

## Лекция 6

### Стандартные архитектуры нейронных сетей

#### Сеть из одного нейрона

Рассмотрим возможности сети состоящей из одного нейрона. Пусть, для начала, этот формальный нейрон не имеет нелинейного преобразователя, т. е. функционирует по формуле  $y = (\bar{w}, \bar{x}) + w_0$ . Какие задачи можно решать с его помощью?

#### Линейная регрессия

Первая, и весьма важная задача, решаемая нейросетью из одного нейрона – это задача линейной регрессии, которая формулируется так: найти наилучшее линейное приближение функции  $F$ , заданной обучающей выборкой  $(\bar{x}^a, y^a)$ ,  $F(\bar{x}^a) = y^a$ ,  $a = 1, \dots, m$ . Для этого требуется найти линейную функцию  $j(\bar{x}) = (\bar{w}, \bar{x}) + w_0$ , ближайшую к  $F$ .

Функцию ошибки можно определить как сумму квадратов разностей:

$$D(j) = \sum_{i=1}^m (F(x^i) - j(x^i))^2 = \sum_{i=1}^m (F(x^i) - (\bar{w}, \bar{x}^i) - w_0)^2 \equiv \sum_{i=1}^m \Delta_i^2. \quad (6.1)$$

**Решение.** Эту функцию необходимо минимизировать, для чего найдём производные по изменяемым параметрам:

$$\frac{\partial D}{\partial w_j} = -2 \sum_{i=1}^m \Delta_i x_j^i, \quad (j = 1, \dots, n); \quad \frac{\partial D}{\partial w_0} = -2 \sum_{i=1}^m \Delta_i. \quad (6.2)$$

Приравнявая эти производные нулю и вводя в рассмотрение  $(n+1)$ -мерные вектора  $x^i$ , в которых все элементы те же, а  $x_0^i \equiv 1$ , мы можем записать

$$\begin{aligned} \Delta_i &= y^i - (\bar{w}, x^i), \\ w_0 &= \frac{1}{m} \left( \sum_{i=1}^m (y^i - (\bar{w}, x^i)) \right). \end{aligned} \quad (6.3)$$

Введем обозначения  $\bar{y}_m = \frac{1}{m} \sum_{i=1}^m y^i$ ;  $\bar{x}_m = \frac{1}{m} \sum_{i=1}^m \bar{x}^i$ . В этих обозначениях (6.3) примет

вид

$$w_0 = \bar{y}_m - (\bar{w}, \bar{x}_m). \quad (6.4)$$

Подставив это значение в (6.2) и приравнявая к нулю, мы получим систему:

$$\sum_{k=1}^n w_k \cdot \left( \sum_{i=1}^m (x_j^i - \bar{x}_j^m)(x_k^i - \bar{x}_k^m) \right) = \sum_{i=1}^m (x_j^i - \bar{x}_j^m)(y^i - \bar{y}_m), \quad (6.5)$$

или, в более короткой форме

$$\sum_{k=1}^n w_k r_{kj} = h_j. \quad (6.6)$$

В векторно-матричной форме полученное равенство имеет вид

$$\bar{w}R = \bar{h}. \quad (6.6')$$

Решение этой системы в случае невырожденности матрицы  $R$  может быть найдено как

$$\bar{w} = \bar{h} \cdot R^{-1}, \quad (6.7)$$

в противном случае решения либо не существует, либо оно не единственно. При этом обычно ищется решение минимальной длины:

$$\bar{w} = \bar{h} \cdot R^+, \quad (6.8)$$

где  $R^+$  - псевдообратная матрица. Способы решения систем линейных уравнений хорошо известны, и могут быть применены и в данном случае.

Пусть теперь нейрон кроме возможности скалярного умножения входного вектора на вектор весов (которое осуществляется адаптивным сумматором) имеет также возможность порогового преобразования, т. е. его функционирование описывается формулой

$$y = F((\bar{w}, \bar{x}) + w_0) = \begin{cases} 1, & (\bar{w}, \bar{x}) + w_0 \geq 0, \\ 0, & (\bar{w}, \bar{x}) + w_0 < 0. \end{cases} \quad (6.9)$$

Рассмотрим его возможности.

