МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования
«Витебский государственный технологический университет»

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

**к практическим занятиям
для студентов специальности 1-40 05 01-01 «Информационные системы и технологии (в проектировании и производстве)»
для дневной и заочной на базе ссуз форм обучения**

Витебск
2020

УДК 004.8

Составители:

Е. Б. Дунина, В. А. Жизневский, А. С. Соколова

Рекомендовано к изданию редакционно-издательским советом УО «ВГТУ», протокол № 2 от 30.10.2020.

Искусственный интеллект: методические указания к практическим занятиям / сост. Е. Б. Дунина, В. А. Жизневский, А. С. Соколова. – Витебск : УО «ВГТУ», 2020. – 48 с.

В методических указаниях изложены теоретические сведения и практические задания по дисциплине «Искусственный интеллект». Издание предназначено для студентов специальности 1-40 05 01-01 «Информационные системы и технологии (в проектировании и производстве)» и может быть использовано на практических занятиях и для самостоятельной работы студентов.

УДК 004.8

СОДЕРЖАНИЕ

1 ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ. АЛГЕБРА ВЫСКАЗЫВАНИЙ

Краткие теоретические сведения

Под ***высказыванием*** понимают повествовательное предложение, о котором можно сказать, что оно либо истинно, либо ложно. Например:

1. Каждый студент бюджетник факультета информационных технологий и робототехники получает стипендию.

2. $2 + 3 = 5$.

Первое из этих высказываний является ложным, второе истинным.

Всякое высказывание является либо истинным, либо ложным и не является одновременно тем и другим. Не всякое повествовательное предложение является высказыванием. Нельзя отнести к высказываниям предложение вида: «розы самые красивые цветы», «$k + 2 > 15$». Действительно, одним людям нравятся розы, другим пионы и т. д. Не может быть единого мнения о том, какие цветы являются самыми красивыми. Истинность второго предложения также нельзя определить, так как неизвестно, о каком k идет речь.

Обычно высказывания обозначаются большими латинскими буквами: *A, B, C, …*, а логические действия над ними – с помощью значков операторов. Поскольку высказывание рассматривается как величина, принимающая одно из двух значений: истина или ложь, то будем обозначать эти значения соответственно через 1 и 0.

Пусть даны два произвольных высказывания A и B, истинность которых неизвестна. Рассмотрим логические операции (связки) над высказываниями.

Определение 1.1. ***Отрицанием*** высказывания A называется новое высказывание, истинное тогда и только тогда, когда A ложно. Отрицание обозначается $\neg A$ или $\overline{A}$. Операция отрицания полностью определяется следующей таблицей истинности:

A	$\overline{A}$
1	0
0	1

Определение 1.2. ***Конъюнкция*** высказывания (оператор И, логическое умножение) – сложное высказывание, которое будет истинным только в случае, когда истинны все его составляющие. Конъюнкция обозначается значком «$\wedge$».

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

Определение 1.3. ***Дизъюнкция*** (оператор ИЛИ, логическое сложение) – высказывание, которое ложно, если ложны все составляющие, и истинно во всех остальных случаях. Обозначается значком «$\vee$».

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

Определение 1.4. Высказывание, обозначаемое $A \to B$ («если A то B»), ложно в том и только в том случае, когда A истинно, а B ложно, называется ***импликацией*** с посылкой A и заключением B.

A	B	$A \to B$
1	1	1
1	0	0
0	1	1
0	0	1

Определение 1.5. Высказывание, обозначаемое через «A эквивалентно B», истинное в том и только в том случае, когда A и B имеют одно и то же истинное значение, называется ***эквиваленцией***.

A	B	$A \leftrightarrow B$
1	1	1
1	0	0
0	1	0
0	0	1

Определение 1.6.

1. Всякая буква, обозначающая высказывание, является формулой.
2. Символы 0 и 1 являются формулами.
3. Если A – формула, то $\overline{A}$ – формула.
4. Если A и B – формулы, то $(A \wedge B)$, $(A \vee B)$, $(A \to B)$, $(A \leftrightarrow B)$ – формулы.
5. Формулами являются только те выражения, которые можно получить при помощи п. 1–4.

Замечание. Операции по силе своего действия в порядке убывания располагаются следующим образом $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$.

Пример: в формуле $B \leftrightarrow \neg C \vee D \wedge A$ скобки восстанавливаются следующими шагами:

$$B \leftrightarrow (\neg C) \vee D \wedge A,$$
$$B \leftrightarrow (\neg C) \vee (D \wedge A),$$
$$B \leftrightarrow ((\neg C) \vee (D \wedge A)),$$
$$(B \leftrightarrow ((\neg C) \vee (D \wedge A))).$$

Определение 1.7. Формула логики высказываний, которая принимает значение «истина» при любом распределении значений, входящих в эту формулу элементов, называется ***тождественно истинной формулой, тавтологией или законом логики.***

Определение 1.8. Формула логики высказываний, которая принимает значение «ложь» при любом распределении значений, входящих в эту формулу элементов, называется ***тождественно ложной формулой или противоречием.***

Пример: доказать, что формула $(P \to Q) \wedge (Q \to R) \to (P \to R)$ является тавтологией.

Составим таблицы истинности для заданных формул:

P	Q	R	$P \to Q$	$Q \to R$	$P \to R$	$(P \to Q) \wedge (Q \to R) \to (P \to R)$
1	1	1	1	1	1	1
1	1	0	1	0	0	1
1	0	1	0	1	1	1
1	0	0	0	1	0	1
0	1	1	1	1	1	1
0	1	0	1	0	1	1
0	0	1	1	1	1	1
0	0	0	1	1	1	1

Задания для контролируемой самостоятельной работы

1.1. Составить таблицу истинности для каждой из формул:

а) $p \to q \leftrightarrow \neg p \vee q$;

б) $p \to \neg(q \wedge r)$;

в) $r \to (p \to q)$;

г) $(p \to q) \to (s \wedge \neg s) \to p \vee s)$.

1.2. Восстановить скобки в логических формулах и составить для них таблицы истинности:

а) $A \vee B \wedge C \vee \neg A \wedge B$;

б) $A \vee \neg B \wedge C \to A \wedge \neg B$;

в) $A \vee \neg B \wedge C \to B \leftrightarrow \neg C$.

1.3. Докажите, что следующие формулы являются тавтологиями:

а) $(A \wedge B) \to C \leftrightarrow A \to (B \to C)$;

б) $(A \wedge B) \to C \leftrightarrow (A \wedge \neg C) \to \neg B$;

в) $\neg(A \to B) \leftrightarrow A \wedge \neg B$;

г) $(A \to B) \wedge \neg B \to \neg A$;

д) $A \to (\neg A \to B)$;

е) $A \to (B \to A)$;

ж) $(\neg A \to A) \to A$;

з) $(A \to B) \to (A \wedge C \to B \wedge C)$;

и) $(A \to B) \wedge (C \to D) \to (A \wedge C \to B \wedge D)$;

к) $(A \to B) \wedge (C \to D) \to (A \vee C \to B \vee D)$;

л) $\neg(A \leftrightarrow B) \leftrightarrow (\neg(A \to B) \vee \neg(B \to A))$.

2 НЕЙРОННЫЕ СЕТИ

Краткие теоретические сведения

Элементы искусственных нейронных сетей

Искусственные нейронные сети (ИНС) – вид математических моделей, которые строятся по принципу организации и функционирования их биологических аналогов – сетей нервных клеток (нейронов) мозга.

Простой персептрон – это искусственный нейрон Маккалоха-Питтса (рис. 2.1). Пусть $x = (x_0,...,x_n)$ – вектор выходных коэффициентов нейрона. Эти коэффициенты обозначают так же x_i, $i = \overline{0,n}$. На практике обычно полагают $x_0 = 1$. Вектор весовых коэффициентов входов нейрона обозначим $\omega = (\omega_0,...,\omega_n)$ или ω_i, $i = \overline{0,n}$. ω_0 – называют пороговым значением нейрона.

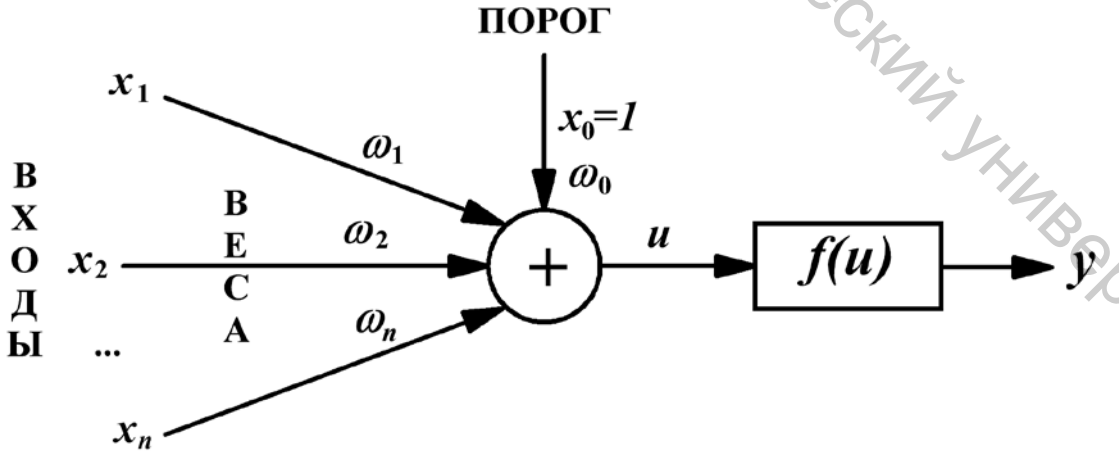


Рисунок 2.1 – Схематическое изображение персептрона-нейрона Маккалока-Питтса

Выходной сигнал сумматора u – это линейная комбинация всех входных сигналов нейрона и соответствующих им весов:

$$u = \sum_{i=0}^{n} \omega_i x_i \,. \tag{2.1}$$

Если считать коэффициенты векторов одномерными матрицами, то (2.1) можно переписать в матричной форме:

$$u = \omega \cdot x \,,$$

где $x = (x_0, \ldots, x_n)^T$ – транспонированная матрица.

В качестве нелинейной функция активации персептрона выступает функция вида

$$y = f(u) = \begin{cases} 1, u \geq 0 \\ 0, u < 0, \end{cases} \qquad y \in \{0,1\}. \tag{2.2}$$

Она является ступенчатой и преобразует выходной сигнал сумматора в выходной сигнал нейрона y.

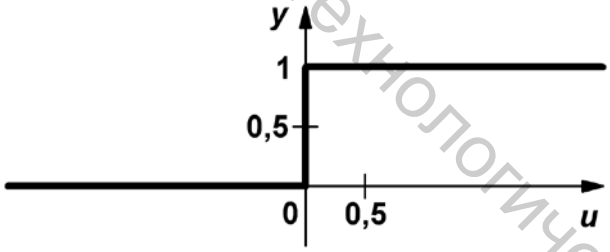


Рисунок 2.2 – График пороговой функции активации

Функция активации нейронов (характеристическая функция) $f(u)$ – нелинейный преобразователь, преобразующий выходной сигнал сумматора, – может быть одной и той же для всех нейронов сети. В этом случае сеть называют однородной (гомогенной). Если же $f(u)$ зависит еще от одного или нескольких параметров, значения которых меняются от нейрона к нейрону, то сеть называют неоднородной (гетерогенной).

Также на практике часто используют и другие функции:

– знаковая или сигнатурная (рис. 2.3):

$$y = f(u) = \begin{cases} 1, & u > 0 \\ -1, u \leq 0 \end{cases}; \tag{2.3}$$

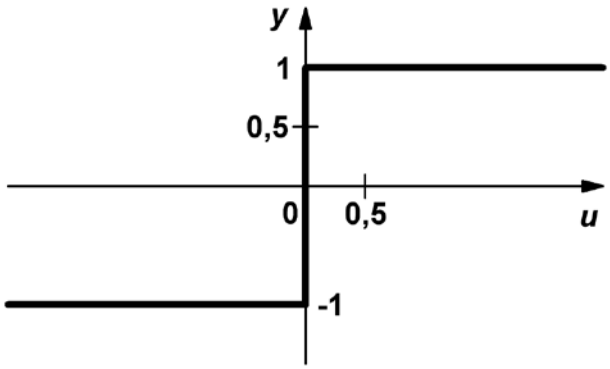


Рисунок 2.3 – График знаковой функции активации

– сигмоидная или сигмоидальная (рис. 2.4):

$$y = f(u) = \frac{1}{1+e^{-u}}; \tag{2.4}$$

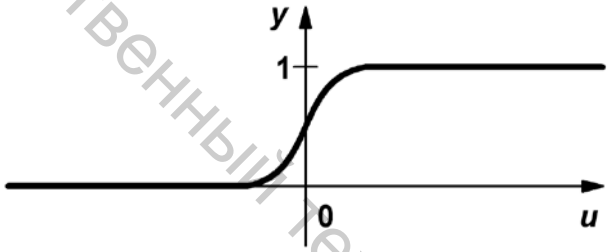


Рисунок 2.4 – График сигмоидальной функции активации

– гиперболический тангенс (рис. 2.5):

$$y = f(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}; \tag{2.5}$$

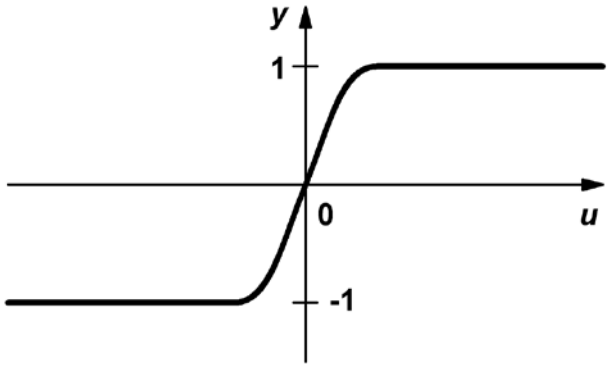


Рисунок 2.5 – График функции активации гиперболического тангенса

– радиальная базисная или гауссова (рис. 2.6):

$$y = f(u) = e^{-u^2}. \qquad (2.6)$$

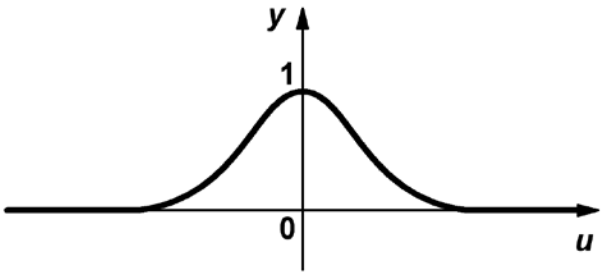


Рисунок 2.6 – График радиальной базисной или гауссовой функции активации

Задача линейного разделения элементов множества на два класса

С персептроном связана классическая задача линейного (однозначного) разделения элементов множества на два класса. Задача линейного разделения элементов множества на два класса состоит в построении линейного решающего правила, то есть нахождении такого вектора весов:

$$\omega = (\omega_0, ..., \omega_n),$$

где ω_0 – пороговое значение, что при значении нелинейной функции активации персептрона

$$y = f(u) > 0 \quad (y = 1), \quad y = \omega x,$$

вектор *x* относится к первому классу, а при

$$y = f(u) \le 0 \quad (y = -1 \vee y = 0)$$

ко второму.

