

НЕЙРОСЕТЕВЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

1. АРХИТЕКТУРА ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Искусственные нейронные сети (НС) – artificial neural networks (ANN) – устройства параллельных вычислений, состоящие из множества взаимодействующих между собой простых процессоров (искусственных нейронов, узлов, элементов сети).

Нейронная сеть является упрощенной моделью биологической нервной системы человека.

В искусственном нейроне (ИН) входные сигналы комбинируются по некоторому правилу, например, суммируются с настраиваемыми весовыми коэффициентами, а затем преобразуются с помощью линейной или нелинейной функции в скалярную величину. Взаимодействие между ИН осуществляется с помощью настраиваемых синаптических связей.

Несмотря на ограниченные вычислительные возможности каждого нейрона в отдельности, вся сеть целиком, объединяя большое количество нейронов, получает *уникальные свойства* и оказывается способной решать сложные задачи.

Основные свойства нейронных сетей:

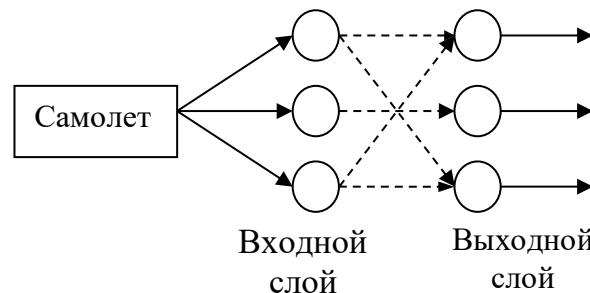
1. *Способность к обучению.* После предъявления заданных входных и выходных сигналов (образов, образцов) сеть обучается (настраивается), чтобы обеспечить желаемую реакцию при произвольных входных воздействиях.

2. *Аппроксимирующие свойства.* Сеть позволяет на ограниченном множестве формировать произвольную непрерывную функцию с заданной точностью.

3. *Ассоциативная память.* Сеть способна запоминать, а затем распознавать даже при неполной или искаженной входной информации предъявленные образы.

Применение. НС предназначены для решения тех задач, в которых в силу неопределенности, например, из-за недостатка информации, традиционные решения не эффективны, а вычисления непомерно трудоемки. Нейронные сети применяются в качестве регуляторов, идентификаторов состояния, для оценки технических процессов (например, на рис. 1.1 приведена схема применения НС для контроля технического состояния самолета, где входной слой – информация с датчиков, выходной слой – индикатор состояния).

Рис. 1.1



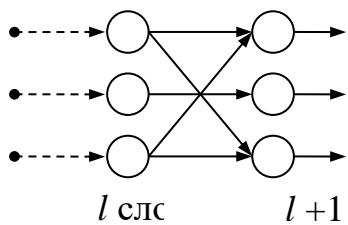
Перспективным является использование НС для управления роботами, при диагностике болезней, для прогнозирования событий, распознавания образов и т. д.

1.1. Классификация НС

НС могут рассматриваться как направленный граф со взвешенными связями, у которых ИН – узлы. НС различаются:

- топологией (способом взаимосвязи ИН, числом слоев);
- методом комбинирования входных сигналов;
- функцией активации;
- алгоритмом обучения;
- сети с непрерывным и с дискретным состояниями;
- сети с непрерывным и с дискретным временами;
- синхронные и асинхронные сети.

Основные способы взаимосвязи ИН:

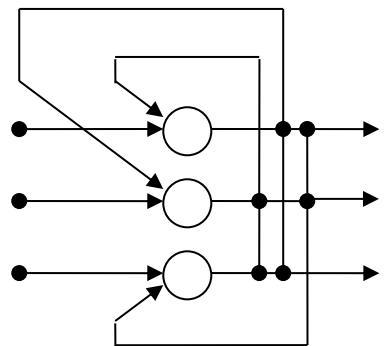


1) полносвязанные сети: все нейроны
связаны между собой;

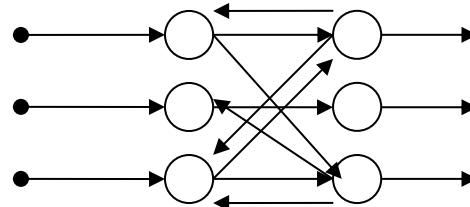
2) сети прямого распространения: сигнал проходит только в одном направлении от входа к выходу,
ИН одного слоя не связаны между собой. Связь с выходов ИН *l*-го слоя подается на входы только *l + 1*-
го слоя (рис. 1.2);

Рис. 1.2

3) рекуррентные сети (сети с обратными связями): сеть Хопфилда (рис. 1.3, *a*); сеть "дву направленная ассоциативная
память" (рис. 1.3, *б*).



а



б

Рис. 1.3

4) сеть Кохонена (рис. 1.4): содержит входной слой и слой Кохонена

(все нейроны расположены на плоскости и связаны между собой).

Сети с непрерывным состоянием. Выход сети – действительные значения произвольного вида или действительные значения из ограниченного интервала ($[0,1]$).

Сети с дискретным состоянием. Выход сети –

дискретные значения (например, бинарные): $\{0, 1\}$; $\{-1, 1\}$, где 1 – возбужденное состояние; 0 (–1) – заторможенное состояние.

Асинхронные сети: в каждый момент меняется состояние только одного нейрона слоя.

Синхронные сети: в каждый момент меняется состояние группы нейронов, как правило, всего слоя.

Сети прямого распространения – *статические*, так как не имеют динамических элементов и обратных связей. Выход сети зависит только от заданного множества на входе.

Рекуррентные сети – *динамические*, так как содержат обратные связи, текущее состояние сети зависит от предыдущего.

1.2. Искусственный нейрон (персептрон)

ИН – элементарный преобразующий элемент, имеющий n -вектор входа r , суммирующий блок, блок преобразования сигнала с помощью функции активации, скалярный выход q (рис. 1.5). Каждому входу r_j приписан весовой коэффициент W_j , соответствующий силе биологической "синаптической" связи. Весовые коэффициенты настраиваются в процессе обучения.

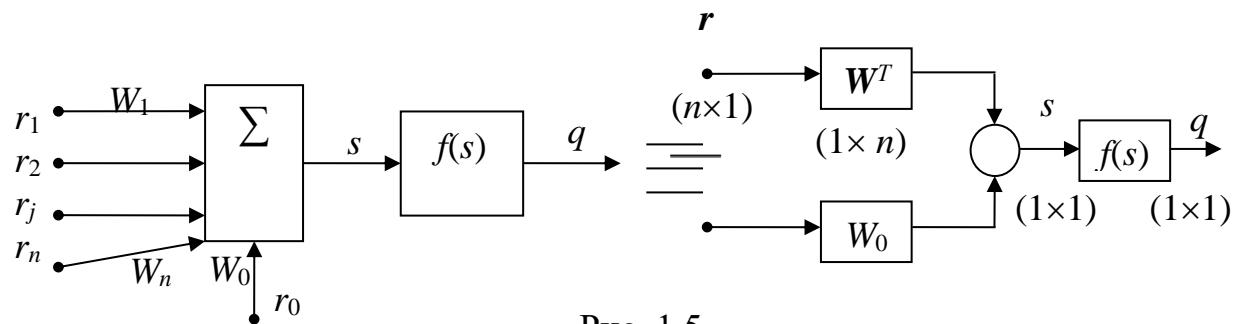


Рис. 1.5

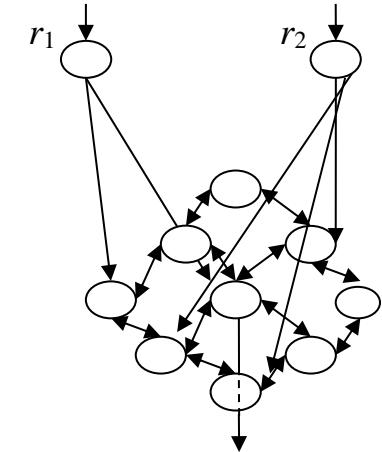


Рис. 1.4

В зависимости от весовых коэффициентов передаваемый сигнал усиливается или подавляется. Нулевой коэффициент означает отсутствие связи между нейронами.

Дополнительный вход r_0 с коэффициентом W_0 , равным пороговому значению функции активации с противоположным знаком, чтобы пороговое значение было нулевым (смещение ИН). Обычно $r_0 = 1$.

Таким образом, ИН осуществляет преобразование входного вектора \mathbf{r} в скаляр q , выполняя два действия.

1. Комбинирование входных сигналов – вычисление взвешенной суммы s :

$$s = \sum_{j=1}^n W_j r_j + W_0 r_0 = \sum_{j=0}^n W_j r_j = \mathbf{W}^T \mathbf{r} + W_0 r_0 = \tilde{\mathbf{W}}^T \tilde{\mathbf{r}}, \quad (1.1)$$

где $\mathbf{r} = [r_1 \ r_2 \ \dots \ r_n]^T$ – входной вектор; $\tilde{\mathbf{r}}^T = [r_0 : r_1 \ r_2 \ \dots \ r_n]$ – расширенный входной вектор; $\mathbf{W}^T = [W_1 \ W_2 \ \dots \ W_n]$ – вектор весовых коэффициентов ИН; $\tilde{\mathbf{W}}^T = [W_0 : W_1 \ W_2 \ \dots \ W_n]$ – расширенный вектор весовых коэффициентов ИН, память нейрона.

2. Преобразование взвешенной суммы с помощью функции активации для получения выхода нейрона:

$$q = f(s) = f\left(\sum_{j=0}^n W_j r_j\right) = f(\tilde{\mathbf{W}}^T \tilde{\mathbf{r}}). \quad (1.2)$$

Основные правила вычисления функции активации:

1) линейная (тождественная) функция:

$$f(s) = s, \quad (1.3)$$

График функции представлен на рис. 1.6, а. Выход нейрона $q = s = \sum_{j=0}^n W_j r_j$. ИН с линейной функцией активации в публикациях

часто называют линейным взвешенным сумматором или линейным ИН.

$$2) \text{ пороговая функция активации: } f(s) = \begin{cases} 1, & s \geq \Theta; \\ 0, & s < \Theta, \end{cases}$$

где Θ – пороговое значение. Как правило, удобнее вычесть из взвешенной суммы пороговое значение, названное *смещением*, $W_0 = -\Theta$. Функция активации будет вычисляться

$$f(s) = \begin{cases} 1, & s \geq 0; \\ 0, & s < 0, \end{cases} \quad (1.4)$$

где 0 – пороговое значение. График функции представлен на рис. 1.6, б. Компонент смещения обычно интерпретируют как связь, исходящую из дополнительного нейрона предыдущего слоя, выход которого $q_0 = 1$, либо как связь из дополнительного входа $r_0 = 1$. Модель ИН с пороговой функцией активации предложена МакКаллоком и Питтсом (1943) и называется *персептроном*;

3) *сигмоидные* функции активации:

- логистическая (униполярная) функция:

$$f(s) = \frac{1}{1 + e^{-\beta s}}, \quad (1.5) \quad f'(s) = \beta \cdot f(s)[1 - f(s)],$$

где β – положительный коэффициент; выходное значение нейрона лежит в диапазоне (0, 1) (рис. 1.7, а);

- гиперболический тангенс (биполярная) функция:

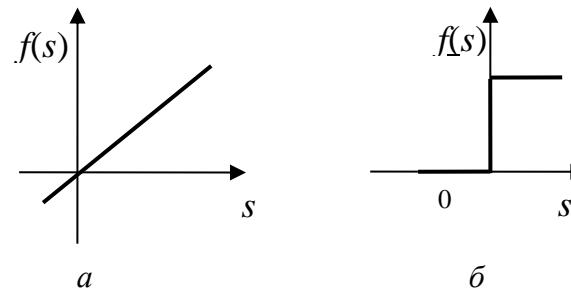


Рис. 1.6

$$f(s) = \frac{e^{\beta s} - e^{-\beta s}}{e^{\beta s} + e^{-\beta s}}, \quad f'(s) = \beta \cdot [1 - f(s)^2]; \quad (1.6)$$

выходное значение нейрона лежит в диапазоне $(-1, 1)$ (рис. 1.7, δ).



Рис. 1.7

При уменьшении β сигмоид становится более пологим, в пределе при $\beta = 0$ вырождаясь в горизонтальную линию, при увеличении β сигмоид приближается к виду пороговой функции.

Следует отметить, что сигмоидные функции дифференцируемы на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того, они обладают свойством усиливать слабые сигналы лучше, чем большие. Модель ИН с сигмоидной функцией активации называют *сигмоидным нейроном*;

4) *радиальная базисная* функция (функция Гаусса) используется в *радиальных* нейронах. Функция активации радиального нейрона (рис. 1.8)

$$f(s) = e^{-\beta s^2}. \quad (1.7)$$

Пример 1.1. Определить величину выхода нейрона с пороговой функцией активации (1.4), изображенного на рис. 1.9.

Взвешенная сумма ИН $s = \sum_{j=0}^3 W_j r_j = (-1) \cdot 1 + 0.5 \cdot 1 + (-2) \cdot 0.4 + 4 \cdot 0.25 = -0.3$.

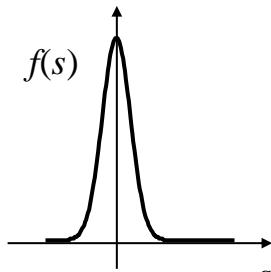


Рис. 1.8

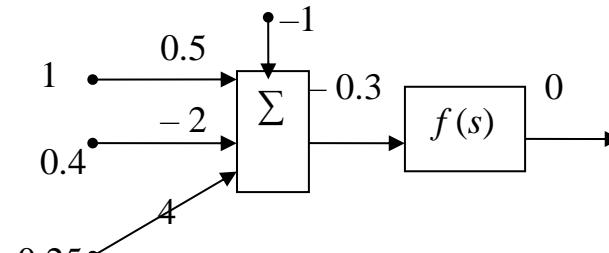


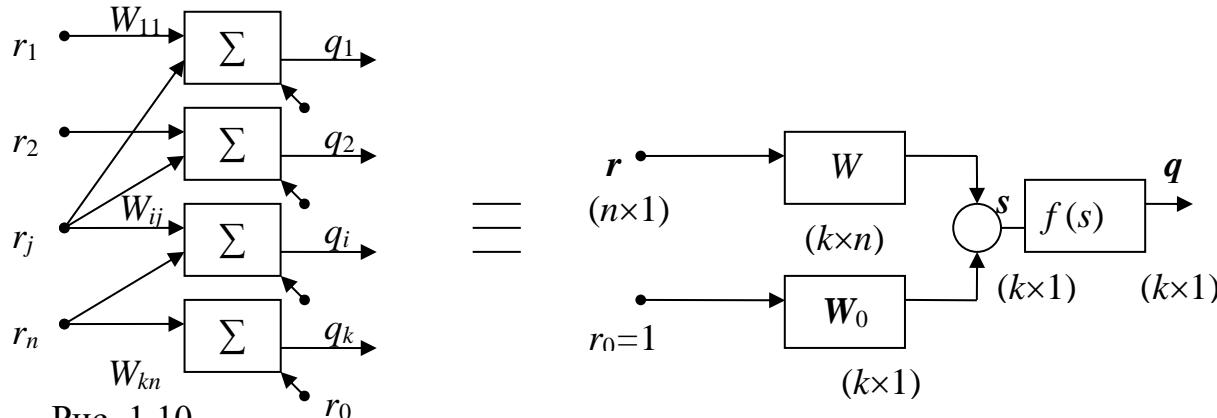
Рис. 1.9

Выход нейрона $q = f(s) = \begin{cases} 1, & s \geq 0, \\ 0, & s < 0, \end{cases}$ $q = f(-0.3) = 0$.

1.3. Линейные однослойные сети прямого распространения

Сеть имеет n -размерный вектор входов \mathbf{r} и выходной вектор \mathbf{q} размером k . Функция активации – линейная (1.3).

Единственный слой состоит из k линейных, не связанных между собой ИН (рис. 1.10).



Индексы коэффициента W_{ij} : i – номер ИН, j – номер входа.

(Индексация зависит от способа записи уравнений ИН. Если в формуле (1.8) коэффициент W_{ij} умножается на составляющую вектора входа r_j , то 1-й индекс соответствует номеру ИН. Возможна запись, в которой первый сомножитель r_j , тогда первый индекс соответствует номеру входа).

Выход i -го нейрона ($f(s) = s$)

$$q_i = \sum_{j=1}^n W_{ij} r_j + W_{i0} r_0 = \sum_{j=0}^n W_{ij} r_j, \quad i = \overline{1, k}. \quad (1.8)$$

Выход сети в векторной форме: $\mathbf{q} = \mathbf{Wr} + \mathbf{W}_0 \mathbf{r}_0 = \tilde{\mathbf{W}} \cdot \tilde{\mathbf{r}}$.

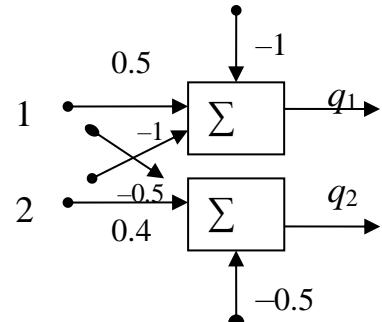
Расширенная матрица весовых коэффициентов:

$$\tilde{W} = \begin{bmatrix} W_{10} & W_{11} & \dots & W_{1n} \\ W_{20} & \dots & \dots & \dots \\ \vdots & \dots & \dots & \dots \\ W_{k0} & W_{k1} & \dots & W_{kn} \end{bmatrix},$$

W_0 W

где W_0 – вектор весовых коэффициентов смещения; W – матрица весовых коэффициентов слоя.

Пример 1.2. Определить выход сети (рис. 1.11).



Расширенная матрица весовых коэффициентов $\tilde{W} = \begin{bmatrix} -1 & \vdots & 0.5 & -1 \\ -0.5 & \vdots & -0.5 & 0.4 \end{bmatrix}$. Выход сети $\mathbf{q} = \tilde{W} \cdot \tilde{\mathbf{r}} =$

$$= \begin{bmatrix} -1 & \vdots & 0.5 & -1 \\ -0.5 & \vdots & -0.5 & 0.4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -2.5 \\ -0.2 \end{bmatrix}.$$

Рис. 1.11

1.4. Многослойные нейронные сети прямого распространения

Входной вектор подается на входы ИН только первого – входного слоя, переработанные сетью сигналы снимаются с выходов последнего выходного слоя, остальные – промежуточные слои – скрытые (рис. 1.12).

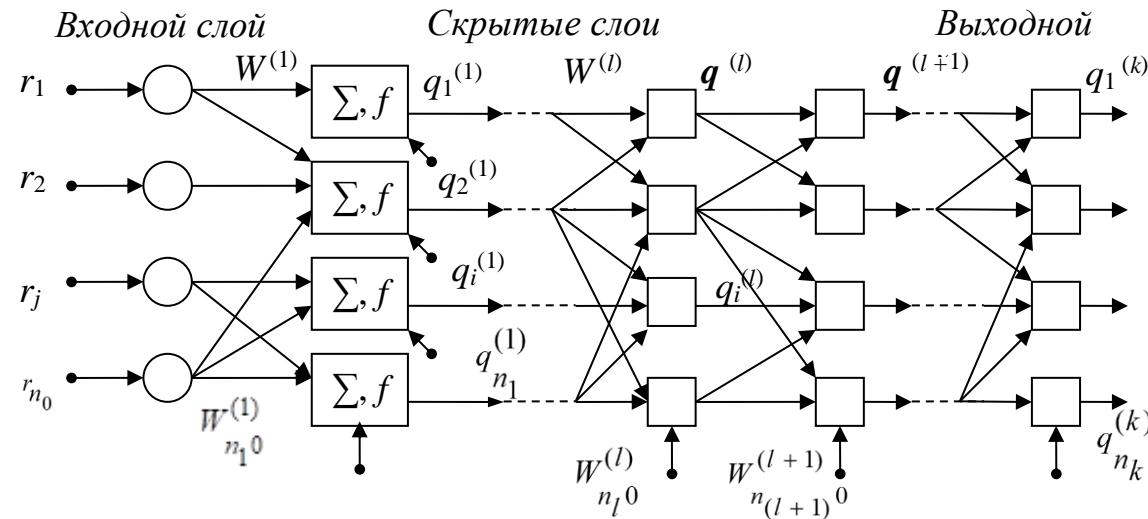


Рис. 1.12

$W^{(l)}$ – матрица весовых коэффициентов l -го слоя, $l = \overline{1, k}$; n_l – число ИН в l слое; $\mathbf{q}^{(l)}$ – вектор выхода l -го скрытого слоя; $q_i^{(l)}$ – выход i -го нейрона l -го скрытого слоя; $W_{n_l 0}^{(l)}$ – весовой коэффициент связи входа n_l нейрона l -го слоя и выхода дополнительного нейрона $l-1$ слоя $q_0^{(l-1)}$.

Входной слой служит только для распределения сигналов между ИН первого скрытого слоя, не осуществляя преобразование сигнала. ИН одного слоя не связаны между собой. Сигналы с выходов ИН l -го слоя поступают на входы $l+1$ -го слоя. Функция активации принимается одинаковой для нейронов всех слоев, кроме выходного. Выходной слой, как правило, с линейными функциями активации (1.3).

Обозначение многослойной нейронной сети: $N_{n_0, n_1, \dots, n_i, \dots, n_k}^k$,

где k – число слоев; n_0 – число входов; n_i – число ИН в i слое, $i = \overline{1, k - 1}$; n_k – число нейронов выходного слоя и число выходов.

Замечания:

1. Многослойная НС прямого распространения с пороговой или с сигмоидной функциями активации называется *многослойным персепtronом*.
2. Двухслойная НС, у которой нейроны скрытого слоя с радиальной базисной функцией активации, а нейроны выходного слоя с линейной, называется *радиальной сетью*.

1.5. Послойное описание НС

Выход i -го ИН l -го слоя согласно выражению (1.8) (для линейного ИН $q_i = \sum_{j=1}^n W_{ij} r_j + W_{i0} r_0 = \sum_{j=0}^n W_{ij} r_j$):

$$q_i^{(l)} = f\left(\sum_{j=1}^{n_{l-1}} W_{ij}^{(l)} q_j^{(l-1)} + W_{i0}^{(l)} q_0^{(l-1)}\right), \quad i = \overline{1, n_l}, \quad (1.9)$$

где n_{l-1} – число ИН $l-1$ -го слоя; $q_0^{(l-1)} = 1$ – дополнительный вход для формирования смещений нейронов l -го слоя; $W_{ij}^{(l)}$ – весовой коэффициент, связывающий i -й нейрон l -го слоя и j -й нейрон $l-1$ -го слоя; $W_{i0}^{(l)}$ – весовой коэффициент смещения.

Вектор выхода l -го слоя:

$$\mathbf{q}^{(l)} = f(\mathbf{W}^{(l)} \mathbf{q}^{(l-1)} + \mathbf{W}_0^{(l)} q_0^{(l-1)}) = f(\tilde{\mathbf{W}}^{(l)} \tilde{\mathbf{q}}^{(l-1)}), \quad (1.10)$$

где $\mathbf{W}_0^{(l)}$ – вектор весовых коэффициентов смещения l -го слоя; $\tilde{\mathbf{W}}^{(l)}$ – расширенная матрица весовых коэффициентов l -го слоя размером $n_l \times (n_{l-1} + 1)$:

$$\tilde{W}^{(l)} = \begin{bmatrix} W_{10}^{(l)} & W_{11}^{(l)} & \dots & W_{1n_{l-1}}^{(l)} \\ W_{20}^{(l)} & \dots & \dots & \dots \\ \dots & \vdots & \dots & \dots \\ W_{n_l 0}^{(l)} & W_{n_l 1}^{(l)} & \dots & W_{n_l n_{l-1}}^{(l)} \end{bmatrix}.$$

Функция активации выходного слоя линейная, поэтому вектор выхода последнего слоя $\mathbf{q}^{(k)} = \tilde{W}^{(k)} \tilde{\mathbf{q}}^{(k-1)}$.

1.6. Связь между входом и выходом НС

Согласно уравнению (1.10) запишем векторы выходов 1-го, 2-го и 3-го слоев. Обычно r_0 и q_0 в формулах опускают.

Вектор выхода нейронов 1-го слоя:

$$\mathbf{q}^{(1)} = f^{(1)}(W^{(1)}\mathbf{r} + W_0^{(1)}).$$

Вектор выхода нейронов 2-го слоя:

$$\mathbf{q}^{(2)} = f^{(2)}(W^{(2)}\mathbf{q}^{(1)} + W_0^{(2)}) = f^{(2)}(W^{(2)}f^{(1)}(W^{(1)}\mathbf{r} + W_0^{(1)}) + W_0^{(2)}).$$

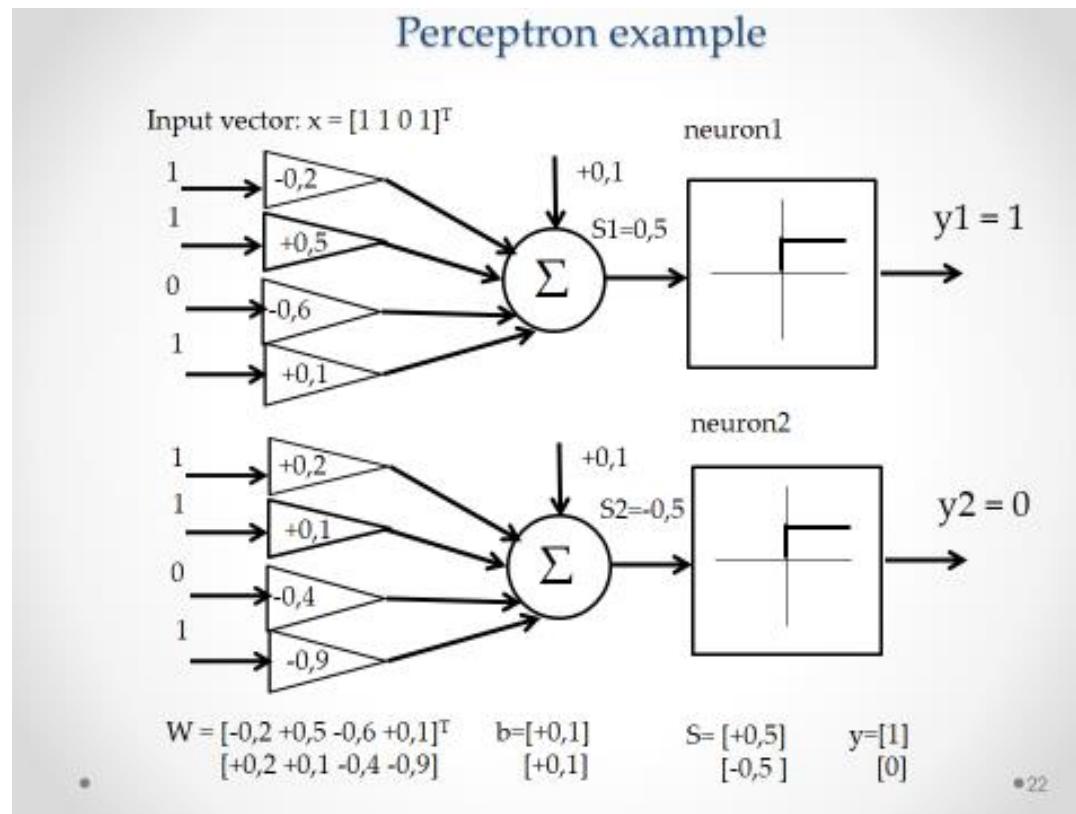
Вектор выхода нейронов 3-го слоя:

$$\begin{aligned} \mathbf{q}^{(3)} &= f^{(3)}(W^{(3)}\mathbf{q}^{(2)} + W_0^{(3)}) = f^{(3)}(W^{(3)}f^{(2)}(W^{(2)}\mathbf{q}^{(1)} + W_0^{(2)}) + W_0^{(3)}) = \\ &= f^{(3)}(W^{(3)}f^{(2)}(W^{(2)}f^{(1)}(W^{(1)}\mathbf{r} + W_0^{(1)}) + W_0^{(2)}) + W_0^{(3)}). \end{aligned}$$

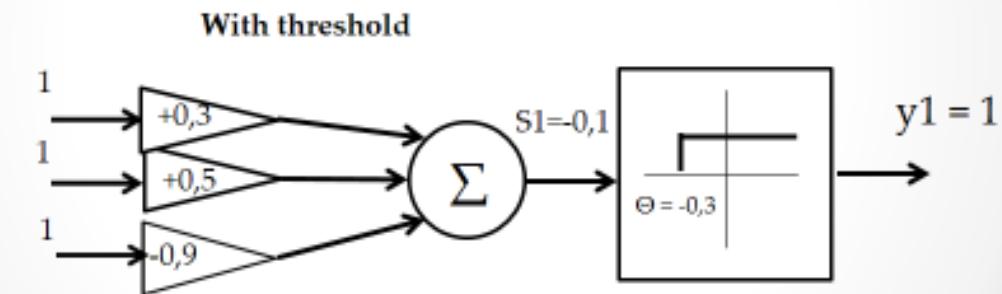
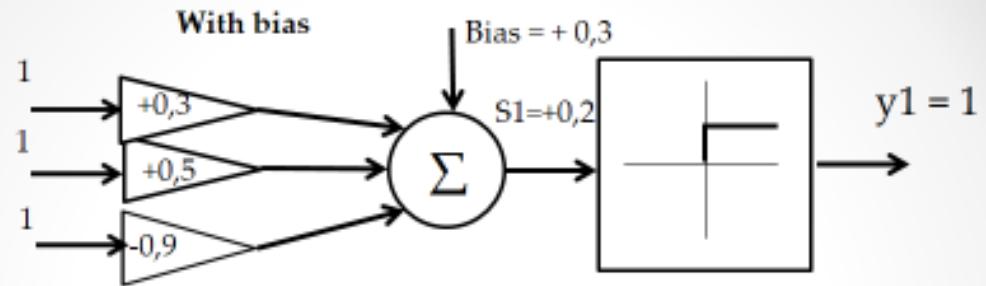
Для НС из k слоев получим рекуррентное соотношение:

$$\mathbf{u} = \mathbf{q}^{(k)} = f^{(k)}(\mathbf{W}^{(k)} f^{(k-1)}(\mathbf{W}^{(k-1)} f^{(k-2)}(\mathbf{W}^{(k-2)} f^{(k-3)}(\dots f^{(1)}(\mathbf{W}^{(1)} \mathbf{r} + \\ + \mathbf{W}_0^{(1)}) + \dots + \mathbf{W}_0^{(k-1)}) + \mathbf{W}_0^{(k)}) = F(\mathbf{r}).$$

Таким образом, НС осуществляет сложное нелинейное преобразование F входного вектора \mathbf{r} в зависимости от выбранных функций активации и матриц весовых коэффициентов.



The bias interpretation as the threshold



The threshold $\theta = -0,3$

*

*23

1.7. Классификация образов с помощью нейронной сети

Входной образ (речевой сигнал, рукописный символ) принадлежит n -пространству (гиперпространству) и характеризуется вектором признаков.

Задача: отнесение образов к заранее определенным классам; разбиение гиперпространства гиперплоскостями.

Каждая отдельная область – область определенного класса. Гиперплоскость – *разделяющая поверхность*.

Решение задачи классификации с помощью одного искусственного нейрона с пороговой функцией активации, $f(s) = \begin{cases} 1, & s \geq 0; \\ 0, & s < 0 \end{cases}$.

Искусственный нейрон формирует $(n - 1)$ -мерную гиперплоскость, задаваемую уравнением

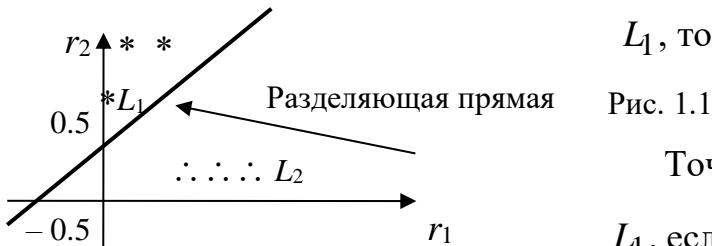
$$s = \sum_{j=1}^n W_j r_j + W_0 r_0 = \sum_{j=0}^n W_j r_j = 0, \quad (1.11)$$

разделяющую n -мерное пространство входных векторов на две области, либо $\sum_{j=0}^n W_j r_j > 0$, либо $\sum_{j=0}^n W_j r_j < 0$.

