ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Липецкий государственный технический университет»

П.В. Сараев

Нейросетевые методы искусственного интеллекта

Учебное пособие

Липецк 2007

УДК 004.8
С 20

Сараев, П.В. Нейросетевые методы искусственного интеллекта: учебное пособие/ П.В. Сараев.– Липецк: ЛГТУ, 2007.– 64 с.

ISBN 978-5-88247-319-7

В учебном пособии представлены структуры и алгоритмы применения нейронных сетей в задачах искусственного интеллекта. Основное внимание уделено нейронным сетям прямого распространения. Рассмотрены самоорганизующиеся карты Кохонена и их использование в задачах кластеризации объектов. Приведено описание нейронных сетей с обратными связями на примере сетей Хопфилда и их применение в задачах распознавания образов. Учебное пособие предназначено для студентов математических и технических специальностей при изучении дисциплин «Интеллектуальные системы» и «Системы искусственного интеллекта».

Табл. 2. Ил. 26. Библиогр.: 20 назв.

Рецензенты: канд. техн. наук Бурцев В.Д.;
кафедра электроники, телекоммуникаций и
компьютерных технологий Липецкого
государственного педагогического университета
(зав. каф. — д-р техн. наук, проф. Малыш В.Н.)

ISBN 978-5-88247-319-7

Содержание

Введение

Барр (Barr) и Фейгенбаум (Feigenbaum) дали следующее определение искусственного интеллекта (ИИ): «Искусственный интеллект – это область информатики, которая занимается разработкой интеллектуальных компьютерных систем, т.е. систем, обладающих возможностями, которые мы традиционно связываем с человеческим разумом,– понимание языка, обучение, способность рассуждать, решать проблемы и т.д.». Как видно, основные усилия в области ИИ направлены на моделирование мышления человека.

Существует два подхода к построению интеллектуальных систем: нейробионический и информационный. Суть первого подхода состоит в том, что деятельность мозга моделируется на основе представления о его строении и протекающих в нем процессах с нейрофизиологической точки зрения. Для второго подхода неважно, как именно устроен мозг, важен способ мышления, обработки данных и знаний. Нейронные сети (НС) появились именно как модели строения головного мозга, что соответствует нейробионическому подходу (поэтому перед упоминанием НС часто указывается прилагательное «искусственные», чтобы отличить их от «реальных», биологических нейронных сетей человеческого головного мозга). НС могут быть рассмотрены как упрощенные математические модели мозга, функционирующие в виде параллельных распределенных вычислительных систем. Тем не менее, в настоящее время НС рассматриваются преимущественно с точки зрения информационного подхода, что связано с развитием математических основ построения и использования НС.

Важным элементом ИИ является обучение, т.е. способность приобретать знания на основе опыта взаимодействия с окружающей средой. С другой стороны, именно возможность обучаться является важнейшим аспектом работы НС. Таким образом, обучаемость также позволяет рассматривать НС как часть ИИ.

Началом теории НС считается 1943 г. и работа МакКаллока (McCulloc) и Питтса (Pitts) «Логическое исчисление идей, относящихся к нервной дея-

тельности» («A Logical Calculus of Ideas Immanent in Nervous Activity») и чуть более поздняя работа Хебба (Hebb) «Организация поведения» («Organization of Behavior») (1949 г.). Интерес к НС был потерян после выхода в 1969 г. работы Минского (Minsky) и Пейперта (Papert) «Перцептроны» («Perceptrons»), в которой была показана ограниченность возможностей использования НС. Новый интерес к НС разгорелся в 80-е годы XX века, чему способствовали следующие события:

— 1982 г. – выход работы Хопфилда (Hopfield) по математическим основам динамики НС;

— 1984 г. – Кохоненом (Kohonen) были разработаны сети, обучающиеся без учителя;

— 1986 г. – Румельхартом (Rumelhart) и МакКлеландом (McCleland) был представлен алгоритм обратного распространения ошибки для обучения многослойных НС.

В настоящее время область применения НС очень широка. Они используются в различных задачах идентификации систем, системах управления, прогнозирования, распознавания образов, классификации и кластеризации объектов.

1. Основы теории нейронных сетей

1.1. Основные понятия

Основным элементом НС является *нейрон*, преобразующий векторный вход $x \in \mathbb{R}^n$ в скалярный выход $y \in \mathbb{R}$. Каждому входному сигналу x_i, $i = 1, \ldots, n$, ставится в соответствие число $w_i \in \mathbb{R}$, называемое *весовым коэффициентом* входа, *весом* или *синапсом* (рис. 1.1). Преобразование происходит в два этапа:

1. Вычисляется *уровень активации* нейрона net по формуле

$$net = \langle w, x \rangle = \sum_{i=1}^{n} w_i x_i, \tag{1.1}$$

   где $w = \begin{bmatrix} w_1 & w_2 \ldots & w_n \end{bmatrix}^T \in \mathbb{R}^n$ — вектор весов, $x = \begin{bmatrix} x_1 & x_2 \ldots & x_n \end{bmatrix}^T \in \mathbb{R}^n$ — входной вектор, $\langle \cdot, \cdot \rangle$ — скалярное произведение векторов.

2. К значению уровня активации применяется, как правило, нелинейная *функция активации* $\sigma : \mathbb{R} \to \mathbb{R}$

$$y = \sigma(net). \tag{1.2}$$

Таким образом, нейрон осуществляет отображение (обычно нелинейное) из пространства $\mathbb{R}^n$ в пространство $\mathbb{R}$.

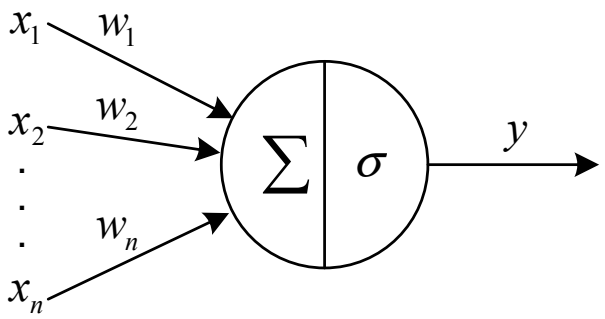

Рис. 1.1. Нейрон

Для удобства работы в вектор входных сигналов вводят дополнительный – фиктивный постоянный единичный входной сигнал $x_0 = 1$, которому

соответствует вес w_0 (рис. 1.2). В результате этого формула вычисления уровня активации нейрона представляется в виде

$$net = \sum_{i=0}^{n} w_i x_i. \tag{1.3}$$

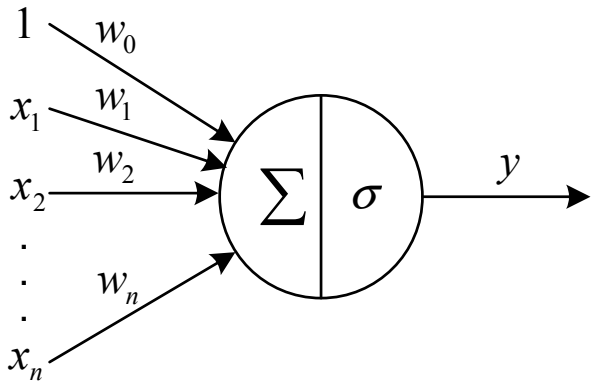

Рис. 1.2. Нейрон с фиктивным единичным входным сигналом

Формула (1.3) является частным случаем формулы вычисления уровня активации в полиномиальном нейроне k-го порядка, который осуществляет следующее преобразование:

$$net = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i_1=1}^{n}\sum_{i_2=1}^{n} w_{i_1 i_2} x_{i_1} x_{i_2} + \ldots + \\ + \sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\ldots\sum_{i_k=1}^{n} w_{i_1 i_2 \ldots i_k} x_{i_1} x_{i_2} \ldots x_{i_k}. \tag{1.4}$$

Возможно вычисление уровня активации еще одним способом:

$$net = \frac{\sum\limits_{i=0}^{n} w_i x_i}{\sum\limits_{i=0}^{n} \widetilde{w}_i x_i}. \tag{1.5}$$

Нейрон, использующий функцию (1.5), называется Паде-нейроном, т.к. аппроксимация дробно-рациональными функциями носит имя Паде. На практике нейроны, в которых уровень активации рассчитывается по формулам (1.4) и (1.5), не используются, т.к. они усложняют процесс построения нейросетевой модели, не внося значительных дополнительных возможностей.

В качестве функции активации (1.2) обычно используется одна из следующих:

— пороговая-1 (рис. 1.3)

$$\sigma(net) = \begin{cases} 1, & net \geqslant T, \\ 0, & net < T, \end{cases}$$

где T — значение порога;

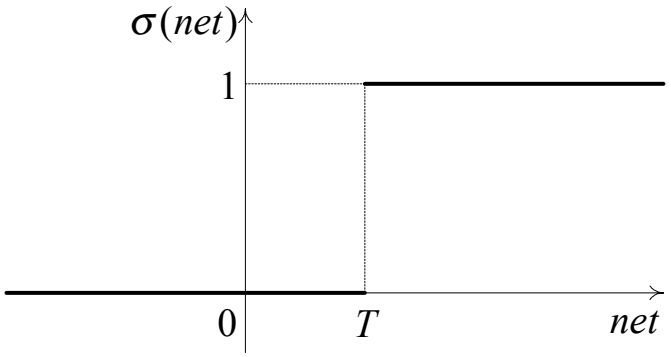

Рис. 1.3. Пороговая функция-1

— пороговая-2 (называемая также знаковой) (рис. 1.4)

$$\sigma(net) = \begin{cases} 1, & net \geqslant T, \\ -1, & net < T, \end{cases}$$

где T — значение порога;

— линейная пороговая (рис. 1.5)

$$\sigma(net) = \begin{cases} 1, & net \geqslant T_2, \\ \dfrac{net - T_1}{T_2 - T_1}, & T_1 < net < T_2, \\ 0, & net \leqslant T_1, \end{cases}$$

где T_1 и T_2 — пороговые константы;

— гиперболический тангенс (рис. 1.6)

$$\sigma(net) = \mathrm{th}(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}};$$

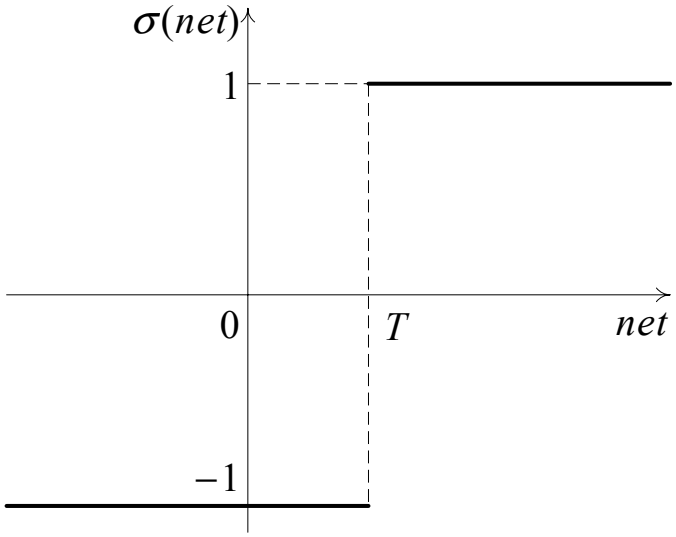

Рис. 1.4. Пороговая функция-2

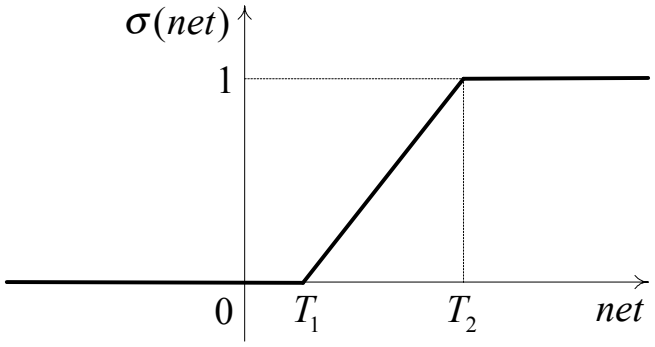

Рис. 1.5. Линейная пороговая функция

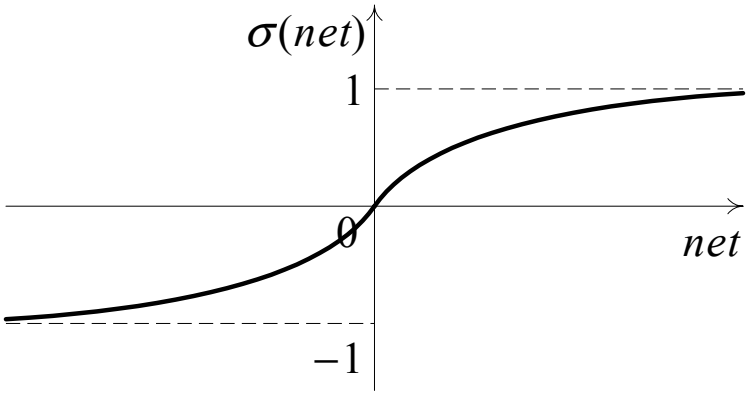

Рис. 1.6. Гиперболический тангенс

— сигмоидная логистическая-1 (рис. 1.7)

$$\sigma(net) = \frac{1}{1 + e^{-net}}.$$

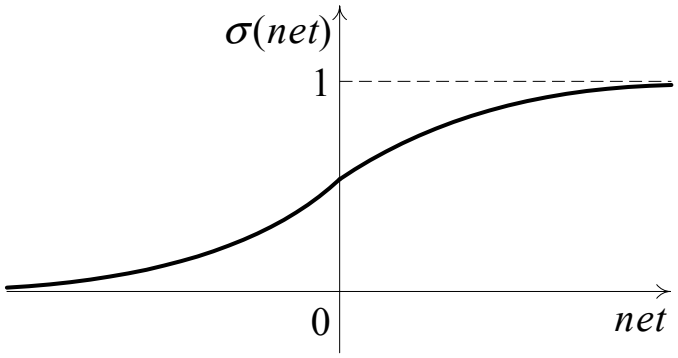

Рис. 1.7. Сигмоидная логистическая функция-1

— сигмоидная логистическая-2 (рис. 1.8)

$$\sigma(net) = \frac{net}{|net| + a}, a > 0.$$

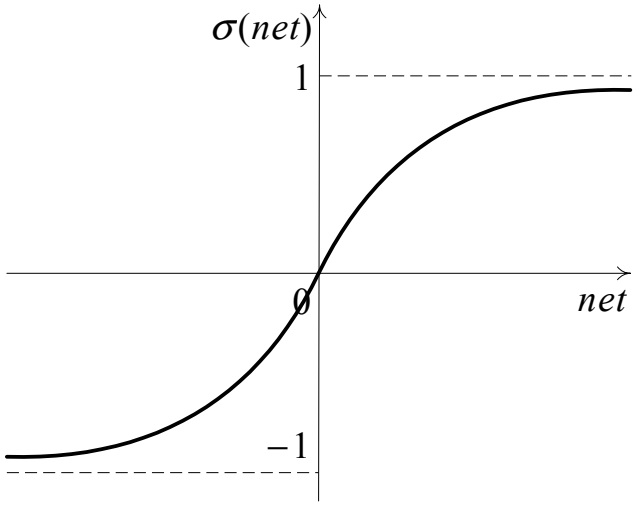

Рис. 1.8. Сигмоидная логистическая функция-2 (a=5)

Отметим следующие особенности приведенных функций активации:

— Области значений функций активации ограничены. Как правило, это диапазоны $(0; 1)$, $[0; 1]$, $(-1; 1)$, $[-1; 1]$.

— Функции активации монотонно возрастают, т.е. $\forall\, net_1, net_2 \in \mathbb{R}$ из условия $net_1 < net_2$ следует, что $\sigma(net_1) \leqslant \sigma(net_2)$.

— Существуют пределы функции при $net \to -\infty$ и при $net \to \infty$.

Эти особенности означают, что функции активации обладают свойством насыщения: при больших по модулю уровнях активации нейронов функции дают практически одни и те же значения. Указанное свойство насыщения в ряде случаев является недостатком. Например, сигмоидная логистическая функция-1, примененная к аргументу $net = 5$, выдает значение $\sigma(5) \approx 0.993$, что уже довольно близко к 1. В связи с этим часто используется параметризованная функция вида

$$\sigma_\alpha(net) = \frac{1}{1 + e^{-\alpha \cdot net}}, \alpha > 0,$$

где α — параметр крутизны (рис. 1.9). При $\alpha \to \infty$ параметризованная сигмоидная логистическая функция стремится к пороговой функции-1, а при $\alpha \to 0$ — к линейной. Обычно значение α задается из диапазона от 0.1 до 0.8.

