Раздел 2 ТЕХНОЛОГИИ РЕШЕНИЯ ЗАДАЧ В ЭС

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Нейрон представляет собой следующую схему:

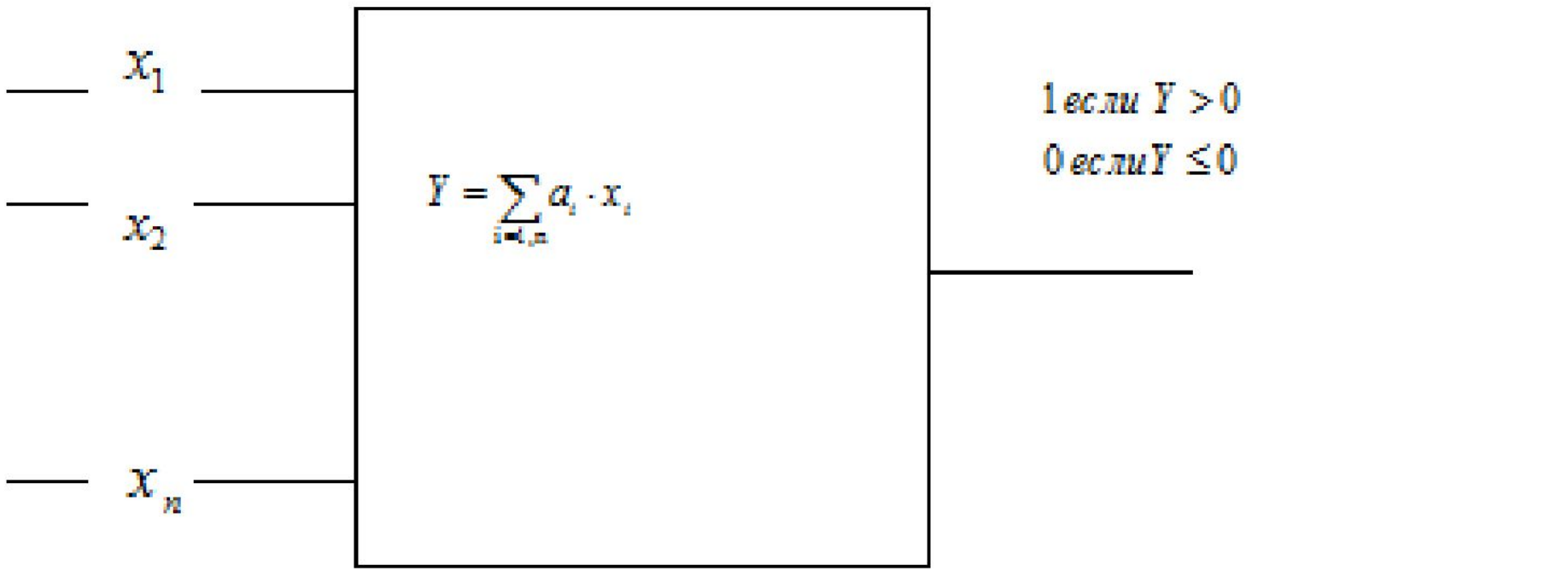

Рисунок 4.1

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

На вход нейрона поступают сигналы x_i . В наших рассмотрениях эти сигналы будут двоичными. Нейрон считает сумму

$$Y = \sum_{i=1,n} a_i \cdot x_i \quad . \ (4.1)$$

Коэффициенты a_i определяются в ходе обучения нейрона или путем решения системы линейных алгебраичесаких неравенств. Если сумма *Y* больше 0, то считается, что нейрон распознал входной вектор. В противном случае считают, что нейрон отклонил входной образец. Разумеется, можно было бы вычислять в нейроне и более сложные функции, однако в этом случае не ясно, как выполнять настройку коэффициентов a_i. Поэтому на практике нейроны объединяют в сеть, например, такого вида:

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

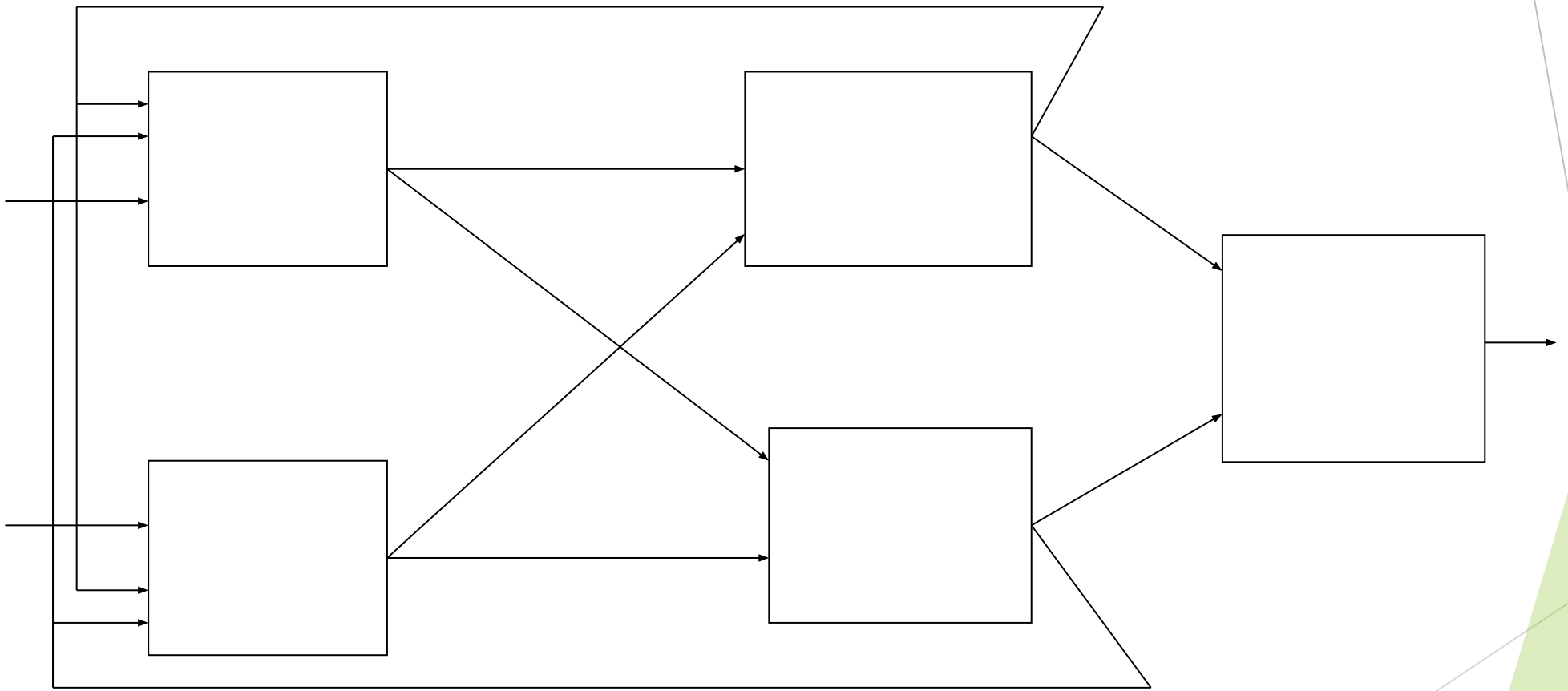

Рисунок 4.2

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Такая сеть является многослойной. В нашем примере – трехслойной. Трехслойная сеть также называется персептроном. **Первый** слой является входным, **второй** – промежуточным и **третий** – выходным. Каждый нейрон сети вычисляет простую линейную функцию. Однако благодаря обратным связям функция выходного слоя становится существенно нелинейной.

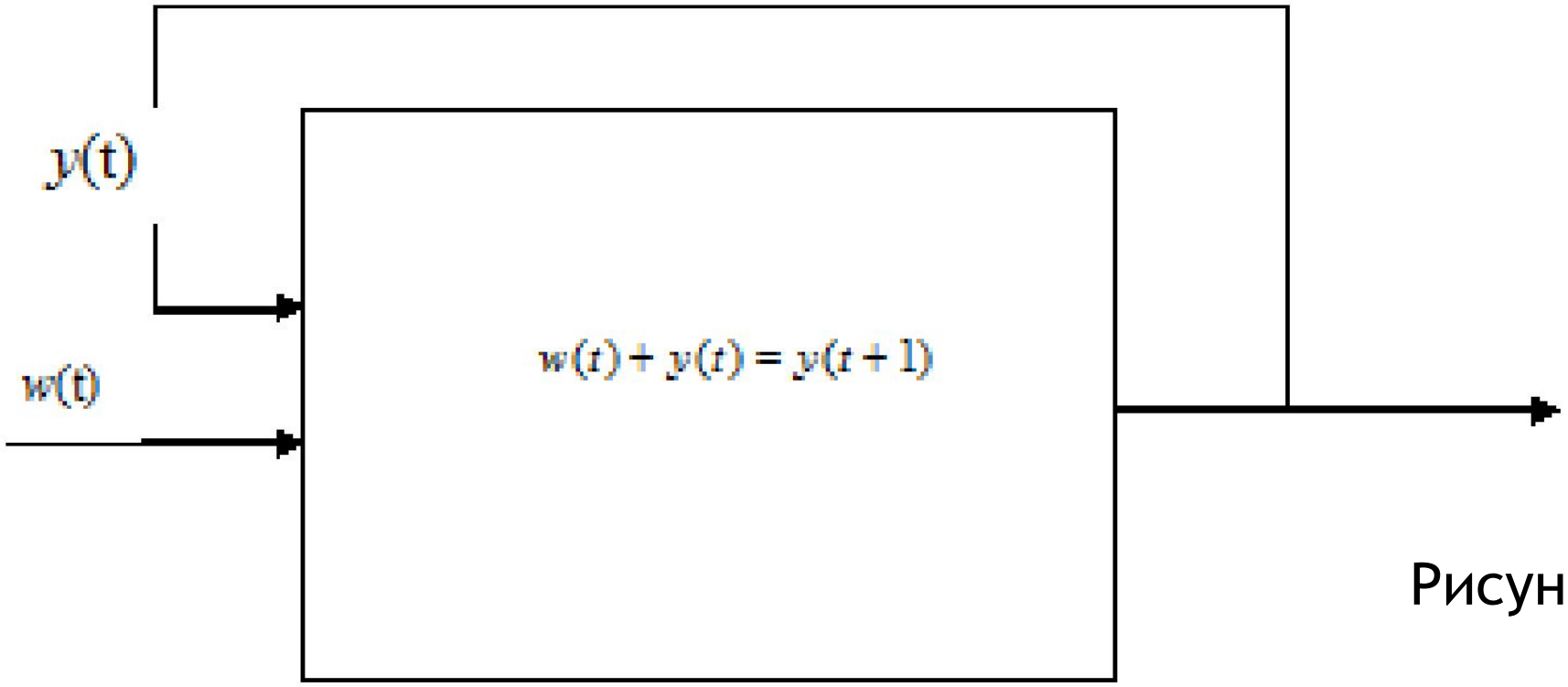

Рисунок 4.3

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Здесь вычисляется рекуррентная формула

