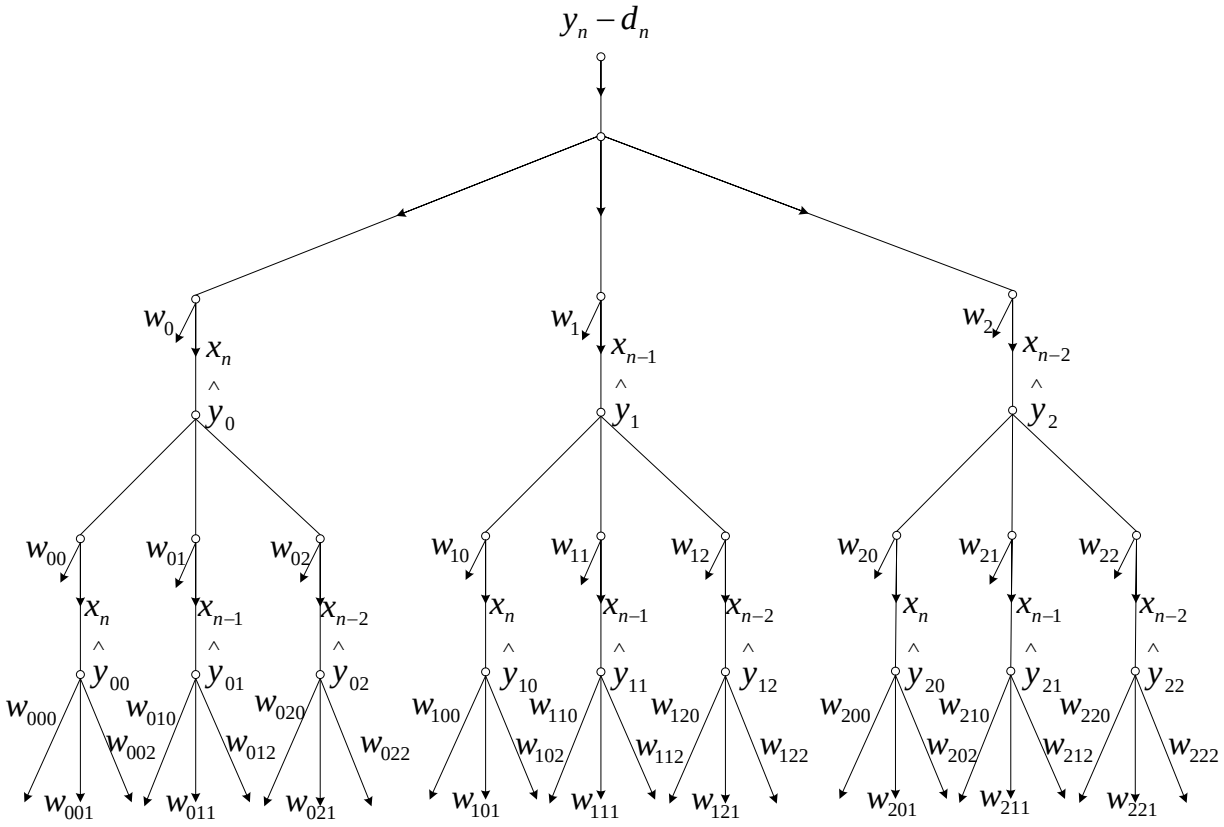


Рисунок 7.2 – Сопряженный граф сети Вольтерри

В соответствии с обозначениями, принятыми на рисунках, возбуждением сопряженного графа служит разностный сигнал $(y_t - d_t)$, где d_t обозначает ожидаемое, а y_t - фактическое значение в выходном узле системы в момент t. Формулы для определения конкретных компонентов вектора градиента имеют вид:

$$\frac{\partial E}{\partial w_{i_1}} = x_{t-i_1}(y_t - d_t),$$

$$\frac{\partial E}{\partial w_{i_1 i_2}} = x_{t-i_2}\hat{y}_{i_1},$$

$$\frac{\partial E}{\partial w_{i_1 i_2 i_3}} = x_{t-i_3}\hat{y}_{i_1 i_2},$$

$$\frac{\partial E}{\partial w_{i_1 i_2 \ldots i_{K-1} i_K}} = x_{t-i_k}\hat{y}_{i_1 i_2 \ldots i_{K-1}} \qquad (7.5)$$

$$\hat{y}_{i_1} = (x_{t-i_1}(y_t - d_t))$$

$$\hat{y}_{i_1 i_2} = (x_{t-i_2}\hat{y}_{i_1})$$

$$\hat{y}_{i_1 i_2 i_3} = (x_{t-i_3}\hat{y}_{i_2}))$$

В приведенных формулах сигналы, обозначенные символом ^, соответствуют сопряженному, а остальные – исходному графу системы. После определения конкретных компонентов градиента обучение сети с

применением оптимизационного метода наискорейшего спуска может быть сведено к решению дифференциальных уравнений:

$$\begin{aligned}
&\frac{dw_{i_1}}{dt} = -\mu \frac{\partial E}{\partial w_{i1}},\\
&\frac{dw_{i_1 i_2}}{dt} = -\mu \frac{\partial E}{\partial w_{i_1 i_2}},\\
&\frac{dw_{i_1 i_2 i_3}}{dt} = -\mu \frac{\partial E}{\partial w_{i_1 i_2 i_3}},\\
&\frac{dw_{i_1 i_2 .. i_{K-1} i_K}}{dt} = -\mu \frac{\partial E}{\partial w_{i_1 i_2 .. i_{K-1} i_K}}.
\end{aligned} \qquad (7.6)$$

где μ - коэффициент обучения.

Важным достоинством метода сопряженных графов считается простота учета равных значений весов в различных ветвях сети. Симметрия ядер Вольтерри приводит к равенству весов $w_{i_1 i_2 .. i_K}$ для всех перестановок индексов $i_1, i_2, ..., i_K$. Для двухиндексных весов это означает, что $w_{i_1 i_2} = w_{i_2 i_1}$. Подобные соотношения можно написать и для трёхиндексных и четырёхиндексных и так далее весов. Анализ обозначений весов сети, изображённой на рисунке 7.1, позволяет найти веса, которые имеют одни и те же значения.

С учетом симметрии ядер Вольтерри выражения, описывающие компоненты градиента относительно весов $w_{i_1 i_2 .. i_K}$, могут быть определенным образом модифицированы:

$$\begin{aligned}
&\frac{\partial E}{\partial w_{i_1 i_2}} = x_{t-i_2} \hat{y}_{i_1} + x_{t-i_1} \hat{y}_{i_2},\\
&\frac{\partial E}{\partial w_{i_1 i_2 i_3}} = x_{t-i_3} \hat{y}_{i_1 i_2} + x_{t-i_2} \hat{y}_{i_1 i_3} + x_{t-i_3} \hat{y}_{i_2 i_1} + x_{t-i_1} \hat{y}_{i_2 i_3} + x_{t-i_2} \hat{y}_{i_3 i_1} + x_{t-i_1} \hat{y}_{i_3 i_2}
\end{aligned} \qquad (7.7)$$

7.2 Сеткаскадной корреляции Фальмана

Сеть каскадной корреляции Фальмана – специализированная многослойная нейронная конструкция, в которой подбор структуры сети происходит параллельно с ее обучением путем добавления на каждом этапе обучения одного скрытого нейрона[2].

Таким образом, определение структуры сети и реализацию алгоритма ее обучения можно трактовать как выполнение подбора оптимальной архитектуры искусственной нейронной сети.

Архитектура сети каскадной корреляции представляет собой объединение нейронов взвешенными связями в виде развивающегося каскада (см. рисунок 7.3). Каждый очередной добавляемый нейрон подключается к входным узлам и ко всем уже существующим скрытым нейронам. Выходы

всех скрытых нейронов и выходные узлы сети напрямую подключаются также и к выходным нейронам.

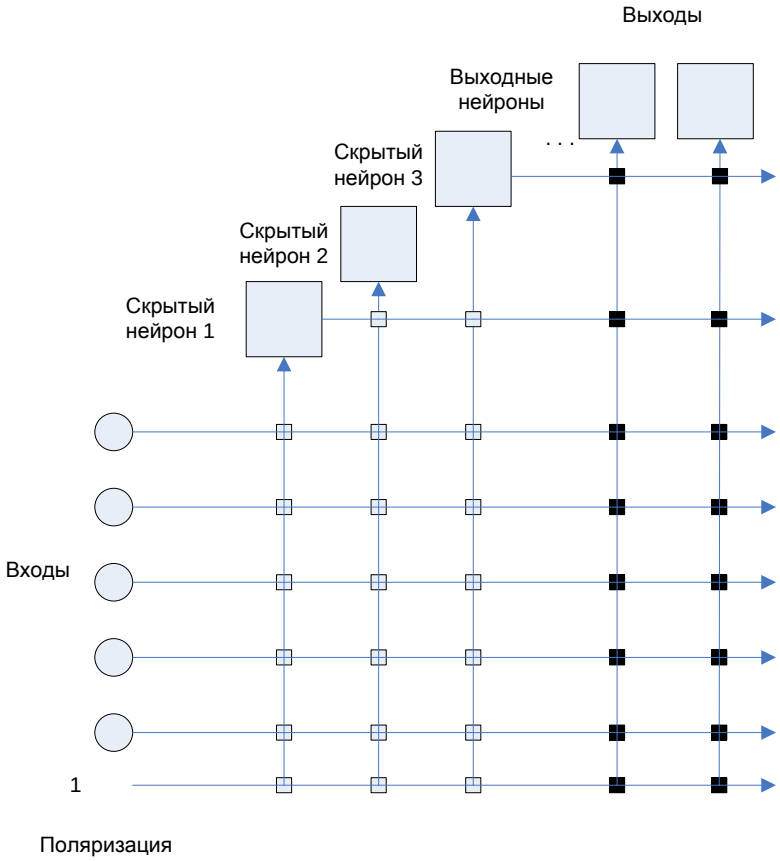


Рисунок 7.3 – Классическая архитектура сети каскадной корреляции Фальмана

Каждый нейрон старается так адаптировать веса своих связей, чтобы быть востребованным для выполняемого сетью отображения данных. На начальном этапе формируется сеть, состоящая только из входных узлов и выходных нейронов. Количество входов и выходов зависит исключительно от специфики решаемой задачи и не подлежит модификации. Скрытые нейроны добавляются в сеть по одному. Каждый добавляемый нейрон подключается ко всем входам сети и ко всем ранее добавленным скрытым нейронам. Выходные нейроны могут быть как линейными, так и нелинейными с произвольной функцией активации. Скрытые нейроны добавляются всеть по одному и подключаются ко всем входам сети и ко всем добавленным ранее скрытым нейронам. В момент подключения скрытого нейрона фиксируются его входные связи, которые в дальнейшем не изменяются, а выходные связи модифицируются в процессе обучения. Каждый добавляемый скрытый нейрон образует отдельный одноэлементный слой.

Процесс обучения начинается до ввода в неё скрытых нейронов. Сеть обучается на входных векторах и выходных нейронах путём минимизации целевой функции. Для этого могут применяться любые алгоритмы обучения. В оргинальной работе Фальмана использрвался эвристический алгоритм *Quickprop*.

Если результат функционирования сети считается удовлетворительным с точки зрения ожидаемой или допустимой погрешности, процесс обучения и формирования структуры сети завершается. В противном случае следует расширить структуру сети добавлением одного скрытого нейрона. Для этого применяется специальная процедура, в которой сначала формируются и фиксируются входные веса нового нейрона, а затем выход нейрона подключается ко всем выходным нейронам посредством связей с соответствующими весами, после чего происходит уточнение выходных весов при помощи выбранного алгоритма обучения.

Если полученный результат признаётся удовлетворительным, процесс обучения и формирования структуры сети завершается. В противном случае процедура добавления скрытых нейронов повторяется вплоть до достижения желаемых результатов.

Формирование скрытого нейрона начинается с подключения его в виде кандидата. Скрытый нейрон представляет собой обособленный элемент, соединённый со всеми входами сети и выходами всех скрытых нейронов. Веса связей нейрона-кандидата модифицируются в процессе обучения, но его выходной сигнал никуда не подаётся. Новый нейрон обучается на всём множестве обучающих выборок. Цель обучения состоит в таком подборе весов, при котором максимизируется корреляция между выходным значением сигнала нейрона и значением погрешности на выходе сети. Данная зависимость определяется коэффициентом корреляции *S*в виде:

$$S = \sum_{s=1}^{M} \left| \sum_{t-1}^{P} \left(v^t - \bar{v}\right)\left(e_s^t - \bar{e}_s\right) \right|, \qquad (7.8)$$

где p – число обучающих выборок, M – число выходных нейронов, v^t - выходной сигнал нейрона-кандидата при t-ой обучающей выборке, e_s^t - значение погрешности s-того выходного нейрона при t-ой обучающей выборке, $e_s^t = d_s^t - y_s^t$. $\bar{v}$ *и* $\bar{e}_s$ обозначены средние значения v *и* e_s, рассчитанные по всем обучающим выборкам.

Для максимизации значения *S*следуетопределить производную относительно всех входных весов нейрона-кандидата. На основе разложения составной функции получаем:

$$\frac{\partial S}{\partial w_i} = \sum_s \sum_t \sigma_s \left(e_s^t - \bar{e}_s\right) f''^t I_i^t, \qquad (7.9)$$

где σ_s - обозначение корреляции между s-тым выходом и нейроном кандидатом, f''^t - рассчитанная для t-ой обучающей выборки производная фукнции активации нейрона-кандидата относительно взвешенной суммы его выходных сигналов, I_i^t - импульс от i- тоговозбуждающего сигнала (входного сигнала сети или выходного сигнала от ранее введённых скрытых нейронов). После определение вектора градиента функции *S*относительно всех подбираемых весов выполняется максимизация *S*(на практике выполняется минимизация *S*)на основе любого метода оптимизации. В оригинальной работе Фальмана для этих целей использовался тот же алгоритм *Quickprop*,

что и для обучения весов выходных нейронов. После достижения максимального значения *S*нейрон-кандидат включается в структуру нейронной сети, подобранные веса его входных связей фиксируются, и продолжается процесс подбора выходных весов за счёт минимизации целевой функции.

Длядостижения наилучших результатов в процессе обучения, как правило, используются несколько нейронов-кандидатов. Так как обучение начинается со случайных значений весов, каждый кандидат получает разные конечные значения весов, характеризующиеся разными значениями коэффициента корреляции. Среди таких обученных кандидатов, выбирается кандидат с максимальным значением *S*, который и включается в структуру сети. Параллельное корреляционное обучение нескольких нейроно-кандидатов уменьшает вероятность попадания в точку локального минимума и ввода в структуру сети нейрона с плохо подобранными весами, которые на последующих этапах обучения уже невозможно будет откорректировать. Рекомендуемое авторами метода число нейронов-кандидатов составляет 5-10 нейронов.

Каждый нейрон-кандидат может иметь свою функцию активации: сигмоидальную, гауссову, радиальную и т.д. Побеждает тот нейрон, который лучше приспособлен к условиям обучающих выборок. Вследствие такого подхода сеть каскадной корреляции может объединять нейроны с разными функциями активации.

8 Нечёткие множества и нечёткий вывод

8.1 Основные понятия и определения теории нечётких множеств

Понятие нечётких множеств (*fuzzysets*) как обобщение чётких множеств было введено Л.Заде в 1965 году. Причиной создания теории нечётких множеств стала необходимость описания таких явлений и понятий, которые имеют многозначный и неточный характер. Известные до этого математические методы, использовавшие классическую теорию множеств и двузначную логику, не позволяли решать такие проблемы.

Перед формулированием определения нечёткого множества необходимо задать так называемую область рассуждений или пространство рассуждений, которое является чётким множеством. В случае неоднозначного понятия «много денег» большой будет признаваться одна сумма, если мы ограничимся диапазоном [0, 1000 руб.] и совсем другая в диапазоне [0, 1000000 руб.][5].

Традиционный способ представления элементов множества A состоит в применении характеристической функции $\mu_A(x)$, которая равна 1, если этот элемент принадлежит множеству A, или равна 0 в противном случае. В нечётких системах элемент может частично принадлежать к любому множеству.

Нечётким множеством A в некотором непустом пространстве X, что обозначается $A \subseteq X$, называется множество пар $A = \{(x, \mu_A(x)); x \in X\}$, где $\mu_A : X \to [0,1]$ - функция принадлежности нечёткого множества A[4].

Эта функция для каждого элемента $x \in X$ приписывет степень его принадлежности к нечёткому множеству A, при этом можно выделить три случая:

1. $\mu_A(x) = 1$ означает полную принадлежность элемента x множеству A, то есть $x \in A$;
2. $\mu_A(x) = 0$ означает отсутствие принадлежности элемента x множеству A, то есть $x \notin A$;
3. $0 < \mu_A(x) < 1$ означает частичную принадлежность элемента x множеству A.

Степень принадлежности элемента к множеству A, представляющая собой обобщение характеристической функции, называется функцией принадлежности $\mu_A(x)$, причём $\mu_A(x) \in [0,1]$. Конкретное значение функции принадлежности называется *степенью или коэффициентом принадлежности*[2].

Эта степень может быть определена явно функциональной зависимостью $\mu_A(x)$, либо дискретно – путём задания конечной последовательности значений $x \in \{x_n\}$ в виде:

$$A(x)=\left\{\frac{\mu(x_1)}{x_1},\frac{\mu(x_2)}{x_2},\ldots,\frac{\mu(x_n)}{x_n}\right\} \qquad (8.1)$$

В теории нечётких множеств, помимо переменных цифрового типа, существуют лингвистические переменные, с приписываемыми им значениями. Пусть переменная x обозначает температуру. Можно определить нечёткие множества: «температура, подходящая для купания в Балтийском море», «температура, подходящая для купания в Чёрном море», «температура, подходящая для купания в Средиземном море», характеризуемые функциями принадлежности $\mu_{балт}(x)$, $\mu_{Чёрное}(x)$, $\mu_{Средиз}(x)$. Определим все три нечётких множества A,B,C. Пусть $X=[15^\circ,\ldots,28^\circ]$. Множество, определяющее подходящую температуру Балтийского моря, можно задать следующим образом:

$$A=\frac{0{,}1}{15}+\frac{0{,}2}{16}+\frac{0{,}4}{17}+\frac{0{,}7}{18}+\frac{0{,}9}{19}+\frac{1}{20}+\frac{0{,}9}{21}+\frac{0{,}85}{22}+\frac{0{,}8}{23}+\frac{0{,}75}{24}+\frac{0{,}7}{25}+\frac{0{,}6}{26}+\frac{0{,}5}{27}+\frac{0{,}4}{28}$$

Множество, определяющее подходящую температуру Чёрного моря, можно задать следующим образом:

$$B=\frac{0}{15}+\frac{0{,}1}{16}+\frac{0{,}2}{17}+\frac{0{,}3}{18}+\frac{0{,}45}{19}+\frac{0{,}6}{20}+\frac{0{,}75}{21}+\frac{0{,}9}{22}+\frac{1}{23}+\frac{0{,}9}{24}+\frac{0{,}85}{25}+\frac{0{,}8}{26}+\frac{0{,}75}{27}+\frac{0{,}7}{28}$$

Множество, определяющее подходящую температуру Средиземного моря, можно задать следующим образом:

$$C=\frac{0}{15}+\frac{0}{16}+\frac{0}{17}+\frac{0{,}1}{18}+\frac{0{,}25}{19}+\frac{0{,}4}{20}+\frac{0{,}55}{21}+\frac{0{,}7}{22}+\frac{0{,}8}{23}+\frac{0{,}9}{24}+\frac{1}{25}+\frac{0{,}95}{26}+\frac{0{,}9}{27}+\frac{0{,}85}{28}$$

Существуют стандартные функции принадлежности, которые можно определить следующим образом[4]:

1. Функция принадлежности класса s определяется как:

$$s(x;a,b,c)=\begin{cases}0 & для\ x\le a,\\ 2\left(\dfrac{x-a}{c-a}\right)^2 & для\ a\le x\le b,\\ 1-2\left(\dfrac{x-c}{c-a}\right)^2 & для\ b\le x\le c,\\ 1 & для\ x\ge c,\end{cases} \qquad (8.2)$$

где $b=(a+c)/2$. Функция, принадлежащая к данному классу, имеет сигмоидальный вид, причём её форма зависит от параметров $a,b\,и\,c$. В точке $x=b=(a+c)/2$ она имеет значение, равное 0.5.