Если размер входного вектора (число признаков) $n = 2$, гиперплоскость преобразуется в *разделяющую прямую*, задаваемую уравнением $s = W_1 r_1 + W_2 r_2 + W_0 = 0$. Если $n = 3$, формируется плоскость.

Определим, к какому из двух классов, обозначенных символами L_1 и L_2 , принадлежит входной вектор r (рис. 1.13).

На рис. 1.13 точка (r_1, r_2) , $(*)$, лежащая над разделяющей прямой, относится к классу



L_1 , точка (r_1, r_2) (\therefore), лежащая под разделяющей прямой, относится к классу L_2 .

Рис. 1.13

Точки, лежащие на прямой, могут относиться и к L_1 , и к L_2 . Вектор r относится к классу L_1 , если выходной сигнал ИН q принимает значение 0, и к классу L_2 , если выходной сигнал q принимает значение 1.

Пример 1.3. Схема ИН приведена на рис. 1.14. Функция активации пороговая (1.4): $q = f(s) = \begin{cases} 1, & s \geq 0 - L_2 (\ldots); \\ 0, & s < 0 - L_1 (*). \end{cases}$ Определить,

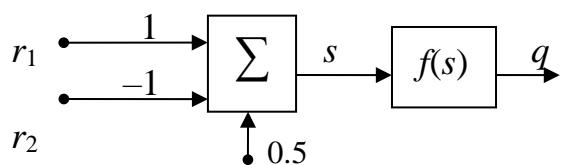
к какому классу принадлежит входной вектор, если $r_1 = 0.4$, $r_2 = 0.5$.

Уравнение разделяющей прямой: $s = 0.5 + r_1 - r_2 = 0$; $r_2 = r_1 + 0.5$.

Подставив значения входного вектора, получим $s = 0.5 + 1 \cdot 0.4 + (-1) \cdot 0.5 = 0.4 > 0$,

следовательно, $q = 1$, вектор принадлежит классу L_2 .

Рис. 1.14



1.8. Линейно разделимые и линейно неразделимые функции

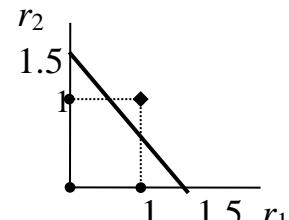
Если прямая (для размерности два) или гиперплоскость (для размерности n) могут разделить все образы на классы, то функция называется **линейно разделимой**. Если для разбиения образов на классы требуется несколько прямых или гиперплоскостей, то функция называется **линейно неразделимой**.

Пример 1.4. Реализация логических операции "И" (рис. 1.15), "ИЛИ" (рис. 1.16) на основе персептрана.

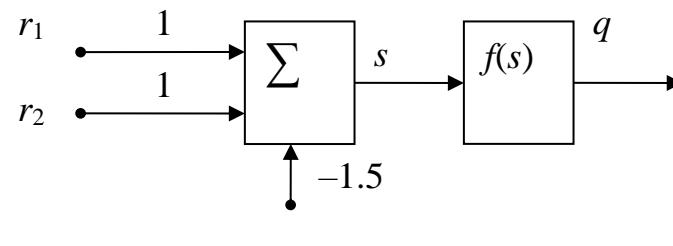
Таблица истинности операции "И", значения взвешенной суммы s и выхода ИН q , график разделяющей прямой, схема персептрана показаны на рис. 1.15, $a - e$ соответственно.

r_1	r_2	s	q
1	1	0.5	1
1	0	-0.5	0
0	1	-0.5	0
0	0	-1.5	0

a



δ



e

Рис. 1.15

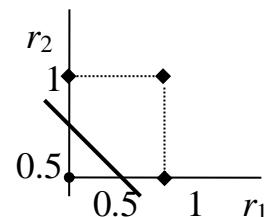
Взвешенная сумма входных сигналов $s = 1 \times r_1 + 1 \times r_2 - 1.5$. Уравнение разделяющей прямой ($s = 0$) $r_2 = -r_1 + 1.5$.

Реализация логической операции "ИЛИ"

Таблица истинности операции "ИЛИ", график разделяющей прямой, схема персептрана показаны на рис. 1.16 $a - e$.

r_1	r_2	s	q
1	1	1.5	1
1	0	0.5	1
0	1	0.5	1
0	0	-0.5	0

a



δ

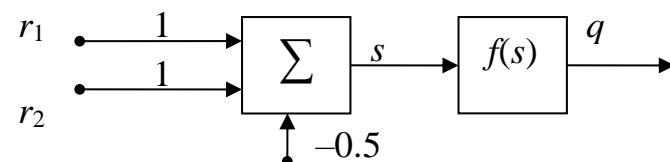


Рис. 1.16

Взвешенная сумма входных сигналов $s = 1 \times r_1 + 1 \times r_2 - 0.5$. Уравнение разделяющей прямой $r_2 = -r_1 + 0.5$.

Прямая на рис. 1.15, б делит все образы на классы, векторы $\mathbf{r}^T = [0 \ 0]$ и $\mathbf{r}^T = [0 \ 1]$ $\mathbf{r}^T = [1 \ 0]$ принадлежат одному классу:

$r_1 \text{ И } r_2 = 0$, вектор $\mathbf{r}^T = [1 \ 1]$ принадлежат другому классу $r_1 \text{ И } r_2 = 1$.

Прямая на рис. 1.16, б делит все образы на классы, векторы $\mathbf{r}^T = [1 \ 1]$ и $\mathbf{r}^T = [0 \ 1]$ $\mathbf{r}^T = [1 \ 0]$ принадлежат одному классу:

$r_1 \text{ ИЛИ } r_2 = 1$, вектор $\mathbf{r}^T = [0 \ 0]$ принадлежат другому классу $r_1 \text{ ИЛИ } r_2 = 0$.

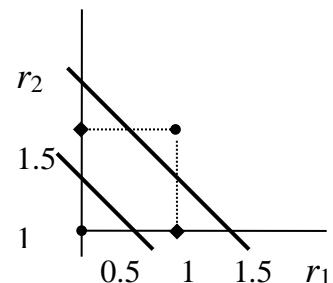
Таким образом, логические операции "И" и "ИЛИ" – линейно разделимые функции.

Реализация логической операции "ИСКЛЮЧАЮЩЕ ИЛИ" ("XOR")

Таблица истинности операции "XOR", график разделяющей прямой, схема персептрона показаны на рис. 1.17а – в.

r_1	r_2	q
1	1	0
1	0	1
0	1	1
0	0	0

а



б

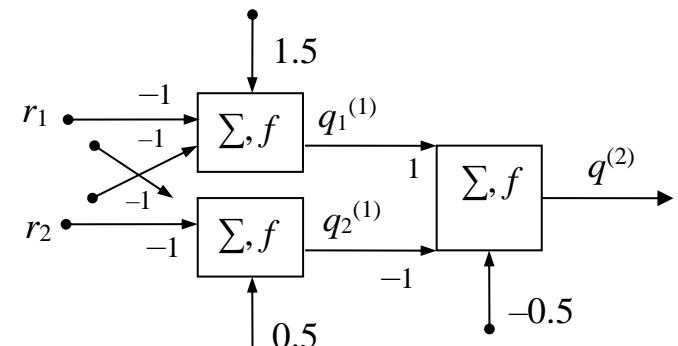


Рис. 1.17

в

Одна прямая (рис. 1.17, б) не может разделить образы (0 и 1) на два класса, функция является линейно неразделимой.

Для реализации функции возможен один из двух способов:

- 1) использовать сеть, которая будет строить две и более прямых для разделения данных;
 - 2) увеличить число вводимых признаков (размер входного вектора).
- 1). Сеть должна включать два ИН, которые служат для формирования двух разделяющих прямых, и третий ИН, объединяющий информацию об этих прямых (рис. 1.17, в).

Пример 1.5. Реализация логической операции "ИСКЛЮЧАЮЩЕЕ ИЛИ" ("XOR").

Матрица весовых коэффициентов 1-го слоя: $\tilde{W}^{(1)} = \begin{bmatrix} 1.5 & -1 & -1 \\ 0.5 & -1 & -1 \end{bmatrix}$,

матрица весовых коэффициентов 2-го слоя: $\tilde{W}^{(2)} = [-0.5 \ 1 \ -1]^T$.

Взвешенная сумма 1-го ИН 1-го (скрытого) слоя: $s_1^{(1)} = -1 \times r_1 - 1 \times r_2 + 1.5$. Уравнение разделяющей прямой ($s = 0$): $r_2 = -r_1 + 1.5$.

Взвешенная сумма 2-го ИН 1-го (скрытого) слоя: $s_2^{(1)} = -1 \times r_1 - 1 \times r_2 + 0.5$. Уравнение разделяющей прямой: $r_2 = -r_1 + 0.5$.

Два ИН первого (скрытого) слоя формируют две разделяющие прямые.

Для 2-го слоя $s^{(2)} = 1 \times q_1^{(1)} - 1 \times q_2^{(1)} - 0.5$. Уравнение разделяющей прямой $q_2^{(1)} = q_1^{(1)} - 0.5$.

В табл. 1.1 приведены значения взвешенных сумм и выходов ИН 1-го и 2-го слоев НС, показанной на рис. 1.17, в.

Таблица 1.1

r_1	r_2	$s_1^{(1)}$	$q_1^{(1)}$	$s_2^{(1)}$	$q_2^{(1)}$	$s^{(2)}$		$q^{(2)}$
1	1	-0.5	0	-1.5	0	-0.5		0
1	0	0.5	1	-0.5	0	0.5		1
0	1	0.5	1	-0.5	0	0.5		1
0	0	1.5	1	0.5	1	-0.5		0

Согласно табл. 1.1 и рис. 1.18:

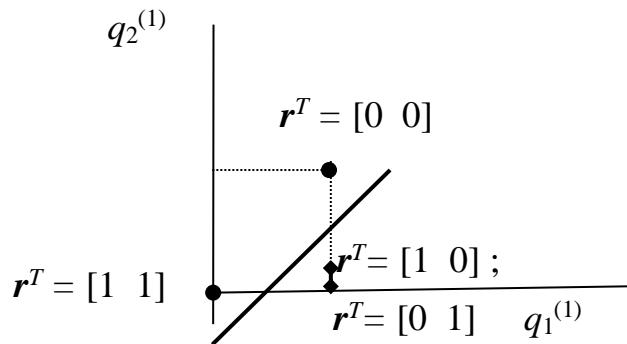


Рис. 1.18

входному вектору $\mathbf{r}^T = [0 \ 0]$ соответствует вектор выхода ИН 1-го слоя $\mathbf{q}^{(1)} = [1 \ 1]^T$;

вектору $\mathbf{r}^T = [1 \ 1]$ – вектор $\mathbf{q}^{(1)} = [0 \ 0]^T$;

векторам $\mathbf{r}^T = [1 \ 0]$ и $\mathbf{r}^T = [0 \ 1]$ – вектор $\mathbf{q}^{(1)} = [1 \ 0]$.

Векторы $\mathbf{r}^T = [0 \ 0]$ и $\mathbf{r}^T = [1 \ 1]$ принадлежат одному классу: $r_1 \text{ XOR } r_2 = 0$.

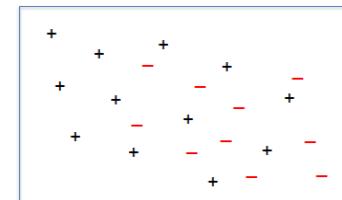
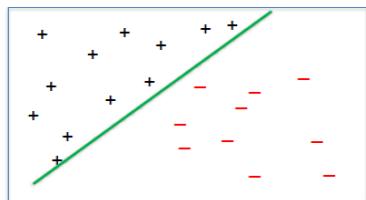
Векторы $\mathbf{r}^T = [1 \ 0]$ и $\mathbf{r}^T = [0 \ 1]$ также принадлежат одному классу: $r_1 \text{ XOR } r_2 = 1$.

Таким образом, входные векторы 2-го слоя линейно разделимы.

Вывод

С помощью двухслойной сети с пороговыми функциями активации можно сформировать логическую операцию "ИСКЛЮЧАЮЩЕЕ ИЛИ".

Линейно разделимые и линейно неразделимые функции



Ограничения сетей с линейными функциями активации

Две сети прямого распространения с линейными функциями активации (1.3) ($f(s) = s$):

двухслойная сеть – $N_{2,2,3}^2$ (рис. 1.19, а) и однослойная сеть – $N_{2,3}^1$ (рис. 1.19, б).

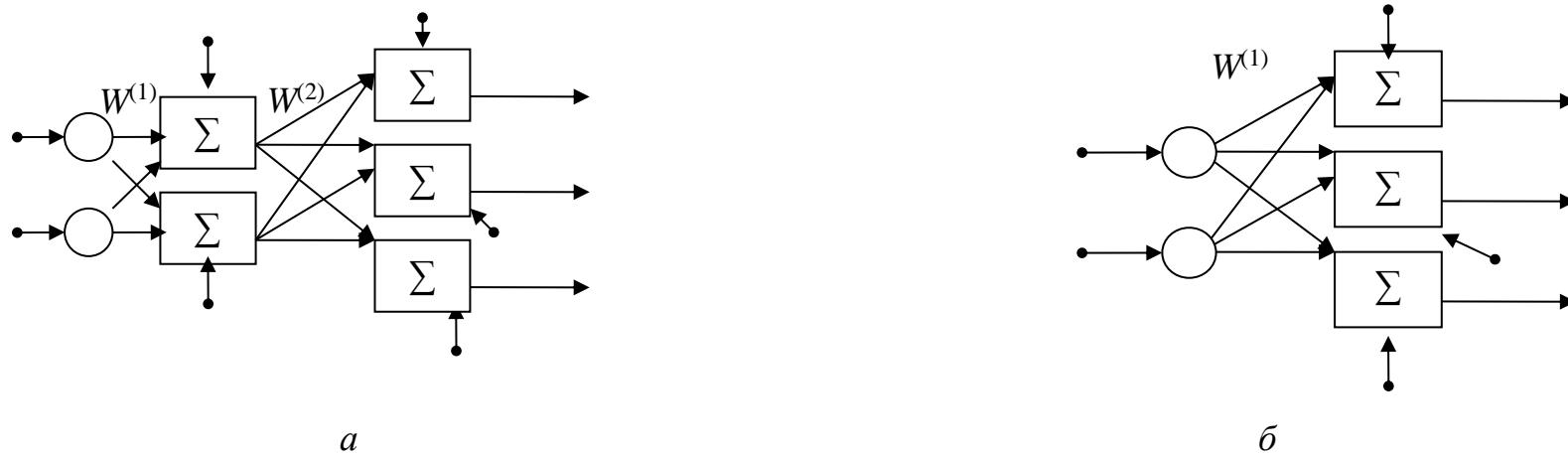


Рис. 1.19

Пусть входной вектор $\tilde{r} = [1; 0 \ 1]^T$.

Матрицы весовых коэффициентов первой сети $\tilde{W}^{(1)} = \begin{bmatrix} 1 & 2 & -3 \\ -2 & 0.5 & 1 \end{bmatrix}$, $\tilde{W}^{(2)} = \begin{bmatrix} 2 & -1 & -3 \\ 1 & 5 & 1 \\ 3 & 4 & 2 \end{bmatrix}$.

В расширенных матрицах весовых коэффициентов выделен столбец весовых коэффициентов смещения.

В расширенном векторе входа выделен дополнительный вход $r_0 = 1$.

Выход нейронов первого слоя первой сети

$$q^{(1)} = \tilde{W}^{(1)} \tilde{r} = \begin{bmatrix} -2 \\ -1 \end{bmatrix}.$$

Выход первой сети

$$q^{(2)} = \tilde{W}^{(2)} \tilde{q}^{(1)} = \begin{bmatrix} 7 \\ -10 \\ -7 \end{bmatrix}.$$

Матрицы весовых коэффициентов второй сети

$$\tilde{W}^{(1)} = \begin{bmatrix} 5 & -3.5 & 2 \\ 3 & 10.5 & -13 \\ 3 & 9 & -10 \end{bmatrix}.$$

Выход сети

$$q = \tilde{W}^{(1)} \tilde{r} = \begin{bmatrix} 5 & -3.5 & 2 \\ 3 & 10.5 & -13 \\ 3 & 9 & -10 \end{bmatrix} \times \begin{bmatrix} 1 \\ \dots \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ -10 \\ -7 \end{bmatrix}.$$

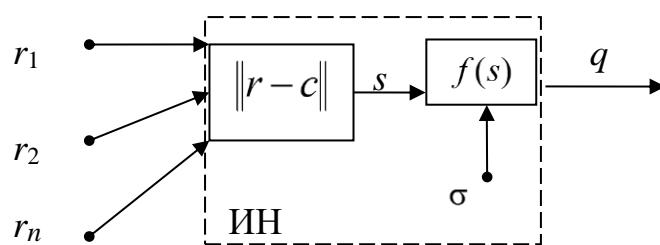
Таким образом, многослойную нейронную сеть с линейными функциями активации можно заменить *однослоиной сетью*, следовательно, с помощью данной сети можно решать только те задачи, которые могут быть решены однослоиной сетью, т. е. можно реализовать только *линейно разделимые функции*.

Выход двухслойной сети с линейными функциями активации $\mathbf{q}^{(2)} = \mathbf{W}^{(2)}(\mathbf{W}^{(1)}\mathbf{r} + \mathbf{W}_0^{(1)}) + \mathbf{W}_0^{(2)}$.

(Из (1.10) $\mathbf{q}^{(2)} = f^{(2)}(\mathbf{W}^{(2)}\mathbf{q}^{(1)} + \mathbf{W}_0^{(2)}) = f^{(2)}(\mathbf{W}^{(2)}f^{(1)}(\mathbf{W}^{(1)}\mathbf{r} + \mathbf{W}_0^{(1)}) + \mathbf{W}_0^{(2)})$).

1.9. Структура радиального нейрона и радиальной сети

Радиальный ИН включает n -вектор входа \mathbf{r} , блок вычисления евклидова расстояния s вектора входа от центра \mathbf{c} радиальной базисной функции (опорной точки, опорного вектора), блок преобразования с помощью радиальной базисной функции активации (рис. 1.20) и скалярный выход q . Опорные точки \mathbf{c} определяются на основе обучающей последовательности и имеют размерность входного вектора. В простейшем случае \mathbf{r}_i – ому вектору обучающей последовательности соответствует i -ый ИН скрытого слоя



с весовыми коэффициентами \mathbf{W}_i , определяющий \mathbf{c}_i опорную точку $\mathbf{r}_i = \mathbf{W}_i = \mathbf{c}_i$.
Дополнительно водится параметр рассеяния σ , от значения которого зависит ширина функции активации (рис. 1.20) σ^2 – дисперсия.
Чаще всего в качестве радиальной функции принимается функция Гаусса вида

Рис. 1.20 а

$$f(s) = f(\|\mathbf{r} - \mathbf{c}\|) = \exp\left(-\frac{\|\mathbf{r} - \mathbf{c}\|^2}{2\sigma^2}\right).$$

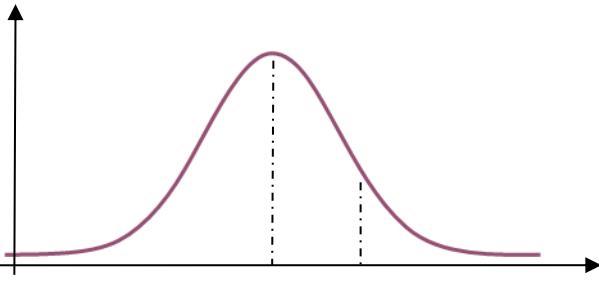


Рис. 1.20 б – Функция Гаусса

ИН выполняет два действия:

- 1) вычисление расстояния между входным и опорным векторами нейрона

$$s = \|\mathbf{r} - \mathbf{c}\| = \left[\sum_j (r_j - c_j)^2 \right]^{1/2}; \quad (1.12)$$

- 2) преобразование с помощью радиальной базисной функции активации $f(s) = f(\|\mathbf{r} - \mathbf{c}\|) = \exp\left(-\frac{\|\mathbf{r} - \mathbf{c}\|^2}{2\sigma^2}\right)$.

Персептрон представляется в многомерном пространстве гиперплоскостью, разделяющей пространство на два класса, в

которых выполняется одно из двух условий: либо $\sum_{j=0}^n W_j r_j > 0$, либо $\sum_{j=0}^n W_j r_j < 0$ (рис. 1.21, а)

Радиальный нейрон представляет собой гиперсферу, осуществляющую шаровое разделение пространства вокруг центральной точки (рис. 1.21, б).

Радиальная сеть – сеть прямого распространения, содержит два слоя: скрытый слой с радиальными ИН и выходной слой с линейными ИН.

В первой слое сети выполняется предварительная обработка входных векторов \mathbf{r} , вычисление их близости к центрам \mathbf{c}_i . Нейроны выходного слоя образуют линейную комбинацию выходов нейронов скрытого слоя.

Пример 1.6. Применение радиальной сети для реализации логической операции "ИСКЛЮЧАЮЩЕЕ ИЛИ". Определить выходы скрытого слоя. Задано: опорные точки $\mathbf{c}_1 = [1 \ 1]^T$; $\mathbf{c}_2 = [0 \ 0]^T$; $2\sigma^2 = 1$, вектор входа $\mathbf{r}^T = [0 \ 1]$.

Расстояние вектора входа от центра \mathbf{c}_1 : $s_1^{(1)} = [(r_1 - c_{11})^2 + (r_2 - c_{12})^2]^{1/2} = [(0-1)^2 + (1-1)^2]^{1/2} = 1$. Выход первого ИН скрытого слоя $q_1^{(1)} = f(s_1^{(1)}) = e^{-1} = 0.368$. Расстояния вектора входа от центра \mathbf{c}_2 $s_2^{(1)} = [(0-0)^2 + (1-0)^2]^{1/2} = 1$. Выход второго ИН скрытого слоя: $q_2^{(1)} = f(s_2^{(1)}) = e^{-1} = 0.368$. Результаты расчетов приведены в табл. 1.2.

Таблица 1.2

\mathbf{r}^T	$q_1^{(1)}$	$q_2^{(1)}$	$q^{(2)}$
[0 1]	0.368	0.368	1
[1 0]	0.368	0.368	1
[0 0]	0.135	1	0
[1 1]	1	0.135	0

Входные векторы 2-го слоя линейно разделимы (рис. 1.22), линейного ИН выходного слоя можно сформировать требуемую функцию.

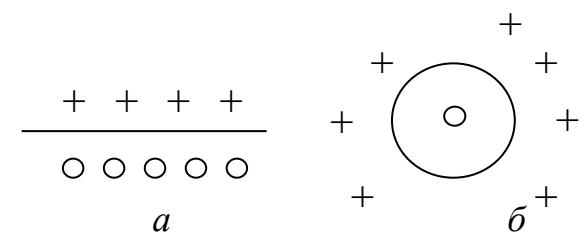


Рис. 1.21

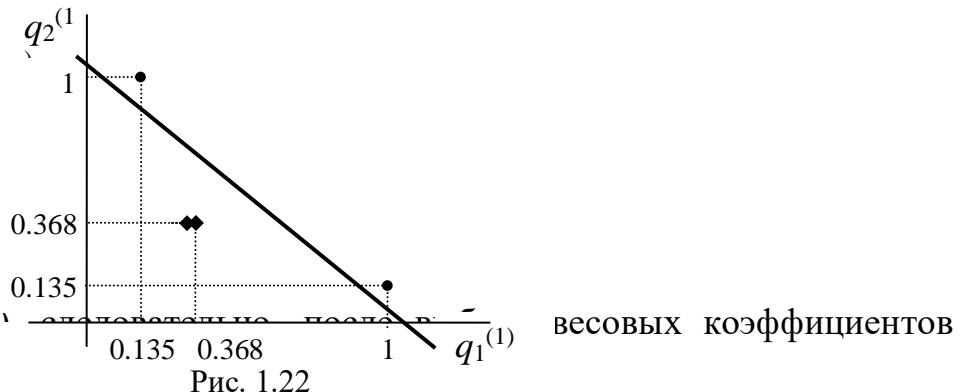


Рис. 1.22

весовых коэффициентов

1.10. Автоассоциативная сеть Хопфилда

Отдельные информационные единицы, хранимые сетью, называются *образами* (образцами). Главная задача ассоциативной памяти сводится к запоминанию входных (обучающих) выборок таким образом, чтобы при предоставлении новой выборки сеть смогла сформировать ответ: какая из запомненных ранее выборок наиболее близка к вновь поступившему образу.

Наиболее часто в качестве меры близости отдельных множеств применяется *мера Хемминга*.

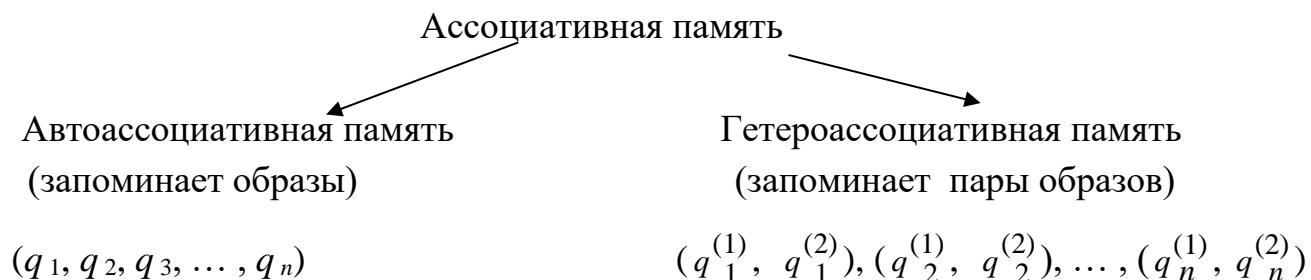


Рис. 1. 23

Сеть Хопфилда – упрощенная модель памяти человека, является автоассоциативной памятью (в Matlab функция newhop). Задача сети – из произвольного неидеального сигнала, поданного на ее вход, восстановить соответствующий образ (если такой есть) или "дать заключение" о том, что входные данные не соответствуют ни одному из образ.

В общем случае, любой сигнал может быть описан вектором $r = \{r_i : i = 1..n\}$, n – число нейронов в сети и размерность входных и выходных векторов. Каждый элемент r_i равен либо +1, либо -1.

Пусть вектор, описывающий k -ый образ, обозначен \mathbf{r}^k , компоненты вектора – r_i^k , $k=1\dots m$, где m – число образов.

Если сеть распознает образ на основе предъявленных искаженных данных, то вектор выхода будет равен $\mathbf{q} = \mathbf{r}^k$, где

$\mathbf{q} = \{q_i : i = 1\dots n\}$, иначе выходной вектор \mathbf{q} не совпадет ни с одним образом.

На рис. 1.24 z^{-1} – оператор задержки; \mathbf{r} – вектор начального импульсного воздействия $r_i(k) \neq 0$ при $k = 0$, $r_i(k) = 0$ при $k \neq 0$, $i = \overline{1, n}$.

Каждый вход r_i подается только на соответствующий i -й ИН. Вектор \mathbf{r} обеспечивает начальную установку выходного вектора для $k = 0$: $q_i(0) = r_i(0)$.

После подачи на вход вектора \mathbf{r} сигнал с выхода нейрона подается на входы всех остальных нейронов, вектор начинает циркулировать по сети, пока не

придет в установившееся состояние, при котором нейрон на каждом следующем цикле вырабатывает тот же сигнал, что и на предыдущем.

Возможна циркуляция входного вектора без установившегося состояния.

(На рисунке 1.25 изменена индексация весовых коэффициентов: 1-й индекс – номер выхода нейрона, 2-й – номер входа нейрона после обратной связи.)

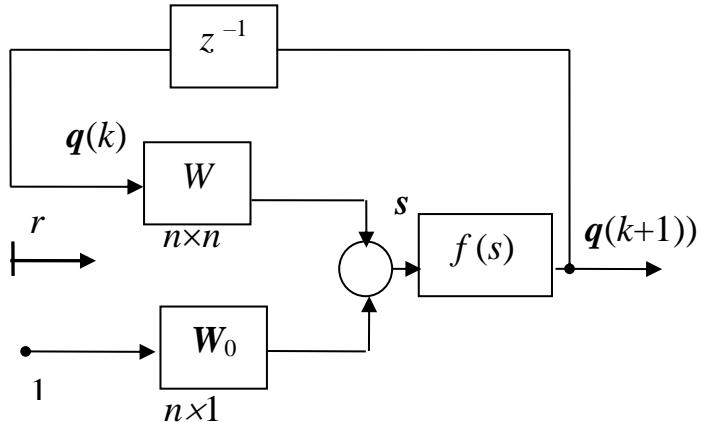


Рис. 1.24

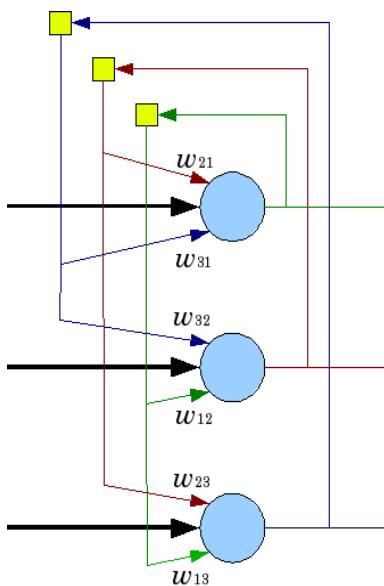


Рис. 1.25

Без учета z^{-1} , вектор выхода сети

$$\mathbf{q}(k+1) = f(W\mathbf{q}(k) + \mathbf{W}_0) \quad (1.13)$$

расширенный выход

$$\tilde{\mathbf{q}}(k+1) = f(\tilde{W}\tilde{\mathbf{q}}(k)) . \quad (1.14)$$

Динамическое изменение состояния сети может быть выполнено синхронно и асинхронно.

При **синхронном режиме** на каждом шаге модифицируются все ИН (в многослойных сетях модифицируются нейроны одного слоя).

Обновление состояния ИН в сети Хопфилда выполняется с использованием следующего правила:

$$q_i = \begin{cases} +1, & \text{если } \sum_j w_{ij} q_j \geq \Theta i \\ -1, & \text{в противном случае} \end{cases} .$$

где Θi – пороговое значение.

В **асинхронном режиме** в момент времени $k + 1$ вычисляется один нейрон, состояние остальных остается без изменений.

Активный нейрон вычисляется следующим образом: $q_i(k+1) = \begin{cases} f(s_i(k+1)), & s_i(k+1) \neq 0; \\ q_i(k), & s_i(k+1) = 0, \quad i = \overline{1, n}. \end{cases}$

Затем выбирается следующий нейрон и процесс повторяется.

Условия устойчивости сети:

1. Симметричность матрицы весовых коэффициентов: $W_{ij} = W_{ji}$.
2. Равенство нулю диагональных элементов: $W_{ii} = 0$.

Поведение сети в пространстве состояний аналогично движению шарика в лунку (рис. 1.27).

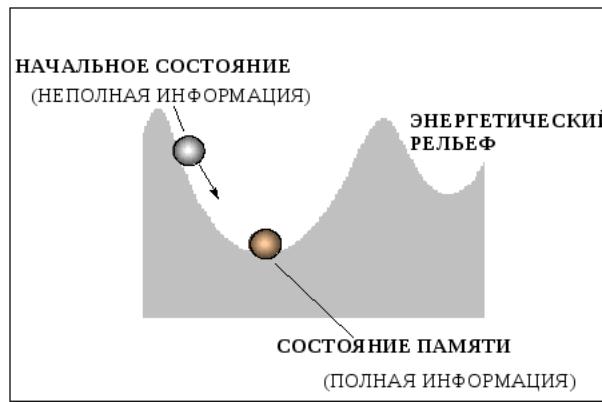


Рис. 1.26

Сеть с *одними и теми же* весовыми коэффициентами может хранить и воспроизводить *несколько* различных образов (эталонов). Каждый эталон – **аттрактор** (стационарное оптимальное состояние), вокруг которого существует **область притяжения** (attract – притягивать). Любая система с несколькими аттракторами – память. Если системе задано начальное

отличное от эталонного состояние, т.е. известна частичная информация об эталоне, и если начальное состояние достаточно близко к эталону, то сеть попадает в область притяжения и начинает двигаться к этому эталону ("вспоминает" его).