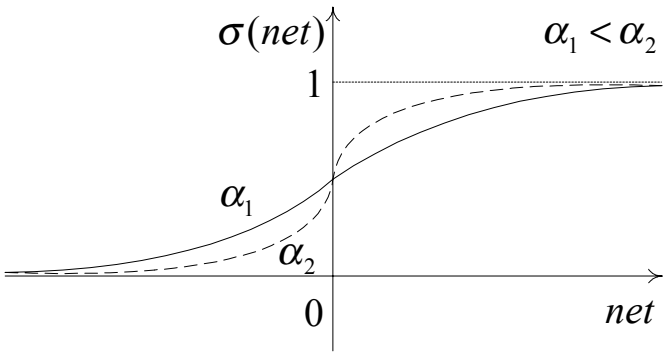

Рис. 1.9. Параметризованная сигмоидная логистическая функция

В ряде задач требуется на выходе получать произвольные действительные значения, а не только ограниченные областью значений функций активации. В таких ситуациях говорят, что в нейронах активационная функция отсутствует или применяется единичная функция

$$\sigma(net) = net.$$

Функции активации (1.3)—(1.8) разделяются на два класса: дифференцируемые на всей числовой оси $\mathbb{R}$ (например, сигмоидные логистические) и недифференцируемые в некоторых точках (например, пороговые). Наиболее часто при решении задач применяются дифференцируемые функции, хотя

существуют задачи (например, распознавание образов), когда применяются пороговые функции (в нейронных сетях Хопфилда). Производные дифференцируемых функций активации по аргументу легко выражаются в том числе через значения самих функций:

— производная гиперболического тангенса:

$$\sigma(net)'_{net} = 1 - (\sigma(net))^2;$$

— производная сигмоидной логистической функции-1:

$$\sigma(net)'_{net} = \sigma(net)(1 - \sigma(net));$$

— производная параметризованной сигмоидной логистической функции:

$$\sigma_\alpha(net)'_{net} = \alpha\sigma(net)(1 - \sigma(net));$$

— производная сигмоидной логистической функции-2:

$$\sigma(net)'_{net} = \frac{a}{(|net| + a)^2}.$$

Совокупность взаимосвязанных нейронов образует *нейронную сеть*. Существуют различные архитектуры (структуры) НС, которые могут быть разделены на два больших класса: *НС прямого распространения* (НС ПР) и *НС с обратными связями* (НС ОС, рекуррентные НС). Эти классы будут рассмотрены далее.

Чтобы НС могла быть использована для решения задач, следует задать или определить значения весов по некоторым известным данным. Другими словами, НС должна научиться решать задачи.

1.2. Постановка задачи обучения

Обучением НС называется процесс определения значений весов НС фиксированной структуры на основе набора известных вход-выходных данных. Эти данные задаются таблицей, называемой *обучающим множеством*,

в которой отражается зависимость значений выходов от комбинации значений входных величин. Строки обучающего множества называются *примерами*, а значения выходов — *указаниями учителя*. Состоящее из k примеров обучающее множество, включающее n входов и r выходов, имеет вид:

$$\left[\begin{array}{cccc|cccc} \widetilde{x}_{11} & \widetilde{x}_{12} & \dots & \widetilde{x}_{1n} & \widetilde{y}_{11} & \widetilde{y}_{12} & \dots & \widetilde{y}_{1r} \\ \widetilde{x}_{21} & \widetilde{x}_{22} & \dots & \widetilde{x}_{2n} & \widetilde{y}_{21} & \widetilde{y}_{22} & \dots & \widetilde{y}_{2r} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \widetilde{x}_{k1} & \widetilde{x}_{k2} & \dots & \widetilde{x}_{kn} & \widetilde{y}_{k1} & \widetilde{y}_{k2} & \dots & \widetilde{y}_{kr} \end{array}\right].$$

В кратком виде обучающее множество представляется так:

$$\{\widetilde{x}_i, \widetilde{y}_i\}, i = 1, \dots, k,$$

где $\widetilde{x}_i \in \mathbb{R}^n$ и $\widetilde{y}_i \in \mathbb{R}^r$ — соответственно входной вектор и вектор указаний учителя из i-го примера. Если задан вектор весов НС $w \in \mathbb{R}^s$, то при подаче на НС входного вектора НС выдаст набор r выходных значений. Степень близости вектора выходов НС на i-м примере $y_i(w)$ и указаний учителя $\widetilde{y}_i$ характеризуется мгновенным функционалом качества

$$Q(\varepsilon_i(w)) = \varepsilon_i^T(w) V \varepsilon_i(w), \tag{1.6}$$

где $\varepsilon_i(w) = y_i(w) - \widetilde{y}_i$ — вектор отклонений выходов сети от указаний учителя, $V \in \mathbb{R}^{r \times r}$ – положительно определенная матрица, задающая взвешенную норму вектора $\varepsilon_i(w)$. Обычно в роли матрицы V выступает единичная, поэтому функционал представляет собой евклидову норму вектора отклонений

$$Q_i(\varepsilon(w)) = \varepsilon_i^T(w)\varepsilon_i(w) = (y_i(w) - \widetilde{y}_i)^T(y_i(w) - \widetilde{y}_i) = \sum_{j=1}^{r}(y_{ij}(w) - \widetilde{y}_{ij})^2.$$

Интегральная степень соответствия (по всем примерам) нейросетевой модели данным из обучающего множества задается функционалом

$$J(w) = \frac{1}{2}\sum_{i=1}^{k} Q_i(w) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{r}(y_{ij}(w) - \widetilde{y}_{ij})^2. \tag{1.7}$$

Множитель $\frac{1}{2}$ введен для удобства преобразований, на вектор оптимальных весов он не влияет.

Для случая с одним выходом ($r = 1$) функционал (1.7) упрощается и принимает следующий вид:

$$J(w) = \frac{1}{2}\sum_{i=1}^{k} Q_i(w) = \frac{1}{2}\sum_{i=1}^{k}(y_i(w) - \widetilde{y}_i)^2 = \frac{1}{2}\|y(w) - \widetilde{y}\|^2, \qquad (1.8)$$

где $y(w)$ и $\widetilde{y}$ — соответственно вектор выходов НС ПР на обучающем множестве и вектор указаний учителя. Цель обучения НС — определение такого вектора весов w^*, чтобы функционал (1.7) или (1.8) принимал минимальное значение:

$$w^* = \arg\min_{w\in\mathbb{R}^s} J(w). \qquad (1.9)$$

Фактически, обучение НС — многоэкстремальная задача параметрической идентификации нелинейной модели, которая может осуществляться с помощью обширного аппарата теории оптимизации.

1.3. Перцептроны

Пороговая функция активации — первая функция, которая применялась в НС. Нейрон, у которого функция активации является пороговой, называется *перцептроном* (*персептроном* — в зависимости от прочтения английского слова «perceptron»). Следует иметь в виду, что в ряде работ под перцептроном понимают НС, состоящие из нескольких слоев нейронов с пороговыми функциями активации. Такие НС в отличие от перцептрона-нейрона называют многослойными перцептронами. Если же такая сеть содержит и несколько выходов, то перцептрон называется многовыходным. В данной работе под перцептроном понимается только нейрон. Выход перцептрона в случае использования пороговой функции-1 получается по формуле

$$\sigma(net) = \begin{cases} 1, & w_1x_1 + w_2x_2 + \ldots + w_nx_n \geqslant T, \\ 0, & w_1x_1 + w_2x_2 + \ldots + w_nx_n < T, \end{cases}$$

в случае пороговой функции-2

$$\sigma(net) = \begin{cases} 1, & w_1x_1 + w_2x_2 + \ldots + w_nx_n \geqslant T, \\ -1, & w_1x_1 + w_2x_2 + \ldots + w_nx_n < T, \end{cases}$$

Перцептрон может, в частности, реализовывать булевы функции.

Пример 1.1.

Рассмотрим булеву функцию $y = f(x_1, x_2) = x_1 \vee x_2 \in \{0, 1\}$ двух переменных $x_1, x_2 \in \{0, 1\}$, реализующую функцию «ИЛИ». Таблица истинности для этой функции приведена в табл. 1.1. Эта таблица задает соответствие выхода функции комбинациям значений входных переменных и может рассматриваться как обучающее множество для перцептрона, состоящее из четырех примеров. Последний столбец — значения функции — содержит указания учителя.

Таблица 1.1. Таблица истинности для булевой функции $y = x_1 \vee x_2$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Рассмотрим перцептрон, содержащий два входа (рис. 1.10). Задача его обучения состоит в определении весов w_1 и w_2, а также порогового значения T, с тем, чтобы перцептрон выдавал правильные выходные значения на всех примерах обучающего множества из табл. 1.1.

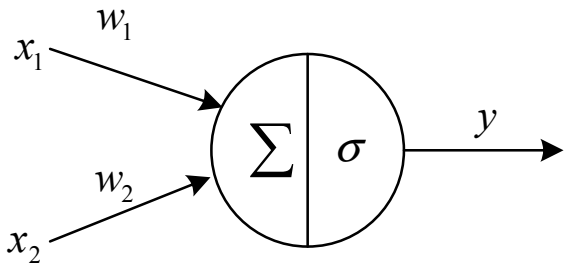

Рис. 1.10. Перцептрон с двумя входами

Данная задача имеет бесконечное множество решений. Рассмотрим следующие значения весов: $w_1 = w_2 = 0.5$, $T = 0.4$. Легко проверить, что при данных значениях весов перцептрон будет выдавать значения, совпадающие

с указаниями учителя. Например, при $x_1 = 1$, $x_2 = 0$:

$$net = \langle w, x \rangle = 0.5 \cdot 1 + 0.5 \cdot 0 = 0.5$$

и т.к. $net = 0.5 \geqslant T = 0.4$, то на выходе перцептрона значение будет равно 1. Аналогично и на других примерах обучающего множества.

Заметим, что пороговая функция разбивает плоскость переменных x_1 и x_2 на две полуплоскости: границей является прямая $0.5x_1 + 0.5x_2 = 0.4$. Решение задачи может быть проиллюстрировано следующим образом (рис. 1.11). На этом рисунке закрашенные точки соответствуют единичным значениям булевой функции, а пустые — нулевым. Прямая проходит таким образом, что все закрашенные точки находятся в одной полуплоскости (в эту же полуплоскость включается и прямая), а пустые точки — в другой.

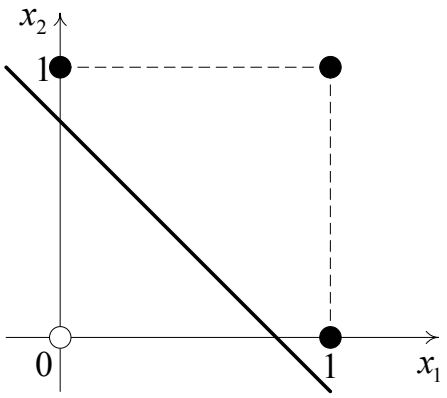

Рис. 1.11. Решение задачи моделирования перцептроном булевой функции $y = x_1 \vee x_2$

В качестве других значений весов, решающих задачу, могут быть выбраны, например, такие: $w_1 = 0.6$, $w_2 = 0.4$, $T = 0.3$. ◁

Чтобы унифицировать и упростить запись алгоритмов работы с перцептронами, пороговое значение представляют в виде дополнительного веса, соответствующего фиктивному единичному входному значению (рис. 1.2). Действительно, если перенести из неравенства

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \geqslant T$$

пороговое значение T в левую часть, то получится

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n - 1 \cdot T \geqslant 0.$$

Если теперь обозначить $(-T)$ через w_0, то получится такое неравенство:

$$w_0x_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n = \sum_{i=0}^{n} w_ix_i \geqslant 0,$$

где $x_0 = 1$. При этом следует заметить, что обучающее множество изменится: добавится первый столбец, состоящий из единиц, соответствующий входу x_0.

Если существуют такие веса w_i, $i = 0, \ldots, n$, что $\langle w, x\rangle \geqslant 0$ для всех точек $x \in C_1$ и $\langle w, x\rangle < 0$ для всех точек $x \in C_2$, то такая задача называется *линейно разделимой*. Как показано в примере 1.1, булева функция $y = x_1 \vee x_2$ является линейно разделимой.

Если в примере 1.1 веса перцептрона каким-либо образом подбирались, то конструктивный подход состоит в применении алгоритма обучения перцептрона 1.

Алгоритм 1. Обучение перцептрона

1. Задание константы $\eta > 0$ — шага обучения.

2. Инициализация весов w_i, $i = 0, \ldots, n$ небольшими случайными значениями. Ошибка работы перцептрона $E := 0$. Номер примера обучающего множества $m := 1$.

3. Начало обучения. $\widetilde{x} := \widetilde{x}_m$, $\widetilde{y} := \widetilde{y}_m$. Вычисление выхода перцептрона при текущем входном векторе $\widetilde{x}$: $y = y(\widetilde{x})$.

4. Обновление весов

$$w := w - \eta(\widetilde{y} - y)\widetilde{x}.$$

5. Накопление ошибки

$$E := E + \frac{1}{2}\|\widetilde{y} - y\|^2.$$

6. Если $m < k$, то $m := m + 1$ и переход на шаг 3.

7. Если $E = 0$, то прекращение вычислений (решение найдено), иначе $E := 0$, $m := 1$ и переход на шаг 3.

◁

Сделаем следующие замечания к алгоритму:

— Алгоритм легко обобщается на случай многовыходных перцептронов. Обобщить его на многослойные перцептроны не получается, что связано с отсутствием информации об указаниях учителя для перцептронов внутренних слоев.

— По сути, константа обучения — длина шага вдоль направления минимизации функции. Вследствие этого константа η должна задаваться достаточно малой (в пределах 0.1–1.0), чтобы не перескочить минимум.

— На шаге 4 веса будут обновлены лишь в том случае, когда на текущем примере перцептрон будет выдавать выход, отличный от указания учителя.

— На шаге 7 условие $E = 0$ выполнится в том случае, когда перцептрон выдаст значения, совпадающие со всеми указаниями учителя, т.е. для всех $m = 1, \ldots, k$.

Алгоритм обучения перцептрона найдет некоторое решение задачи, если оно существует. Об этом говорит следующая теорема.

Теорема (о сходимости алгоритма обучения перцептрона). Если задача является линейно разделимой, то цикл в алгоритме выполнится конечное число раз. □

Пример 1.2.

Рассмотрим булеву функцию $y = \overline{x}_1 \vee x_2$. Обучающее множество для этой функции приведено в табл. 1.2. Количество примеров $k = 4$.

Применим для нахождения весов перцептрона алгоритм 1.

1. Зададим константу обучения: $\eta := 0.8$.

2. Инициализируем значения: $E := 0$, $m := 1$, а также значения вектора весов:

$$w := \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

Таблица 1.2. Таблица истинности для булевой функции $y = \overline{x}_1 \vee x_2$

x_0	x_1	x_2	y
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

3. Выбираем первый пример (т.к. $m = 1$) из обучающего множества

$$\widetilde{x} := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \widetilde{y} := 1.$$

Уровень активности нейрона: $net = 1 \cdot 1 + 0 \cdot 0 + 2 \cdot 0 = 1$. Выход перцептрона: $y = 1$.

4. Т.к. $y = \widetilde{y}$, то обновление весов не происходит.

5. $E := 0 + \dfrac{1}{2} \cdot 0 = 0$.

6. Т.к. $m < 4$, увеличиваем m: $m := 1 + 1 = 2$ и переходим на шаг 3.

3. Выбираем второй пример из обучающего множества

$$\widetilde{x} := \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \widetilde{y} := 1.$$

Уровень активности нейрона: $net = 3$, выход перцептрона: $y = 1$.

4. Т.к. $y = \widetilde{y}$, то обновление весов не происходит.

5. $E := 0 + \dfrac{1}{2} \cdot 0 = 0$.

6. Т.к. $m < 4$, увеличиваем m: $m := 2 + 1 = 3$ и переходим на шаг 3.

3. Выбираем третий пример из обучающего множества

$$\widetilde{x} := \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \widetilde{y} := 0.$$

Уровень активности нейрона: $net = 1$, выход перцептрона: $y = 1$.

4. Т.к. $y \neq \widetilde{y}$, то обновляем веса

$$w := \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} + 0.8 \cdot (0 - 1) \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.2 \\ -0.8 \\ 2 \end{bmatrix}.$$

5. $E := 0 + \frac{1}{2} \cdot \|0 - 1\|^2 = \frac{1}{2}$.

6. Т.к. $m < 4$, увеличиваем m: $m := 3 + 1 = 4$ и переходим на шаг 3.

3. Выбираем четвертый пример из обучающего множества

$$\widetilde{x} := \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \widetilde{y} := 1.$$

Уровень активности нейрона рассчитывается на основании обновленного вектора весов: $net = 1.4$, выход перцептрона: $y = 1$.

4. Т.к. $y = \widetilde{y}$, то обновление весов не происходит.