Метод разделения центров масс – простейший способ построения решающего правила. В этом методе начальный вектор весов персептрона вычисляется по формуле:

$$\omega = \frac{\sum_{i=1}^{k} x_i - \sum_{j=1}^{l} y_j}{k + l},$$

где векторы x_i, $i = \overline{1, k}$ относятся к первому классу, а векторы y_j, $j = \overline{1, l}$ – ко второму классу.

Линейные решающие правила, построенные на основании разделения центров масс, могут ошибаться на примерах из обучающей выборки даже в тех случаях, когда существует их точное линейное разделение. Однако метод

центров масс часто используют для определения начального значения весового вектора ω для алгоритмов обучения персептрона.

Геометрическая интерпретация. Пусть задан стандартный персептрон Маккалоха-Питтса. Линейное решающее правило делит входное векторное пространство на две части гиперплоскостью, классифицируя входные векторы как относящиеся к первому классу, если выходной сигнал $y > 0$, или ко второму классу, если выходной сигнал $y \le 0$.

Гиперплоскость – подпространство с размерностью, на единицу меньшей, чем объемлющее пространство. Например, для двумерного пространства гиперплоскость есть прямая, для трёхмерного – плоскость, для четырёхмерного – трёхмерное пространство («трёхмерная плоскость») и т. д.

Уравнение разделяющей гиперплоскости задаётся в виде скалярного произведения

$$(\omega, x) = 0$$

или, расписав скалярное произведение векторов, – через их координаты:

$$\sum_{i=0}^{n} \omega_i x_i = 0, \; x_0 = 1. \tag{2.7}$$

В n-мерном векторном пространстве (пространстве входных сигналов персептрона) вектор нормали

$$\omega' = (\omega_1, \ldots, \omega_n),$$

перпендикулярен разделяющей гиперплоскости. Длину проекции вектора x на вектор нормали вычислим как:

$$x_{\omega'} = \frac{(\omega', x)}{\|\omega'\|}$$

или через координаты:

$$x_{\omega'} \frac{\sum_{i=0}^{n} \omega_i' x_i}{\sqrt{\sum_{i=1}^{n} \omega_i'^2}}. \tag{2.8}$$

Персептрон при заданном векторе входных сигналов

$$x' = (x_1, ..., x_n),$$

для которого будем опускать при его записи $x_0 = 1$ (если не оговорено иное), дает выход:

$$y = \begin{cases} 1, x_{\omega'} > -\dfrac{\omega_0}{\|\omega'\|}, \\ 0, x_{\omega'} \le -\dfrac{\omega_0}{\|\omega'\|}. \end{cases} \qquad (2.9)$$

Поясним первую строчку выражения (2.9). Для *y=1*, *u>0*. Мы можем записать:

$$u = \sum_{i=0}^{n} \omega_i x_i > 0,$$
$$\omega_1 x_1 + \omega_2 x_2 + ... + \omega_0 > 0.$$

Разделим правую и левую часть на $\|\omega'\|$:

$$\frac{\omega_1 x_1 + \omega_2 x_2 + ...}{\|\omega'\|} + \frac{\omega_0}{\|\omega'\|} > 0.$$

Получаем

$$x_{\omega'} > -\frac{\omega_0}{\|\omega'\|}.$$

Пример: в двухмерном пространстве входных сигналов уравнение гиперплоскости задаётся уравнением прямой:

$$\omega_0 + \omega_1 x_1 + \omega_2 x_2 = 0.$$

Пусть $\omega_1 = \omega_2 = 1, \quad \omega_0 = -2$. Тогда получаем уравнение прямой:

$$-2 + x_1 + x_2 = 0,$$

которая представлена на рисунке 2.7 линией, пересекающей оси координат в точках: (2;0), (0;2). Причем $\omega' = (1;1)$ – нормаль к прямой.

Пусть заданы два радиус-вектора: ***a*** = (2;2), ***b*** = (-3;2).

Требуется отнести их к одному из классов, в зависимости от положения точек относительно разделяющей прямой.

Длины проекций векторов ***a*** и ***b*** на нормаль вычислим по формуле (2.8):

$$x_{\omega'}\frac{\sum_{i=0}^{n}\omega_i' x_i}{\sqrt{\sum_{i=1}^{n}\omega_i'^2}},$$

$$a_{\omega'}=\frac{2\cdot 1+2\cdot 1}{\sqrt{1^2+1^2}}=\frac{4}{\sqrt{2}},\ b_{\omega'}=-\frac{1}{\sqrt{2}}.$$

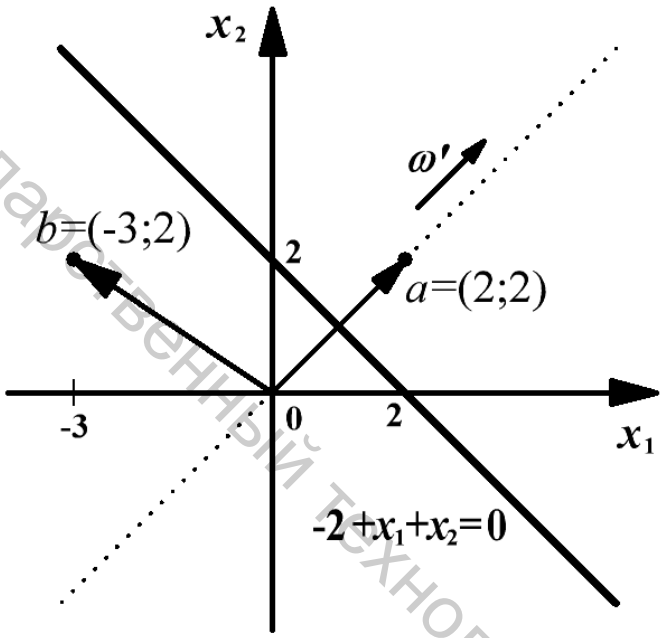


Рисунок 2.7 – Задача разделения плоскости на два класса

Вектор ***a*** = (2;2) относится к первому классу, так как на выходе персептрона по формуле (2.9) получаем значение $y = 1$, поскольку $a_{\omega'} > \frac{2}{\sqrt{2}}$.

Вектор ***b*** = (-3;2) относится ко второму классу, так как на выходе персептрона получаем значение $y = 0$, поскольку $b_{\omega'} < \frac{2}{\sqrt{2}}$.

Реализация булевых функций AND и OR при помощи персептрона

Легко убедиться, что однослойный персептрон способен реализовать булевы функции AND и OR.

Далее приводятся данные для реализации функции AND (рис. 2.8, 2.9).

гиперплоскость: $x_1 + x_2 = 1{,}5$	Результаты вычислений персептрона		
	x_1	x_2	x_1 *AND* x_2
весовой вектор: $\omega = (-1{,}5;1;1)$	0	0	0
	0	1	0
Вектор нормали: $\omega' = (1;1)$	1	0	0
	1	1	1

В данном примере уравнение гиперплоскости $x_1 + x_2 = 1{,}5$. Это означает, что $\omega_1 = 1$ и $\omega_2 = 1$. Рассмотрим, какие значения может принимать ω_0. Для данной гиперплоскости $\omega_0 = -1{,}5$.

Для таблицы истинности булевой функций AND проведем вычисления построчно.

Если x_1 *AND* x_2 принимает значение 0 $(x_1 = 0, x_2 = 0, y = 0)$, это соответствует $\omega_0 + x_1\omega_1 + x_2\omega_2 < 0$. Или: $\omega_0 + 0 \cdot 1 + 0 \cdot 1 < 0$. Следовательно, $\omega_0 < 0$.

При $x_1 = 0, x_2 = 1, y = 0$, получим $\omega_0 + 0 \cdot 1 + 1 \cdot 1 < 0$. Следовательно, $\omega_0 < -1$.

При $x_1 = 1, x_2 = 0, y = 0$, получим $\omega_0 + 1 \cdot 1 + 0 \cdot 1 < 0$. Следовательно, $\omega_0 < -1$.

При $x_1 = 1, x_2 = 1, y = 1$, получим $\omega_0 + 1 \cdot 1 + 1 \cdot 1 \geq 0$. Следовательно, $\omega_0 \geq -2$.

Мы получили простую систему:

$$\begin{cases} \omega_0 < 0 \\ \omega_0 < -1 \\ \omega_0 < -1 \\ \omega_0 \geq -2 \end{cases}.$$

Из решения системы видно, что при $\omega_1 = 1, \omega_2 = 1$, ω_0 может принимать значения из промежутка $\omega_0 \in [-2;-1)$. Например: $\omega_0 = -1{,}5$.

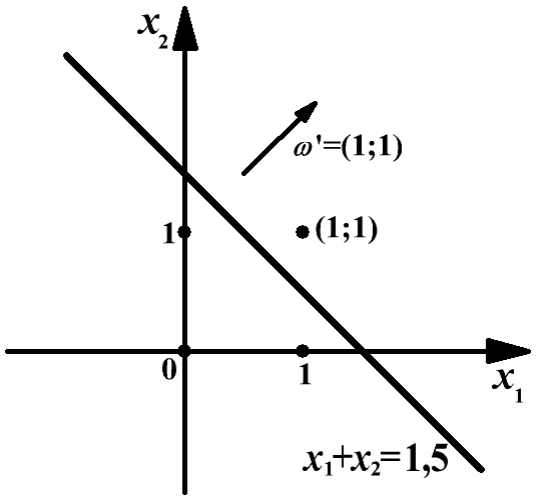


Рисунок 2.8 – Гиперплоскость $x_1 + x_2 = 1{,}5$, реализующая булеву функцию AND

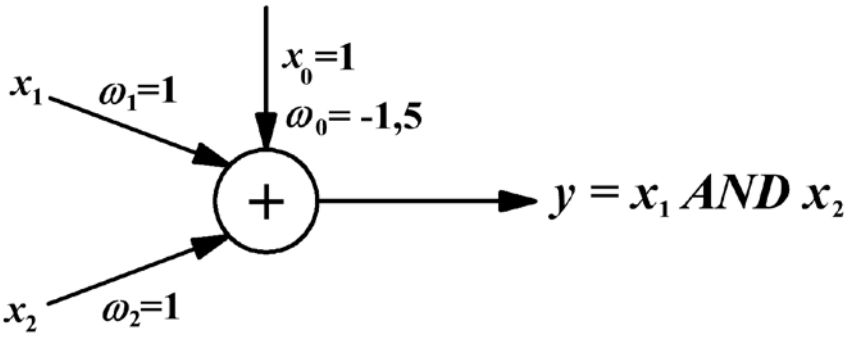


Рисунок 2.9 – Однослойный персептрон, реализующий булеву функцию AND

Аналогично реализуется функция OR (рис. 2.10, 2.11).

гиперплоскость: $x_1 + x_2 = 0{,}5$	Результаты вычислений персептрона		
весовой вектор: $\omega = (-0{,}5;1;1)$	x_1	x_2	x_1 *OR* x_2
	0	0	0
	0	1	1
вектор нормали: $\omega' = (1;1)$	1	0	1
	1	1	1

Уравнение гиперплоскости $x_1 + x_2 = 0{,}5$. Это означает, что $\omega_1 = 1$ и $\omega_2 = 1$. Рассмотрим, какие значения может принимать ω_0. Для данной гиперплоскости $\omega_0 = -0{,}5$.

Для таблицы истинности булевой функций OR проведем вычисления построчно.

Если x_1 *OR* x_2 принимает значение 0 $(x_1 = 0, x_2 = 0, y = 0)$, это соответствует $\omega_0 + x_1\omega_1 + x_2\omega_2 < 0$. Или $\omega_0 + 0 \cdot 1 + 0 \cdot 1 < 0$. Следовательно, $\omega_0 < 0$.

При $x_1 = 0, x_2 = 1, y = 1,$ получим $\omega_0 + 0 \cdot 1 + 1 \cdot 1 \geq 0$. Следовательно, $\omega_0 \geq -1$.

При $x_1 = 1, x_2 = 0, y = 1,$ получим $\omega_0 + 1 \cdot 1 + 0 \cdot 1 \geq 0$. Следовательно, $\omega_0 \geq -1$.

При $x_1 = 1, x_2 = 1, y = 1,$ получим $\omega_0 + 1 \cdot 1 + 1 \cdot 1 \geq 0$. Следовательно, $\omega_0 \geq -2$.

Мы получили систему

$$\begin{cases} \omega_0 < 0 \\ \omega_0 \geq -1 \\ \omega_0 \geq -1 \\ \omega_0 \geq -2 \end{cases}.$$

Очевидно, что ω_0 может принимать значения из промежутка $\omega_0 \in [-1;0)$. Например: $\omega_0 = -0{,}5$.