Минимумы – устойчивые состояния памяти, точки на склонах – переходными состояниями.

Характер рельефа определяется видом целевой функции E и формируется в процессе обучения сети.

Целевая функция («обобщенная энергия») – $E = W_{ij} q_i q_j = \min, i j .$

Расстояние между состояниями сети измеряется в метрике Хэмминга.

Расстоянием Хэмминга называется число отличающихся битов в двух бинарных векторах. Пусть векторы имеют вид

А = (1, -1, -1, -1, 1) и Б = (1, 1, -1, -1, -1), то Хэммингово расстояние между ними будет равно двум.

Если входные образы представляют изображения, то, отобразив графически данные с выхода сети (в Matlab функция Imagesc), получим либо совпадающий эталонный образ, либо «вольную импровизацию» сети из-за неудачи (рис. 1.27 и 1.28).

В примерах на рисунках включены 10 цифр и 22 буквы (показаны 10), представленных в пиксельной форме размером 9×9.

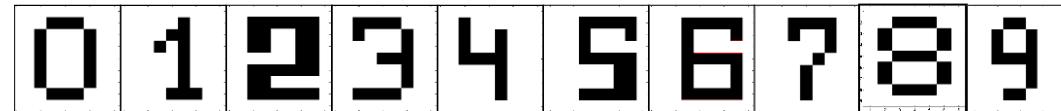
Наибольшее число искаженных пикселей составило 27 %.

Для повышения эффективности распознавания образов необходимо:

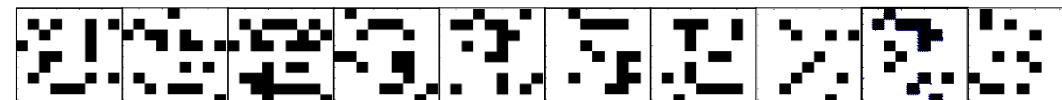
- увеличить размер входного вектора (число пикселей);
- обучать сеть на частично искаженных образах.

Если два образа A и B сильно похожи, они, возможно, будут вызывать у сети перекрестные ассоциации.

ЭТАЛОННЫЕ ОБРАЗЫ



ТЕСТОВЫЕ ОБРАЗЫ



ВОССТАНОВЛЕННЫЕ ОБРАЗЫ



Рис. 1.27

ЭТАЛОННЫЕ ОБРАЗЫ



ТЕСТОВЫЕ ОБРАЗЫ



ВОССТАНОВЛЕННЫЕ ОБРАЗЫ



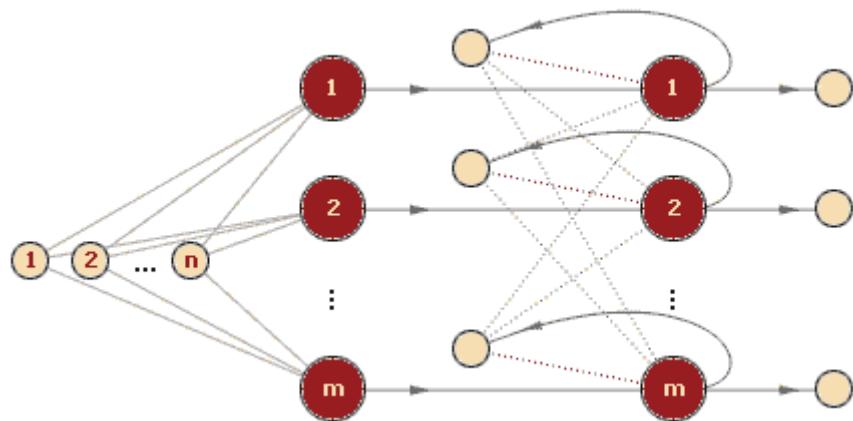
Рис. 1.28

1.11. Сеть Хэмминга

Сеть Хэмминга реализует гетероассоциативную память, при которой достаточно получать только номер образа.

Характеризуется меньшими затратами на память и объемом вычислений по сравнению с сетью Хопфилда.

При обучении на входной слой поступает желаемый обучающий образ, а на выход выходного слоя поступает значение желаемого класса, к которому принадлежит вектор.



Сеть состоит из двух слоев. Первый и второй слои имеют по m нейронов, где m -число образов. Нейроны первого слоя имеют n входов. Нейроны второго слоя связаны между собой обратными отрицательными связями ($-\varepsilon$). Единственная положительная обратная связь с выхода на вход нейрона (+1).

Рис. 1.29

Идея работы сети состоит в нахождении расстояния Хэмминга от тестируемого образа до всех образцов.

Выделяются две фазы.

- 1) После предъявления входного n -вектора r на выходах нейрона 1-го слоя генерируются сигналы, задающие начальные состояния нейронов 2-го слоя. Вычисляется расстояние Хэмминга между входным вектором r и каждым из m закодированных векторов-образов.
- 2) Из сформированного начального состояния нейронов 2-го слоя запускается итерационный процесс внутри слоя, который завершается, когда все нейроны кроме одного (победителя с выходным сигналом, равным 1), перейдут в нулевое состояние. Сеть Хэмминга выбирает образ с минимальным расстоянием Хэмминга до неизвестного входного сигнала, в результате чего будет активизирован только один выход сети, соответствующий этому образцу.

Модификация сети

Нейрон-победитель посредством весов, связывающих его с нейронами выходного слоя, формирует выходной вектор q , соответствующий вектору r .

1.12. Двунаправленная ассоциативная память (Bidirectional Associative Memory, BAM)

Сеть представляет собой гетероассоциативную память, запоминает пары образов

$(r_1, q_1), (r_2, q_2), \dots, (r_n, q_n)$ и позволяет восстанавливать выходной образ поискаженному входному образу (рис. 1.30). Сигналы с выходов 2-го слоя поступают на входы 1-го. Матрица весовых коэффициентов прямого канала W . Матрица весовых коэффициентов обратной связи W^T . (Bidirectional Associative Memory, BAM)

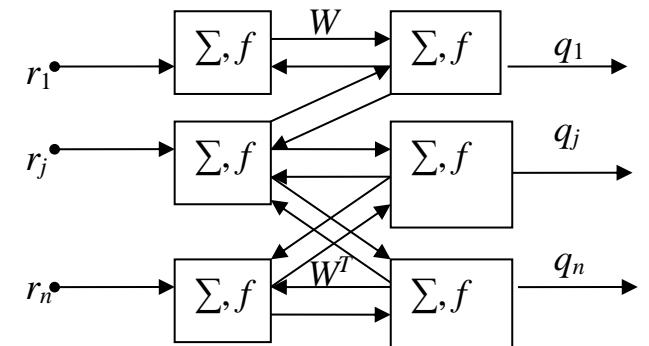
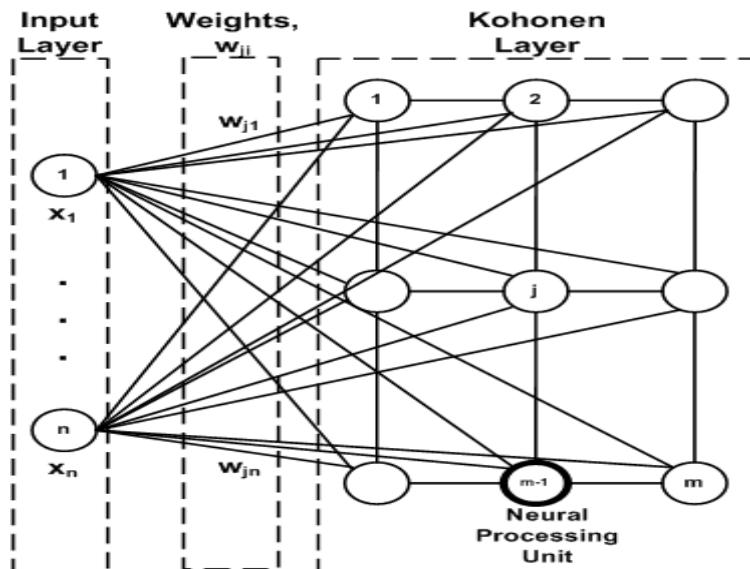


Рис. 1.30

1.12. Кластеризация образов

Кластеризация векторов (образов) – это разделение их на группы по степени схожести. Такие группы называются *кластерами*. Отличие задачи кластеризации от задачи классификации в том, что набор групп изначально не задан и определяется в процессе обучения сети. Образы внутри одного кластера должны быть в некотором смысле подобны (иметь общие признаки). Подобные кластеры должны размещаться близко друг к другу.

Сеть Кохонена. Состоит из входного слоя и слоя Кохонена. Входной слой выполняет функцию распределения сигналов между ИН слоя Кохонена (рис. 1.31). Величина взаимодействия между ИН слоя Кохонена определяется расстоянием между ними. Когда на слой Кохонена подается входной вектор r (x на рисунке 1.31) между ИН начинается "конкуренция", какой из



ИН ближе (в смысле какой-либо меры, например, расстояния) к входному вектору. Сравниваются расстояния между входным вектором и векторами весовых коэффициентов. ИН с минимальным расстоянием – "победитель", его выход равен 1, а выходы остальных равны 0. В качестве меры может быть использован квадрат евклидова расстояния: $s_i = \sum_j (W_{ij} - r_j)^2$ – расстояние

между входным вектором \mathbf{r} и вектором весовых коэффициентов \mathbf{W}_i , $i = \overline{1, m}$, m – число ИН, $j = \overline{1, n}$, n – число входов.

Рис. 1.31

Часто для сравнения нейронов используют скалярное произведение входного вектора и вектора весовых коэффициентов нейрона, сферическое дуговое расстояние $\Delta(\mathbf{w}, \mathbf{r}) = 1 - \mathbf{w}^T \mathbf{r} = 1 - \cos\theta$, где $\mathbf{w}^T \mathbf{r}$ – скалярное произведение, причем предполагается, что $\|\mathbf{w}\| = \|\mathbf{r}\| = 1$. В некоторых модификациях сети вычисляется взвешенная сумма элементов входного вектора, аналогично (1.1). В этом случае победителем считается ИН слоя Кохонена с максимальным значением взвешенной суммы. В результате обучения формируется набор векторов \mathbf{W}_i , каждый из которых указывает на центр кластера.

Самоорганизующиеся карты Кохонена (Self Organizing Maps – SOM). Центры кластеров упорядочены в некоторую структуру – обычно двумерные сетки с прямоугольными или шестиугольными ячейками. При обучении изменяется не только нейрон-победитель, но и его соседи (рис.1.32).

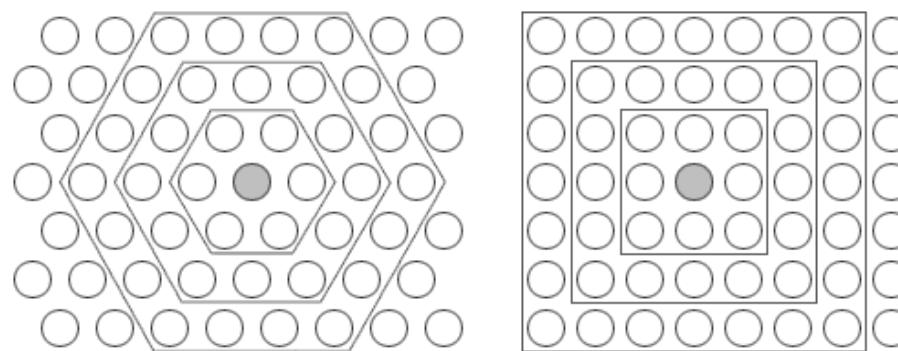


Рис. 1.32. Расположение нейронов в узлах двумерной сетки с четырехугольными и шестиугольными ячейками

Применение: сжатие информации; обработка изображений; классификация образов; тематический анализ коллекций документов; построение репрезентативной выборки.

Пример 1.7. Разделение цветов на кластеры (рис. 1.33).

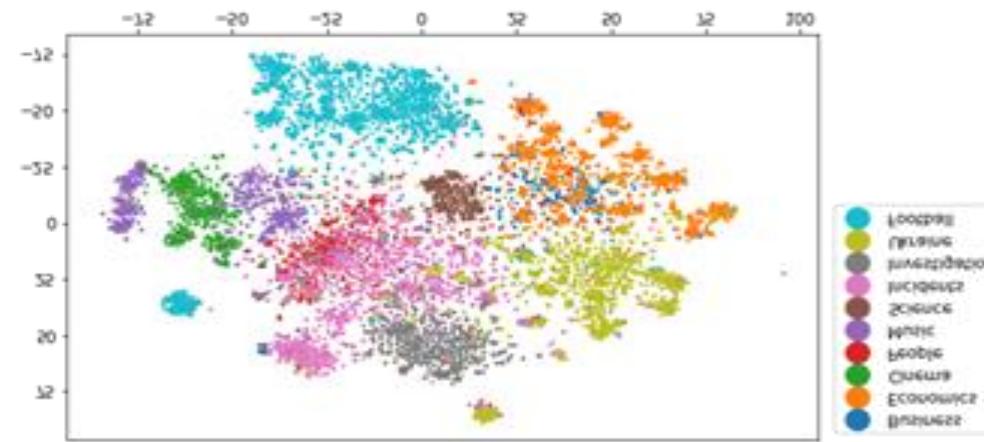
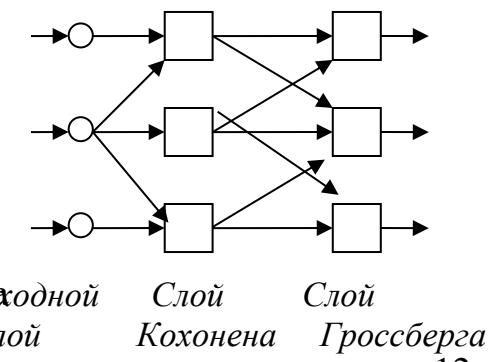


Рис. 1.33

1.13. Сети встречного распространения

Нейронные сети встречного распространения относятся к классу каскадных сетей, содержат два слоя с последовательными связями: слой Кохонена и слой Гроссберга (рис. 1.34). Слой Гроссберга – с линейными функциями активации, входы нейронов слоя Гроссберга являются взвешенной суммой выходов слоя Кохонена. Вначале на множестве входных векторов обучается слой Кохонена. Если сеть функционирует таким образом, что выход ИН "победителя" равен единице, а выходы остальных ИН равны нулю, то выход ИН слоя Гроссберга



определяется только величиной весового коэффициента связи с ИН-"победителем".

Точность сети будет ограничена. Повышение точности обеспечивается формированием целой группы нейронов слоя Кохонена, имеющих максимальные выходы, которые передают свои выходные сигналы в слой Гроссберга. Число нейронов в такой группе определяется экспериментально в зависимости от решаемой задачи.

Рис. 1.34

2. АЛГОРИТМЫ ОБУЧЕНИЯ

Для построения сети необходимо выполнить:

- ✚ формирование архитектуры;
- ✚ настройку весовых коэффициентов сети.

Формирование архитектуры сети

Количество нейронов во входном и выходном слоях определяется размерностью входного вектора и ожидаемого выходного вектора. Задача – выбор количества скрытых слоев и числа нейронов в слоях (рис. 2.1).

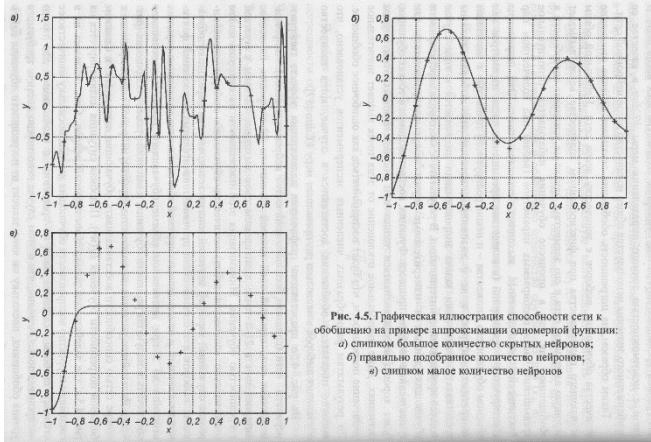
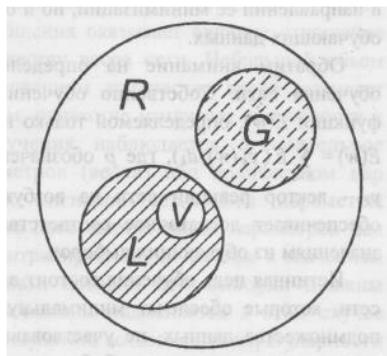


Рис. 2.1

Для определения *весовых коэффициентов* выделяют данные (рис. 1.36):

- ❖ для обучения L (обучающая выборка содержит часть возможных вариантов входных сигналов);
- ❖ для верификации степени обучения сети V ;
- ❖ тестовые данные G .



Задача обучения – получение правильной реакции на любой допустимый входной сигнал G , т. е. *приобретение способности к обобщению*.

Элементы в L и G – типичные данные множества R .

Данные обучающей выборки задают опорные точки в пространстве входов – выходов сети. Цель обучения состоит в выполнении правильной интерполяции между опорными точками.

Рис. 2.2

Способность сети к обучению и обобщению полученных результатов – черты искусственного интеллекта.

Способы определения весовых коэффициентов:

1) *обучение с "учителем"*. Сеть располагает правильными ответами – выходами сети на каждый входной пример (образ).

Веса настраиваются так, чтобы сеть производила ответы, по-возможности, более близкие к правильным;

2) *обучение без "учителя"*. Не требуется знание ответов, результат зависит только от структуры входных данных;

3) часть весовых коэффициентов обучается с "учителем", часть – без "учителя";

4) расчет весовых коэффициентов без обучения.

Свойства обучения:

1) *ёмкость сети* – количество образов, которые может запомнить сеть, и количество функций, которые могут быть сформированы;

2) *сложность образов* – число обучающих примеров, необходимых для приобретения сетью способности к обобщению;

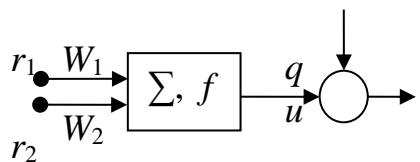
3) *вычислительная сложность* – сложность реализации.

2.1. Настойка весовых коэффициентов ИН

Пусть используется ИН с линейной функцией активации (1.3), включающий два входа, $W_0 = 0$ (рис. 2.3).

Используем принцип обучения с "учителем".

Получим зависимость ошибки обучения от весовых коэффициентов $E = f(W_1, W_2)$.



Пусть ошибка обучения: $E = \frac{1}{2}(u - q)^2$, где u , q – желаемое и действительное значение выходного сигнала нейрона. Весовые коэффициенты настраиваются так, чтобы обеспечить минимум ошибки обучения.

Выход ИН $q = s = \sum_{j=1}^2 W_j r_j = W_1 r_1 + W_2 r_2$, тогда ошибка обучения

$$E = \frac{1}{2}(u^2 - 2u(W_1 r_1 + W_2 r_2) + W_1^2 r_1^2 + 2W_1 r_1 W_2 r_2 + W_2^2 r_2^2) = f(W_1^2, W_2^2).$$

Продифференцировав по W_1 , получим: $\frac{dE}{dW_1} = [-u + W_1 r_1 + W_2 r_2]r_1$. Приравняв к нулю правую часть уравнения, можно

найти минимум ошибки.

На рис. 2.4 изображены кривая и поверхность, представляющие величину ошибки обучения для разных комбинаций весовых коэффициентов.

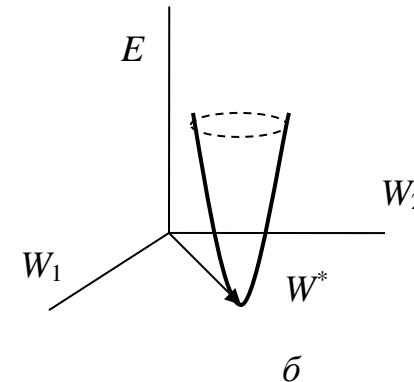
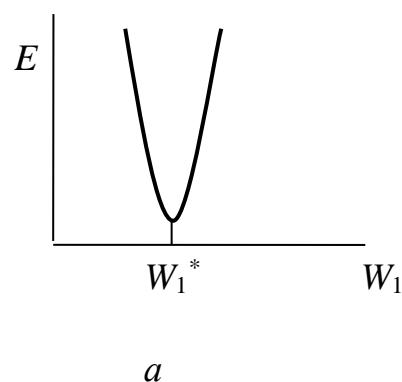


Рис. 2.4

Минимум ошибки обучения обеспечивают весовой коэффициент W_1^* (при заданном коэффициенте W_2) и вектор \mathbf{W}^* .

2.2. Дельта-правило (Видроу–Хоффа)

Дельта-правило – алгоритмом обучения градиентного типа:

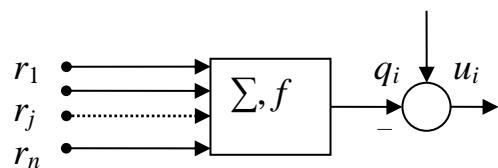
$$\Delta \mathbf{W} = -\gamma \nabla_{\mathbf{W}} E = -\gamma \frac{\partial E}{\partial \mathbf{W}}, \quad (2.1)$$

где γ – постоянный положительный коэффициент обучения.

Правило, основанное на принципе "обучение с учителем", формулируется как

$$\Delta W_{ij} = \gamma \delta_i r_j, \quad (2.2)$$

где $\delta_i = u_i - q_i$; u_i , q_i – желаемый и действительный выходы i -го ИН (рис. 2.5); r_j – сигнал, поступающий от j -го входа.



Новый вектор весовых коэффициентов определяется в виде

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \Delta \mathbf{W}(k).$$

Пусть ошибка обучения p -й обучающей выборки: $E_p = \frac{1}{2} \sum_i (u_i - q_i)^2$.

Рис. 2.5

Суммарная ошибка равняется p обучающих выборок $E = \sum_p E_p$.

Частный случай, пусть нейрон содержит линейную функцию активации (1.3).

Согласно (2.1)

$$\Delta W_{ij} = -\gamma \frac{\partial E_p}{\partial W_{ij}}. \quad (2.3)$$

Возьмём частную производную:

$$\frac{\partial E_p}{\partial W_{ij}} = \frac{\partial E_p}{\partial q_i} \frac{\partial q_i}{\partial W_{ij}}. \quad (2.4)$$

Обозначим:

$$\frac{\partial E_p}{\partial q_i} = \frac{\partial [1/2(u_i - q_i)^2]}{\partial q_i} = -(u_i - q_i) = -\delta_i. \quad (2.5)$$

При наличии линейной функции активации выход нейрона равен взвешенной сумме $q_i = s_i = \sum_j W_{ij} r_j$, следовательно, получим

$\frac{\partial q_i}{\partial W_{ij}} = r_j$. Подставив полученные результаты в (2.4), определим значение производной: $\frac{\partial E_p}{\partial W_{ij}} = -\delta_i r_j$.

В результате подстановки в (2.3) получим $\Delta W_{ij} = \gamma \delta_i r_j$.

2.3. Алгоритм обратного распространения ошибок

Алгоритм является обобщением дельта-правила. В алгоритме учитываются два потока: от входного слоя к выходному (прямой); от выходного слоя к входному (обратный) (рис. 2.6). В обратном потоке ИН скрытого слоя получает сигналы ошибок от каждого нейрона выходного слоя.

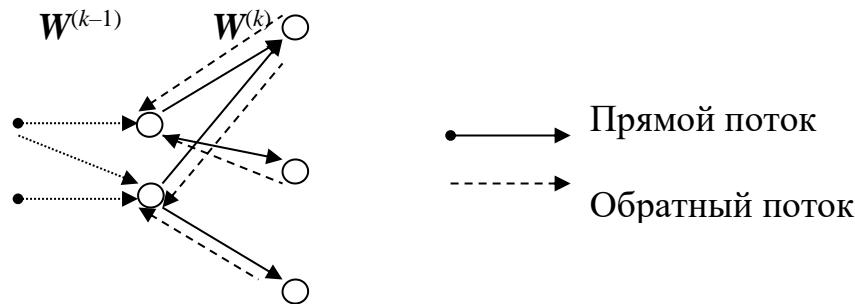


Рис. 2.6

С учетом (2.3) ($\Delta W_{ij} = -\gamma \frac{\partial E_p}{\partial W_{ij}}$) определим производную ошибки обучения для общего случая – ИН с нелинейной

функцией активации:

$$\frac{\partial E_p}{\partial W_{ij}} = \frac{\partial E_p}{\partial q_i} \frac{\partial q_i}{\partial s_i} \frac{\partial s_i}{\partial W_{ij}}. \quad (2.6)$$

Введем обозначение для производной ошибки:

$$\frac{\partial E_p}{\partial s_i} = -\delta_i. \quad (2.7)$$

Уравнение (2.7) согласовано с (2.5): $\frac{\partial E_p}{\partial q_i} = -\delta_i$, так как (2.5) получено для ИН с линейной функцией активации.

Уравнение (2.7) можно переписать в виде $\frac{\partial E_p}{\partial s_i} = \frac{\partial E_p}{\partial q_i} \frac{\partial q_i}{\partial s_i}$.

Раньше было получено (см. 2.5): $\frac{\partial E_p}{\partial q_i} = -(u_i - q_i)$.

Выход ИН $q_i = f(s_i)$, поэтому $\frac{\partial q_i}{\partial s_i} = f'(s_i)$. Тогда с учетом (2.7) получим $\frac{\partial E_p}{\partial s_i} = -(u_i - q_i)f'(s_i)$, следовательно,

$$\delta_i = (u_i - q_i)f'(s_i). \quad (2.8)$$

С учетом выражения для взвешенной суммы: $s_i = \sum_j W_{ij}r_j$, получим $\frac{\partial s_i}{\partial W_{ij}} = r_j$. В результате определим (2.6) в виде

$$\frac{\partial E_p}{\partial W_{ij}} = -(u_i - q_i)f'(s_i)r_j = -\delta_i r_j. \quad (2.9)$$

После подстановки (2.9) в (2.3): $\Delta W_{ij} = \gamma \delta_i r_j$.

Алгоритм коррекции весовых коэффициентов

$$W_{ij}(k+1) = W_{ij}(k) + \gamma(u_i - q_i)f'(s_i)r_j = W_{ij}(k) + \gamma\delta_i r_j.$$

Уравнение (2.8) соответствует ошибке обучения ИН выходного слоя $\delta_i = \delta_i^{(k)}$.

Ошибка ИН скрытого слоя непосредственно не связана с целевым выходом u_i , поэтому весовые коэффициенты ИН скрытого слоя следует корректировать пропорционально "вкладу" нейрона в величину ошибки нейронов следующего слоя.

Для i -го нейрона l -го скрытого слоя ошибка

$$\delta_i^{(l)} = f_i'^{(l)}(s_i) \sum_j W_{ij}^{(l+1)} \delta_j^{(l+1)}, \quad (2.10)$$

где $\sum_j W_{ij}^{(l+1)} \delta_j^{(l+1)}$ – вклад i -го ИН l скрытого слоя в ошибку на выходе $l+1$.

Вектор ошибки l -го скрытого слоя: $\boldsymbol{\delta}^{(l)} = f'(s)^{(l)} W^{(l+1)T} \boldsymbol{\delta}^{(l+1)}$.

Замечание. Следует выбирать число образцов обучающей выборки N равным $N > n / e$, где n – число настраиваемых весовых коэффициентов, e – допустимая ошибка.

Пример 2.1. Вычислить прямой и обратный проходы сети прямого распространения $N_{2,2,1}^2$, используя алгоритм обратного распространения ошибок. Вход $\mathbf{r} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$, выход $u = 0.9$. Расширенные матрицы весовых коэффициентов:

$$\tilde{W}^{(1)} = \begin{bmatrix} 0.1 & -0.2 & 0.1 \\ 0.1 & -0.1 & 0.3 \end{bmatrix}, \quad \tilde{W}^{(2)} = [0.2 : 0.2 \ 0.3].$$

Логистическая функция активации (1.5): $f(s) = \frac{1}{1 + e^{-s}}$. Производная функции активации $f'(s) = f(s)(1 - f(s)) = q(1 - q)$.

Прямой проход:

1) Взвешенная сумма 1-го ИН скрытого слоя

$$s_1^{(1)} = \sum_{j=0}^2 W_{1j}^{(1)} r_j = 0.1 \cdot 1 - 0.2 \cdot 0.1 + 0.1 \cdot 0.9 = 0.17.$$

2) Выход 1-го ИН скрытого слоя

$$q_1^{(1)}(s) = \frac{1}{1 + e^{-s}} = \frac{1}{1 + e^{-0.17}} = 0.542.$$

3) Взвешенная сумма 2-го ИН скрытого слоя

$$s_2^{(1)} = \sum_{j=0}^2 W_{2j}^{(1)} r_j = 0.1 \cdot 1 - 0.1 \cdot 0.1 + 0.3 \cdot 0.9 = 0.36.$$

4) Выход 2-го ИН скрытого слоя

$$q_2^{(1)}(s) = \frac{1}{1 + e^{-s}} = \frac{1}{1 + e^{-0.36}} = 0.589.$$

5) Взвешенная сумма ИН выходного слоя

$$s_1^{(2)} = \sum_{j=0}^2 W_{1j}^{(2)} q_j^{(1)} = 0.2 \cdot 1 + 0.2 \cdot 0.542 + 0.3 \cdot 0.589 = 0.485.$$

6) Выход ИН выходного слоя

$$q_1^{(2)}(s) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-0.485}} = 0.619.$$

Обратный проход:

1) Ошибка ИН выходного слоя (2.8)

$$\delta_1^{(2)} = \delta^{(2)} = (u - q^{(2)})q^{(2)}(1 - q^{(2)}) = (0.9 - 0.619) \cdot 0.619(1 - 0.619) = 0.066.$$

2) Вклад 1-го ИН скрытого слоя в ошибку на выходе

$$z_1^{(1)} = \sum_j W_{1j}^{(2)} \delta_j^{(2)} = W_{11}^{(2)} \delta^{(2)} = 0.2 \cdot 0.066 = 0.0132.$$

3) Ошибка 1-го ИН скрытого слоя (2.10)

$$\delta_1^{(1)} = q_1^{(1)}(1 - q_1^{(1)})z_1^{(1)} = 0.542(1 - 0.542)0.0132 = 0.003.$$

4) Вклад 2-го ИН скрытого слоя в ошибку на выходе

$$z_2^{(1)} = \sum_j W_{2j}^{(2)} \delta_j^{(2)} = W_{21}^{(2)} \delta^{(2)} = 0.9 \cdot 0.066 = 0.0198.$$

5) Ошибка 2-го ИН скрытого слоя

$$\delta_2^{(1)} = q_2^{(1)}(1 - q_2^{(1)})z_2^{(1)} = 0.589(1 - 0.589)0.0198 = 0.005.$$

Для элемента смещения ошибка не вычисляется, так как ИН входного слоя с элементом смещения не связаны.