5. $E := \frac{1}{2} + \frac{1}{2} \cdot 0 = \frac{1}{2}$.

6. Т.к. $m = 4$, переходим на следующий шаг 7.

7. Т.к. $E = \frac{1}{2} \neq 0$, это означает, что перцептрон выдал неверные ответы на некоторых примерах, поэтому придется заново повторить цикл обучения. Присваиваем: $E := 0$, $m := 1$. Переходим на шаг 3 (начиаем новый цикл обучения).

3. Выбираем первый пример (т.к. $m = 1$) из обучающего множества

$$\widetilde{x} := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \widetilde{y} := 1.$$

Уровень активности нейрона: $net = 0.2$, выход перцептрона: $y = 1$.

4. Т.к. $y = \widetilde{y}$, то обновления весов не происходит.

5. $E := 0 + \frac{1}{2} \cdot 0 = 0$.

6. Т.к. $m < 4$, увеличиваем m: $m := 2$ и переходим на шаг 3.

3. Выбираем второй пример (т.к. $m = 2$) из обучающего множества

$$\widetilde{x} := \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \widetilde{y} := 1.$$

Уровень активности нейрона: $net = 2.2$, выход перцептрона: $y = 1$.

4. Т.к. $y = \widetilde{y}$, то обновления весов не происходит.

5. $E := 0 + \frac{1}{2} \cdot 0 = 0$.

6. Т.к. $m < 4$, увеличиваем m: $m := 2$ и переходим на шаг 3.

3. Выбираем третий пример из обучающего множества

$$\widetilde{x} := \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \widetilde{y} := 0.$$

Уровень активности нейрона: $net = -0.6$, выход перцептрона: $y = 0$.

4. Т.к. $y = \widetilde{y}$, то обновления весов не происходит.

5. $E := 0 + \frac{1}{2} \cdot 0 = 0$.

6. Т.к. $m < 4$, увеличиваем m: $m := 2$ и переходим на шаг 3.

3. Выбираем четвертый пример из обучающего множества

$$\widetilde{x} := \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \widetilde{y} := 1.$$

Уровень активности нейрона: $net = 1.4$, выход перцептрона: $y = 1$.

4. Т.к. $y = \widetilde{y}$, то обновления весов не происходит.

5. $E := 0 + \frac{1}{2} \cdot 0 = 0$.

6. Т.к. $m = 4$, переходим на следующий шаг 7.

7. Т.к. $E = 0$, это означает, что процесс обучения завершился.

Итак, получили следующий вектор весов:

$$w = \begin{bmatrix} 0.2 \\ -0.8 \\ 2 \end{bmatrix}.$$

На рис. 1.12 приведен график, отражающий поведение перцептрона. Разделяет плоскость прямая $0.2 - 0.8x_1 + 2x_2 = 0$.

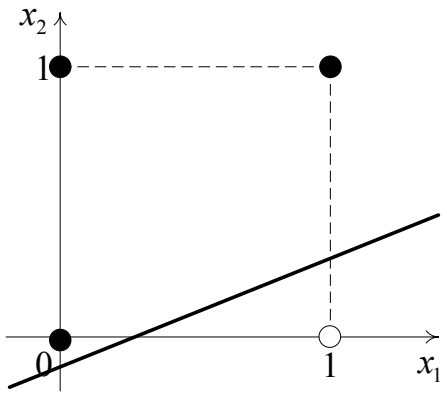

Рис. 1.12. Решение задачи моделирования перцептроном булевой функции $y = \overline{x}_1 \vee x_2$

◁

В упомянутой выше работе Минского и Пейперта было показано, что перцептрон не может успешно решить задачу «исключающего ИЛИ» (XOR). Эта задача не является линейно разделимой. Неразрешимость данной задачи можно проверить, посмотрев на рис. 1.13. Действительно, нельзя найти ни одну такую прямую, чтобы она разделяла плоскость на две части, в одной части которой были бы закрашенные точки, а в другой — пустые.

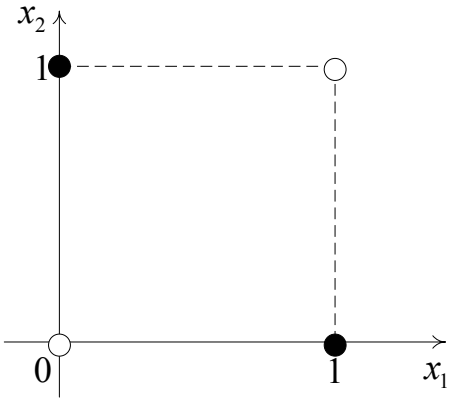

Рис. 1.13. Иллюстрация проблемы «исключающего ИЛИ»

2. Нейронные сети прямого распространения

2.1. Структура

Совокупность нейронов, состоящих из нескольких слоев и связанных друг с другом таким образом, что выходы нейронов одного слоя передают сигналы на входы нейронов следующего слоя, называется *нейронной сетью прямого распространения* (НС ПР) (рис. 2.1). Нейроны одного слоя не связаны друг с другом, также не связаны нейроны слоев, не являющихся соседними. Входы НС ПР выделяют в отдельный слой – входной. Элементы входного слоя фактически не являются нейронами, так как не преобразуют информацию, а лишь распределяют ее. В связи с этим, при подсчете количества слоев НС ПР входной слой обычно (за исключением некоторых работ) не учитывают. Слой, выходы нейронов которого не передаются на входы других нейронов, называется выходным — эти выходы образуют выходы всей НС. Все остальные слои называются *скрытыми* или *промежуточными*. Таким образом, m-слойная НС ПР состоит из одного входного слоя, $(m-1)$ скрытых слоев и одного выходного, m-го, слоя. Вычисление результатов производится сетью последовательно, в направлении от входного слоя к выходному. Такие сети называют также *слоистыми* (более точно, к НС ПР можно относить еще и сети с радиальными базисными функциями, и сети Кохонена, но в в силу своей специфики они выделяются в отдельные классы). На рис. 2.1 приведена многовыходная НС ПР.

Пусть НС ПР состоит из m слоев, в l-ом слое которой находится N_l нейронов, $l = 0, \ldots, m$; $N_0 = n$ — количество входов НС ПР. Пусть $w_j^{(l,i)}$ — j-й вес i-го нейрона l-го слоя, $l = 1, \ldots, m$, $i = 1, \ldots, N_l$, $j = 0, \ldots, N_{l-1}$. Функционирование i-го нейрона l-го слоя задается формулой

$$y^{(l,i)} = \sigma^{(l,i)}\left(net^{(l,i)}\right) = \sigma^{(l,i)}\left(\sum_{j=0}^{N_{l-1}} w_j^{(l,i)} y^{(l-1,j)}\right), \qquad (2.1)$$

где $y^{(l-1,0)} = 1$ — фиктивный единичный вход нейрона; $y^{(l-1,j)}$ — выход j-го нейрона $(l-1)$-го слоя, $j = 1, \ldots, N_{l-1}$; $y^{(0,j)} = x_j$ — j-й вход НС ПР,

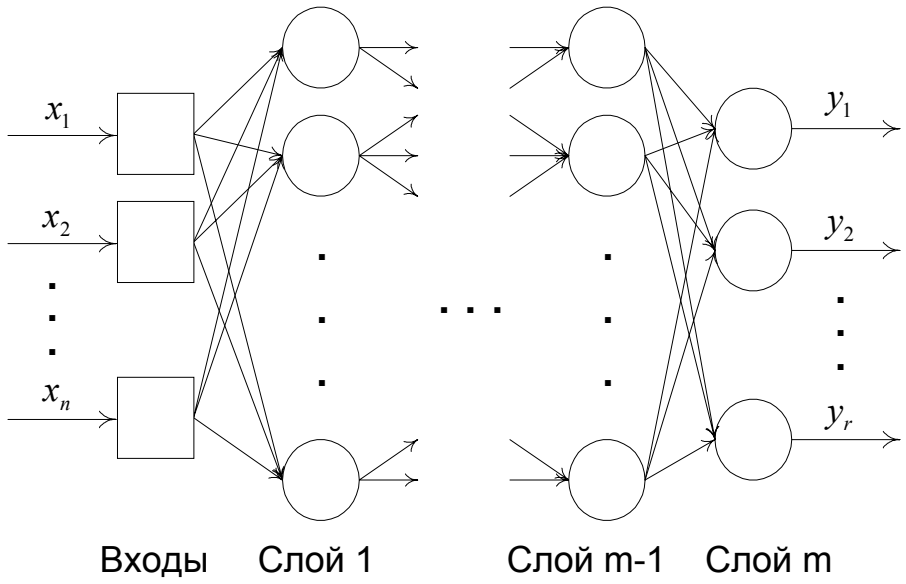

Рис. 2.1. m-слойная нейронная сеть прямого распространения

$j = 1, \ldots, n$. Хотя в общем случае функции активации нейронов могут отличаться друг от друга, обычно для всех нейронов скрытых слоев они берутся одинаковыми (чаще всего — сигмоидными логистическими-1). Нейроны выходного слоя в подавляющем большинстве случаев содержат единичную функцию активации для того, чтобы сеть могла выдавать любые действительные выходные значения, а не только ограниченные областью значений функции активации.

Выходы нейронов l-го слоя представляются вектор-столбцом $y^{(l)} \in \mathbb{R}^{N_l}$, веса нейронов этого слоя — матрицей $W^{(l)} \in \mathbb{R}^{N_l \times (1+N_{l-1})}$, в которой i-я строка состоит из весов нейрона (l, i). Матрица $W^{(l)}$ представляет блочную матрицу $W^{(l)} = [w_0^{(l)} \mid W_1^{(l)}]$, где $w_0^{(l)} \in \mathbb{R}^{N_l}$ — веса нейронов, соответствующие фиктивным единичным входам; $W_1^{(l)} \in \mathbb{R}^{N_l \times N_{l-1}}$ — матрица весов между нейронами $(l-1)$-го и l-го слоев. На основе введенных обозначений функционирование l-го слоя НС ПР в векторно-матричной форме описывается следующей формулой:

$$y^{(l)} = \sigma^{(l)} \left(w_0^{(l)} + W_1^{(l)} y^{(l-1)} \right), \tag{2.2}$$

где $\sigma^{(l)}$ — векторная функция векторного аргумента, соответствующая применению функций активации нейронов к каждому нейрону l-го слоя. На основе данного соотношения функционирование всей НС ПР может быть представ-

лено в виде

$$y = \sigma^{(m)}\Big(w_0^{(m)} + W_1^{(m)}\sigma^{(m-1)}\Big(\dots\Big(w_0^{(2)} + W_1^{(2)}\sigma^{(1)}\Big(w_0^{(1)} + W_1^{(1)}x\Big)\Big)\dots\Big)\Big), \quad (2.3)$$

где $x \in \mathbb{R}^n$ — вектор входов, $y \in \mathbb{R}^r$ — вектор выходов НС НС (выходы нейронов выходного слоя). В случае применения единичной функции активации в выходных нейронах и одинаковой функции активации во всех скрытых нейронах преобразование (2.3) упрощается и принимает вид

$$y = w_0^{(m)} + W_1^{(m)}\sigma\left(\dots\left(w_0^{(2)} + W_1^{(2)}\sigma\left(w_0^{(1)} + W_1^{(1)}x\right)\right)\dots\right). \quad (2.4)$$

Из формул (2.3) и (2.4) видно, что НС ПР реализует сложную нелинейную функциональную зависимость между входными и выходными величинами. НС ПР имеет характерную суперпозиционную линейно-нелинейную по весам структуру. Линейно-нелинейный характер означает, что веса состоят из двух частей — линейно входящих ($W^{(m)}$) и нелинейно входящих (все остальные веса). Более того, эта «линейно-нелинейность» характерна для каждого слоя, а в контексте всей НС она, как слои гамбургера, суперпозиционным образом накладывается друг на друга. Указанная специфика НС ПР, в особенности ее суперпозиционный характер, лежит в основе многих алгоритмов обучения НС ПР.

Заметим, что многослойные сети не приводят к увеличению вычислительной мощности по сравнению с однослойными, если нейроны скрытых слоев обладают единичными активационными функциями. Действительно, в этом случае вычисление выхода слоя заключается в умножении входного вектора на первую весовую матрицу с последующим умножением результирующего вектора на вторую весовую матрицу и т.д. Это означает, что любая многослойная линейная сеть может быть заменена эквивалентной ей однослойной сетью:

$$y = w_0^{(m)} + W_1^{(m)}\left(\dots\left(w_0^{(2)} + W^2\left(w_0^{(1)} + W_1^{(1)}x\right)\right)\dots\right) = w_0 + Wx.$$

Одновыходной *стандартной НС ПР* (иногда называемой полутораслойным предиктором) называется двухслойная НС ПР с единичной функцией

активации у нейронов выходного слоя. Работа одновыходной стандартной НС ПР описывается формулой

$$y = \sum_{i=1}^{q} w_i \sigma \left(w_{i0} + \sum_{j=1}^{n} w_{ij} x_j \right) = \sum_{i=1}^{q} w_i \sigma \left(\sum_{j=0}^{n} w_{ij} x_j \right), \qquad (2.5)$$

где q — количество нейронов единственного скрытого слоя; w_{ij} — вес i-го нейрона скрытого слоя, идущий от j-го входа, $j = 1, \ldots, n$, w_{i0} — вес фиктивного единичного входа, w_i — вес нейрона выходного слоя, идущий от i-го нейрона скрытого слоя. В многовыходной стандартной НС ПР соотношение (2.5) описывает работу каждого выхода. Схема многовыходной стандартной НС ПР приведена на рис. 2.2.

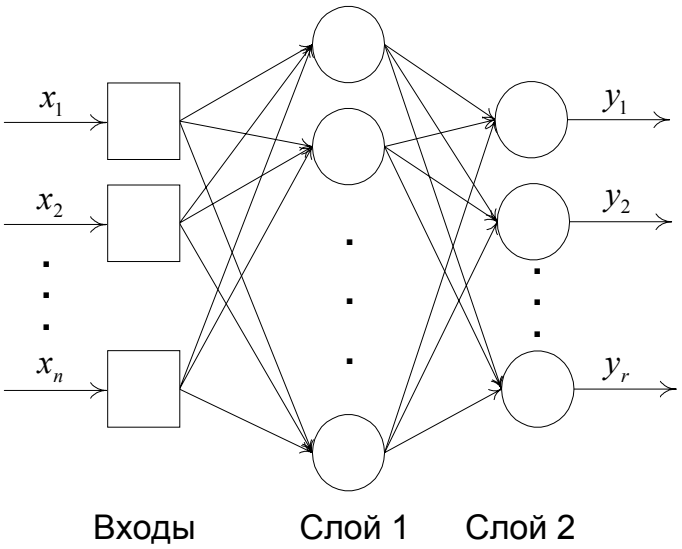

Рис. 2.2. Стандартная нейронная сеть прямого распространения

Теоретической основой успешного применения НС ПР на практике являются их универсальные аппроксимационные способности. Уже стандартные НС, т.е. двухслойные НС ПР, являются аппроксиматорами непрерывных функций нескольких переменных, заданных на компактном множестве. Другими словами, они способны приблизить непрерывную функцию с любой заданной точностью. Эти возможности восходят к хорошо известным теоремам из теории аппроксимации (теорема Вейерштрасса о приближении полиномами, теорема Стоуна, работы В.И. Арнольда и А.Н. Колмогорова). Результаты исследований универсальных аппроксимационных свойств НС ПР были полу-

чены в 1989 г. одновременно несколькими авторами. Результат исследований, приведенный Фунахаши (Funahashi), отражен в следующей теореме.

Теорема. Пусть $f^*(x_1, x_2, \ldots, x_n)$ — непрерывная функция, определенная на компактном множестве, и $\varepsilon > 0$ – точность аппроксимации. Существует такое число $q \in \mathbb{N}$ и набор чисел $w_{ij}, w_i \in \mathbb{R}$, что функция

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{q} w_i \sigma \left(w_{i0} + \sum_{j=1}^{n} w_{ij} x_j \right),$$

где σ — непостоянная ограниченная монотонно возрастающая непрерывная функция (например, сигмоидная логистическая), приближает исходную функцию $f^*(x_1, x_2, \ldots, x_n)$ с погрешностью, не превышающей ε на всей области определения, т.е.

$$\sup_{(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n} | f(x_1, x_2, \ldots, x_n) - f^*(x_1, x_2, \ldots, x_n) | \leqslant \varepsilon.$$

□

Это теорема как раз и говорит о том, что любую непрерывную функцию нескольких переменных можно с любой точностью приблизить с помощью стандартной НС ПР с достаточным количеством нейронов в скрытом слое. Теорема не говорит о том, сколько нейронов и какие веса необходимы для аппроскимации функции с заданной точностью.

