В. С. РОСТОВЦЕВ

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Учебник

Издание второе, стереотипное

САНКТ-ПЕТЕРБУРГ
МОСКВА
КРАСНОДАР
2021

УДК 004.7
ББК 32.973я73

Р 78 **Ростовцев В. С.** Искусственные нейронные сети : учебник для вузов / В. С. Ростовцев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2021. — 216 с. : ил. — Текст : непосредственный.

ISBN 978-5-8114-7462-2

В учебнике приведены основные теоретические и практические сведения по разработке, обучению и применению искусственных нейронных сетей с использованием среды MatLab.

Учебник предназначен для студентов магистратуры направления «Информатика и вычислительная техника» и может быть полезен студентам других специальностей при изучении нейросетевых технологий, а также для слушателей курсов повышения квалификации и профессиональной переподготовки.

УДК 004.7
ББК 32.973я73

Обложка
Е. А. ВЛАСОВА

3. МНОГОСЛОЙНАЯ НЕЙРОННАЯ СЕТЬ

3.1. Принципы построения многослойных нейронных сетей

Среди различных архитектур нейронных сетей (НС) одной из популярных является многослойная архитектура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя. Для первого слоя нейрон первого скрытого слоя связан со всеми входами НС. НС, содержащая входной слой, один или несколько скрытых слоев и один выходной слой, называется многослойным персептроном.

Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идет в направлении, которое минимизирует ошибку на выходе сети. По этому принципу реализован, например, алгоритм обучения однослойного персептрона [1, 3, 5–7, 10].

В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, неизвестны, и обучить такой персептрон невозможно, руководствуясь только величинами ошибок на выходах сети.

Один из вариантов решения этой проблемы — разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, 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 «проб и ошибок», несмотря на свою кажущуюся простоту, требует громоздких рутинных вычислений. Третий, более приемлемый вариант, — распространение сигналов ошибки от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название процедуры обратного распространения.

Третий вариант — обучение многослойного персептрона с помощью генетических алгоритмов [1, 5], которые демонстрируют неплохие результаты обучения только для относительно простых НС. В качестве генотипа используются веса синапсов и коэффициенты крутизны функций активации всех нейронов нейронной сети. Генетические алгоритмы позволяют проводить «распараллеливание» программы для сокращения времени обучения.

3.2. Алгоритм обратного распространения ошибки

В настоящее время выпущено достаточно много различной литературы по описанию нейронных сетей. В [1–3, 5, 10, 18, 22] подробно разобран алгоритм обратного распространения ошибки (ОРО) как метод обучения НС, рас-

смотрены его достоинства, недостатки и способы устранения возникающих в ходе обучения проблем[7].

В [1] приведены общие сведения о НС, рассмотрены различные топологии сетей, предложены методы обучения с учителем и без учителя.

Многими исследователями были предложены улучшения и обобщения алгоритма обратного распространения.

Общепринятый от 0 до 1 динамический диапазон входов и выходов скрытых нейронов не оптимален. Так как величина коррекции веса пропорциональна выходному уровню нейрона, то нулевой уровень ведет к тому, что вес не меняется.

При использовании двоичных входных векторов половина входов в среднем будет равна нулю, и веса, с которыми они связаны, не будут обучаться. Решение состоит в приведении входов к значениям ±1/2 и добавлении смещения к сжимающей функции, чтобы она также принимала значения ±1/2.

С помощью таких простых средств время сходимости сокращается в среднем от 30 до 50%. Это является одним из примеров практической модификации, существенно улучшающей характеристику алгоритма.

Способом обратного распространения (back propogation) называется способ обучения многослойных НС. В таких НС связи между собой имеют только соседние слои, при этом каждый нейрон предыдущего слоя связан со всеми нейронами последующего слоя. Нейроны обычно имеют сигмоидальную функцию возбуждения. Первый слой нейронов называется входным и содержит число нейронов, соответствующее распознаваемому образу. Последний слой нейронов называется выходным и содержит столько нейронов, сколько классов образов распознается. Между входным и выходным слоями располагается один или более скрытых (теневых) слоев. Определение числа скрытых слоев и числа нейронов в каждом слое для конкретной задачи является неформальной задачей.