### ***Задача линейного разделения двух классов***

Эта задача ставится как задача построения решающего правила, в соответствии с которым, часть векторов из выборки  $\{\bar{x}^a\}_{a=1, \dots, p}$  будет отнесена к одному классу, а остальные вектора выборки – к другому. В терминах нейросети из одного нейрона, это означает необходимость подобрать такие весовые коэффициенты  $w_j$  для нашего нейрона, что выход нейрона (6.9) равен единице, если входной вектор принадлежит первому классу и нулю, если входной вектор принадлежит другому классу.

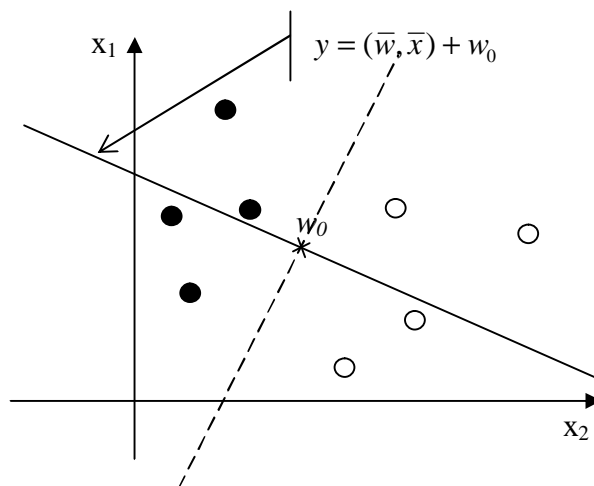
**Решение.** Поиск такого решающего правила можно рассматривать как разделение точек посредством проекции их на прямую. Вектор  $\bar{w}$  задает прямую, на которую ортогонально проектируются все точки, а число  $w_0$  - точку на этой прямой, отделяющую первый класс от второго.

Простейший и подчас очень удобный выбор состоит в проектировании на прямую, соединяющую центры масс выборок. Центр масс вычисляется в предположении, что массы всех точек одинаковы и равны 1. Это соответствует заданию  $\bar{w}$  в виде

$$\bar{w} = \frac{1}{m} \sum_{i=1}^m \bar{x}^{a_i} + \frac{1}{n} \sum_{i=m+1}^p \bar{x}^{a_j}, \quad (6.10)$$

где  $a_1, a_2, \dots, a_m$  - номера векторов принадлежащих первому классу,  $a_{m+1}, a_{m+2}, \dots, a_p$  - номера векторов принадлежащих второму классу. Простейший вариант выбора значения  $w_0$  - посередине между центрами масс групп векторов:

$$w_0 = \frac{1}{2} \left( \left( \frac{1}{m} \sum_{i=1}^m \bar{x}^{a_i}, \bar{w} \right) + \left( \frac{1}{n} \sum_{i=m+1}^p \bar{x}^{a_j}, \bar{w} \right) \right). \quad (6.11)$$



**Рисунок 6.1. Разделение векторов на классы при помощи прямой**

Более чувствительные методы выбора границы раздела классов  $w_0$  учитывают различные вероятности появления объектов разных классов, и оценки плотности распределения точек классов на прямой.

Рассмотрим теперь более сложные структуры - сети, состоящие из нескольких нейронов.

### **Однослойный перцептрон Розенблатта**

Самым простым и исторически первым вариантом многослойного перцептрона является перцептрон Розенблатта – нейросеть с единственным слоем нейронов (F.Rosenblatt, 1957). Перцептрон рассматривался его автором не как конкретное техническое вычислительное устройство, а как модель работы мозга.

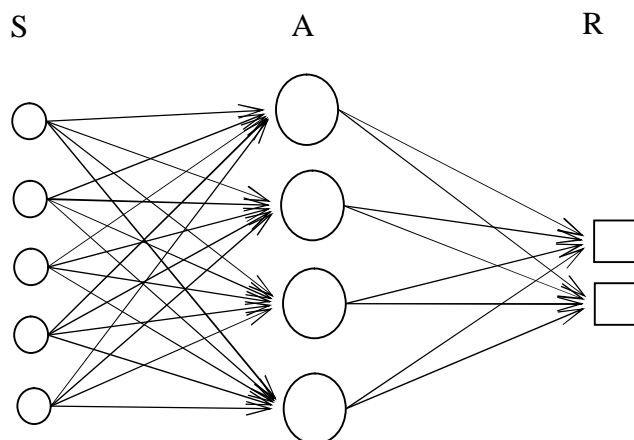


Рисунок 6.2.