2.2. Методика применения нейронных сетей

На рис. 2.3 схематично представлена методика практического применения НС ПР. Дадим краткую характеристику методики применения НС ПР.

Этап 1. Информативное множество входных $x \in \mathbb{R}^n$ и выходных $y \in \mathbb{R}^r$ величин выбирается, исходя из поставленной задачи. Следует обратить внимание на то, что неосновательный и поспешный выбор входных и выходных переменных отрицательно влияет на качество построения моделей. Данный процесс выбора неформализуем, он в сильной мере зависит от конкретной задачи, а также от опыта и интуиции эксперта. После определения входов и выходов составляется множество вход-выходных данных, еще не являющееся, однако, обучающим множеством. Эти данные могут либо собираться в

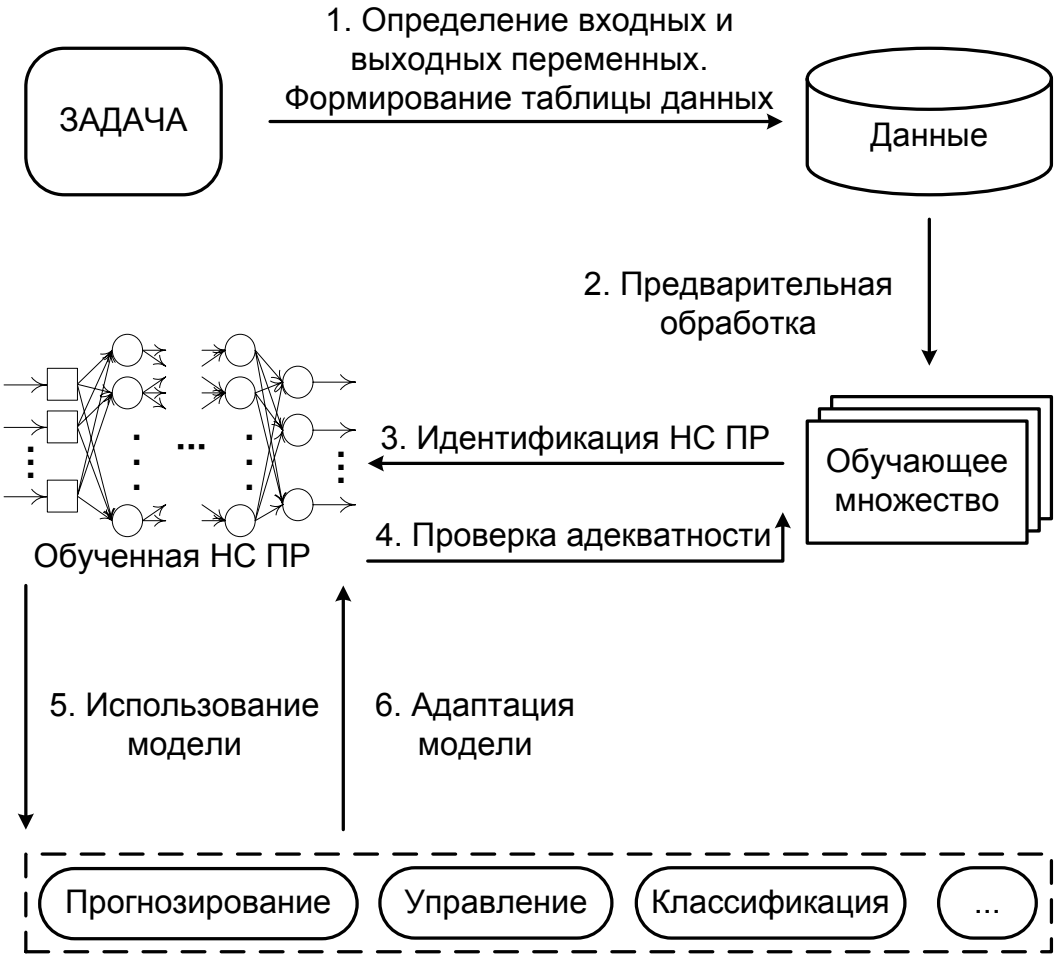

Рис. 2.3. Схема применения НС ПР

ходе проведения экспериментов, либо быть заданными. Множество должно быть репрезентативным (отражать реальное поведение моделируемого объекта). Чем больше объем, тем лучше может быть настроена сеть на решение задач.

Этап 2. С целью дальнейшего использования данных в обучении НС ПР применяется предварительная обработка (предобработка) данных. Можно выделить следующие шаги предобработки информации, которые применяются в зависимости от конкретной задачи:

1. Преобразование данных, заданных в качественном виде (то есть в виде понятий, вербальных оценок), к числовому виду.
2. Восстановление отсутствующих данных в обучающем множестве (восстановление пробелов).
3. Удаление неестественных, ошибочных, сильно искаженных данных, не

отражающих реального поведения моделируемого объекта.

4. Переход от абсолютных значений к относительным.
5. Нормировка значений переменных.

Первые четыре шага актуальны для всех задач моделирования. Восстановление данных, выявление и удаление ошибочной информации обычно производится с применением статистических процедур. Переход к относительным значениям целесообразен при моделировании динамических зависимостей (например, при прогнозировании временных рядов это позволяет получить более стационарный ряд). Спецификой нейросетевого моделирования является нормировка значений переменных.

Как отмечалось выше, используемые в НС функции активации обладают свойством насыщения: начиная с определенных входных значений, функция будет выдавать практически одно и то же значение. Данное обстоятельство может отрицательно отразиться на обучении НС, т.к. при начальной инициализации весов даже в небольших пределах (например, значениями из отрезка $[-0.1, 0.1]$) входные значения могут приводить к появлению на выходе неотличимых значений. В связи с этим, данные обучающего множества рекомендуется нормализовать, т.е. привести (обычно с помощью линейных преобразований) к некоторому диапазону $[a, b]$. Наиболее часто выбираются диапазоны $[0, 1]$ и $[-1, 1]$. Для каждой j-й входной переменной x_j определяется диапазон изменения значений $[x_j^{min}, x_j^{max}]$ на основе вход-выходной таблицы данных: $x_j^{min} = \min\limits_{i=1}^{k} \widetilde{x}_{ij}$ и $x_j^{max} = \max\limits_{i=1}^{k} \widetilde{x}_{ij}$.

Нормализация значений $\widetilde{x}_{ij}$ в диапазон $[0, 1]$ осуществляется по формуле

$$\frac{\widetilde{x}_{ij} - x_j^{min}}{x_j^{max} - x_j^{min}}, \tag{2.6}$$

а в диапазон $[-1, 1]$ — по формуле

$$2\frac{\widetilde{x}_{ij} - x_j^{min}}{x_j^{max} - x_j^{min}} - 1. \tag{2.7}$$

Заметим, что нормализация выходных значений необязательна.

Этап 3. Центральным этапом решения практических задач с использованием НС ПР является идентификация нейросетевой модели. Данная задача, как и вообще задача построения математической модели в аналитическом виде, состоит из двух компонентов:

1. Структурной идентификации.
2. Параметрической идентификации.

При построении нейросетевых моделей структурная идентификация является частично решенной, т.к. задана структура формирования выходных значений НС ПР. Остается открытым вопрос о количестве скрытых слоев и количестве нейронов в каждом из них. Эта задача, называемая также задачей определения оптимальной структуры НС ПР, должна решаться таким образом, чтобы НС ПР могла показывать хорошие результаты на данных, не входящих в обучающее множество. Теоретически, увеличение количества слоев и нейронов в сети приводит к уменьшению ошибки обучения. Однако это с определенного момента начинает отрицательно сказываться на способности НС ПР описывать зависимость на данных, не включенных в обучающее множество. Возможны три стратегии выбора структуры НС ПР:

— Генерация набора НС, не зависимых друг от друга.
— Контрастирование НС, заключающееся в оценке значимости нейронов в текущей НС и удалении наименее значимых из них.
— Конструктивный подход, заключающийся в последовательном наращивании количества слоев и/или нейронов в скрытых слоях, начиная с сети минимальной структуры.

Каждый из подходов имеет свои преимущества и недостатки. Процедура генерации независимых структур НС проста в реализации, однако может привести к пропуску хорошей структуры. Для осуществления контрастирования необходимо иметь начальную сеть с достаточно большим количеством весов. Конструктивный подход представляется наиболее рациональным.

Параметрическая идентификация НС ПР — это, как указывалось выше, и есть обучение. Процесс обучения будет подробно рассмотрен далее.

Этап 4. Проверка адекватности модели очень важна при практическом применении. Основной подход для решения этой задачи состоит в выделении из исходной таблицы вход-выходных данных тестового множества (около 20%). Остальная часть является обучающим множеством. Тестовое множество состоит из данных, не использующихся в процессе обучения (т.е. фактически в обучающее множество они не входят). Часто производится обучение нескольких НС ПР фиксированной структуры, далее среди которых по ошибке на тестовом множестве выбирается наилучшая. После окончания процесса обучения на обучающем множестве рассчитывается *ошибка обучения*, а на тестовом множестве — *ошибка обобщения*. Более адекватной считается та НС ПР, которая показывает меньшую ошибку обобщения (ошибка обучения может больше, чем для других структур).

Этап 5. Широкие возможности применения НС при решении широкого круга задач, особенно часто встречающихся в экономической и технической сферах деятельности, опираются на их способность строить зависимости произвольной нелинейной сложности. Можно выделить следующие основные способы применения НС ПР:

— Прогнозирование — получение выходных величин для заданных входных значений.

— Управление — формирование управляющих сигналов с целью получения желаемых выходов управляемой системы.

— Классификация — соотнесение входного вектора, описывающего некоторый объект, к определенному классу объектов.

Многие другие задачи могут быть сведены к одному из рассмотренных способов использования НС ПР.

Этап 6. При использовании НС ПР с течением времени встает необходимость учета новой информации о предметной области, рассматриваемой в задаче, например, связанная с поступлением новой вход-выходной информации. Процесс построения новой адекватной нейросетевой модели очень трудоемок, поэтому перспективным является подход, заключающийся в подстройке используемой модели (обычно только значений весов сети).

2.3. Методы обучения нейронных сетей

Задача обучения НС (1.9) является нелинейной задачей о наименьших квадратах (НЗНК). Сущность оптимизации (в случае НС — минимизации) заключается в построении релаксационной последовательности векторов $w_0, w_1, \ldots$, на каждой итерации (шаге) уменьшающих значение целевой функции ($J(w_0) > J(w_1) > \ldots$). Процесс построения такой последовательности прекращается на t-й итерации в случае удовлетворения одного или нескольких критериев:

1. $\mid J(w_{t+1}) - J(w_t) \mid \leqslant \varepsilon_1$;
2. $\|w_{t+1} - w_t\| \leqslant \varepsilon_2$;
3. $\|\nabla_w J(w_t)\| \leqslant \varepsilon_3$,

где $\varepsilon_i, i = 1, 2, 3$ — предварительно задаваемые точности оптимизации. Наиболее часто используется 1-й критерий останова (практически перестает изменяться значение целевой функции), 2-й — реже всех (практически перестает изменяться вектор). Последний, 3-й, критерий может применяться только в случае недифференцируемых функций (он означает, что при данном векторе невозможно отыскать направление уменьшения значения целевой функции).

Для решения НЗНК, как правило, используются два типа методов оптимизации:

1. Общие, т.е. предназначенные для класса дифференцируемых функций.
2. Специальные, т.е. учитывающие квадратичный характер минимизируемого функционала.

Многовыходная НС ПР (r выходов) может быть заменена совокупностью r одновыходных НС, поэтому рассмотрим методы обучения лишь для случая одновыходных НС ПР ($r = 1$). Основная схема методов оптимизации представлена на рис. 2.4. Основными моментами при реализации алгоритма являются определение направления минимизации на текущем шаге и определение длины шага вдоль этого направления. Способ выбора направления дает название самому оптимизационному методу. Выбор длины шага — как правило, одномерная минимизация функции вдоль выбранного направления.

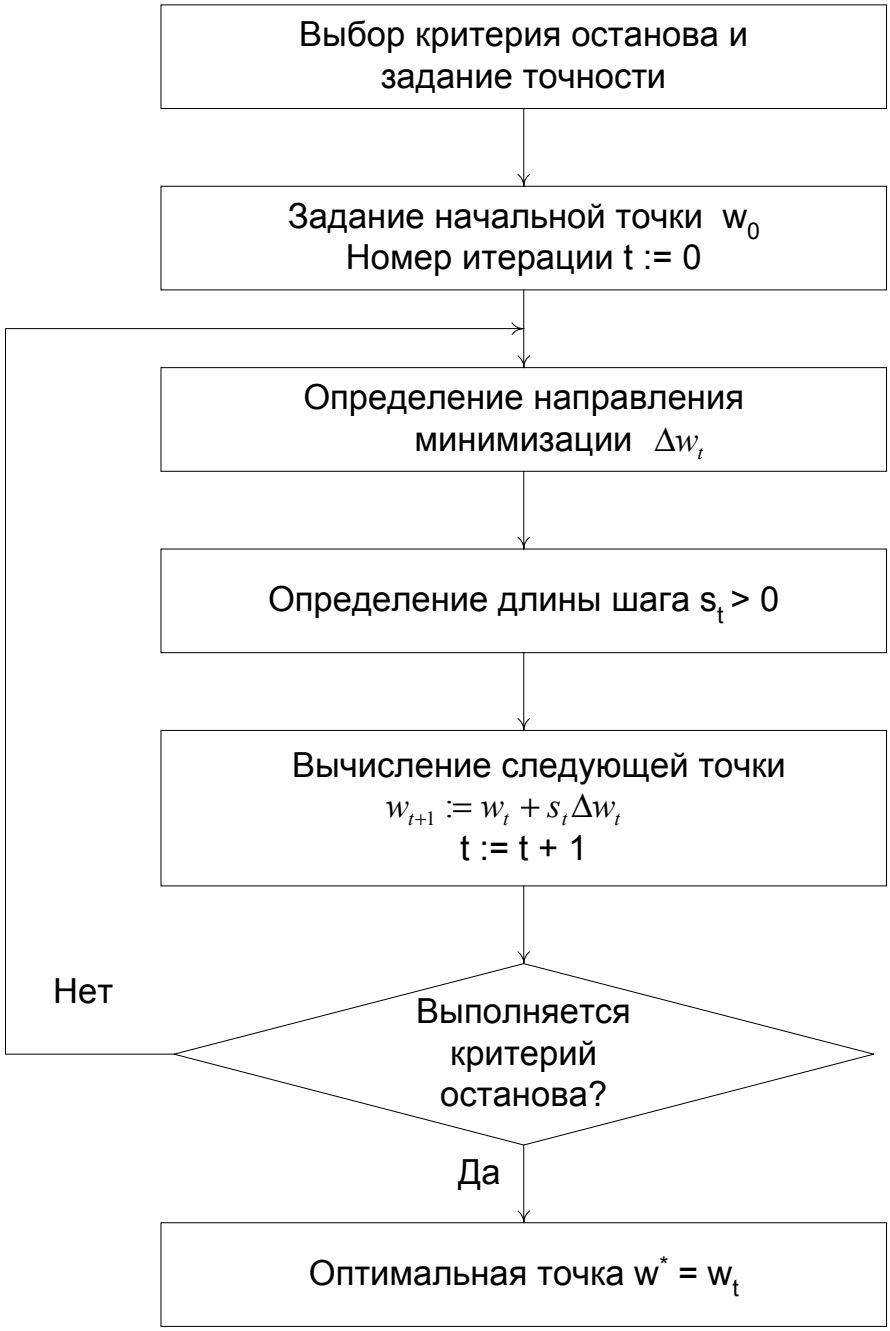

Рис. 2.4. Схема оптимизации функций

Направление оптимизации определяется исходя из поведения функции в окрестности текущей точки w. Эта информация заложена, прежде всего, в градиенте целевой функции по вектору оптимизируемых параметров. Аналогично тому, что производная показывает направление возрастания функции от одной переменной, градиент показывает направление возрастания функции от нескольких переменных.