2. Функция принадлежности класса π определяется через функцию принадлежности класса s:

$$\pi(x;b,c)=\begin{cases}s(x;c-b,c-b/2,c) & для\ x\le c,\\ 1-s(x;c,c+b/2,c+b) & для\ x\ge c.\end{cases} \qquad (8.3)$$

Функция, принадлежащая к данному классу, имеет колоколообразный вид, и принимает нулевые значения в точках $x \geq c + b$ и $x \leq c - b$. В точках $x = c \pm b/2$ её значение равно 0.5.

3. Функция принадлежности класса γ задаётся выражением:

$$\gamma(x; a, b) = \begin{cases} 0 & для\ x \leq a, \\ \dfrac{x-a}{b-a} & для\ a \leq x \leq b, \\ 1 & для\ x \geq b. \end{cases} \quad (8.4)$$

Функции данного класса похожи на функции класса s.

4. Функция принадлежности класса t имеет вид:

$$t(x; a, b, c) = \begin{cases} 0 & для\ x \leq a, \\ \dfrac{x-a}{b-a} & для\ a \leq x \leq b, \\ \dfrac{c-x}{c-b} & для\ b \leq x \leq c, \\ 0 & для\ x \geq c. \end{cases} \quad (8.5)$$

Функции данного класса имеют треугольный вид.

5. Функция принадлежности класса L определяется выражением:

$$L(x; a, b) = \begin{cases} 0 & для\ x \leq a, \\ \dfrac{b-x}{b-a} & для\ a \leq x \leq b, \\ 1 & для\ x \geq b. \end{cases} \quad (8.6)$$

Так, например, для множества «большие деньги» функция принадлежности будет иметь класс s.

Основные определения свойств нечётких множеств:

1. Каждое нечёткое множество имеет определённый *носитель* (*support*). Носителем множества $Supp(A)$ является подмножество тех элементов A, для которых коэффициент принадлежности к A не равен нулю, то есть $Supp(A) = \{x, \mu_A(x) > 0\}$.
2. Два множества $A(x)$ и $B(x)$ равны между собой, когда $\mu_A(x) = \mu_B(x)$ для каждого элемента обоих множеств.
3. *Кардинальное число* нечёткого множества A равно сумме коэффициентов принадлежности всех элементов к этому множеству, $M(A) = \sum \mu_A(x)$. Это обобщение аналогичного понятия обычных множеств, для которых кардинальное число равно сумме элементов множества.
4. Нечёткое множество является *нормальным*, если хотя бы один элемент этого множества имеет коэффициент принадлежности, равный 1.
5. *Сечение* α нечёткого множества A образуется подмножеством A_α, содержащим те элементы множества A, для которых $\mu_A(x) > \alpha$

(слабое сечение) или $\mu_A(x) \geq \alpha$ (сильное сечение), причём $\alpha \in [0,1]$.

6. *Нечёткое множество называется пустым и обозначается* $A = \varnothing$ тогда и только тогда, когда $\mu_A(x) = 0$ для каждого $x \in X$.
7. *Нечёткое множество* A содержится в нечётком множестве $B, A \subset B$, тогда и только тогда, когда $\mu_A(x) \leq \mu_B(x)$ для каждого $x \in X$.

8.2 Операции на нечётких множествах

На нечётких множествах можно определить ряд математических операций, являющихся обобщением аналогичных операций, выполняемых на чётких множествах:

1. *Логическая сумма множеств* $A \cup B$

$$\mu_{A \cup B}(x) = \mu_A(x) \cup \mu_B(x) = Max[\mu_A(x), \mu_B(x)], \qquad (8.7)$$

где знак $\cup$ обозначает оператор Max.

Пример 8.1.

Пусть даны два нечётких множества A и B, определённые следующим образом:

$$A = \left\{ \frac{1,0}{x_1}, \frac{0,7}{x_2}, \frac{0,5}{x_3}, \frac{0,1}{x_4} \right\}$$

$$B = \left\{ \frac{0,2}{x_1}, \frac{0,5}{x_2}, \frac{0,6}{x_3}, \frac{0,7}{x_4} \right\}$$

Логическая сумма этих множеств $C = A \cup B$ равна:

$$C = \left\{ \frac{1,0}{x_1}, \frac{0,7}{x_2}, \frac{0,6}{x_3}, \frac{0,7}{x_4} \right\}.$$

2. *Логическоепроизведение множеств* $A \cap B$

$$\mu_{A \cap B}(x) = \mu_A(x) \cap \mu_B(x) = Min[\mu_A(x), \mu_B(x)], \qquad (8.8)$$

где знак $\cap$ обозначает оператор Min. Для данных примера 8.1 множество $C = A \cap B$ будет иметь вид:

$$C = \left\{ \frac{0,2}{x_1}, \frac{0,5}{x_2}, \frac{0,5}{x_3}, \frac{0,1}{x_4} \right\}.$$

3. *Отрицание (дополнение) множества* A называется множество $\overline{A}$, с функцией принадлежности

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x). \qquad (8.9)$$

В отличие от обычных множеств, где отрицание элементов, принадлежащих к множеству, даёт пустое множество, отрицание нечёткого множества определяет непустое множество, состоящее из элементов, функции принадлежности которых, также определены на интервале [0,1].

4. *Равенство множеств* A и B.

Нечёткие множества $A(x)$ и $B(x)$ равны между собой, когда для всех элементов x_i обоих множеств выполняется условие $\mu_A(x_i) = \mu_B(x_i)$.

5. *Операция концентрациимножества* $A\ CON(A)$

$$\mu_{CON}(x) = [\mu_A(x)]^2. \qquad (8.10)$$

Эта операция часто выполняется при действиях с лингвистическими переменными, в которых она отождествляется с интенсификатором «очень».

6. *Операция растяжениямножества* $A\ DIL(A)$

$$\mu_{DIL}(x) = [\mu_A(x)]^{0,5}. \qquad (8.11)$$

Лингвистическое значение этой операции формулируется как «примерно» или «приблизительно».

7. *Ограниченнаяразностьдвух нечётких множеств* $A|-|B$

$$\mu_{A|-|B}(x) = \max\{0, \mu_A(x) - \mu_B(x)\}, \qquad (8.12)$$

8. *Нормализация множества* $NORM(A)$

$$\mu_{NORM}(x) = \frac{\mu_A(x)}{\max\{\mu_A(x)\}}. \qquad (8.13)$$

Следует отметить, что множество A считается подмножеством множества B, то есть $A \subset B$, когда для всех элементов выполняется неравенство: $\mu_A(x_i) \le \mu_B(x_i)$.

9. *Декартово произведение нечётких множеств* $A \times B$

$$\mu_{A\times B}(x) = \mu_A(x) \cap \mu_B(x) = Min[\mu_A(x), \mu_B(x)] \qquad (8.14)$$

или

$$\mu_{A\times B}(x) = \mu_A(x) * \mu_B(x) \qquad (8.15)$$

для каждого $x \in X$ *и* $y \in Y$.

Определённые выше операции обладают свойствами ассоциативности, коммутативности и дистрибутивности, которые определяются следующим образом:

- ассоциативность: $(A * B) * C = A * (B * C)$;
- коммутативность: $A * B = B * A$ (за исключением ограниченной разности);
- дистрибутивность: $A * (B \circ C) = (A * B) \circ (A * C)$,

где операторы * и ∘ обозначают любую операцию на нечётких множествах. Из свойств нечётких множеств следует, что в отличие от произведения обычных множеств логическое произведение множества и его отрицания не обязательно образуют пустое множество, что можно записать в виде:

$$A \cap \overline{A} \neq \varnothing. \qquad (8.16)$$

Точно также логическая сумма нечёткого множества A и его отрицание не образуют полное множества $\bigcup$, что можно записать в виде:

$$A \cup \overline{A} \neq \bigcup. \qquad (8.17)$$

8.3 Треугольные нормы

Операции произведения и суммы нечётких множеств могут быть определены не только как минимум и максимум соответственно. Часто встречаются следующие определения [5]:

1. $\mu_{A\cup B}(x)=\mu_A(x)+\mu_B(x)-\mu_A(x)*\mu_B(x),$ (8.18)

2. *Алгебраическоепроизведениедвух множеств* $A*B$

$$\mu_{A\cap B}(x)=\mu_A(x)*\mu_B(x).$$ (8.19)

3.*Ограниченная сумма двух нечётких множеств* $A\left|+\right|B$

$$\mu_{A\cup B}(x)=\min\{1,\mu_A(x)+\mu_B(x)\},$$ (8.20)

4.*Ограниченноепроизведениедвух нечётких множеств* $A\left|\cdot\right|B$

$$\mu_{A\cap B}(x)=\max\{0,\mu_A(x)+\mu_B(x)-1\},$$ (8.21)

$$5.\ \mu_{A\cup B}(x)=\begin{cases}\mu_A(x), если\ \mu_B(x)=0\\ \mu_B(x), если\ \mu_A(x)=0\\ 1, если\ \mu_A(x),\mu_B(x)>0\end{cases},$$ (8.22)

$$6.\ \mu_{A\cap B}(x)=\begin{cases}\mu_A(x), если\ \mu_B(x)=1\\ \mu_B(x), если\ \mu_A(x)=1\\ 0, если\ \mu_A(x),\mu_B(x)<1\end{cases}$$ (8.23)

Данные соотношения можно определить как примеры действия более общих понятий треугольных норм, *S*-нормы и*T*-нормы соответственно:

$$\mu_{A\cup B}(x)=S(\mu_A(x),\mu_B(x))$$ (8.24)

$$\mu_{A\cap B}(x)=T(\mu_A(x),\mu_B(x))$$ (8.25)

8.4 Нечёткий вывод

8.4.1 Основные правила вывода

В нечётких продукционных моделях используются в основном два способа вывода заключений:прямой и обратный. Базовое правило прямого вывода типа «если-то» (продукционное правило), называется нечёткой импликацией, принимающей форму:

если x это A, то y это B, (8.26)

где X – область определения условия правила, Y – область определния заключения, $x\in X$, $y\in Y$, A и B - это нечёткие множества, идентифицированные через соответствующие функции принадлежности для переменных x и y. Часть продукционного правила «x *это* A» называется *условием (предпосылкой)*, а другая часть «y *это* B» - *следствием (заключением)*. Это обобщённое (нечёткое) правило *modusponens*.

Нечёткое рассуждение – это процедура, которая позволяет определить заключение, вытекающее из множества правил «если – то». Такое множество при N переменных x_i может принять вид:

если x_1 *это* A_1 *и* x_2 *это* A_2 *и…и* x_N *это* A_N, *то y это B*. (8.27)

Переменные $x_1, x_2, ..., x_N$ образуют $N-$мерный входной вектор x, составляющий аргумент условия, в котором $A_1, A_2, ..., A_N$ и B обозначают величины соответствующего коэффициента принадлежности $\mu_A(x)$ и $\mu_B(y)$. Необходимо обратить внимание на то, что здесь присутствуют индивидуальные функции принадлежности для каждой переменной x_i, и отдельно для y. Случайное значение функции принадлежности $\mu_A(x)$, где x – это вектор $x = x_1, x_2, ..., x_N$, относящееся к условию импликации (уровень активации правила), должно в дальнейшем интерпретироваться с использованием введённых ранее нечётких операций.

Операция нечёткой импликации является основной в нечётких продукционных моделях. В настоящее время существует большое число (несколько десятков) интерпретаций нечёткой импликации, наиболее эффективными из нихявляются следующие [5]:

1. Правило типа минимум (правило Мамдани):

$$\mu_{A\to B}(x,y) = \min[\mu_A(x), \mu_B(y)] \quad (8.28)$$

2. Правило типа произведение (правило Ларсена):

$$\mu_{A\to B}(x,y) = \mu_A(x) * \mu_B(y) \quad (8.29)$$

3. Правило Лукашевича (Лукасевича):

$$\mu_{A\to B}(x,y) = \min[1, 1-\mu_A(x)+\mu_B(y)] \text{ или}$$
$$\mu_{A\to B}(x,y) = \max[0, \mu_A(x)+\mu_B(y)-1] \quad (8.30)$$

4.Правило типа максмин (правило Заде):

$$\mu_{A\to B}(x,y) = \max\{\min[\mu_A(x), \mu_B(y)], 1-\mu_A(x)\} \quad (8.31)$$

5.Бинарное правило Клине-Дэнса:

$$\mu_{A\to B}(x,y) = \max[1-\mu_A(x), \mu_B(y)], \text{ при } \mu_A(x) \ge \mu_B(y) \quad (8.32)$$

6.Правило ограниченной суммы:

$$\mu_{A\to B}(x,y) = \min\{1, [\mu_A(x)+\mu_B(y)]\} \quad (8.33)$$

7.Правило Гёделя :

$$\mu_{A\to B}(x,y) = \begin{cases} 1, \text{если } \mu_A(x) \le \mu_B(y), \\ \mu_B(y), \text{если } \mu_A(x) > \mu_B(y) \end{cases} \quad (8.34)$$

8. Правило Гогуэна:

$$\mu_{A\to B}(x,y) = \min[1, \mu_B(y)/\mu_A(x)], \text{при } \mu_A(x) > 0 \quad (8.35)$$

9. Правило стандартной чёткой импликации:

$$\mu_{A\to B}(x,y) = \begin{cases} 1, \text{если } \mu_A(x) \le \mu_B(y), \\ 0, \text{если } \mu_A(x) > \mu_B(y) \end{cases} \quad (8.36)$$

10. Правило Гейнса:

$$\mu_{A\to B}(x,y) = \begin{cases} 1, \text{если } \mu_A(x) \le \mu_B(y), \\ \mu_B(y)/\mu_A(x), \text{если } \mu_A(x) > \mu_B(y) \end{cases} \quad (8.37)$$

11. Правило Клине-Дэнса-Лукашевича:

$$\mu_{A\to B}(x,y) = 1-\mu_A(x)+\mu_A(x)*\mu_B(y) \quad (8.38)$$

12. Правило вероятностной импликации:

$$\mu_{A\to B}(x,y)=\min\{1,1-\mu_A(x)+\mu_A(x)*\mu_B(y)\} \quad (8.39)$$

13. Правило Н.Вади:

$$\mu_{A\to B}(x,y)=\max\{\mu_A(x)*\mu_B(y),1-\mu_A(x)\} \quad (8.40)$$

14. Правило Ягера:

$$\mu_{A\to B}(x,y)=\mu_A(x)^{\mu_B(y)} \quad (8.41)$$

Набор этих правил не исчерпывает все известные определения нечёткой импликации.

Приписывание единственного значения функции принадлежности, описывающей многомерное условие, называется *агрегированием предпосылки*.

Каждой импликации $A\to B$, определённой выражением (8.27), можно приписать единственное значение функции принадлежности $\mu_{A\to B}(x,y)$.

Приписывание единственного значения функции принадлежности всей импликации называется *процедурой агрегирования на уровне импликации*.

Функции принадлежности в нечётких продукционных правилах зависят от функции принадлежности $\mu_{A\to B}(x,y)$ нечёткой импликации $A\to B$, которая равнозначна некоторому нечёткому отношению $R\subseteq X\times Y$.

8.5 Система нечёткого вывода Мамдани-Заде

Элементы теории нечётких множеств, правила импликации и нечётких рассуждений образуют систему нечёткого вывода. В ней можно выделить базу правил, содержащую множество используемых в системе нечётких правил и описания функций принадлежности, механизм вывода и агрегирования, который формируется применяемыми правилами нечёткой импликации. Кроме того, если в качестве входных и выходных сигналов системы выступают чёткие величины, то в состав системы должны входить *фуззификатор* и *дефуззификатор*. Структура такой системы представлена на рисунке 8.1.

Фуззификатор преобразует точное множество входных данных в нечёткое множество, определяемое с помощью значений функций принадлежности, дефуззификатор решает обратную задачу – формирует однозначное решение значения выходной переменной на основании многих нечётких выводов, вырабатываемых исполнительным модулем нечёткой системы.

Выходной сигнал модуля вывода может иметь вид M нечётких множеств, определяющих изменения выходной переменной. Дефуззификатор преобразует этот диапазон в одно конкретное значение, принимаемое в качестве выходного сигнала всей системы. Конкретная форма функции дефуззификации зависит от применяемой T-нормы, определения нечёткой импликации и от способа определения декартова произведения.

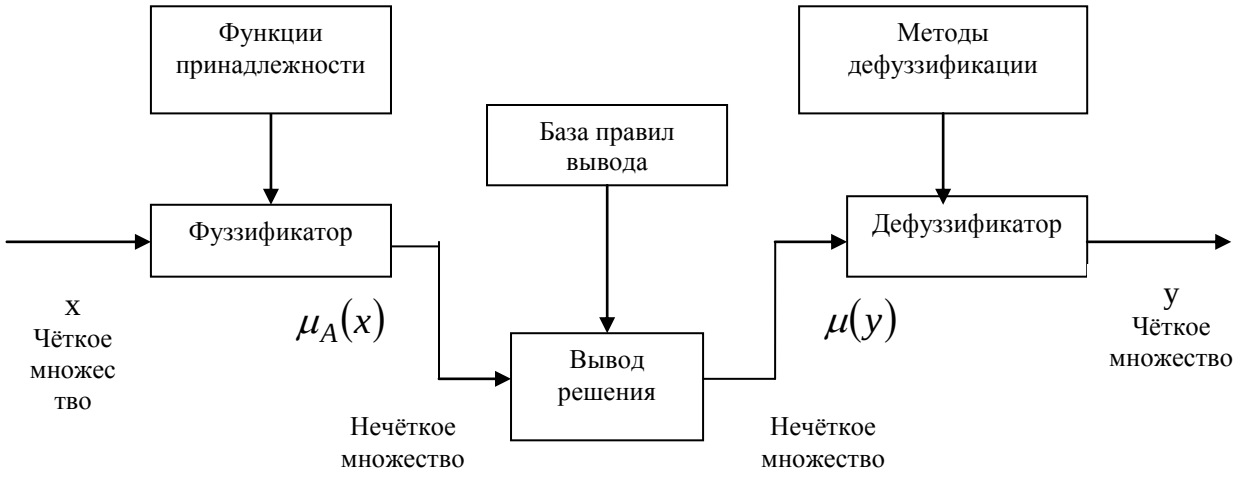


Рисунок 8.1 Структура нечёткой системы.

Так как допускается применение множества нечётких правил, в модуле вывода предусматривается блок агрегирования, чаще всего реализуемый в виде логического суммы. Описанная система вывода называется системой Мамдани - Заде. Как правило, в модели Мамдани - Заде присутствуют следующие операторы [2]:

- оператор логического или алгебраического произведения для агрегации всех компонентов вектора x условия;
- оператор логического или алгебраического произведения для определения значения функции принадлежности для всей импликации $A \to B$;
- оператор логической суммы как агрегатор равнозначных результатов импликации многих правил;
- оператор дефуззификации, трансформирующий нечёткий результат $\mu(y)$ в чёткое значение переменной y.

8.6 Фуззификатор

Фуззификатор преобразует N-мерный входной вектор $x = [x_1, x_2, .., x_N]^T$ в нечёткое множество A, характеризуемое функцией принадлежности $\mu_A(x)$ с чёткими переменными. Нечёткие системы могут иметь функции принадлежности произвольной структуры, в частности, стандартные формы, приведённые в разделе 8.1., но наиболее распространёнными являются функции гауссовского типа, а также треугольные и трапецеидальные функции. Обобщённая гауссовская функция с центром c радиусом (шириной) σ и парметром формы кривой b определяется формулой:

$$\mu_A(x) = \exp\left[-\left(\frac{x-c}{\sigma}\right)^{2b}\right]. \qquad (8.42)$$

Значение b=1 соответствует стандартной функции Гаусса, также можно подобрать значения параметра b, при которых формула (8.42) будет определять треугольную и трапецеидальную функции. На практике часто используется симметричная треугольная функция:

$$\mu_A(x) = \begin{cases} 1 - \dfrac{|x-c|}{d} \text{ для } x \in [c-d, c+d], \\ 0 \quad \text{для остальных} \end{cases} \qquad (8.43)$$

где c - точка середины основания треугольника, а d - половина основания треугольника.