Принцип обучения такой нейронной сети базируется на вычислении отклонений значений сигналов на выходных процессорных элементах от эталонного и обратного «прогона» этих отклонений до породивших их элементов с целью коррекции ошибки. Еще в 1974 г. Поль Дж. Вербос изобрел значительно более эффективную процедуру для вычисления величины, называемой производной ошибки по весу, когда работал над своей докторской диссертацией в Гарвардском университете. Процедура, известная теперь как алгоритм обратного распространения, стала одним из наиболее важных инструментов в обучении нейронных сетей. Однако этому алгоритму свойственны и недостатки, главный из которых — отсутствие сколько-нибудь приемлемых оценок времени обучения. Понимание того, что нейронная сеть, в конце концов обучится, мало утешает, если на это могут уйти дни и месяцы. Тем не менее, алгоритм обратного распространения имеет широчайшее применение. Например, успех фирмы NEC в распознавании букв был достигнут именно благодаря алгоритму обратного распространения.

[7] Короткий С. Нейронные сети: алгоритм обратного распространения [Электронный ресурс]. Режим доступа: http://www.gotai.net/documents/doc-nn-003.aspx.

Алгоритм обратного распространения ошибки (ОРО) — итеративный градиентный алгоритм обучения, который используется с целью минимизации среднеквадратичного отклонения текущего выхода и желаемого выхода многослойных НС. На каждый нейрон первого слоя подаются все элементы внешнего входного сигнала. Нейроны выполняют взвешенное суммирование элементов входных сигналов, прибавляя к сумме смещение нейрона. Над полученной суммой выполняется нелинейное преобразование активационной функцией. Значение функции активации и есть выход нейрона [1, 3, 7–10].

Среди различных структур нейронных сетей одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС. Такие НС называются полносвязными. Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идет в направлении, которое минимизирует ошибку на выходе сети. По этому принципу строится, например, алгоритм обучения однослойного персептрона. В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, неизвестны, и двух- или более «слойный» персептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС. Один из вариантов решения этой проблемы — разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, 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, несмотря на свою кажущуюся простоту, требует огромного количества вычислений. И, наконец, третий, наиболее приемлемый вариант — распространение сигналов ошибки от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название процедуры обратного распространения.

Цель обучения многослойного персептрона, архитектура которого представлена на рисунке 3.1, состоит в подборе таких весов W_{ik} между входным и скрытым слоем, также весов W_{jM} между скрытым и выходным слоем, чтобы при заданном входном векторе X получить на выходе значения вектора Y, которые с требуемой точностью будут совпадать с ожидаемыми значениями выходного вектора D, взятого из обучающей выборки [1,3].

X — входной вектор; Y — расчетные значения на выходе сети; D — эталонное ожидаемое значение выхода, представленного в обучающей выборке.

Выходной сигнал i-го нейрона скрытого слоя:

$$V_j = f(\sum_{j=0}^{N} w1_{ij} \times x_j), X_0 = 1. \qquad (3.1)$$

Выходной сигнал *i*-го нейрона выходного слоя:

$$Y_i = f(\sum_{i=0}^{K} w2_{ki} \times V_j) = f(\sum_{i=0}^{K} w2_{ki} \times \sum_{j=0}^{N} w1_{ij} \times x_j). \quad (3.2)$$

Значение выхода нейронной сети определяется весами сигналов обоих слоев.

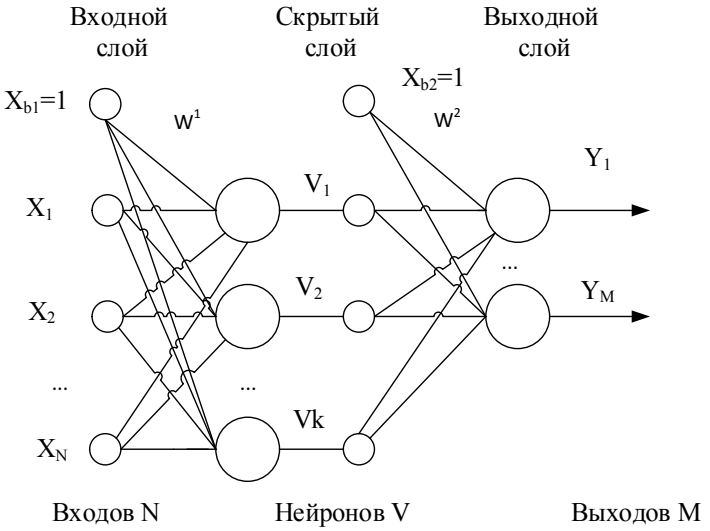

Рис. 3.1
Архитектура многослойного персептрона:

$X_{b1} = 1$, $X_{b2} = 1$ — смещения нейронов скрытого и выходного слоев; W^1 — вектор весовых коэффициентов между входным и скрытым слоем; W^2 — вектор весовых коэффициентов между скрытым и выходным слоем.

Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки НС является величина

$$E(w) = \frac{1}{2} \sum_{j,p} \left(y_{jp}{}^{M} - d_{jp} \right)^2, \quad (3.3)$$

где $y_{jp}{}^{M}$ — реальное выходное состояние нейрона j выходного слоя M нейронной сети при подаче на ее входы p-го примера обучающей выборки; d_{jp} — идеальное (желаемое) выходное состояние этого нейрона; p — число нейронов в выходном слое.

Для функции (3.3) используются градиентные методы обучения. Наиболее простой из них — *метод наискорейшего спуска*.

Модификация вектора весов производится в направление отрицательного градиента целевой функции

$$\mathrm{Grad}(j)E = \frac{dE}{dw_{ij}} = e_i \; x_j \frac{df(S_i)}{dS_i},$$

где $e_i = (y_i - d_i)$ — разница между фактическим и ожидаемым значением выходного сигнала.

Если ввести $\delta_i = e_i \dfrac{df(S_i)}{dS_i}$, то Grad($j$)$E = \delta_i x_j$.

При разработке нейронных сетей преимущественно используется унимодальная (3.4) или биполярная функция активации (3.5).

$$F(S) = \frac{1}{1 + e^{-\beta S}}, \tag{3.4}$$

$$F(S) = -1 + \frac{2}{1 + e^{-\beta S}}. \tag{3.5}$$

Параметр крутизны β влияет на форму функции активации и подбирается экспериментально. При параметре β более 10 функция активации приближается к функции единичного скачка.

Производные функций, приведенных в (3.4) и (3.5), вычисляются сравнительно просто:

$$\frac{dF}{dS} = \beta F(S)\left[1 - F(S)\right], \tag{3.6}$$

$$\frac{dF}{dS} = \beta\left[1 - F^2(S)\right]. \tag{3.7}$$

Суммирование ведется по всем нейронам выходного слоя и по всем обрабатываемым сетью образам. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(n)} = -\eta \times \frac{dE}{dw_{ij}}. \tag{3.8}$$

Здесь w_{ij} — весовой коэффициент синаптической связи, соединяющей i-й нейрон слоя n–1 с j-м нейроном слоя n, η — коэффициент скорости обучения, $0 < \eta < 1$.

$$\frac{dE}{dw_{ij}} = \frac{dE}{dy_j} \times \frac{dy_j}{ds_j} \times \frac{ds_j}{dw_{ij}}. \tag{3.9}$$

Здесь под y_j, как и раньше, подразумевается выход нейрона j, а под s_j — взвешенная сумма его входных сигналов, т. е. аргумент активационной функции. Так как множитель является производной этой функции по ее аргументу, производная активационной функции должна быть определена на всей оси абсцисс. В связи с этим функция единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемых НС. В них применяются такие гладкие функции, как гиперболический тангенс или классический сигмоид с экспонентой. В случае гиперболического тангенса

$$\frac{dy}{ds} = 1 - s^2. \tag{3.10}$$

Третий множитель, очевидно, равен выходу нейрона предыдущего слоя $y_i^{(n-1)}$.