Изменение весовых коэффициентов выходного слоя:

$$1) \Delta W_{ij} = \gamma \delta_i^{(k)} q_j^{(k-1)} = \gamma(u_i - q_i^{(k)})f'(k)(s_i)q_j^{(k-1)},$$

$$\Delta W_{11}^{(2)} = \gamma \delta^{(2)} q_1^{(1)} = 0.25 \cdot 0.066 \cdot 0.542 = 0.09,$$

$$W_{11}^{(2)}(k+1) = W_{11}^{(2)}(k) + \Delta W_{11}^{(2)} = 0.2 + 0.009 = 0.209;$$

$$2) \Delta W_{12}^{(2)} = \gamma \delta^{(2)} q_2^{(1)} = 0.25 \cdot 0.066 \cdot 0.589 = 0.01,$$

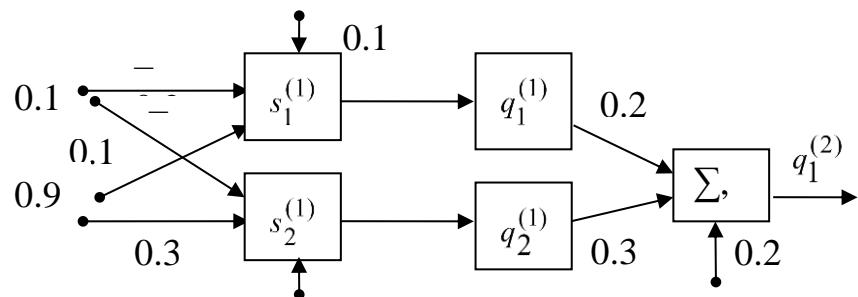
$$W_{12}^{(2)}(k+1) = W_{12}^{(2)}(k) + \Delta W_{12}^{(2)} = 0.3 + 0.01 = 0.31;$$

$$3) \Delta W_{10}^{(2)} = \gamma \delta^{(2)} q_0^{(1)} = 0.25 \cdot 0.066 \cdot 1 = 0.017,$$

$$W_{10}^{(2)}(k+1) = W_{10}^{(2)}(k) + \Delta W_{10}^{(2)} = 0.2 + 0.017 = 0.217.$$

Результаты расчета приведены на рис. 2.7.

Прямой проход



Обратный проход

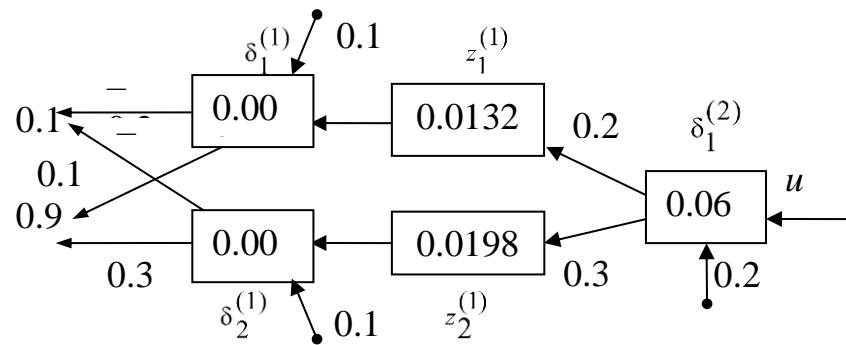


Рис. 2.7

Недостатки алгоритма обратного распространения ошибок:

- Невысокая скорость сходимости, большое число требуемых итераций для достижения критерия обучения – минимума ошибки обучения.
- Возможность достижения не глобального, а локального минимума.
- Возможность "паралича" сети.

Как правило, используются следующие сигмоидные функции активации:

- логистическая функция (1.5): $f(s) = \frac{1}{1 + e^{-s}}$;

- гиперболический тангенс (1.6): $f(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$.

При установке начальных значений весовых коэффициентов (начальная инициализация) множество всех ИН разбивается на подмножества слабоактивируемых при $|s| > a$, $f'(s) \approx 0$ и сильноактивируемых при $|s| < a$, $f'(s) \neq 0$.

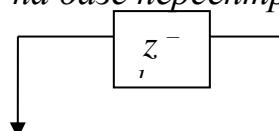
Поскольку согласно дельта-правилу $\Delta W_{ij} = \gamma \delta_i r_j$ ошибка ИН выходного слоя $\delta_i^{(k)} = (u_i - q_i) f'(s_i)$, ошибка ИН скрытого слоя

$$\delta_i^{(k-1)} = f'(s_i)^{(k-1)} \sum_i W_{ij}^{(k)} \delta_j^{(k)}, \text{ то коррекция весовых коэффициентов } \Delta W_{ij} \approx f'(s_i).$$

В силу произвольного выбора начальных коэффициентов возможна ситуация, когда при поступлении на вход вектора r почти все нейроны будут слабоактивируемыми ("паралич сети"). Для выхода из такой ситуации существуют только эвристические методы.

Алгоритм обучения рекуррентных сетей на базе персептрона. Алгоритм обучения – модификация алгоритма обратного распространения ошибок. Модификация опирается на утверждение [5], что для любой рекуррентной сети существует сеть с прямой связью, имеющая идентичное поведение (рис. 2.7). Благодаря элементам задержки поток сигналов может считаться односторонним.

Рекуррентная сеть
на базе персептрона



Развернутая сеть, имитирующая
поведение сети для двух шагов времени

Рис. 2.7

Динамическая сеть рекуррентного многослойного персептрана характеризуется запаздыванием входных и выходных сигналов, объединяемых во входной вектор сети [4]. Выходной сигнал предыдущего временного цикла рассматривается как априори заданный, который просто увеличивает размер входного вектора сети r .

Рассматривается сеть (рис. 2.8), имеющая один входной и один выходной ИН. На рисунке обозначено: m – число элементов задержки входного сигнала; n – число элементов задержки выходного сигнала; n_1 – число нейронов скрытого слоя.

Подаваемый на вход сети вектор r имеет вид

$$\tilde{r}(k) = [1 : r(k) \ r(k-1) \dots r(k-m) \ q(k-1) \dots q(k-n)]^T.$$

Сеть реализует отображение

$$q(k+1) = F(r(k), r(k-1), \dots, r(k-m), q(k-1), \dots, q(k-n)).$$

Взвешенная сумма i -го ИН скрытого слоя $s_i^{(1)} = \sum_{j=0}^{m+p} W_{ij}^{(1)} r_j$.

Выход ИН скрытого слоя $q^{(1)} = f(s^{(1)})$.

Взвешенная сумма ИН выходного слоя $s^{(2)} = \sum_{j=0}^{n_1} W_j^{(2)} q_j^{(1)}$.

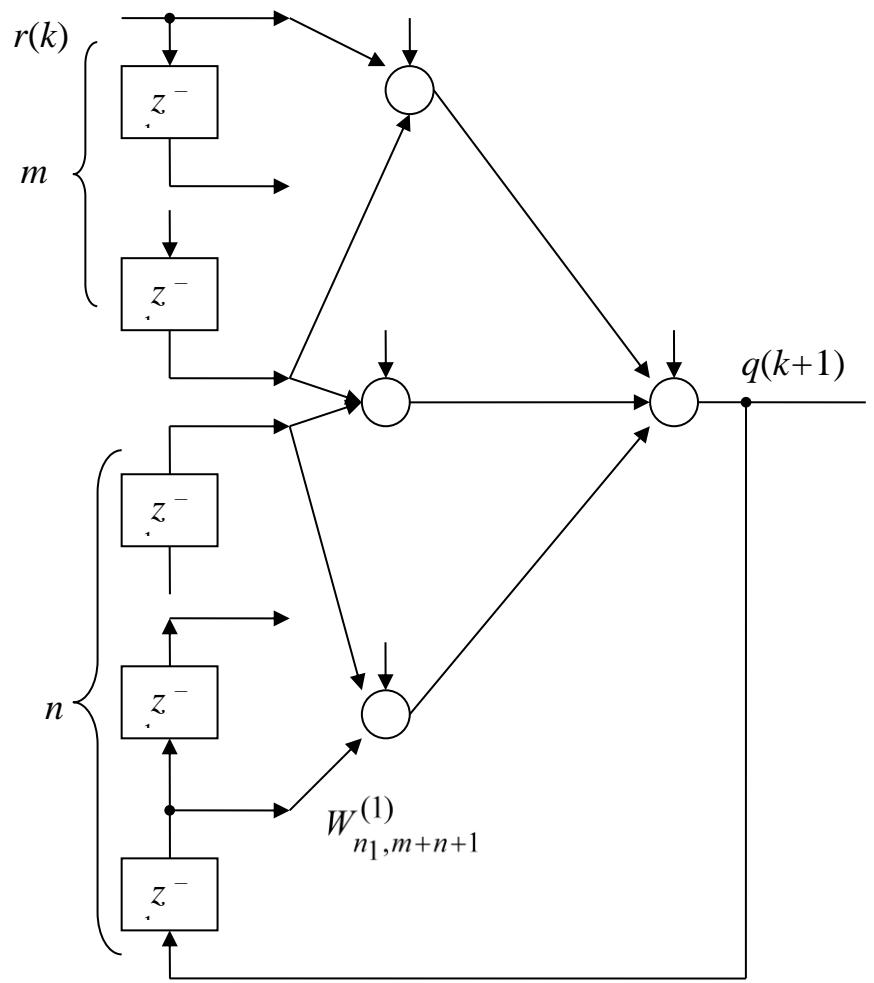


Рис. 2.8

Взвешенная сумма ИН выходного слоя $s^{(2)} = \sum_{j=0}^{n_1} W_j^{(2)} q_j^{(1)}$.

Выход сети $q = q^{(2)} = f(s^{(2)})$.

Коррекция весовых коэффициентов i -го ИН выходного слоя:

$$\Delta W_i^{(2)} = \gamma(u(k) - q(k)) \frac{dq(k)}{dW_i^{(2)}(k)},$$

где $(u(k) - q(k))$ – динамическая ошибка. Производная в текущий момент времени рассчитывается с использованием её значений в предыдущие моменты времени.

Коррекция весовых коэффициентов скрытого слоя:

$$\Delta W_{ij}^{(1)} = \gamma(u(k) - q(k)) \frac{dq(k)}{dW_{ij}^{(1)}(k)}.$$

Алгоритм функционирует в реальном масштабе времени, принимая поступающие входные данные, соответствующие им желаемые значения выходов и оперативно корректируя значения весовых коэффициентов. При использовании нейронной сети в качестве регулятора аргумент алгоритма обучения включает переменные состояния объекта управления.

Алгоритм обратного распространения ошибки (Backpropagation) – алгоритмом обучения градиентного типа

$$\Delta \mathbf{W} = -\gamma \nabla_{\mathbf{W}} E = -\gamma \frac{\partial E}{\partial \mathbf{W}},$$

где $E = \frac{1}{2} \sum_i (u_i - q_i)^2$ – ошибка обучения; u_i, q_i – желаемый и действительный выходы i -го нейрона; γ – положительный коэффициент обучения.

1. Галушкин А. И. Синтез многослойных систем распознавания образов. — М.: «Энергия», 1974.
2. Werbos P. J., Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, Cambridge, MA, 1974.

Изменение весового коэффициента между i -ым нейроном k слоя и j -ым нейроном $(k-1)$ слоя

$$\Delta W_{ij} = \gamma \delta_i^{(k)} q_j^{(k-1)} = \gamma (u_i - q_i^{(k)}) f'^{(k)}(s_i) q_j^{(k-1)}$$

где $\delta_i = (u_i - q_i) f'(s_i)$ – ошибка i -го нейрона выходного слоя;

$\delta_i^{(l)} = f_i'^{(l)}(s_i) \sum_j W_{ij}^{(l+1)} \delta_j^{(l+1)}$ – ошибка i -го нейрона l -го скрытого слоя.

,

.

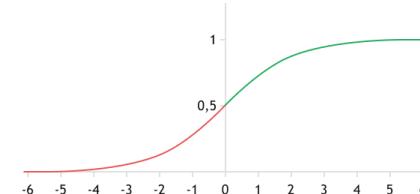
Пример 2.1. Вычислить прямой и обратный проходы сети прямого распространения $N_{2,2,1}^2$, используя алгоритм

обратного распространения ошибок. Вход $\mathbf{r} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$, выход $u = 0.9$, $\gamma = 0.25$. Расширенные матрицы весовых коэффициентов после инициализации сети:

$$\tilde{W}^{(1)} = \begin{bmatrix} 0.1 & -0.2 & 0.1 \\ 0.1 & -0.1 & 0.3 \end{bmatrix}, \tilde{W}^{(2)} = [0.2 : 0.2 \ 0.3].$$

Логистическая функция активации (1.5): $f(s) = \frac{1}{1 + e^{-s}}$.

Производная функции активации $f'(s) = f(s)(1 - f(s)) = q(1 - q)$.



Прямой проход:

1) Взвешенная сумма 1-го нейрона скрытого слоя

$$s_1^{(1)} = \sum_{j=0}^2 W_{1j}^{(1)} r_j = 0.1 \cdot 1 - 0.2 \cdot 0.1 + 0.1 \cdot 0.9 = 0.17.$$

2) Выход 1-го нейрона скрытого слоя

$$q_1^{(1)}(s) = \frac{1}{1 + e^{-s}} = \frac{1}{1 + e^{-0.17}} = 0.542.$$

3) Взвешенная сумма 2-го нейрона скрытого слоя

$$s_2^{(1)} = \sum_{j=0}^2 W_{2,j}^{(1)} r_j = 0.1 \cdot 1 - 0.1 \cdot 0.1 + 0.3 \cdot 0.9 = 0.36.$$

4) Выход 2-го ИН скрытого слоя

$$q_2^{(1)}(s) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-0.36}} = 0.589.$$

5) Взвешенная сумма нейрона выходного слоя

$$s_1^{(2)} = \sum_{j=0}^2 W_{1,j}^{(2)} q_j^{(1)} = 0.2 \cdot 1 + 0.2 \cdot 0.542 + 0.3 \cdot 0.589 = 0.485.$$

6) Выход нейрона выходного слоя определяется также с использованием логистической функции

$$q_1^{(2)}(s) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-0.485}} = 0.619.$$

Обратный проход:

Изменение весовых коэффициентов выходного слоя:

$$\Delta W_{ij} = \gamma \delta_i^{(k)} q_j^{(k-1)} = \gamma(u_i - q_i^{(k)}) f'^{(k)}(s_i) q_j^{(k-1)}.$$

Здесь

- ошибка нейрона выходного слоя $\delta_i = (u_i - q_i) f'(s_i)$ (2.8);
- производная логистической функции активации $-f'(s) = f(s)(1-f(s)) = q^{(2)}(1-q^{(2)})$.

1) Ошибка нейрона выходного слоя (2.8)

$$\delta_1^{(2)} = \delta^{(2)} = (u - q^{(2)})q^{(2)}(1 - q^{(2)}) = (0.9 - 0.619) \cdot 0.619(1 - 0.619) = 0.066.$$

2) Вклад i -го нейрона l скрытого слоя в ошибку на выходе $l+1$

$$z_i^{(l)} = \sum_j W_{ij}^{(l+1)} \delta_j^{(l+1)}$$

Вклад 1-го ИН скрытого слоя в ошибку на выходе

$$z_1^{(1)} = \sum_j W_{1j}^{(2)} \delta_j^{(2)} = W_{11}^{(2)} \delta^{(2)} = 0.2 \cdot 0.066 = 0.0132.$$

3) Для i -го нейрона l -го скрытого слоя ошибка

$$\delta_i^{(l)} = f_i'^{(l)}(s_i) \sum_j W_{ij}^{(l+1)} \delta_j^{(l+1)},$$

Ошибка 1-го нейрона скрытого слоя (2.10)

$$\delta_1^{(1)} = q_1^{(1)}(1 - q_1^{(1)})z_1^{(1)} = 0.542(1 - 0.542)0.0132 = 0.003.$$

4) Вклад 2-го ИН скрытого слоя в ошибку на выходе

$$z_2^{(1)} = \sum_j W_{2j}^{(2)} \delta_j^{(2)} = W_{21}^{(2)} \delta^{(2)} = 0.9 \cdot 0.066 = 0.0198.$$

5) Ошибка 2-го ИН скрытого слоя

$$\delta_2^{(1)} = q_2^{(1)}(1 - q_2^{(1)})z_2^{(1)} = 0.589(1 - 0.589)0.0198 = 0.005.$$

Для элемента смещения ошибка не вычисляется, так как ИН входного слоя с элементом смещения не связаны.

Изменение весовых коэффициентов выходного слоя:

$$1) \Delta W_{ij} = \gamma \delta_i^{(k)} q_j^{(k-1)} = \gamma(u_i - q_i^{(k)}) f'(s_i) q_j^{(k-1)},$$

$$\Delta W_{11}^{(2)} = \gamma \delta^{(2)} q_1^{(1)} = 0.25 \cdot 0.066 \cdot 0.542 = 0.09,$$

$$W_{11}^{(2)}(k+1) = W_{11}^{(2)}(k) + \Delta W_{11}^{(2)} = 0.2 + 0.009 = 0.209;$$

$$2) \Delta W_{12}^{(2)} = \gamma \delta^{(2)} q_2^{(1)} = 0.25 \cdot 0.066 \cdot 0.589 = 0.01,$$

$$W_{12}^{(2)}(k+1) = W_{12}^{(2)}(k) + \Delta W_{12}^{(2)} = 0.3 + 0.01 = 0.31;$$

$$3) \Delta W_{10}^{(2)} = \gamma \delta^{(2)} q_0^{(1)} = 0.25 \cdot 0.066 \cdot 1 = 0.017,$$

$$W_{10}^{(2)}(k+1) = W_{10}^{(2)}(k) + \Delta W_{10}^{(2)} = 0.2 + 0.017 = 0.217.$$

Результаты расчета приведены на рис. 2.7.

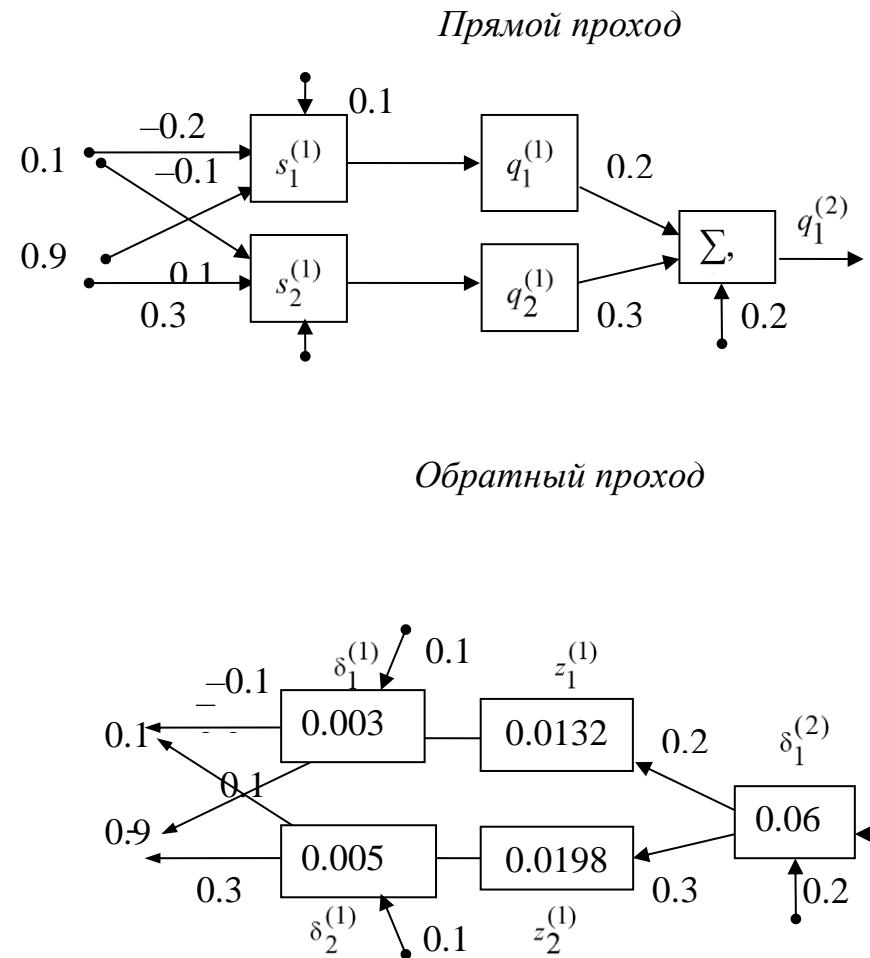


Рис. 2.7

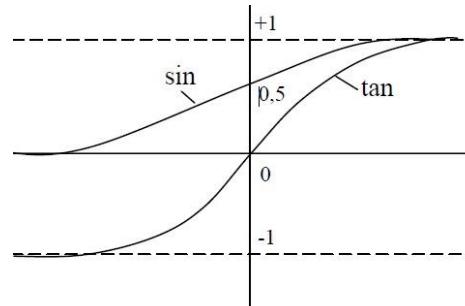
Недостатки алгоритма обратного распространения ошибок:

- Невысокая скорость сходимости, большое число требуемых итераций для достижения критерия обучения – минимума ошибки обучения.

Значение коэффициента γ может оставаться постоянным или подбираться адаптивным способом в зависимости от текущей ошибки обучения. При малом постоянном коэффициенте γ сходимость алгоритма будет невысокой, при большом значении возможна расходимость процесса обучения или попадание в локальные минимумы.

- Возможность достижения не глобального, а локального минимума.
- Возможность "паралича" сети.

Как правило, используются следующие сигмоидные функции активации:



- логистическая функция (1.5): $f(s) = \frac{1}{1 + e^{-s}}$;
- гиперболический тангенс (1.6): $f(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$.

При установке начальных значений весовых коэффициентов (начальная инициализация) множество всех ИН разбивается на подмножества слабоактивируемых при $|s| > a$, $f(s) \approx 0$ и сильноактивируемых при $|s| < a$, $f'(s) \neq 0$.

Поскольку согласно дельта-правилу $\Delta W_{ij} = \gamma \delta_i r_j$ ошибка ИН выходного слоя

$$\delta_i^{(k)} = (u_i - q_i) f'(s_i),$$

ошибка ИН скрытого слоя

$$\delta_i^{(k-1)} = f'(s_i)^{(k-1)} \sum_i W_{ij}^{(k)} \delta_j^{(k)},$$

то коррекция весовых коэффициентов $\Delta W_{ij} \approx f'(s_i)$.

В силу произвольного выбора начальных коэффициентов возможна ситуация, когда при поступлении на вход вектора r почти все нейроны будут слабоактивируемыми ("паралич сети"). Для выхода из такой ситуации существуют только эвристические методы.

Алгоритм обучения рекуррентных сетей на базе персептрана. Алгоритм обучения – модификация алгоритма обратного распространения ошибок. Модификация опирается на утверждение, что для любой рекуррентной сети существует сеть с прямой связью, имеющая идентичное поведение (рис. 2.8). Благодаря элементам задержки поток сигналов может считаться односторонним.

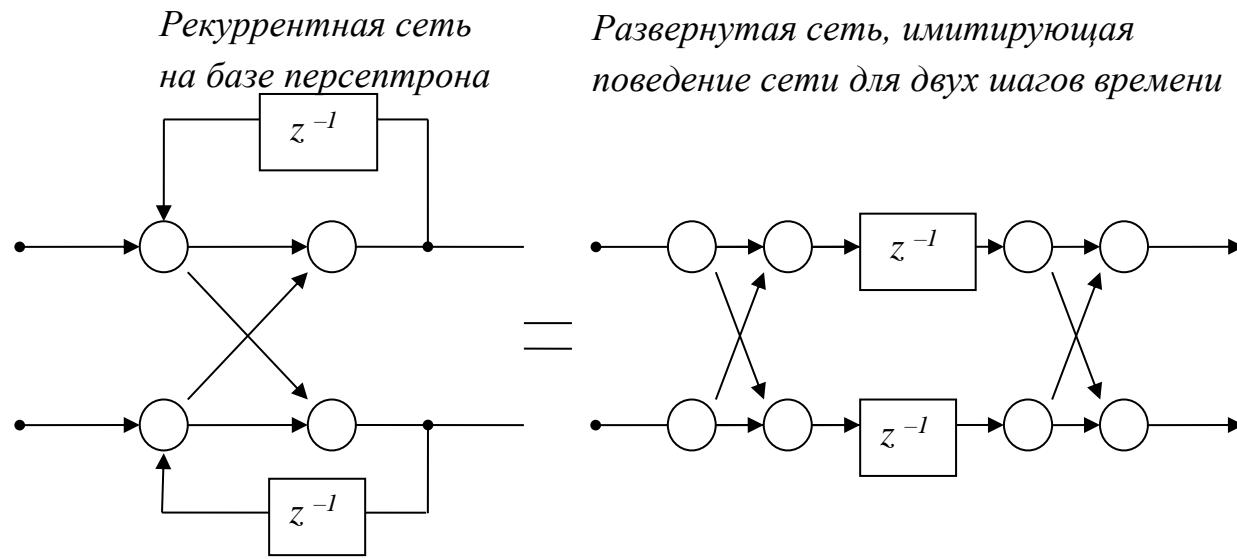


Рис. 2.8

Динамическая сеть рекуррентного многослойного персептрона характеризуется запаздыванием входных и выходных сигналов, объединяемых во входной вектор сети. Выходной сигнал предыдущего временного цикла рассматривается как априори заданный, который просто увеличивает размер входного вектора сети r .

Рассматривается сеть (рис. 2.8), имеющая один входной и один выходной ИН. На рисунке обозначено: m – число элементов задержки входного сигнала; n – число элементов задержки выходного сигнала; n_1 – число нейронов скрытого слоя.

Подаваемый на вход сети вектор r имеет вид

$$\tilde{r}(k) = [1 : r(k) \ r(k-1) \dots r(k-m) \ q(k-1) \dots q(k-n)]^T.$$

Сеть реализует отображение

$$q(k+1) = F(r(k), r(k-1), \dots, r(k-m), q(k-1), \dots, q(k-n)).$$

Взвешенная сумма i -го ИН скрытого слоя $s_i^{(1)} = \sum_{j=0}^{m+n} W_{ij}^{(1)} r_j$.

Выход ИН скрытого слоя $q^{(1)} = f(s^{(1)})$.

Взвешенная сумма ИН выходного слоя $s^{(2)} = \sum_{j=0}^{n_1} W_j^{(2)} q_j^{(1)}$.

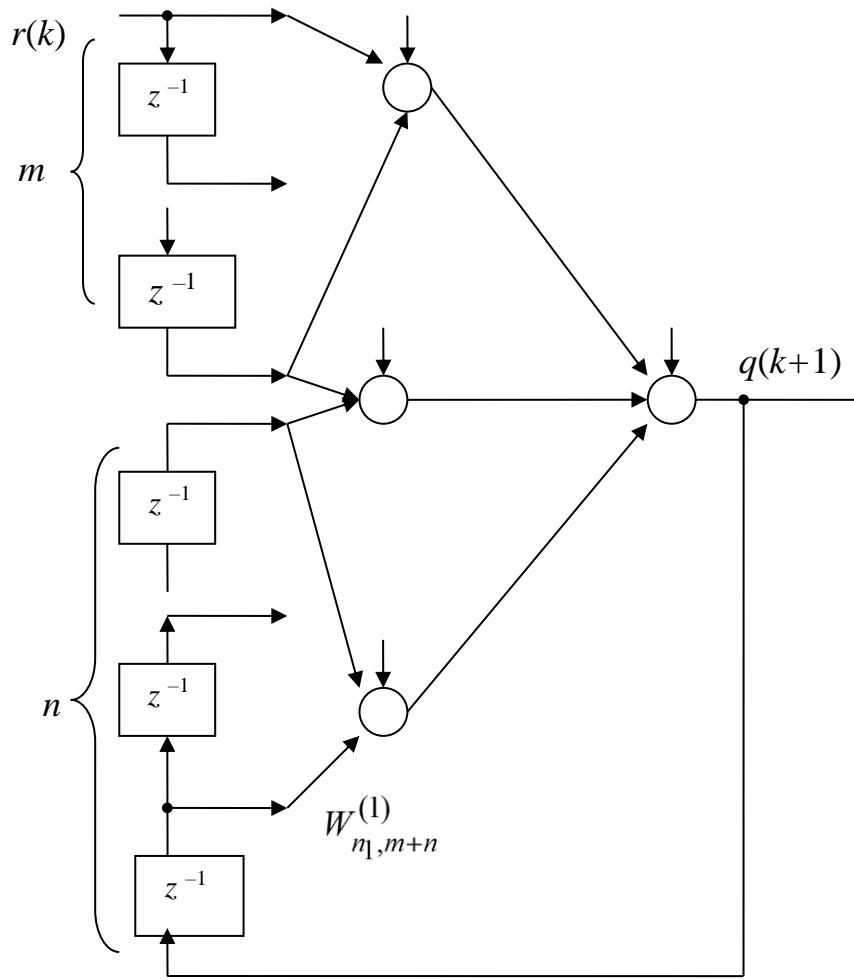


Рис. 2.9

$$\text{Взвешенная сумма ИН выходного слоя } s^{(2)} = \sum_{j=0}^{n_1} W_j^{(2)} q_j^{(1)}.$$

Выход сети $q = q^{(2)} = f(s^{(2)})$.

Коррекция весовых коэффициентов i -го ИН выходного слоя:

$$\Delta W_i^{(2)} = \gamma(u(k) - q^{(2)}(k)) \frac{dq^{(2)}(k)}{dW_i^{(2)}},$$

где $(u(k) - q^{(2)}(k))$ – динамическая ошибка.

Производная в текущий момент времени рассчитывается с использованием её значений в предыдущие моменты времени.

$$(\Delta W_{ij} = -\gamma \frac{\partial E_p}{\partial W_{ij}} = -\gamma \frac{\partial E_p}{\partial q_i} \frac{\partial q_i}{\partial W_{ij}} = \gamma(u_i - q_i) \frac{\partial q_i}{\partial W_{ij}},$$

$$\text{где } E_p = \frac{1}{2} \sum_i (u_i - q_i)^2, \quad \frac{\partial E_p}{\partial q_i} = -(u_i - q_i).$$

Коррекция весовых коэффициентов скрытого слоя:

$$\Delta W_{ij}^{(1)} = \gamma(u(k) - q^{(2)}(k)) \frac{dq^{(2)}(k)}{dW_{ij}^{(1)}},$$

$$(\delta_i^{(l)} = f_i'^{(l)}(s_i) \sum_j W_{ij}^{(l+1)} \delta_j^{(l+1)}).$$

Алгоритм функционирует в реальном масштабе времени, принимая поступающие входные данные, соответствующие им желаемые значения выходов и оперативно корректируя значения весовых коэффициентов.

При использовании нейронной сети в качестве регулятора аргумент алгоритма обучения включает переменные состояния объекта управления.

2.4. Алгоритм обучения сети Хопфилда

Выход сети для синхронного режима работы (1.13), (1.14):

$$\mathbf{q}(k+1) = f(W\mathbf{q}(k) + W_0). \quad (2.11)$$

Для упрощения рассуждений предполагается, что постоянная составляющая, определяющая порог срабатывания отдельного ИН (q_0), является компонентом вектора \mathbf{q} .

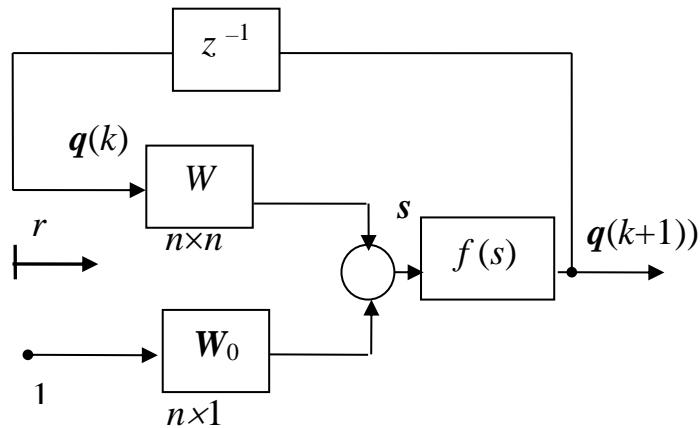
Весовые коэффициенты рассчитываются один раз перед началом работы сети на основе информации об обрабатываемых данных. Алгоритм обучения основан на правиле Хебба:

«При одновременной активации 2-х нейронов (+1 на выходе) следует увеличить величину весового коэффициента между ними».