Обобщением треугольной функции является трапецеидальная функция. Если обозначить y и z соответственно координаты точек начала и конца нижнего основания, t - длину верхнего основания, c - точку середины нижнего основания, а s - угол наклона между боковыми сторонами и нижним основанием, то трапецеидальная функция описывается следующей зависимостью:

$$\mu_A(x) = \begin{cases} 0 \quad \text{для } x > z \text{ или } x < y \\ 1 \quad \text{для } c - \dfrac{t}{2} \le x \le c + \dfrac{t}{2} \\ s(z-x) \quad \text{для } c + \dfrac{t}{2} \le x \le z \\ s(z-y) \quad \text{для } y \le x \le c - \dfrac{t}{2} \end{cases} \qquad (8.44)$$

Изменение значения параметра t может преобразовать трапецеидальную функцию в треугольную.

8.7 Дефуззификатор

Дефуззификатор трансформирует нечёткое множество в полностью детерминированное точечное решение y. Нечёткое множество представляет зависимость $\mu(y) = \mu_{A \to B}(y)$ как функцию от выходной переменной y. Преобразование этого множества в единственное точечное решение возможно следующими наиболее распространёнными способами[2]:

1. Дефуззификация относительно суммы центров (если на выходе блока выработки решения формируется k нечётких множеств):

$$y_c = \frac{\sum_i \mu(y_i) y_i}{\sum_i \mu(y_i)}; \qquad (8.45)$$

2. Дефуззификация относительно среднего центра (если на выходе блока выработки решения формируется k нечётких множеств):

$$y_c = \frac{\sum_i \mu(y_{ci}) y_{ci}}{\sum_i \mu(y_{ci})}, \quad (8.46)$$

где y_{ci} обозначает центр i-го нечёткого множества, в котором функция $\mu(y_{ci})$ - принимает максимальное значение, то есть $\mu(y_{ci}) = \max_{y_{ci}} \mu(y_{ci})$;

3. Дефуззификация относительно среднего максимума (если на выходе блока выработки решения формируется едиственное нечёткое множество):

$$y_M = \frac{\sum_{i=1}^{M} y_i}{M}, \quad (8.47)$$

где m обозначает количество точек переменной y, в которых функция $\mu(y)$ достигает максимального значения. Если функции $\mu(y)$ имеет максимальное значение только в одной точке $y_{\max}$, то $y_M = y_{\max}$. Если функции $\mu(y)$ достигает свои максимальные значения между y_l и y_p, то $y_M = \frac{1}{2}(y_l + y_p)$;

4. Дефуззификация в форме выбора минимального из максимальных значений y (левый максимум) (если на выходе блока выработки решения формируется 1 нечёткое множество):

y_s - *наименьшее значение* y *,для которого* $\{\mu(y) = \max\}$; (8.48)

5. Дефуззификация в форме выбора максимального из максимальных значений y (правый максимум) (если на выходе блока выработки решения формируется 1 нечёткое множество):

y_s - *наибольшее значение* y *,для которого* $\{\mu(y) = \max\}$. (8.49)

На практике чаще всего применяется дефуззификация относительно среднего центра.

8.8 Модель вывода Такаги-Сугено-Канга

Наибольшую популярность среди нечётких систем адаптивного типа приобрела модель вывода Такаги-Сугено-Канга (TSK) [2]. В этой модели функция заключения определяется функциональной зависимостью. Благодаря этому, дефуззификатор на выходе системы не требуется, а модель вывода значительно упрощается. Общая форма модели TSK:

если x_1 *это* A_1 *И* x_2 *это* A_2 *И...И* x_N *это* A_N, *то* $y = f(x_1, x_2, ..., x_N)$. (8.50)

В векторной записи её можно записать:

если x *это* A, *то* $y = f(x)$, (8.51)

где $f(x) = f(x_1, x_2, ..., x_N)$ - чёткая функция. Условие модели TSK аналогично модели Мамдани-Заде, принципиальное отличие касается заключения,

которое представляется в форме функциональной зависимости, чаще всего – в виде полиномиальной функции нескольких переменных.

Классическое представление этой функции – это полином первого порядка:

$$y = f(x) = p_0 + \sum_{j=1}^{N} p_j x_j \,, \tag{8.52}$$

в котором коэффициенты $p_0, p_1,...,p_N$ - это веса, подбираемые в процессе обучения.

Если в модели TSK используется M правил вывода, то выход системы определяется как среднее нормализованное взвешенное значение. Если приписать каждому правилу вес w_i (интерпретируются как $\mu_A{}^{(i)}(x)$ в форме алгебраического произведения), то выходной сигнал можно представить в виде:

$$y = \frac{\sum_{i=1}^{M} w_i y_i}{\sum_{i=1}^{M} w_i}, \tag{8.53}$$

или

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{i=1}^{M} w_i} y_i = \sum_{i=1}^{M} w_i' y_i \,. \tag{8.54}$$

Необходимо отметить, что в выражении (8.54) веса w_i отвечают условию нормализации: $\sum_{i=1}^{M} \frac{w_i}{\sum_{i=1}^{M} w_i} = 1$. Если для каждого i-го правила реализуется функция вида (8.34), то можно получить описание выходной функции модели TSK в виде:

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{i=1}^{M} w_i} \left(p_{io} + \sum_{J=1}^{N} p_{ij} x_j \right), \tag{8.55}$$

которая линейна относительно всех входных переменных системы x_j для $j = 1,2,...,N$. Веса w_i являются нелинейными параметрами функции y, которые уточняются в процессе обучения.

8.9 Модель вывода Цукамото

В модели Цукамото в качестве функций заключения используются монотонные (возрастающие или убывающие) функции f^{-1}[5]. Заключения правил формируются путём обратного преобразования этих функций по полученным значениям предпосылок данных правил:

если x_1 *это* A_1 *И* x_2 *это* A_2 *И*...*И* x_N *это* A_N, *то* $y = f^{-1}(w)$. (8.56)

где w - уровень срабатывания предпосылки правила.

Условие модели Цукамото аналогично модели Мамдани-Заде, принципиальное отличие касается заключения.

Если в модели Цукамото используется M правил вывода, то выход системы определяется как среднее взвешенное значение. Если приписать каждому правилу вес w_i (интерпретируются как $\mu_A{}^{(i)}(x)$ в форме алгебраического произведения), то выходной сигнал можно представить в виде [2]:

$$y = \frac{\sum_{i=1}^{M} w_i f_i^{-1}(w_i)}{\sum_{j=1}^{M} w_j}, \quad (8.57)$$

где $y_i = f_i^{-1}(w_i)$- значение аргумента функции f_i, при котором $w_i = f_i(y_i)$.

8.10 Модель вывода Ларсена

Правила вывода в модели Ларсена формируются аналогично модели Мамдани-Заде. В модели Ларсена присутствуют следующие операторы [5]:

- оператор алгебраического или логического произведения для агрегации всех компонентов вектора x условия;
- оператор алгебраического произведения для определения значения функции принадлежности для всей импликации $A \to B$;
- оператор логической суммы как агрегатор равнозначных результатов импликации многих правил;
- оператор дефуззификации, трансформирующий нечёткий результат $\mu(y)$ в чёткое значение переменной y.

9 Нейронные продукционные нечёткие сети

9.1 Нечеткая нейронная сеть Ванга-Менделя

Если в модели Мамдани-Заде в качестве агрегатора использовать оператор алгебраического произведения, то дефуззификация относительно среднего центра, приводит к модели Менделя-Ванга [2]. Следует отметить, что $\mu(y)$ состоит из суммы нечетких функций для импликаций всех M правил, образующих систему нечеткого вывода. В модели Мамдани-Заде каждое из этих M правил определяется уровнем активации условия, $\mu(y_i)=\prod_{j=1}^{N}\mu_A(x_j)$, тогда как y_i - это значение y, при котором величина $\mu(y)$ становится максимальной (либо принимает среднее из максимальных значений). Пусть величина y_i обозначает центр нечеткого множества заключения i-го правила вывода. Тогда дефуззификация относительно среднего центра в модели Менделя- Ванга , в соответствии с которой:

$$y=\frac{\sum_{i=1}^{M} y_i\left[\prod_{j=1}^{N}\mu_{A_i}(x_j)\right]}{\sum_{i=1}^{M}\left[\prod_{j=1}^{N}\mu_{A_i}(x_j)\right]}. \qquad (9.1)$$

Допустим, что существует нечеткая система, описываемая зависимостью (9.66), на вход которой подается последовательность векторов $x=[x_1,x_2,...,x_N]^T$. При использовании фуззификатора в виде обобщенной гауссовской функции $\mu(x)=\exp\left[-\left(\frac{x-c}{\sigma}\right)^{2b}\right]$ выходной сигнал у этой системы определяется по формуле :

$$y=f(x)=\frac{\sum_{i=1}^{M} y_i\left[\prod_{j=1}^{N}\exp\left[-\left(\frac{x_j-c_j^{(i)}}{\sigma_j^{(i)}}\right)^{2b_j^{(i)}}\right]\right]}{\sum_{i=1}^{M}\left[\prod_{j=1}^{N}\exp\left[-\left(\frac{x_j-c_j^{(i)}}{\sigma_j^{(i)}}\right)^{2b_j^{(i)}}\right]\right]}, \qquad (9.2)$$

в которой $\left(c_j^{(i)},\sigma_j^{(i)},b_j^{(i)}\right)$ обозначают параметры центра, ширины и формы (условия) j-го компонента вектора x для i-го нечеткого правила вывода.

Выражение (9.2) определяет непрерывную функцию, которая может использоваться для аппроксимации произвольно заданной непрерывной функции $g(x)$ от многих переменных x_j, образующих вектор x. При соответствующем подборе параметров условия $\left(c_j^{(i)},\sigma_j^{(i)},b_j^{(i)}\right)$ и заключения

(y_i), функция (9.2) может аппроксимировать заданную функцию $g(x)$ с произвольной точностью ε. Способность нечёткой системы, характеризующейся рядом нелинейных функций от одной переменной, к аппроксимации нелинейной функции от многих переменных, свидетельствует о возможностях практического применения нечётких систем.

Если использовать в качестве основы дальнейших рассуждений выражение (9.2), можно получить структуру нечеткой сети (рисунок 9.1), определенную Л.Вангом и Дж.Менделем[2].

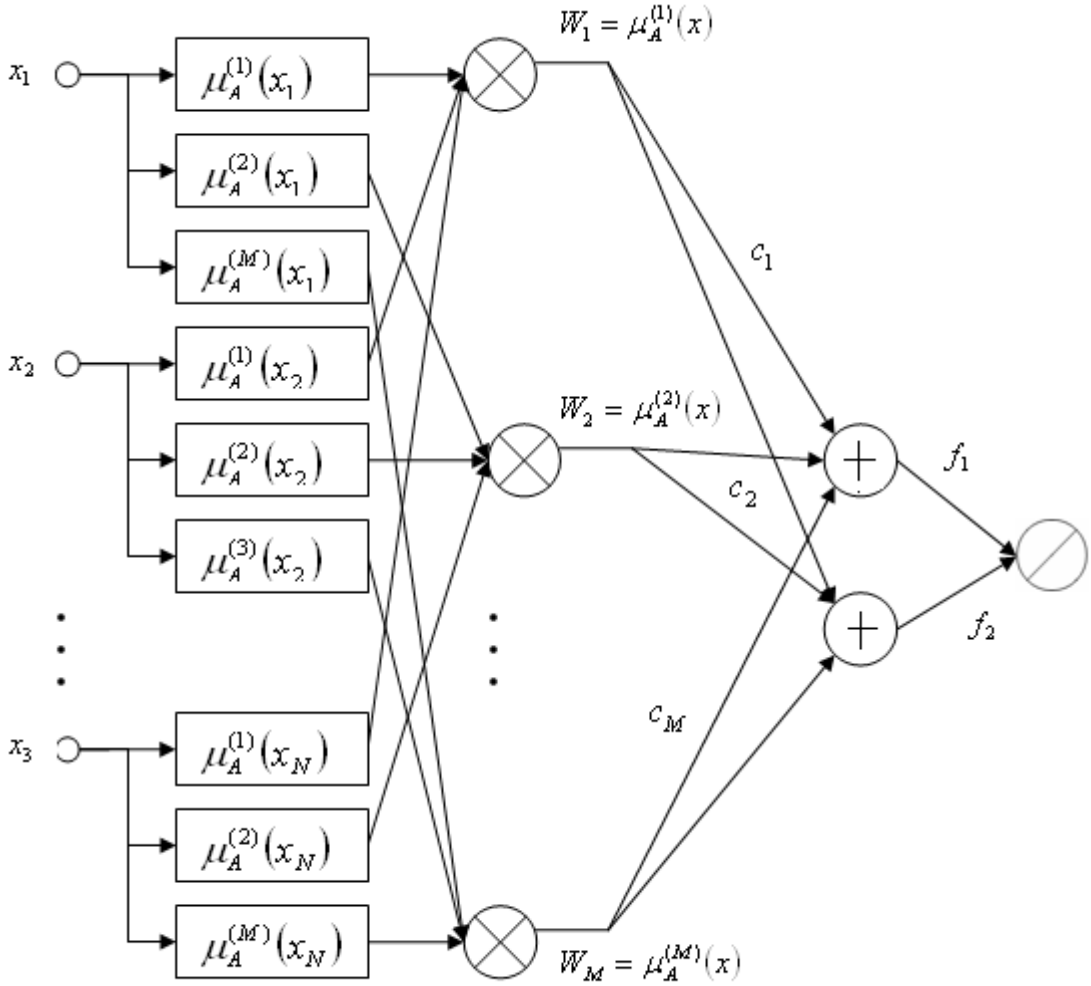


Рисунок 9.1 Структура нечеткой нейронной сети Ванга-Менделя.

Это четырехслойная структура, в которой первый слой выполняет фуззификацию входных переменных, второй – агрегирование значений отдельных переменных x_j в условии *i*-го правила вывода, третий (линейный) – агрегирование M правил вывода (первый нейрон) и генерацию нормализующего сигнала (второй нейрон), тогда как состоящий из одного нейрона выходной слой осуществляет нормализацию, формируя выходной сигнал $y(x)$.

Только первый и третий слои являются параметрическими. В первом слое это параметры функции фуззификации $\left(c_j^{(i)}, \sigma_j^{(i)}, b_j^{(i)}\right)$, а в третьем слое -

веса $w_1, w_2, ..., w_M$, интерпретируемые как центры y_i функции принадлежности следствия i -того нечеткого правила вывода.

Представленная на рисунке 9.4 сетевая структура реализует функцию аппроксимации (9.3), которую с учетом введенных обозначений можно записать в виде

$$y(x) = \frac{1}{\sum_{i=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(i)}(x_j)\right]}\sum_{i=1}^{M} y_i\left[\prod_{j=1}^{N}\mu_A^{(i)}(x_j)\right]. \qquad (9.3)$$

9.2 Адаптивный алгоритм обучения нечёткой сети Ванга- Менделя

Важным вопросом обучения сети является подбор начальных значений для параметров нейронной сети. Существует несколько подходов к решению этого вопроса. Так, начальное приближение для весов сети выбирается случайным образом. Вопрос об инициализации центров c_j требует более тонкого подхода, так как положение функции в пространстве входных данных тесно связано с классифицирующей способностью сети. Естественно в таком случае ориентироваться на характер обучающих данных, и в качестве начального значения центра функции выбрать среднее арифметическое всех значений входных переменных, т.е. разместить функцию равноудаленно от всех точек пространства входных данных. Дальнейший подбор центров будет произведен при работе адаптивного метода подбора параметров так же, как и общее количество центров.

Адаптивный алгоритм был сформулирован только для гауссовской функции $(b=1)$ с использованием обобщенной модели Ванга-Менделя [6]. В результате его реализации определяются: количество центров и их расположение в части, соответствующей условиям (множество векторов x_t) и заключениям (множество скалярных ожидаемых значений d_t). Этот алгоритм можно описать следующим образом.

1) При старте с первой пары данных (x_1, d_1) создается первый кластер с центром $c_1=x_1$. Принимается, что $w_1=d_1$ и что мощность множества $L_1=1$. Пусть r обозначает предельное эвклидово расстояние между вектором x и центром, при котором данные будут трактоваться как принадлежащие к созданному кластеру (то есть максимальный радиус функции). Для сохранения общности решения принимается, что в момент начала обучения существует M кластеров с центрами $c_1, c_2, ..., c_M$ и соответствующие им значения w_i и L_i ($i=1,2,...,M$).
2) После считывания t-ой обучающей пары (x_t, d_t) рассчитываются расстояния между вектором x_t и всеми существующими центрами $||x_t-c_i||$ для i=1, 2, ..., M и выбирается центр, ближайший к x_t. Допустим, что ближайший центр — это c_{it}. В таком случае в зависимости от значения $||x_t-c_{it}||$ может возникнуть одна из двух ситуаций:
   - если $||x_t-c_{it}||>r$, то создается новый кластер $c_{M+1}=x_t$, причем $w_{M+1}(t)=d_t$, $L_{M+1}(t)=1$. Параметры созданных до этого кластеров не

изменяются, т.е. $w_i(t)=w_i(t-i)$, $L_i(t)=L_i(t-i)$ для $i=1, 2, ..., M$. Количество кластеров M увеличивается на *1 ($M \leftarrow M+1$)*;

- если $||x_t-c_{it}||<=r$, то данные включаются в *it*-й кластер, параметры которого следует уточнить в соответствии с формулами:

$$w_{it}(t)=w_{it}(t-1)+d_t, \quad (9.4)$$

$$L_{it}(t)=L_{it}(t-1)+1, \quad (9.5)$$

$$c_{it}(t)=\frac{c_{it}(t-1)L_{it}(t-1)+x_t}{L_{it}(t)}. \quad (9.6)$$

тогда как остальные кластеры не изменяются, т.е. при $i \neq it$ $w_i(t)=w_i(t-1), L_i(t)=L_i(t-1)$ и $c_i(t)=c_i(t-1),$ для $i=1,2,...,M$.

В другой версии алгоритма фиксируется положение центров c_{it} после инициализации, и их координаты уже не изменяются. Во многих случаях такой прием улучшает результаты адаптации.

3) После уточнения параметров нечеткой системы функция, аппроксимирующая входные данные системы, определяется в виде:

$$f(x)=\frac{\sum_{i=1}^{M} w_i(t)\exp\left(-\frac{\|x-c_i(t)\|^2}{\sigma^2}\right)}{\sum_{i=1}^{M} L_i(t)\exp\left(-\frac{\|x-c_i(t)\|^2}{\sigma^2}\right)}, \quad (9.7)$$

При повторении перечисленных этапов алгоритма до $t=p$ с уточнением каждый раз значения M,пространство данных разделяется на M кластеров, при этом мощность каждого из них определяется как $L_i=L_i(t)$, центр - как $c_i=c_i(t)$, а значение приписанной ему накопленной функции d -как $w_i = w_i(t)$.

Этот алгоритм называется самоорганизующимся, поскольку разделение пространства данных на кластеры происходит самостоятельно и без участия человека, в соответствии с заданным значением порога r. При малом значении r количество кластеров возрастает, в результате чего аппроксимация данных становится более точной, однако это достигается за счет более сложной функции и увеличения объема необходимых вычислений при одновременном ухудшении обобщающих свойств сети. Если значение r слишком велико, то вычислительная сложность уменьшается, однако возрастает погрешность аппроксимации. При подборе оптимальной величины порога r должен соблюдаться компромисс между точностью отображения и вычислительной сложностью. Как правило, оптимальное значение r подбирается методом проб и ошибок с использованием вычислительных экспериментов, по существу rэто максимальное значение параметра σ функции Гаусса. Таким образом, следует подобрать оптимальные значения σ и затем взять максимальное из них. Оптимальные значения σможно подобрать в соответствии с формулой (3.8), то есть так же, как для радиально-базисных сетей.