Что касается первого множителя в (3.7), он легко раскладывается следующим образом [1–4]:

$$\frac{dE}{dy_j} = \sum_k \frac{dE}{dy_k}\frac{dy_k}{ds_k}\frac{ds_k}{dy_j} = \sum_k \frac{dE}{dy_k}\frac{dy_k}{ds_k} w_{jk}^{(n+1)}. \tag{3.11}$$

Здесь суммирование по k выполняется среди нейронов слоя $n + 1$.

Введя новую переменную

$$\delta_j^{(n)} = \frac{dE}{dy_j}\frac{dy_j}{ds_j}, \tag{3.12}$$

получается рекурсивная формула для расчетов величин $\delta_j^{(n)}$ слоя n из величин $\delta_k^{(n+1)}$ более старшего слоя $n + 1$.

$$\delta_j^{(n)} = \left[\sum_k \delta_j^{(n+1)} w_{jk}^{(n+1)}\right]\frac{dy_j}{ds_j}. \tag{3.13}$$

Для выходного же слоя

$$\delta_l^{(N)} = \left(y_l^{(N)} - d_l\right)\frac{dy_l}{ds_l}. \tag{3.14}$$

Теперь мы можем записать в раскрытом виде:

$$\Delta w_{ij}^{(n)} = -\eta \delta_j^{(n)} y_i^{(n-1)}. \tag{3.15}$$

Иногда для придания процессу коррекции весов некоторой инерционности, сглаживающей резкие скачки при перемещении по поверхности целевой функции, дополняется значением изменения веса на предыдущей итерации:

$$\Delta w_{ij}^{(n)}(t) = -\eta\left(\mu \Delta w_{ij}^{(n)}(t-1) + (1-\mu)\delta_j^{(n)} y_i^{(n-1)}\right), \tag{3.16}$$

где μ — коэффициент инерционности (момента); t — номер текущей итерации.

Таким образом, полный алгоритм обучения НС с помощью процедуры обратного распространения строится так:

1. Подать на входы сети один из возможных образов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних. Напомним, что

$$s_j^{(n)} = \sum_{i=0}^{M} y_i^{(n-1)} w_{ij}^{(n)}, \tag{3.17}$$

где M — число нейронов в слое $n - 1$ с учетом нейрона с постоянным выходным состоянием +1, задающего смещение; $y_i^{(n-1)} = x_{ij}^{(n)}$ — i-й вход нейрона j слоя n.

$$y_j^{(n)} = f\left(s_j^{(n)}\right), \tag{3.18}$$

где $f()$ — сигмоид,

$$y_q^{(0)} = I_q, \tag{3.19}$$

где I_q — q-я компонента вектора входного образа.

2. Рассчитать $\delta^{(N)}$ для выходного слоя по формуле (3.14).

Рассчитать по формуле (3.15) или (3.16) изменения весов $\Delta w^{(N)}$ слоя N.

3. Рассчитать по формулам (3.13) и (3.15) (или (3.13) и (3.16)) соответственно $\delta^{(n)}$ и $\Delta w^{(n)}$ для всех остальных слоев, $n = N - 1, N - 2, ..., 1$.

4. Скорректировать все веса в НС

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t). \qquad (3.20)$$

5. Если ошибка сети (3.1) превышает заданное пользователем значение, перейти на шаг 1. В противном случае — конец.

Сети на шаге 1 попеременно в случайном порядке предъявляются все тренировочные образы, чтобы сеть, образно говоря, не забывала одни по мере запоминания других.

Полностью алгоритм обучения с помощью метода обратного распространения ошибки выглядит следующим образом.

На первом шаге на все входы сети подается один из возможных образов из обучающей выборки. В режиме нормального функционирования сети по формулам (3.1) и (3.2) вычисляются выходные значения.

На втором шаге рассчитывается по формуле (3.12) ошибка нейронов и по формуле (3.14) ошибка синаптических весов выходного слоя.

На третьем шаге вычисляются ошибки нейронов и синаптических весов для скрытых слоев в порядке, обратном нормальному выполнению суммирования.

На четвертом этапе происходит корректировка по формуле (3.20) синаптических весов всех слоев.

На пятом шаге полученная ошибка обучения сравнивается с максимально допустимой ошибкой и принимается решение об окончании или продолжении процесса обучения.