$$w(t) + y(t) = y(t+1)$$

Для простоты примем, что $w(t) = 1$. Будем иметь

$$y(t+1) - y(t) = 1$$

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Кроме того, пусть $y(0) = 0$. Решая данное рекуррентное уравнение с помощью техники z-преобразования, получим, опуская все промежуточные выкладки:

$$y(t) = t.$$

Если иметь в виду, что преобразуется входная функция-константа от входа. Полученная функция существенно нелинейна по отношению к входному сигналу. Следовательно, наличие обратных связей в нейронной сети приводит к возникновению нелинейной зависимости выхода сети от ее входов. , то результат на выходе есть не что иное как интеграл

$$\int 1 \cdot dt = t$$

от входа. Полученная функция существенно нелинейно по отношению к входному сигналу. Следовательно, наличие обратных связей в нейронной сети приводит к возникновению нелинейной зависимости выхода сети от ее входов.

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Обратимся снова к нейрону на рисунке 4.1. Входы x_i можно связать с ключевыми словами. Если x_i=1, то соответствующее ключевое слово присутствует во входном текстовом образце (вопросе). Нейрон должен среагировать на вопрос, если этот вопрос относится к соответствующему тексту. Сначала покажем, как вычислять коэффициенты a_i. Для вычисления a_i составим уравнение:

$$\sqrt{(X_i - X_э)^2} \to \min .$$

Здесь X_i – вектор на входе нейрона; $X_э$ – эталонное значение (стандартный набор ключевых слов для данной темы). Значение функционала не изменится, если отбросить корень и рассматривать:

$$(X_i - X_э)^2 \to \min$$

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Раскроем скобки и получим:

$$X_i^2 - 2X_i \cdot X_э + X_э^2 \to \min$$

или

$$2X_i \cdot X_э - X_э^2 \to \max$$

В скалярной форме это выражение можно записать таким образом:

$$2x_1 \cdot x_{1э} + 2x_2 \cdot x_{2э} + \ldots + 2x_n \cdot x_{nэ} - x_{1э}^2 - x_{2э}^2 \ldots - x_{nэ}^2 \to \max$$

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Последнее выражение дает нам линейную распознающую функцию нейрона, если подставить эталонные значения для переменных $x_{iэ}$. Эти начальные значения, вообще говоря, не играют критической роли, поскольку коэффициенты нейрона настраиваются в процессе обучения. Для нейрона с линейной функцией распознавания процедура обучения такова. На вход нейрона последовательно подаются объекты, для которых известно, относятся они к данной теме, или нет. Если нейрон правильно реагирует на входной объект, то никакой коррекции коэффициентов не выполняется. Ошибочная реакция нейрона может состоять в следующем:

- нейрон отклонил правильный образец
- нейрон принял неправильный образец.

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

В первом случае коэффициенты нейрона пересчитываются по формуле:

$$a_i = a_i + \gamma \cdot x_i$$

Во втором случае пересчет выполняется по формуле:

$$a_i = a_i - \gamma \cdot x_i$$

Здесь γ – положительный коэффициент, обычно равный 1 и влияющий на скорость обучения. Чем меньше значение γ, тем длительнее процесс обучения, но точность его выше. Такой простой механизм обучения все же сойдется к результату, если обучающая выборка нейрона принципиально отделима на основе линейной функции вида (**4.1**), что графически может быть иллюстрировано таким образом:

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

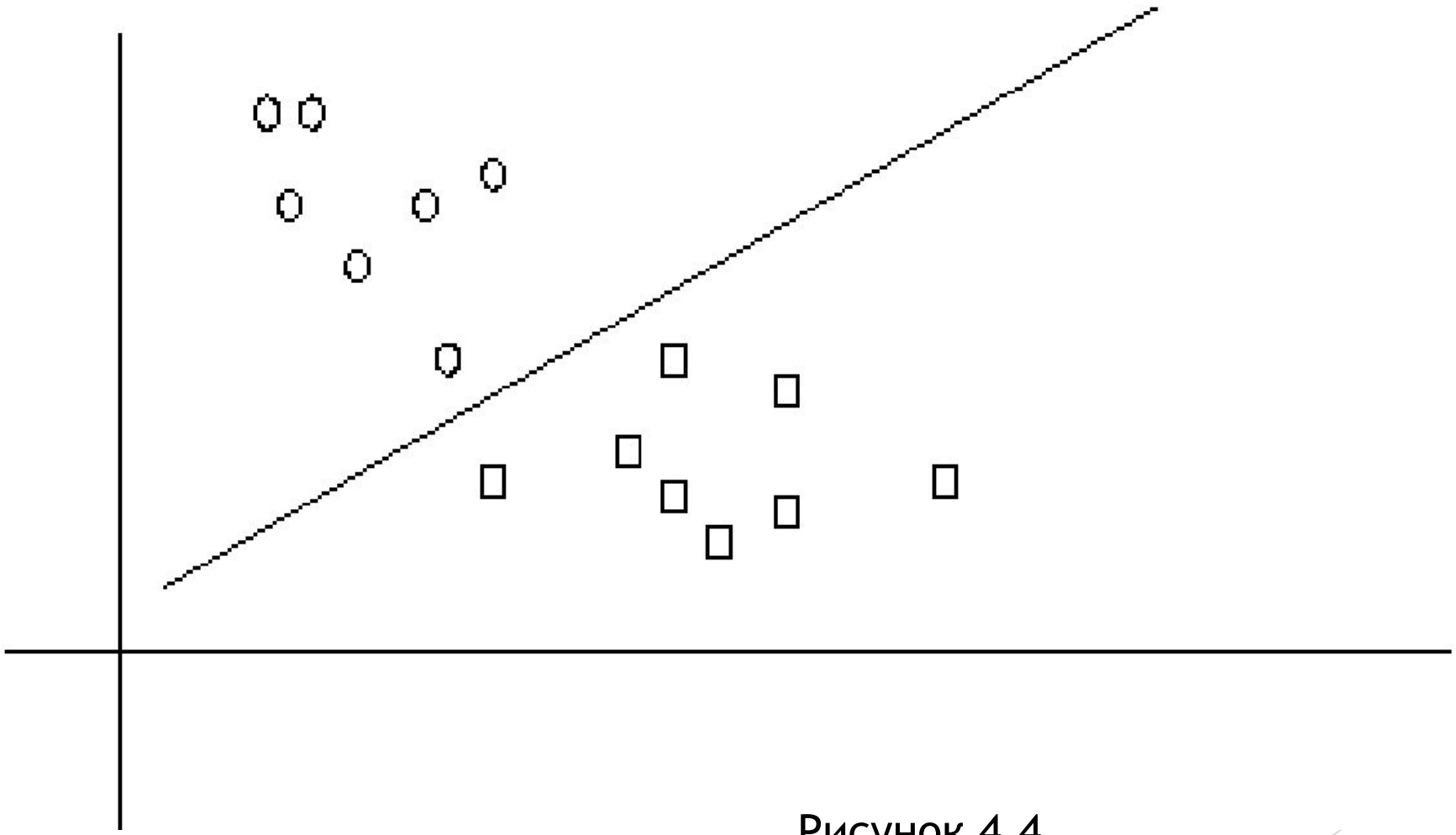

Рисунок 4.4

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

На рис.4.4 овалы соответствуют, например, входным образцам, относящимся к теме, а прямоугольники – входным образцам, не относящимся к теме. Два этих множества линейно отделимы. В этом случае процесс обучения завершится за конечное время. Рассмотрим пример. Пусть дана обучающая выборка следующего вида:

Класс	Документ	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*j*	*k*	*q*	*t*
w_1	D1	1	1	1	1	0	0	0	0	0	0	0	0
w_1	D2	0	1	1	0	1	0	0	0	0	0	0	0
w_1	D3	0	0	1	1	1	0	1	1	0	0	0	0
w_1	D4	1	0	1	0	0	0	0	0	1	0	0	0
w_2	D5	0	0	1	0	0	1	0	0	0	1	0	0
w_2	D6	0	0	0	0	0	1	0	0	1	0	1	0
w_2	D7	1	0	0	0	0	0	0	0	0	0	1	1
w_2	D8	0	0	0	0	0	0	0	0	0	1	1	1
w_2	D9	0	0	0	0	1	0	1	0	1	0	1	0

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Здесь двоичные переменнные обозначены литерами: *a-h,j,k,q,t* (играют роль ключевых слов). Пусть наш нейрон распознает класс w_1. Положим, что эталонный документ этого класса есть <1,1,1,1,1,0,0,0,0,0,0,0> (значения представлены в порядке следования имен переменных в столбцах таблицы). Тогда исходное значение распознающей функции нейрона таково:

$$2a+2b+2c+2d+2e+0f+0g+0h+0j+0k+0q+0t-5.$$

Согласно общей концепции, значение этой функции на объектах класса w_1 должно быть больше нуля, а вне этого класса – меньше либо равно 0. Мы видим, что для документа D4 с набором <10100000100> это правило не сработало (значение распознающей функции меньше 0, хотя документ относится к классу w_1). Корректируем коэффициенты по формулам (4.2, 4.3). Теперь распознающая функция примет такой вид:

$$3a+2b+3c+2d+2e+0f+0g+0h+1j+0k+0q+0t-5.$$

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Легко убедиться, что теперь все образцы обучающей таблицы распознаются верно, так что достаточно было только одной коррекции.

Конечно, можно «заставить» нейрон распознавать образцы с помощью существенно нелинейной функции распознавания и такую функцию можно построить следующим образом. Базовой является следующая.

Теорема (Колмогоров-Габор). Любую непрерывную дифференцируемую функцию можно аппроксимировать на заданном конечном интервале полиномом подходящей степени сколь угодно точно.