Следует обратить внимание, что алгоритм самоорганизации нечёткой сети позволяет одновременно определять как параметры сети, так и ее структуру (количество нейронов скрытого слоя). Его реализация подобна модели Ванга- Менделя, описываемой формулой(9.3), в которой можно выделить центры c_i, соответствующие множеству векторов x, и коэффициенты w_i, связанные с положением центров через последовательность заданных функций $\{d\}$. В связи с накопительным характером формирования параметров w_i (формула(9.4)), в знаменателе выражения(9.7) суммирование производится с весами L_i, отражающими количество уточнений параметров конкретных групп данных (размерность кластера).

Нейронная сеть Ванга-Менделя может быть представлена как частный случай нечёткой продукционной сети TSK, в случае, когда полином в заключении правил вывода содержит только свободный член. В этой связи, алгоритмы обучения и методы модификации структуры для сети TSK, могут быть использованы и для сети Ванга-Менделя.

9.3 Нечёткая сетьTSK

Структура нечёткой сети TSK основана на системе нечёткого вывода Такаги- Сугэно- Канга. При этом в качестве функции фуззификации для каждой переменной x_j используется обобщённая функция Гаусса:

$$\mu_A(x_j)=\frac{1}{1+\left(\frac{x_j-c_j}{\sigma_j}\right)^{2b_j}} \tag{9.8}$$

Для агрегации условия i-го правила в системе вывода TSK используется операция алгебраического произведения:

$$\mu_A^{(i)}(x)=\prod_{j=1}^{N}\left[\frac{1}{1+\left(\frac{x_j-c_j^{(i)}}{\sigma_j^{(i)}}\right)^{2b_j^{(i)}}}\right]. \tag{9.9}$$

При M правилах вывода агрегирование выходного результата сети производится по формуле (9.10), которую можно представить в виде:

$$y(x)=\frac{1}{\sum_{i=1}^{M} w_i}\sum_{i=1}^{M} w_i y_i(x), \tag{9.10}$$

где $y_i(x)=p_{i0}+\sum_{j=1}^{N} p_{ij}x_j$, агрегация импликации. Присутствующие в этом выражении веса w_i интерпретируются как компоненты $\mu_A^{(i)}(x)$,

определённые формулой (9.9). При этом формуле (9.10) можно сопоставить многослойную структуру сети, изображённую на рисунке 9.2.

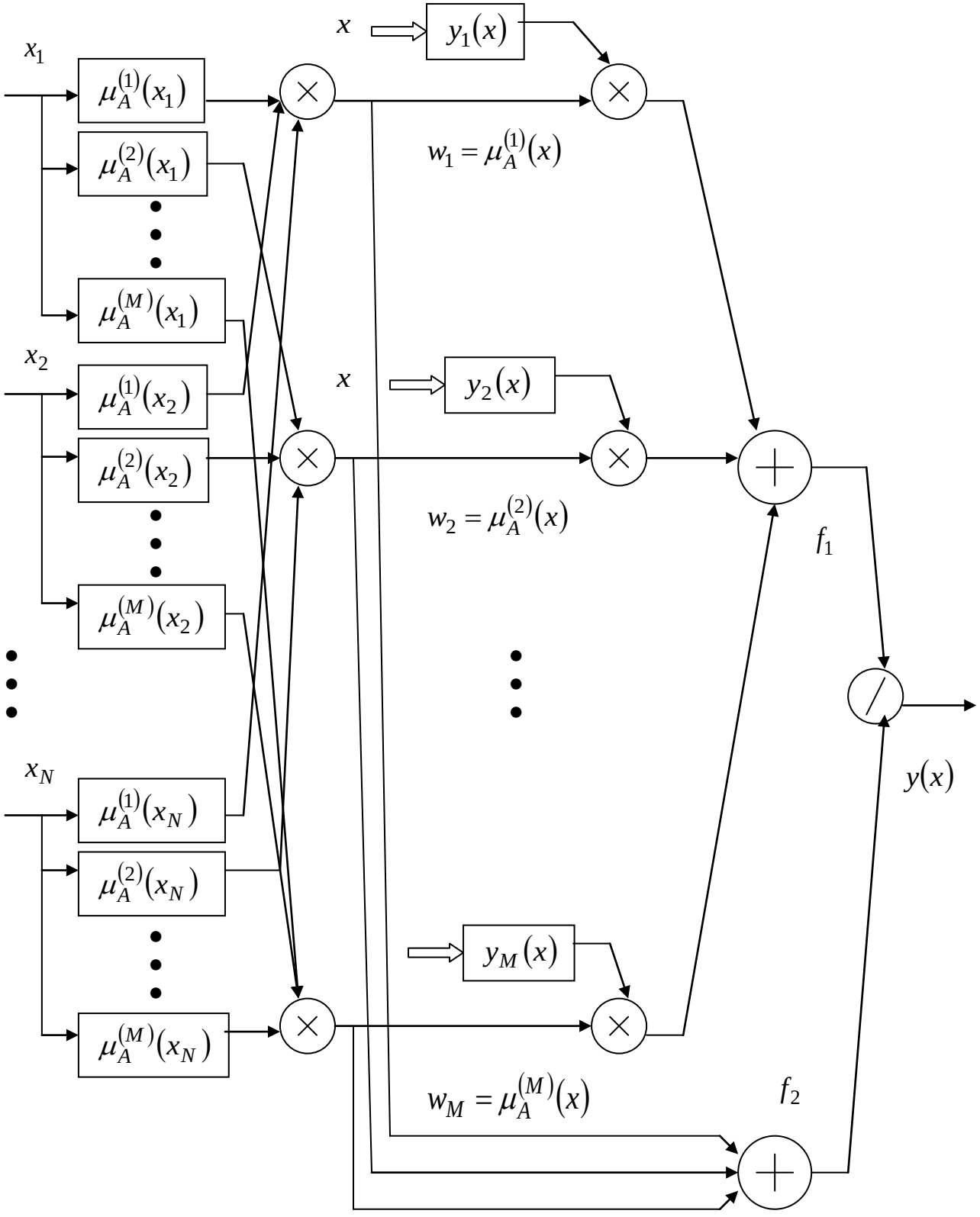


Рисунок 9.2 Структура нечёткой нейронной сети TSK.

В такой сети выделяется пять слоёв:

- Первый слой выполняет раздельную фуззификацию каждой переменной $x_j(j=1,2,...,N)$, определяя для каждого i-го правили вывода значение коэффициента принадлежности $\mu_A^{(i)}(x_j)$ в соответствии с применяемой функцией фуззификации. Это параметрический слой с параметрами $(c_j^{(i)}, \sigma_j^{(i)}, b_j^{(i)})$, подлежащими адаптации в процессе обучения.
- Второй слой выполняет агрегирование отдельных переменных x_j, определяя результирующее значение коэффициента принадлежности $w_i = \mu_A^{(i)}(x)$ для вектора x в соответствии с формулой (9.9). Этот слой непараметрический.
- Третий слой представляет собой генератор функции TSK, рассчитывающий значения $y_i(x) = p_{i0} + \sum_{j=1}^{N} p_{ij} x_j$. В этом слое также происходит умножение сигналов $y_i(x)$ на значения w_i, сформированные в предыдущем слое. Это параметрический слой, в котором адаптации подлежат линейные веса p_{ij} для $i=1,2,...,M$ и $j=1,2,...,N$, определяющие функцию следствия модели TSK.
- Четвёртый слой составляют два нейрона- сумматора, один из которых рассчитывает взвешенную сумму сигналов $y_i(x)$, а второй определяет сумму весов $\sum_{i=1}^{M} w_i$. Это непараметрический слой.
- Пятый слой состоит из одного выходного нейрона- это нормализующий слой, в котором веса подвергаются нормализации в соответствии с формулой (9.10). Выходной сигнал $y(x)$ определяется выражением, соответствующим зависимости (9.10),

$$y(x) = f(x) = \frac{f_1}{f_2}. \qquad (9.11)$$

Это также непараметрический слой.

При уточнении функциональной зависимости (9.10) для сети TSKполучаем:

$$y(x) = \frac{1}{\sum_{i=1}^{M}\left[\prod_{j=1}^{N} \mu_A^{(i)}(x_j)\right]} \sum_{i=1}^{M}\left[\prod_{j=1}^{N} \mu_A^{(i)}(x_j)\right]\left[p_{i0} + \sum_{j=1}^{N} p_{ij} x_j\right]. \qquad (9.12)$$

Если принять, что в конкретный момент времени параметры условия зафиксированы, то функция $y(x)$ является линейной относительно переменных $x_j\ (j=1,2,...,N)$.

При наличии N входных переменных каждое правило формирует $N+1$ переменных p_{ij} линейной зависимости TSK. При M правилах вывода это даёт $M(N+1)$ линейных параметров сети. В свою очередь каждая функция принадлежности использует три параметра, подлежащих адаптации. Так как каждая переменная x_j характеризуется собственной функцией принадлежности, то мы получим $3MN$ нелинейных параметров. В сумме это даёт $M(4N+1)$ линейных и нелинейных параметров, значения которых должны подбираться в процессе обучения сети.

9.4 Гибридный алгоритм обучения нечеткой сети TSK

Гибридный алгоритм обучения может применяться как для сетей Ванга-Менделя, так и для сетей TSK. Сеть Ванга- Менделя может при этом трактоваться как сеть TSK, у которой все параметры p_{ij}, кроме p_{i0}, равны нулю.

В гибридном алгоритме подлежащие адаптации параметры разделяются на две группы: линейных параметров p_{ij} третьего слоя и параметров нелинейной функции принадлежности первого слоя. Уточнение параметров проводится в два этапа:

На первом этапе при фиксации определенных значений параметров функции принадлежности путем решения системы линейных уравнений рассчитываются линейные параметры (в первом цикле – это значения, полученные в результате инициализации). При известных значениях функции принадлежности зависимость (9.12) можно представить в линейной форме [2]:

$$y(x)=\sum_{i=1}^{M} w'_i \left(p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \right), \qquad (9.13)$$

где

$$w'_i = \frac{\prod_{j=1}^{N} \mu_A^{(i)}(x_j)}{\sum_{i=1}^{M} \left[\prod_{j=1}^{N} \mu_A^{(i)}(x_j) \right]} = const \qquad (9.14)$$

для *i=1, 2, ..., M*. При *p* обучающих выборках *(x(t), d(t)) (t= 1, 2, ..., p)* и замене выходного сигнала сети ожидаемым значением *d(t)* получим систему из *p* линейных уравнений вида

$$\begin{bmatrix} w'_{11} & w'_{11}x_1^{(1)} & \dots & w'_{11}x_N^{(1)} & \dots & w'_{1M} & w'_{1M}x_1^{(1)} & \dots & w'_{1M}x_N^{(1)} \\ w'_{21} & w'_{21}x_1^{(2)} & \dots & w'_{21}x_N^{(2)} & \dots & w'_{2M} & w'_{2M}x_1^{(2)} & \dots & w'_{2M}x_N^{(2)} \\ & & \dots & & \dots & & \dots & & \\ w'_{p1} & w'_{p1}x_1^{(p)} & \dots & w'_{p1}x_N^{(p)} & \dots & w'_{pM} & w'_{pM}x_1^{(p)} & \dots & w'_{pM}x_N^{(p)} \end{bmatrix} \begin{bmatrix} p_{10} \\ \dots \\ p_{1N} \\ \dots \\ p_{M0} \\ \dots \\ p_{MN} \end{bmatrix} = \begin{bmatrix} d^{(1)} \\ d^{(2)} \\ \dots \\ d^{(p)} \end{bmatrix}$$

, (9.15)

где w'_{ti} обозначает уровень активации (вес) условия *i-го* правила при предъявлении *t-го* входного вектора *x*. Это выражение можно записать в сокращенной матричной форме

$$\boldsymbol{Ap} = \boldsymbol{d}\,, \tag{9.16}$$

Размерность матрицы*A*равна $p \times (N+1)M$, при этом количество строк значительно больше количества столбцов $(N+1)M$.

При помощи псевдоинверсии матрицы *A*решение можно получить за один шаг:

$$p = A^{+}d\,, \tag{9.17}$$

где A^{+} псевдоинверсия матрицы *A*. Псевдоинверсия матрицы *A*заключается в проведении декомпозиции *SVD* с последующим сокращением её размерности.

На втором этапе после фиксации значений линейных параметров p_{ij} рассчитываются фактические выходные сигналы *y(t)* сети для *t=1,2, ..., p*, для чего используется линейная зависимость

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(p)} \end{bmatrix} = Ap, \tag{9.18}$$

и следом за ними – вектор ошибки *ε=y-d*. Сигналы ошибок направляются через подключенную сеть по направлению ко входу сети (обратное распространение) вплоть до первого слоя, где могут быть рассчитаны компоненты градиента целевой функции относительно конкретных параметров $\left(c_j^{(i)}, \sigma_j^{(i)}, b_j^{(i)}\right)$. После формирования вектора градиента параметры уточняются с использованием одного из градиентных методов обучения, например, метода наискорейшего спуска.

$$c_j^{(i)}(t+1) = c_j^{(i)}(t) - \eta_c \frac{\partial E(t)}{\partial c_j^{(i)}} \tag{9.19}$$

$$\sigma_j^{(i)}(t+1) = \sigma_j^{(i)}(t) - \eta_\sigma \frac{\partial E(t)}{\partial \sigma_j^{(i)}} \tag{9.20}$$

$$b_j^{(i)}(t+1) = b_j^{(i)}(t) - \eta_b \frac{\partial E(t)}{\partial b_j^{(i)}} \tag{9.21}$$

После уточнения нелинейных параметров вновь запускается процесс адаптации линейных параметров функции TSK (первый этап) и нелинейных параметров (второй этап). Этот цикл повторяется вплоть до стабилизации всех параметров процесса. Формулы (9.19) – (9.21) требуют расчёта градиента целевой функции принадлежности и для одной пары обучающих данных *(x,d)* принимают значения:

$$\frac{\partial E}{\partial c_j^{(i)}} = \left(y(x) - d\right)\sum_{i=1}^{M}\left[p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \right]\frac{\partial w_i'}{\partial c_j^{(i)}} \tag{9.22}$$

$$\frac{\partial E}{\partial \sigma_j^{(i)}} = \left(y(x) - d\right)\sum_{i=1}^{M}\left[p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \right]\frac{\partial w_i'}{\partial \sigma_j^{(i)}} \tag{9.23}$$

$$\frac{\partial E}{\partial b_j^{(i)}} = \left(y(x) - d\right)\sum_{i=1}^{M}\left[p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \right]\frac{\partial w_i'}{\partial b_j^{(i)}} \tag{9.24}$$

$$\frac{\partial w_k'}{\partial c_j^{(i)}} = \frac{\delta_{ki} m(x_j) - l(x_j)}{\left[m(x_j)\right]^2} \prod_{s=1, s\neq j}^{N}\left[\mu_A^{(i)}(x_s)\right] \frac{\left[\dfrac{2b_j^{(i)}}{\sigma_j^{(i)}}\left(\dfrac{x_j - c_j^{(i)}}{\sigma_j^{(i)}}\right)^{2b_j^{(i)}-1}\right]}{\left[1 + \left(\dfrac{x_j - c_j^{(i)}}{\sigma_j^{(i)}}\right)^{2b_j^{(i)}}\right]^2} \tag{9.25}$$

$$\frac{\partial w_k'}{\partial \sigma_j^{(i)}} = \frac{\delta_{ki} m(x_j) - l(x_j)}{\left[m(x_j)\right]^2} \prod_{s=1, s\neq j}^{N}\left[\mu_A^{(i)}(x_s)\right] \frac{\left[\dfrac{2b_j^{(i)}}{\sigma_j^{(i)}}\left(\dfrac{x_j - c_j^{(i)}}{\sigma_j^{(i)}}\right)^{2b_j^{(i)}}\right]}{\left[1 + \left(\dfrac{x_j - c_j^{(i)}}{\sigma_j^{(i)}}\right)^{2b_j^{(i)}}\right]^2} \tag{9.26}$$

$$\frac{\partial w_k'}{\partial b_j^{(i)}} = \frac{\delta_{ki} m(x_j) - l(x_j)}{\left[m(x_j)\right]^2} \prod_{s=1, s\neq j}^{N}\left[\mu_A^{(i)}(x_s)\right] \frac{\left[-2\left(\dfrac{x_j - c_j^{(i)}}{\sigma_j^{(i)}}\right)^{2b_j^{(i)}} \ln\left(\dfrac{x_j - c_j^{(i)}}{\sigma_j^{(i)}}\right)\right]}{\left[1 + \left(\dfrac{x_j - c_j^{(i)}}{\sigma_j^{(i)}}\right)^{2b_j^{(i)}}\right]^2}, \tag{9.27}$$

для $k=1,2,..,M$, где δ_{ki} обозначает дельту Кронекера, $l(x_j) = \prod_{s=1}^{N} \mu_A^{(i)}(x_s)$, $m(x_j) = \sum_{i=1}^{M}\left[\prod_{s=1}^{N} \mu_A^{(i)}(x_s)\right]$.

При практической реализации гибридного метода обучения нечетких сетей доминирующим фактором их адаптации считается первый этап, на котором веса p_{ij} подбираются с использованием псевдоинверсии за один шаг. Для уравновешивания его влияния второй этап (подбор нелинейных параметров градиентным методом) многократно повторяется в каждом цикле.

9.5 Алгоритм обратного распространения ошибки для сети TSK

Нечеткая нейронная сеть TSK имеет многослойную структуру с прямым распространением сигнала, значение выхода которой можно изменять, корректируя параметры элементов слоев, что позволяет для обучения этой сети использовать алгоритм обратного распространения ошибки[2]. Для этого потребуется обучающая выборка в виде пар *(x, d)*, где $x=[x_1, \ldots, x_N]^T$ – это входной вектор, а d – эталонный сигнал. Задача заключается в такой коррекции параметров сети, описанной выражением 9.12 чтобы мера погрешности, задаваемая выражением:

$$E = \frac{1}{2}\left(y(x) - d\right)^2, \tag{9.28}$$

была минимальной.

Если применяется простейший метод наискорейшего спуска, то соответствующие формулы адаптации принимают форму:

$$p_{ij}(t+1) = p_{ij}(t) - \eta_p \frac{\partial E(t)}{\partial p_{ij}}, \tag{9.29}$$

Параметры $\left(c_j^{(i)}, \sigma_j^{(i)}, b_j^{(i)}\right)$корректируются в соответствии с формулами (9.19)-(9.21).

Формулы (9.19-9.21 и 9.29) требуют расчета градиента целевой функции относительно параметров функции принадлежности. Окончательный вид этих формул зависит от используемого определения функции погрешности на выходе сети, так и от формы функции принадлежности. Например, при использовании функции Гаусса

$$\mu_A(x_i) = \exp\left[-\left(\frac{x_i - c_i}{\sigma_i}\right)^2\right], \tag{9.30}$$

Соответствующие формулы градиента целевой функции для одной пары обучающих данных (x, d) принимают вид [6]:

$$\frac{\partial E}{\partial p_{ij}} = \left(y - d\right) w_k x_j,$$

$$\frac{\partial E}{\partial c_{ij}} = \left(y - d\right)\left(y_i(x) - y\right) w_k \frac{2\left(x_j - c_{ij}\right)}{\left(\sigma_{ij}\right)^2},$$

$$\frac{\partial E}{\partial \sigma_{ij}} = (y - d)(y_i(x) - y) w_k \frac{2(x_j - c_{ij})^2}{(\sigma_{ij})^3}, \quad (9.31)$$

где $w_k = \frac{\prod_{j=1}^{N} \mu_A^{(i)}(x_j)}{\sum_{i=1}^{M}\left[\prod_{j=1}^{N} \mu_A^{(i)}(x_j)\right]}$, $y_i(x) = p_{i0} + \sum_{j=1}^{N} p_{ij} x_j$ (9.32)

Если использовать функцию Гаусса, представленную формулой (9.8), то корректировку следует проводить по формулам (9.25-9.27).

Несмотря на сложную структуру приведенных формул, выражающих компоненты вектора градиента, они позволяют аналитически определить величины, необходимые для уточнения параметров нечеткой сети.