Простейший классический персептрон содержит нейроподобные элементы трех типов. S-элементы формируют входной вектор двоичных сигналов, поступающих из внешнего мира. Далее сигналы поступают в слой ассоциативных или A-элементов. Только ассоциативные элементы, представляющие собой формальные нейроны, выполняют нелинейную обработку информации и имеют изменяемые веса связей. Нелинейная часть A-элемента реализуется посредством пороговой функции:

$$f(x) = \begin{cases} 1, & \text{если } x \geq q, \\ 0, & \text{если } x < q. \end{cases} \quad (6.12)$$

R-элементы с фиксированными весами формируют сигнал реакции персептрона на входной стимул.

Розенблатт называл такую нейронную сеть трехслойной, однако по современной терминологии, используемой в этой книге, представленная сеть обычно называется однослойной, так как имеет только один слой нейропроцессорных элементов. Однослойный персептрон характеризуется матрицей синаптических связей  $W$  от S- к A-элементам. Элемент матрицы  $W_{ij}$  отвечает связи, ведущей от  $i$ -го S-элемента к  $j$ -му A-элементу.

### Алгоритм обучения персептрона

Обучение сети состоит в подстройке весовых коэффициентов каждого нейрона. Пусть имеется набор пар векторов  $(\bar{x}^a, \bar{y}^a)$ ,  $a = 1, \dots, p$ ,  $\bar{x}^a, \bar{y}^a \in R^N$ , называемый обучающей выборкой. Будем называть нейронную сеть обученной на данной обучающей выборке, если при подаче на входы сети каждого вектора  $\bar{x}^a$  на выходах всякий раз получается соответствующий вектор  $\bar{y}^a$ .

1. **Функционирование.** Допустим, что вектор  $\bar{x}^a$  является образом распознаваемой демонстрационной карты, а значит, подаётся на каждый входной А-элемент. Например, для  $j$ -го элемента - компоненты  $\bar{x}_i^a$  – умножаются на соответствующие компоненты вектора весов  $w_{ij}$ . Эти произведения суммируются. Если сумма превышает порог  $\theta$ , то выход  $j$ -го нейрона равен единице, в противном случае он – ноль. Эта операция компактно записывается в векторной форме как

$$\bar{y}^* = F(\bar{x}^a \cdot W), \quad (6.13)$$

$$F(\bar{x}) = (f(x_1), f(x_2), \dots, f(x_p)). \quad (6.14)$$

2. **Обучение.** Для обучения сети образ  $\bar{x}^a$  подается на вход и вычисляется выход  $\bar{y}^*$  (в соответствии с описанием в п. 1). Если  $\bar{y}^*$  правильный, то ничего не меняется. Однако, если выход неправилен, то веса, присоединенные к входам, усиливающим ошибочный результат, модифицируются, чтобы уменьшить ошибку: если выход неправильный и равен нулю, то добавить все входы к соответствующим им весам; или если выход неправильный и равен единице, то вычесть каждый вход из соответствующего ему веса:

$$w_{ij}(t+1) = w_{ij}(t) + (y_j^{a(t)} - y_j^*(t)) \cdot x_i^{a(t)}. \quad (6.15)$$

3. Если ошибка существенна, перейти на шаг 1.

В работах Розенблатта был сделано заключение о том, что нейронная сеть рассмотренной архитектуры будет способна к воспроизведению любой логической функции, однако, как было показано позднее М.Минским и С.Пейпертом (М.Мinsky, S.Papert, 1969), этот вывод оказался неточным. Были выявлены принципиальные неустранимые ограничения однослойных персептронов, и в последствии стал в основном рассматриваться многослойный вариант персептрона, в котором имеются несколько слоев процессорных элементов.

Сегодня однослойный персептрон представляет только исторический интерес, однако на его примере могут быть изучены основные понятия и простые алгоритмы обучения нейронных сетей.

## Слоистые архитектуры

Особый интерес в нейроинформатике представляют слоистые однородные нейросети, состоящие из однотипных формальных нейронов, каждый из которых может

быть соединён с любыми нейронами следующего слоя не более чем одной связью. Нейросеть с такими свойствами называется *многослойным персептроном*.