Матрица весовых коэффициентов (синаптическая карта)

$$W = \sum_p (\mathbf{q}_p \mathbf{q}_p^T - I), \quad (2.12)$$

где \mathbf{q}_p – p -й входной образ. Согласно условию устойчивости диагональные элементы матрицы весовых коэффициентов должны равняться нулю: $W_{ii} = 0$, поэтому в алгоритме присутствует единичная матрица I .



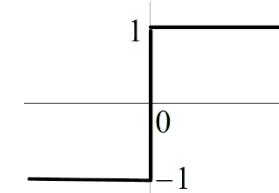
Алгоритм функционирования сети:

1. Формирование матрицы весовых коэффициентов (2.12) по серии обучающих входных образов.
2. Начальная активация сетей тестовым входным образом r :
 $r_i(0) = q_i(0), i = \overline{1, n}, n$ – число ИН.
3. Итерационные вычисления выходных сигналов согласно (2.11), пока сеть не придет в установившееся состояние.

Емкость сети Хопфилда: если количество ИН равно n , то число качественно распознаваемых образов равно $0.15 n$.

Пример 2.2. Вычисление восстановленного выходного образа при использовании искаженного входного. Используется пороговая функция активации (производная функции не требуется).

Даны три входных образа: $\mathbf{q}_1 = [1 1 1 1 1]^T$; $\mathbf{q}_2 = [-1 -1 1 1 1]^T$; $\mathbf{q}_3 = [1 1 -1 -1 -1]^T$.



1. Матрица весовых коэффициентов:

$$\begin{aligned}
 W = \sum_p (\mathbf{q}_p \mathbf{q}_p^T - I) &= \left(\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \cdot [1 \ 1 \ 1 \ 1 \ 1] - \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \right) + \\
 &+ \left(\begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \cdot [-1 \ -1 \ 1 \ 1 \ 1] - \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \right) + \\
 &+ \left(\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \cdot [1 \ 1 \ -1 \ -1 \ -1] - \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \right) = \begin{bmatrix} 0 & 3 & -1 & -1 & -1 \\ 3 & 0 & -1 & -1 & -1 \\ -1 & -1 & 0 & 3 & 3 \\ -1 & -1 & 3 & 0 & 3 \\ -1 & -1 & 3 & 3 & 0 \end{bmatrix} \quad (W_{ij} = W_{ji}, W_{ii} = 0).
 \end{aligned}$$

2. Подаем искаженный образ: $\mathbf{r} = [1 \ 1 \ 1 \ -1 \ -1]^T = \mathbf{q}(0)$.

3. Выход ИН – первый цикл обучения: $\mathbf{q}(1) = f(W\mathbf{q}(0)) = f([4 \ 4 \ -8 \ -2 \ -2]^T) = [1 \ 1 \ -1 \ -1 \ -1]^T$.

Выход ИН – второй цикл обучения: $\mathbf{q}(2) = f[W\mathbf{q}(1)] = f([6 \ 6 - 8 - 8 - 8]^T) = [1 \ 1 - 1 - 1 - 1]^T$.

Установившееся состояние – нейрон на каждом следующем цикле вырабатывает тот же сигнал, что и на предыдущем.

Таким образом, сеть скорректировала искаженный образ к ближайшему эталонному образу \mathbf{q}_3 .

2.5. Обучение сети "дву направленная ассоциативная память"

Весовые коэффициенты сети рассчитываются один раз перед началом работы сети на основе правила Хебба. Матрица весовых коэффициентов вычисляется по формуле

$$W = \sum_p \mathbf{q}_p^{(2)} \mathbf{q}_p^{(1)T}, \quad (2.13)$$

где $\mathbf{q}_p^{(1)}, \mathbf{q}_p^{(2)}$ – пара ассоциированных образов, p -е образы. Сеть сохраняет пары образов и позволяет по искаженному входному образу восстановить ближайший выходной образ.

Ёмкость сети: если размер $\mathbf{q}_p^{(1)}, \mathbf{q}_p^{(2)} – n_1, n_2$ соответственно, то число качественно распознаваемых образов $k < \sqrt{\min(n_1, n_2)}$.

Алгоритм функционирования сети:

1. Вычисление матрицы весовых коэффициентов по формуле (2.13).
2. Активация сетей начальным образом $r(0)$. Приведение нейронов 1-го слоя в начальное состояние: $q_i^{(1)}(0) = r_i(0), \quad i = \overline{1, n}$.

3. Вычисление нейронов 2-го слоя по формуле: $q_i^{(2)} = f\left(\sum_j W_{ij} q_j^{(1)}\right)$.

4. Поступление сигналов с выходов нейронов 2-го слоя на входы нейронов 1-го слоя и вычисление новых состояний нейронов 1-го слоя по формуле

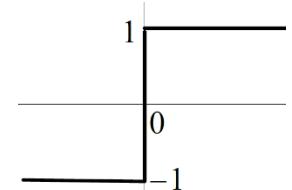
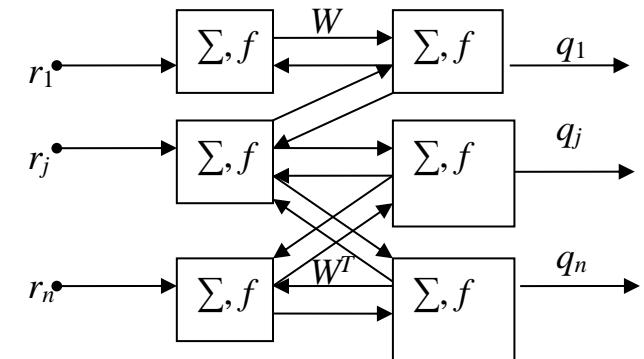
$$q_i^{(1)}(k+1) = f\left[\sum_{j=0}^{n_2} W_{ij} q_j^{(2)}(k)\right] \quad (2.14)$$

или в векторной форме $\mathbf{q}^{(1)}(k+1) = f[W^T \mathbf{q}^{(2)}(k)]$.

Повторение шагов 3 и 4, пока сеть не придет в установившееся состояние.

Пример 2.3. Функционирование сети "дву направленная ассоциативная память".

Используется пороговая функция активации (1.4). Сеть состоит из четырех ИН 1-го слоя и трех ИН 2-го слоя. Обучающая выборка состоит из следующих образов:



$$\mathbf{q}_1^{(1)} = [-1 -1 1 1]^T, \mathbf{q}_1^{(2)} = [1 -1 1]^T, \mathbf{q}_2^{(1)} = [1 1 -1 -1]^T, \mathbf{q}_2^{(2)} = [-1 1 -1]^T.$$

1. Матрица весовых коэффициентов (2.13): $W = \sum_{p=1}^2 q_p^{(2)} {q_p^{(1)}}^T =$

$$= \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -2 & -2 & 2 & 2 \\ 2 & 2 & -2 & -2 \\ -2 & -2 & 2 & 2 \end{bmatrix}.$$

2. Приведение нейронов 1-го слоя в начальное состояние:

$$\mathbf{r}(0) = \mathbf{q}^{(1)}(0) = [-1 \ 1 \ 1 \ 1]^T \text{ (исходный образ).}$$

3. Вычисление нейронов 2-го слоя по формуле

$$\mathbf{q}^{(2)}(0) = f[W\mathbf{q}^{(1)}(0)] = f([4 -4 4]^T) = [1 -1 1]^T.$$

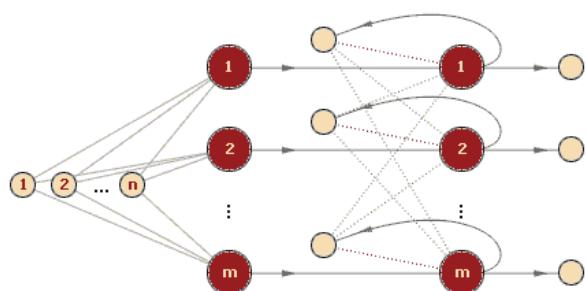
4. Вычисление новых состояний нейронов 1-го слоя (2.14):

$$\mathbf{q}^{(1)}(1) = f[W^T \mathbf{q}^{(2)}(0)] = f([-6 -6 6 \ 6]^T) = [-1 -1 1 \ 1]^T.$$

Таким образом, сеть скорректировала исходный образ к ближайшему примеру $\mathbf{q}_1^{(2)}$ и выдала соответствующую ему ассоциацию $\mathbf{q}_1^{(1)}$.

Алгоритм функционирования сети Хэмминга:

- На стадии инициализации весовым коэффициентам *первого слоя* присваиваются значения: $w_i^p = \frac{r_i^p}{2}, i = \overline{1, n}, p = \overline{1, m}$, n – размерность вектора входа; r_i^p – i -ый элемент p -ого образа. Порог функции активации – $T = \frac{n}{2}$.



Весовые коэффициенты обратной связи с выходов нейронов *второго слоя* отрицательны и равны $0 < \varepsilon < 1/m$, если сигнал подается на входы всех остальных нейронов. Если сигнал подается с выхода на вход нейрона, весовой коэффициент равен +1.

- На входы нейронной сети подается неизвестный входной вектор $\mathbf{r} = \{r_i\}, i = \overline{1, n}$, с

учетом которого рассчитывается взвешенная сумма нейронов первого слоя $s_i^{(1)} = \sum_{j=0}^n w_{ij} r_j + T, i = \overline{1, m}$. Выход

нейронов первого слоя $q_i^{(1)} = f(s_i^{(1)})$.

После этого полученными значениями инициализируются нейроны второго слоя: $q_i^{(2)} = q_i^{(1)}$.

- Вычисляются новые состояния нейронов второго слоя, взвешенная сумма i -го нейрона:

$$s_i^{(2)}(k+1) = q_i^{(2)}(k) - \varepsilon \sum_{p=1}^m q_p^{(2)}(k), p \neq i, i = \overline{1, m}, \text{ выход } i\text{-го нейрона } q_i^{(2)}(k+1) = f(s_i^{(2)}(k+1)), i = \overline{1, m}.$$

Используется линейная с насыщением функция активации, причем величина предела должна быть достаточно большой, чтобы любые возможные значения аргумента не приводили к насыщению.

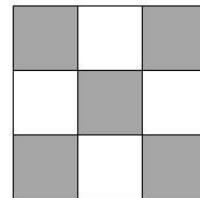
4. Проверить, изменились ли выходы нейронов второго слоя за последнюю итерацию. Если да, то перейди к шагу 2. Иначе – конец.

Из оценки алгоритма видно, что роль первого слоя весьма условна: воспользовавшись один раз на первом шаге значениями его весовых коэффициентов, сеть больше не обращается к нему, поэтому первый слой может быть заменен на матрицу весовых коэффициентов.

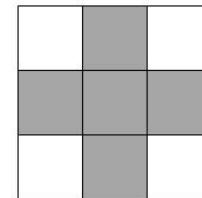
Пример 2.2.

Пусть входной вектор содержит 9 элементов, выходной – 3 элемента.

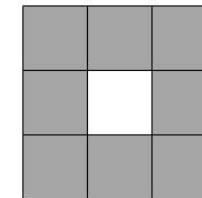
Образ 1

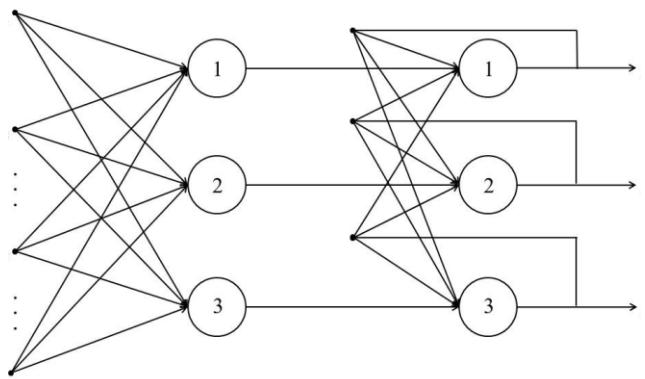


Образ 2



Образ 3





Наличие рисунка в любом из 9 элементов кодируется значением 1, отсутствие – значением –1.

Создаваемая нейронная сеть будет включать 9 входных переменных и по 3 нейрона в первом и втором (выходном) слоях

Эталонные образы

№ образа	№ входной переменной								
	1	2	3	4	5	6	7	8	9
1	1	-1	1	-1	1	-1	1	-1	1
2	-1	1	-1	1	1	1	-1	1	-1
3	1	1	1	1	-1	1	1	1	1

Весовые коэффициенты нейронов первого слоя

№	№ входной переменной

образа	1	2	3	4	5	6	7	8	9
1	0,5	-0,5	0,5	-0,5	0,5	-0,5	0,5	-0,5	0,5
2	-0,5	0,5	-0,5	0,5	0,5	0,5	-0,5	0,5	-0,5
3	0,5	0,5	0,5	0,5	-0,5	0,5	0,5	0,5	0,5

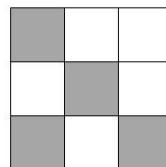
По формуле $T=n/2$ определяем порог активационной функции $T = 9/2 = 4,5$. С учетом ограничения $\varepsilon \in (0, 1/3]$ абсолютное значение весового коэффициента принимается $\varepsilon = 0,3$. Пусть максимальная ошибка распознавания входного образа равна $E_{\max} = 0,1$.

Весовые коэффициенты обратных связей нейронов второго слоя $\varepsilon_{ij} = \begin{cases} 1, & i = j, \\ -\varepsilon, & i \neq j, \end{cases} \varepsilon \in (0, \frac{1}{3}]$.

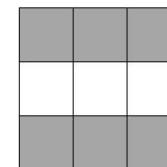
№ выхода	№ нейрона второго слоя		
	1	2	3
1	1	$-\varepsilon$	$-\varepsilon$
2	$-\varepsilon$	1	$-\varepsilon$
3	$-\varepsilon$	$-\varepsilon$	1

Для тестирования настроенной сети используем два зашумленных графических образа:

Образ 4



Образ 5



Подаем на входы сети бинарный вектор, соответствующий образу 4: $r^T = [1, -1, -1, -1, 1, -1, 1, -1, 1]$.

Пусть функция активации задана выражением $f(s) = \begin{cases} 0, & s \leq 0; \\ s, & 0 < s \leq T; \\ T, & s \geq T. \end{cases}$

С учетом пункта 2 алгоритма функционирования сети получим значение взвешенной суммы первого нейрона первого слоя

$$s_1^{(1)} = \sum_{j=1}^9 w_{1j} r_j + T = 0,5 + 0,5 + (-0,5) + 0,5 + 0,5 + 0,5 + 0,5 + 0,5 + 0,5 + 4,5 = 8.$$

Значение взвешенной суммы второго нейрона первого слоя

$$s_2^{(1)} = \sum_{j=1}^9 w_{1j} r_j + T = -0,5 + (-0,5) + 0,5 + (-0,5) + 0,5 + (-0,5) + (-0,5) + (-0,5) + (-0,5) + 4,5 = 2.$$

Значение взвешенной суммы третьего нейрона первого слоя

$$s_3^{(1)} = \sum_{j=1}^9 w_{1j} r_j + T = 0,5 + (-0,5) + (-0,5) + (-0,5) + (-0,5) + (-0,5) + 0,5 + (-0,5) + 0,5 + 4,5 = 3.$$

Выход нейронов первого слоя $\mathbf{q}^{(1)} = [4,5 \ 2 \ 3]^T$.

Выходам нейронной сети присваиваются соответствующие выходные значения нейронов первого слоя $\mathbf{q}^{(2)} = [4,5 \ 2 \ 3]^T$.

Далее итерационно рассчитывается серия выходных векторов (расчеты выполняются аналогично) до выполнения условия стабилизации.

№ итерации	Взвешенная сумма нейронов выходного слоя			Выход сети			Ошибка распознавания
1	8,00	2,00	3,00	4,50	2,00	3,00	—
2	3,00	-0,25	1,05	3,00	0,00	1,05	10,05
3	2,69	-1,22	0,15	2,69	0,00	0,15	0,91
4	2,64	-0,85	-0,66	2,64	0,00	0,00	0,02

Вторая итерация, взвешенная сумма нейронов второго слоя: $s_1^{(2)}(k+1) = q_1^{(2)}(k) - \varepsilon \sum_{p=1}^3 q_p^{(2)}(k) = 4,5 - 0,3(2 + 3) = 3,$

$$s_2^{(2)}(k+1) = q_2^{(2)}(k) - \varepsilon \sum_{p=1}^3 q_p^{(2)}(k) = 2 - 0,3(4,5 + 3) = -0,25,$$

$$s_3^{(2)}(k+1) = q_3^{(2)}(k) - \varepsilon \sum_{p=1}^3 q_p^{(2)}(k) = 3 - 0,3(4,5 + 2) = 1,05.$$

Выход нейронов второго слоя после второй итерации $f(s) = \begin{cases} 0, & s \leq 0; \\ s, & 0 < s \leq T; \\ T, & s \geq T. \end{cases}$

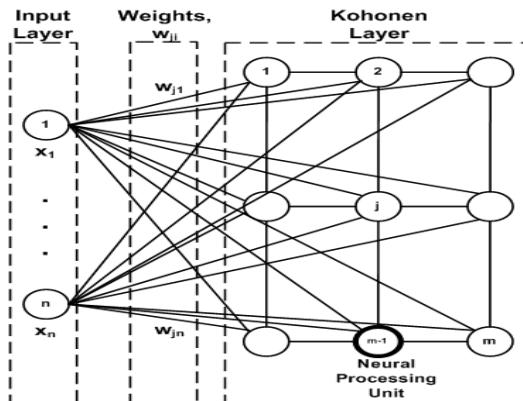
$$\mathbf{q}^{(2)} = [3 \ 0 \ 1,05]^T.$$

После четвертой итерации ошибка распознавания, рассчитанная как квадрат евклидова расстояния, равна $E = (0,05^2 + 0,15^2) = 0,02.$

Критерий остановки цикла возврата сигнала по обратным связям выполнен после 4-й итерации. Положительное выходное значение первого нейрона указывает на то, что зашумленный входной образ следует отнести к 1-му классу.

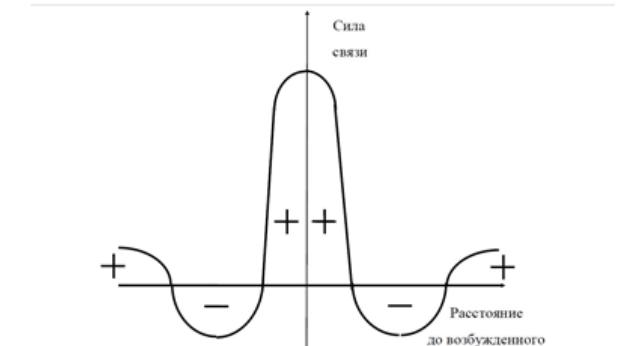
2.6. Алгоритм обучения сети Кохонена

Нейроны слоя Кохонена расположены на плоскости и связаны между собой весовыми коэффициентами, величина



которых зависит от расстояния между нейронами.

Такой вид связей обеспечивает взаимное усиление сигнала близкими нейронами и ослабление влияния далеких нейронов, что делает более контрастной границу раздела возбужденных нейронов от остальных, ложное возбуждение которых этим подавляется.



Входные векторы, как правило, нормализуют: $r_j' = \frac{r_j}{\sqrt{\sum_i r_i^2}}, i, j = \overline{1, n}.$

После поступления на вход сети вектора \mathbf{r} вычисляется расстояние между входным вектором и вектором коэффициентов нейронов. Нейрон, вектор весовых коэффициентов которого наиболее близок к входному вектору, считается "победителем", его весовые коэффициенты корректируются: $\|\mathbf{W}_* - \mathbf{r}\| = \min_i \|\mathbf{W}_i - \mathbf{r}\|, i = \overline{1, n}$, где \mathbf{W}_* – вектор весовых коэффициентов ИН "победителя"; n – число ИН слоя Кохонена. Весовые коэффициенты остальных нейронов остаются без изменений.

Коррекция вектора весовых коэффициентов:

$$\mathbf{W}_i(k+1) = \mathbf{W}_i(k) + \gamma(\mathbf{r} - \mathbf{W}_i(k)). \quad (2.15)$$

Коррекция j -го весового коэффициента i -го ИН:

$$W_{ij}(k+1) = W_{ij}(k) + \gamma(r_j - W_{ij}(k)).$$

Вектор весовых коэффициентов вращается к входному вектору (рис. 2.10). Коррекция весовых коэффициентов осуществляется в направлении уменьшения разности между \mathbf{r} и \mathbf{W}_i .

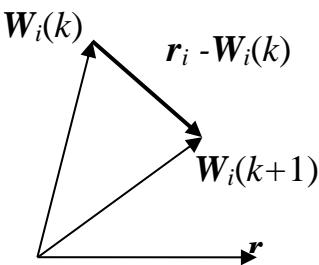


Рис. 2.10

Если бы число нейронов слоя Кохонена равнялось числу входных образов, то при $\gamma = 1$ можно было бы обучить весь слой, выполняя одну коррекцию на каждый нейрон. На практике, как правило, обучающее множество включает группы сходных векторов. НС должна быть обучена активизировать один и тот же нейрон для близких входных векторов. Это достигается уменьшением γ . В результате вектор весовых коэффициентов ИН, ассоциированного с группой входных образов, будет стабилизироваться около среднего значения, соответствующего "центру" данной группы, как показано на рис. 2.11, где

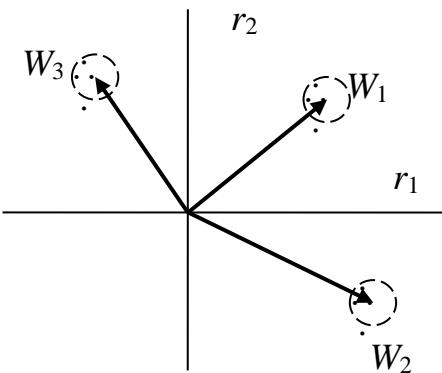


Рис. 2.11

обозначено: W_i – вектор весовых коэффициентов слоя Кохонена; – входные данные.

Пример 2.4. Для обучения сети Кохонена с тремя входными ИН и двумя ИН слоя Кохонена используется учебный вектор

$$\mathbf{r} = [0.8 \ 0.7 \ 0.4]^T. \text{ Начальные значения весовых коэффициентов: } W = \begin{bmatrix} 0.5 & 0.6 & 0.8 \\ 0.4 & 0.2 & 0.5 \end{bmatrix},$$

коэффициент обучения $\gamma = 0.5$. Вычислить изменения весовых коэффициентов в ходе первого цикла обучения.

Вычисление расстояний между входным вектором r и вектором весовых коэффициентов W_i : $d_i = \sum_{j=1}^3 (W_{ij} - r_j)^2$,

$$d_1 = (0.5 - 0.8)^2 + (0.6 - 0.7)^2 + (0.8 - 0.4)^2 = 0.26,$$

$$d_2 = (0.4 - 0.8)^2 + (0.2 - 0.7)^2 + (0.5 - 0.4)^2 = 0.42;$$

$d_1 < d_2$, поэтому изменяются коэффициенты первого ИН .

2) Коррекция весового коэффициента первого ИН слоя Кохонена.

$$W_{11}(k+1) = W_{11}(k) + \gamma(r_1 - W_{11}(k)) = 0.5 + 0.5(0.8 - 0.5) = 0.65;$$

$$W_{12} = 0.6 + 0.5(0.7 - 0.6) = 0.65;$$

$$W_{13} = 0.8 + 0.5(0.4 - 0.8) = 0.6.$$

Скорректированное значение матрицы весовых коэффициентов: $W = \begin{bmatrix} 0.65 & 0.65 & 0.6 \\ 0.4 & 0.6 & 0.5 \end{bmatrix}$.

После первого цикла обучения расстояние между вектором весовых коэффициентов первого ИН и входным вектором уменьшилось.

2.8. Алгоритм встречного распространения ошибок

(Counter propagation Networks)

Предназначен для обучения сетей встречного распространения ошибок (см. рис. 1.34).

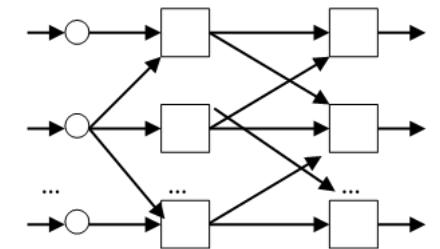
В процессе обучения сети представляются обучающие примеры (\mathbf{r}, \mathbf{u}) , где \mathbf{r} – входной образ, \mathbf{u} – желаемый выходной образ.

Слой Кохонена обучается в соответствии с алгоритмом (2.15)

$$s_i = \sum_j (W_{ij}^{(1)} - r_j)^2$$

$$\mathbf{W}_i^{(1)}(k+1) = \mathbf{W}_i^{(1)}(k) + \gamma(\mathbf{r} - \mathbf{W}_i^{(1)}(k)),$$

где \mathbf{r} – желаемый входной вектор.



Входной слой *Слой Кохонена* *Слой Гроссберга*

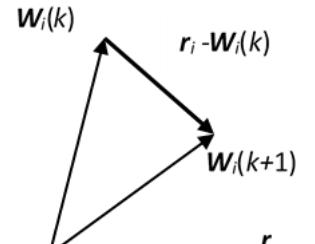


Рис. 2.10

Если сеть функционирует таким образом, что выход ИН-“победителя” равен единице, а выходы остальных ИН равны нулю, то выход ИН слоя Гроссберга определяется только величиной весового коэффициента связи с ИН-“победителем”.

Повышение точности обеспечивается формированием целой группы нейронов слоя Кохонена, имеющих максимальные выходы, которые передают свои выходные сигналы в слой Гроссберга.

Обучение 2-го слоя согласно правилу Гроссберга проводится по уравнению

$$W_{ij}^{(2)}(k+1) = W_{ij}^{(2)}(k) + \gamma q_j^{(1)} (u_i - W_{ij}^{(2)}(k)),$$

где $q_j^{(1)}$ – выход нейрона «победителя» слоя Кохонена.

В процессе обучения весовые коэффициенты первого слоя корректируются по значениям входных образов r , весовые коэффициенты 2-го слоя – по значениям желаемых выходных образов u , поэтому принцип обучения называется *алгоритмом встречного распространения ошибок*. Из названия алгоритма следует название сети.

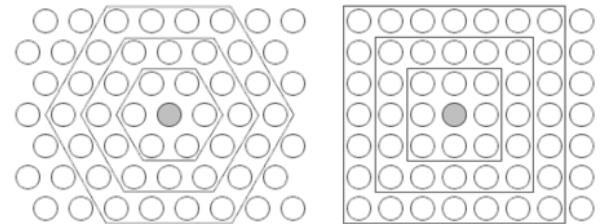
Недостатки сети встречного распространения

1. Высокая вычислительная сложность.
2. Чувствительность к начальным значениям весов.

Неправильный выбор начальных значений может привести к застреванию в локальных минимумах или медленному сходимости обучения.

3. Подверженность переобучению.

Сеть “запоминает” обучающие примеры, но не может обобщать на новые данные.



4. Сложно анализировать работу сети. (Неинтерпретируемость).

Применение сети встречного распространения

Распознавание образов

Например, распознавание рукописных цифр на почтовых конвертах или для классификации изображений на основе их содержимого.

Прогнозирование временных рядов

Они могут анализировать исторические данные и предсказывать будущие значения на основе обнаруженных закономерностей и трендов.

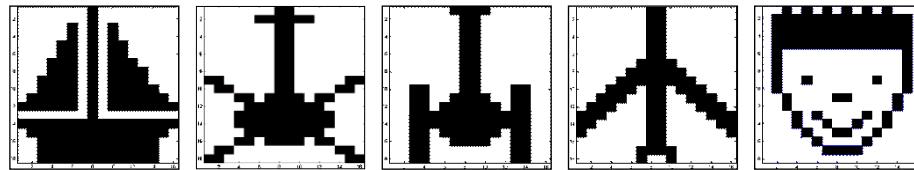
Рекомендательные системы

Они могут анализировать предпочтения и поведение пользователей и предлагать им персонализированные рекомендации, такие как фильмы, музыка, товары или новости.

Эффективность классификации образов демонстрирует рис. 2.12. Результаты в примере получены с использованием функции *newlvg* пакета Matlab. Тестовые образы включают 32 % пикселей в искаженном состоянии.

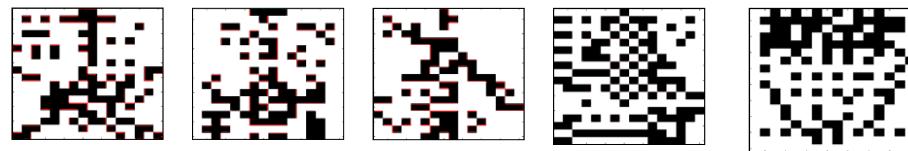
Набор данных для обучения

ВХОДНЫЕ ОБРАЗЫ



КЛАССЫ 1 2 3 4 5

ТЕСТОВЫЕ ОБРАЗЫ

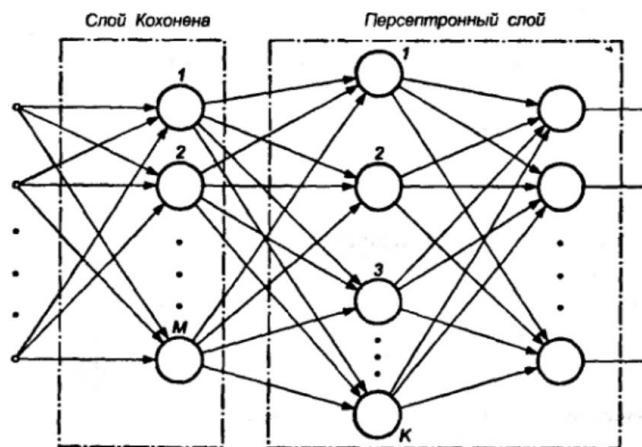


КЛАССЫ 1 2 3 4 5

Рис. 2.12

2.9. Обучение гибридной сети

Структура сети представляет каскадное соединения слоя Кохонена и персепtronной сети (часто оказывается достаточным включение одного персепtronного слоя, обычно линейного). Появляется возможность совместить способности слоя Кохонена к локализации и свойство аппроксимации, присущее персептрону.



Обучение гибридной сети состоит из двух этапов

- 1) Слой Кохонена обучается в соответствии с алгоритмом (2.15)

$$\mathbf{W}_i^{(1)}(k+1) = \mathbf{W}_i^{(1)}(k) + \gamma(\mathbf{r} - \mathbf{W}_i^{(1)}(k)),$$

где r – желаемый входной вектор. Возможно применение и других алгоритмов с самоорганизацией.

Выход нейрона «победителя» или, для повышения точности вычислений, выходы целой группы нейронов слоя Кохонена, имеющих максимальные выходы, нормализуются на основе формулы Гаусса

$$q_i^{(1)} = f_i^{(1)}(s) = \exp\left(-\frac{\|s_{\max} - s_i^{(1)}\|^2}{\sigma^2}\right).$$

где $q_i^{(1)}$ – выход нейрона слоя Кохонена, $s_{\max} = q_{\max} = 1$ – выход нейрона «победителя», σ^2 подбирается экспериментально.

2) Персептронная сеть обучается с учителем на основе алгоритма обратного распространения ошибок

$$W_{ij}^{(2)(k+1)} = W_{ij}^{(2)(k)} + \gamma q_j^{(1)} (u_i - q_i^{(2)}).$$

где u_i – желаемый выход i -го нейрона выходного персептронного слоя сети.

Сеть эффективна при решении задач распознавания образов и прогнозирования.

2.10. Обучение сети на основе теории адаптивного резонанса

(Adaptive Resonance Theory, ART)

Сеть, предложенная С. Гроссбергом и Г. Карпентером, – упрощенная модель восприятия человеком информации. Полученная информация идентифицируется. Часть такой информации игнорируется, как неважная и неинтересная. Небольшая ее составляющая, имеющая ценность, обрабатывается и сохраняется в долговременной памяти без нарушения существующих. Говорят, что сеть адаптивного резонанса теории (ART) предлагает решение проблемы "пластичности/стабильности" памяти.

В отличие от большинства архитектур нейронных сетей АРТ-сети *продолжают обучаться на протяжении всего времени* их практического использования.