Метод наискорейшего спуска имеет линейную сходимость, поскольку в нем используются только слагаемые первого порядка при разложении целевой функции в ряд Тейлора. Указанный недостаток, а также резкое замедление минимизации в ближайшей окрестности точки оптимального решения, когда градиент принимает очень малые значения, делают алгоритм наискорейшего спуска низкоэффективным. Повысить эффективность удается путем эвристической модификации выражения, определяющего направление градиента.

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

$$par_{ij}(t+1) = par_{ij}(t) - \eta \cdot \frac{\partial E(t)}{\partial par_{ij}(t)} + \alpha(par_{ij}(t) - par_{ij}(t-1)) \quad (9.33)$$

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

9.6 Модификация структуры с несколькими выходами

Традиционно, модели нечётких нейронных сетей, в том числе и нейронные сети TSK и Ванга-Менделя конструируются с едиственным выходом, то есть осуществляют преобразование вектора входных значений в число. Такая конфигурация сетей называется MISO — MultipleInput, SingleOutput.

Тем не менее, не существует препятствий и для создания MIMO (MultipleInput, MultipleOutput)-конфигураций данных сетей, то есть сетей, осуществляющих преобразование входного вектора значений произвольной длины в выходной вектор также произвольной длины.

Очевидно, MIMO-конфигурация делает сеть более гибкой и подходящей для решения более широкого спектра задач. В силу этого было решено включить поддержку создания сетей с несколькими выходами в разрабатываемыйнейроимитатор.

На рисунке 9.3 приведён пример нейронной сети TSK с двумя входами, тремя правилами и двумя выходами. Данную структуру нетрудно обобщить и на произвольное число выходов. Аналогичный подход применим и к нейронной сети Ванга-Менделя. На рисунке 9.4 приведена структура нечёткой сети Ванга-Менделя с двумя входами, тремя правилами и двумя выходами.

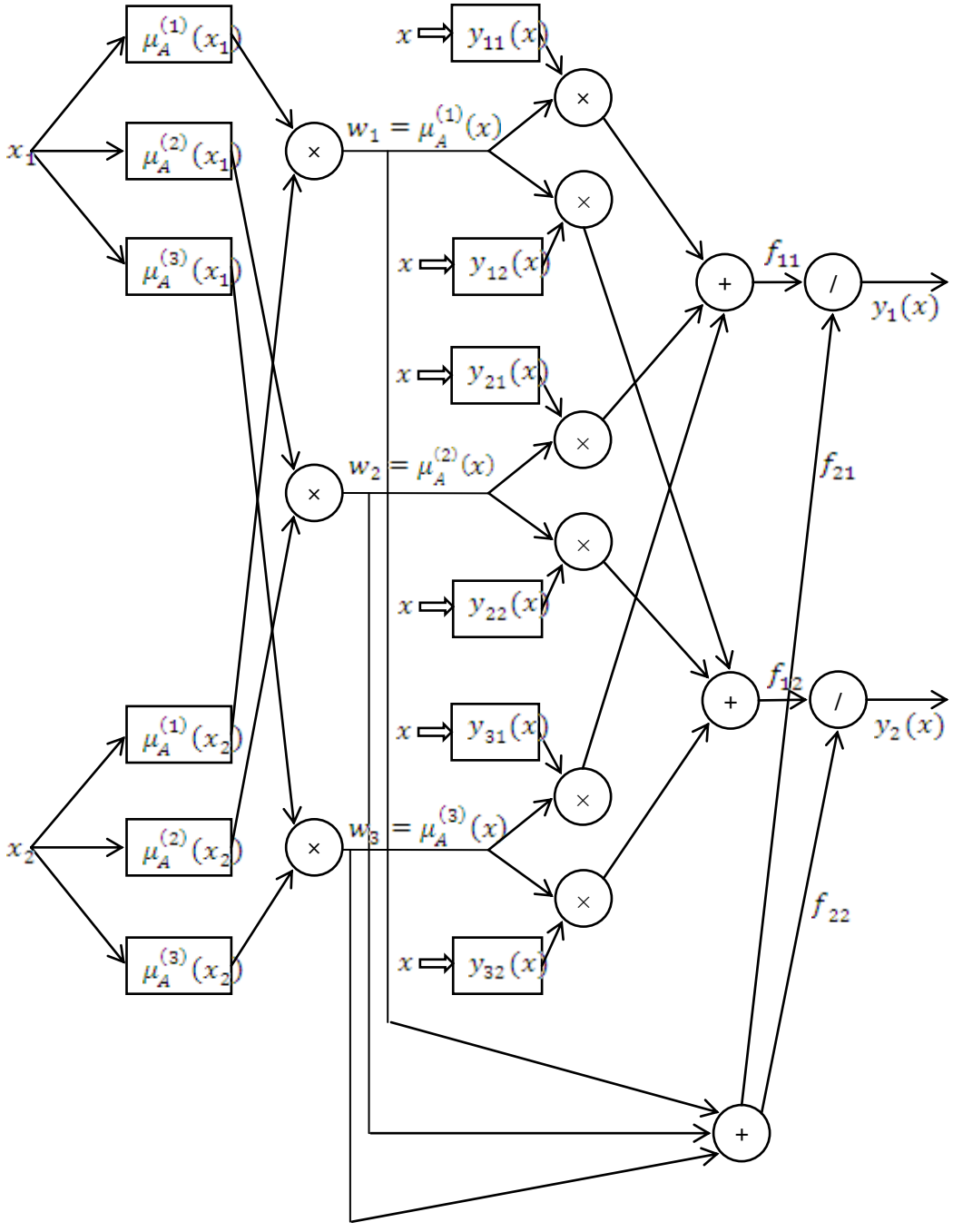


Рисунок 9.3Структура нечёткой нейронной сети TSK с несколькими выходами

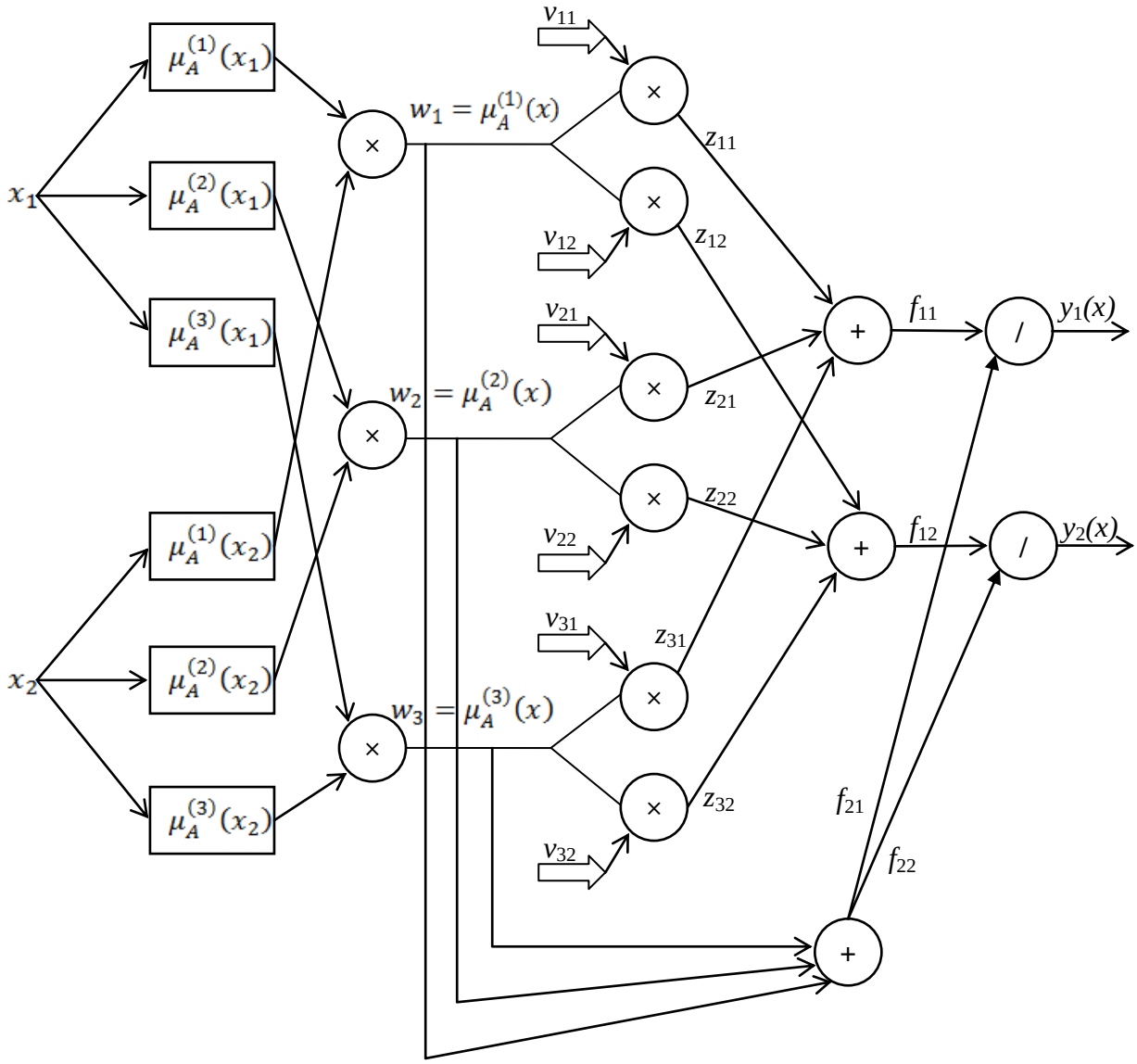


Рисунок 9.4. Структура нейронной нечёткой сети Ванга-Менделя с двумя выходами.

В отличие от стандартной четырёхслойной конфигурации сети Ванга-Менделя, данная модель состоит из пяти слоёв нейронов.

Первый и второй слои предложенной модели не отличаются от стандартной сети Ванга-Менделя и реализуют фуззификацию по функции Гаусса N – мерного входного вектора x, а также агрегацию условий правил вывода в соответствии с операцией T -нормы в форме произведения.

Третий слой реализует операцию импликации в форме произведения, это параметрический слой, в процессе обучения подбираются параметры v_{is}, где i - это номер правила вывода, а s - индекс выходного нейрона.

Четвёртый слой осуществляет агрегирование M правил вывода (первый и второй нейрон) и генерацию нормализующего сигнала (третий нейрон). Число агрегирующих нейронов в данном слое равно числу выходов сети. В отличие от стандартной модели это непараметрический слой.

Пятый слой состоит из двух выходных нейронов и выполняет нормализацию, формируя выходной сигнал y_s. Это непараметрический слой.

Предложенная структура нейронной сети легко модифицируется на случай с числом выходов, больше чем два. Таким образом, нейронная сеть реализует функции, которые можно записать в виде:

$$y_s(x) = \frac{1}{\sum_{i=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(i)}(x_j)\right]} \cdot \left(\sum_{i=1}^{M} v_{is} \cdot \left[\prod_{j=1}^{N}\mu_A^{(i)}(x_j)\right]\right) \qquad (9.34)$$

Слой дефуззификации в приведённой выше модели нейронной сетей позволяет сформировать на выходе сети чёткое значение, что необходимо при решении задачи прогнозирования или задачи аппроксимации функций. Для решения задач классификации, идентификации или распознавания чёткое значение на выходе сети не является обязательным, а при пересекающихся классах объектов вообще не имеет смысла, так как на выходе сети требуется получить степень принадлежности предъявленного входного вектора к конкретному классу.

9.7 Использование комбинированных правил вывода

При наличии N входных переменных каждое правило формирует $\mathrm{N}+1$ переменных $\mathrm{p_{ij}}$ линейной зависимости TSK. При M правилах вывода это даёт $(\mathrm{N}+1)*\mathrm{M}$ линейных параметров сети. В свою очередь каждая функция принадлежности использует три параметра, подлежащих адаптации. Так как, каждая переменная $\mathrm{x_j}$ характеризуется собственной функцией принадлежности, то мы получим $3*\mathrm{N}*\mathrm{M}$ нелинейных параметров. В сумме это даёт $\mathrm{M}*(4*\mathrm{N}+1)$ линейных и нелинейных параметров, значения которых должны подбираться в процессе обучения сети.

На практике для уменьшения количества адаптируемых параметров оперируют меньшим количеством независимых функций принадлежности для отдельных переменных, руководствуясь правилами, в которых комбинируются функции принадлежности различных переменных. Если принять, что каждая переменная $\mathrm{x_j}$ имеет K различных функций принадлежности к классам, то максимальное количество правил, которое можно создать при их комбинировании, составит $\mathrm{M}=\mathrm{K^N}$ (при трёх функциях принадлежности, распространяющихся на две переменные, это $3^2=9$ правил вывода). Таким образом, суммарное количество нелинейных параметров сети при M правилах вывода уменьшается с $3*\mathrm{N}*\mathrm{M}$ в общем случае до $3*\mathrm{N}*\mathrm{M}^{\frac{1}{\mathrm{N}}}$. Количество линейных параметров при подобной модификации остаётся без изменений.

Рассматриваемые до сих пор структуры нейронных сетей содержали столько правил, сколько и классов принадлежности. Покажем, как можно уменьшить число классов принадлежности для нейронной сети Ванга-

Менделя с двумя входами, тремя правилами вывода и одним выходом, до двух, оставив число правил неизменным и равным трём (рисунок 9.5).

Как и в случае с несколькими выходами, данная модификация применима как к сети Ванга-Менделя, так и к сети TSK.

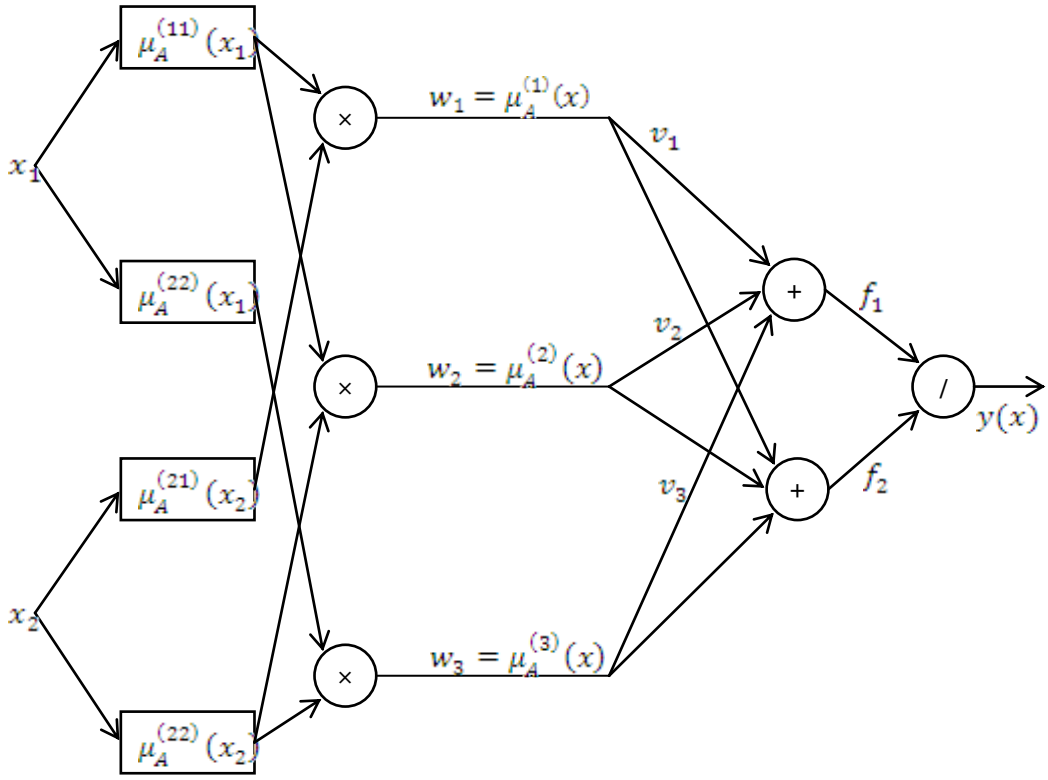


Рисунок 9.5 Структура нейронной сети Ванга-Менделя с числом правил, большим числа классов принадлежности

9.8 Сети, основанные на модели нечёткого вывода Цукамото

Отдельный тип нейронных нечётких продукционных сетей, обучаемых при помощи градиентных алгоритмов составляют сети типа ANFIS (адаптивная сетевая нечёткая система вывода), предложенные Чангом в 1992 году.

Рассмотрим нечёткую сеть типа ANFIS, реализующую алгоритм вывода Цукамото и имеющую следующую базу продукционных правил [5]:

П1: если x_1 есть H_1 и x_2 есть H_2 и x_3 есть H_3, то у есть Б,

П2: если x_1 есть B_1и x_2 есть B_2 и x_3 есть H_3, то у есть С,

П3: если x_1 есть B_1 и x_2 есть B_2 и x_3 есть B_3, то у есть М,

где x_1, x_2, x_3 – входные переменные, y – выходная переменная, а H_1, H_2, H_3, B_1, B_2, B_3,*Б, С, М*– некоторые нечёткие множества с функциями принадлежности сигмоидного типа:

$$H_j(t)=\frac{1}{1+e^{b_j(t-c_j)}},\ B_j(t)=\frac{1}{1+e^{b_j(t-c_j)}},\ j=1,2,3. \qquad (9.35)$$

$$Б(t)=\frac{1}{1+e^{b_4(t-c_4+c_5)}},\ C(t)=\frac{1}{1+e^{b_4(t-c_4)}},\ M(t)=\frac{1}{1+e^{b_4(t-c_4)}}. \qquad (9.36)$$

Для определения выходной переменной будем использовать алгоритм Цукамото:

1. Подсчитаем значения истинности предпосылок для каждого правила:
$$\begin{aligned}\alpha_1 &= H_1(x_1)\wedge H_2(x_2)\wedge H_3(x_3)\\ \alpha_2 &= B_1(x_1)\wedge B_2(x_2)\wedge H_3(x_3)\\ \alpha_3 &= B_1(x_1)\wedge B_2(x_2)\wedge B_3(x_3)\end{aligned}, \quad (9.37)$$
где логическое*И* (Т-норма) обычно используется в форме минимума, а x_1, x_2, x_3 - текущие значения входов системы;
2. Для каждого правила определим выходы как обратные функции:
$$z_1 = Б^{-1}(\alpha_1) = c_4 + c_5 + \frac{1}{b_4}\ln\frac{1-\alpha_1}{\alpha_1},$$
$$z_2 = C^{-1}(\alpha_2) = c_4 + \frac{1}{b_4}\ln\frac{1-\alpha_2}{\alpha_2}, \quad (9.38)$$
$$z_3 = M^{-1}(\alpha_3) = c_4 + \frac{1}{b_4}\ln\frac{1-\alpha_3}{\alpha_3}.$$
3. Находим общий выход всей системы:
$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3}{\alpha_1 + \alpha_2 + \alpha_3}. \quad (9.39)$$

Нейронная сеть, реализующая данный алгоритм нечёткого вывода, приведена на рисунке 9.6 и относится к сетям типа ANFIS.