Эта теорема означает следующее. Пусть дана функция (жирная кривая черного цвета на рис.4.4). Эту кривую можно описать приблизительно прямой линией (линейной функцией) достаточно грубо. Более точную аппроксимацию дает кривая голубого цвета (полином примерно второй степени)

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

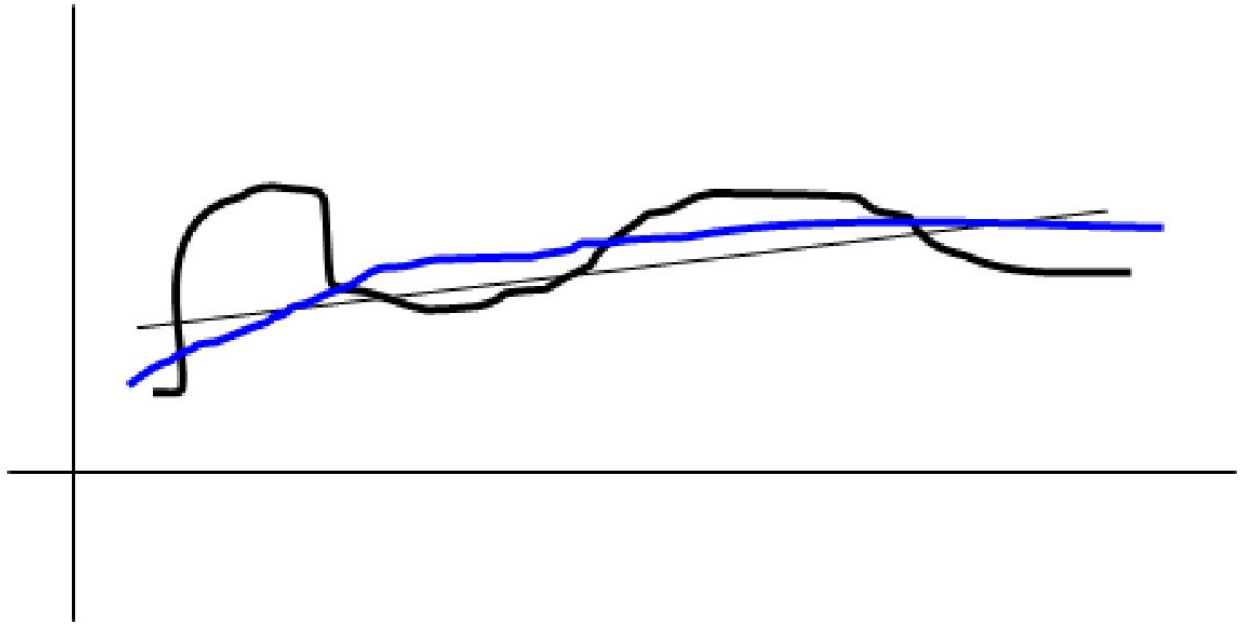

Рисунок 4.4

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Полином второй степени от двух переменных есть в общем виде функция

$$P(x_1, x_2) = c_1 x_1^2 + c_2 x_2^2 + c_3 x_1 x_2 + c_4 x_1 + c_5 x_2 \quad (2.4)$$

Полином третьей степени имеет уже существенно более громоздкий вид:

$$P(x_1, x_2) = c_1 x_1^3 + c_2 x_2^3 + c_3 x_1^2 + c_4 x_2^2 + c_5 x_1 x_2 + c_6 x_1 + c_7 x_2 + c_8 x_1^2 x_2 + c_9 x_1 x_2^2$$

и т.д.

Для отыскания коэффициентов полинома второй, третьей и более высокой степени можно использовать ту же технику, что и для полинома первой степени. Так, рассмотрим полином второй степени (2.4). Пусть обучающая выборка представлена следующей таблицей

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

x_1	x_2	**Класс**
1	1	A
1	0	A
1	0.5	A
0.5	0.5	B
0.2	4	B
0	0	B

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Допустим, что линейную функцию построить для этой таблицы нельзя.
Расширим таблицу членами второго порядка:

x_1	x_2	x_1^2	x_2^2	$x_1 x_2$	**Класс**
1	1	1	1	1	A
1	0	0	0	0	A
1	0.5	1	0.25	0.5	A
0.5	0.5	0.25	0.25	0.25	B
0.2	4	0.04	16	0.8	B
0	0	0	0	0	B

2.4 ПОИСК РЕШЕНИЙ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Теперь можно строить линейную функцию вида

$$Y = a_1 x_1 + a_2 x_2 + a_3 z_1 + a_4 z_2 + a_5 z_3$$

где

$$z_1 = x_1^2, z_2 = x_2^2, z_3 = x_1 x_2$$

Если и квадратичной функции не хватит для построения разделяющей кривой, то переходят к полиному третьего порядка и т.д. Теорема Колмогорова-Габора гарантирует, что при весьма общих условиях требуемый полином построить все же удастся.

Все-таки использование нейронных сетей представляется более простым способом построения разделяющей функции. Используя сеть с обратными связями и механизм обучения нейросети, можно, в принципе, построить распознающую систему не менее сильную, чем на основе нейрона с полиномиальной разделяющей функцией высокого порядка. Решающим этапом является процедура обучения нейросети. В настоящее время основу такой процедуры составляет алгоритм обратного распространения ошибки в нейросети. Выделяют входной слой нейронов, промежуточный слой нейронов и выходной слой нейронов. В процессе обучения корректируются веса связей от промежуточного (скрытого) слоя к выходному.