Структура любой АРТ-сети содержит единственный слой нейронов. Количество входных переменных равно числу признаков, характеризующих объект. Количество выходов непостоянно. В момент начала функционирования сети выходов нет вовсе. Постепенно их количество возрастает с каждым новым незнакомым входным образом, образующим, по сути, новый кластер. Таким образом, АРТ-сети – самоорганизующиеся.

Отличие АРТ-1 связаны с выходами не одним, а парой весовых коэффициентов (синапсов): w_{ij} – синапсы кратковременной памяти; t_{ij} – синапсы долговременной памяти. Назначение кратковременной памяти – установление кластеров, к которым может быть отнесен входной образ. Назначение долговременной памяти – установление степени соответствия входного образа кластерам, определение нейрона-победителя или принятие решения о создании нового кластера.

АРТ-1 содержит бинарные входы (0 или 1), нормализация входных данных не требуется. Алгоритм функционирования АРТ-1 включает 3 стадии: инициализацию сети, распознавание образа, кластеризацию образа (сравнение).

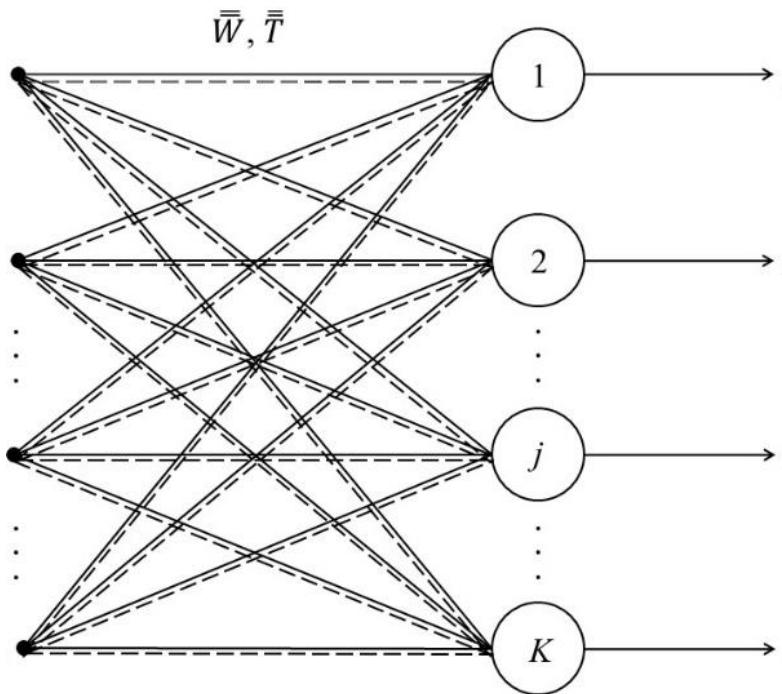


Рис. 2.13. Структура нейронной сети АРТ-1

где λ – положительная ($> 1,0$) константа, определяющая степень *влияния нового входного образа на кратковременную память*. Чаще всего λ принимается равной 2,0.

Стадия распознавания образа включает следующие этапы:

2.1. На входы сети подается новый входной образ. Для каждого нейрона рассчитывается значение его выхода с учетом весовых коэффициентов кратковременной памяти:

На стадии инициализации последовательно выполняются следующие этапы:

- 1.1. Устанавливается *параметр сходства* R_{kp} ($0 < R_{kp} < 1$), причем чем он больше, тем выше должно быть *сходство образа и кластера*.
- 1.2. Для первого поданного на входы сети образа создается первый нейрон (кластер), значения весовых коэффициентов которого равны

$$W_{1j} = \frac{\lambda \cdot r_j}{\lambda - 1 + \sum_{k=1}^n r_k}, \quad (2.16)$$

$$t_{1j} = r_j, \quad (2.17)$$

$$q_i = \sum_{j=1}^n W_{ij} r_j . \quad (2.18)$$

2.2. Положительные выходы нейронов указывают на кластеры, имеющие качественное сходство с входным образом. Если же все выходы оказались нулевыми, входной образ не соответствует ни одному из кластеров, создается новый нейрон с весовыми коэффициентами, рассчитываемыми по соотношениям (2.16) и (2.17), после чего алгоритм продолжает работу с п. 2.1.

Количественное сходство входного образа с кластерами определяется на стадии **кластеризации образа (сравнения)**:

3.1. Рассчитывается количественная мера *сходства входного образа с кластером*, имеющим наибольшее значение выхода нейрона в п. 2.1.

$$R_i = \frac{\sum_{j=1}^n t_{ij} r_j}{\sum_{j=1}^n r_j} . \quad (2.19)$$

3.2. Если выполняется условие $R_i > R_{kp}$, i -й нейрон считается нейроном-победителем, а входной образ – соответствующим i -му кластеру. В этом случае для нейрона выполняется пересчет весовых коэффициентов по соотношениям:

$$W_{ij}(k+1) = (1 - \gamma) \cdot W_{ij}(k) + \gamma \frac{\lambda \cdot r_j}{\lambda - 1 + \sum_{k=1}^n r_k} , \quad (2.20)$$

$$t_{ij}(k+1) = (1 - \nu) \cdot t_{ij}(k) + \nu \cdot r_j , \quad (2.21)$$

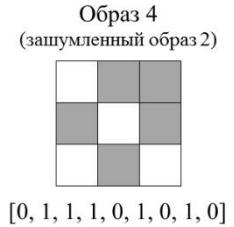
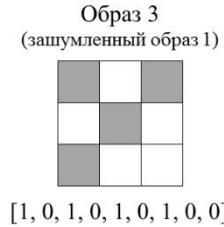
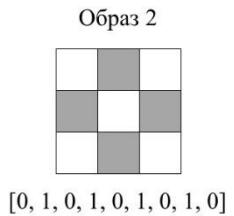
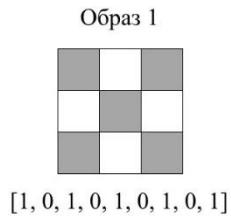
где γ – коэффициент скорости адаптации ($0 < \gamma < 1$). Большее значение γ соответствует быстрой адаптации.

3.3. Если условие $R_i > R_{kp}$ не выполняется, алгоритм повторяется с п. 3.1 для остальных кластеров, следующие в порядке убывания их выходных значений, рассчитанных в п. 2.1, до тех пор, пока не будет получен нейрон-победитель или не будут использованы все кластеры с положительным выходом нейрона.

3.4. Если нейрон-победитель не найден, создается новый кластер с весовыми коэффициентами нейрона, рассчитанными по соотношениям (2.16) и (2.17), после чего алгоритм продолжает работу с п. 2.1.

Рассмотрим пример работы сети АРТ-1 на протяжении одного цикла. Последовательно подадим на ее входы 4 бинарных вектора, состоящих из 9 элементов, кодирующих различные образы. Графическая и цифровая интерпретации образов приведены на рис. 2.14. Структура нейронной сети включает 9 входов. Зададим параметры настройки нейронной сети: $R_{kp} = 0,7$, $\lambda = 2,0$,

$$\gamma = 0,6.$$



На *стадии инициализации* подаем на входы сети образ 1, $\mathbf{r}^{(1)} = [1, 0, 1, 0, 1, 0, 1, 0, 1]^T$. Он формирует первый нейрон, весовые коэффициенты которого рассчитываются по соотношениям (2.16), (2.17) :

$$W_{1j} = \frac{\lambda \cdot r_j}{\lambda - 1 + \sum_{k=1}^n r_k}, \quad t_{1j} = r_j.$$

$$W_{11} = \frac{2 \cdot 1}{2 - 1 + (1 + 0 + 1 + 0 + 1 + 0 + 1 + 0 + 1)} = 0,33; \quad W_{11} = W_{13} = W_{15} = W_{17} = W_{19} = 0,33;$$

Рис. 2.14 – Набор входных образов для кластеризации с помощью сети АРТ-1

$$W_{12} = \frac{2 \cdot 0}{2 - 1 + (1 + 0 + 1 + 0 + 1 + 0 + 1 + 0 + 1)} = 0, \quad W_{12} = W_{14} = W_{16} = W_{18} = 0; \quad t_{11} = t_{13} = t_{15} = t_{17} = t_{19} = 1,00; \quad t_{12} = t_{14} = t_{16} = t_{18} = 0,00.$$

Стадия распознавания образа

Подаем на входы сети образ 2. $\mathbf{r}^{(2)} = [0, 1, 0, 1, 0, 1, 0, 1, 0]^T$.

Выходное значение единственного существующего на данный момент нейрона, рассчитанное по соотношению (2.18),

$$q_i = \sum_{j=1}^n W_{ij} r_j, \quad q_1 = 0,33 \cdot 0 + 0 \cdot 1 + 0,33 \cdot 0 = 0.$$

Таким образом, образ 2 не соответствует первому кластеру, и для него создается новый нейрон с весовыми коэффициентами,

$$\text{рассчитанными по соотношениям (2.16), (2.17) : } W_{22} = -\frac{\lambda \cdot r_j}{\lambda - 1 + \sum_{k=1}^n r_k} = \frac{2 \cdot 1}{2 - 1 + (0 + 1 + 0 + 1 + 0 + 1 + 0 + 1 + 0)} = 0,4,$$

$$W_{21} = W_{23} = W_{25} = W_{27} = W_{29} = 0,00; W_{22} = W_{24} = W_{26} = W_{28} = 0,40; t_{21} = t_{23} = t_{25} = t_{27} = t_{29} = 0,00; t_{22} = t_{24} = t_{26} = t_{28} = 1,00.$$

Подаем на входы нейронной сети образ 3, $\mathbf{r}^{(3)} = [1, 0, 1, 0, 1, 0, 1, 0, 0]^T$.

Выходные значения имеющихся двух нейронов равняются соответственно:

$$q_1 = \sum_{j=1}^n W_{ij} r_j, q_1 = 0,33 \cdot 1 + 0 \cdot 0 + 0,33 \cdot 0 = 1,32,.$$

$$q_2 = 0 \cdot 1 + 0,4 \cdot 0 + 0 \cdot 0 = 0.$$

Стадия кластеризации образа

$$\sum_{j=1}^n t_{ij} r_j$$

$$\text{Проверим соответствие образа 3 первому кластеру согласно (2.19) } R_i = \frac{\sum_{j=1}^n t_{ij} r_j}{\sum_{j=1}^n r_k},$$

$$R_1 = \frac{\sum_{j=1}^n t_{ij} r_j}{\sum_{j=1}^n r_k} = \frac{1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 0 \cdot 0}{1 + 0 + 1 + 0 + 1 + 0 + 1 + 0 + 0} = 1, R_1 = 1,0 > R_{kp}.$$

Таким образом, соответствие образа 3 ($\mathbf{r}^{(3)} = [1, 0, 1, 0, 1, 0, 1, 0, 0]^T$) первому кластеру считаем установленным.

По соотношениям (2.20) и (2.21) пересчитываются коэффициенты кратковременной и долговременной памяти первого нейрона:

$$(2.20) W_{ij}(k+1) = (1 - \gamma) \cdot W_{ij}(k) + \gamma \frac{\lambda \cdot r_j}{\lambda - 1 + \sum_{k=1}^n r_k} .$$

Коэффициенты кратковременной памяти

$$w_{11}(k+1) = (1 - 0,6) \cdot 0,33 + 0,6 \frac{2 \cdot 1}{2 - 1 + (1 + 0 + 1 + 0 + 1 + 0 + 1 + 0 + 0)} = 0,37 ,$$

$$w_{19}(k+1) = (1 - 0,6) \cdot 0,33 + 0,6 \frac{2 \cdot 0}{2 - 1 + (1 + 0 + 1 + 0 + 1 + 0 + 1 + 0 + 0)} = 0,13 ,$$

$$w_{12}(k+1) = (1 - 0,6) \cdot 0 + 0,6 \frac{2 \cdot 0}{2 - 1 + (1 + 0 + 1 + 0 + 1 + 0 + 1 + 0 + 0)} = 0,00.$$

Новые значения составят $W_{11} = W_{13} = W_{15} = W_{17} = 0,37$; $W_{19} = 0,13$; $W_{12} = W_{14} = W_{16} = W_{18} = 0,00$.

Коэффициенты долговременной памяти, (2.21) $t_{ij}(k+1) = (1 - \gamma) \cdot t_{ij}(k) + \gamma \cdot r_j$.

$$t_{11}(k+1) = (1 - 0,6) \cdot 1 + 0,6 \cdot 1 = 1, \quad t_{19}(k+1) = (1 - 0,6) \cdot 1 + 0,6 \cdot 0 = 0,4,$$

$$t_{12}(k+1) = (1 - 0,6) \cdot 0 + 0,6 \cdot 0 = 0.$$

Новые значения составят $t_{11} = t_{13} = t_{15} = t_{17} = 1,00$; $t_{19} = 0,40$; $t_{12} = t_{14} = t_{16} = t_{18} = 0,00$.

Подаем на входы нейронной сети образ 4, $\mathbf{r}^{(4)} = [0, 1, 1, 1, 0, 1, 0, 1, 0]^T$.

Выходные значения нейронов составят соответственно:

$$q_i = \sum_{j=1}^n W_{ij} r_j, \quad q_1 = 0,37 \cdot 0 + 0 \cdot 1 + 0,37 \cdot 1 + 0 \cdot 1 + 0,37 \cdot 0 + 0 \cdot 1 + 0,37 \cdot 0 + 0 \cdot 1 + 0,13 \cdot 0 = 0,37,$$

$$q_2 = 0 \cdot 0 + 0,4 \cdot 1 + 0 \cdot 1 + 0,4 \cdot 1 + 0 \cdot 0 + 0,4 \cdot 1 + 0 \cdot 0 + 0,4 \cdot 1 + 0 \cdot 0 = 1,6.$$

$$t_{21} = t_{23} = t_{25} = t_{27} = t_{29} = 0,00; \quad t_{22} = t_{24} = t_{26} = t_{28} = 1,00.$$

Таким образом, сходство возможно с обоими образами, но $q_2 > q_1$, рассчитываем количественная мера сходства входного

$$\text{образа со вторым кластером: } R_j = \frac{\sum_{j=1}^n t_{ij} r_j}{\sum_{j=1}^n r_k}; R_2 = \frac{0 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0}{0 + 1 + 1 + 1 + 0 + 1 + 0 + 1 + 0} = 0,8.$$

$R_2 > R_{kp}$, следовательно образ 4 должен быть отнесен ко второму кластеру.

По соотношениям (2.20) и (2.21) пересчитываются весовые коэффициенты второго нейрона:
 $W_{21} = W_{25} = W_{27} = W_{29} = 0,00; W_{23} = 0,20; W_{22} = W_{24} = W_{26} = W_{28} = 0,36; t_{21} = t_{25} = t_{27} = t_{29} = 0,00; t_{23} = 0,60; t_{22} = t_{24} = t_{26} = t_{28} = 1,00$

Функционирование нейронной сети продолжается далее в соответствии с алгоритмами стадий распознавания и кластеризации образа.

Проблемы обучения рекуррентных нейронных сетей

Затухание градиента ошибки является одной из основных проблем, с которой сталкиваются рекуррентные нейронные сети. Это явление возникает во время обучения, когда градиенты ошибки, используемые для обновления весов сети, становятся настолько малыми, что процесс обучения практически останавливается. Особенно это актуально для длинных последовательностей, где влияние входных данных на выходные данные через множество временных шагов уменьшается до незначительного. Это приводит к тому, что сеть теряет способность учиться из дальних зависимостей в данных, что значительно снижает ее эффективность.

Противоположной проблемой является *взрыв градиента*, который происходит, когда градиенты ошибки велики настолько, что они начинают вызывать нестабильность в процессе обучения. Это может привести к ситуации, когда веса модели обновляются слишком резко, что может привести к неустойчивости и расхождению процесса обучения. Эта проблема также чаще всего возникает при работе с длинными последовательностями и может привести к значительному ухудшению производительности сети.

Алгоритм обучения градиентного типа:

$$\Delta \mathbf{W} = -\gamma \nabla_{\mathbf{W}} E = -\gamma \frac{\partial E}{\partial \mathbf{W}},$$

2.11. Долгая краткосрочная память

(Long Short-Term Memory, LSTM)

Создатели LSTM, Юрген Шмидбауэр и [Сепп](#) Хохрайтер (1997) разработали модель, способную запоминать информацию на продолжительные периоды времени.

Основная проблема эффективного обучения RNN на длинных последовательностях — затухание («паралич сети») или взрыв градиента ошибки. В LSTM сетях проблема решена за счет введения специальных структур, называемых «вентилями» (воротами) (gate – ворота), которые регулируют поток информации внутри сети. С помощью вентиляй сеть может сохранять или забывать информацию, что обеспечивает более стабильный градиент ошибки во время обучения и позволяет учитывать, как краткосрочные, так и долгосрочные зависимости в данных.

Эти вентили реализованы в виде сигмоидных функций для вычисления значения в диапазоне [0 1]. Умножение на это значение используется для частичного допуска или запрещения потока информации в ячейке памяти, для управления памятью.

Основные компоненты LSTM-ячейки и их функции:

- **Входное значение (X_{t-1}, X_t, X_{t+1}):** Последовательные входные данные для каждого временного шага, представлены снизу ячейки.
- **Скрытое состояние (h_{t-1}, h_t, h_{t+1}):** Предыдущее скрытое состояние передается обратно в ячейку на каждом шаге, обозначено горизонтальными стрелками у нижней части ячейки.
- **Внутреннее состояние (A, c_t):** Долгосрочная "память" ячейки, которая сохраняет информацию в течение времени, изображена верхней горизонтальной стрелкой, проходящей через ячейку.
- **Вентиль забывания:** Контролирует меру сохранения значения в памяти. (Представлен логистической функцией активации " σ ", влияющей на внутреннее состояние A).
- **Вентиль входа:** Контролирует меру вхождения нового значения в память. (Включает логистическую функцию " σ " и функцию активации гиперболического тангенса, которые формируют новые значения для обновления состояния A (c_t)).

- **Вентиль выхода:** Контролирует меру того, в какой степени значение, находящееся в памяти, используется при расчёте выходной функции активации для блока. (Обозначены сигмоидной функцией "σ" в центральной части ячейки).

Принцип работы LSTM-слоя представлен на схеме (рис. 2.15).

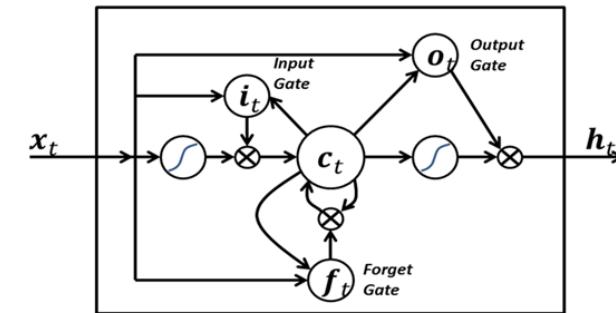
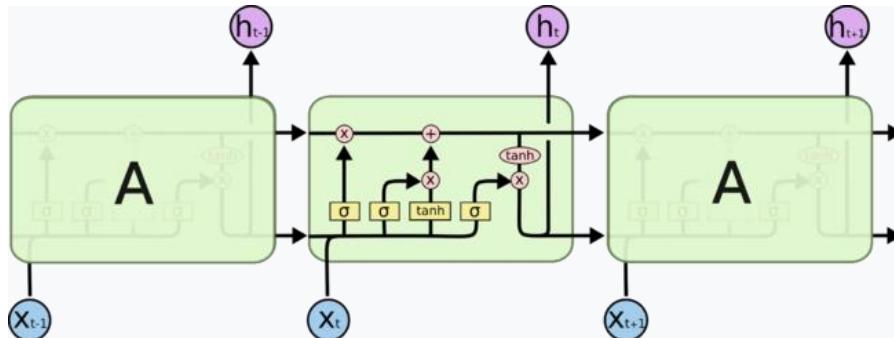


Рис. 2.15

Традиционная LSTM с вентилями забывания^{[2][7]} $c_0 = 0$ и $h_0 = 0$ (○ обозначает произведение Адамара):

$$\begin{aligned}f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\h_t &= o_t \circ \sigma_h(c_t)\end{aligned}$$

Переменные:

- x_t — входной вектор,
- h_t — выходной вектор,
- c_t — вектор состояний,
- W , U и b — матрицы параметров и вектор,
- f_t , i_t и o_t — векторы вентиляй,
 - f_t — вектор вентиля забывания, вес запоминания старой информации,
 - i_t — вектор входного вентиля, вес получения новой информации,
 - o_t — вектор выходного вентиля, кандидат на выход.

W — матрица весовых коэффициентов связи с входным вектором, U — матрица весовых коэффициентов связи с выходом ячейки,

b — вектор весовых коэффициентов смещения.

На протяжении работы ячейки, входные данные X_t и предыдущее скрытое состояние h_{t-1} сначала соединяются (конкатенируются), затем обрабатываются через различные вентили, определяющие, какая информация будет сохранена, обновлена или передана дальше. В результате этих операций LSTM ячейка вычисляет текущее скрытое состояние h_t , которое

затем используется для последующих вычислений или как выходной сигнал модели. Внутреннее состояние A (c_t), несущее в себе информацию для долгосрочной памяти, передается далее, в то время как скрытое состояние, выходящее из вентиля выхода, представляет собой выход h_t текущего временного шага, отображаемый на схеме нижней горизонтальной стрелкой. Состояние ячейки обновляется на каждом шаге времени, получая информацию от вентиля забывания и входного вентиля, что позволяет ему сохранять значимую информацию на протяжении многих шагов.

Указанное свойство определяет название –LSTM – *добавление структуры длинной цепи элементов краткосрочной памяти*.

Один из способов включения долгосрочных зависимостей – модель, работающая в нескольких временных масштабах, так что одни части модели работают в мелком масштабе и могут обрабатывать мелкие детали, а другие, работающие в крупном масштабе, эффективно передают информацию из отдаленного прошлого в настоящее.

Эта способность к долгосрочному запоминанию делает LSTM особенно эффективными в задачах, где важен контекст и последовательность, например, в обработке временных рядов.

Применяется Google для смартфонов.

3. УПРАВЛЕНИЕ ДИНАМИЧЕСКИМИ ОБЪЕКТАМИ С ПРИМЕНЕНИЕМ НЕЙРОННЫХ СЕТЕЙ

3.1. Нейронные сети – универсальные аппроксиматоры

Применение НС как универсальных аппроксиматоров основывается на работах А. Н. Колмогорова и В. И. Арнольда.

Согласно теореме А. Н. Колмогорова (1957) любая непрерывная функция n -переменных на замкнутом ограниченном множестве представима с помощью операций сложения, произведения и суперпозиции непрерывных функций от одной переменной.

$$\text{Пример 3.1. } f(x, y) = \sum_{q=1}^5 [\Psi_q(x) + \varphi_q(y)].$$

R. Hecht-Nielsen на основе теоремы Колмогорова доказал (1987), что для реализации такой функции необходима сеть прямого распространения с двумя скрытыми слоями, имеющая n ИН в 1-м слое и $2n+1$ – во 2-м скрытом слое. В качестве функции активации может быть использована практически произвольная дважды непрерывно дифференцируемая функция от одной переменной.

Замечание.

Результат получен для сети с нейронами традиционной структуры, содержащими суммирующий блок и блок преобразования сигнала с помощью функций активации (1.1), (1.2). Для радиальной сети, включающей радиальные нейроны (1.12), нужен один скрытый слой.

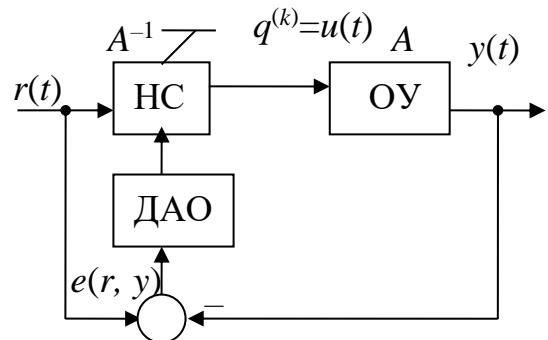
Аппроксимирующее свойство НС используется для решения ряда практических задач:

- НС – регулятор: r – вход, q – выход регулятора.
- НС – идентификатор состояния: r – вход и выход объекта управления, q – недоступные измерению переменные состояния.
- НС – прогнозирующее устройство. Предсказание будущих значений процесса q по прошлым r .

- Использование НС для распознавания и классификации образов: r – образ, подлежащий распознаванию, q – номер класса, к которому данный образ принадлежит.
- Кодирование и декодирование произвольных образов.

3.2. Типовые модели включения НС в систему управления

❖ Схема с "инверсной" моделью объекта управления представлена на рис. 3.1, где обозначено: НС – нейронная сеть,



выполняющая функции регулятора; ОУ – объект управления; ДАО – динамический алгоритм обучения; $q^{(k)}$ – выход нейронной сети. Если сеть обучена с нулевой ошибкой $e(r, y) = r(t) - y(t) = 0$, т. е. задающее воздействие $r(t)$ точно воспроизводится на выходе $y(t)$ объекта управления, заданного оператором A , то идеальное управляющее воздействие $u(t) = A^{-1}r(t)$, т. е. сеть отображает инверсный оператор A^{-1} . Таким образом, НС, обученная на достижение нулевой ошибки, ведет себя как инверсная модель объекта управления.

Рис. 3.1.

❖ Схема с "прямой" моделью объекта управления (рис. 3.2) является схемой с настраиваемой моделью, которая используется для идентификации нелинейных объектов.

Проблема идентификации сводится к построению такой параметризованной модели, чтобы реакции объекта управления $y(t)$ и модели $\hat{y}(t)$ совпадали с точностью до ошибки $e(r, y) = y(t) - \hat{y}(t)$.

Настраиваемая модель может быть реализована с помощью НС прямого распространения и рекуррентной сети.

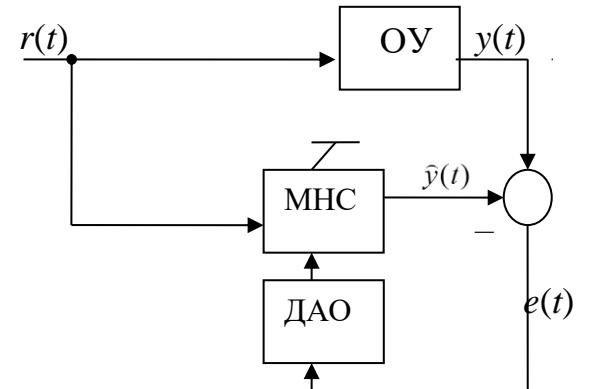


Рис. 3.2

Замечание

Нейронные сети, в обучении которых используются входные (управляющие) $u(k)$ и выходные $y(k+1)$ сигналы объекта управления, часто называются в литературе «нейроэмулаторы». Инверсная модель ОУ – «инверсный нейроэмулатор» (specialised inverse neurocontrol при обучении on-line; generalized inverse neurocontrol, при обучении off-line).

Прямая модель ОУ – «прямой нейроэмулатор». Нейросетевое управление – нейроуправление.

Пример 3.2. Нейронная сеть – универсальный аппроксиматор.

Формирование функции с помощью персептрана

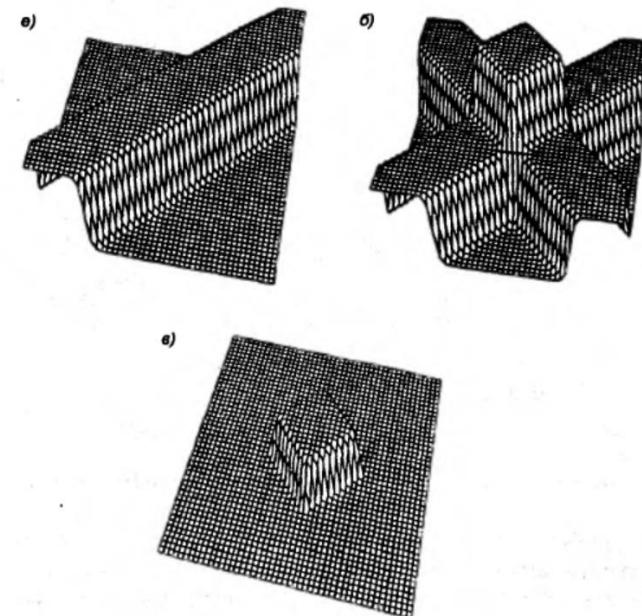
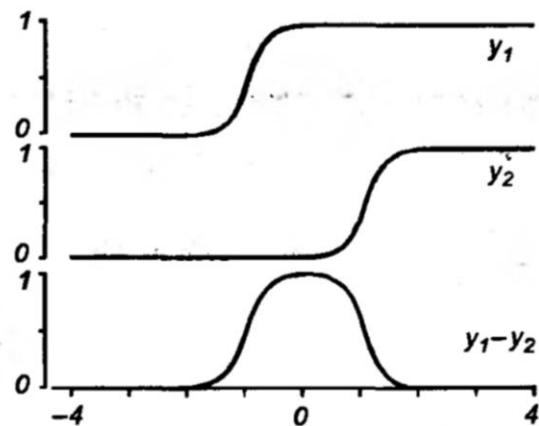


Рис. 3.3

Две сдвинутые относительно друг друга идентичные сигмоидные функции создают в результате вычитания импульс с длительностью, пропорциональной разности смещения этих функций. Подбором параметров сети можно сформировать импульс, которые будет иметь требуемую ширину и крутизну.

Формирование функции с использованием RBF сети

Пример представления функции в виде наложения (линейной суперпозиции) четырех радиальных одномерных базисных функций с одинаковыми дисперсиями σ^2 . Сеть состоит из четырех радиальных нейронов скрытого слоя и одного линейного нейрона выходного слоя. На выходе радиальной сети формируется функция следующего вида

$$q(r) = 0,3f(\|r - 0,5\|) + 0,5f(\|r - 2\|) + 0,4f(\|r - 4\|) - 0,3f(\|r - 5\|),$$

представленная в виде взвешенной суммы отдельных функций Гаусса с опорными точками $c_1 = 0,5$, $c_2 = 2$, $c_3 = 4$, $c_4 = 5$ и весовыми коэффициентами выходного слоя $W_1^{(2)} = 0,3$, $W_2^{(2)} = 0,5$, $w_3^{(2)} = 0,4$ и $W_4^{(2)} = -0,3$; f – радиальная базисная функция.

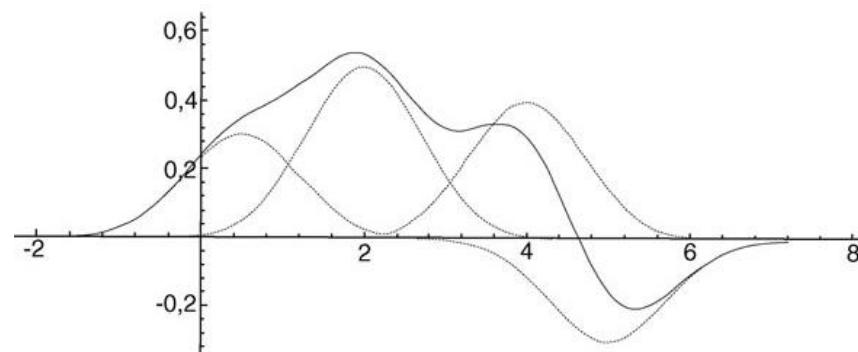


Рис. 3.4

В качестве опорных точек в простейшем случае могут быть использованы образы обучающей последовательности.