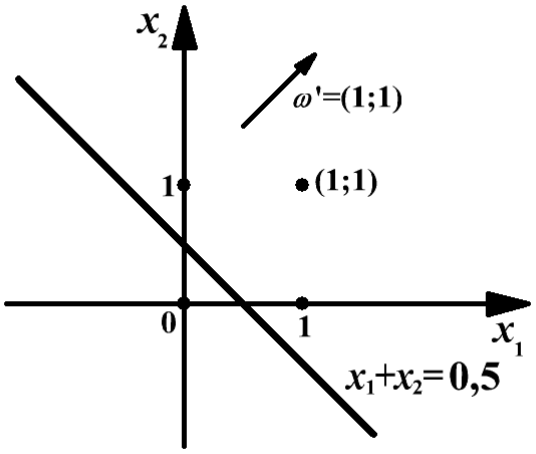


Рисунок 2.10 – Гиперплоскость $x_1 + x_2 = 0{,}5$, реализующая булеву функцию OR

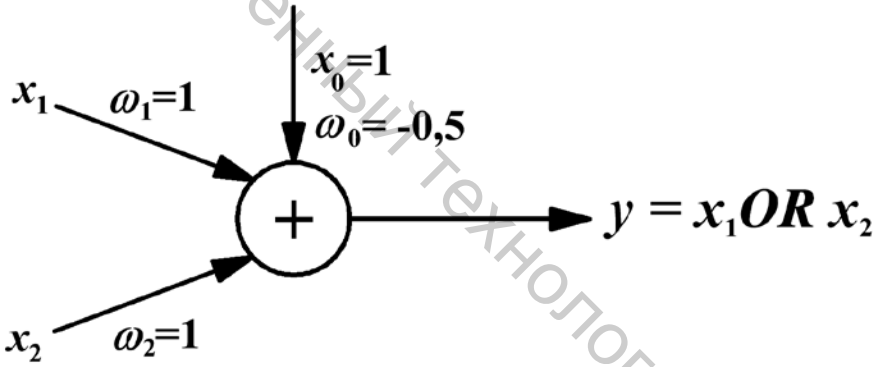


Рисунок 2.11 – Однослойный персептрон, реализующий булеву функцию OR

Реализация булевой функции XOR при помощи сети персептронов

Функцию XOR возможно реализовать при помощи двухслойной нейронной сети. Первый слой сети состоит из двух персептронов, реализующих функции OR и NOT(AND). Во втором слое находится персептрон, реализующий функцию AND от двух выходных сигналов первого слоя. Составим таблицу истинности.

x_1	x_2	$x_1 AND\, x_2$	$\overline{x_1 AND\, x_2}$	$x_1 OR\, x_2$	$\overline{x_1 AND\, x_2} AND (x_1 OR\, x_2)$
0	0	0	1	0	0
1	0	0	1	1	1
0	1	0	1	1	1
1	1	1	0	1	0

Данные для реализации функции XOR:

гиперплоскость 1: $x_1 + x_2 = 0{,}5$ гиперплоскость 2: $-x_1 - x_2 = -1{,}5$ гиперплоскость 3: $x_1 + x_2 = 1{,}5$	Результат вычисления сети
весовой вектор 1: $\omega^1 = (-0{,}5;1;1)$ весовой вектор 2: $\omega^2 = (1{,}5;-1;-1)$ весовой вектор 3: $\omega^3 = (-1{,}5;1;1)$	
вектор нормали 1: $\omega'^1 = (1;1)$ вектор нормали 2: $\omega'^2 = (-1;-1)$ вектор нормали 3: $\omega'^3 = (1;1)$	

x_1	x_2	$x_1 XOR\, x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Гиперплоскость 1 описывает функцию OR (см. предыдущий пример). Гиперплоскость 2 описывает функцию $\overline{x_1 AND\, x_2}$. Действительно, пусть $\omega_1 = -1,\ \omega_2 = -1$. Тогда:

x_1	x_2	$\overline{x_1 AND\, x_2}$	$\omega_0 + x_1\omega_1 + x_2\omega_2 = \omega_0 - x_1 - x_2$
0	0	1	$\omega_0 - 0 - 0 \ge 0,\ \omega_0 \ge 0$
1	0	1	$\omega_0 - 1 - 0 \ge 0,\ \omega_0 \ge 1$
0	1	1	$\omega_0 - 0 - 1 \ge 0,\ \omega_0 \ge 1$
1	1	0	$\omega_0 - 1 - 1 < 0,\ \omega_0 < 2$

Таким образом $\omega_0 \in [1;2)$. В частном случае $\omega_0 = 1{,}5$. Мы получаем уравнение гиперплоскости 2: $1{,}5 - x_1 - x_2 = 0$. Или $-x_1 - x_2 = -1{,}5$.

Запишем уравнение гиперплоскости 3. Составим вспомогательную таблицу:

$\overline{x_1 AND\, x_2}$ (как x_1)	$x_1 OR\, x_2$ (как x_2)	$\overline{x_1 AND\, x_2} AND\, (x_1 OR\, x_2)$	$\omega_1 = 1,\ \omega_2 = 1$ $\omega_0 + x_1 + x_2$
1	0	0	$\omega_0 + 1 + 0 < 0, \omega_0 < -1$
1	1	1	$\omega_0 + 1 + 1 \ge 0, \omega_0 \ge -2$
1	1	1	$\omega_0 + 1 + 1 \ge 0, \omega_0 \ge -2$
0	1	0	$\omega_0 + 0 + 1 < 0, \omega_0 < -1$

Таким образом $\omega_0 \in [-2;-1)$. В частном случае $\omega_0 = -1{,}5$. Получили третью гиперплоскость $x_1 + x_2 = 1{,}5$.

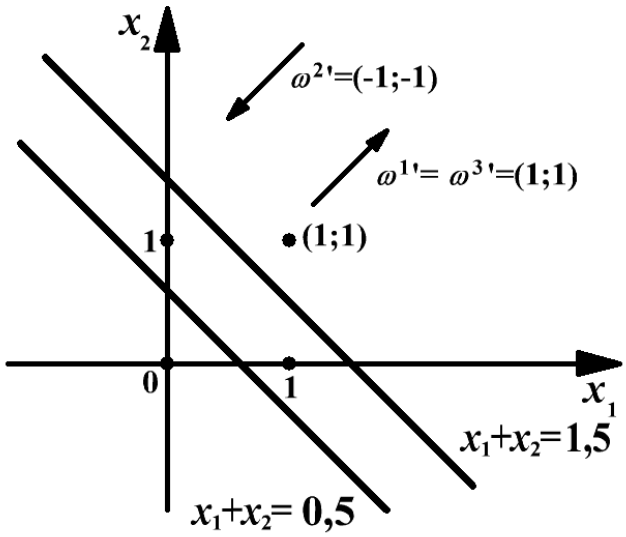


Рисунок 2.12 – Гиперплоскости, реализующие функцию XOR

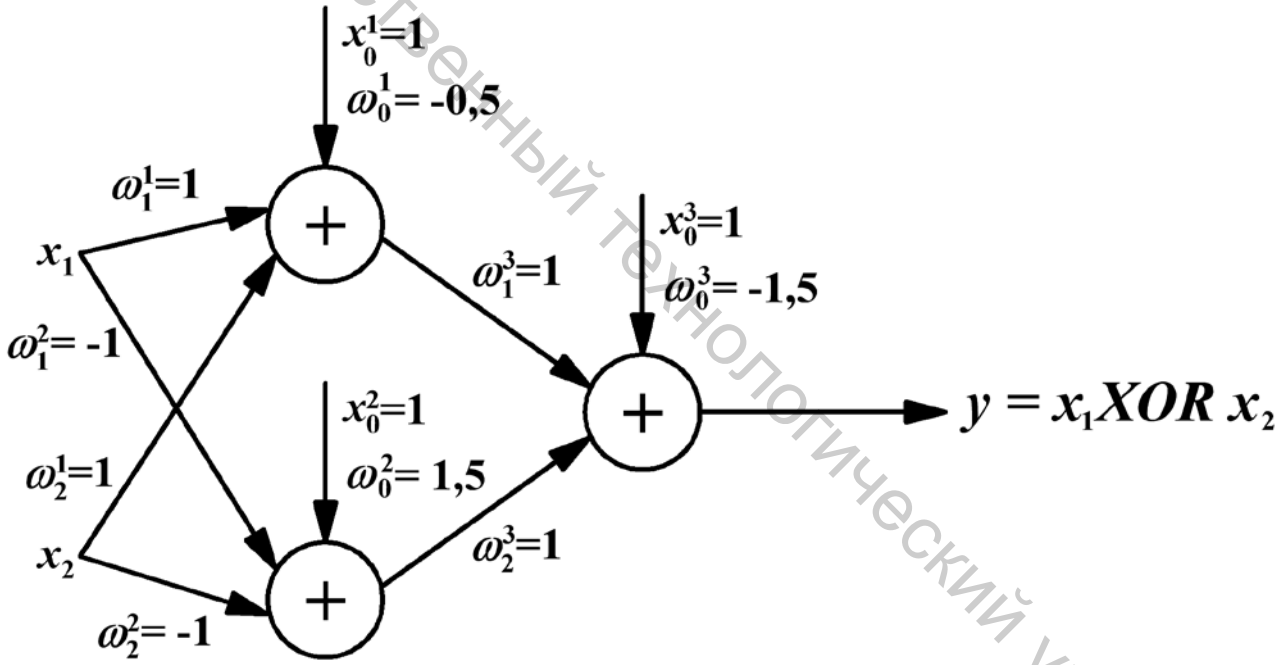
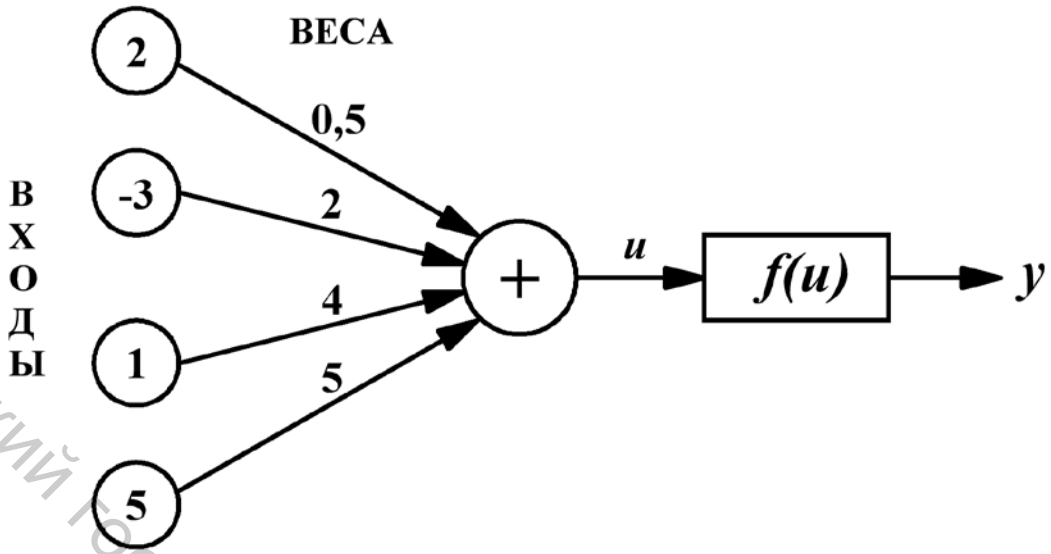


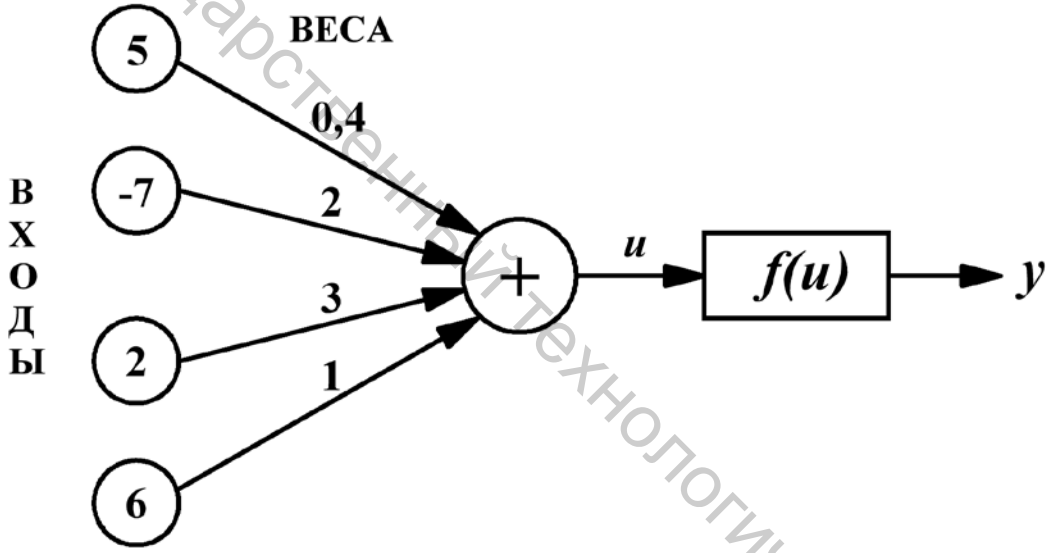
Рисунок 2.13 – Двухслойная сеть, реализующая функцию XOR

Задания для контролируемой самостоятельной работы

2.1. Вычислите взвешенную сумму нейронов, модели которых представлены на рисунке 2.14.



а



б

Рисунок 2.14 – Модели нейронов для нахождения взвешенной суммы

2.2. Составьте таблицу истинности логической функции NOT(AND). Изобразите ее на плоскости и проведите прямую, по одну сторону от которой будут находиться точки, для которых значения этой функции равно 1, а по другую – 0.

2.3. Составьте таблицу истинности логической функции NOT(OR). Изобразите ее на плоскости и проведите прямую, по одну сторону от которой будут находиться точки, для которых значения этой функции равно 1, а по другую – 0.