2.3.1. Процедура обратного распространения ошибки

Эффективным способом нахождения градиента минимизируемого функционала (1.8) по вектору весов, учитывающим суперпозиционный характер сети на основе формулы производной сложной функции, является процедура *обратного распространения ошибки (ОРО)* (error backpropagation), предложенная Румельхартом. Вследствие аддитивности производной

$$\nabla_w J(w) = \nabla_w \left(\sum_{i=1}^{k} Q_i(w) \right) = \sum_{i=1}^{k} \nabla_w Q_i(w),$$

достаточно вывести формулу вычисления градиента по ошибке на одном примере обучающего множества (для мгновенного функционала качества). Обозначив для упрощения записи $Q_i(w)$ через $Q(w)$. Градиент $\nabla_w Q(w)$ состоит из частных производных $Q(w)$ по весам нейронов $w_j^{(m,i)}$, которые, исходя из (2.1), рассчитываются по формуле

$$\frac{\partial Q(w)}{\partial w_j^{(m,i)}} = \frac{\partial Q(w)}{\partial y^{(m,i)}} \cdot \frac{\partial y^{(m,i)}}{\partial net^{(m,i)}} \cdot \frac{\partial net^{(m,i)}}{\partial w_j^{(m,i)}}, \tag{2.8}$$

где последние два множителя определяются легко

$$\frac{\partial y^{(m,i)}}{\partial net^{(m,i)}} = \sigma'_{net}(net^{(m,i)}), \tag{2.9}$$

$$\frac{\partial net^{(m,i)}}{\partial w_j^{(m,i)}} = y^{(m-1,j)}. \tag{2.10}$$

Для нахождения первого множителя формулы (2.8) обозначим его как

$$s^{(m,i)} = \frac{\partial Q(w)}{\partial y^{(m,i)}}.$$

Нахождение $s^{(m,i)}$ опирается на рекуррентную процедуру, которая получается из формулы производной сложной функции от нескольких переменных:

$$s^{(m,i)} = \sum_{j=1}^{N_{m+1}} \frac{\partial Q}{\partial y^{(m+1,j)}} \frac{\partial y^{(m+1,j)}}{\partial y^{(m,i)}} = \sum_{j=1}^{N_{m+1}} s^{(m+1,j)} \sigma'_{net}(net^{(m+1,j)}) w_i^{(m+1,j)}. \tag{2.11}$$

Для вычислений по формуле (2.11) необходимо начальное условие. Оно получается из формулы (1.6) с учетом множителя $\frac{1}{2}$ из функционала (1.7):

$$s^{(M,1)} = \frac{\partial Q(w)}{\partial y^{(M,1)}} = \varepsilon(w) = y(w) - \widetilde{y}, \tag{2.12}$$

где $\widetilde{y}$ — указание учителя для поданного примера. Ошибка работы сети $\varepsilon(w)$ словно распространяется в направлении, обратном функционированию сети. Это и обусловило такое название метода нахождения градиента (обратное распространение ошибки).

Пример 2.1.

Рассмотрим стандартную НС ПР, содержащую один вход, один выход и два нейрона в скрытом слое. На рис. 2.5 приведена структура этой сети, обозначены входы и выходы каждого нейрона, а также веса нейронов. Внутри каждого нейрона указан его номер. Например, обозначение $(1, 2)$ показывает, что это 2-й нейрон 1-го слоя.

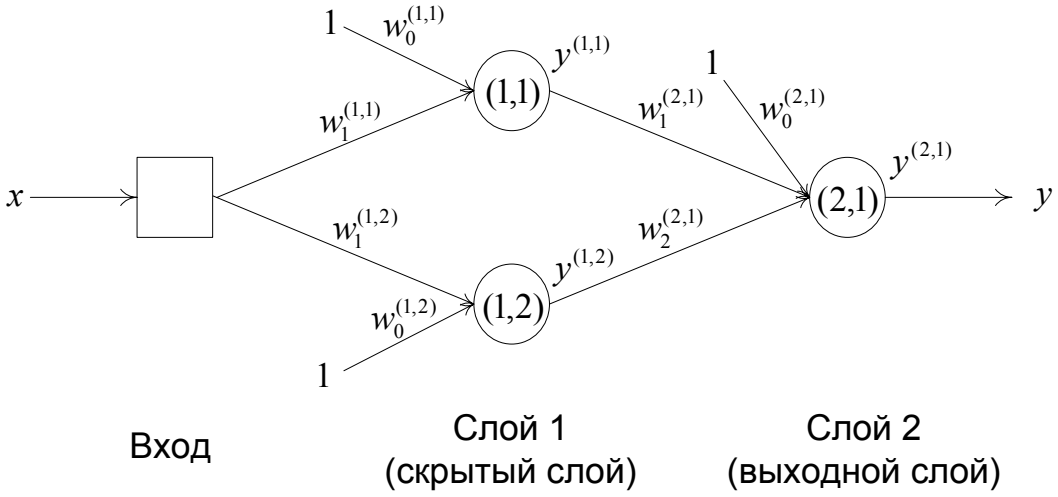

Рис. 2.5. Стандартная нейронная сеть с одним входом, одним выходом и двумя нейронами в скрытом слое

Распишем для этой сети формулы для вычисления градиента по методу ОРО. Вектор весов состоит из семи элементов:

$$w = \begin{bmatrix} w_0^{(1,1)} \\ w_1^{(1,1)} \\ w_0^{(1,2)} \\ w_1^{(1,2)} \\ w_0^{(2,1)} \\ w_1^{(2,1)} \\ w_2^{(2,1)} \end{bmatrix}.$$

Используя начальное условие (2.12), получаем для нейрона выходного слоя

$$s^{(2,1)} = \frac{\partial Q(w)}{\partial y^{(2,1)}} = y(w) - \widetilde{y}.$$

Нейрон выходного слоя имеет единичную функцию активации, поэтому

$$y^{(2,1)} = w_0^{(2,1)} + w_1^{(2,1)} y^{(1,1)} + w_2^{(2,1)} y^{(1,2)}.$$

Тогда из (2.11) получаем следующие выражения для нейронов скрытого слоя:

$$s^{(1,1)} = s^{(2,1)} \frac{\partial y^{(2,1)}}{\partial y^{(1,1)}} = s^{(2,1)} w_1^{(2,1)};$$

$$s^{(1,2)} = s^{(2,1)} \frac{\partial y^{(2,1)}}{\partial y^{(1,2)}} = s^{(2,1)} w_2^{(2,1)}.$$

Теперь можно записать формулу для вычисления частных производных, используя формулу (2.8). Например, для веса $w_0^{(1,1)}$ в предположении, что используется в качестве функции активации нейронов скрытого слоя используется сигмоидная логистическая функция-1, получаем

$$\frac{\partial Q(w)}{\partial w_0^{(1,1)}} = s^{(1,1)} \cdot \frac{\partial y^{(1,i)}}{\partial net^{(1,i)}} \cdot \frac{\partial net^{(1,1)}}{\partial w_0^{(1,1)}} = s^{(1,1)} \sigma(net^{(1,1)})(1 - \sigma(net^{(1,1)})) y^{(0,0)} =$$

$$= s^{(1,1)} y^{(1,1)} (1 - y^{(1,1)}).$$

Здесь учтено, что $y^{(0,0)} = 1$. Для веса $w_1^{(1,1)}$: $y^{(0,1)} = x$, поэтому

$$\frac{\partial Q(w)}{\partial w_1^{(1,1)}} = s^{(1,1)} \sigma(net^{(1,1)})(1 - \sigma(net^{(1,1)})) y^{(0,1)} = s^{(1,1)} y^{(1,1)} (1 - y^{(1,1)}) x.$$

Аналогично получаются и остальные формулы. В итоге градиент мгновенного функционала качества будет получаться следующим образом:

$$\nabla_w Q(w) = \begin{bmatrix} s^{(1,1)} y^{(1,1)} (1 - y^{(1,1)}) \\ s^{(1,1)} y^{(1,1)} (1 - y^{(1,1)}) x \\ s^{(1,2)} y^{(1,2)} (1 - y^{(1,2)}) \\ s^{(1,2)} y^{(1,2)} (1 - y^{(1,2)}) x \\ s^{(2,1)} y^{(2,1)} (1 - y^{(2,1)}) \\ s^{(2,1)} y^{(2,1)} (1 - y^{(2,1)}) y^{(1,1)} \\ s^{(2,1)} y^{(2,1)} (1 - y^{(2,1)}) y^{(1,2)} \end{bmatrix}.$$

◁

2.3.2. Обучение на основе локальных методов безусловной оптимизации дифференцируемых функций

Заменим (аппроксимируем) функционал (1.8) в окрестности текущего вектора весов w_t квадратичной моделью

$$J(w) \approx J(w_t) + \nabla_w J(w_t)(w - w_t) + \frac{1}{2}(w - w_t)^T \nabla_w^2 J(w_t)(w - w_t) \qquad (2.13)$$

или линейной

$$J(w) \approx J(w_t) + \nabla_w J(w_t)(w - w_t), \qquad (2.14)$$

где $\nabla_w J(w_t)$ — градиент функции по вектору w в точке w_t, $\nabla_w^2 J(w_t)$ — матрица Гессе в точке w_t. В зависимости от того, какая модель используется, выделяют следующие методы:

— градиентные (первого порядка), использующие градиент функции;

— ньютоновского типа (второго порядка), использующие градиент функции и матрицу Гессе;

— квазиньютоновского типа (методы переменной метрики, МПМ), использующие градиент и приближения матрицы Гессе.

Зная градиент целевой функции, можно использовать метод оптимизации дифференцируемой функции, называемый методом наискорейшего спуска. В этом методе функция минимизируются вдоль направления Δw_t, противоположного градиенту

$$\Delta w_t = -\frac{\nabla_w J(w_t)}{\|\nabla_w J(w_t)\|}.$$

Длина шага определяется на основе одномерной минимизации

$$s_t = \arg \min_{s_t \in \mathbb{R}, s_t > 0} J(w_t + s_t \Delta w_t).$$

Данный метод обычно показывает невысокую скорость сходимости. Следует заметить, что в некоторой литературе под ОРО понимается метод наискорейшего спуска, а не только способ вычисления градиента НС ПР, учитывающий ее суперпозиционную структуру.

Ускорение сходимости может быть достигнуто за счет использования информации о локальном поведении функции на предыдущей итерации. Распространенными методами данного типа являются методы сопряженных градиентов Флэтчера-Ривса и Полака-Рибьера. Выбор направления в этих методах осуществляется таким способом:

1. На начальном этапе ($t = 0$)

$$\Delta w_0 = -\nabla_w J(w_0).$$

2. На t-й итерации по методу Флэтчера-Ривса вычисляется значение

$$\beta_t = \frac{\|\nabla_w J(w_t)\|^2}{\|\nabla_w J(w_{t-1})\|^2},$$

а по методу Полака-Рибьера —

$$\beta_t = \frac{\nabla_w J(w_t)\left(\nabla_w J(w_t) - \nabla_w J(w_{t-1})\right)}{\|\nabla_w J(w_{t-1})\|^2}.$$

Направление минимизации находится по формуле

$$\Delta w_t = -\nabla_w J(w_t) + \beta_t \Delta w_{t-1}.$$

Эти методы эффективны и довольно просты в реализации.

Метод Ньютона основывается на расчете направления по формуле

$$\Delta w_t = -H(w_t)^{-1} \nabla_w J(w_t),$$

где $H(w_t) = \nabla_w^2 J(w_t)$ — матрица Гессе функционала качества. Сложность практического применения данного метода состоит в том, что вторые производные функционала часто сложно или совсем невозможно находить. Поэтому прибегают к использованию аппроксимаций матрицы Гессе, лежащих в основе квазиньютоновских методов оптимизации.

В квазиньютоновских методах на каждой итерации строятся положительно определенные (матрица Гессе является таковой) матрицы H_t на основе рекуррентного соотношения $H_{t+1} = H_t + \Delta H_t$ (аналогичным образом строится и аппроксимация обратной матрицы Гессе). Способ определения матрицы ΔH_t определяет название метода.

Формула пересчета обратной матрицы Гессе $W_t = H_t^{-1}$ по методу BFGS имеет вид

$$W_{t+1} = W_t + \frac{(s_t - W_t g_t)\, s_t^T + s_t\, (s_t - W_t g_t)^T}{g_t^T s_t} - \frac{(s_t - W_t g_t)\, g_t s_t s_t^T}{\left(g_t^T s_t\right)^2},$$

где $s_t = w_t - w_{t-1}$, $g_t = \nabla_w J(w_t) - \nabla_w J(w_{t-1})$. На начальной итерации $W_0 = I$ — направление оптимизации совпадает с антиградиентом. Пересчет по методу DFP опирается на формулу

$$W_{t+1} = W_t + \frac{s_t s_t^T}{s_t^T g_t} + \frac{W_t g_t g_t^T W_t}{g_t^T W_t g_t}.$$

2.3.3. Обучение на основе методов решения нелинейных задач о наименьших квадратах

Учет квадратичности минимизируемого функционала приводит к разработке алгоритмов, ориентированных на решение НЗНК. Применяя к квадратичной аппроксимации функционала (2.13) необходимое условие оптимума функции, приходим к уравнению

$$\nabla J(w_t) + \nabla^2 J(w_t)(w - w_t) = 0.$$

Для оптимизации данной модели может быть использован метод Ньютона с псевдообращением

$$\Delta w_t = -\left[\nabla^2 J(w_t)\right]^+ \nabla J(w_t), \tag{2.15}$$

где $\left[\nabla^2 J(w_t)\right]^+$ — псевдообратная матрица к исходной (матрица Мура-Пенроуза), являющаяся обобщением обратной матрицы на случай вырожденных и прямоугольных матриц (краткие сведения о псевдообратных матрицах см. в приложении). Использование псевдообращения позволяет не заботиться о невырожденности матрицы Гессе оптимизируемой функции. Минимизируемый функционал (1.8) для НЗНК можно представить в форме

$$J(w) = \frac{1}{2} R(w)^T R(w), \tag{2.16}$$

где $R(w) = y(w) - \widetilde{y}$ — вектор невязок, $y(w)$ — вектор выходов НС ПР, составленный на примерах обучающего множества, $\widetilde{y}$ — вектор указаний учителя.

Тогда будут справедливы следующие формулы:

$$\nabla J(w) = R'^T(w)R(w), \tag{2.17}$$

где $R'^T(w)$ — матрица Якоби вектора невязок, и

$$\nabla^2 J(w) = R'^T(w)R'(w) + G(w), \tag{2.18}$$

где $G(w)$ — матрица, содержащая информацию о вторых частных производных элементов вектора $R(w)$. Более точно,

$$G(w) = \sum_{i=1}^{k} J_i(w)G_i(w),$$

где $J_i(w)$ — i-й элемент вектора $J(w)$, а $G_i(w)$ – матрица Гессе для элемента $J_i(w)$. Подставляя формулы (2.17) и (2.18) в (2.15), получаем ньютоновское направление минимизации с псевдообращением:

$$\Delta w_t = -\left[R'^T(w_t)R'(w_t) + Q(w_t)\right]^+ R'^T(w_t)R(w_t). \tag{2.19}$$

Матрицы $G_i(w)$ сложно рассчитываются, поэтому формула (2.19) в чистом виде не применяется.

В основе большинства алгоритмов решения НЗНК лежит предположение о том, что с каждой итерацией слагаемое $R'^T(w_t)R'(w_t)$ становится все более значимым по сравнению с $G(w_t)$. Действительно, при $\|J(w_t)\|_{t\to\infty} \to 0$ матрица $G(w_t)$ стремится к нулевой. В методе Гаусса-Ньютона с псевдообращением полагается, что $G(w) = 0$, поэтому

$$\Delta w_t = -\left[R'^T(w_t)R'(w_t)\right]^+ R'^T(w_t)R(w_t) = -\left[R'(w_t)\right]^+ R(w_t). \tag{2.20}$$

Классический метод Гаусса-Ньютона получается при полном столбцовом ранге матрицы $R'(w_t)$; в этом случае

$$\left[R'(w_t)\right]^+ = \left[R'^T(w_t)R'(w_t)\right]^{-1} R'^T(w_t)$$

и направление определяется по формуле

$$\Delta w_t = -\left[R'^T(w_t)R'(w_t)\right]^{-1} R'^T(w_t)R(w_t).$$

Если матрица $G(w_t)$ аппроксимируется на каждом шаге диагональной матрицей $\mu_t I$, $\mu_t \geqslant 0$, получается метод Левенберга-Марквардта

$$\Delta w_t = -\left[R'^T(w_t)R'(w_t) + \mu_t I\right]^+ R'^T(w_t)R(w_t). \tag{2.21}$$

При этом шаг спуска вдоль найденного направления всегда является единичным, а параметр μ_t выбирается так, чтобы минимизировать целевую функцию. В отличие от других метод Левенберга-Марквардта не попадает под рассматриваемую схему оптимизации; он относится к классу методов доверительной окрестности. При $\mu_t \to 0$ направление почти совпадает с направлением метода Гаусса-Ньютона, при $\mu \to \infty$ — с направлением антиградиента.