3.3. Нормализация входной и выходной информации

Входная и выходная информация для нейронной сети формируется из входных векторов (входные значения обучающей выборки) и выходных векторов обучающей выборки.

Нормализация значений входных и выходных данных требуется, чтобы исключить доминирование одних значений обучающей выборки перед другими. Обычно нельзя подавать для нейронной сети входные сигналы в их истинном диапазоне величин и получать от сети выходные сигналы в требуемом диапазоне.

Нормирование исключает доминирование одних входных сигналов перед другими и обеспечивает качественное обучение [1, 3, 10].

Поэтому перед подачей входных сигналов их необходимо нормировать в одном из диапазонов: от минус 1 до 1 (формула (3.21)); от минус 0,5 до 0,5 (формула (3.22)); от 0 до 1 (формула (3.23)).

Наиболее простое нормирование сигналов можно выполнить следующим образом. Каждая компонента входного вектора данных x_i заменяется величиной, вычисляемой по формулам

$$x_{i0} = \frac{x_i - 0,5\left(\max\{x_i\} + \min\{x_i\}\right)}{0,5\left(\max\{x_i\} - \min\{x_i\}\right)}, \tag{3.21}$$

$$x_{i0} = \frac{x_i - 0,5\left(\max\{x_i\} + \min\{x_i\}\right)}{\left(\max\{x_i\} - \min\{x_i\}\right)}, \tag{3.22}$$

где x_{i0} — нормализованное значение величины; x_i — нормализуемая величина; $\max x_i$ и $\min x_i$ — соответственно максимальное и минимальное значения для данной компоненты, вычисленные по всей обучающей выборке.

Можно нормировать и по-другому, например, пересчитывая выборку так, чтобы разброс данных был единичным, по формуле

$$x_{i0} = \frac{x_i - \min\{x_i\}}{\max\{x_i\} - \min\{x_i\}}. \tag{3.23}$$

Здесь имеется одна сложность. Любое изменение обучающей выборки должно соответственно менять и правило нормирования данных.

Выбор способа нормализации и значений зависит от конкретной обучающей выборки и выбирается экспериментально.

Для денормализации данных используются обратные преобразования, выполняемые по формулам (3.24)–(3.26):

Диапазон от 0 до 1.

$$x_i = \min\{x_i\} + x_{i0}(\max\{x_i\} - \min\{x_i\}), \tag{3.24}$$

Диапазон от минус 1 до 1.

$$x_i = 0,5(\max\{x_i\} + \min\{x_i\}) + 0,5x_{i0}(\max\{x_i\} - \min\{x_i\}), \tag{3.25}$$

Диапазон от минус 0,5 до 0,5.

$$x_i = 0,5(\max\{x_i\} + \min\{x_i\}) + x_{i0}(\max\{x_i\} - \min\{x_i\}). \tag{3.26}$$

3.4. Пример расчета параметров сети в алгоритме обучения

Обратное распространение — это самый популярный алгоритм для обучения НС с помощью изменения весов связей. Ошибка распространяется от выходного слоя к входному, т. е. в направлении, противоположном направлению прохождения сигнала при нормальном функционировании сети [1, 3–7, 10].

Выполнение алгоритма начинается с генерации произвольных весов для многослойной сети. Затем процесс, описанный ниже, повторяется до тех пор, пока средняя ошибка на входе не будет признана достаточно малой:

1. Берется пример входного сигнала с соответствующим правильным значением выхода, т. е. входной и выходной векторы.

2. Рассчитывается прямое распространение сигнала через сеть (определяются весовые суммы S_i и активаторы u_i для каждого нейрона).

3. Начиная с выходов, выполняется обратное движение через ячейки выходного и промежуточного слоя, при этом программа рассчитывает значения ошибок по формуле (3.26) и (3.27):

– для нейрона выходного слоя

$$\delta_0 = (C_i - u_5)u_5(1 - u_5); \tag{3.27}$$

– для всех нейронов скрытого слоя

$$\delta_i = (\sum mm > iw_{mi} \times \delta_0)u_i(1 - u_i). \tag{3.28}$$

Здесь m обозначает все нейроны, связанные со скрытым узлом; w — заданный вектор веса; u — выход активационной функции.