2.4. В двухмерном пространстве входных сигналов персептрона реализовать булевую функцию AND для $\omega_0 = -1{,}3$:

а) найти ω_1, ω_2;

б) записать уравнение гиперплоскости, найти вектор нормали;

в) выполнить геометрическую интерпретацию.

2.5. В двухмерном пространстве входных сигналов персептрона реализовать булевую функцию OR для $\omega_0 = -1{,}7$:

а) найти ω_1, ω_2;

б) записать уравнение гиперплоскости, найти вектор нормали;

в) выполнить геометрическую интерпретацию.

2.6. Для логической функции AND подобрать два различных набора параметров нейрона, обеспечивающих ее моделирование. Выполнить геометрическую интерпретацию.

2.7. Для логической функции OR подобрать два различных набора параметров нейрона, обеспечивающих ее моделирование. Выполнить геометрическую интерпретацию.

2.8. Чему должно быть равно t в уравнении $\alpha x_1 + \beta x_2 + \gamma x_3 = t$, чтобы соответствующий нейтрон реализовал функцию OR. Значения α, β, γ взять из таблицы 2.1.

Таблица 2.1 – Варианты задания

Номер варианта	α	β	γ	Номер варианта	α	β	γ
1	0,2	1	0,7	6	0,8	0,7	0,9
2	0,1	1	0,9	7	0,9	1	0,9
3	0,9	0,7	1	8	1	0,9	0,7
4	0,5	0,9	1	9	1	0,9	0,7
5	0,9	1	0,7	10	0,4	0,9	0,8

3 МНОГОСЛОЙНЫЙ ПЕРСЕПТРОН И ОБРАТНОЕ РАСПРОСТРАНЕНИЕ ОШИБОК

Краткие теоретические сведения

На рисунке 3.1 изображена трехслойная сеть, состоящая из *входного слоя* (в нем 3 нейрона), *1-го скрытого* слоя (в нем тоже 3 нейрона) и *выходного слоя* (здесь 2 нейрона). Очень часто в литературе по нейросетям первый слой черные круги не считается первым, и вообще некоторые не считают элементы этого слоя нейронами. Средний слой называется скрытым потому, что он не имеет непосредственного «соприкосновения» с внешней средой, в отличие от *входного* и *выходного*.

Каждый элемент некоторого слоя связан с каждым элементом следующего слоя, но ни один элемент не связан с самим собой или с другим нейроном в том же слое, что и он сам. Связи в такой сети являются однонаправленными. По одной связи сигнал может проходить только в одном направлении, и он идет от входного слоя к выходному.

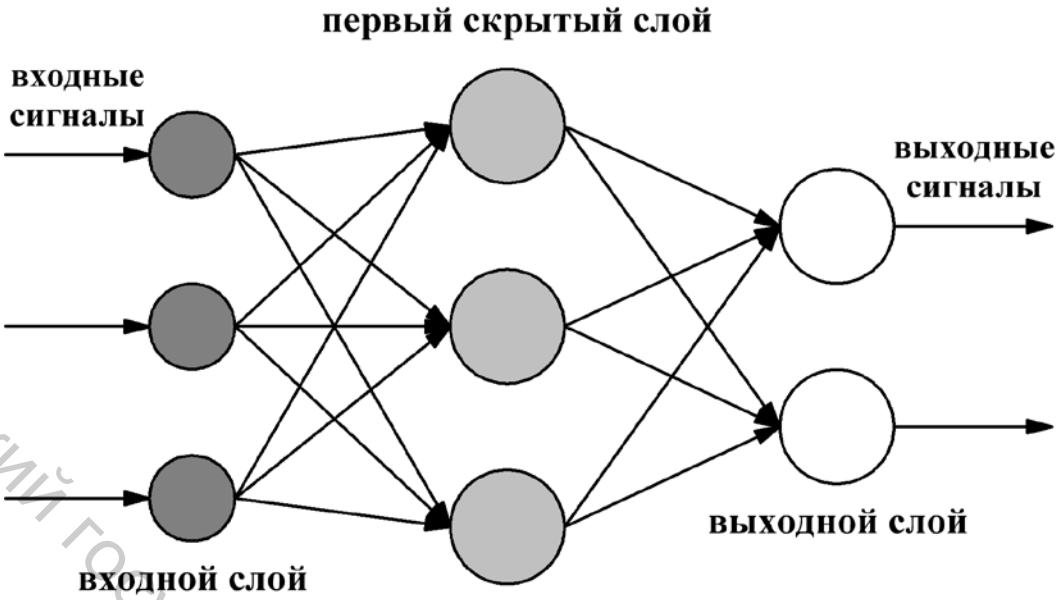


Рисунок 3.1 – Трехслойная сеть

Нейроны входного слоя имеют только один вход, идущий, например, от какого-нибудь датчика. Выход такого нейрона в точности равен входу. Таким образом, он распределяет неизмененный входной сигнал на все нейроны следующего (второго) слоя. Поэтому их иногда не считают нейронами.

Нейроны в остальных слоях одинаковы. Каждый нейрон некоторого слоя получает сигналы ото всех нейронов предыдущего слоя. Работу нейрона можно условно разделить на два этапа:

– комбинирование входа (сигнал нейрона предыдущего слоя умножается на вес связи, ведущий к этому нейрону и произведения складываются);

– вычисление активности. В многослойном персептроне активационная функция обязательно должна быть нелинейной (если она будет линейной, то эта многослойная сеть будет эквивалентна однослойной). Чаще всего используется сигмоидальная функция. Персептроны с сигмоидными активационными функциями с одним выходом назвали адалайн, с несколькими выходами – мадалайн.

Результат вычисляется «послойно». То есть сначала вычисляются комбинированные входы и активности второго слоя. Затем для третьего и так до выходного слоя.

Б. Уидроу и М. Е. Хофф предложили минимизировать квадратичную ошибку формулой:

$$\varepsilon = \frac{1}{2}\sum_{i=1}(d_i - y_i)^2 ,$$

где d_i – требуемый (желаемый) выход i-го нейрона, y_i – тот, который получился в результате вычислений персептрона.

Рассмотрим алгоритм коррекции весовых коэффициентов персептрона, имеющего *J* выходов и *I* входов.

Квадратичная ошибка обучения персептрона ε зависит от весовых коэффициентов ω_{ij}: $\varepsilon = \varepsilon(\omega_{ij})$. Функция-ошибка персептрона зависит от большого количества аргументов ω_{ij}, поэтому для ее представления требуется многомерная система координат, в которой функция $\varepsilon = \varepsilon(\omega_{ij})$ изображается в виде многомерной поверхности, называемой ***гиперповерхностью.***

Например, пусть все аргументы ω_{ij} имеют постоянные значения за исключением двух, например ω_{11} и ω_{12}, которые являются переменными. Тогда в трехмерной системе координат ω_{11}, ω_{12}, ε – гиперповерхность будет иметь вид фигуры, напоминающий параболоид.

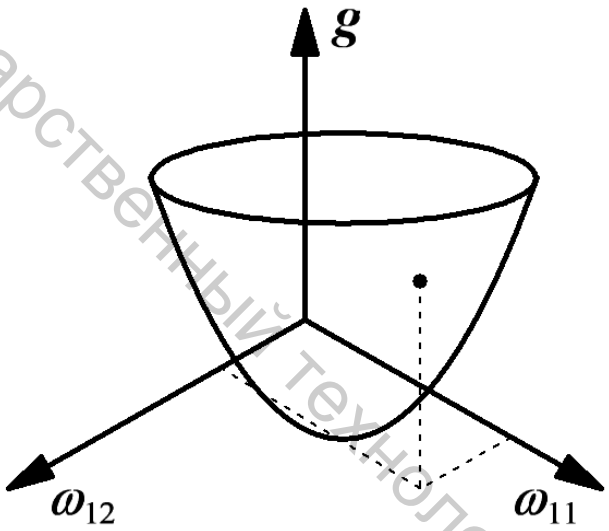


Рисунок 3.2 – Гиперповерхность в трехмерном пространстве

Процесс обучения персептрона можно представить как отыскание такого сочетания весовых коэффициентов ω_{ij}, которому соответствует самая нижняя точка псевдопараболоида. Мы имеем дело с задачей на оптимизацию.

Рассмотрим метод градиентного спуска, согласно которому изменение каждого весового коэффициента $\Delta\omega_{ij}$ производится в сторону, противоположную градиенту функции ε. Градиент функции $\varepsilon = \varepsilon(\omega_{ij})$ представляет собой вектор, проекциями которого на оси координат являются производные от функции ε по этим координатам $\frac{\partial \varepsilon}{\partial \omega_{ij}}$ и градиент функции всегда направлен в сторону ее наибольшего возрастания. Наша задача состоит в отыскании минимума функции, т. е. мы двигаемся в сторону, противоположную градиенту. Отсюда и название – метод градиентного спуска.

Поэтому на каждой итерации к координатам текущей точки ω_{ij} мы будем добавлять величину, прямо пропорциональную производной по координате, взятую с противоположным знаком:

$$\Delta\omega_{ij} = -\alpha \frac{\partial \varepsilon}{\partial \omega_{ij}}, \tag{3.1}$$

где α – коэффициент скорости обучения, обычно задаваемый в пределах от 0,05 до 1.

Квадратичная ошибка ε является сложной функцией, зависящей от выходных сигналов персептрона y_i, которые в свою очередь зависят от ω_{ij}, т. е. $\varepsilon = \varepsilon(y_i(\omega_{ij}))$. По правилу дифференцирования сложной функции:

$$\frac{\partial \varepsilon}{\partial \omega_{ij}} = \frac{\partial \varepsilon}{\partial y_i} \frac{\partial y_i}{\partial \omega_{ij}}. \tag{3.2}$$

Выходные сигналы нейронов y_i вычисляются с помощью сигмоидальной активационной функцией $y_i = f_\sigma(S_i), \quad S_i = \sum_{j=1}^{J} \omega_{ij} x_j$. Следовательно

$$\frac{\partial y_i}{\partial \omega_{ij}} = \frac{\partial f_\sigma(S_i)}{\partial S_i} \frac{\partial S_i}{\partial \omega_{ij}} = f'_\sigma(S_i) x_j. \tag{3.3}$$

Кроме того,

$$\varepsilon = \frac{1}{2} \sum_{i=1} (d_i - y_i)^2,$$

поэтому

$$\frac{\partial \varepsilon}{\partial y_i} = -(d_i - y_i). \tag{3.4}$$

Подставив (3.3) и (3.4) в (3.2) и затем полученное выражение в (3.1), получим:

$$\Delta\omega_{ij} = -\alpha\frac{\partial\varepsilon}{\partial\omega_{ij}} = -\alpha\frac{\partial\varepsilon}{\partial y_i}\frac{\partial y_i}{\partial\omega_{ij}} = -\alpha\big(-(d_i - y_i)\big)f'_\sigma(S_i)x_j,$$

$$\Delta\omega_{ij} = \alpha(d_i - y_i)f'_\sigma(S_i)x_j.$$

Учитываем, что

$$f'_\sigma(S) = \left((1+e^{-S})^{-1}\right)' = -(1+e^{-S})^{-2}(-e^{-S}) =$$

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

получим

$$\Delta\omega_{ij} = \alpha(d_i - y_i)f_\sigma(S_i)\big(1-f(S_i)\big)x_j,$$

$$\Delta\omega_{ij} = \alpha(d_i - y_i)y_i\big(1-y_i\big)x_j. \qquad (3.5)$$

Мы получили итерационную формулу для обучения персептрона:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \Delta\omega_{ij}, \qquad (3.6)$$

где

$$\Delta\omega_{ij} = \alpha\delta_i x_j, \qquad (3.7)$$

$$\delta_i = (d_i - y_i)y_i\big(1-y_i\big). \qquad (3.8)$$

δ_i называют нейронной ошибкой. Алгоритм (3.6)–(3.8) называют обобщенным дельта-правилом. Введение сигмоидальной функции расширяет область применения персептрона. Теперь он может оперировать не только с бинарными, но и с непрерывными выходными сигналами.

Алгоритм обратного распространения ошибки (Back propagation algorithm)

Алгоритм обратного распространения ошибки относится к методам обучения с учителем, поэтому требует, чтобы в обучающих примерах были заданы целевые значения.

Рассмотрим пошаговый пример работы сети.

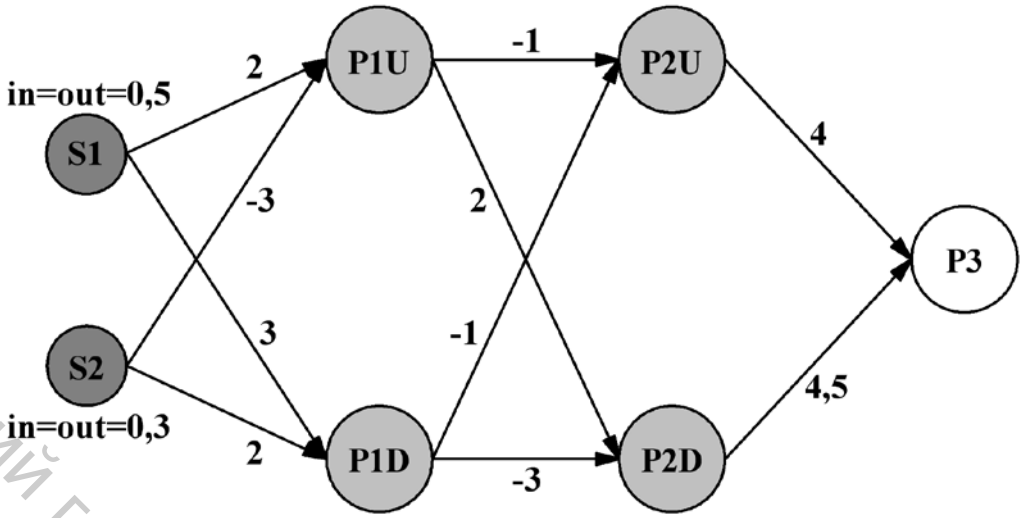


Рисунок 3.3 – Пример архитектуры персептрона для расчета алгоритма обратного распространения ошибки

Предположим, что на вход сети подаются числа 0,5 и 0,3. Цифрами над стрелками обозначены веса связей. Веса связей взяты случайными. Как правило, начальные значения весов берутся случайными в диапазоне от -0,5 до 0,5. Но в данном примере взяты простые числа, чтобы легче было считать. Допустим, что нам требуется получить на выходе значение 1. В качестве функции активации нейронов возьмем функцию вида:

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

Наша задача – изменить веса связей таким образом, чтобы выход сети стал ближе к требуемому. То есть научить сеть выдавать определенный результат при предъявлении ей некоторых данных.