В задачах с большим значением целевой функции (с большой невязкой) целесообразно использовать квазиньютоновские приближения H_t матрицы $G(w_t)$. Если используется формула BFGS для нахождения приближения, то получается следующее:

$$H_{t+1} = H_t - \frac{1}{s_t^T M_t s_t} M_t s_t s_t^T M_t + \frac{1}{g_t^T s_t} g_t g_t^T,$$

где $M_t = R_t'^T R_t' + H_t$, $s_t = w_t - w_{t-1}$ и $g_t = R_t'^T J(w_t) - R_{t-1}'^T J(w_{t-1})$. На начальной итерации берется $H = 0$. Периодически для уменьшения влияния погрешностей округлений снова полагают $H = 0$. Если приближение осуществляется на основе DFP-метода, то

$$H_{t+1} = H_t + \frac{(\widetilde{g}_t - H_t s_t)\, g_t^T + g_t\, (\widetilde{g}_t - H_t s_t)^T}{g_t^T s_t} - \frac{(\widetilde{g}_t - H_t s_t)^T\, s_t g_t g_t^T}{\left(g_t^T s_t\right)^2},$$

где $\widetilde{g}_t = R_t'^T J(w_t) - R_{t-1}'^T J(w_t)$. Следует отметить, что специальные методы решения НЗНК не всегда работают лучше методов оптимизации произвольных дифференцируемых функций.

Расчет производных выходов сети по весам НС ПР возможен с помощью процедуры, учитывающей суперпозиционный характер структуры сети аналогично методу ОРО. При этом процедура построения матрицы Якоби даже несколько упрощается $\nabla_w^T y(w_t)$ в связи с тем, что подлежит вычислению производная не функционала, а выхода сети.

2.3.4. Обучение на основе декомпозиции вектора весов с использованием псевдообращения

Простейшей НС ПР является НС, не содержащая скрытых слоев, с единичной функции активации (рис. 2.6). Функция $y = \sum_{i=1}^{n} w_n x_n$, реализуемая данной сетью, представляет собой линейную по весам функцию.

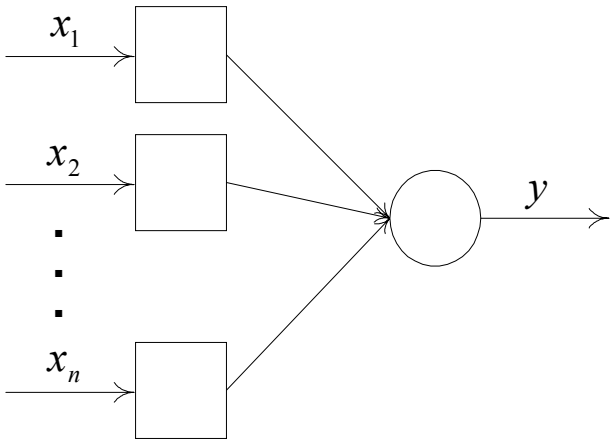

Рис. 2.6. Простейшая нейронная сеть

Ее обучение может быть произведено за один шаг и аналитически представлено в виде

$$w = \widetilde{X}^{+}\widetilde{y}, \tag{2.22}$$

где $\widetilde{X} \in \mathbb{R}^{k \times n}$ — матрица, составленная из входов обучающего множества:

$$\widetilde{X} = \begin{bmatrix} \widetilde{x}_{11} & \widetilde{x}_{12} & \dots & \widetilde{x}_{1n} \\ \widetilde{x}_{21} & \widetilde{x}_{22} & \dots & \widetilde{x}_{2n} \\ \dots & \dots & \dots & \dots \\ \widetilde{x}_{k1} & \widetilde{x}_{k2} & \dots & \widetilde{x}_{kn} \end{bmatrix}.$$

Формула (2.22) является псевдорешением уравнения

$$\widetilde{X}w = \widetilde{y}.$$

Перейдя к рассмотрению стандартной НС ПР, заметим, что ее функционирование можно представить формулой

$$y = \Psi(v)u,$$

где $\Psi(v)$ — матрица выходов нейронов скрытого слоя, зависящая от вектора нелинейно входящих весов v (веса матрицы $W^{(1)}$); u — вектор линейно входящих весов, состоящий из весов нейрона выходного слоя ($W^{(2)}$). Таким образом, вектор весов w разбит на два подвектора — v и u.

Для фиксированного вектора v аналогично (2.22) следует справедливость формулы

$$u = \Psi(v)^{+}\widetilde{y}. \tag{2.23}$$

Формулу (2.23) будем называть базовым линейно-нелинейным соотношением (БЛНС), связывающим линейно и нелинейно входящие в НС веса. Она позволяет находить оптимальный вектор u при заданном v, однако, как (2.22), полностью решить задачу обучения НС ПР не в состоянии. Для вектора v формулы, аналогичной (2.22), нет, т.к. неизвестно, какие значения должны выдавать нейроны скрытого слоя.

Можно записать, что НС ПР реализует функцию

$$y(u, v; x) = \sum_{j=1}^{q} u_j \psi_j(v, x), \tag{2.24}$$

где $\psi_j(v, x)$ — выход j-го нейрона скрытого слоя при входном векторе сети x. Здесь $u \in \mathbb{R}^q$, $v \in \mathbb{R}^p$, $p + q$ — суммарное количество весов НС. Функционал (1.8) можно переписать в следующем виде:

$$J(u, v) = \sum_{i=1}^{k} \sum_{j=1}^{q} \left(y(u, v; \widetilde{x}_i) - \widetilde{y}_i\right)^2 = \sum_{i=1}^{k} \sum_{j=1}^{q} \left(u_j \psi_j(v, \widetilde{x}_i) - \widetilde{y}_i\right)^2. \tag{2.25}$$

Функция (2.24) может быть представлена в векторно-матричной форме

$$y(u, v; x) = \psi(v, x)^T u, \tag{2.26}$$

где $\psi(v, x) = \begin{bmatrix} \psi_1(v, x) & \psi_2(v, x) & \dots & \psi_q(v, x) \end{bmatrix}^T \in \mathbb{R}^q$. На обучающем множестве рассматриваемая функция $\psi(v, x)$ образует матрицу $\Psi(v) \in \mathbb{R}^{k \times q}$:

$$\Psi = \begin{bmatrix} \psi_1(v, x_1) & \psi_2(v, x_1) & \dots & \psi_q(v, x_1) \\ \psi_1(v, x_2) & \psi_2(v, x_2) & \dots & \psi_q(v, x_2) \\ \dots & \dots & \dots & \dots \\ \psi_1(v, x_p) & \psi_2(v, x_p) & \dots & \psi_q(v, x_p) \end{bmatrix}.$$

Для НС ПР матрица $\Psi(v)$ состоит из выходов нейронов скрытого слоя на обучающем множестве.

С учетом этого задача оптимизации функционала (1.8) (или, что то же самое, (2.25)) может быть записана как

$$J(u,v) = \frac{1}{2}\|\Psi(v)u - \widetilde{y}\|^2 \to \min\,. \tag{2.27}$$

Из БЛНС (2.23) следует, что НЗНК (2.27) эквивалентна задаче

$$\widehat{J}(v) = \|\Psi(v)\Psi(v)^+\widetilde{y} - \widetilde{y}\|^2 \to \min\,. \tag{2.28}$$

Преимущество минимизации функционала в задаче (2.28) состоит в том, что оптимизация производится лишь по нелинейно входящим параметрам v_i, линейные определяются на основе соотношения (2.23). Недостатком является то, что снижение размерности задачи приводит к необходимости оптимизации более сложной функции. Данная задача впервые была поставлена Голубом (Golub) и Перейрой (Pereyra) в 1973 г.

Для вывода метода обучения НС на основе декомпозиции вектора весов с псевдообращением введем обозначение

$$H(v) = \Psi(v)\Psi(v)^+\widetilde{y} - \widetilde{y} = (\Psi(v)\Psi(v)^+ - I_k)\widetilde{y}, \tag{2.29}$$

где I_k — единичная матрица порядка k. Основной задачей является определение матрицы Якоби $H'(v)$ по вектору нелинейно входящих весов v. На основе матрицы $H'(v)$ можно реализовывать алгоритмы обучения НС, базирующиеся на методах решения НЗНК. Соотношение

$$\nabla\widehat{J}(v) = H'(v)^T H(v) \tag{2.30}$$

позволяет использовать другие алгоритмы оптимизации, использующие информацию о градиенте минимизируемой функции $\nabla\widehat{J}(v)$ по вектору v.

Производная матрицы $A(B) \in \mathbb{R}^{m\times n}$ по матрице $B \in \mathbb{R}^{k\times p}$ определяется следующим образом (для упрощения записи аргумент в дальнейшем будем

опускать):

$$A'_B = \frac{\partial A}{\partial B} = \begin{bmatrix} \frac{\partial A}{\partial b_{11}} & \frac{\partial A}{\partial b_{12}} & \cdots & \frac{\partial A}{\partial b_{1p}} \\ \frac{\partial A}{\partial b_{21}} & \frac{\partial A}{\partial b_{22}} & \cdots & \frac{\partial A}{\partial b_{2p}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial A}{\partial b_{k1}} & \frac{\partial A}{\partial b_{k2}} & \cdots & \frac{\partial A}{\partial b_{kp}} \end{bmatrix} \in \mathbb{R}^{mk \times np}.$$

Матрицу Якоби $A'_v = A'$ по вектору $v \in \mathbb{R}^p$ будем рассчитывать как производную матрицы по вектор-строке v^T:

$$A' = \frac{\partial A}{\partial v^T} = \begin{bmatrix} \frac{\partial A}{\partial v_1} & \frac{\partial A}{\partial v_2} & \cdots & \frac{\partial A}{\partial v_p} \end{bmatrix} \in \mathbb{R}^{m \times np}. \tag{2.31}$$

Такой способ вычисления связан с определением матрицы Якоби как матрицы, составленной из транспонированных градиентов.

В общем случае для производной произведения AB двух матриц $A \in \mathbb{R}^{m \times n}$ и $C \in \mathbb{R}^{n \times l}$ по третьей $C \in \mathbb{R}^{k \times p}$, справедлива следующая формула:

$$(AC)'_B = \frac{\partial (AC)}{\partial B} = \frac{\partial A}{\partial B}(I_p \otimes C) + (I_k \otimes A)\frac{\partial C}{\partial B} = A'(I_p \otimes C) + (I_k \otimes A)C',$$

где $\otimes$ — символ тензорного (кронекеровского) произведения матриц. Тензорное произведение определяется как блочная матрица следующего вида:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mk \times np}.$$

Если дифференцирование ведется по вектор-строке $v^T \in \mathbb{R}^{1 \times p}$, то производная произведения матриц вычисляется как

$$(AC)'_v = \frac{\partial (AC)}{\partial v^T} = \frac{\partial A}{\partial v^T}(I_p \otimes C) + \frac{\partial C}{\partial v^T} = A'(I_p \otimes C) + C'. \tag{2.32}$$

Пример 2.2.

Проверим справедливость формулы (2.32). Пусть даны зависящие от $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ векторы $a = \begin{bmatrix} a_1(v) & a_2(v) \end{bmatrix}$ и $c = \begin{bmatrix} c_1(v) \\ c_2(v) \end{bmatrix}$. Прямое вычисление $(ac)'_v = \frac{\partial (ac)}{\partial v^T}$

дает следующий вектор:

$$(ac)'_v = (a_1c_1 + a_2c_2)'_v = \left[(a_1c_1 + a_2c_2)'_{v_1} \qquad (a_1c_1 + a_2c_2)'_{v_2}\right] =$$

$$= \left[a_1{}'_{v_1}c_1 + a_1c_1{}'_{v_1} + a_2{}'_{v_1}c_2 + a_2c_2{}'_{v_1} \qquad a_1{}'_{v_2}c_1 + a_1c_1{}'_{v_2} + a_2{}'_{v_2}c_2 + a_2c_2{}'_{v_2}\right],$$

а применение формулы (2.32) приводит к

$$(ac)'_v = \begin{bmatrix} a_1{}'_{v_1} & a_2{}'_{v_1} & a_1{}'_{v_2} & a_2{}'_{v_2} \end{bmatrix} \begin{bmatrix} c_1 & 0 \\ c_2 & 0 \\ 0 & c_1 \\ 0 & c_2 \end{bmatrix} + \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} c_1{}'_{v_1} & c_1{}'_{v_2} \\ c_2{}'_{v_1} & c_2{}'_{v_2} \end{bmatrix} =$$

$$= \left[a_1{}'_{v_1}c_1 + a_2{}'_{v_1}c_2 \qquad a_1{}'_{v_2}c_1 + a_2{}'_{v_2}c_2\right] + \left[a_1c_1{}'_{v_1} + a_2c_2{}'_{v_1} \qquad a_1c_1{}'_{v_2} + a_2c_2{}'_{v_2}\right] =$$

$$= \left[a_1{}'_{v_1}c_1 + a_2{}'_{v_1}c_2 + a_1c_1{}'_{v_1} + a_2c_2{}'_{v_1} \qquad a_1{}'_{v_2}c_1 + a_2{}'_{v_2}c_2 + a_1c_1{}'_{v_2} + a_2c_2{}'_{v_2}\right].$$

Оба способа дают идентичный результат. ◁

Для разработки метода обучения найдем способ вычисления производной псевдообратной матрицы $(A^+(v))'$, состоящей из элементов-функций. Получение в общем случае аналитической формулы для вычисления требуемой производной невозможно (прямой способ). Имеется возможность вычисления $(A^+(v))'$ косвенным способом. Вывод формулы для вычисления производной псевдообратной матрицы осуществляется в четыре этапа:

1. Нахождение производной обратной матрицы.
2. Определение производной в случае, когда дана прямоугольная матрица полного столбцового ранга.
3. Получение аналогичной формулы для прямоугольной матрицы полного строкового ранга.
4. Получение искомой формулы на основе скелетного разложения матрицы.

Этап 1.

Для квадратной невырожденной матрицы A с учетом (2.32) имеем

$$\left(AA^{-1}\right)' = A'\left(I_p \otimes A^{-1}\right) + A\left(A^{-1}\right)' = I' = 0,$$

откуда следует

$$\left(A^{-1}\right)' = -A^{-1}A'\left(I_p \otimes A^{-1}\right). \tag{2.33}$$

Этап 2.

Пусть B — матрица полного столбцового ранга. В этом случае ее псевдообратная выражается как

$$B^{+} = \left(B^T B\right)^{-1} B^T. \tag{2.34}$$

Применение формулы (2.32) дает

$$\left(B^{+}\right)' = \left[(B^T B)^{-1}\right]'\left(I_p \otimes B^T\right) + \left(B^T B\right)^{-1}\left(B^T\right)'.$$

Матрица $B^T B$ — квадратная невырожденная, применение к ней (2.33), приводит к формуле

$$\begin{aligned}\left(B^{+}\right)' = &-(B^T B)^{-1}(B^T B)'(I_p \otimes (B^T B)^{-1})(I_p \otimes B^T) + \left(B^T B\right)^{-1}\left(B^T\right)' = \\ &= -(B^T B)^{-1}(B^T)'(I_p \otimes B)(I_p \otimes (B^T B)^{-1})(I_p \otimes B^T) - \\ &\quad - (B^T B)^{-1}B^T B'(I_p \otimes (B^T B)^{-1})(I_p \otimes B^T) + (B^T B)^{-1}(B^T)'.\end{aligned}$$

Обычное и тензорное произведение связаны следующим образом:

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD),$$

если матрицы имеют согласованные размеры. Из этой формулы получаем важное для нас следствие

$$(I_p \otimes B)(I_p \otimes D) = (I_p I_p) \otimes (BD) = I_p \otimes (BD), \tag{2.35}$$

и поэтому

$$\begin{aligned}\left(B^{+}\right)' = &-(B^T B)^{-1}(B^T)'(I_p \otimes B(B^T B)^{-1}B^T) - \\ &\quad - (B^T B)^{-1}B^T B'(I_p \otimes (B^T B)^{-1}B^T) + (B^T B)^{-1}(B^T)'.\end{aligned}$$

На основе (2.34) и равенства $(B^T B)^{-1} = (B^T B)^{+} = B^{+}B^{+T}$ получается

$$\left(B^{+}\right)' = -B^{+}B^{+T}(B^T)'(I_p \otimes BB^{+}) - B^{+}B'(I_p \otimes B^{+}) + B^{+}B^{+T}(B^T)'.$$

Группируя первое слагаемое с третьим и считая, что матрица B состоит из k строк, приходим к формуле

$$\left(B^{+}\right)' = B^{+}B^{+T}(B^{T})'(I_{kp} - (I_p \otimes BB^{+})) - B^{+}B'(I_p \otimes B^{+}).$$

Операция тензорного произведения дистрибутивна относительно сложения матриц, а $I_{kp} = I_p \otimes I_k$, поэтому:

$$I_{kp} - (I_p \otimes BB^{+}) = I_p \otimes I_k - (I_p \otimes BB^{+}) = I_p \otimes (I_k - BB^{+}),$$

на основании чего приходим к окончательной формуле для производной псевдообратной матрицы полного столбцового ранга

$$\left(B^{+}\right)' = B^{+}B^{+T}(B^{T})'(I_p \otimes (I_k - BB^{+})) - B^{+}B'(I_p \otimes B^{+}). \qquad (2.36)$$

Этап 3.