### **Многослойная нейронная сеть и алгоритм обратного распространения ошибки**

История многослойных нейронных сетей началась в 1960-х годах и связана с работами Розенблатта, Минского, Пейперта и др. Лишь в середине 1980-х несколькими исследователями независимо друг от друга был предложен эффективный алгоритм обучения многослойных персептронов, основанный на вычислении градиента функции ошибки. Алгоритм был назван "обратным распространением ошибки".

Алгоритм обратного распространения - это итеративный градиентный алгоритм, который используется с целью минимизации среднеквадратичного отклонения текущего выхода многослойного персептрона и желаемого выхода.

В сетях, обучаемых по методу обратного распространения ошибки могут быть целые или действительные входные сигналы. Выходные сигналы сети - это действительные числа из интервала, заданного передаточной функцией нейронов.

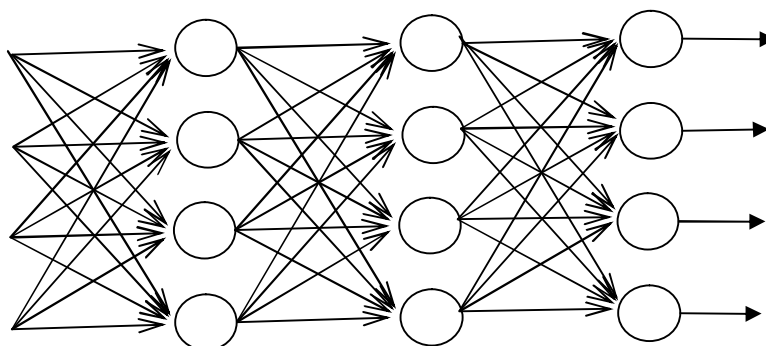


Рисунок 6.3. Многослойная сеть с последовательными связями

В нейронных сетях могут применяться несколько вариантов передаточных функций.

Дискретные:

$$f(s) = \text{sign}(s) = \begin{cases} 1, & s > 0, \\ -1, & s \leq 0. \end{cases}, \text{ или } f(s) = \begin{cases} -1, & x < -1, \\ x, & -1 \leq x \leq 1, \\ 1, & x > 1. \end{cases}$$

Функция Ферми (экспоненциальная сигмоида):

$$f(s) = \frac{1}{1 + e^{-as}}, \quad f'(s) = a f(s)(1 - f(s)),$$

где  $s$  - выход сумматора нейрона,  $a$  - некоторый параметр.

Рациональная сигмоида:

$$f(s) = \frac{s}{|s| + a}, \quad f'(s) = (1 - |f(s)|)^2.$$

Гиперболический тангенс:

$$f(s) = th \frac{s}{a}, \quad f'(s) = \frac{1}{a} (1 - f^2(s)).$$

Гауссиан:

$$f(s) = e^{-as^2}, \quad f'(s) = -2ase^{-as^2}.$$

Сигмоидальные функции являются монотонно возрастающими и имеют отличные от нуля производные на всей области определения. Эти характеристики обеспечивают правильное функционирование и корректное обучение сети.

По скорости вычислений наиболее эффективной функцией активации является рациональная сигмоида. Для вычисления гиперболического тангенса требуется больше всего тактов работы процессора.

**Задание.** Нарисовать графики перечисленных выше функций.

Рассмотрим многослойный персептрон со стандартными элементами.

### **Функционирование многослойного персептрона**

Функционирование в такой нейросети описывается следующими формулами:

$$\begin{aligned} y_i^0 &= x_i^0, \\ s_i^k &= \sum_{j=1}^{P_{k-1}} y_j^{k-1} w_{ij}^k, \\ y_i^k &= f(s_i^k), \end{aligned} \quad (6.16)$$

где

$x_i^0$  - значение подаваемое в момент  $t=0$  на  $i$ -й синапс,

$k$  - номер слоя ( $k = 1, \dots, q$ );

$P_k$  - число нейронов в  $k$ -м слое;

$y_i^k$  - выход  $i$ -го нейрона в  $k$ -м слое;

$w_{ij}^k$  - вес синапса связи  $i$ -го нейрона в  $(k-1)$ -м слое с  $j$ -м нейроном в  $k$ -м слое;

$s_i^k$  - выход сумматора  $i$ -го нейрона в  $k$ -м слое;

$f(s)$  - функция активации нейрона, одинаковая для всех нейронов сети.