Данная сеть может быть описана следующим образом:

1. Слой 1. Выходы нейронов этого слоя представляют собой значения функций принадлежности при заданных значениях входов. Это параметрический слой. В процессе обучения подбираются параметры функций принадлежности для множеств H_1, H_2, H_3, B_1, B_2, B_3.
2. Слой 2. Выходами нейронов этого слоя, реализующим операцию Т-нормы, являются результаты агрегации условий (предпосылок) каждого нечёткого продукционного правила, вычисляемые по формулам (9.37). Это непараметрический слой.
3. Слой 3. Это непараметрический слой. Нейроны данного слоя, обозначенные буквой N, вычисляют следующие величины:
$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3}, \beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2 + \alpha_3}, \beta_3 = \frac{\alpha_3}{\alpha_1 + \alpha_2 + \alpha_3}. \quad (9.40)$$
4. Слой 4. Нейроны данного слоя вычисляют значения выходов нейронов по формулам (9.41):
$$y_1' = Б^{-1}(\alpha_1) = c_4 + c_5 + \frac{1}{b_4}\ln\frac{1-\alpha_1}{\alpha_1},$$
$$y_2' = C^{-1}(\alpha_2) = c_4 + \frac{1}{b_4}\ln\frac{1-\alpha_2}{\alpha_2}, \quad (9.42)$$
$$y_3' = M^{-1}(\alpha_3) = c_4 + \frac{1}{b_4}\ln\frac{1-\alpha_3}{\alpha_3}.$$

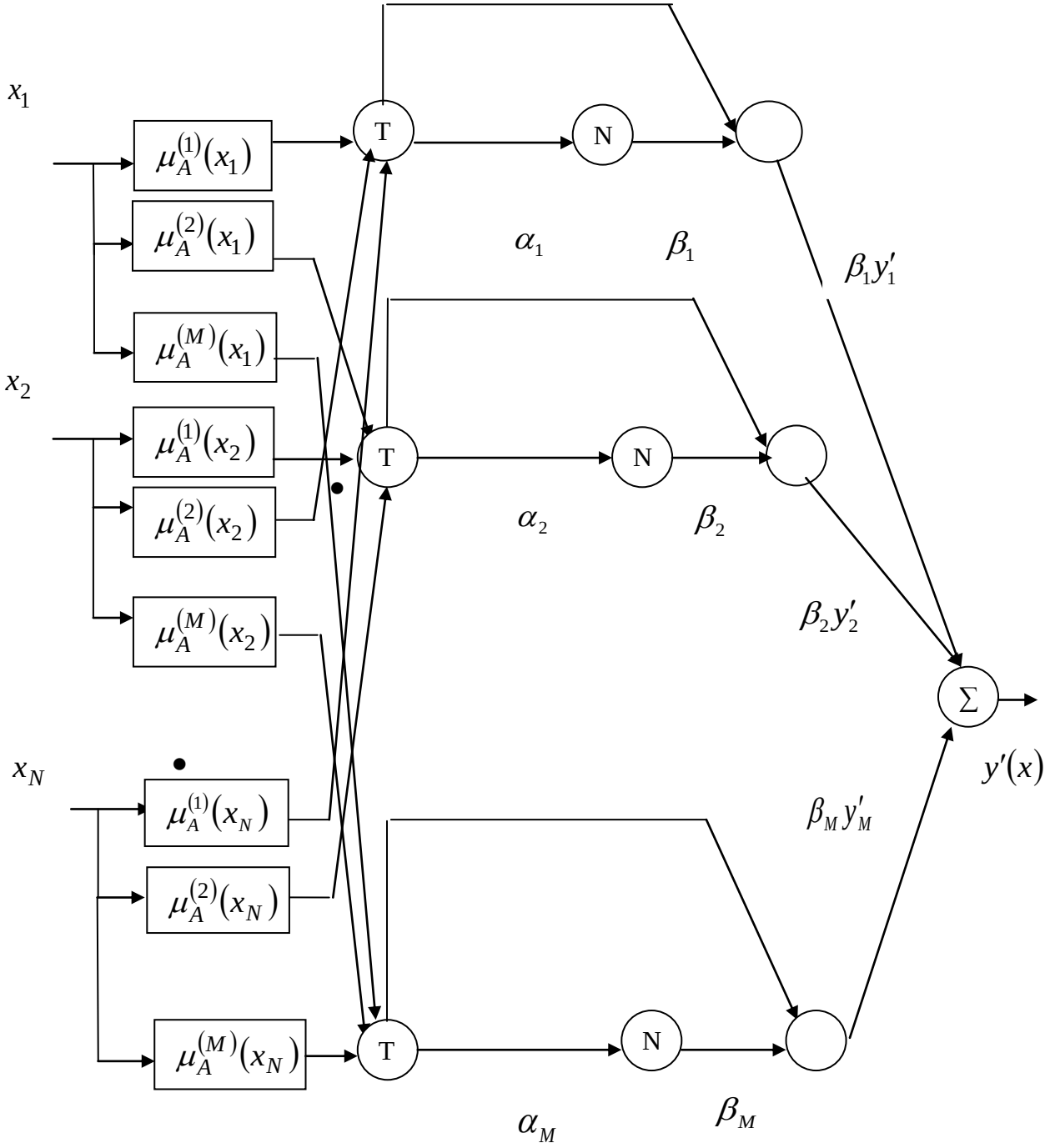


Рисунок 9.6 Структура сети типа ANFIS.

Затем приводят результаты к удобному виду:

$$\beta_1 y_1' = \beta_1 B_1^{-1}(\alpha_1), \beta_2 y_2' = \beta_2 B_2^{-1}(\alpha_2), \beta_3 y_3' = \beta_3 B_3^{-1}(\alpha_3).$$

Это параметрический слой. В процессе обучения подбираются параметры функций принадлежности для множеств Б, С и М.

5. Слой 5. Единственный нейрон этого слоя вычисляет выход сети:

$$y' = \beta_1 y_1' + \beta_2 y_2' + \beta_3 y_3'. \qquad (9.43)$$

Это непараметрический слой.

Настройка параметров сети производится в соответствии с градиентным алгоритмом наискорейшего спуска по следующим формулам:

$$b_4(t+1) = b_4(t) - \eta \frac{\partial E(t)}{\partial b_4(t)} = b_4(t) - \frac{\eta}{b_4^2}(z_0(t) - d(t)) \frac{\alpha_1 + \alpha_2 - \alpha_3}{\alpha_1 + \alpha_2 + \alpha_3}, \quad (9.44)$$

$$c_4(t+1) = c_4(t) - \eta \frac{\partial E(t)}{\partial c_4(t)} = c_4(t) + \eta(z_0(t) - d(t)) \frac{\alpha_1 + \alpha_2 + \alpha_3}{\alpha_1 + \alpha_2 + \alpha_3} = c_4(t) + \eta(z_0(t) - d(t)), \quad (9.45)$$

$$c_5(t+1) = c_5(t) - \eta \frac{\partial E(t)}{\partial c_5(t)} = c_5(t) - \eta(z_0(t) - d(t)) \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3}. \quad (9.46)$$

Где $d(t)$ - ожидаемое значение выхода на *t*-ом шаге обучения.

10 Гибридные нечёткие нейронные сети

10.1 Обучение нечётких нейронных сетей

Для обучения нечётких нейронных сетей используются два вида обучения – с учителем и без учителя. Алгоритмы обучения с учителем основаны на минимизации целевой функции, как правило, функции погрешности. Алгоритмы обучения без учителя, называемые алгоритмами самоорганизации, состоят в группировании (кластеризации) входных данных. Рассмотрим подробнее эти алгоритмы.

При алгоритме с учителем целевая функция погрешности, оценивающая данную конфигурацию сети, задается извне – в зависимости от того, какую цель преследует обучение. Далее сеть начинает постепенно модифицировать свою конфигурацию – состояние всех своих синаптических весов – таким образом, чтобы минимизировать эту функцию. В итоге, в процессе обучения сеть все лучше справляется с возложенной на нее задачей.

Погрешность сети зависит от конфигурации сети – совокупности всех ее синаптических весов. Но эта зависимость не прямая, а опосредованная, так как непосредственные значения весов скрыты от внешнего наблюдателя. Для него сеть – своего рода черный ящик, и оценивать ее работу он может, лишь основываясь на ее поведении, т.е. на том, каковы значения выходов сети при данных входах. Иными словами, в общем виде целевая функция имеет вид: $E(w) = E\{x^t, d^t, y(x^t, w)\}$. Здесь $\{x^t, d^t\}$ – набор примеров (т.е. пар входов-выходов), на которых обучается нейронная сеть, а $\{y(x^t, w)\}$ – реальные значения выходов нейросети, зависящие от конкретных значений ее синаптических весов.

Алгоритм самоорганизации приписывает значения входного вектора x к соответствующей группе данных, представляемых центом c_i. Базовая форма алгоритма позволяет точно определять положение центров c_i соответствующих групп данных (кластеров), на которые подразделяется многомерное пространство. Можно также, полученные значения центров использовать при обучении с учителем в качестве начальных значений, что существенно ускоряет процесс обучения и гарантирует сходимость решения. В дальнейшем рассмотрим некоторые алгоритмы обучения нечётких сетей.

10.2 Гибридный нечёткий многослойный персептрон

В случае линейной неразделимости классов, то есть когда классы пересекаются, а обучающие примеры приводят к неустойчивому поведению алгоритма обучения персептрона, имеет смысл использовать гибридную сеть, включающую в себя слой с нечёткой самоорганизацией и многослойный персептрон.

Основная идея построения нечеткого персептрона и алгоритма его обучения заключается в уменьшении влияния векторов, лежащих в зоне перекрытияклассов, на изменение весовых коэффициентов [2].

На рисунке 10.1 приведена структура нечеткого многослойного персептрона. Он состоит из двух частей: нейронов «нечеткого слоя» и собственно многослойного персептрона. Функции активации нейронов «нечеткого слоя» такой сети являются радиальными базисными функциями (в виде функции Гаусса), моделирующими функции принадлежности. Эти нейроны предназначены для определения степеней принадлежности компонентов входных векторов (которые могут быть и нечеткими). На выходах нейронов этого слоя формируются коэффициенты в требуемой для дальнейшего распознавания форме. Выходы нейронов «нечеткого слоя» употребляются в качестве входов традиционного многослойного персептрона.

Если на вход сети подается $x=[x_1,x_2,...,x_n]^T$, то на выходе «нечеткого слоя» формируется вектор, состоящий из степеней принадлежности x к конкретным центрам (радиально базисным функциям); $\mu(x)=[\mu_1(x),\mu_2(x),...,\mu_n(x)]^T$. Конкретные компоненты $\mu_i(x)$ рассчитываются таким образом, чтобы удовлетворять условию нормализации $\sum_{i=1}^{m}\mu_i(x^{(t)})=1$ для каждого вектора $x^t, t=1,...,p$, где p – число векторов в обучающей выборке.

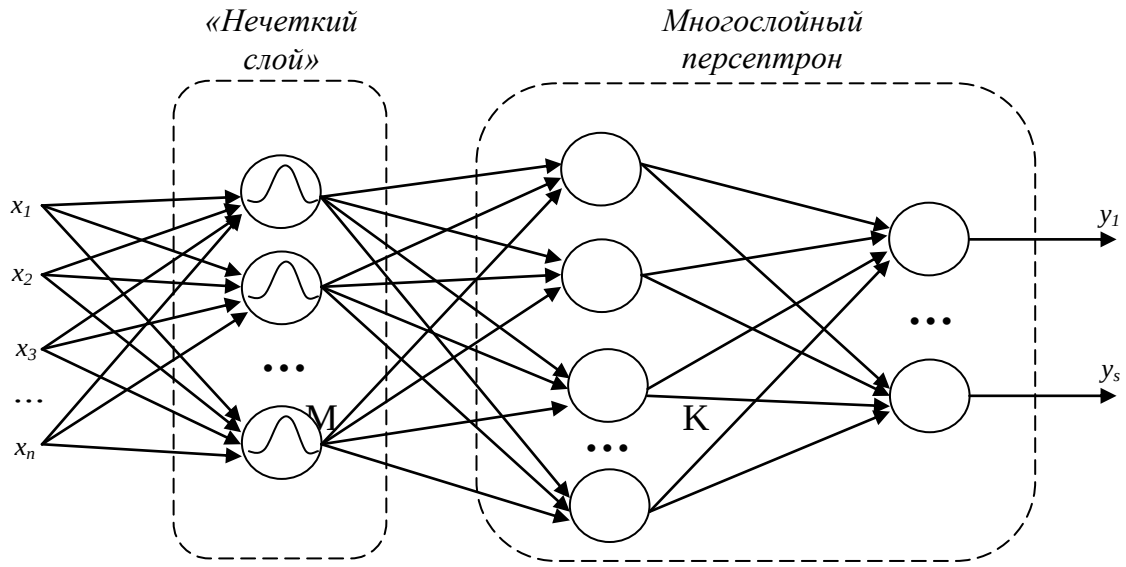


Рисунок 10.1 Структура нечеткого многослойного персептрона

Выходы нечёткого многослойного персептрона трактуются как степени принадлежности предъявленного объекта соответствующему классу.

С учётом ярко выраженной двухкомпонентной структуры гибридной сети для её обучения применяется алгоритм, состоящий из двух этапов. На первом из них проводится обучение самоорганизующегося слоя, состоящее в подборе центров. Для этого может быть использован алгоритм *C-means*или другие подобные алгоритмы (например, алгоритм *Густавсона-Кесселя*[2]). По завершении первого этапа, начинается второй этап обучения, на котором уточняются только веса персептронной компоненты. Это обычное обучение

многослойного персептрона, для которого входом является множество коэффициентов принадлежности вектора *x* к центрам самоорганизующегося слоя. В зависимости от типа решаемой задачи выходом сети может быть код класса, к которому принадлежит входной вектор *x*, либо ожидаемое значение *d* выходного вектора.

Для обучения весов скрытых слоёв и выходного слоя нечёткого многослойного персептрона могут быть использованы градиентные алгоритмы обучения или эвристические алгоритмы обучения, используемые для обучения чёткого многослойного персептрона.

Для уточнения всех параметров нечёткого многослойного персептрона, включая параметры функции Гаусса нейронов слоя фуззификации можно использовать градиентный алгоритм наискорейшего спуска или его модификацию с моментами по аналогии с обучением радиально-базисных сетей. При этом алгоритм самоорганизации может быть использован для первичной инициализации центров функции Гаусса.

10.3 Алгоритм нечёткой самоорганизации C-means

Алгоритм самоорганизации приписывает вектор *x* к соответствующей группе данных, представляемых центром c_i, с использованием обучения конкурентного типа подобно сетям Кохонена. При обучении этого типа процесс самоорганизации становится возможным при предъявлении вектора *x*.

Допустим, что в сети существует M нечётких нейронов с центрами в точках $c_i (i = 1,2,...,M)$. Начальные значения этих центров могут быть выбраны случайным образом из областей допустимых значений соответствующих компонентов векторов x $(t = 1,2,...,p)$, использованных для обучения. Пусть функция фуззификации задана в форме обобщенной функции Гаусса, выраженной формулой (9.8).

Подаваемый на вход сети вектор x будет принадлежать к различным группам, представляемым центрами c_i, в степени u_{it}, причем $0 \le u_{it} \le 1$, а суммарная степень принадлежности ко всем группам, очевидно, равна единице. Поэтому

$$\sum_{i=1}^{M} u_{it} = 1 \qquad (10.1)$$

для $t = 1,2,...,p$. Функцию погрешности, соответствующую такому представлению, можно определить как сумму частных погрешностей принадлежности к центрам c_i с учетом степени принадлежности u_{it} Следовательно

$$E = \sum_{i=1}^{M} \sum_{t=1}^{P} \left(u_{it}\right)^m \left\| c_i - x_t \right\|^2 , \qquad (10.2)$$

где m — это весовой коэффициент, который принимает значения из интервала [1, ∞), на практике часто принимают $m=2$. Цель обучения с

самоорганизацией состоит в таком подборе центров c_i, чтобы для заданного множества обучающих векторов $x^{(t)}$ обеспечить достижение минимума функции (10.2) при одновременном соблюдении условий ограничения (10.1). Таким образом возникает задача минимизации нелинейной функции (10.2) с p ограничениями типа (10.1). Решение этой задачи можно свести к минимизации функции Лагранжа, определенной в виде [2].

$$LE = \sum_{i=1}^{M}\sum_{t=1}^{P}\left(u_{it}\right)^m \left\|c_i - x_t\right\|^2 + \sum_{t=1}^{P}\lambda_t\left(\sum_{i=1}^{M}u_{it} - 1\right), \qquad (10.3)$$

где $\lambda_t\left(t = 1,2,...,p\right)$ - это множители Лагранжа. Доказано, что решение задачи (10.3) можно представить в виде

$$c_i = \frac{\sum_{t=1}^{P}\left(u_{it}\right)^m x_t}{\sum_{t=1}^{P}\left(u_{it}\right)^m}, \qquad (10.4)$$

$$u_{it} = \frac{1}{\sum_{k=1}^{M}\left(\frac{d_{it}^2}{d_{kt}^2}\right)^{\frac{1}{m-1}}}, \qquad (10.5)$$

где d_{it} - это эвклидово расстояние между центром c_i и вектором x_t, $d_{it} = \left\|c_i - x_t\right\|$. Поскольку точные значения центров c_i в начале процесса не известны, алгоритм обучения должен быть итерационным:

1. Выполнить случайную инициализацию коэффициентов u_{it}, выбирая их значения из интервала [0,1] таким образом, чтобы соблюдать условие (10.1).

2. Определить *M* центров c_i в соответствии с (10.4).

3. Рассчитать значение функции погрешности в соответствии с (10.2). Если её значение ниже установленного порога, либо если уменьшение этой погрешности относительно предыдущей итерации пренебрежимо мало, то закончить вычисления. Иначе, перейти к п.4.

4. Рассчитать новые значения u_{it} по формуле (10.5) и перейти к п.2.

Многократное повторение итерационной процедуры ведёт к достижению минимума функции *E*, который необязательно будет глобальным.

10.4 Алгоритм разностного группирования

Алгоритм разностного группирования данных является модификацией алгоритма пикового группирования, предложенного З.Егером и Д.Филёвым. В качестве меры плотности размещения векторов x_t используются так называемые пиковые функции. В алгоритме разностного группирования в качестве потенциальных центров пиковых функций используются обучающие векторы x_i. Пиковая функция задаётся в виде [2]:

$$D(x_i)=\sum_{t=1}^{p}\exp\left(-\frac{\|x_i-x_t\|^{2b}}{\left(\frac{r_a}{2}\right)^2}\right). \qquad (10.6)$$

Значение коэффициента r_a определяет сферу соседства векторов. При большой плотности векторов x_t вокруг точки x_i, значение функции $D(x_i)$ велико, следовательно, точка является «удачным» кандидатом в центры. После расчёта значений пиковой функции для всех точек x_i, отбирается та, для которой значение функции $D(x_i)$ оказалось максимальным. Именно эта точка становится первым отобранным центром c_1. Выбор следующего центра возможен после исключения предыдущего и всех точек, лежащих в его окрестности. Для этого пиковая функция переопределяется в виде:

$$D_{new}(x_i)=D(x_i)-D(c_1)\exp\left(-\frac{\|x_i-c_1\|^{2b}}{\left(\frac{r_b}{2}\right)^2}\right). \qquad (10.7)$$

Обычно при выборе новой константы r_b соблюдается условие $r_b \geq r_a$. Пиковая функция $D_{new}(x_i)$ принимает нулевое значение при $x_i = c_1$ и близка к нулю в ближайшей окрестности этой точки.

После модификации пиковой функции отыскивается следующая точка x_i, для которой величина $D_{new}(x_i)$ оказывается максимальной. Эта точка становится следующим центром c_2. Процесс поиска очередного центра продолжается после исключения всех предыдущих центров и их окрестностей. Процесс инициализации центров завершается в момент фиксации всех центров, предусмотренных начальными условиями.

Если существует множество обучающих данных в виде пар векторов (x_t, d_t) так, как это происходит при обучении с учителем, то для нахождения центров, соответствующих множеству векторов d_t, достаточно сформировать расширенную версию векторов x_t в виде $x_t \leftarrow [x_t, d_t]$. При этом определяют расширенные версии центров c_i с размерностью, равной сумме размерностей векторов x_t и d_t. Тогда в описании центров можно выделить часть v_i, соответствующую векторам x_t (первые Nкомпонентов) и остаток q_i, соответствующий вектору d_t. Таким образом, можно получить центры, как входных переменных, так и ожидаемых выходных значений $c_i = [v_i, q_i]$. В случае применения нечётких правил с одним выходом векторы d_t и q_i сводятся к скалярным величинам.