4. Веса в сети w_{ij}^* обновляются следующим образом по формулам (3.29) и (3.30):

– для весов синапсов между скрытым и выходным слоем

$$w_{ij}^* = w_{ij} - \eta\delta_0 u_i; \tag{3.29}$$

– для весов синапсов между скрытым и входным слоем

$$w_{ij}^* = w_{ij} - \eta\delta_i u_i. \tag{3.30}$$

Параметр η характеризует коэффициент скорости обучения (или размер шага). Это небольшое значение ограничивает изменение, которое может произойти на каждом шаге. Параметр η можно определить экспериментальным путем. Лучше начать обучение с небольшого значения ($\eta = 0{,}1$) и затем постепенно его повышать.

Продвижение вперед по сети соответствует активации нейронов, а продвижение назад — коррекции весов синапсов и весов смещений нейронов. Затем веса обновляются таким образом, чтобы минимизировать ошибку для данного входного вектора. Если коэффициент обучения слишком велик, сеть может никогда не сойтись, т. е. не будут найдены нужные веса синапсов при минимальной среднеквадратичной ошибке обучения. При малом значении коэффициента обучения увеличивается время обучения сети.

Пример расчета весовых коэффициентов нейронной сети.

Проход вперед. Сначала выполняется расчет движения входного сигнала по сети (рис. 3.2). Значение смещения для каждого нейрона равно 1.

$$U_3 = f(w_{3,1}u_1 + w_{3,2}u_2 + w_{3,b}1), \tag{3.31}$$

$$U_3 = f(0*1 + 0{,}5*1 + 1*1) = f(1{,}5),$$

$$f(S) = \frac{1}{1 + e^{-\beta S}}. \tag{3.32}$$

$U_3 = 0{,}81757$ (при коэффициенте крутизны функции активации $\beta = 1$).

Сначала выполняется расчет движения входного сигнала по сети (рис. 3.2).

$$U_4 = f(w_{4,1}u_1 + w_{4,2}u_2 + w_{4,b}1), \tag{3.33}$$

$$U_4 = f(-1*0 + 2*1 + 1*1 = f(3),$$

$$f(S) = \frac{1}{1 + e^{-\beta S}}.$$

$U_4 = 0{,}952574$ (при коэффициенте крутизны функции активации $\beta = 1$).

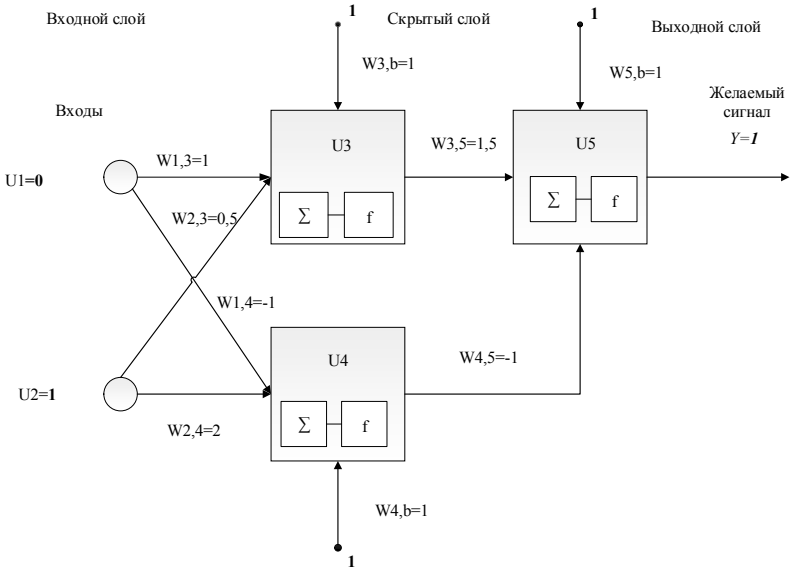

Рис. 3.2
Иллюстрация фрагмента нейронной сети

Теперь сигнал дошел до скрытого слоя. Конечный шаг — переместить сигнал в выходной слой и рассчитать значение на выходе сети:

$$U_5 = f(w_{5,3}u_3 + w_{5,4}u_4 + w_{5,b}1), \qquad (3.34)$$

$$U_5 = f(0{,}81757*1{,}5 + 0{,}952574*(-1) + 1*1 = f(1{,}2195),$$

$$f(S) = \frac{1}{1+e^{-\beta S}}.$$

$U_5 = 0{,}78139$ (при коэффициенте крутизны функции активации $\beta = 1$).