2.5 Алгоритм обратного распространения ошибки

Рассмотрим сеть на рис.5.1. Пусть на входной слой подана некая комбинация, приведшая к появлению сигналов OUT_i на выходе, которую можно сравнить с желаемой.

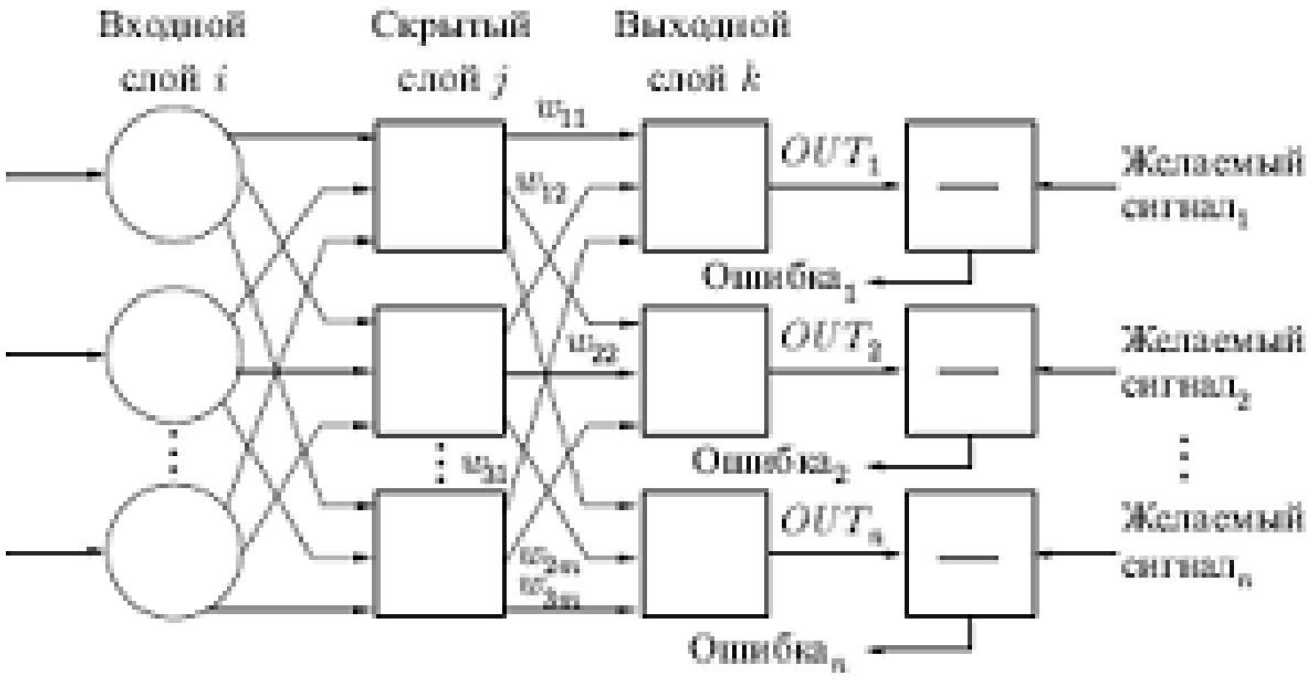

Рисунок 5.1

2.5 Алгоритм обратного распространения ошибки

Отклонение сигнала от желаемого трактуется как ошибка. Для каждого выхода вычисляется величина

$$\delta = OUT(1 - OUT)(T\,\text{arg}\,et - OUT) \qquad (5.1)$$

Здесь *Target* – ожидаемое значение сигнала на выходе. Значения всех сигналов являются двоичными. Величина *OUT(1-OUT)* представляет значение производной от сигмоидальной функции и характеризует скорость изменения сигнала на выходе. Пропорционально величине δ изменяется вес связи от нейрона предыдущего слоя к данному выходному нейрону по формуле

$$\Delta w_{i,k} = \eta \delta_k OUT_i$$
$$w_{i,k}(n+1) = w_{i,k}(n) + \Delta w_{i,k} \qquad (4.6)$$

Здесь OUT_j – выход j-го нейрона предыдущего слоя, поступающий на вход k-го выходного нейрона. Алгоритм обратного распространения позволяет обучить сеть, если условия удовлетворяют теореме Колмогорова-Габора. Эффективность обучения регулируется коэффициентом η.

2.5 Алгоритм обратного распространения ошибки

Отыскивать коэффициенты распознающей функции нейрона можно также с помощью алгоритма решения системы линейных алгебраических неравенств. Пусть дана таблица с данными, причем значение y определяет класс, к которому принадлежит данный вектор данных.

x_1	x_2	y
2	4	≥ 0
-1	3	≥ 0
0	1	≥ 0
2	5	≥ 0
3	-2	< 0
6	3	< 0
1	1	< 0
4	3	< 0

2.5 Алгоритм обратного распространения ошибки

По данной таблице запишем следующую систему неравенств:

$$\begin{aligned}
&2a_1 + 4a_2 \geq 0,\\
&-1a_1 + 3a_2 \geq 0,\\
&0a_1 + 1a_2 \geq 0,\\
&2a_1 + 5a_2 \geq 0,\\
&3a_1 - 2a_2 < 0,\\
&6a_1 + 3a_2 < 0,\\
&1a_1 + 1a_2 < 0,\\
&4a_1 + 3a_2 < 0.
\end{aligned} \qquad \textbf{(5.2)}$$