Если нам известна фактическая ошибка нейрона, то мы можем вычислить, насколько надо изменить вес связи для «приближения» результата к требуемому. Сначала определим ошибки выходных нейронов. В принципе ошибка выходного нейрона равна разности требуемого результата и реально полученного. Но поскольку нам необходимо по этой ошибке для каждой связи, ведущей к этому выходному нейрону, вычислить новое значение, то нам необходимо найти зависимость ошибки от комбинированного входного сигнала. Будем обозначать: IN – комбинированный вход, OUT – активность нейрона. Приведем пошаговый пример работы сети.

Во-первых, вычислим результат работы сети при заданных весовых множителях. Мы «прогоняем» входной сигнал через сеть от входного слоя к выходному.

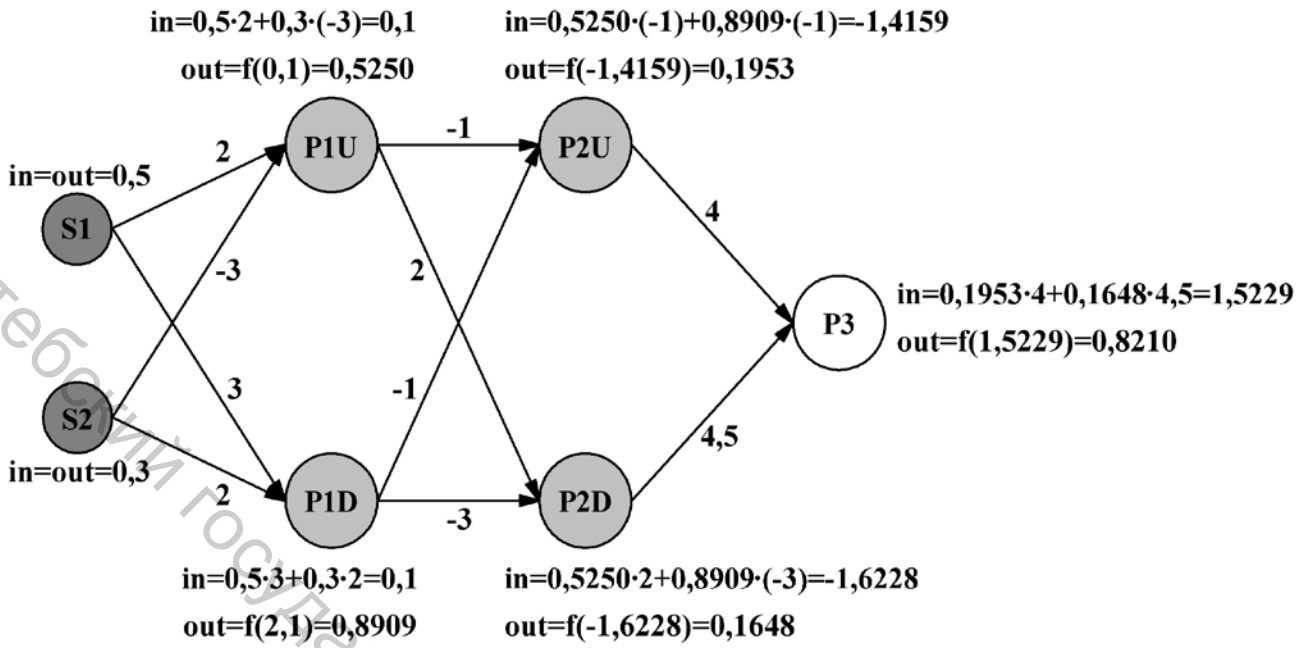


Рисунок 3.4 – Прямое распространение ошибки

В нашем примере сеть вывела значение 0,8210. Нам требовалось получить значение 1. Фактическая ошибка единственного нашего выходного нейрона равна 0,0263 и вычислена по формуле (3.8) следующим образом:

$$(1-0{,}8210)\cdot 0{,}8210\cdot(1-0{,}8210)=0{,}0263\,.$$

Во-вторых, при использовании формул возникает вопрос о вычислении нейронной ошибки $(d_i - y_i)$, которая для скрытого слоя неизвестна. Идея авторов рассматриваемого алгоритма состояла в том, чтобы в качестве этой ошибки использовать суммарные нейронные ошибки с выходного слоя, помноженные на силы соответствующих синаптических связей, т. е.

$$\delta_i = (d_i - y_i)y_i(1-y_i),$$
$$(d_i - y_i) = \sum_{i=1}\delta_i\omega_{ij}\,.$$

Для верхнего нейрона на рисунке комбинированный вход – это ошибка выходного нейрона, умноженная на вес связи, ведущей от верхнего нейрона к выходному.

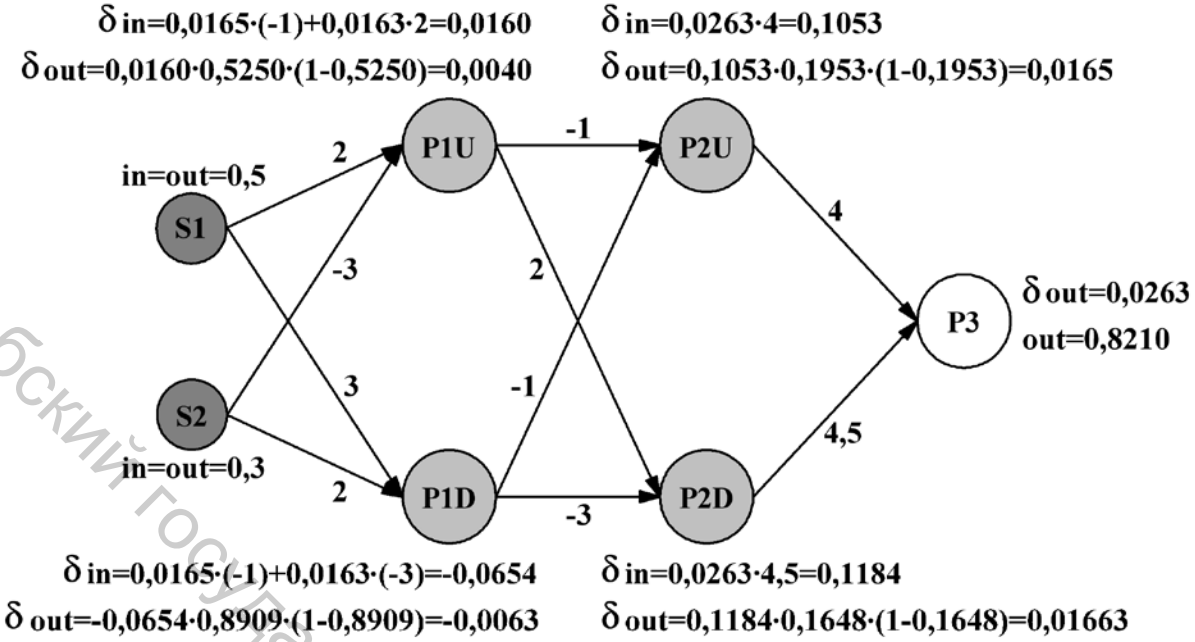


Рисунок 3.5 – Обратное распространение ошибки

Считать ошибки для входных нейронов не имеет никакого смысла, поскольку они никак не изменяют входные сигналы, а сами сигналы – это отправная точка всех вычислений.

В-третьих, необходимо скорректировать веса связей. Рассмотрим простую схему из двух нейронов: нам требуется изменить вес связи между ними; нам известна ошибка одного нейрона, активность другого нейрона и текущий вес связи:

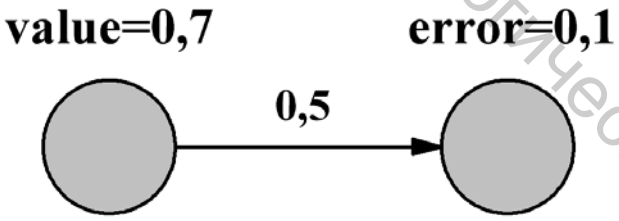


Рисунок 3.6 – Схема двух нейронов для корректировки веса связи

Изменение связи прямо пропорционально значению ошибки нейрона, к которому идет связь, значению активности нейрона, от которого идет связь, и некоторому значению α, называемому *коэффициентом обучения*:

$$\Delta\omega_{ij} = \alpha\delta_i x_j,$$

где δ_i – значение ошибок.

Пусть коэффициент обучения равен 1, тогда изменение веса будет равно

$\Delta\omega_{ij} = 1 \cdot 0{,}1 \cdot 0{,}7 = 0{,}007$. Новые значения весов равны сумме старых значений и поправке к весам $\Delta\omega_{ij}$. Повторяя вычисления для новых значений, мы увидим, что ошибка постепенно уменьшается (то есть выход сети приближается к 1). При этом коэффициент обучения влияет на скорость обучения. Но это не значит, что чем он больше, тем лучше. Этот коэффициент определяет примерную величину изменения веса, а это означает, что за раз вес изменится на некоторое значение, при этом чем больше коэффициент, тем больше это значение. А это означает, что при достижении примерно того значения веса, которое нужно, вес будет постоянно «скакать» около требуемого значения.

Задания для контролируемой самостоятельной работы

Согласно номеру в журнале вычислить результат работы сети, если считать, что функция активации нейронов:

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

Изменить веса связей таким образом, чтобы выход сети стал ближе к требуемому *d*. Расчет провести для трех итераций. Коэффициент обучения выбрать самостоятельно.

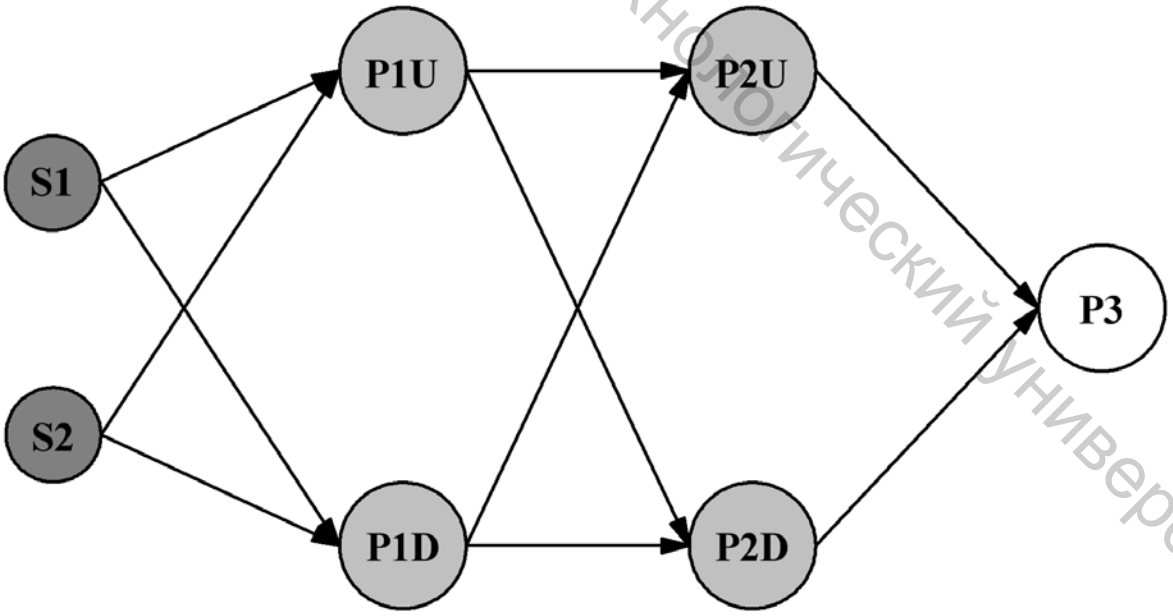


Рисунок 3.7 – Пример двухслойного персептрона для расчета алгоритма обратного распространения ошибки

Таблица 3.1 – Варианты задания

Номер варианта	*S1*	*S2*	ω_{S1P1U}	ω_{S2P1D}	ω_{P1UP2U}	ω_{P1DP2D}	ω_{S1P1D}	ω_{S2P1U}	ω_{P1UP2D}	ω_{P1DP2U}	ω_{P2UP3}	ω_{P2DP3}	*d*
1	0,5	0,9	2	2	-1	2	3	-3	-3	-1	4	-2	1
2	0,4	0,5	3	2	1	-1	2	4	1	5	3	-1	0,9
3	0,5	0,8	2	1	-2	1	3	2	-1	-2	3	-1	0,9
4	0,6	0,3	2	3	-1	2	3	-3	-2	-1	4	-2	1
5	0,5	0,4	2	2	-1	2	3	-2	-3	1	4	-3	1
6	0,6	0,2	3	2	-2	2	3	-2	-2	-1	3	-3	1
7	0,5	0,8	1	1	1	2	3	-2	-2	-2	4	-3	0,9
8	0,4	0,3	2	1	-1	3	2	-3	-2	-1	2	-2	0,9
9	0,6	0,4	3	2	-1	1	2	-3	-3	-2	4	-2	0,9
10	0,7	0,3	3	2	-1	2	3	-3	-3	-1	3	-2	0,9

4 НЕЙРОННАЯ СЕТЬ ХОПФИЛДА

Краткие теоретические сведения

Сеть Хопфилда является однослойной сетью, потому что в ней используется лишь один слой нейронов, каждый из которых связан со всеми остальными. Поэтому такая сеть позволяет реализовать так называемую ассоциативную память. Человеческая память ассоциативна, то есть некоторое воспоминание может порождать большую связанную с ним область. Один предмет напоминает нам о другом, а этот другой – о третьем. Искусственная нейронная сеть с обратной связью так же формирует ассоциативную память. Подобно человеческой памяти по заданной части нужной информации вся информация извлекается из «памяти». Сеть Хопфилда запоминает образы так, что когда мы предъявляем зашумленный образ, то сеть Хопфилда может найти исходный образ, который ей был представлен в процессе обучения. Поскольку нейрон связан со всеми остальными, то весовые множители представляются в виде матрицы.