3.3. Идентификация нелинейных объектов на основе НС

Реализация настраиваемой модели НС прямого распространения

Описание нелинейного объекта управления (рис. 3.5):

$x(k+1) = \Phi(x(k), u(k))$ – уравнение состояния;

$y(k) = \psi(x(k))$ – уравнение выхода,

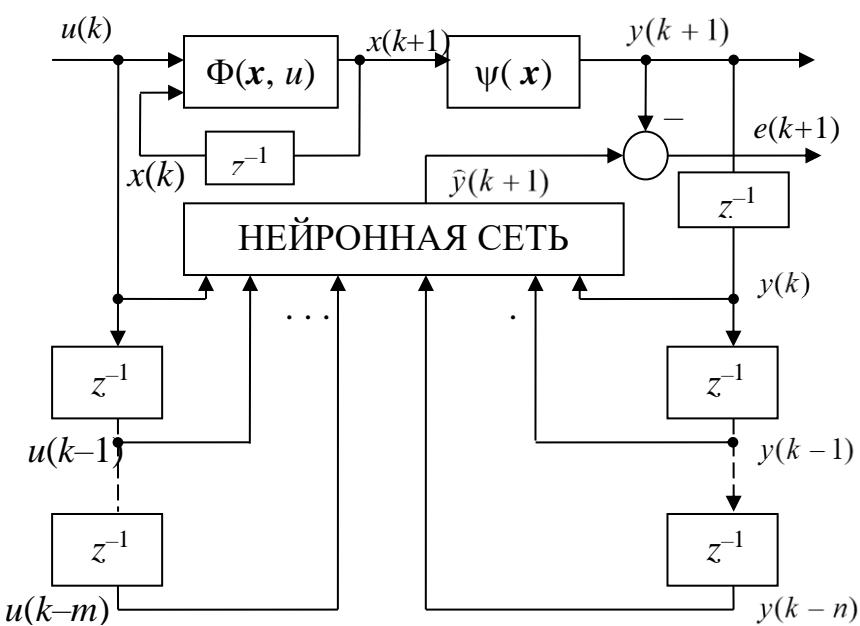


Рис. 3.5

где $x(k) \in R^n$ – вектор состояния; $u(k) \in R^m$ – вектор управления; $y(k) \in R^s$ – вектор измерений; $\Phi(\cdot)$, $\psi(\cdot)$ – векторные нелинейные нестационарные функции, $\Phi(\cdot) \in R^n$, $\psi(\cdot) \in R^s$, m , n – количество элементов запаздывания входного и выходного векторов.
Включение согласно-параллельное

Если ограничиться одним входом и одним выходом, то с учетом элементов запаздывания получим:

- вектор входа сети $u(k) = [u(k) \ u(k-1) \ \dots \ u(k-m)]^T$;
- вектор выхода объекта $y(k+1) = [y(k) \ y(k-1) \ \dots \ y(k-n)]^T$;
- выход сети $\hat{y}(k+1) = F(y(k), u(k))$.

В обучении НС на основе алгоритма обратного распространения ошибок используется ошибка

$$e(k+1) = \hat{y}(k+1) - y(k+1).$$

В сети учитывается запаздывание входных и выходных сигналов объекта управления, которые объединяются во входной вектор односторонней сети. Вход-выходные сигналы предыдущих временных циклов, рассматриваемых как заранее заданные, которые увеличивают размер входного вектора сети.

Полученная сеть может использоваться вместо объекта управления при различных экспериментах.

Замечание

Для использования статической нейронной сети в системе управления необходимо предварительно выполнить, в соответствии с заданием, обучение сети. При включении в систему управления восстанавливается выходное значение сети: $q(k) \approx u(k)$ при $\tilde{r}(k) \approx r(k)$, где r, u – векторы обучающей выборки. Такой режим работы называется автономным (off-line).

Для обучения в реальном масштабе времени (on-line) необходимо ввести в сеть динамические (инерционные) звенья. Задержка необходима для сходимости алгоритма обучения. Время обучения – аналог времени адаптации в адаптивных системах.

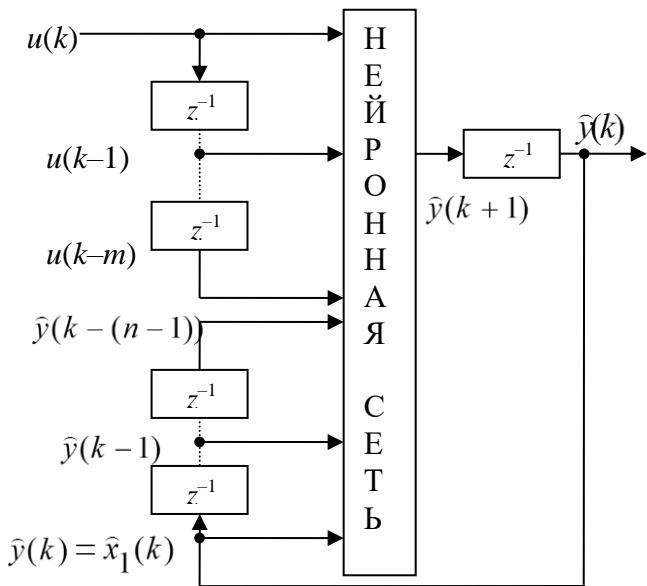
Основные способы введения динамических звеньев в структуру НС:

1. Введение динамических звеньев в структуру ИН.
2. Введение временных задержек в рекуррентных сетях.
3. Введение динамики в алгоритм обучения.

В рекуррентных сетях в каждом контуре присутствует единичный элемент задержки z^{-1} . Обратные связи исходят из выходного слоя (рекуррентный многослойный персепtron) либо из **скрытых слоев (НС Элмана)**.

Реализация настраиваемой модели рекуррентной НС

В сети учитывается запаздывание входных сигналов объекта управления и выходных сигналов сети, которые объединяются во входной вектор рекуррентной сети.



Если ограничиться одним входом и одним выходом, то описание сети (рис. 3.6)

будет иметь следующий вид:

- вектор состояния сети

$$\hat{x}(k) = [\hat{y}(k) \ z^{-1}\hat{y}(k) \ ... \ z^{-(n-1)}\hat{y}(k)]^T = [\hat{x}_1(k) \ \hat{x}_2(k) \ ... \ \hat{x}_n(k)]^T;$$

- выход сети

$$\hat{y}(k+1) = F(\hat{y}(k), z^{-1}\hat{y}(k), \dots, z^{-(n-1)}\hat{y}(k), u(k), z^{-1}u(k), \dots, z^{-m}u(k), W_i^{(l)}) \quad l = \overline{1, K},$$

где K – число слоев НС; $i = \overline{1, n_l}$, n_l – число ИН в l слое.

Рис. 3.6 Результаты идентификации – настройка весовых коэффициентов сети $W_i^{(l)}$ и оценка вектора состояния $\hat{x}(k)$.

3.4. Функциональные структуры систем управления с НС

1. Включение НС вместо четкого регулятора

После обучения нейронная сеть в точности воспроизводит функции исходного регулятора. Например, схема, подобная структуре системы управления с ПИД-регулятором (рис. 3.7). В НС используется значение ошибки $e(t)$, задержанное на 1 такт и на 2 такта. Поскольку ошибка может быть "положительной" и "отрицательной", в регуляторе целесообразно использовать функции активации: гиперболический тангенс (1.6) и линейную функцию с насыщением. Схема обучения и управления с использованием нейросетевого регулятора показаны на рис. 3.8.

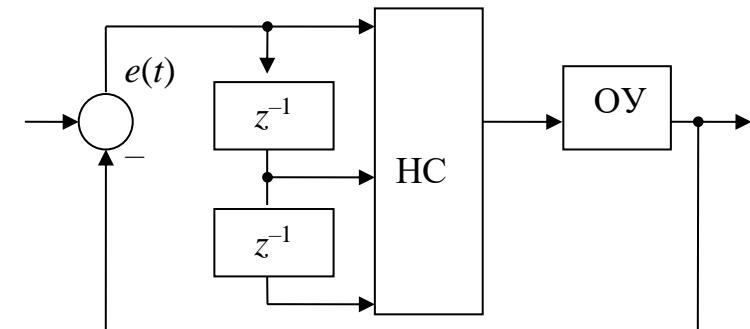


Рис. 3.7

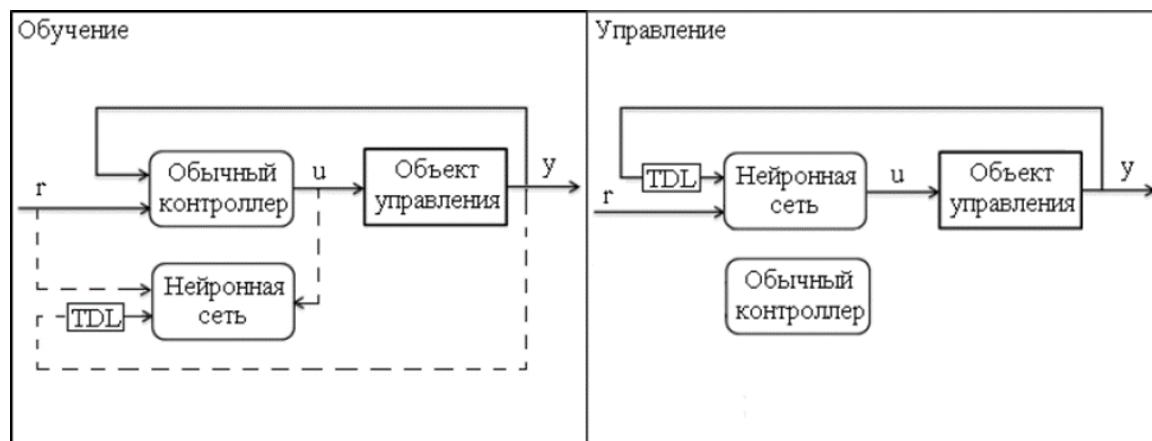


Рис. 3.8.

Использован модуль линии задержек «TDL» (Tapped Delay Line).
В обучающей выборке в качестве примеров динамики регулятора может быть использована запись поведения человека-оператора.

Замечание

Нейросетевые САУ с включением НС вместо обычного регулятора называют «подражающее нейроуправление» (Neurocontrol learning based on mimic).

2. Инверсное нейроуправление

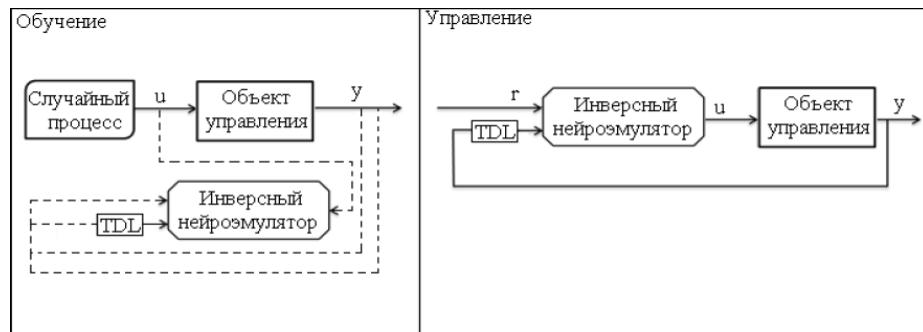
2.1 *Обобщенное инверсное нейроуправление*. (Generalized Inverse Neurocontrol). Обучение сети в автономном режиме.

Обучающая выборка (рис. 3.9):

вектор входа – $y(k+1) = [y(k) \ y(k-1) \ \dots \ y(k-n)]^T$;

вектор выхода объекта управления – $u(k) = [u(k) \ u(k-1) \ \dots \ u(k-m)]^T$.

Для обучения нейронной сети используют метод обратного распространения ошибки.



Возможны два способа подключения: замкнутый и разомкнутый.

Модификация обобщенного инверсного управления: в качестве задания вместо одного целевого значения подается задающее воздействие, которое включает задание на L тактов вперед:

$$r(k) = [r(k) \ r(k+1) \ \dots \ r(k+L)]^T.$$

При разомкнутом подключении на вход сети поступают

Рис. 3.9

только значения входного воздействия с задержкой

$$\mathbf{r}(k) = [r(k) \ r(k-1) \dots r(k-m)]^T.$$

Разомкнутая нейросетевая система обладает высоким быстродействием, но качество управления оказывается низким.

Преимущества обобщенного инверсного нейроуправления

- 1) Обучения НС в автономном режиме.
- 2) Отсутствие необходимости в точной математической модели объекта управления.

Недостатки

- 1) Сложность формирования обучающей выборки из-за необходимости подбора идентифицирующего случайногопроцесса, подаваемого на вход системы для выявления особенностей объекта управления.
- 2) Низкое качество работы в тех случаях, когда обратная модель объекта управления оказывается неоднозначной функцией, что вызывает противоречия в обучающей выборке.

2.2. Специализированное инверсное управление (Specialised Inverse Neurocontrol)

Нейронная сеть обучается в реальном режиме времени, используя в качестве аргумента ошибку управления. Обучение – по методу наискорейшего спуска. Сложность реализации схемы вызвана необходимостью знания точной модели объекта

управления. Для решения указанной проблемы вводят сеть в виде прямой модели объекта управления, которая выполняет роль идентификатора состояния. Качество управления выше, чем в случае 2.1.

3. Адаптивная система управления с прямой и инверсной моделями объекта управления

Сеть НС₁ (рис. 3.10) – регулятора, инверсная модель ОУ. В динамическом алгоритме обучения НС₁ (ДАО₁) используются входное воздействие $g(t)$ и восстановленный вектор переменных состояния $\hat{x}(t)$.

НС₂ – идентификатор, прямая модель ОУ, выход сети – $\hat{y}(t)$.

Ошибка обучения НС₂ $e(t) = \hat{y}(t) - y(t)$. ДАО₂ – динамический алгоритм обучения НС₂. На рисунке обозначено: $f(t)$ – внешнее возмущение; Φ – фильтр, задающий системе управления желаемую динамику.

Преобразуем структурную схему и исключим контуры обучения (рис. 3.11). Объект управления задан оператором A ;

$v(t)$ – возмущение, приведенное к входу объекта управления.

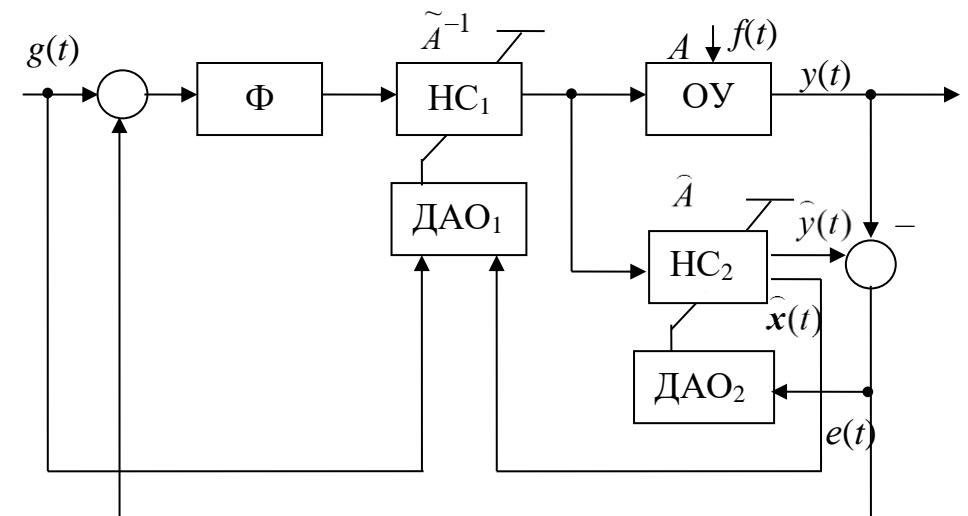


Рис. 3.10

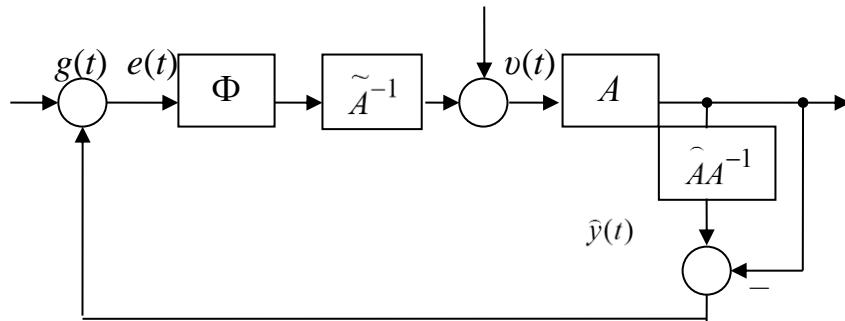


Рис. 3.11

Передаточная функция по возмущающему воздействию $v(t)$, выход – ошибка $e(t)$:

$$W_v(p) = \frac{e}{v} = \frac{A\hat{A}A^{-1} - A}{1 + \Phi\tilde{A}^{-1}A - \Phi\tilde{A}^{-1}A\hat{A}A^{-1}} \Bigg|_{\begin{matrix} \hat{A} = A \\ A = A \end{matrix}} = 0$$

Если сеть НС₂ настроена точно, то ошибка по возмущающему воздействию равна нулю, система управления инвариантна к внешнему воздействию.

Передаточная функция по задающему воздействию

$$W_g(p) = \frac{y}{g} = \frac{\Phi\tilde{A}^{-1}A}{1 + \Phi\tilde{A}^{-1}A - \Phi\tilde{A}^{-1}A\hat{A}A^{-1}} \Bigg|_{\begin{matrix} \tilde{A}^{-1} = A^{-1} \\ \hat{A} = A \end{matrix}} = \Phi$$

Если сети HC_1 и HC_2 настроены точно, то выходная переменная воспроизводит входной сигнал с желаемой динамикой.

Если фильтр равен единичной матрице: $\Phi = I$, выход ОУ равен задающему воздействию: $y(t) = g(t)$.

Адаптивность системы обусловлена минимумом априорной информации об объекте и подстройкой регулятора и модели при действии внешних и параметрических возмущений.

4. Прогнозирующее модельное нейроуправление (NN Predictive Control) (Рис. 3.12)

Минимизирует функционал качества на основе интегральной ошибки, прогнозируемой на L тактов вперед:

$$Q = \sum_{i=L_1}^{L_2} e(k+i)^2 + \rho \sum_{i=0}^{i=L_2} (u(k+i) - u(k+i-1))^2.$$

Здесь e – ошибка управления, $L_1 < L_2$, ρ – вклад изменения управляющего сигнала в общий функционал стоимости Q .

Особенностью схемы является отсутствие нейросетевого регулятора, вместо которого введен оптимизационный модуль, работающий в режиме реального времени.

Оптимизационный модуль получает на такте k целевую траекторию на L тактов вперед, а если ее нет, то L раз дублируется значение текущего задания $r(k+1)$, как задающего воздействия.

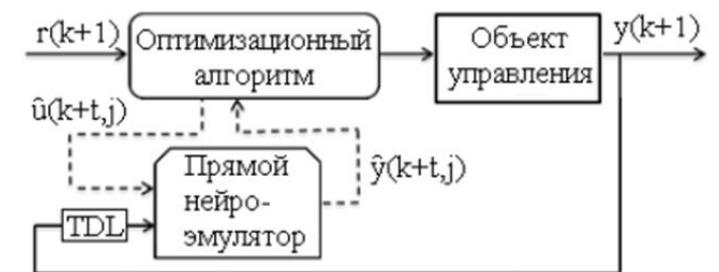


Рис. 3.12

Далее, для выбора оптимального управляющего воздействия, вычисления происходят во внутреннем цикле системы (итерации внутреннего цикла – j). За время одного такта управления оптимизационный модуль подает на вход НС серию различных воздействий $u^*(k + t, j)$, где t – глубина прогнозирования, $0 \leq t \leq L-1$, получает прогноз поведения системы $y^*(k + t + 1, j)$, вычисляет функцию стоимости $Q = f(e, u)$ и определяет оптимальную, в смысле минимизации функционала стоимости, стратегию управления. В итоге, на объект подается управляющий сигнал $u(k, j) = u^*(k + t, j)$.

Недостаток прогнозирующего нейроуправления

Невозможность применения в системах с большой частотой дискретизации, так как оптимизационный алгоритм, работающий в режиме реального времени, за время одного такта не сможет определить наилучшую стратегию действий.

5. Нейроуправление на основе адаптивного анализа («адаптивной критики») (Adaptive Critiks) (3.13)

Системы адаптивного анализа выбирают управляющий сигнал на основе оценок ошибок будущего с бесконечным

горизонтом (подобно системам прогнозирующего модельного управления), функционал качества

$$J(k) = \sum_{i=0}^{\infty} \gamma^i e(k+i)^2,$$

где γ – коэффициент забывания, $0 < \gamma \leq 1$;

$e(k)$ – ошибка управления.

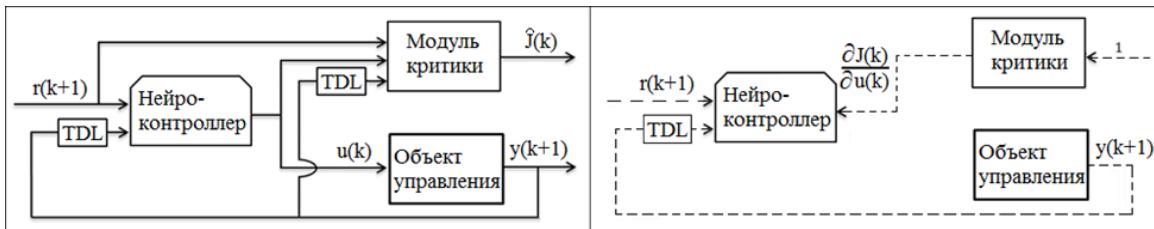


Рис.3.13

Система включает две НС: нейросетевой регулятор, обученный минимизировать функционал $J(k)$, который играет ту же роль, что и ошибка $e(k)$ в методах обучения обратного распространения ошибки, и «модуль критики». «Модуль критики» выполняет аппроксимацию значений функции качества.

Модуль критики, получая на входе вектор $z(k) = [r(k) \ u(k) \ y(k)]^T$, производит оценку функции $J(k)$. На следующем такте вычисляются новые значения $e(k+1)$ и $J(k+1)$, процесс повторяется.

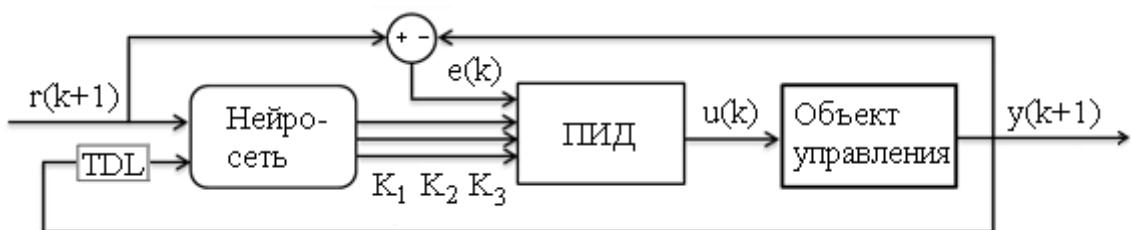
Коррекция весовых коэффициентов выполняется по методу наискорейшего спуска, значение градиента функционала качества рассчитывается по методу обратного распространения ошибки.

Таким образом, на каждом шаге улучшается закон управления, путем обучения нейронной сети, а также повышается способность системы оценивать ситуацию, путем обучения модуля критики.

Обучение – on-line. Сначала происходит обучение модуля критики, потом регулятора.

6. Гибридное нейро-ПИД управление (NNPID)

В гибридных САУ нейронные сети работают совместно с обычными регуляторами (рис. 3.14). Настройка коэффициентов



ПИД – регулятора осуществляется в схемах с последовательно соединенными нейросетевым и ПИД регуляторами в режиме on-line с помощью НС.

На такте k нейронная сеть получает задание $r(k+1)$ и формирует коэффициенты управления ПИД-регулятора K_1, K_2, K_3 , которые поступают на

Рис. 3.14

ПИД-регулятор вместе со значением текущей ошибки управления $e(k)$.

Обучение нейросети происходит в режиме реального времени по ошибке обратной связи, методом наискорейшего спуска.

Градиенты алгоритма вычисляют методом обратного распространения ошибки.

Преимущества гибридного нейро-ПИД управления

- 1) Упрощение эксплуатации вследствие устранения процедуры настройки ПИД-регулятора вручную.
- 2) Кроме того, в случае применения НС с нелинейными функциями активации, ПИД-контроллер фактически превращается в нелинейный регулятор, что обеспечивает высокое качество управления нелинейными объектами.

Недостатки гибридного нейро-ПИД управления

- 1) Сложность оценки устойчивости полученного нелинейного регулятора.
- 2) Необходимость в точной математической модели объекта управления, необходимой для вычисления якобиана объекта управления. Сложность преодолевается введением НС в виде прямой модели объекта управления, используемой в качестве идентификатора состояния.

7. Гибридное параллельное нейроуправления (Рис. 3.15)

В схемах на параллельно соединенные нейросетевой и ПИД-регуляторы подается одинаковое задающее воздействие.

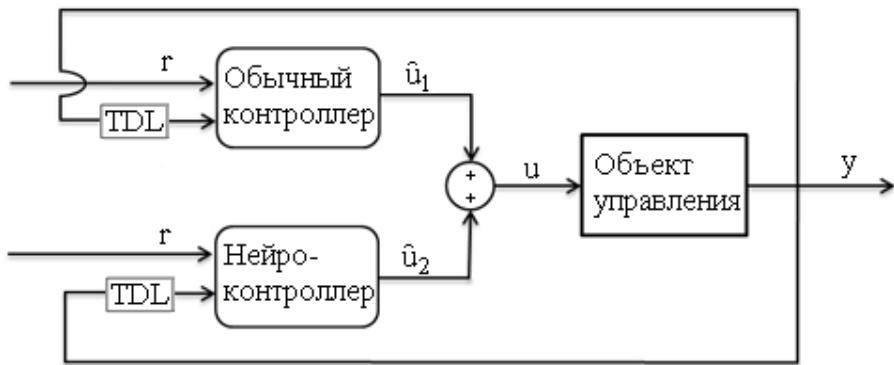


Рис. 3.15

Варианты подключения регуляторов.

- 1) К объекту управления подключается традиционный обычный регулятор, после чего НС обучается управлять замкнутой системой. После обучения нейронная сеть подключается к системе, а управляющие сигналы регуляторов суммируются;
- 2) Сначала выполняется обучение НС (нейросетевой регулятор), после обучения начинает функционировать в штатном режиме.

Далее, для управления замкнутой НС системой, настраивается обычный регулятор. После настройки обычного контроллера, он подключается к системе, управляющий сигнал обоих контроллеров суммируется.

- 2) Области действия обычного и нейросетевого регуляторов разграничиваются. При параллельной работе обоих регуляторов, управляющий сигнал поступает на объект либо от нейросетевого, либо от обычного регулятора.

8. Метод нейросетевой фильтрации внешних возмущений (Рис. 3.16)

В схеме используются две предварительно обученные НС: прямая и инверсная модели объекта управления (инверсный и прямой нейроэмулаторы).

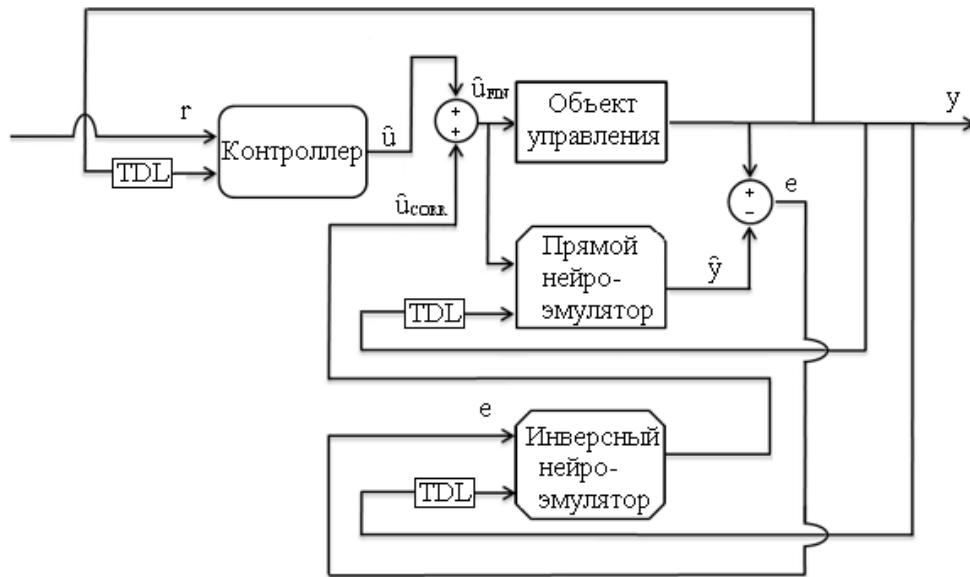


Рис. 3.16

Для подавления нежелательного эффекта, сигнал $e(k)$ поступает на инверсный нейроэмулатор, который рассчитывает корректирующий сигнал $\hat{u}^{(k+1)}_{corr}$ для корректировки управляющего сигнала нейроконтроллера $\hat{u}^{(k+1)}_{fin}$ на следующем такте.

9. Нейроуправление с эталонной моделью на входе САУ (Model Reference Adaptive Control, Neural Adaptive Control)

Задающее воздействие r предварительно подается на эталонную модель, представляющую, как правило, линейную динамическую систему невысокого порядка (рис. 3.17).



Рис.3.17

На выходе эталонной модели формируется траектория r' , которая поступает на нейросетевой регулятор в качестве нового задающего воздействия. ЭМ вводят для повышения устойчивости САУ. Если за один такт требуемое состояние объекта управления не достигнуто, процесс управления становится плохо прогнозируемым.

Эталонная модель подбирают таким образом, чтобы формируемое задающее воздействие было должным образом отработано объектом управления.

Модификация схемы; вместо сети используется ПИД-регулятор, в качестве эталонной модели нейросетевая система типа «адаптивной критики», генерирующая на выходе для отслеживания ПИД-регулятором различные задающие траектории.

10. Адаптивная схема с эталонной моделью объекта управления (рис. 3.18).

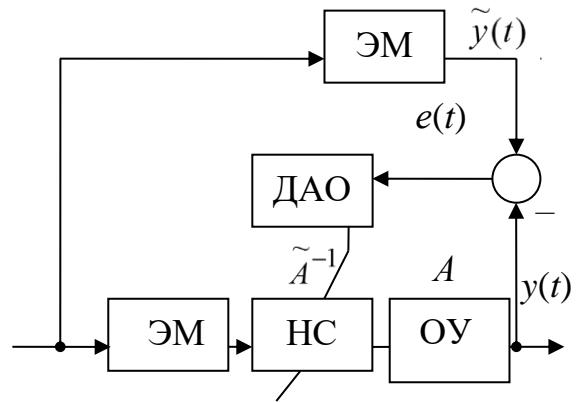


Рис. 3.18

При точной настройке сети ($\tilde{A}^{-1} = A^{-1}$) система обеспечивает заданное качество управления, которое задается с помощью эталонной модели ЭМ. Сеть выполняет функцию регулятора; включена в контур управления по ошибке между выходными сигналами ЭМ, и ОУ является инверсной моделью ОУ.

11. Адаптивная схема с эталонной моделью и идентификатором состояния в виде настраиваемой модели (рис. 3. 19).

На схеме обозначено: $\tilde{x}(t)$ – вектор переменных состояния ЭМ; $\hat{x}(t)$ – восстанавливаемый с помощью HC_2 вектор состояния ОУ; $e_1(t) = \tilde{x}(t) - \hat{x}(t)$ – ошибка обучения HC_1 (регулятора); $e_2(t) = y(t) - \hat{y}(t)$ – ошибка обучения HC_2 (идентификатора состояния).