Предполагается, что выход  $i$ -го нейрона в  $(k-1)$ -м подается на  $i$ -й вход  $j$ -го нейрона в  $k$ -м слое.

Очевидно, что формулы (6.16) можно записать и в матрично-векторном виде:

$$\begin{aligned}\bar{y}^0 &= \bar{x}^0, \\ \bar{s}^k &= W^k \bar{y}^{k-1}, \\ \bar{y}^k &= F(\bar{s}^k),\end{aligned}\tag{6.16'}$$

где

$\bar{x}^0$  - вектор, подаваемый на вход нейросети,

$W^k = (w_{ij}^k)$  - матрица размерности  $(P_k \times P_{k-1})$ , элементами которой являются веса связей нейронов  $(k-1)$ -го слоя с нейронами  $k$ -го слоя,

$\bar{y}^k$  - вектор полученный на выходе нейронов  $k$ -го слоя,

$F(\bar{s}^k) := (f(s_1^k), f(s_2^k), \mathbf{K}, f(s_{p_k}^k))$  - функция нелинейного преобразования.

Функционирование сети происходит следующим образом:

1-й шаг: на вход сети подается вектор сигналов  $X = (x_1, x_2, \mathbf{K}, x_p)$ . Происходит срабатывание нейронов первого слоя.

$k$ -й шаг: выход каждого нейрона  $k$ -го слоя определяемый формулами (6.16) передается по синаптическим связям нейронам  $(k+1)$ -го слоя.

$q$ -й шаг: на выходе нейронной сети снимается вектор сигналов  $Y = (y_1, y_2, \mathbf{K}, y_{p_q})$ , который и является результатом.

### **Алгоритм обучения многослойного перцептрона**

В многослойных сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и многослойный перцептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС. Одним из вариантов решения этой проблемы является так называемое «распространение сигналов ошибки» от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название *процедуры обратного распространения*. Именно он будет рассмотрен далее.



Пусть имеется обучающая выборка  $(\bar{x}(i), \bar{d}(i))$ ,  $i = 1, \dots, N$ . Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки НС является величина:

$$D(\bar{W}) = \frac{1}{2} \sum_{j,p} (y_j^q(p) - d_j(p))^2 \quad (6.17)$$

где  $y_j^q(p)$  – реальное выходное состояние нейрона  $j$  выходного слоя  $q$  нейронной сети при подаче на ее входы  $p$ -го образа;  $d_j(p)$  – идеальное (желаемое) выходное состояние этого нейрона.

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

$$\begin{aligned} w_{ij}^k(t+1) &= w_{ij}^k(t) + \Delta w_{ij}^k, \\ \Delta w_{ij}^k &= -h \cdot \frac{\partial D}{\partial w_{ij}^k}. \end{aligned} \quad (6.18)$$

Здесь  $w_{ij}^k$  – весовой коэффициент синаптической связи, соединяющей  $i$ -ый нейрон слоя  $k-1$  с  $j$ -ым нейроном слоя  $k$ ,  $h$  – коэффициент скорости обучения,  $0 < h < 1$ ,  $t$  – номер итерации алгоритма

Из (6.17) и (6.18) следует равенство

$$\frac{\partial D}{\partial w_{ij}^k} = \frac{\partial D}{\partial y_j^k} \cdot \frac{dy_j^k}{ds_j^k} \cdot \frac{\partial s_j^k}{\partial w_{ij}^k}. \quad (6.19)$$

Так как множитель  $\frac{dy_j^k}{ds_j^k}$  является производной этой функции по ее аргументу, из этого следует, что производная активационной функция должна быть определена на всей оси абсцисс. В связи с этим функция единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемых НС. В них применяются такие гладкие функции, как гиперболический тангенс или классический сигмоид с экспонентой.

В случае гиперболического тангенса  $f(s) = th \frac{s}{a}$ ,

$$\frac{dy}{ds} = 1 - s^2. \quad (6.20)$$

В случае функции Ферми  $f(s) = \frac{1}{1 + e^{-as}}$ , получим

$$\frac{dy}{ds} = a \cdot f(s) \cdot (1 - f(s)) \quad (6.20')$$

Третий множитель  $\frac{\partial s_j^k}{\partial w_{ij}^k}$ , очевидно, равен выходу нейрона предыдущего слоя  $y_i^{(k-1)}$ .