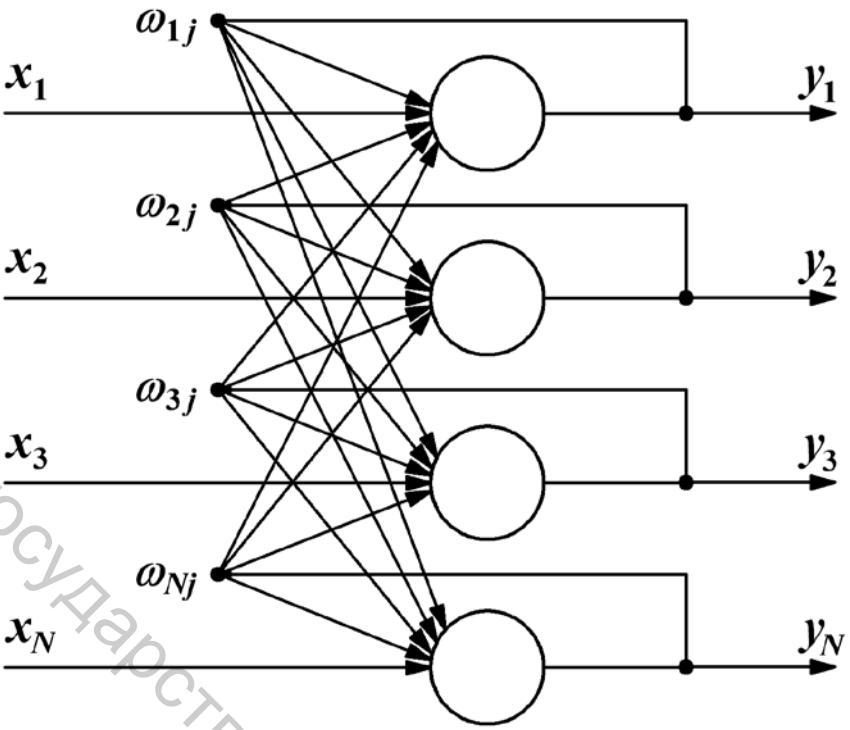


Рисунок 4.1 – Соединение нейронов в нейросети Хопфилда

Важной особенностью сети Хопфилда является то, что она распознает только те образы, которые ей представлены в качестве тестовых данных в процессе обучения. Обычно процесс обучения занимает некоторое время. В сети Хопфилда после представления объектов обучения матрица весов формируется сразу.

Достоинством сети Хопфилда является и то, что она имеет огромное историческое событие. С этой модели началось возрождение к нейронным сетям в середине 1980-х г. Сеть Хопфилда используют так же как ассоциативную память для решения задач оптимизации.

К сожалению, сеть Хопфилда не может решить задачу распознавания, если изображение смещено или перевернуто относительно его исходного запомненного состояния.

Алгоритм обучения сети Хопфилда существенно отличается от алгоритма обратного распространения и других итерационных процедур. Вместо последовательного приближения к нужному состоянию с вычислением ошибок и многократными коррекциями весов, все коэффициенты сети рассчитываются по одной формуле, за один шаг, после чего сеть готова к работе.

Сеть Хопфилда – это искусственная нейронная сеть, содержащая один рекуррентный слой с шаговой (симметрично линейной с ограничениями) функцией активации (наиболее часто используется сигнум-функция (4.1)).

$$sign(u) = f(u) = \begin{cases} 1, & \text{если } u \geq 0 \\ -1, & \text{если } u < 0 \end{cases}. \qquad (4.1)$$

Из-за их биполярной природы нейронные сети Хопфилда иногда называют спинами.

Для наглядности рассмотрим пример распознавания образов. Допустим, фотографию представляем пиксельным способом (рис. 4.2). Изображение можно представить в виде матрицы, в которой есть строки и столбцы.

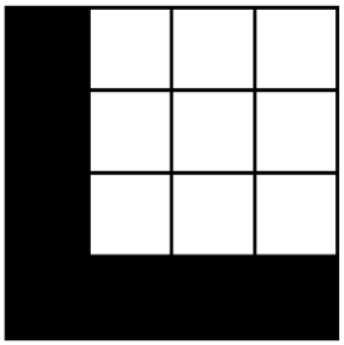


Рисунок 4.2 – Пиксельное представление фотографии буквы L

Если имеем закрашенный квадрат, то ставим 1, если квадрат не закрашен, то -1. Букву L можем представить в виде вектора:

(1; -1; -1; -1; 1; -1; -1; -1; 1; -1; -1; -1; 1; 1; 1; 1).

Сеть Хопфилда предназначена распознать образ, даже если он немного «испорчен». Рассмотрим, как сеть работает.

Пусть есть три «правильных» образа, с помощью которых мы систему должны обучить:

$$x_1 = (-1;1;-1;1)$$
$$x_2 = (1;-1;1;1)$$
$$x_3 = (-1;1;-1;-1)$$

и «испорченный образ»:

$$y = (1;-1;1;-1).$$

Наша задача определить, что это за образ.

Сеть Хопфилда – это некоторый алгоритм с матрицами. Вначале составляется матрица:

$$W = \sum_{k=1}^{3} x_k^T x_k . \qquad (4.2)$$

Для первого образа:

$$\begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}\begin{pmatrix} -1 & 1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix},$$

для второго образа:

$$\begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix}\begin{pmatrix} 1 & -1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{pmatrix},$$

для третьего образа:

$$\begin{pmatrix} -1 \\ 1 \\ -1 \\ -1 \end{pmatrix}\begin{pmatrix} -1 & 1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{pmatrix}.$$

Затем складываем данные трех матриц:

$$\begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 3 & -3 & 3 & 1 \\ -3 & 3 & -3 & -1 \\ 3 & -3 & 3 & 1 \\ 1 & -1 & 1 & 3 \end{pmatrix}.$$

По алгоритму необходимого и достаточного условия сходимости или для устойчивости этой сети надо обнулить диагональ:

$$W = \begin{pmatrix} 0 & -3 & 3 & 1 \\ -3 & 0 & -3 & -1 \\ 3 & -3 & 0 & 1 \\ 1 & -1 & 1 & 0 \end{pmatrix}. \qquad (4.3)$$

На этом обучение заканчивается. То есть матрица W далее не изменяется. Матрица весовых коэффициентов для сети Хопфилда симметрична $W_{ij} = W_{ji}$ и имеет нулевую главную диагональ $W_{ii} = 0$, т. е. отсутствует обратная связь нейрона на себя.

Далее умножаем вектор W на распознаваемый вектор y^T:

$$W \cdot y^T = y^*. \tag{4.4}$$

На полученный вектор действуем функцией активации (4.1):

$$f(y^*) = y'^*. \tag{4.5}$$

Если y'^* есть в множестве известных образов, то это искомый образ. Если нет y'^*, подставляем в (4.4):

$$Wy'^* = y''^*$$

и опять на полученный результат действуем функцией активации. И далее продолжаем итерационный процесс.

В нашем примере

$$W \cdot y^T = \begin{pmatrix} 0 & -3 & 3 & 1 \\ -3 & 0 & -3 & -1 \\ 3 & -3 & 0 & 1 \\ 1 & -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 5 \\ -5 \\ 5 \\ 3 \end{pmatrix}.$$

На полученный вектор действуем функцией активации:

$$f\begin{pmatrix} 5 \\ -5 \\ 5 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix}.$$

Пример простой, мы сразу получили второй образ. Если умножить слева полученный вектор на W, то

$$\begin{pmatrix} 0 & -3 & 3 & 1 \\ -3 & 0 & -3 & -1 \\ 3 & -3 & 0 & 1 \\ 1 & -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 7 \\ -7 \\ 7 \\ 3 \end{pmatrix}.$$

Действуем на нее функцией активации

$$f\begin{pmatrix} 7 \\ -7 \\ 7 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix}.$$

Получили то же самое значение, т. е. система установилась.

Для сети Хопфилда могут существовать две модификации, отличающиеся по времени передачи сигнала: асинхронный и синхронный режимы. Синхронная работа, когда весь вектор, который получили после обработки функции *f,* подставили в итерационный процесс. Асинхронная работа – когда получили вектор и только часть подставили под матрицу *W*, а часть оставили прежней.

Опишем алгоритм работы сети Хопфилда. На первом этапе нужно обучить сеть, предъявляя ей различные эталонные образы. Пусть каждый образ представляет собой *N*-мерный вектор:

$$x = (x_1, x_2, ..., x_N), \tag{4.6}$$

где x_i равняется -1 или 1.

Результатом обучения сети Хопфилда будет матрица, которая отражает веса взаимодействия нейронов. В сети Хопфилда все нейроны взаимодействуют со всеми. В начале обучения все веса принимают нулевые значения. Далее для всех *i* и *j,* которые меняются от 1 до *N*, мы осуществляем следующую процедуру. Для всех образов выполняем корректировку весов:

$$W_{ij} = W_{ij} + x_i x_j.$$

Далее положим

$$W_{ii} = 0$$

и нормируем

$$W_{ij} = \frac{W_{ij}}{N}.$$

Наша сеть обучена.

Процедура распознавания образов осуществляется следующим алгоритмом. Пусть мы ищем образ:

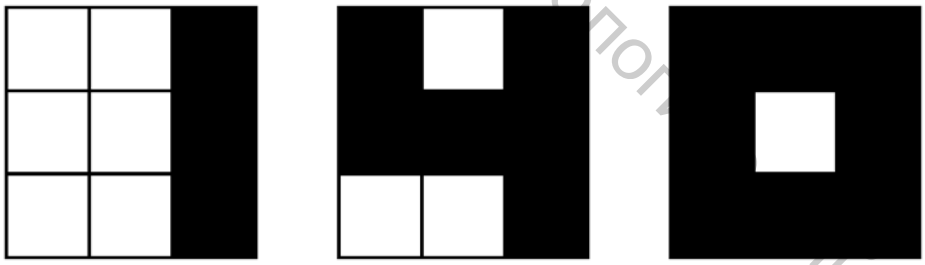
$$y = (y_1; y_2; ...; y_N).$$

1. Цикл по j от 1 до N:
2. Положим $d = 0$.
3. Вложенный цикл по i от 1 до N:
4. $d = d + W_{ij} \cdot y_i$.
5. Конец вложенного цикла.
6. Если $d > 0$, то положить $z_j = 1$, иначе $z_j = -1$.
7. Конец внешнего цикла.
8. Мы получили вектор $Z = [Z_1, Z_2, ..., Z_N]$, если вектор Z содержится во множестве исходных образов, то считаем, что алгоритм нашел образ Z, который соответствует исходному образу Y.
9. Положим $Y=Z$.
10. Переход к шагу 1.

Если наш алгоритм не может найти образ, то значит, что сеть Хопфилда не смогла «вспомнить» такой образ.

Задания для контролируемой самостоятельной работы

Даны три правильных образа:



а) составьте матрицу весовых коэффициентов для сети Хопфилда;

б) «испорченный образ» выберите согласно номеру в журнале и определите, что это за образ. Решение проводить до тех пор, пока система установится.

«Испорченный образ»:

1. $y = (1;1;1;1;1;1;1;1;1)$;
2. $y = (-1;1;1;-1;-1;1;-1;-1;1)$;
3. $y = (-1;-1;1;-1;1;1;-1;-1;1)$;
4. $y = (-1;-1;1;-1;-1;1;-1;1;1)$;
5. $y = (-1;-1;1;-1;-1;1;1;-1;1)$;
6. $y = (1;1;1;1;-1;1;1;1;-1)$;
7. $y = (1;1;1;1;-1;1;1;-1;1)$;
8. $y = (1;1;1;1;-1;1;-1;1;1)$;
9. $y = (1;1;1;-1;-1;1;1;1;1)$;
10. $y = (-1;1;1;1;-1;1;1;1;1)$.

5 ГЕТЕРОАССОЦИАТИВНАЯ ПАМЯТЬ

Краткие теоретические сведения

Нейронная сеть Коско

Одна из характерных особенностей человеческой памяти – возможность ассоциации каких-либо объектов (образов) с другими, непохожими на них, или даже с образами-антагонистами. Например, понятие «зима» ассоциируется с понятием «холод», «праздник» – с «подарком». В случае противоположностей мозгом легко выстраиваются пары: «черный» – «белый», «большой» – «маленький».

На имитацию этой особенности человеческой памяти ориентированы нейронные сети гетероассоциативной памяти – сети Коско, иначе называемые еще сетями двунаправленной ассоциативной памяти.

В структуре этой сети (рис. 5.1) два слоя нейронов. Количество нейронов первого слоя K равно количеству бинарных выходных переменных. Число нейронов второго слоя M равно числу бинарных переменных, которые кодируют входной образ, соответствующий выходному. Сами значения входных и выходных переменных принадлежат бинарному множеству {–1; 1}.