Правильный реакцией НС на входной сигнал является значение $U_5 = 1{,}0$; значение рассчитанное сетью, составляет 0,78139.

Для коррекции весовых коэффициентов обычно используется среднеквадратичная ошибка, вычисляемая по формуле среднеквадратичной ошибки:

$$E = 0{,}5*\Sigma(y^{\text{эталонное}} - y^{\text{расчетное}})^2.$$

Значение $y^{\text{эталонное}}$ берется из обучающей выборки, а $y^{\text{расчетное}}$ вычисляется после подачи на входы нейронной сети входного вектора:

$$E = 0{,}5*(1{,}0 - 0{,}78139)^2 = 0{,}023895.$$

Обратный проход процедуры обратного распространения ошибки.

По формуле (3.27) рассчитается ошибка в выходном узле:

$$\delta_0 = (C - u_5)*u_5*(1 - u_5) = (1{,}0 - 0{,}78139)*0{,}78139*(1 - 0{,}78139) = 0{,}0373.$$

Теперь следует рассчитать ошибку δ_{u3} и δ_{u4} для двух скрытых нейронов по формуле (3.28):

$$\delta_{u3} = (\delta_0*w_{5,3})*u_3*(1 - u_3),$$

$$\delta_{u3} = (0{,}0373*1{,}5*0{,}81757*(1 - 0{,}81757) = 0{,}0083449,$$

$$\delta_{u4} = (\delta_0 * w_{5,4}) * u_4 * (1 - u_4),$$

$$\delta_{u4} = (0{,}0373 * (-1) * 0{,}952574 * (1 - 0{,}952574) = -0{,}0016851.$$

Изменение весовых коэффициентов сети.

1. Сначала обновляются веса w_{ij}^* между выходным и скрытым слоем:

$$w_{ij}^*(t+1) = w_{ij}(t) + \eta\delta_0 \times u_i, \tag{3.35}$$

$$w_{5,3}(t+1) = w_{5,3}(t) + (\eta * 0{,}0373 * u_3),$$

$$w_{5,3} = 1{,}5 + (0{,}5 * 0{,}0373 * 0{,}81757) = 1{,}51525,$$

$$w_{5,4}(t+1) = w_{5,4}(t) + (\eta * 0{,}0373 * u_4),$$

$$w_{5,4}(t+1) = -1{,}0 + (0{,}5 * 0{,}0373 * 0{,}952574) = -0{,}9882.$$

2. Теперь нужно обновить веса смещений для выходного нейрона:

$$w_{5,b}(t+1) = w_{5,b}(t) + (\eta * 0{,}0373 * 1),$$

$$w_{5,b}(t+1) = 1 + (0{,}5 * 0{,}0373 * 1) = 1{,}01865.$$

Для $w_{5,3}$ вес увеличен с 1,5 до 1,51525, а для $w_{5,4}$ вес изменен с –1 до –0,9882. Смещение обновлено для уменьшения среднеквадратичной ошибки $E = 0{,}78139$.

3. Теперь обновляются веса между входным и скрытым слоем:

$$w_{3,1}(t+1) = w_{3,1}(t) + (0{,}5 * 0{,}0083449 * u_1),$$

$$w_{3,1}(t+1) = 1{,}0 + (0{,}5 * 0{,}0083449 * 0) = 1{,}0,$$

$$w_{3,2}(t+1) = w_{3,2}(t) + (\eta * 0{,}0083449 * u_2),$$

$$w_{3,2}(t+1) = 0{,}5 + (0{,}5 * 0{,}0083449 * 1) = 0{,}50417,$$

$$w_{4,1}(t+1) = w_{4,1}(t) + (0{,}5 * -0{,}0016851 * u_1),$$

$$w_{4,1}(t+1) = -1{,}0 + (0{,}5 * -0{,}0016851 * 0) = -1{,}0,$$

$$w_{4,2}(t+1) = w_{4,2}(t) + (\eta * -0{,}0016851 * u_2),$$

$$w_{4,2}(t+1) = 2 + (0{,}5 * -0{,}0016851 * 1) = 1{,}999916.$$

4. Теперь обновляются веса смещений для скрытых нейронов:

$$W_{3,b}(t+1) = w_{3,b}(t) + (\eta * 0{,}0083449 * 1),$$

$$W_{3,b}(t+1) = 1 + (0{,}5 * 0{,}0083449 * 1) = 1{,}00417,$$

$$W_{4,b}(t+1) = w_{4,b}(t) + (\eta * -0{,}0016851 * 1),$$

$$W_{4,b}(t+1) = 1 + (0{,}5 * -0{,}0016851 * 1) = 0{,}99915.$$

5. Обновление весов для скрытого слоя завершается. Выполним еще один проход вперед:

$$U_3 = f(w_{3,1} * u_1 + w_{3,2} * u_2 + w_{3,b} * 1),$$

$$U_3 = f(0 * 1 + 0{,}50417 * 1 + 1{,}00417 * 1 = f(1{,}50834),$$

$$f(S) = \frac{1}{1 + e^{-\beta S}},$$

$U_3 = 0{,}8188$ (при коэффициенте крутизны функции активации $\beta = 1$).

$$U_4 = f(w_{4,1} * u_1 + w_{4,2} * u_2 + w_{4,b} * 1),$$

$$U_4 = f(-1 * 0 + 1{,}99915 * 1 + 1{,}99915 * 1 = f(2{,}99831),$$

$$U_4 = 0{,}952497,$$

$$U_5 = f(w_{5,3}*u_3 + w_{5,4}*u_4 + w_{5,b}*1),$$

$$U_5 = f(1{,}51525*0{,}81888 + (-0{,}9822)*0{,}952497 + 1{,}01865*1 = f(1{,}32379),$$

$$U_5 = 0{,}7898 \text{ (при коэффициенте крутизны функции активации } \beta = 1),$$

$$E = 0{,}5*(1{,}0 - 0{,}7898)^2 = 0{,}022.$$

На первой итерации ошибка была равна 0,023895, а на второй значение E уменьшилось до 0,022. Следовательно, алгоритм обратного распространения ошибки обеспечивает уменьшение среднеквадратичной ошибки в процессе обучения НС.

3.5. Параметры, влияющие на обучение многослойной нейронной сети

При разработке нейронной сети требуется подобрать параметры и ее обучение и выбрать архитектуру (табл. 3.1).

Анализ параметров, влияющих на качество обучения и выбор архитектуры нейронной сети имеет несколько *«узких мест»* [1, 3, 5–7, 10, 19].

Таблица 3.1

Выбор параметров, влияющих на обучение и архитектуру нейронной сети

Наименование параметра	Краткое описание «узких мест»	Рекомендации
Характеристика обучающей выборки	Представительная обучающая выборка обеспечивает качество обучения и обобщение	Обеспечить полноту, равномерность и непротиворечивость обучающей выборки, объем которой должен в несколько раз превосходить общее количество синапсов сети
Нормализация обучающей выборки	Исключение доминирования ряда параметров обучающей выборки	Подбор производится экспериментально и зависит от опыта специалиста
Последовательность выбора примеров из обучающей выборки	Случайный выбор примеров обеспечивает отсутствие «привыкания» НС к ряду примеров	Рекомендуется случайный выбор примеров из обучающей выборки
Тип активационной функции	Выбор типа функции активации влияет на погрешность обучения нейронной сети	Подбор производится экспериментально и зависит от опыта специалиста. Для скрытых слоев рекомендуется сигмоидальная функция, а для выходного слоя — линейная
Параметр крутизны активационной функции	Выбор типа коэффициента крутизны функции активации влияет на погрешность обучения нейронной сети и возможный «паралич» сети	Подбор производится экспериментально и зависит от опыта специалиста, создающего НС