Что касается первого множителя в (6.19), он легко раскладывается следующим образом:

$$\frac{\mathcal{J}D}{\mathcal{J}y_j^k} = \sum_i \frac{\mathcal{J}D}{\mathcal{J}y_i^{k+1}} \cdot \frac{dy_i^{k+1}}{ds_i^{k+1}} \cdot \frac{\mathcal{J}s_i^{k+1}}{\mathcal{J}y_j^{k+1}} = \sum_i \frac{\mathcal{J}D}{\mathcal{J}y_i^{k+1}} \cdot \frac{dy_i^{k+1}}{ds_i^{k+1}} \cdot w_{ji}^{k+1}. \quad (6.21)$$

Здесь суммирование по  $i$  выполняется среди нейронов слоя  $k+1$ . Введя новую переменную

$$d_j^k = \frac{\mathcal{J}D}{\mathcal{J}y_j^k} \cdot \frac{dy_j^k}{ds_j^k} \quad (6.22)$$

мы получим рекурсивную формулу для расчетов величин  $d_j^k$  слоя  $n$  из величин  $d_i^{k+1}$  более старшего слоя  $k+1$ .

$$d_j^k = \left[ \sum_i d_i^{k+1} \cdot w_{ji}^{k+1} \right] \cdot \frac{dy_j^k}{ds_j^k} \quad (6.23)$$

Для выходного же слоя

$$d_l^q = (y_l^q - d_l) \cdot \frac{dy_l^q}{ds_l^q}. \quad (6.24)$$

Теперь мы можем записать (6.18) в раскрытом виде:

$$\Delta w_{ij}^k = -h \cdot d_j^k \cdot y_i^{k-1}. \quad (6.25)$$

Иногда для придания процессу коррекции весов некоторой инерционности, сглаживающей резкие скачки при перемещении по поверхности целевой функции, (6.25) дополняется значением изменения веса на предыдущей итерации

$$\Delta w_{ij}^k(t) = -h \cdot (m \cdot \Delta w_{ij}^k(t-1) + (1-m) \cdot d_j^k \cdot y_i^{k-1}) \quad (6.26)$$

где  $m$  – коэффициент инерционности,  $t$  – номер текущей итерации. Такой вариант алгоритма получил название «метод тяжелого шарика».

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

1. Подать на входы сети один из возможных образов  $\bar{x}^0 = \bar{x}(p)$  и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних по формулам (6.16).

2. Рассчитать  $d^q$  для выходного слоя по формуле (6.24):

$$d_i^q = (y_i^q - d_i) \cdot \frac{dy_i^q}{ds_i^q}.$$

Рассчитать по формуле (6.25) или (6.26) изменения весов  $Dw^q$  слоя  $q$ :

$$\Delta w_{ij}^k(t) = -h \cdot (m \cdot \Delta w_{ij}^k(t-1) + (1-m) \cdot d_j^k \cdot y_i^{k-1})$$

3. Рассчитать по формулам (6.23) и (6.25) (или (6.23) и (6.26)) соответственно  $d^k$  и  $Dw^k$  для всех остальных слоев,  $k=q-1, \dots, 1$ :

$$d_j^k = \left[ \sum_i d_i^{k+1} \cdot w_{ji}^{k+1} \right] \cdot \frac{dy_j^k}{ds_j^k},$$

$$\Delta w_{ij}^k(t) = -h \cdot (m \cdot \Delta w_{ij}^k(t-1) + (1-m) \cdot d_j^k \cdot y_i^{k-1}).$$

4. Скорректировать все веса в НС

$$w_{ij}^k(t) = w_{ij}^k(t-1) + \Delta w_{ij}^k(t). \quad (6.27)$$

5. Если ошибка сети существенна, перейти на шаг 1. В противном случае – конец.

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

Из выражения (6.25) следует, что когда выходное значение  $y_i^{k-1}$  стремится к нулю, эффективность обучения заметно снижается. При двоичных входных векторах в среднем половина весовых коэффициентов не будет корректироваться, поэтому область возможных значений выходов нейронов  $[0,1]$  желательно сдвинуть в пределы  $[-0.5,+0.5]$ , что достигается простыми модификациями логистических функций. Например, сигмоид с экспонентой преобразуется к виду

$$\text{[Redacted Formula]} \quad (6.28)$$