Неравенства (5.2) составлены согласно описанному принципу. Если найти коэффициенты a_1 и a_2, то все множество записей будет разделено нужным нам образом на две условные половины: «0» и «1». Имеются две проблемы. Во-первых, следует избавиться от жестких неравенств (<). Это сделать нетрудно, если ввести достаточно малую величину $\xi>0$, такую, что

$$c_1 a_1 + c_2 a_2 < 0$$

2.5 Алгоритм обратного распространения ошибки

можно заменить на

$$c_1 a_1 + c_2 a_2 \leq \xi$$

Заметим, что такая замена может сделать систему неравенств несовместной, однако , величина может быть выбрана исходя из пределов точности представления данной ЭВМ. В иллюстративных целях зададим ξ=1 и приведем все неравенства к виду ≥:

$$\begin{aligned}
2a_1 + 4a_2 &\geq 0, \\
-1a_1 + 3a_2 &\geq 0, \\
0a_1 + 1a_2 &\geq 0, \\
2a_1 + 5a_2 &\geq 0, \\
-3a_1 + 2a_2 &\geq 1, \\
-6a_1 - 3a_2 &\geq 1, \\
-1a_1 - 1a_2 &\geq 1, \\
-4a_1 - 3a_2 &\geq 1.
\end{aligned} \qquad (5.3)$$

2.5 Алгоритм обратного распространения ошибки

Для решения системы линейных алгебраических неравенств используем алгоритм устранения невязок.

Определение. Неравенство с положительной правой частью называется *невязкой*. Если в системе нет невязок, то ее решение доставляется нулевыми значениями переменных. В противном случае описываемый алгоритм пытается избавиться от невязок. При этом выделяются две фазы. На первой фазе нужно получить систему базисных неравенств вида $a_i \geq 0$. Вторая фаза выполняется также как и первая, но уже при наличии базисных неравенств. Если на второй фазе в процессе итераций встречается невязка, причем все коэффициенты в левой части неположительны, то устанавливаем факт неразрешимости (несовместности) системы неравенств. Этот факт мы будем использовать специальным образом. Итак, возьмем любую невязку, например,

$$-3a_1 + 2a_2 \geq 1$$

и выразим из нее переменную a_1 так

$$-3a_1 \geq 1 + 2a_2,$$

$$a_1 \leq -\frac{1}{3} - \frac{2}{3}a_2,$$

$$a_1 = -\frac{1}{3} - \frac{2}{3}a_2 - z_1. \qquad (5.4)$$

2.5 Алгоритм обратного распространения ошибки

Здесь z_1 – новая неотрицательная переменная. Подставим вместо a_1 выражение (5.4) в (5.3). Получим систему:

$$\begin{gathered}
\frac{8}{3}a_2 - 2z_1 \geq \frac{2}{3}, \\
\frac{11}{3}a_2 + z_1 \geq -\frac{1}{3}, \\
a_2 \geq 0, \\
\frac{11}{3}a_2 - 2z_1 \geq \frac{2}{3}, \\
z_1 \geq 0, \\
a_2 + 6z_1 \geq -1, \\
-\frac{1}{3}a_2 + z_1 \geq \frac{2}{3}, \\
-\frac{1}{3}a_2 + 4z_1 \geq -\frac{1}{3}.
\end{gathered} \qquad (5.5)$$

Первая фаза завершена. Имеются два базисных неравенства $a_2 \geq 0$ и $z_1 \geq 0$. Вторая фаза выполняется с небольшим отличием от первой: из невязки выражаем переменную с положительным коэффициентом. Так, возьмем невязку:

$$\frac{8}{3}a_2 - 2z_1 \geq \frac{2}{3}$$

2.5 Алгоритм обратного распространения ошибки

Выражаем a_2:

$$a_2 \geq \frac{1}{4} + \frac{3}{4} z_1,$$

$$a_2 = \frac{1}{4} + \frac{3}{4} z_1 + z_2. \qquad (5.6)$$

В (5.6) z_2 есть новая неотрицательная переменная. Подставляем (5.6) в (5.5):

$$z_2 \geq 0,$$

$$\frac{15}{4} z_1 + \frac{11}{3} z_2 \geq -\frac{5}{4},$$

$$\frac{3}{4} z_1 + z_2 \geq -\frac{1}{4},$$

$$\frac{3}{4} z_1 + \frac{11}{3} z_2 \geq -\frac{1}{4}, \qquad (5.7)$$

$$z_1 \geq 0,$$

$$\frac{27}{4} z_1 + z_2 \geq -\frac{5}{4},$$

$$\frac{3}{4} z_1 - \frac{1}{3} z_2 \geq \frac{3}{4},$$

$$\frac{15}{4} z_1 - \frac{1}{3} z_2 \geq -\frac{1}{4}.$$

2.5 Алгоритм обратного распространения ошибки

Осталась одна единственная невязка:

$$\frac{3}{4}z_1 - \frac{1}{3}z_2 \geq \frac{3}{4},$$

$$z_1 = 1 + \frac{4}{9}z_2 + z_3. \qquad (5.8)$$

Подстановка (5.8) приводит к системе без невязок. В этой системе решение дает $z_2 = z_3 = 0$. Отсюда в силу подстановок найдем: a_1=-2, a_2=1.