Независимо от способа реализации алгоритма обучения сеть с нечёткой самоорганизацией выполняет нечёткое группирование данных путём приписывания их к различным центрам на основании коэффициентов

принадлежности, значения которых изменяются от нуля до единицы. Это означает, что каждый вектор *x*представляется множеством центров, причём влияние каждого из них на значение вектора различно и зависит от величины коэффициента принадлежности. Если считать, что вектор x_t представляется *M*центрами $c_i\left(i=1,2,...,M\right)$, а принадлежность вектора к каждому центру задана коэффициентом u_{it} (формула 10.73), то реконструкция исходного вектора x_t происходит согласно выражению:

$$x_t=\sum_{i=1}^{M}u_{it}c_i\,. \qquad (10.8)$$

В этом состоит существенное различие нечёткой самоорганизации от классической самоорганизации Кохонена, при которой реконструкция вектора выполняется на базе одного центра, ближайшего к данному вектору, путём простого приписывания ему значения этого центра.

10.5 Нейронные нечёткие сети на основе нечётких нейронов

Возможны различные варианты нечётких нейронных сетей данного типа. Функции активации нейронов являются чёткими, а нечёткими могут быть входы, выходы и (или) веса нейронов. В таких нечётких нейронах в отличие от обычнойнейронной нечёткой сети [5]:

- входные сигналы и веса можно комбинировать с использованием *T*-нормы,*S*- нормы или некоторого другого непрерывного оператора;
- результаты комбинаций всех входов и их весов могут агрегироваться с использованием *T*-нормы,*S*- нормы или некоторой другой непрерывной функции;
- функция активации f может быть любой другой (не только сигмоидальной функцией).

Входы, выходы и веса гибридной нечёткой нейронной сети (обычно представляющие собой степени принадлежности к нечётким множествам) являются вещественные числа из интервала [0,1].

Основными разновидностями нечётких нейронов, реализующих нечёткие операции и используемых для создания гибридных нейронных нечётких сетей, являются:

- нечёткий нейрон «И»;
- нечёткий нейрон «ИЛИ»;
- нечёткий нейрон «Импликация - ИЛИ»;
- нечёткие нейроны для реализации композиционных правил вывода.

На рисунке 10.2 приведена структура нечёткого нейрона «И». Сигналы x_j и веса w_j комбинируются с использованием *S*- нормы:

$$r_j=S\left(w_j,x_j\right),\ j=1,2,...,n\,, \qquad (10.9)$$

а выходное значение агрегируется с помощью операции *T*-нормы:

$$y=T\left(r_j,r_k\right)=T\left(S\left(w_1,x_1\right),S\left(w_2,x_2\right),...,S\left(w_n,x_n\right)\right). \qquad (10.10)$$

Если *T- min, S – max,* то нечёткий нейрон «И» реализует min-max–композицию:

$$y = \min\{w_1 \vee x_1, w_2 \vee x_2, \ldots, w_n \vee x_n\}. \quad (10.11)$$

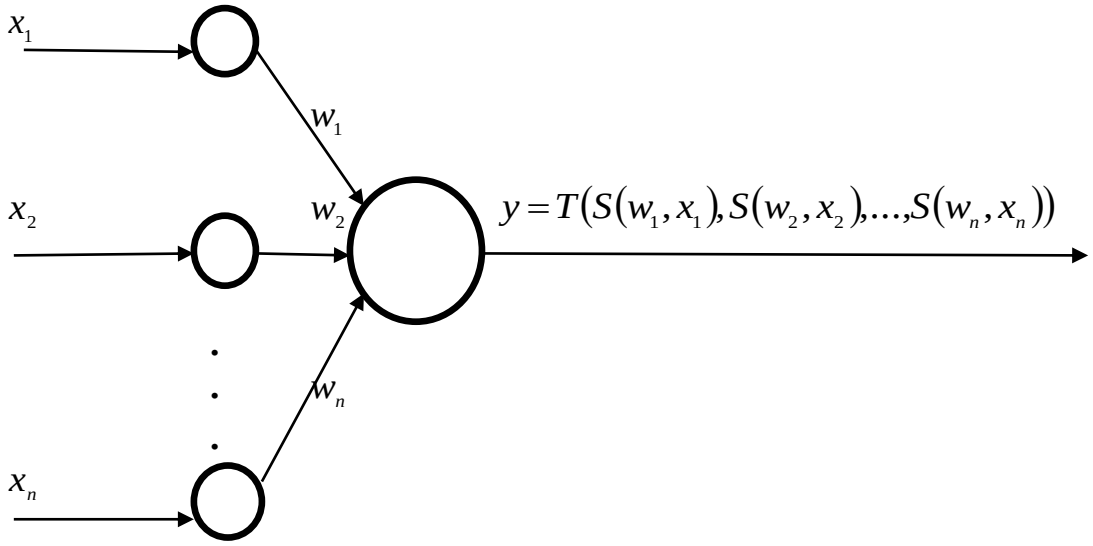


Рисунок 10.2 Структура нечёткого нейрона «И».

На рисунке 10.3 приведена структура нечёткого нейрона «ИЛИ». Сигналы x_j и веса w_j комбинируются с использованием *T*- нормы:

$$r_j = T(w_j, x_j),\ j = 1,2,\ldots,n, \quad (10.12)$$

а выходное значение агрегируется с помощью операции *S*-нормы:

$$y = S(r_j, r_k) = S(T(w_1, x_1), T(w_2, x_2), \ldots, T(w_n, x_n)). \quad (10.13)$$

Если *T- min, S – max,* то нечёткий нейрон «ИЛИ» реализует max-min–композицию:

$$y = \max\{w_1 \wedge x_1, w_2 \wedge x_2, \ldots, w_n \wedge x_n\}. \quad (10.14)$$

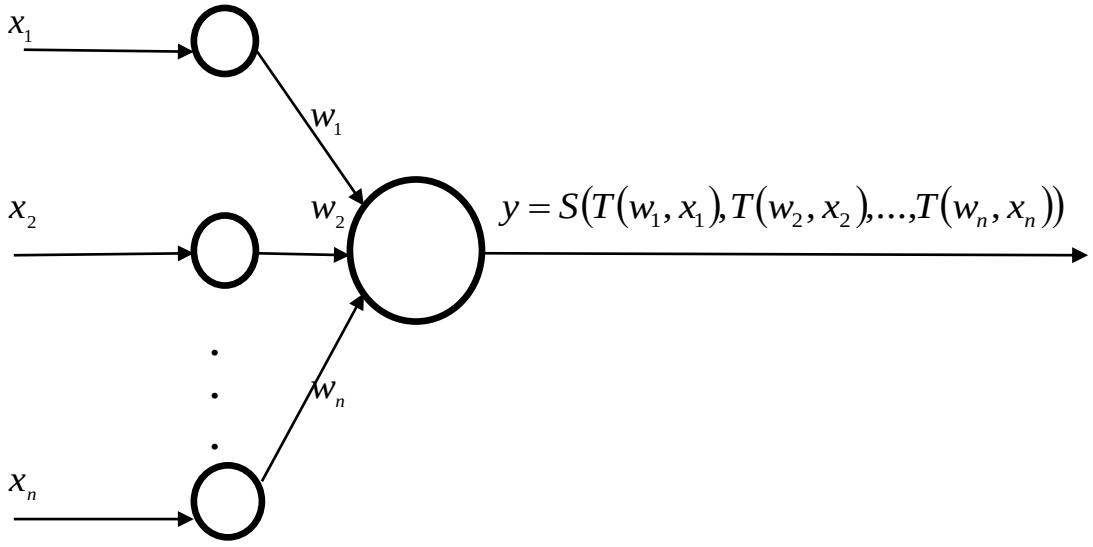


Рисунок 10.3 Структура нечёткого нейрона «ИЛИ».

На рисунке 10.4 приведена структура нечёткого нейрона «Импликация-ИЛИ». Сигналы x_j и веса w_j комбинируются с использованием операции нечёткой импликацииI:

$$r_j = I(w_j, x_j), x_j \to w_j,\ j = 1,2,\ldots,n, \tag{10.15}$$

а выходное значение образуется агрегированием значений r_j посредством операции *S*-нормы:

$$y = S(r_j, r_k) = S(I(w_1, x_1), I(w_2, x_2),\ldots, I(w_n, x_n)). \tag{10.16}$$

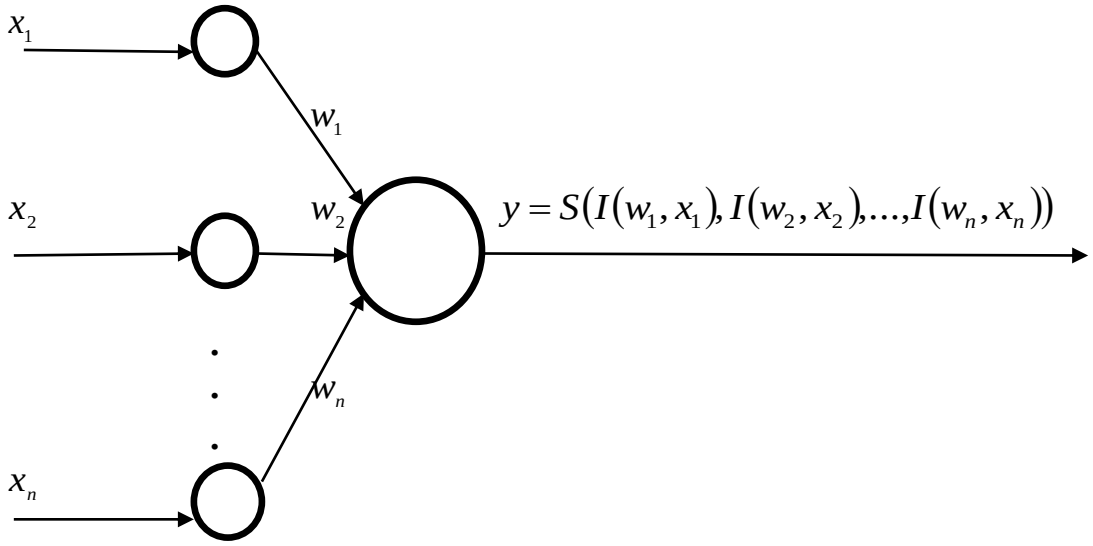
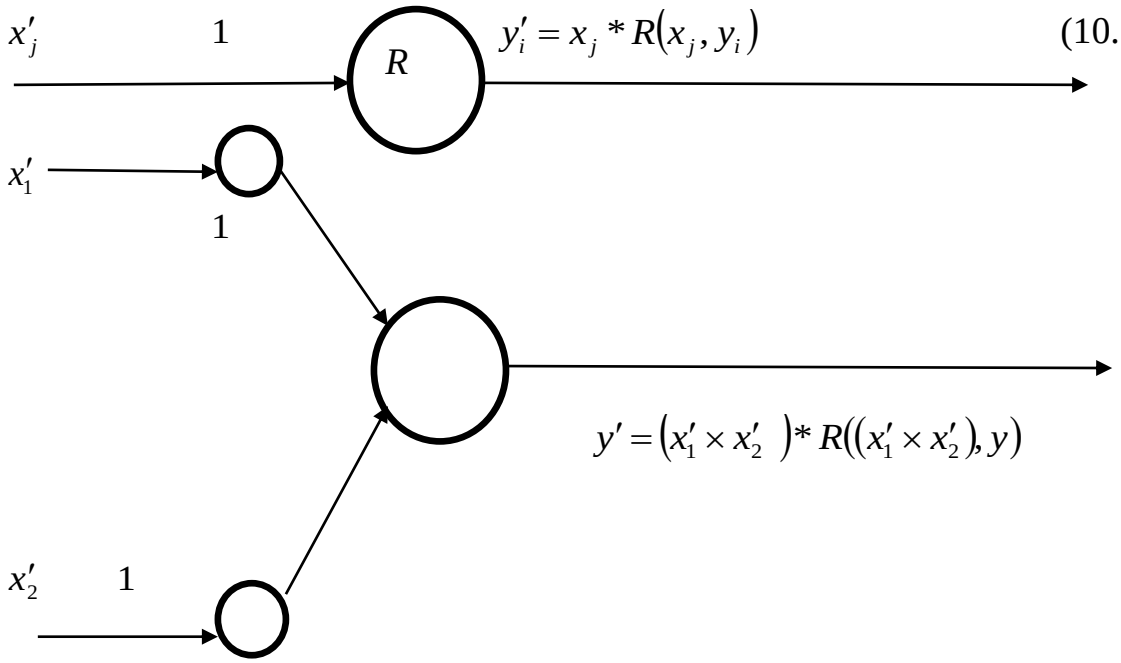


Рисунок 10.4 Структура нечёткого нейрона «Импликация -ИЛИ».

На рисунке 10.5 приведены структуры нечёткого нейрона для реализации композиционных правил вывода.



(10.17)

Рисунок 10.5 Примеры структур нечётких нейронов для композиционных правил вывода.

$R((x_1' \times x_2'), y)$ - нечёткое отношение на основе Ти Snорм или других нечётких операций.

В качестве примеров построения и использования гибридных нечётких нейронных сетей на основе нейронов, реализующих нечёткие операции, можно привести следующие сети:

- гибридные нейронечёткие классификаторы;
- гибридные нейронные нечёткие сети для реализации деревьев классификации;
- гибридные нейронные нечёткие сети для реализации композиционных правил вывода;
- гибридные нейронные нечёткие сети для нечёткого логического вывода;
- гибридные нейронные нечёткие сети для извлечения нечётких продукционных правил из числовых данных.

10.6 Гибридный нейронечёткий классификатор

В качестве примера рассмотрим гибридный нейронечёткий классификатор.

Традиционный подход к классификации образов основан на предварительной кластеризации обучающих примеров и отнесении их к заданным классам. Сложность и ограничения при осуществлении этого предварительного этапа в большой степени обусловлены недостаточной эффективностью определения границ между кластерами. Эта проблема становится ещё более трудноразрешимой, когда число используемых параметров существенно возрастает.

В отличие от традиционного подхода нечёткая классификация допускает непрерывность границы между двумя соседними классами с наложением областей, в каждой из которых классифицируемый объект характеризуется своей степенью принадлежности. Данный подход не только подходит для многих приложений, характеризующихся нечёткими границами между классами, но также обеспечивает достаточно простое представление потенциально сложного разделения пространства признаков [5].

Для гибридного нейронечёткого классификатора типичные нечеткие правила классификации при заданных _*P*образах в виде *n*-мерных четких векторов $x^{(t)} = (x_1^{(t)}, x_2^{(t)}, ..., x_n^{(t)}), t = 1, ..., P$, относящихся к K классам, имеют следующий вид:

ЕСЛИ $x_1^{(t)}$ есть A_1 И $x_2^{(t)}$ есть A_2 И … И $x_n^{(t)}$ есть A_n, ТО $x^{(t)} = (x_1^{(t)}, x_2^{(t)}, \ldots, x_n^{(t)})$ принадлежит к классу C_{iu}, $i = 1, ..., m, u = 1, ..., K$.

A_j – лингвистические термы, характеризующие соответствующие функции принадлежности компонентов входного вектора нечётким множествам.

Уровень активизации условия *i*-го правила обозначаемый как α_i, и интерпретируемый как степень принадлежности *t*-го образа к классу C_{iu}, обычно определяется следующим образом:

$\alpha_i = T(\mu_{A_{ij}}(x_1^{(t)}), \mu_{A_{ij}}(x_2^{(t)}), ..., \mu_{A_{ij}}(x_n^{(t)}))$, где *T*– оператор *T*-нормы, обычно в виде минимума, то есть

$$\alpha_i = \min_{j=1,..,n} \left(\mu_A \left(x_j^{(t)} \right) \right), t = 1,..., P, \quad (10.18)$$

либо другой интерпретации нечёткого «И».

В результате предъявления образа осуществляется параллельное сравнение его компонентов с условиями всех *m* правил с получением уровней активизации $\alpha_i, i = 1,...,m$, затем уровни активизации всех правил агрегируются на основе заданной операции. В качестве такой операции часто используют оператор *T*-нормыили *S*-нормы.

Задача нечеткой классификации заключается в выполнении соответствующего нечеткого разделения признакового пространства.

На рисунке 10.6 представлена структура гибридного нейронечеткого классификатора.

Нейронечёткая сеть состоит из четырёх слоёв.

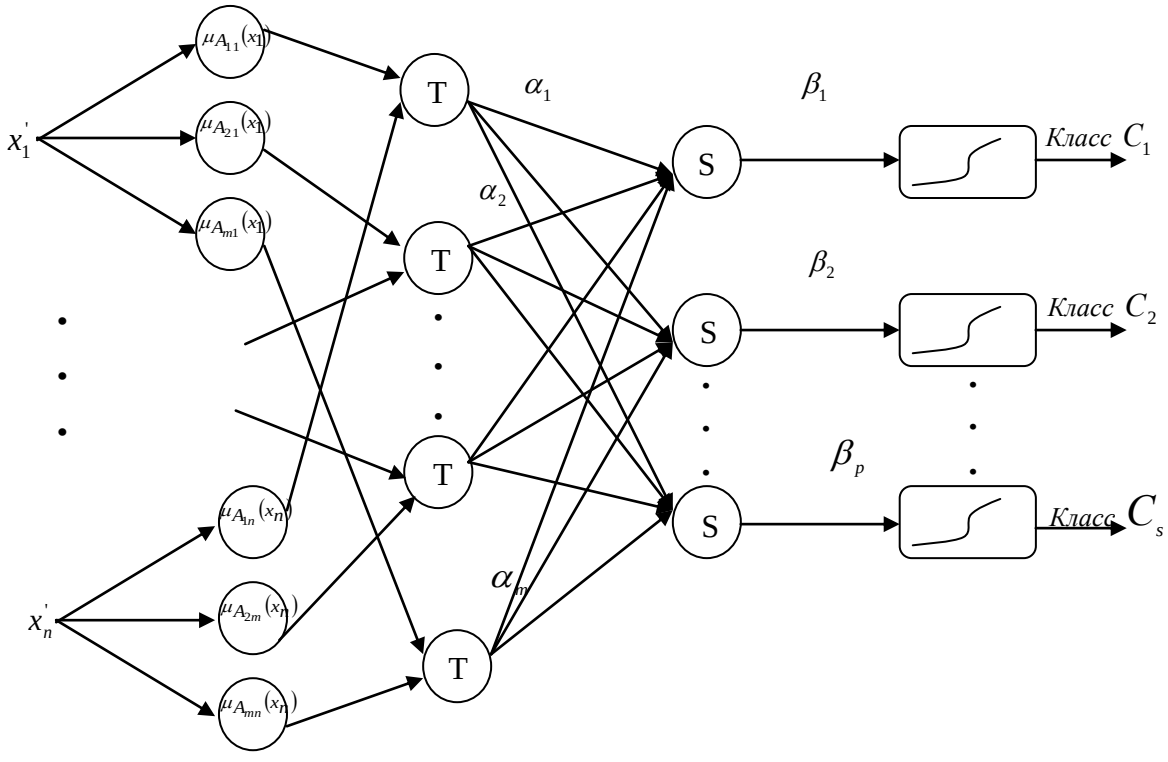


Рисунок 10.6 Структура гибридного нейронечеткого классификатора

1. Элементы первого слоя реализуют операцию фуззификации, то есть формируют степени принадлежности входных переменных к определенным для них нечетким множествам A_{ij}:

$$\mu_{A_{ji}}(x'_j) = \exp\left[-\frac{1}{2}\left(\frac{x'_j - c_{ij}}{\sigma_{ij}}\right)^2\right], \qquad (10.19)$$

где c_{ij}, σ_{ij} – параметры функции принадлежности колоколообразного типа.

Начальные значения этих параметров установлены таким образом, чтобы функции принадлежности удовлетворяли свойствам полноты, нормальности и выпуклости. Значения c_{ij} должны быть равномерно распределены в области входных векторов x. Значения данных параметров корректируется в процессе обучения сети, основанном на градиентном методе или на основе алгоритмов самоорганизации.