Рассмотрим случай с матрицей полного строкового ранга C. Псевдообратная такой матрицы выражается так:

$$C^{+} = C^{T}\left(CC^{T}\right)^{-1}.$$

Вывод формулы для расчета $(C^{+})'$ аналогичен выводу (2.36):

$$\begin{aligned}\left(C^{+}\right)' &= (C^{T})'\left(I_p \otimes (CC^{T})^{-1}\right) + C^{T}\left[(CC^{T})^{-1}\right]' = \\ &= (C^{T})'\left(I_p \otimes (CC^{T})^{-1}\right) - C^{T}(CC^{T})^{-1}(CC^{T})'\left(I_p \otimes (CC^{T})^{-1}\right) = \\ &= (C^{T})'\left(I_p \otimes C^{+T}C^{+}\right) - C^{+}C'\left(I_p \otimes C^{+}\right) - C^{+}C\left(C^{T}\right)'\left(I_p \otimes C^{+T}C^{+}\right),\end{aligned}$$

откуда получается формула

$$\left(C^{+}\right)' = \left(I_q - C^{+}C\right)(C^{T})'\left(I_p \otimes C^{+T}C^{+}\right) - C^{+}C'\left(I_p \otimes C^{+}\right). \qquad (2.37)$$

Здесь q — число столбцов в матрице C, а $(C^{T})' \neq (C')^{T}$. Формулы (2.36) и (2.37) обладают определенной симметрией.

Этап 4.

Рассмотрим матрицу $A \in \mathbb{R}^{k\times q}$ произвольного ранга $r \leqslant \min\{k, q\}$. В этом случае существуют матрица полного столбцового ранга $B \in \mathbb{R}^{k\times r}$ и матрица

полного строкового ранга $C \in \mathbb{R}^{r\times q}$ такие, что $A = BC$. Такое разложение матрицы называется скелетным. При этом справедливо равенство

$$A^+ = C^+B^+. \tag{2.38}$$

Учитывая формулы (2.32), (2.36), (2.37), (2.35), а также некоторые другие свойства псевдообратных матриц, получаем формулу для вычисления производной псевдообратной матрицы произвольного ранга:

$$\begin{aligned}\left(A^+\right)' = (I_q - A^+A)(A^T)'(I_p \otimes A^{+T}A^+)+ \\ + A^+A^{+T}(A^T)'(I_p \otimes (I_k - AA^+)) - A^+A'(I_p \otimes A^+).\end{aligned} \tag{2.39}$$

Здесь $A' \in \mathbb{R}^{k\times qp}$ и $(A^T)' \in \mathbb{R}^{q\times kp}$. Из анализа размерностей матриц видно, что действительно, $(A^+)' \in \mathbb{R}^{q\times kp}$.

Вернемся к нахождению формулы для вычисления $H'(v)$. Для этого необходимо знать производную не самой матрицы Ψ, а производную произведения $\Psi\Psi^+$:

$$\begin{aligned}\left(\Psi\Psi^+\right)' &= \Psi'(I_p \otimes \Psi^+) + \Psi(\Psi^+)' = \\ &= \Psi'(I_p \otimes \Psi^+) + \Psi(I_q - \Psi^+\Psi)(\Psi^T)'(I_p \otimes \Psi^{+T}\Psi^+)+ \\ &\quad + \Psi\Psi^+\Psi^{+T}(\Psi^T)'(I_p \otimes (I_k - \Psi\Psi^+)) - \Psi\Psi^+\Psi'(I_p \otimes \Psi^+).\end{aligned}$$

Принимая во внимание, что

$$\Psi\Psi^+\Psi^{+T} = (\Psi\Psi^+)^T\Psi^{+T} = (\Psi^+\Psi\Psi^+)^T = \Psi^{+T},$$

а слагаемое

$$\Psi(I_q - \Psi^+\Psi) = \Psi - \Psi\Psi^+\Psi = 0,$$

получаем формулу

$$\left(\Psi\Psi^+\right)' = \Psi^{+T}(\Psi^T)'(I_p \otimes (I_k - \Psi\Psi^+)) + (I_k - \Psi\Psi^+)\Psi'(I_p \otimes \Psi^+). \tag{2.40}$$

Размеры матриц согласованы, $(\Psi\Psi^+)' \in \mathbb{R}^{k\times kp}$.

Формула (2.40) позволяет найти производную (точнее, матрицу Якоби) $H(v)$, определенную по (2.29), по вектору нелинейно входящих весов v:

$$H' = (\Psi\Psi^+)'(I_p \otimes \widetilde{y}) =$$
$$= \Psi^{+T}(\Psi^T)'(I_p \otimes (I_k - \Psi\Psi^+)\widetilde{y}) + (I_k - \Psi\Psi^+)\Psi'(I_p \otimes \Psi^+\widetilde{y}) \in \mathbb{R}^{k\times p}. \quad (2.41)$$

Метод обучения НС ПР на основе декомпозиции вектора весов с псевдообращением может быть применен и к многослойным сетям. Функционирование m-слойной НС ПР можно представить в виде

$$y = y(u, v, x) = \sum_{i=1}^{q} u_i y_i^{m-1}(v, x),$$

где $y_i^{m-1}(v, x)$ — выход i-го нейрона последнего скрытого, $(m-1)$-го, слоя. Рассмотренный выше метод обучения одновыходных стандартных НС ПР практически без изменений применяется к многослойным сетям. Изменения касаются только процесса нахождения матрицы Якоби для $\Psi(v)$ (и, соответственно, $\Psi^T(v)$) по вектору весов v. Для этого применяется способ, аналогичный процедуре ОРО и связанный с учетом суперпозиционного характера НС ПР. Требующая вычисления производная $\dfrac{\partial y^{(m-1,l)})}{\partial w_j^{(h,i)}}$, $l = 1, \ldots, N_{m-1}$, $h = 1, \ldots, m-2$, $i = 1, \ldots, N_h$, определяется как

$$\frac{\partial y^{(m-1,l)}}{\partial w_j^{(h,i)}} = \frac{\partial y^{(m-1,l)}}{\partial y^{(h,i)}} \cdot \frac{\partial y^{(h,i)}}{\partial net_j^{(h,i)}} \cdot \frac{\partial net_j^{(h,i)}}{\partial w_j^{(h,i)}},$$

$$s^{(l,h,i)} = \frac{\partial y^{(m-1,i)}}{\partial y^{(h,i)}} = \sum_{d=1}^{N_{h+1}} \frac{\partial y^{(m-1,l)}}{\partial y^{(h+1,d)}} \cdot \frac{\partial y^{(h+1,d)}}{\partial y^{(h,i)}} = \sum_{d=1}^{N_{h+1}} s^{(l,h+1,d)} \frac{\partial y^{(h+1,d)}}{\partial y^{(h,i)}}.$$

Начальное условие получается следующим образом:

$$s^{(l,m-1,i)} = \frac{\partial y^{(m-1,l)}}{\partial y^{(m-1,i)}} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}.$$

2.3.5. Глобальное обучение нейронных сетей

Рассмотренные выше методы оптимизации носят локальный характер, т.е. позволяют оптимизировать функцию не на всей области определения, а

лишь в некоторой окрестности начальной точки. Они позволяют с достаточной степенью точности находить оптимумы унимодальных (например, квадратичных) функций. Их применение в чистом виде неэффективно. Функционал (1.8) является сложной нелинейной функцией, имеющей многоэкстремальный характер. Идеалом является нахождение таких весов, чтобы минимум был глобальным. К сожалению, ни один метод не гарантирует решение задачи в такой постановке.

Почти все способы поиска глобального оптимума в той или иной мере используют случайность. Хотя имеется возможность реализации полностью случайного поиска, такой подход тоже не применяется. В связи с этим, процедуры обучения НС представляют собой некоторую комбинацию, синтез детерминированных и случайных способов поиска оптимума. Наиболее популярным является алгоритм локального обучения с периодической встряской весов. После нахождения локального оптимума производится встряска значений весов НС — внесение небольших случайных изменений (встряски):

$$\widehat{w}_j^{(l,i)} = w_j^{(l,i)} + r_j^{(l,i)},$$

где $r_j^{(l,i)}$ – некоторая случайная величина, равномерно распределенная на отрезке $[a, b]$. Цель этой добавки — «выбить» функцию из локального минимума (рис. 2.7). Обычно выбираются небольшие значения a и b, например, $a = -0.1$, $b = 0.1$. Если a и b выбираются большими, это равносильно тому, что задается новая начальная точка обучения.

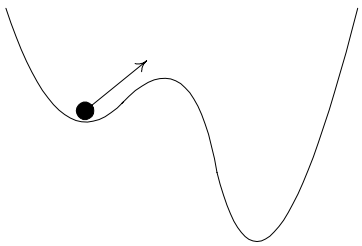

Рис. 2.7. Иллюстрация механизма встряски

Несмотря на наличие случайного компонента, качество и эффективность обучения НС ПР с использованием механизма встряски во многом определяется используемым локальным методом обучения.

3. Архитектуры нейронных сетей для кластеризации и распознавания образов

НС ПР являются наиболее распространенным классом НС. Однако существуют и другие архитектуры НС и методы обучения, широко применяющиеся на практике. К ним относятся, в частности, самоорганизующиеся карты Кохонена и рекуррентные НС.

3.1. Самоорганизующиеся карты Кохонена

Задача кластеризации заключается в разбиении множества объектов, представленных своими векторами $x^i \in \mathbb{R}^n$, $i = 1, \ldots, p$, на группы, называемые *кластерами* или *таксонами*. Объекты группируются таким образом, чтобы в пределах одного кластера они были более похожими друг на друга, чем на объекты других кластеров. В каждом кластере выделяются типичные представители w^i — *ядра*, $i = 1, \ldots, k$, где k — количество кластеров (рис. 3.1).

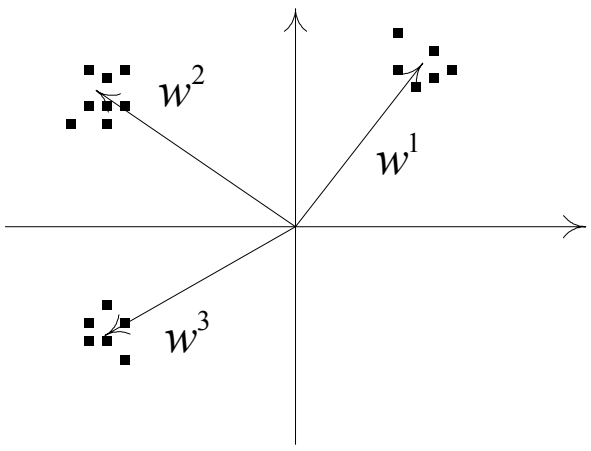

Рис. 3.1. Кластеры и их ядра

Для оценки степени близости (сходства) объектов используется расстояние $d(\cdot, \cdot)$. Чаще всего рассматривается евклидово расстояние или, для

упрощения вычислений, его квадрат

$$d^2(x^i, x^j) = \|x^i - x^j\|^2 = \sum_{l=1}^{n} (x_l^i - x_l^j)^2, \tag{3.1}$$

где x_l^i — l-й элемент вектора x^i. Чем меньше расстояние между векторами, тем ближе друг к другу они располагаются, и следовательно, тем более похожи объекты, описываемые этими векторами.

Будем считать число кластеров k известным. Обозначим ядро кластера, к которому относится вектор x^i, через $w(x^i)$. Ставится следующая задача кластеризации: найти k ядер w^i и разбить объекты на кластеры таким образом, чтобы сумма мер близости между векторами и ядрами соответствующих кластеров была минимальной, т.е.

$$D = \sum_{i=1}^{n} d\left(x^i, w(x^i)\right)^2 \to \min.$$

В пространстве $\mathbb{R}^n$ задано скалярное произведение векторов

$$\langle x^i, x^j \rangle = \sum_{l=1}^{n} x_l^i x_l^j.$$

Рассмотрим квадрат нормы разности произвольного вектора x^i и ядра некоторого кластера w^j, используя скалярное произведение:

$$\|x^i - w^j\|^2 = \langle x^i - w^j, x^i - w^j \rangle = \langle x^i, x^i \rangle - 2\langle x^i, w^j \rangle + \langle w^j, w^j \rangle = \\ = \|x^i\|^2 - 2\langle x^i, w^j \rangle + \|w^j\|^2. \tag{3.2}$$

Если ядра кластеров являются нормированными, т.е. $\|w^j\| = 1$, $j = 1, \ldots, k$, то формулу (3.2) можно записать в виде:

$$\|x^i - w^j\|^2 = \|x^i\|^2 - 2\langle x^i, w^j \rangle + 1. \tag{3.3}$$

Исходя из (3.3), видно, что ближайшим к вектору x^i будет то ядро w^j, для которого скалярное произведение $\langle x^i, w^j \rangle$ является максимальным. Заметим, что величина $\langle x^i, w^j \rangle$ есть проекция вектора x^i на направление w^j.

Рассмотрим НС, состоящую из входного слоя нейронов и еще одного (*слоя Кохонена*), у нейронов которого функция активации является единичной. Входной слой состоит из n нейронов, а слой Кохонена – из k. Вектор весов w^j j-го нейрона слоя Кохонена представляет собой нормированное ядро j-го кластера.

Для определения ядра кластера, ближайшего к вектору x^i, этот вектоп подается на вход НС. Для каждого j-го нейрона слоя Кохонена вычисляется скалярное произведение $\langle x^i, w^j \rangle$ и находится среди них нейрон r, дающий максимальное значение скалярного произведения:

$$r = \arg \max_{j=1}^{k} \langle x^i, w^j \rangle. \tag{3.4}$$

Номер нейрона с максимальным скалярным произведением соответствует номеру кластера, к которому относится выбранный объект. НС такой структуры, функционирующая по принципу выбора максимального скалярного произведения, называется *самоорганизующимися картами (СОК) Кохонена* (рис. 3.2).

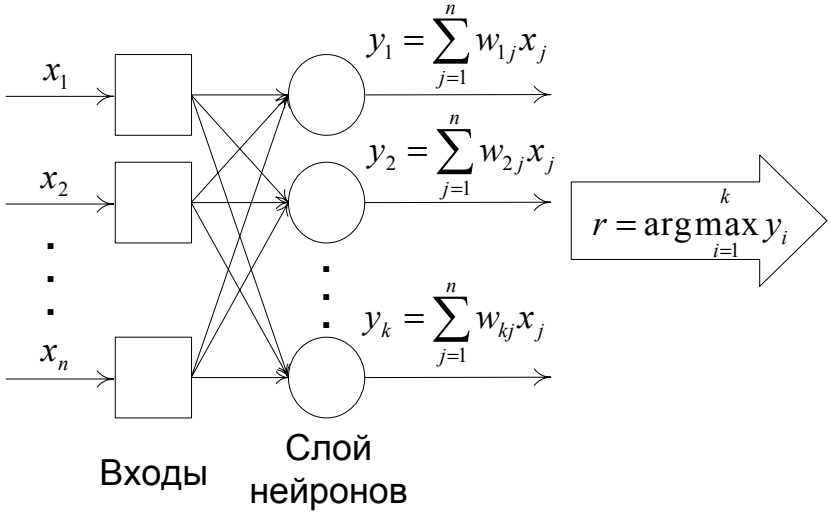

Рис. 3.2. Самоорганизующиеся карты Кохонена

Как и для НС ПР, перед использованием СОК Кохонена необходимо обучить. В отличие от НС ПР обучение СОК относится к задачам *обучения без учителя*, т.е. к ситуациям, когда реакция сети на входной вектор заранее неизвестна (неизвестны кластеры, к которым относятся объекты). Классический алгоритм обучения СОК Кохонена без учителя называется «Победитель

забирает все» («Winner takes all») (алгоритм 2). Данный алгоритм получил свое название из-за того, что на каждом шаге цикла происходит обновление только значений весов нейрона, имеющего максимальное скалярное произведение.

Алгоритм 2. Обучение СОК Кохонена «Победитель забирает все»

1. Инициализация. Весовым коэффициентам нейронов слоя Кохонена присваиваются небольшие случайные числа: $w_j^i \in [-1; 1]$, $i = 1, \ldots, k$, $j = 1, \ldots, n$.

2. Нормализация векторов весов нейронов:

$$w^i := \frac{w^i}{\|w^i\|}, i = 1, \ldots, k.$$

3. Задание константы обучения $\eta \in [0.1, 0.7]$. Выбор количества циклов обучения N_c.

4. N_c циклов по всем векторам x^i, $i = 1, \ldots, p$.

4.1. Определение нейрона-победителя по формуле (3.4).