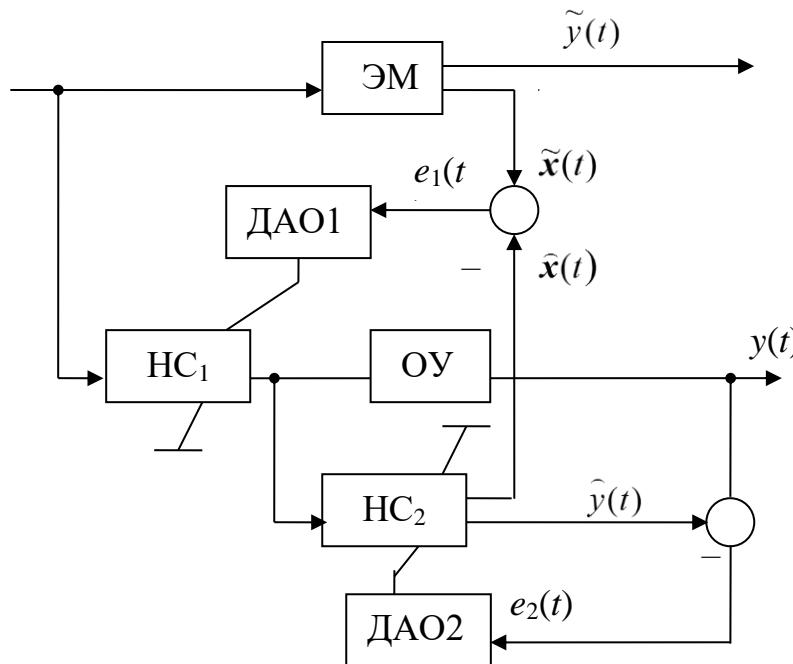


Рис. 3.19

В результате обучения произведение операторов нейронной сети HC_1 и объекта управления равно оператору ЭМ: $A_P A_{OY} = A_{EM}$, где A_P , A_{OY} , A_{EM} – операторы регулятора, объекта управления и эталонной модели соответственно. Выход объекта управления $y(t)$ соответствует желаемой динамике $\tilde{y}(t)$.

12. Многомодульное нейроуправление (Multiple Paired Forward and Inverse Models, Multiple Switched Models)

В рамках многомодульного подхода, исходная задача разделяется на отдельные подзадачи (Рис. 3.20).

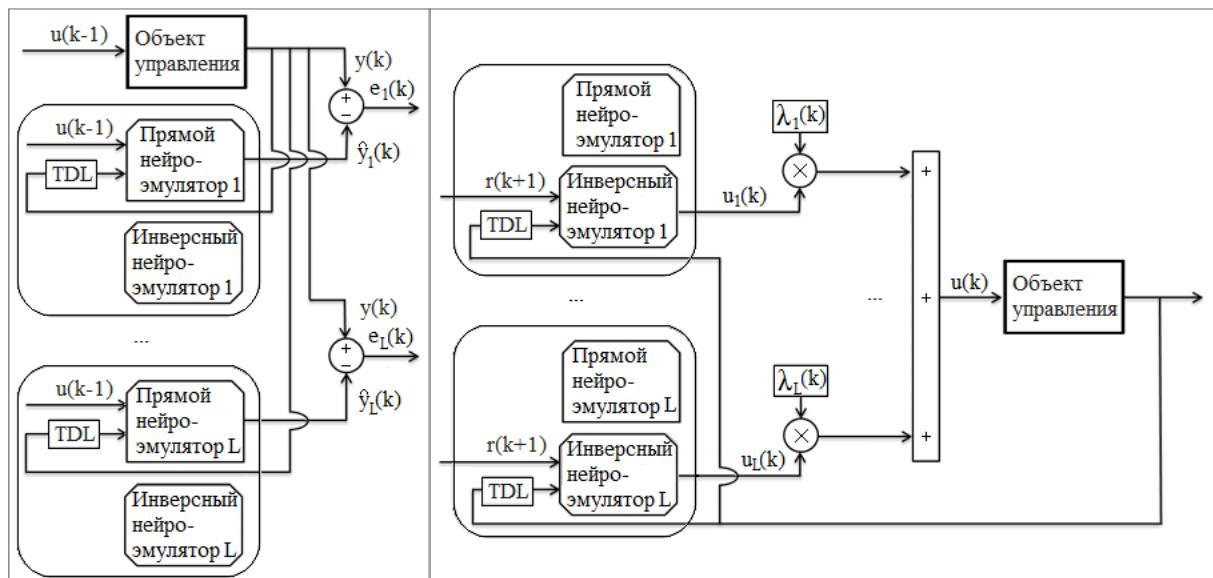


Рис. 3.20

Схема многомодульного нейроуправления на основе пар прямых и инверсных моделей.

2 этапа функционирования:

- 1) переоценки коэффициентов «ответственности» модулей;
- 2) коллективного управления модулями на основе вычисленных коэффициентов «ответственности».

Обучение прямого нейроэмитатора (прямой модели объекта управления) производится на основе метода обратного распространения ошибки. Инверсный нейроэмитатора (инверсной модели объекта управления) обучается по схеме обобщенного инверсного нейроуправления (рис. 3.9).

Каждая пара нейроэмитаторов обучается на своем, существенно отличном, примере динамики объекта управления и специализируется именно на нем.

Итоговый управляющий сигнал $u(k)$ представляет собой взвешенную сумму управляющих сигналов отдельных модулей, при этом управляющий сигнал каждого модуля обеспечивает вклад, пропорциональный коэффициенту «ответственности» соответствующего модуля.

4. ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ

4.1. Генетические алгоритмы

Теория эволюционных алгоритмов основана на работах Холланда Д. (Holland D.), Растрогина Л.А., Букатовой И.Л., Цетлина М.Л., Неймарка Ю.И. и др.

Основные отличия генетических алгоритмов

Генетический алгоритм представляет собой комбинированный метод, в котором сочетаются две группы методов оптимизации: *поисковые алгоритмы стохастической оптимизации*, основанные на механизмах генетики и *градиентные алгоритмы*.

Задача: найти решение x^* , при котором

$$f^* = f(x^*) = \min f(x),$$

где $x^* \in X$, $x_{\min} < x^* < x_{\max}$, $f(x)$ – целевая функция, функции приспособленности.

Отличия генетического алгоритма:

- ✓ оперирует закодированным множеством параметров, а не с самими параметрами;
- ✓ находит популяцию точек, а не отдельную точку;
- ✓ использует значение целевой функции, а не ее производную или другие вспомогательные значения;
- ✓ в генетических алгоритмах применяется вероятностное правило перехода, а не детерминистическое.

Преимущества генетических алгоритмов:

- не требуют никакой информации о поведении функции (например, дифференцируемости и непрерывности);
- относительно стойки к попаданию в локальные оптимумы;
- пригодны для решения крупномасштабных проблем оптимизации;
- просты в реализации.

Недостатки генетических алгоритмов:

- не для всех задач удается найти оптимальное кодирование параметров;
- не гарантируют нахождения глобального оптимума.

Основные заимствованные из генетики понятия

Эволюция – это процесс оптимизации всех живых организмов, каждый биологический вид изменяется для того, чтобы ***наилучшим образом приспособиться*** к окружающей среде.

Молекулы ДНК (дезоксирибонуклеиновой кислоты) – генетический код индивидуума (***особи***).

Хромосома – молекула ДНК.

Ген – часть хромосомы, характеризующая определенное свойство качества особи (цвет глаз, тип волос и т. д.)

Аллель – значения гена.

Популяция – несколько индивидуумов со случайным набором хромосом.

Гены располагаются в различных позициях или ***локусах*** хромосомы, и принимают значения, называемые ***аллелями***.

Если рассматривать хромосому как бинарную строку: ген – бит, локус – его позиция в строке, и аллель – его значение (0 или 1).

Генетические операторы

Скрещивание, кроссовер (cross-over) – хромосомы предков делятся на две части, а затем обмениваются своими половинками.

Мутация – изменения некоторых ген одного из родителей из-за радиоактивности или других влияний. Измененные гены передаются потомку и придают ему новые свойства.

Если эти новые свойства **полезны**, они, скорее всего, сохраняются в данном виде; при этом произойдет скачкообразное **повышение приспособленности** вида.

Отбор – процесс формирования новой популяции из старой, после чего старая популяция погибает. В генетическом алгоритме отбор связан с принципами естественного отбора в природе.

Панмиксия – каждому члену популяции сопоставляется случайное целое число на отрезке $[1; n]$, где n – количество особей в популяции. Эти числа рассматриваются, как номера особей, которые примут участие в скрещивании.

Преимущество: универсален для решения различных классов задач.

Недостаток: эффективность алгоритма, реализующего такой подход, снижается с ростом численности популяции.

Инбридинг – первый родитель выбирается случайным образом, а вторым родителем является член популяции ближайший к первому (смысле минимального расстояния Хемминга (для бинарных строк) или евклидова расстояния между двумя вещественными векторами).

Аутбридинг – родители формируются из максимально далеких особей.

Наиболее распространенный способ отбора – **селекция**, при которой родителями могут стать особи, значение приспособленности которых не меньше пороговой величины.

Пропорциональный метод селекции.

Сначала, пропорциональный отбор назначает каждой структуре вероятность $P_s(i)$ равную отношению ее приспособленности к суммарной приспособленности популяции:

$$P_s(i) = \frac{f(i)}{\sum_{i=1}^N f(i)} \quad (4.1)$$

В результате процесса селекции создается *родительская популяция*, также называемая *родительский пулом* (mating pool) с численностью N , равной численности текущей популяции.

Основные понятия генетических алгоритмов:

Хромосома	Вектор (последовательность) из нулей и единиц (битовая строка)
Ген	Часть хромосомы.
Аллель	Значение гена.
Локус	Позиция гена.
Индивидуум = генетический код	Набор хромосом (вариант решения задачи).
Кроссовер	Операция, при которой две хромосомы обмениваются своими частями.
Мутация	Случайное изменение одной или нескольких позиций в хромосоме.
Отбор	Процесс формирования новой популяции из старой.

Приспособленность индивидуума	Значение целевой функции на этом индивидууме.
----------------------------------	---

Выживание наиболее приспособленных

Популяция следующего поколения формируется в соответствии с целевой функцией. Чем приспособленнее индивидуум, тем больше вероятность его участия в кроссовере, т.е. размножении.

Механизмы *скрещивания и мутации* соответствуют *поисковой* части метода, а *отбор* лучших решений – *градиентному спуску*, что обеспечивает высокую эффективность генетического поиска для решения задач оптимизации.

Классический генетический алгоритм

Основной (классический) генетический алгоритм состоит из следующих шагов:

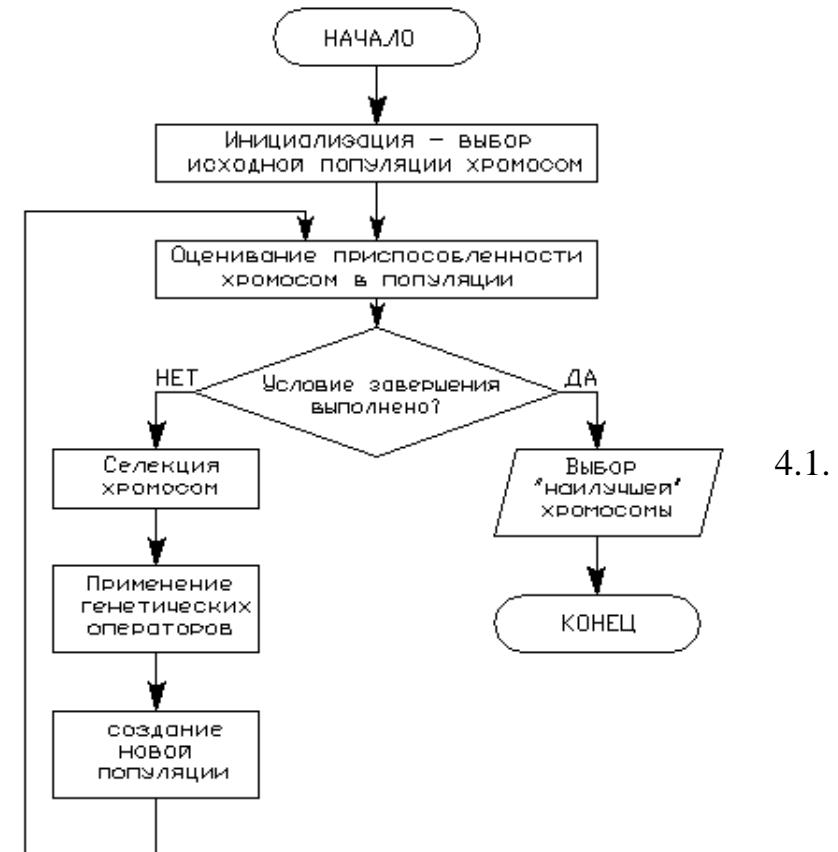
1. инициализация, или выбор исходной популяции хромосом;
2. оценка приспособленности хромосом в популяции;
3. проверка условия остановки алгоритма,
4. селекция хромосом;
5. применение генетических операторов;
6. формирование новой популяции;
7. выбор «наилучшей» хромосомы.

Блок-схема основного генетического алгоритма изображена на рис.

Рассмотрим конкретные этапы этого алгоритма более подробно с использованием дополнительных подробностей, представленных на рис.4.2.

Инициализация, т.е. формирование исходной популяции, заключается в выборе заданного количества хромосом (особей). Простой генетический алгоритм случайным образом генерирует начальную популяцию структур.

Оценивание приспособленности хромосом в популяции состоит в для каждой хромосомы этой популяции. Рис. 4.1. Блок-схема генетического алгоритма



расчете функции приспособленности

Одноточечный *кроссовер* работает следующим образом. Сначала, случайным образом выбирается одна из $L-1$ точек разрыва. (Точка разрыва - участок между соседними битами в строке.) Обе родительские структуры разрываются на два сегмента по этой точке. Затем, соответствующие сегменты различных родителей склеиваются и получаются два генотипа ПОТОМКОВ.

Пример 4.1 Один родитель состоит из 10 нолей, а другой - из 10 единиц. Пусть из 9 возможных точек разрыва выбрана точка 3. Родители и их потомки показаны ниже.

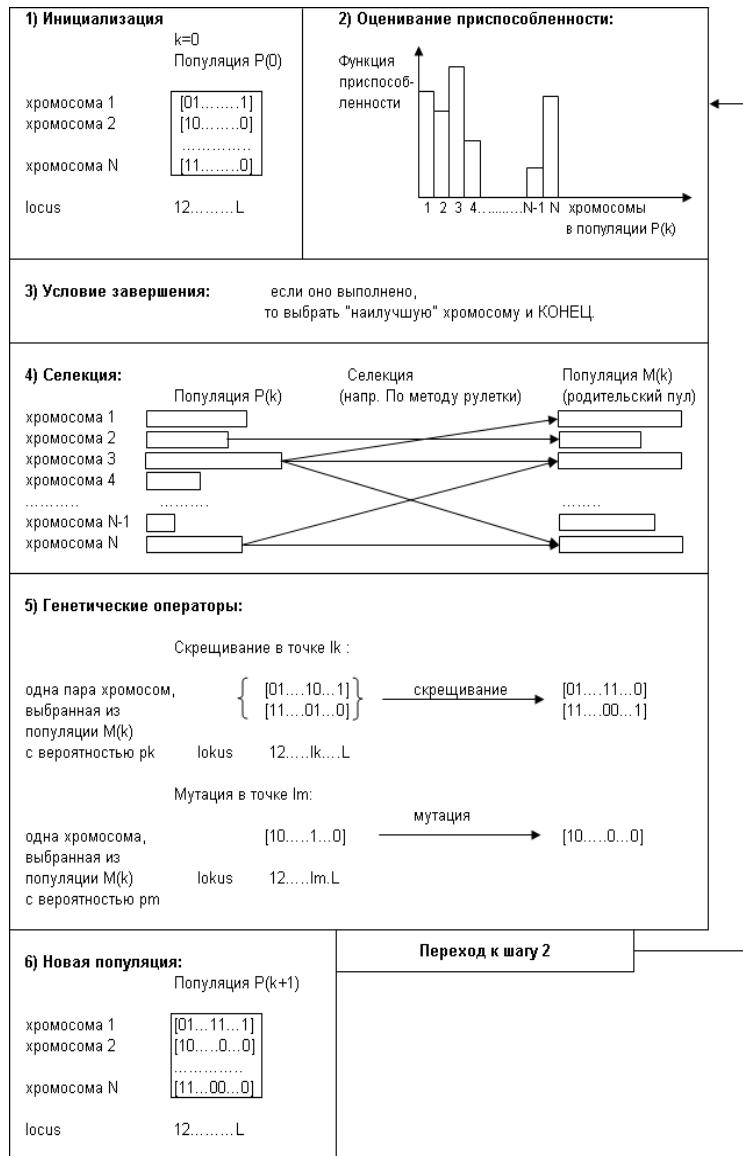


Рис. 4.2. Схема выполнения генетического алгоритма

Кроссовер						
1 Родитель	0000000000	000~0000000	-->	111~0000000	1110000000	Потомок 1
2 Родитель	1111111111	111~1111111	-->	000~1111111	0001111111	Потомок 2

После того, как закончится стадия кроссовера, выполняются операторы мутации. В каждой строке, которая подвергается мутации, каждый бит с вероятностью P_m изменяется на противоположный.

Пример поясняющий мутацию изображен на рисунке 4.2. Популяция, полученная после мутации, записывает поверх старой и этим цикл одного поколения завершается. Последующие поколения обрабатываются таким же образом: отбор, кроссовер и мутация.

Проверка условия остановки алгоритма. Определение условия остановки генетического алгоритма зависит от его конкретного применения. Если условие остановки выполнено, то производится переход к завершающему этапу выбора «наилучшей» хромосомы. В противном случае на следующем шаге выполняется отбор (селекция).

Схема (schema). Строящие блоки

Генетический алгоритм в процессе функционирования обрабатывает схемы (шимы), которые представляют шаблоны подобия между строками.

Схема (шима) – это строка длины l (что и длина любой строки популяции), состоящая из знаков алфавита на фиксированных позициях (локусах) $\{0; 1; *\}$, где $\{*\}$ – неопределенный символ. Каждая шима определяет множество всех бинарных строк длины l , имеющих в соответствующих позициях либо 0, либо 1, в зависимости от того, какой бит находится в соответствующей позиции самой шимы.

Пример 4.2.

Шима, $10**1$, определяет собой множество из четырех пятибитовых строк $\{10001; 10011; 10101; 10111\}$.

Приспособленность шимы определяется как средняя приспособленность примеров, которые ее содержат.

После процедуры отбора остаются только строки с более высокой приспособленностью. Следовательно, строки, которые являются примерами шим с высокой приспособленностью, выбираются чаще. Кроссовер реже разрушает шимы с более короткой определенной *длиной*, а мутация реже разрушает шимы с *низким порядком*. Поэтому такие шимы имеют больше шансов переходить из поколения в поколение.

Порядок схемы $o(S)$ – число фиксированных битов.

Длина (хват) схемы (defining length) $d(S)$ – расстояние между первым и последним фиксированными битами в схеме.

Правило репродукции

Изменение от популяции к популяции количества хромосом, соответствующих данной схеме,

$$[c(S, k+1)] \geq c(S, k) \frac{F(S, k)}{\bar{F}(k)} \left(1 - p_c \frac{d(S)}{L-1}\right) (1 - p_m)^{o(S)},$$

где S – рассматриваемая схема (шима, schema), заданная популяция $P(k)$, $c(S, k)$ – количество хромосом популяции $P(k)$, соответствующих схеме S ; $F(S, k)$ – среднее значение функции приспособленности хромосом из популяции $P(k)$, которые соответствуют схеме S , (приспособленность схемы S на k -ой итерации), $\bar{F}(k)$ – среднее значение функции приспособленности хромосом из популяции $P(k)$, p_m – вероятность мутации, $(1 - p_m)^{o(S)}$ – влияние мутации, p_c – вероятность скрещивания, $\left(1 - p_c \frac{d(S)}{L-1}\right)$ – влияние скрещивания.

$$\text{Для больших популяций } c(S, k+1) = c(S, k) \frac{F(S, k)}{\bar{F}(k)}.$$

Если допустить, что схема S имеет приспособленность на ε % выше средней, т.е. $F(S, k) = \bar{F}(k) + \varepsilon \bar{F}(k)$.

Если приспособленность не изменяется во времени,

$$\begin{aligned} c(S, k) &= c(S, 0)(1 + \varepsilon)^k, \\ \varepsilon &= (F(S, k) - \bar{F}(k)) / \bar{F}(k), \end{aligned} \tag{4.1}$$

$\varepsilon > 0$ для схемы с приспособленностью выше средней и $\varepsilon < 0$ в противном случае. Равенство (4.1) – геометрическая прогрессия. В процесс репродукции схемы, оказавшиеся лучше (хуже) средних, выбираются на очередных итерациях генетического алгоритма в показательно возрастающих (убывающих) количествах.

4.2. Пчелиный алгоритм

Относится к классу методов, получивших название «роевой интеллект». Пчелиный алгоритм основан на поведении пчел.

С вычислительной точки зрения роевые алгоритмы – это стохастические методы поиска, в которых эффективно сочетаются нахождение новых решений и улучшение существующих решений.

Согласно алгоритму каждая особь находится в пространстве поиска возможных решений задачи. Особь меняет свое положение, основываясь на знании трех факторов: текущего положения (*приспособленности*), *предыдущих состояний* данной особи, *предыдущих состояний соседей* данной особи.

Основные шаги работы

1. Случайным образом генерируется начальный набор из n возможных решений (n пчел).
2. Для каждого решения вычисляется его пригодность, которая соответствует значению целевой функции.
3. В зависимости от значения целевой функции выделяются два вида возможных решений – лучшие (l) и перспективные (p), которым соответствует определенное число возможных решений.

4. В окрестности лучших решений случайным образом генерируются N новых решений, а в окрестности перспективных M новых решений, причем на каждое лучшее решение должно приходиться больше новых решений, чем на перспективное ($N > M$).

Окрестность, которая определяет область, в которой генерируются новые возможные решения, можно уменьшать по мере увеличения номера итерации, чтобы постепенно решение сходилось к глобальному экстремуму. Но если область уменьшать слишком быстро, то решение может застрять в локальном экстремуме.

5. Случайным образом генерируются $(n-l-p)$ новых решений, взамен наименее пригодных решений.

6. Переход на шаг 2.

Алгоритм повторяется до тех пор, пока не сработает какой-нибудь из критериев остановки. Критерием остановки может служить, например, предельное число итераций, точность по отношению к заданной нижней оценке целевой функции или наличие того факта, что все пчелы получают одни и те же решения.

Пример 4.3.

Найти минимум функции $f = x^2 + y$, $x \in [0;20]$, $y \in [0;20]$

Параметры пчелиного алгоритма: 5 начальных решений, 1 лучшее, 2 перспективных, 3 решения в окрестности лучших решений, 1 в окрестности перспективных, окрестность 3.

1. Случайным образом генерируется начальный набор из 5 возможных решений (x,y) , Табл.4.1.

Таблица 4.1

x	y	f(x,y)
5	15	40

3	5	14
14	1	197
3	0	9
17	15	304

2. Для каждого решения вычисляется его пригодность, которая соответствует значению целевой функции.

3. В зависимости от значения целевой функции выделяются два вида возможных решений – 1 лучшее и 2 перспективных.

4 – (3,0) – лучшее; 2 – (3,5) – перспективное 1; 3 – (5,15) – перспективное 2

4. В окрестности лучшего решения случайным образом генерируется 3 новых решения (2,1); (6,3); (3,3), а в окрестности перспективных по 1 новому решению (2,3); (3,12).

5. Случайным образом генерируются 2 ($5-1-2=2$) новых решения, взамен наименее пригодных решений

(5,20) – взамен решения 3; (17,0) – взамен решения 5.

6. Для каждого решения вычисляется его пригодность (Табл. 4.2)

Таблица 4.2

x	y	f(x,y)
2	1	5
6	3	39
3	3	12
2	3	7
3	12	21

5	20	45
17	0	289

7. Таким образом имеем 1 лучшее решение $(2,1) - 5$ и 2 перспективных $(2,3) - 7$ и $(3,0) - 9$ (аналогично шагу 3).

8. В их окрестностях происходит поиск (аналогично шагу 4):

$(1,1); (0,0); (4,3)$, а в окрестности перспективных по 1 новому решению $(2,0); (6,2)$.

9. Случайным образом генерируются 2 новых решения: $(5,1); (17,15)$.

На 2 итерации имеем следующий набор решений: $(2,1) - 5; (2,3) - 7; (3,0) - 9; (1,1) - 2; (0,0) - 0; (4,3) - 19, (2,0) - 4, (6,2) - 38, (5,1) - 26; (17,15) - 304$, из которых точка $(0,0)$ имеет значение целевой функции равное 0, которое *соответствует глобальному минимуму функции*.

Особенности пчелиного алгоритма

Настройка алгоритма

Выбор параметров алгоритма: начальный набор из n возможных решений;

- количество лучших решений;
- количество перспективных решений;
- количество новых решений в окрестности лучших решений;
- количество новых решений в окрестности перспективных решений;
- размер окрестности.

Важно отметить, что для улучшения работы алгоритма, данные параметры могут изменяться в процессе выполнения.

Пчелиные алгоритмы относятся к стохастическим методам поиска и не могут гарантировать нахождения глобальный экстремум функции. Задачи, для которых не существует специальных алгоритмов решения, могут с большой долей вероятности качественно решаться с помощью ПА. Можно усовершенствовать тот или иной алгоритм с помощью пчелиного алгоритма.

Алгоритмы не требуют дополнительной информации о поведении целевой функции (например, дифференцируемости и непрерывности) и просты в реализации.

4.3. Муравьиный алгоритм

Идея муравьиного алгоритма заключается в моделировании поведения реальных муравьев.

Основные шаги работы

1. Случайным образом в области допустимых значений генерируется набор возможных решений N .
2. Решения сортируются в зависимости от их пригодности (значения целевой функции), от лучшего к худшему.
3. Происходит обновление показателя интенсивности фермента τ_j^i ($\tau_{j0}^i = 1$) для каждой переменной j решения i согласно позиции в отсортированном списке,

$$\tau_j^i = \tau_{j0}^i + \alpha(N_G - i), \quad i < N_G, \quad (4.2)$$

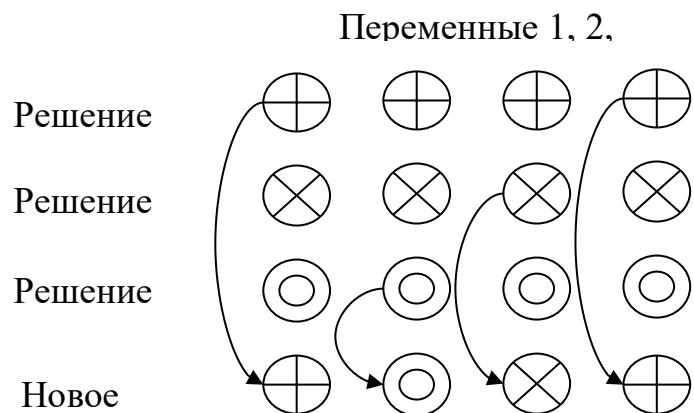
где N_G – число пригодных решений, $\alpha \in [0, 1]$ – вес фермента, характеризующий величину обновления показателя τ_j^i .

4. Каждому решению приписывается значение случайной величины $p \in [0, 1]$. Модификация решения производится, если величина p меньше заданного значения $q \in [0, 1]$ (принимаем $q = 0,9$, чтобы большинство решений участвовали в процессе модификации).

Вероятность для решения i выбрать переменную j определяется случайным выбором из вероятностей p_i^j , где

$$p_i^j = \frac{\tau_i^j}{\sum_{i=1}^N \tau_i^j} \quad (2.3)$$

На рис. 4.3 показан принцип модификации решения i .



5. Отбрасываются худшие решения и случайным образом генерируется такое же количество новых возможных решений.

6. Обновление показателя τ_j^i для каждого решения, согласно новой сортировке

$$\tau_j^i(t+1) = \rho \tau_j^i(t) + \alpha(N_G - i), \quad i < N_G \quad (4.4)$$

Рис.

в противном случае

$$\tau_j^i(t+1) = \rho \tau_j^i(t),$$

где $\tau_j^i(t+1)$ – значение показателя τ_j^i на следующем шаге, $\rho \in [0, 1]$ – интенсивность испарения фермента.

7. Случайный поиск. Чтобы сохранить скорость вычисления алгоритма, случайный поиск применяется только к лучшему решению в некоторой окрестности координат возможного решения.

После нескольких новых решений (d) алгоритм выбирает между лучшим и случайным решением.

8. Переход на шаг 4.

Алгоритм повторяется до тех пор, пока не сработает какой-нибудь из критериев остановки. Критерием остановки может служить, например, предельное число итераций, точность по отношению к заданной нижней оценке целевой функции или наличие того факта, что все решения получают одни и те же значения целевой функции.

Пример 4.3.

Найти минимум функции $f = x^2 + y$, $x \in [0;20]$, $y \in [0;20]$

Параметры муравьиного алгоритма: 5 начальных решений, 4 хороших, 1 плохое, $\alpha = 0,9$ $\rho = 0,7$, окрестность 3.

1. Случайным образом генерируется начальный набор из 5 возможных решений (x,y) и сортируются в зависимости от пригодности (Табл. 4.3)

Таблица 4.3

x	y	f(x,y)
1	3	4
4	2	18
0	20	20

5	4	29
7	1	50

2. Происходит обновление показателя интенсивности фермента τ_j^i ($\tau_{j0}^i = 1$) для каждой переменной j решения i

согласно позиции в отсортированном списке

$$\tau_y^1 = \tau_x^1 = 1 + 0,9(4 - 1) = 3,7$$

$$\tau_y^2 = \tau_x^2 = 1 + 0,9(4 - 2) = 2,8$$

$$\tau_y^3 = \tau_x^3 = 1 + 0,9(4 - 3) = 1,9$$

$$\tau_y^4 = \tau_x^4 = 1 + 0,9(4 - 4) = 1$$

$$\tau_y^5 = \tau_x^5 = 1$$

4. Каждому решению приписывается значение случайной величины $p \in [0, 1]$. Модификация решения производится, если величина p меньше заданного значения $q \in [0, 1]$ (принимаем $q = 0,9$, чтобы большинство решений участвовали в процессе модификации).

Таблица 4.4.

Вероятность для решения

$$p_i^j = \frac{\tau_i^j}{\sum_{i=1}^N \tau_i^j}$$

где

x	y	f(x,y)
1	3	4
0	1	1
4	2	18
0	3	3
4	4	20

i выбрать переменную j
определяется случайным
выбором из вероятностей p_i^j ,

$$p_y^1 = p_x^1 = \frac{3,7}{10,4} = 35,6\% \quad p_y^2 = p_x^2 = 27\%$$

5. Отбрасываются плохие решения и случайным образом генерируется такое же количество новых возможных решений.

$$p_y^3 = p_x^3 = 18,2\% \quad p_y^4 = p_x^4 = 9,6\% \quad p_y^5 = p_x^5 = 9,6\%$$

Плохое решение (4,4), новое взамен него (10,0)

6. Обновление показателя τ_j^i для каждого решения, согласно новой сортировке (Табл. 2.5)

Таблица 4.5

x	y	f(x,y)
0	1	1
0	3	3
1	3	4
4	2	18
10	0	100

$$\tau_y^1 = \tau_x^1 = 0,7 \cdot 2,8 + 0,9(4 - 1) = 4,66$$

$$\tau_y^2 = \tau_x^2 = 0,7 \cdot 1 + 0,9(4 - 2) = 2,5$$

$$\tau_y^3 = \tau_x^3 = 0,7 \cdot 3,7 + 0,9(4 - 3) = 3,49$$

$$\tau_y^4 = \tau_x^4 = 0,7 \cdot 1,9 = 1,33$$

$$\tau_y^5 = \tau_x^5 = 0,7$$

7. Случайный поиск.

Случайный поиск применяем к лучшему решению в окрестности S=3.

Например, получили значения в окрестности (2,3), (1,0), (0,3), которые не улучшили исходное решение (0,1) переходит на следующую итерацию. Переход на Шаг 4 и так далее, пока не будет выполнено заданное число итераций.

Особенности муравьиного алгоритма

Выбор параметров

- начальный набор из n возможных решений;
- количество лучших решений (муравьев);
- количество худших решений (муравьев);
- количество решений, к которым применяется случайный поиск в окрестности;
- размер окрестности участков;
- число решений в окрестности;
- значение веса фермента.

Особенности муравьиного алгоритма:

- ✓ параметры муравьиного алгоритма могут изменяться в процессе его выполнения;
- ✓ не требуется дополнительная информация о поведении целевой функции (например, дифференцируемости и непрерывности);
- ✓ алгоритмы опираются на память о всех решения (обо всей колонии), вместо учета только решений предыдущего поколения;
- ✓ алгоритмы просты в реализации.