2. Каждый элемент второго слоя является нечётким нейроном «И»,выходной сигнал которого, представляет «силу» срабатывания нечёткого правила относительно классифицируемого объекта. Они выполняют агрегирование степеней истинности предпосылок каждого правила базы в соответствии с интерпретацией операции *T*-нормы по формуле (10.18):

$$\begin{aligned}
&\alpha_1 = \min\{\mu_{A_{11}}(x_1), \mu_{A_{12}}(x_2), \ldots, \mu_{A_{1n}}(x_n)\};\\
&\alpha_2 = \min\{\mu_{A_{21}}(x_1), \mu_{A_{22}}(x_2), \ldots, \mu_{A_{2n}}(x_n)\}; \qquad (10.20)\\
&\ldots\\
&\alpha_m = \min\{\mu_{A_{m1}}(x_1), \mu_{A_{m2}}(x_2), \ldots, \mu_{A_{mn}}(x_n)\}.
\end{aligned}$$

3. Элементы третьего слоя выполняют нормализацию и вычисляют следующие значения. Они выполняют агрегирование степеней истинности предпосылок правил базы в соответствии с операцией *S*-нормы по формулам:

$$\begin{aligned}
&\beta_1 = \sum_{i=1}^{m} \frac{\alpha_i}{\alpha_1 + \alpha_2 + \ldots + \alpha_m} \cdot w_{1i};\\
&\beta_2 = \sum_{i=1}^{m} \frac{\alpha_i}{\alpha_1 + \alpha_2 + \ldots + \alpha_m} \cdot w_{2i}; \qquad (10.21)\\
&\ldots\\
&\beta_s = \sum_{i=1}^{m} \frac{\alpha_i}{\alpha_1 + \alpha_2 + \ldots + \alpha_m} \cdot w_{si}.
\end{aligned}$$

4. Элементы четвертого слоя вычисляют значения заключений по каждому правилус использованием функций активации сигмоидного типа. Эти выходы трактуются как степени принадлежности предъявленного объекта к соответствующему классу:

$$\begin{aligned}
&y'_1 = C_1^{-1}(\alpha_1) = a_1 + \frac{1}{b_1}\ln\frac{1-\alpha_1}{\alpha_1};\\
&y'_2 = C_2^{-1}(\alpha_2) = a_2 + \frac{1}{b_2}\ln\frac{1-\alpha_2}{\alpha_2}; \qquad (10.22)\\
&\ldots
\end{aligned}$$

$$y'_s = C_s^{-1}(\alpha_s) = a_s + \frac{1}{b_s}\ln\frac{1-\alpha_s}{\alpha_s},$$

где a_i, b_i – нелинейные параметры функций принадлежности $\mu_{C_{iu}}(y)$ нечетких множеств заключений правил вида:

$$\mu_{C_{iu}}(y_i) = \frac{1}{1+\exp((b_i(y_i - a_i))} \tag{10.23}$$

В данном слое также рассчитываются выходы сети по следующим формулам:

$$\begin{aligned} &y_1 = \beta_1 y'_1 \\ &y_2 = \beta_2 y'_2 \\ &\dots \\ &y_s = \beta_s y'_s \end{aligned} \tag{10.24}$$

10.7 Алгортм обучения гибридного нейронечёткого классификатора

Для обучения нейронечёткого классификатора можно использовать алгоритмы наискорейшего спуска и алгоритм обратного распространения ошибки. В этом случае в качестве операции Т-нормы следует использовать алгебраическое произведение, иначе возникнут сложности при вычислении производной функции активации.

Пустьзадана обучающая выборка, состоящая из множества примеров $(x_1^{(t)}, x_2^{(t)}, ..., x_n^{(t)}, d_1^{(t)}, d_2^{(t)}, ..., d_s^{(t)}), t = 1, ..., P,$ где $x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)}$, значения входных переменных, $d_1^{(t)}, d_2^{(t)}, ..., d_s^{(t)}$ - эталонные значения выходных переменных в *t*-ом примере.

Настраиваемыми параметрами для данной сети являются параметры c_{ij}, σ_{ij} функции принадлежности входных переменных нечётким множествам и параметры a_i, b_i функций принадлежности $\mu_{C_{iu}}(y)$ нечетких множеств заключений (*i=1,2,..,m; j=1,2,…,n*).

Шаг1. Для каждого примера из обучающей выборки по значениям входных переменных $x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)}$ *(t=1,2,…,P)* нечёткая сеть рассчитывает значения выходных переменных $y_1^{(t)}, y_2^{(t)}, ..., y_s^{(t)}$.

Шаг 2. Вычисляется функция ошибки для всех примеров обучающей выборки:

$$E^{(t)} = \frac{1}{2}\left(y^{(t)} - d^{(t)}\right)^2, t = 1,2,\dots,K. \tag{10.25}$$

В данном случае функция ошибки может быть представлена как функция, зависящая от следующих аргументов:

$$E^{(t)} = \frac{1}{2}\left(y^{(t)}\left(c_{ij}, \sigma_{ij}, a_i, b_i, w_{ij}\right) - d^{(t)}\right)^2, t = 1,2,..., P \tag{10.26}$$

Шаг 3. Корректируются значения c_{ij}, σ_{ij} и a_i, b_i по каждому примеру обучающей выборки по следующим формулам:

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

$$\sigma_{ij}(t+1) := \sigma_{ij}(t) - \eta \frac{\partial E^{(t)}(t)}{\partial \sigma_{ij}(t)}, \qquad (10.27)$$

$$a_i(t+1) := a_i(t) - \eta \frac{\partial E^{(t)}(t)}{\partial a_i(t)},$$

$$b_i(t+1) := b_i(t) - \eta \frac{\partial E^{(t)}(t)}{\partial b_i(t)},$$

$$w_{ij}(t+1) := w_{ij}(t) - \eta \frac{\partial E^{(t)}(t)}{\partial w_{ij}(t)}$$

где t – номер итерации обучения, η - коэффициент обучения.

Шаги 1-3 повторяются довыполнения условий завершения:

- либо значение функции ошибки по каждому примеру обучающей выборки не превышает некоторого установленного порога:
  $E^{(t)} < \varepsilon, t = 1,2,..., P;$ (10.28)
- либо оценка средней суммарной погрешности по всем примерам обучения не превышает некоторого установленного порога:
  $E = \frac{1}{P} \sum_{t=1}^{P} \left(y^{(t)} - d^{(t)} \right)^2 < \varepsilon.$ (10.29)

10.8 Гибридные нейронные нечёткие сети для извлечения нечётких правил из данных

Извлечение нечётких продукционных правил из числовых данных осуществляется, как правило, в рамках конкретной задачи, например аппроксимации функций или классификации образов. Существует несколько методов извлечения правил из числовых данных. Одним из известных методов для решения задачи аппроксимации является метод, предложенный Абе С. и Лэном М.-С. Суть метода заключается в следующем [5].

Пусть неизвестная функция характеризуется одномерным выходом y и n-мерным входным вектором x. Область, на которой определена переменная y, разделена на m интервалов:

$$[y_0, y_1], (y_1, y_2], ..., (y_{m-1}, y_m], \qquad (10.30)$$

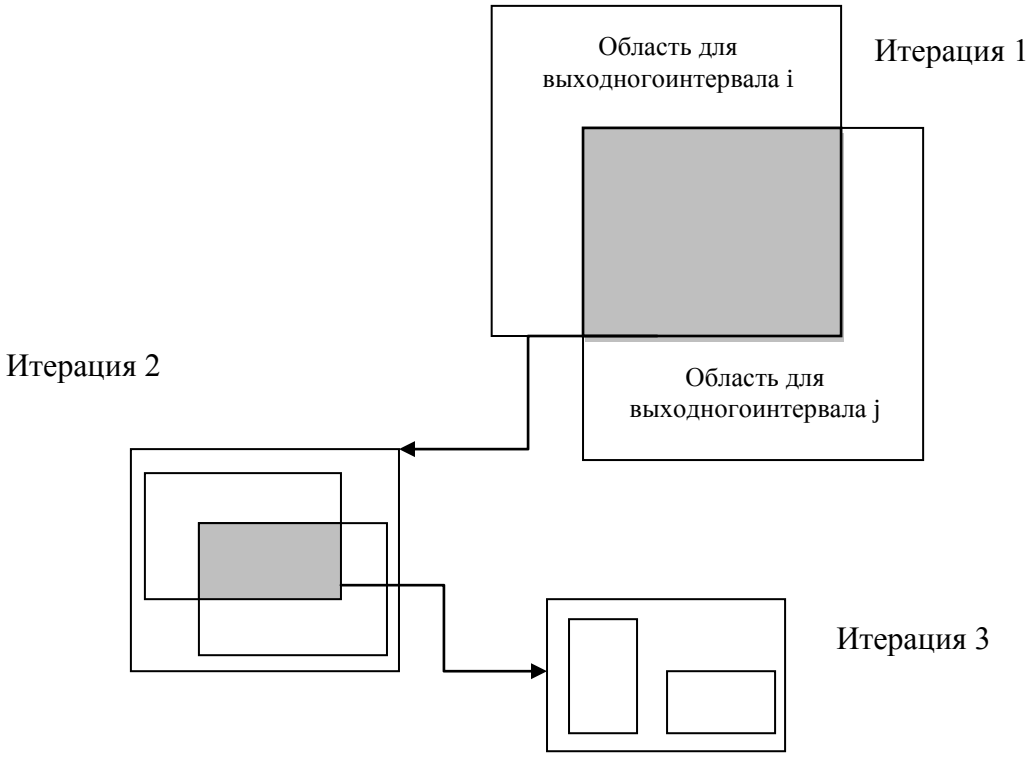
где $y_0 = M_1, y_m = M_2$ - границы области. Назовём i-ый интервал $(y_{i-1}, y_i]$ выходным интервалом i.

Используя заданные входные данные, для которых выходы находятся в выходном интервале i, рекурсивно определяется область входных значений, соответствующая выходному интервалу i. Прежде всего, находятся области

активизации, которые определяют входную область, соответствующую выходному интервалу *i*, посредством вычисления минимального и максимального значений входных данных для каждого выходного интервала.

Если область активизации для выходного интервала *i* перекрывается с областью активизации для выходного интервала *j*, то область перекрытия определяется как область запрещения.

Если входные данные для выходных интервалов *i*и/или *j* находятся внутри области запрещения, то определяется одна или две дополнительные области активизации. Если дополнительные области активизации также перекрываются, то определяется дополнительная область запрещения. Данный процесс повторяется до тех пор, пока проблема наложения областей не будет решена. Рисунок 10.7 иллюстрирует этот процесс.



.

Рис.10.7 Рекурсивное определение областей активации и запрещения

Нечёткие правила определеяются на основе областей активизации либо на основе областей активизации и соответствующих областей запрещения (если они есть).

На рисунке 10.8 показана обобщённая структура гибридной нейронной нечёткой сети, вычисляющая степени принадлежности входных данных к соответствующим выходным интервалам с последующим приведением к чёткому значению. Так, для входного вектора *x* степень принадлежности к выходным интервалам от *1*-го до *m*-ого определяется данной сетью и затем вычисляется чёткое значение *y*.

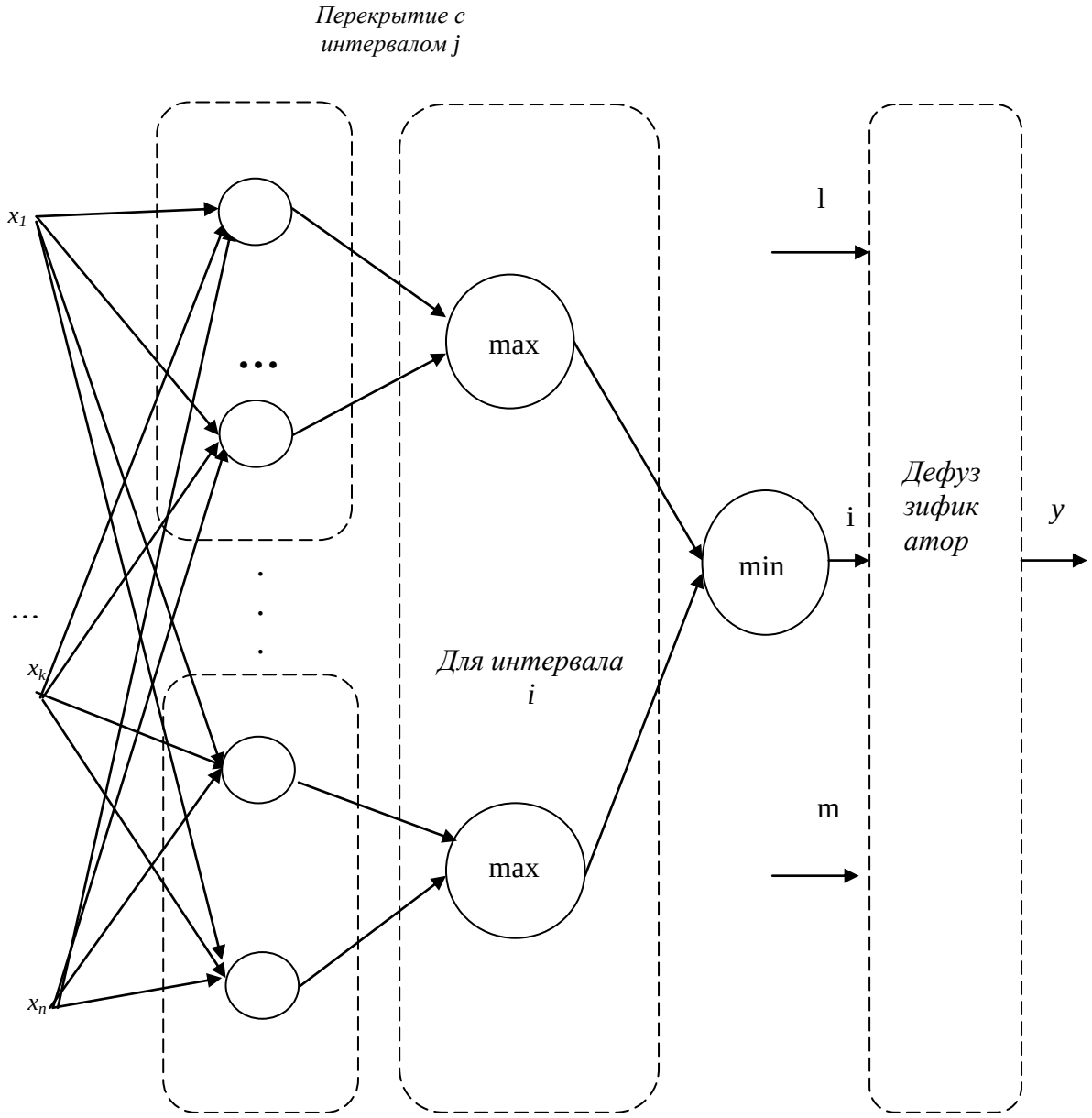


Рис.10.8 Структура гибридной нейронной нечёткой сети Абе и Лэна для извлечения нечётких правил из данных при решении задачи аппроксимации функции

Данная сеть состоит из 4 слоёв. При этом различные выходные интервалы реализуются разными элементами слоёв 2-4. Между элементами этих слоёв, реализующими различные выходные интервалы переменной *y*, отсутствует какя-либо связь.

Слой 2. Элементы этого слоя реализуют нечёткие «ЕСЛИ-ТО» правила, вычисляющие степени принадлежности переменных входного вектора *x*.

Слой 3. Элементы этого слоя вычисляют максимальные значения выходов элементов *слоя 2*, которые представляют собой степени принадлежности, полученные в результате ликвидации перекрытий между парой выходных интервалов. Число элементов *слоя 3* для выходного интервала *i* определяется по числу выходных интервалов, чьи входные

пространства перекрываются со входным протранством данного выходного интервала*i*. Поэтому, если нет перекрытия между входными пространствами выходного интервала*i*и какого-либо другого выходного интервала, сеть для выходного интервала *i*уменьшается до двух слоёв.

Слой 4. Элемент этого слоя для выходного интервала *i* определяет минимальное значение среди максимальных значений, определённых в предыдущем слое и соответствующих перекрытию между двумя выходными интервалами. Поэтому если выходной интервал *i* перекрывается только с одним выходным интервалом, то сеть для выходного интервала *i*уменьшается до трёх слоёв. Вычисление минимума в *слое 4* позволяет ликвидировать перекрытие среди более, чем двух выходных интервалов. Таким образом, в процессе образования входных областей необходимо избавиться от перекрытия между двумя интервалами одновременно.

Другой не менее эффективный по сравнению с вышерассмотренным методом извлечения нечётких правил из данных предложен Кругловым и Борисовым в [4]для решения задачи аппроксимации.

Предположим, что исследуемый объект имеет *n* входов (иначе, векторный вход *x*) и один выход *y*и имеет «истинное» (неизвестное) описание:

$$y = f(x) + e, \qquad (10.31)$$

где *f(x)*– функция неизвестного вида; *e* – случайная аддитивная помеха (отражающая действие неучитываемых системой факторов) с нулевым средним значением и произвольным (неизвестным) распределение на интервале $(-\varepsilon_m, \varepsilon_m)$.

Предположим, что на объекте может быть реализован эксперимент, заключающийся в регистрации *K*пар значений $\langle x^{(k)}, y^{(k)} \rangle$, $k = 1,2,...,K$, при этом величины (*n*-мерные векторы) $x^{(k)}$ измеряются без ошибок; значение *K* при необходимости допускает модификацию.

Алгоритм построения системы может быть описан следующим образом:

Этап 1. Из $m(m < K)$произвольных значений $\langle x^{(k)}, y^{(k)} \rangle$ составляется начальная база знаний модели, отображаемая матрицей $U_{m\times(n+1)}$ со строками вида $\langle x^{(k)}, y^{(k)} \rangle = \langle x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}, y^{(k)} \rangle, k = 1,\ldots,m.$

Такое представление, очевидно, соответствует набору нечётких продукционных правил вида:

$П_k$:ЕСЛИ x_1 есть A_{k1} Иx_2 есть A_{k2} И x_n есть A_{kn}, ТО y=B_k, k=1,…,m.

Этап 2. Для каждой новой экспериментальной точки $\langle x^{(l)}, y^{(l)} \rangle$ рассчитывается прогнозируемое значение по формуле, соответствующей методу дефуззификации среднего центра:

$$y^l = \frac{\sum_{k=1}^{m} y^{(k)} \varphi\left(\left\| x^{(l)} - x^{(k)} \right\|\right)}{\sum_{k=1}^{m} \varphi\left(\left\| x^{(l)} - x^{(k)} \right\|\right)}, \qquad (10.32)$$

где $\varphi(\bullet)$ - функция колокообразной или экспоненциальной формы:

$$\varphi\left(\left\| x^{(l)} - x^{(k)} \right\|\right) = \exp\left(- \lambda \sum_{i=1}^{m} \left| x_i^{(l)} - x_i^{(k)} \right| \right),$$

(10.122)

где λ - параметр функции.

Этап 3.Проверяется неравенство:

$$\left| y^{(l)} - y \right| > d, \qquad (10.33)$$

где d – заданная константа, определяющая погрешность аппроксимации.

При выполнении неравенства база нечётких правил пополняется путём расширения матрицы U (добавлением строки $\left\langle x^{(l)}, y^{(l)} \right\rangle$). В противном случае матрица U остаётся без изменений.

Этап 4.Проверяется правило останова. В данном случае процесс извлечения правил считается завершённым, если в соответствии с этапами 2 и 3 перебраны все K экспериментальных данных (без учёта значений начальной базы).

Если не все экспериментальные точки использованы, то осуществляется переход к этапу 2.

В процессе реализации алгоритма матрица U, а также параметры λ и d считаются априорно заданными.

Очевидно, что база нечётких правил не остаётся фиксированной, а модернизируется по мере поступления экспериментальных данных. Причём непротиворечивость нового продукционного правила относительно набора правил из базы гарантируется предложенной процедурой её пополнения.

Список используемых источников

1. Хайкин С. Нейронные сети: Полный курс: Пер. с англ. - 2-е изд. – М.: Вильямс, 2006. – 1104 с.: ил.
2. Оссовский С. Нейронные сети для обработки информации / Пер. с пол. И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.: ил.
3. Уоссерман Ф. Нейрокомпьютерная техника: теория и практика / Пер. с англ. Ю.А. Зуев. – М.: Мир, 1992.
4. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечёткие системы: Пер. с польск. И.Д.Рудинского, - М.: Горячая линия – Телеком, 2007. – 452 с.: ил.
5. Борисов В.В., Круглов В.В., Федулов А.С. Нечёткие модели и сети. – М.: Горячая линия– Телеком, 2007. -284 с.: ил.