Практическая апробация данного алгоритма показала, что в среднем он порождает

(1.2-1.5)* *M* подстановок, где *M* – число неравенств исходной системы. При несовместности системы неравенств (т.е. невозможности ее разрешить) на некоторой итерации обязательно получится невязка, в левой части которой все коэффициенты при переменных отрицательны либо нулевые.

2.6 ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И РАСПОЗНАВАНИЕ ОБРАЗОВ

В основе генетических (эволюционных) алгоритмов лежит простая идея: берут пару наилучших образцов из порожденной совокупности объектов и порождают их потомка путем взаимного обмена характеристиками (этот механизм называется кроссовером). Потомков порождают все время из элитных представителей текущей совокупности. Когда наступает момент и не удаётся улучшить требуемый функционал, производят мутацию, т.е. порождают новые объекты не в результате «скрещивания» родительских объектов, а путем случайных изменений. Эта процедура ведется до тех пор, пока удается улучшить характеристики функционала.

Пример. Пусть требуется максимизировать функцию

$$F(x,y) = 2 \cdot x + 4 \cdot y - x \cdot y,$$
$$x + y \le 11 \qquad (6.1)$$
$$x \ge 0$$
$$y \ge 0$$

2.6 ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И РАСПОЗНАВАНИЕ ОБРАЗОВ

Сформируем начальную популяцию случайным образом

№	x	y	$F(x,y)$
1	0	1	4
2	1	7	1
3	2	3	10
4	4	2	4
5	9	0	18
6	1	1	5
7	5	5	5
8	2	9	22
9	6	1	10

2.6 ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И РАСПОЗНАВАНИЕ ОБРАЗОВ

Выберем элиту из 4 наилучших образцов, где функция достигает наибольших значений:

№	x	y	$F(x,y)$
3	2	3	10
5	9	0	18
8	2	9	22
9	6	1	10

Породим потомков, например, «скрестим» пятый и восьмой объект путем обмены координат; получим, например:
<9,9>, <2,0>,<9,2>,<0,9> (допускаем перекрестный обмен координат). Объекты <9,9> не допускается ограничениями. Таблица примет такой вид:

№	x	y	$F(x,y)$
3	2	3	10
5	9	0	18
8	2	9	22
9	6	1	10
10	9	9	-27
11	2	0	2
12	9	2	8
13	0	9	36

2.6 ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И РАСПОЗНАВАНИЕ ОБРАЗОВ

Видим, что лишь один потомок дал существенный прирост функционала (6.1). Плохих потомков отбросим и будем манипулировать теперь следующей популяцией

№	x	y	$F(x,y)$
3	2	3	10
5	9	0	18
8	2	9	22
9	6	1	10
13	0	9	36

Скрестим, например, девятого и тринадцатого:
<6,9>,<1,9>,<0,1>,<9,1>. Из этих объектов только <1,9> дает сильного потомка. Наша таблица принимает такой вид:

№	x	y	$F(x,y)$
3	2	3	10
5	9	0	18
8	2	9	22
9	6	1	10
13	0	9	36
14	1	9	29

2.6 ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И РАСПОЗНАВАНИЕ ОБРАЗОВ

Нам не удалось улучшить целевую функцию путем скрещивания. Осуществим мутацию наилучших двух образцов. Породим объекты:

<0,10>,<1,10>. Таблица примет такой вид

№	x	y	$F(x,y)$
3	2	3	10
5	9	0	18
8	2	9	22
9	6	1	10
13	0	9	36
14	1	9	29
15	0	10	40
16	1	10	32

2.6 ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И РАСПОЗНАВАНИЕ ОБРАЗОВ

Таким образом, удалось улучшить функционал. Снова возобновляем скрещивания и т.д., и т.п. Процесс завершаем, когда ни скрещивание, ни мутация не улучшают функционал.

Распознавание образов является самостоятельной и сложной задачей. Несколько измененную технологию генетических алгоритмов можно использовать также и для решения задачи распознавания.

Вначале образуется некоторая исходная популяция (множество объектов из различных комбинаций признаков), над которой проводятся операции скрещивания (у родителей произвольным образом выбирается точка раздела, относительно которой они обмениваются частями) или мутации (случайно выбранный параметр заменяется другим). Затем из полученного поколения выбираются наиболее рациональные комбинации, которые выступают в роли новых родителей, и процесс запускается заново. Это происходит до тех пор, пока возможно улучшение потомков. Если же улучшение невозможно, то из полученных объектов выбирается наилучший, который будет представлять идеальный объект данной популяции $X^* = \langle x^*_1, x^*_2, \ldots, x^*_n \rangle$.

2.6 ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И РАСПОЗНАВАНИЕ ОБРАЗОВ

Теперь, пусть требуется распознать, относится ли объект $X_i = < x_{i1}, x_{i2}, \ldots, x_{in} >$

где разряд x_{ij} представляет числовое значение критерия K_j объекта, к рассматриваемому классу. Для этой цели необходимо использовать интегральную функцию выбора в форме

$$F = 2 \cdot x_1^* \cdot x_{i1} + 2 \cdot x_2^* \cdot x_{i2} + \ldots + 2 \cdot x_n^* \cdot x_{in} - \sum_{i=1} (x_i^*)^2$$

Если значение этой функции неотрицательно, то объект $X_i = < x_{i1}, x_{i2}, \ldots, x_{in} >$

принадлежит к рассматриваемому классу, в противном случае – не принадлежит.