4.2. Корректировка весов нейрона:

$$w_r := w_r + \eta(x - w_r).$$

4.3. Нормализация вектора весов нейрона-победителя:

$$w^r := \frac{w^r}{\|w^r\|}.$$

◁

На шаге 4.2 алгоритма 2 по сути находится выпуклая комбинация векторов w^r и x^i, т.к.

$$w_r := w_r + \eta(x - w_r) = \eta x + (1 - \eta)w_r.$$

В результате обновления весов ядро кластера w^r несколько приближается к вектору x^i (рис. 3.3). Чем больше величина константы обучения η, тем сильнее изменяется вектор w^r.

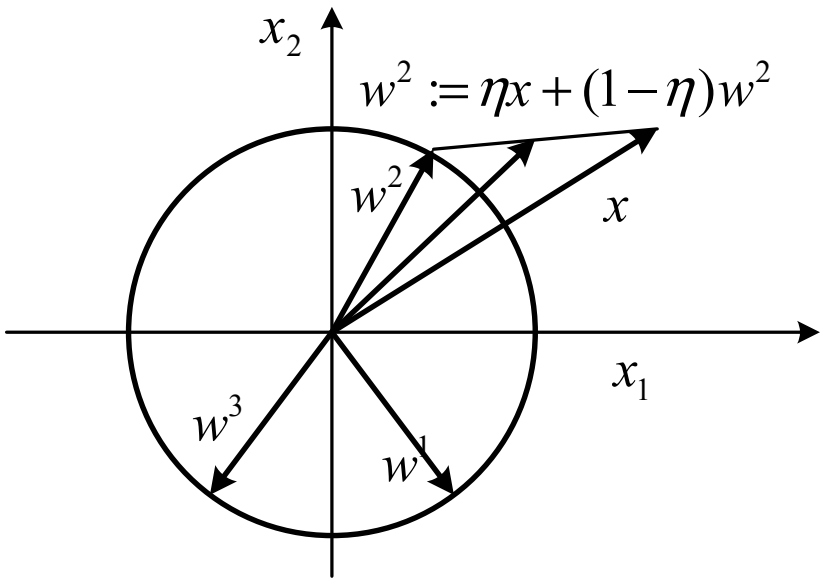

Рис. 3.3. Обновление весов нейрона-победителя

Алгоритм «Победитель забирает все» является не очень эффективным методом обучения. Это связано с тем, что вследствие инициализации весов большинство векторов x^i может попадать в один и тот же кластер, т.е. некоторые нейроны могут оказаться незадействованными в процессе обучения. Это так называемая проблема *мертвых нейронов*.

Приведем модификации, способные улучшить процесс обучения СОК Кохонена:

— Уменьшение с течением времени (каждой цикл) константы обучения, т.е. $\eta = \eta(t) \to 0$ при $t \to \infty$.

— Обновление весов нейронов пропорционально величине скалярного произведения. В этом случае на каждом шаге цикла участвует не только нейрон-победитель.

— Использование «чувства справедливости»: для часто выигрывающего нейрона уменьшается вероятность обновления весов. Частный случай — «механизм утомления», когда каждому нейрону приписывается потенциал p_i. В определении победителя используются только те нейроны, у которых $p_i \geqslant p_{min}$, где p_{min} — некоторый пороговый потенциал. Если обновляются веса r-го нейрона, то для него полагают

$$p_r := p_r - p_{min}.$$

В случае неучастия нейрона в обновлении весов потенциал увеличивается:

$$p_i := \max\left\{1, p_i + \frac{1}{n}\right\}.$$

Если $p_{min} = 0$, то получается классический алгоритм «Победитель забирает все». Хорошие результаты получаются при $p_{min} \approx 0,75$.

3.2. Нейронные сети с обратными связями

Принципиально иной архитектурой НС по сравнению с рассмотренными выше являются *НС с обратными связями* или же *рекуррентными НС (РНС)*. В отличие от НС ПР, которые можно назвать статическими, РНС реализуют динамический отклик сети. Суть этого состоит в том, что сигналы некоторых нейронов подаются на входы нейронов, располагающихся в этом же или предыдущих слоях. Традиционной сферой применения РНС являются задачи распознавания образов.

Рассмотрим один из типов РНС, называемых *НС Хопфилда*. Такие сети состоят из одного слоя нейронов (не считая входного), причем выходы всех нейронов единственного слоя подаются на свои входы (рис. 3.4). В НС Хопфилда считается, что элементы входного вектора могут принимать значение -1 и 1. Такая сеть функционирует потактово (алгоритм 3).

Алгоритм 3. Функционирование НС Хопфилда

1. На входы НС подается вектор весов $x \in \mathbb{R}^n$, состояние сети считается равным $x[0] = x$.

2. Вычисляются выходы нейронов НС Хопфилда (новое состояние) по формуле

$$x_i[t+1] = \sigma\left(\sum_{j=1}^{n} w_{ij} x_j[t]\right),$$

где w_{ij} — вес, идущий от j-го нейрона к i-му, $i, j = 1, \ldots, n$; σ — пороговая функция активации-2.

3. Проверяется критерий остановки: если

$$x[t+1] = x[t],$$

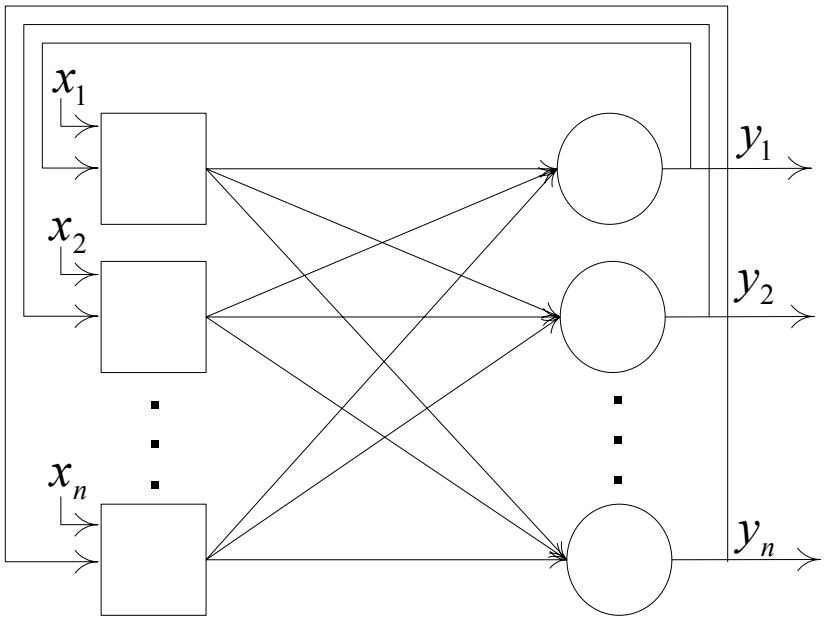

Рис. 3.4. Нейронная сеть Хопфилда

т.е. НС находится в устойчивом состоянии, то выходные сигналы $y = x[t+1]$, иначе $t := t + 1$ и происходит переход на шаг 2. $\triangleleft$

Для успешного функционирования НС Хопфилда необходима соответствующая настройка весов, позволяющих в процессе функционирования приходить к устойчивым точкам – запомненным образам. В отличие от НС ПР и СОК Кохонена в НС Хопфилда веса определяются безытерационным способом.

Пусть задано множество векторов $\{x^l\}$, $l = 1, \ldots, k$, подлежащих запоминанию — *эталонов* (*образцов*). В качестве меры близости векторов выберем евклидово расстояние между ними. Хопфилд показал, что исходные векторы будут точками минимума, если веса сети задать по формуле

$$w_{ij} = \begin{cases} \sum\limits_{l=1}^{k} x_i^l x_j^l, & i \neq j, \\ 0, & i = j. \end{cases} \tag{3.5}$$

Формула (3.5) приводит к формированию симметричной матрицы весов с нулевой главной диагональю. Это является достаточным, но не необходимым условием устойчивости сети в исходных точках x^l.

Обозначим через $W \in \mathbb{R}^{n \times n}$ матрицу весов w_{ij}. Формулу (3.5) можно

представить в векторно-матричной форме

$$W = \sum_{l=1}^{k} x^l \cdot (x^l)^T$$

с последующим обнулением весов на главной диагонали.

При подаче какого-либо вектора на вход в результате работы сети будет получаться один из запомненных образов. Заметим, что в ходе функционирования сети на выходе могут получаться образы, не совпадающие ни с одним из эталонов — химеры.

При проектировании НС Хопфилда следует учитывать два ограничения:

1. Число образцов не должно превосходит 14% от количества входов, т.е. емкость сети составляет $0,14n$ эталонов.
2. Образцы должны быть слабо скоррелированы, т.е. как можно больше отличаться друг от друга. Для этого достаточно выполнения условия

$$\sum_{l=1, l\neq p}^{k} |\langle x^l, x^p \rangle| < n$$

для каждого эталона $p = 1, \ldots, k$.

Сети Хопфилда реализуют так называемую ассоциативную память — выдает ассоциацию на входной вектор x. НС Хопфилда восстанавливает по набору признаков сам объект. Для представления образов в виде двоичных векторов следует разбить исходный двумерный образ на клетки и затем закодировать каждую клетку 1 или -1.

Заключение

Более детальную информацию о нейросетевых архитектурах и применении НС можно получить из работ [8, 11, 14–16, 18, 19]. Алгоритм ОРО подробно рассмотрен в работе [1]. Некоторые методы предварительной обработки данных рассмотрены в [10]. Про синтез НС и нечеткой логики можно рекомендовать литературу [12, 17]. Методам оптимизации посвящена значительная литература [3, 7, 9, 13]. Много информации о псевдообращении и его применении в задачах оптимизации можно почерпнуть из работ [2, 4–6, 20].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Аведьян, Э.Д. Алгоритмы настройки многослойных нейронных сетей / Э.Д. Аведьян // Автоматика и телемеханика.— 1995.— N4.— С. 106-118.
2. Алберт, А. Регрессия, псевдоинверсия и рекуррентное оценивание / А. Алберт.— М.: Наука, 1977.— 224 с.
3. Аттетков, А.В. Методы оптимизации (Сер. Математика в техническом университете; вып.XIV) / А.В. Аттетков, С.В. Галкин, В.С. Зарубин.— М.: Изд-во МГТУ им. Н.Э.Баумана, 2001.— 440 с.
4. Блюмин, С.Л. Псевдообращение: учебное пособие / С.Л. Блюмин, С.П. Миловидов.— Воронеж: ВорПИ-ЛипПИ, 1990.— 72 с.
5. Блюмин, С.Л. Взвешенное псевдообращение: учебное пособие / С.Л. Блюмин, С.П. Миловидов.— Воронеж: ВорПИ-ЛипПИ, 1991.— 64 с.
6. Блюмин, С.Л. Нелинейный метод наименьших квадратов и псевдообращение: учебное пособие / С.Л. Блюмин, С.П. Миловидов.— Липецк: ЛипПИ, 1992.— 80 с.
7. Гилл, Ф. Практическая оптимизация / Ф. Гилл, У. Мюррей, М. Райт.— М.: Мир, 1985.— 509 с.
8. Горбань, А.Н. Нейронные сети на персональном компьютере / А.Н. Горбань, Д.А. Россиев.— Новосибирск: Наука. Сиб. издат. фирма РАН, 1996.— 276 с.
9. Дэннис, Дж. Численные методы безусловной оптимизации и решения нелинейных уравнений / Дж. Дэннис, Р. Шнабель.— М.: Мир, 1988.— 440 с.
10. Загоруйко, Н.Г. Прикладные методы анализа данных и знаний / Н.Г. Загоруйко.— Новосибирск: Ин-т математики, 1999.— 270 с.
11. Круглов, В.В. Нечеткая логика и искусственные нейронные сети / В.В. Круглов, М.И. Дли, Р.Ю. Голунов.— М.: Физматлит, 2001.— 224 с.
12. Круглов, В.В. Искусственные нейронные сети. Теория и практика / В.В. Круглов, В.В. Борисов.— М.: Горячая линия – Телеком, 2001.— 382 с.

13. Мину, М. Математическое программирование / М. Мину.— М.: Наука, 1990.— 488 с.

14. Миркес, Е.М. Нейрокомпьютер: проект стандарта / Е.М. Миркес.— Новосибирск: Наука, 1999.— 337 с.

15. Нейроинформатика / А.Н. Горбань, В.Л. Дунин-Барковский, А.Н. Кирдин [и др.].— Новосибирск: Наука. Сибирское предприятие РАН, 1998.— 296 с.

16. Нейросетевые системы управления / В.А. Терехов, Д.В. Ефимов, И.Ю. Тюкин [и др.].— СПб: С.-Петербургский университет, 1999.— 264 с.

17. Нечеткая логика: алгебраические основы и приложения / С.Л. Блюмин, И.А. Шуйкова, П.В. Сараев, И.В. Черпаков.— Липецк: ЛЭГИ, 2002.— 112 с.

18. Осовский, С. Нейронные сети для обработки информации / С. Осовский.— Москва: Финансы и статистика, 2002.— 344 с.

19. Уоссермен, Ф. Нейрокомпьютерная техника / Ф. Уоссермен.— М.: Мир, 1992.— 184 с.

20. Golub, G.H. The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate / G.H. Golub, V. Pereyra // SIAM J. Num. Anal., 1973.— V. 10.— P. 413-432.

Приложение

Сведения о псевдообратных матрицах

Понятие *псевдообратной матрицы* является обобщением понятия обратной матрицы на случай прямоугольных матриц. Пусть $A \in \mathbb{R}^{k\times n}$ — исходная матрица, тогда $A^+ \in \mathbb{R}^{n\times k}$ называется псевдообратной (или матрицей Мура-Пенроуза) к A, если выполняются четыре условия (условия Мура-Пенроуза):

1. $AA^+A = A$;
2. $A^+AA^+ = A^+$;
3. $(AA^+)^T = AA^+$;
4. $(A^+A)^T = A^+A$.

Заметим, что если матрица удовлетворяет только первым двум условиям, то это определяет более широкое понятие *обобщенно обратной* или $G-$матрицы.

Имеют место следующие важные свойства:

1. Псевдообратная матрица существует для любой матрицы A.
2. Псевдообратная матрица единственна.
3. Для квадратной невырожденной матрицы псевдообратная матрица совпадает с обратной: $A^+ = A^{-1}$.

Для нахождения псевдообратной матрицы могут быть использованы, например, такие способы:

1. Если A — матрица полного столбцового ранга, т.е. все столбцы являются линейно независимыми, то $A^+ = \left(A^TA\right)^{-1}A^T$.
2. Если A — матрица полного строкового ранга, то $A^+ = \left(AA^T\right)^{-1}A$.

К универсальным способам нахождения псевдообратной матрицы относятся рекуррентные алгоритмы Гревиля и Фадеева. В данной работе приведем алгоритм Гревиля для псевдообращения матриц.

Пусть дана матрица $A \in \mathbb{R}^{m\times n}$ и a_k — ее k-й столбец, $k = 1, \ldots, n$. Пусть A_k — матрица, составленная из k первых столбцов матрицы A:

$$A_k = \begin{bmatrix} a_1 & a_2 & \ldots & a_k \end{bmatrix}.$$

При $k = 1$: $A_1 = a_1$, а при $k = 2, \ldots, n$: $A_k = \begin{bmatrix} A_{k-1} & a_k \end{bmatrix}$; $A_n = A$. Матрица $A^+ \in \mathbb{R}^{n \times m}$ может быть вычислена с помощью рекуррентного алгоритма 4.

Алгоритм 4. Псевдообращение матрицы по Гревилю

1. Инициализация.

$$A_1^+ = \begin{cases} 0, & \text{если } a_1 = 0, \\ \dfrac{a_1^T}{\|a_1\|^2}, & \text{иначе,} \end{cases}$$

   где $\|\cdot\|$ — евклидова норма вектора.

2. Цикл по $k = 2, \ldots, n$.

$$A_k^+ = \begin{bmatrix} A_{k-1}^+(I - a_k f_k) \\ f_k \end{bmatrix},$$

   где I — единичная матрица порядка m,

$$f_k = \begin{cases} \dfrac{c_k^T}{\|c_k\|^2}, \quad c_k = (I - A_{k-1}A_{k-1}^+)a_k, & c_k \neq 0, \\ \dfrac{a_k^T (A_{k-1}^+)^T A_{k-1}^+}{1 + \|A_{k-1}^+ a_k\|^2}, & c_k = 0. \end{cases}$$

Полученная на последнем шаге матрица A_n^+ и есть искомая матрица A^+. $\triangleleft$

Сараев Павел Викторович

Нейросетевые методы искусственного интеллекта

Учебное пособие

Редактор Е.А. Федюшина

Подписано в печать 21.12.07. Формат 60x84 1/16. Бумага офсетная.

Ризография. Печ.л. 4,3. Тираж 100 экз. Заказ N 1.

Липецкий государственный технический университет.

398600 Липецк, ул. Московская, 30.

Типография ЛГТУ. 398600 Липецк, ул. Московская, 30.