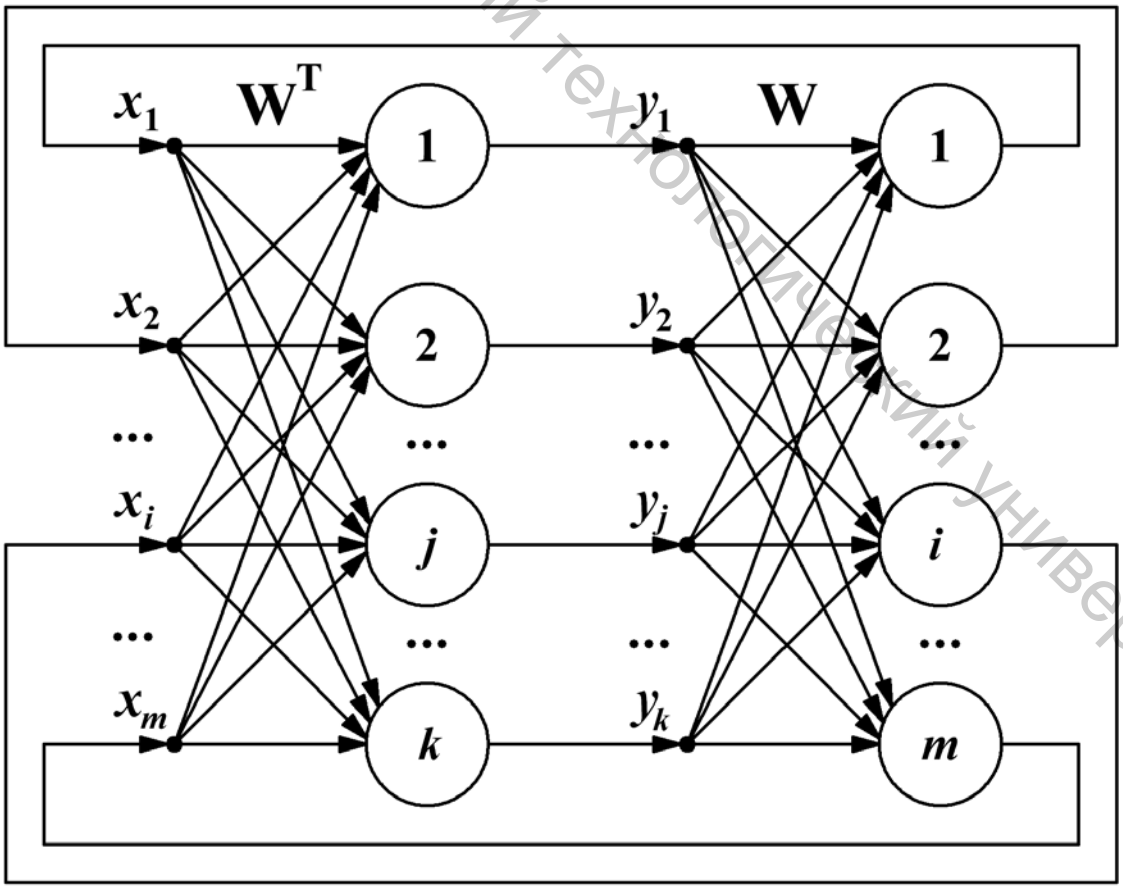
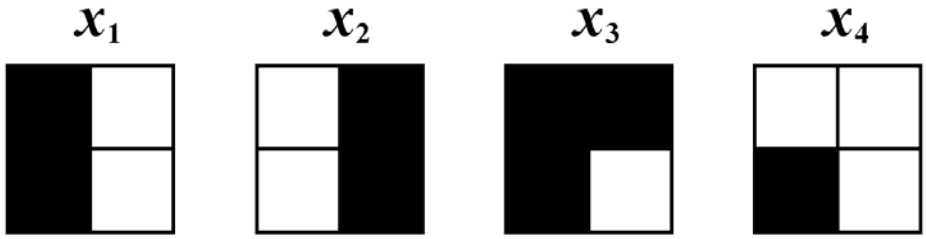


Рисунок 5.1 – Структура двухслойной гетероассоциативной нейронной сети

Алгоритм жизненного цикла нейронной сети Коско включает две стадии: обучения и практического использования.

Рассмотрим следующую задачу. Имеется набор векторов (4 образа):



Если имеем закрашенный квадрат, то ставим 1, если квадрат не закрашен, то -1. В биполярной кодировке мы можем записать:

$$x_1 = (1;-1;1;-1)$$
$$x_2 = (-1;1;-1;1)$$
$$x_3 = (1;1;1;-1)$$
$$x_4 = (-1;-1;1;-1)$$

Это входные образы (матрица выходных образов для настройки нейронной сети Коско).

Запишем индексы в двоичной системе координат. В полярной кодировке:

$$1 \to 0\ 0\ 1$$
$$2 \to 0\ 1\ 0$$
$$3 \to 0\ 1\ 1$$
$$4 \to 1\ 0\ 0$$

В биполярной кодировке индексы примут вид:

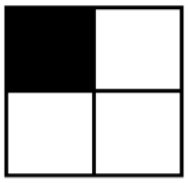
$$1 \to -1\ -1\ 1$$
$$2 \to -1\ 1\ -1$$
$$3 \to -1\ 1\ 1$$
$$4 \to 1\ -1\ -1$$

Это выходные образы. Таким образом, каждой картинке поставлен в соответствие индекс, записанный в биполярной кодировке. Входной вектор содержит 4 элемента, выходной – 3 элемента.

Для тестирования сети Коско на стадии практического использования воспользуемся зашумленным входным графическим образом:

$$x_q = (1; -1; -1; -1).$$

x_q



Вопрос: какой номер мы должны поставить вместо q?

Стадия обучения содержит следующую последовательность действий:

1. Формируются матрицы ассоциированных входных X и выходных образов Y.
2. Рассчитываются элементы ω_{ij} матрицы весовых коэффициентов W:

$$W = X^T Y.$$

Для нашей задачи (в случае четырех образов):

$$\begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} -1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix};$$

$$\begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} -1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{pmatrix};$$

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} -1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 1 & 1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \end{pmatrix};$$

$$\begin{pmatrix} -1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} -1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix}.$$

Матрица весовых коэффициентов нейронной сети Коско запишется как сумма матриц:

$$W = \begin{pmatrix} -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} + \begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{pmatrix} + \begin{pmatrix} -1 & 1 & 1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \end{pmatrix} + \begin{pmatrix} -1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix}.$$

Выполнив сложение матриц, получим:

$$W = \begin{pmatrix} -2 & 0 & 4 \\ -2 & 4 & 0 \\ 0 & -2 & 2 \\ 0 & 2 & -2 \end{pmatrix}.$$

На этом обучение заканчивается, т. е. матрица W далее не изменяется. В качестве функции активации будем брать:

$$sign(u) = f(u) = \begin{cases} 1, & \text{если } u > 0 \\ -1, & \text{если } u \le 0 \end{cases}.$$

На стадии практического использования выполняются следующие операции:

1. Неизвестный, возможно, зашумленный образ подается в виде значений элементов входного вектора-столбца в нейронную сеть. Рассчитываются состояния нейронов первого слоя:

$$s_1^{(q+1)} = W^T x_q .$$

2. К рассчитанным на предыдущем шаге состояниям нейронов применяется пороговая активационная функция, в результате чего получается новый вектор выходных значений $y^{(q+1)}$.

3. Для всех итераций, кроме первой (k ≠ 1), проверяется условие окончания – стабилизация элементов выходного вектора $\left| y^{(k)} - y^{(k-1)} \right| = 0$.

Если оно не выполняется, цикл расчетов продолжается, иначе – переход к п. 6.

4. Полученные значения элементов выходного образа подаются на входы нейронов второго слоя, для которого, в свою очередь, рассчитываются новые состояния нейронов:

$$s_2^{(q+1)} = Wy^{(k+1)}.$$

5. К рассчитанным состояниям нейронов второго слоя также применяется пороговая активационная функция, и вычисляются элементы нового вектора входного образа x_{q+1}.

6. Проверяется соответствие стабилизировавшегося вектора выходных значений одному из идеальных образов матрицы Y. Если соответствие установлено, можно сделать вывод, что нейронной сети удалось найти ассоциацию между зашумленным входом и одним из идеальных выходов. В противном случае можно сказать, что входной вектор был слишком сильно зашумлен и соответствие не может быть установлено.

Для нашего примера вычислим $s_1^{(1)} = W^T x_q$ (итерация 1):

$$s_1^{(1)} = W^T x_q = \begin{pmatrix} -2 & -2 & 0 & 0 \\ 0 & 4 & -2 & 2 \\ 4 & 0 & 2 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ -4 \\ 4 \end{pmatrix}.$$

Применяем пороговую активационную функцию:

$$y_1^{(1)} = f\begin{pmatrix} 0 \\ -4 \\ 4 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}.$$

Переходим к 4 пункту. Полученные значения элементов выходного образа подаются на входы нейронов второго слоя:

$$s_2^{(1)} = Wy_1^{(1)};$$

$$Wy_1^{(1)} = \begin{pmatrix} -2 & 0 & 4 \\ -2 & 4 & 0 \\ 0 & -2 & 2 \\ 0 & 2 & -2 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 6 \\ -2 \\ 4 \\ -4 \end{pmatrix}.$$

Применяем пороговую активационную функцию:

$$x_q^{(1)} = f \begin{pmatrix} 6 \\ -2 \\ 4 \\ -4 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}.$$

Итерация 2:

$$s_1^{(2)} = W^T x_q^{(1)} = \begin{pmatrix} -2 & -2 & 0 & 0 \\ 0 & 4 & -2 & 2 \\ 4 & 0 & 2 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ -8 \\ 8 \end{pmatrix};$$

$$y_1^{(2)} = f \begin{pmatrix} 0 \\ -8 \\ 8 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}.$$

Стабилизация выходного вектора наступила уже на второй итерации, и она оказалась идентичной первому выходному эталонному образу $(q = 1)$. Следовательно, нейронная сеть, моделирующая работу гетероассоциативной памяти, успешно решила задачу для зашумленного набора входных значений.

Нейронная сеть Кохонена

Нейронные сети Кохонена – типичный пример нейросетевой архитектуры, обучающейся без учителя. Отсюда и перечень решаемых ими задач: кластеризация данных или прогнозирование свойств.

Ранее рассмотренные архитектуры нейронных сетей обучались с учителем на выборках данных, включающих множество примеров, состоящих из соответствующих друг другу пар входных и выходных векторов. При этом выходные значения принимали самое непосредственное участие в настройке весовых коэффициентов. В нейронных сетях Кохонена (рис. 5.2) выходные вектора в обучающей выборке могут быть, но могут и отсутствовать, и в любом случае они не принимают участия в процессе обучения. Именно поэтому

данный принцип настройки нейронной сети называется самообучением.

В рассматриваемой архитектуре сигнал распространяется от входов к выходам в прямом направлении. Структура нейронной сети содержит единственный слой нейронов (слой Кохонена).

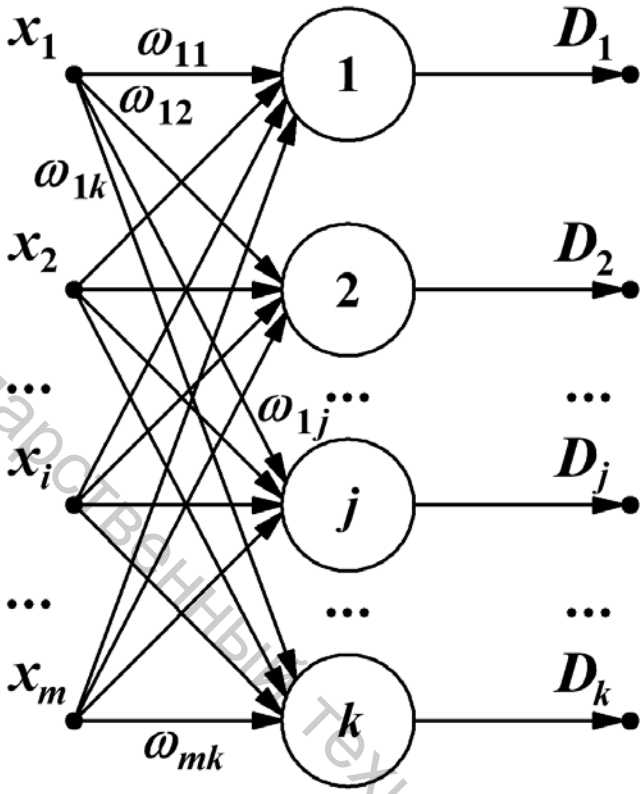


Рисунок 5.2 – Структура нейронной сети Кохонена

Задача классификации может быть сформулирована следующим образом. Пусть есть некоторое множество *A*. Мы хотим разбить его на классы 1, 2, 3, 4. Точки, которые будут принадлежать одному классу, объединены какими-то свойствами. В данном случае у нас нет обучающей выборки. Вначале исследования мы можем не знать, какие элементы относятся к какому классу. Возникает вопрос: «Можно ли классифицировать каким-то образом элементы множества *A,* просто указывая, что мы хотим разбить множество *A* на *k* классов?» Далее система обучается сама таким образом, что сможет классифицировать все наши элементы множества *A* на эти *k* классов.

Пусть у нас есть набор из *M* исходных образов, который нужно классифицировать, т. е. разбить на *K* классов. Каждый образ будем описывать некоторым вектором:

$$x^m = (x_1^m; x_2^m; ...; x_N^m),$$

где x_i^m – действительные числа, m=1, …, M.

Эти вектора мы хотим разбить на *K* классов.

Обучать мы будем весовые коэффициенты:

$$W^k = (\omega_1^k; \omega_2^k; ...; \omega_N^k),\ k=1,2,\ldots,K,$$

где *K* – количество классов, на которые мы разбиваем исходные образы.

Заметим, что каждый вектор весов имеет такую же размерность, как и входные вектора. Алгоритм обучения сети:

1. Нормировка исходных векторов $x^m = (x_1^m; x_2^m; ...; x_N^m)$. Все значения компонент должны быть взяты из интервала [0;1]. Причем нормировку будем производить по столбцам. Для этого мы должны вычислить минимумы и максимумы по столбцам наших данных. Вычисляем $Max_n = \max\limits_m x_n^m$, $Min_n = \min\limits_m x_n^m$ (т. е. находим максимум и минимум для каждой компоненты n для всех векторов m). Далее находим:

$$a_n = \frac{1}{Max_n - Min_n},\ b_n = \frac{-Min_n}{Min_n - Min_n}.$$

Для каждой компоненты определяем $x_n^m = a_n x_n^m + b_n$. Мы закончили нормировку.

2. Векторам $W^k = (\omega_1^k; \omega_2^k; ...; \omega_N^k)$ придаем случайные значения. Предполагается, что ω_n^k должна быть равномерно распределенная случайная величина $\omega_n^k \approx R[0{,}1;0{,}3]$. Инициализируем веса случайным образом значениями от 0,1 до 0,3.

3. Выбираем коэффициент обучения λ=0,3. Это параметр скорости обучения.

4. Пока λ>0, выполнять шаги 5-6-7.

5. Для каждого вектора x^m находим ближайший вектор ω^k.

6. И для ближайшего вектора веса мы выполняем процедуру обучения.

$$W^k = W^k + \lambda(x^m - W^k)$$ (записано в векторном виде).

Шаги 5 и 6 повторить несколько раз.

7. Уменьшаем коэффициент обучения

$$\lambda = \lambda - \Delta\lambda.$$

И возвращаемся к шагу 4.

8. К тому времени, когда λ станет отрицательным, заканчивается процесс обучения.

Мы получаем набор векторов W^k. Это уже обученные веса. Мы можем заняться задачей классификации. Для того чтобы определить, к какому классу относится вектор $x^m = (x_1^m; x_2^m; ...; x_N^m)$ (из набора или любой другой), надо найти ближайший вектор веса, т. е. такой W^k, который наиболее близок к x^m, и тогда вектор x^m принадлежит классу *k*.

Задания для контролируемой самостоятельной работы

5.1. Рассмотреть работу гетероассоциативной памяти для двух графических объектов.

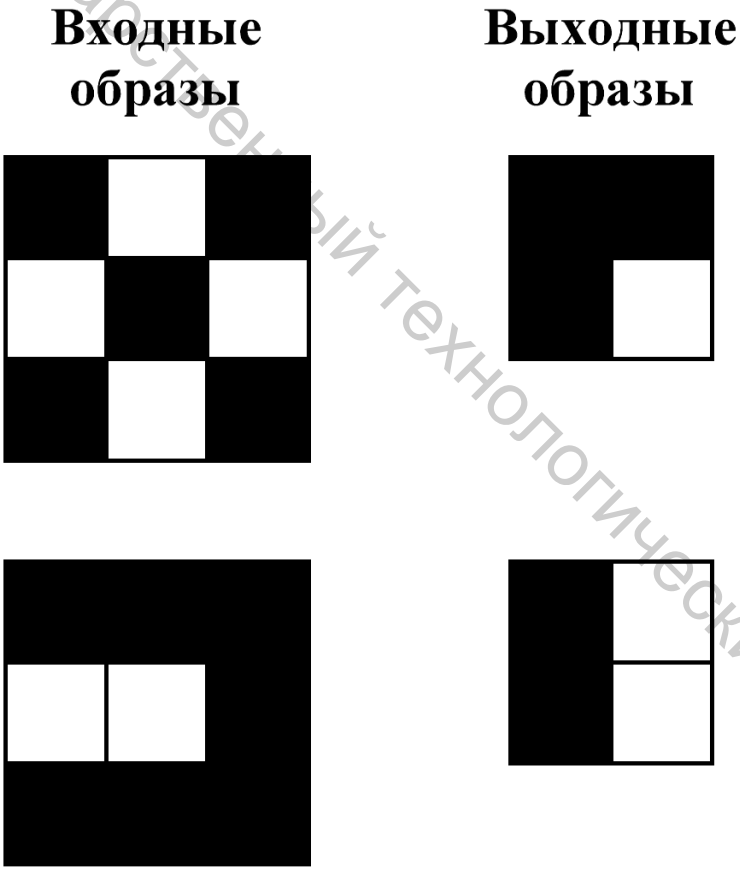


Рисунок 5.3 – Эталонные входные и выходные образы для обучения сети Коско

Для тестирования сети Коско воспользоваться зашумленным входным графическим образом. Выберите его согласно номеру в журнале и определите, какой выходной образ ему соответствует. Решение проводить до тех пор, пока наступит стабилизация выходного вектора.

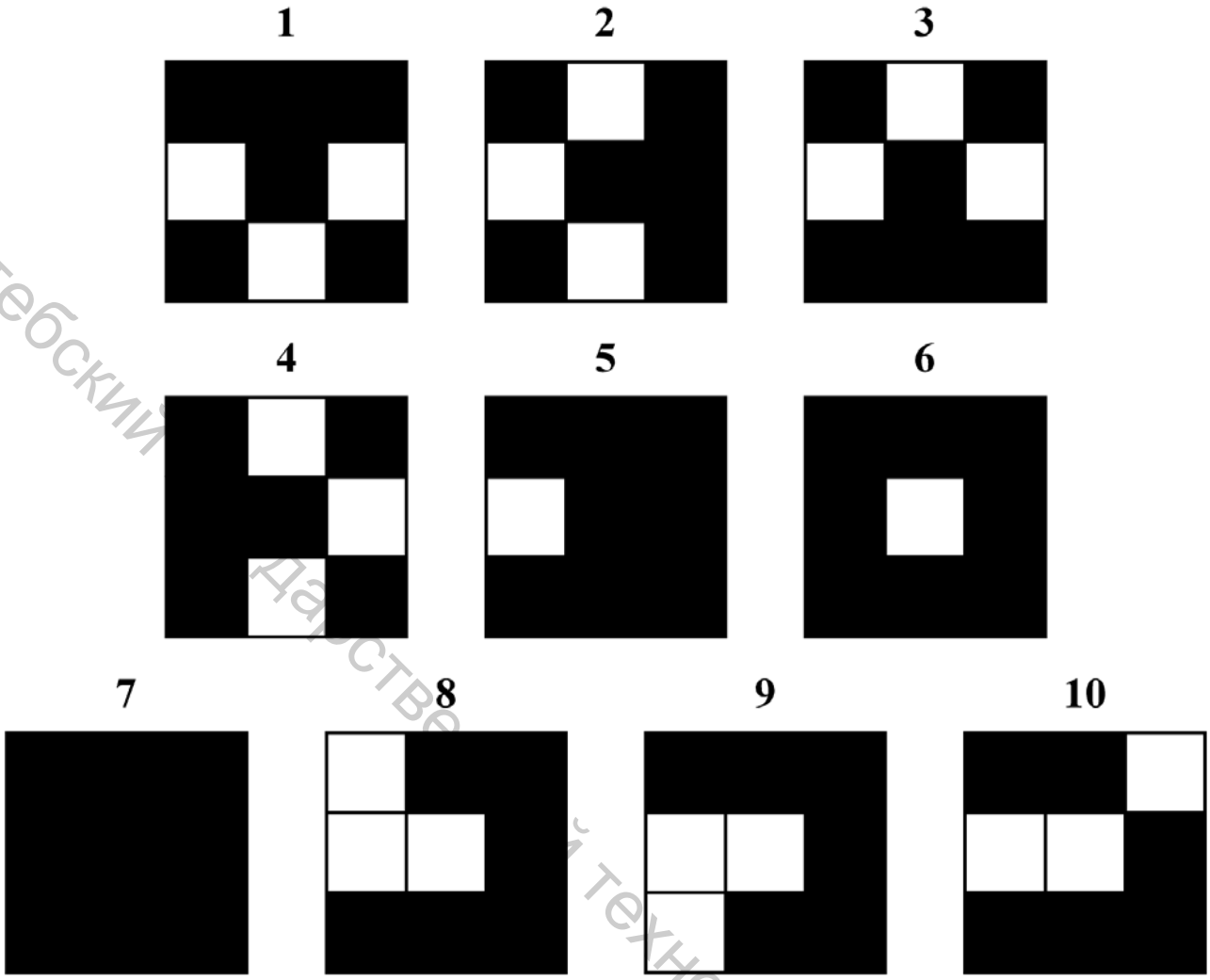


Рисунок 5.4 – Зашумленный входной графический образ

Примечание: при решении воспользоваться функцией активации:

$$sign(u) = f(u) = \begin{cases} 1, \text{ если } u > 0 \\ -1, \text{ если } u \le 0 \end{cases}.$$

5.2 Рассмотреть работу гетероассоциативной памяти для двух графических объектов (рис. 5.5).

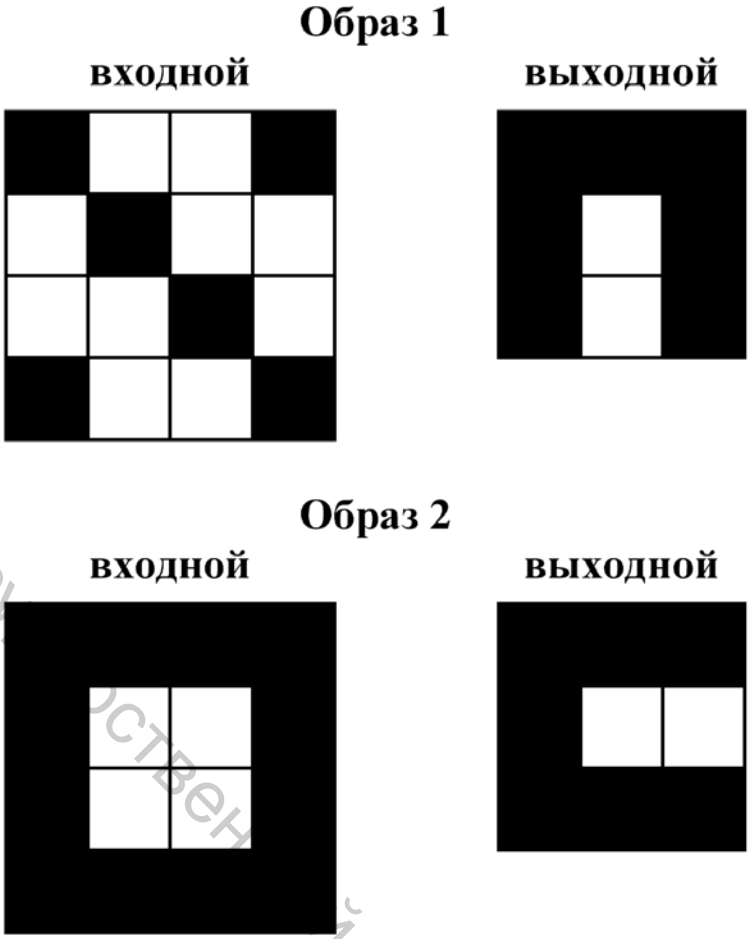
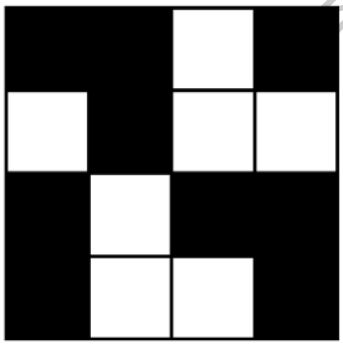


Рисунок 5.5 – Эталонные входные и выходные образы для обучения сети Коско

Для тестирования сети Коско воспользоваться зашумленным входным графическим образом:



Решение проводить до тех пор, пока наступит стабилизация выходного вектора.

Примечание: при решении воспользоваться функцией активации:

$$sign(u) = f(u) = \begin{cases} 1, \text{ если } u \geq 0 \\ -1, \text{ если } u < 0 \end{cases}.$$

ЛИТЕРАТУРА

1. Андрейчиков, А. В. Интеллектуальные цифровые технологии концептуального проектирования инженерных решений: учебник / А. В. Андрейчиков, О. Н. Андрейчикова. – Москва : ИНФРА-М, 2019. – 510 с.

2. Афонин, В. Л. Интеллектуальные робототехнические системы: курс лекций : учебное пособие / В. Л. Афонин, В. А. Макушкин. – Москва : Интернет-Университет информационных технологий, 2017. – 200 с.

3. Корячко, В. П. Интеллектуальные системы и нечеткая логика: учебник / В. П. Корячко, М. А. Бакулева, В. И. Орешков. – Москва : КУРС, 2017. – 347 с.

4. Станкевич, Л. А. Интеллектуальные системы и технологии: учебник и практикум для бакалавриата и магистратуры, для студентов высших учебных заведений, обучающихся по инженерно-техническим направлениям / Л. А. Станкевич. – Москва : Юрайт, 2019. – 397 с.

5. Уинстон, П. Искусственный интеллект = Artificialintelligence: [монография] / П. Уинстон; пер. с англ. В. Л. Стефанюка; под ред. Д. А. Поспелова. – Москва : Мир, 1980. – 520 с.

6. Шалютин, С. М. Искусственный интеллект. Гносеологический аспект / С. М. Шалютин. – Москва : Мысль, 1985. – 199 с.

7. Электронный учебно-методический комплекс по учебной дисциплине «Компьютерное зрение и распознавание образов» для специальности второй ступени высшего образования 1-53 80 01 «Автоматизация» [Электронный ресурс] / УО «ВГТУ» ; сост. Е. Б. Дунина. – Витебск, 2020. – 1 CD-ROM (10,2 Мб).

8. Сайт «Интуит» [Электронный ресурс] / Нейрокомпьютерные системы. – Режим доступа: https://intuit.ru/studies/courses/61/61/info.

9. Сайт «AI.Lector.ru» [Электронный ресурс] / Лекции по искусственному интеллекту. – Режим доступа: https://ai.lector.ru/?go=lection04.

Учебное издание

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Методические указания к практическим занятиям

Составители:
Дунина Елена Брониславовна
Жизневский Валерий Анатольевич
Соколова Анна Сергеевна

Редактор *Т.А. Осипова*
Корректор *Т.А. Осипова*
Компьютерная верстка *А.С. Соколова*

Подписано к печати 05.11.2020. Формат 60x90 1/16. Усл. печ. листов 3,0.
Уч.-изд. листов 3,8. Тираж 35 экз. Заказ № 310.

Учреждение образования «Витебский государственный технологический университет»
210038, г. Витебск, Московский пр., 72.
Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/172 от 12 февраля 2014 г.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1497 от 30 мая 2017 г.