МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
НЕВИННОМЫССКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ
(ФИЛИАЛ) СКФУ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по дисциплине

«ИНТЕЛЛЕКТУАЛИЗАЦИЯ СИСТЕМ УПРАВЛЕНИЯ»

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

Методические указания к выполнению лабораторных

работ для студентов направления

15.03.04 – «Автоматизация технологических процессов»

Невинномысск 2021

Методические указания предназначены для студентов направления 15.03.04 – «Автоматизация технологических процессов» и других технических специальностей и направлений. Они содержат основы теории, описание опытных установок, порядок проведения лабораторных работ и обработки экспериментальных данных, перечень контрольных вопросов для самоподготовки и список рекомендуемой литературы. Работы подобраны и расположены в соответствии с методикой изучения дисциплины «Интеллектуализация систем управления». Объем и последовательность выполнения работ определяются преподавателем в зависимости от количества часов, предусмотренных учебным планом дисциплин.

Методические указания разработаны в соответствии с требованиями ФГОС ВО в части содержания и уровня подготовки выпускников направления 15.03.04 – «Автоматизация технологических процессов».

Составители канд. техн. наук, доцент Э.Е. Тихонов

Ответственный редактор канд. техн. наук, доцент Болдырев Д.В.

Содержание

Введение

В последние годы активно используются модели и методы, основанные на теории нечетких множеств, гибридных и нейронных сетей. Системы искусственного интеллекта применяются для решения задач аппроксимации, кластеризации, распознавания образов, а также принятия решений и управления при нечеткой и неформализованной исходной информации. В настоящее время сформирована совокупность методов и приемов использования искусственного интеллекта, доказана ее эффективность и практическая значимость.

Данное учебно-методическое пособие предназначено для изучения способов практической реализации нечетких и гибридных систем, нейронных сетей в системах управления.

В результате выполнения представленных лабораторных работ студенты изучат способы построения систем нечеткого вывода и построения адаптивных нечетких систем (гибридных нейронных сетей). Полученные практические знания позволят студентам направления 15.03.04 – «Автоматизация технологических процессов» разрабатывать Интеллектуализация систем управления на базе аппарата нечеткой логики и нейронных сетей.

Нейронные сети – это раздел искусственного интеллекта, в котором для обработки сигналов используются явления, аналогичные происходящим в нейронах живых существ. Важнейшая особенность сети, свидетельствующая о ее широких возможностях и огромном потенциале, состоит в параллельной обработке информации всеми звеньями. При громадном количестве межнейронных связей это позволяет значительно ускорить процесс обработки информации. Во многих случаях становится возможным преобразование сигналов в реальном времени. Кроме того, при большом числе межнейронных соединений сеть приобретает устойчивость к ошибкам, возникающим на некоторых линиях. Функции поврежденных связей берут на себя исправные линии, в результате чего деятельность сети не претерпевает существенных возмущений.

Другое не менее важное свойство – способность к обучению и обобщению накопленных знаний. Нейронная сеть обладает чертами искусственного интеллекта. Натренированная на ограниченном множестве данных сеть способна обобщать полученную информацию и показывать хорошие результаты на данных, не использовавшихся в процессе обучения.

Характерная особенность сети состоит также в возможности ее реализации с применением технологии сверхбольшой степени интеграции. Различие элементов сети невелико, а их повторяемость огромна. Это открывает перспективу создания универсального процессора с однородной структурой, способного перерабатывать разнообразную информацию.

В учебно-методическом пособии в краткой форме приведены теоретические сведения об основных парадигмах НС и примеры их реализации в компьютерной среде математического моделирования MATLAB для решения типовых задач.

Лабораторная работа № 1 «Исследование мозжечковой модели суставного регулятора»

Цель работы: Изучить свойства и возможности ассоциативной нейронной сети CMAC.

Программа работы

1. Исследовать аппроксимирующие возможности сети CMAC.
2. Исследовать обобщающую способность нейронной сети CMAC.
3. Составить и защитить отчет по результатам исследований, в котором должны быть приведены архитектура CMAC с пояснением принципа ее работы, графики изменения ошибки обучения сети CMAC, выводы по обобщающей способности и емкости сети CMAC.

Краткие сведения из теории

CMAC – сокращенное название нейронной сети, которое происходит от первых букв ее полного английского названия: Cerebellar Model Articulation Controller (мозжечковая модель суставного регулятора). Эта нейронная сеть, в основу которой положена нейрофизиологическая модель мозжечка, была разработана американским ученым Альбусом. Первоначально нейронная сеть CMAC была предназначена для управления роботом-манипулятором. В последующих работах Толле и его соавторов было показано, что эта нейронная сеть может быть успешно применена для идентификации и управления нелинейными динамическими объектами.

CMAC предназначена для запоминания, восстановления и интерполяции функций многих переменных. Как и в любой нейронной сети, в CMAC существуют два основных принципиально различных процесса:

- процесс обучения, который осуществляется по измерениям значений функции и ее векторного аргумента с помощью соответствующего алгоритма;
- процесс восстановления, когда по входному вектору восстанавливается или оценивается значение этой функции.

Структура нейронной сети CMAC

Следующие два отличительных момента характеризуют эту нейронную сеть, которая, как отмечалось выше, предназначена для запоминания и восстановления функций многих переменных:

- значения аргументов функции принимают только дискретные значения;
- нелинейное преобразование аргументов функции осуществляется неявно с помощью алгоритма вычисления адресов ячеек ассоциативной памяти, в которых хранятся числа, определяющие значение функции.

Область определения функции в CMAC

Чтобы запомнить функцию многих переменных с помощью нейронной сети CMAC, сначала задается область определения этой функции на гиперпараллелепипеде, вершинами которого являются минимальные и максимальные значения аргументов этой функции; далее ребра гиперпараллелепипеда квантуются с постоянным шагом по каждому ребру, и дискретизованным значениям аргументов присваиваются целочисленные номера. Эти номера однозначно связаны со значениями аргументов функции. Так, для функции одной переменной *y(x)* аргумент x_i связан со своим номером *i* следующим соотношением

$$x_i = x_{\min} + (i-1)\Delta_x, \qquad (1)$$

где $\Delta_x = \dfrac{x_{\max} - x_{\min}}{i_{\max} - 1}$ – шаг квантования; $x_{\min}$, $x_{\max}$ – минимальное и максимальное значения аргумента *x* из области определения; $i = \overline{1, i_{\max}}$, где $i_{\max}$ – число точек, на которых задана функция *y(x)*.

Номер *i* можно рассматривать как относительный аргумент функции, поскольку соотношение (1) описывает процедуру сдвига и масштабирования исходной области определения функции. Поэтому в дальнейшем будем считать, что значения аргументов запоминаемой в CMAC функции

принимают только целочисленные значения, т. е. функция *y(x)* от*N* переменных $x = (x_1, x_2, ..., x_N)^T$ определена на целочисленной *N*-мерной сетке *y(x):*

$$X = \{x_1 = \overline{1, x_{\max}^{(1)}}; x_2 = \overline{1, x_{\max}^{(2)}}; ...; x_N = \overline{1, x_{\max}^{(N)}}\}. \quad (2)$$

Алгоритм нелинейного преобразования аргументов, или алгоритм вычисления адресов ассоциативной памяти

При построении алгоритма вычисления адресов ассоциативной памяти и алгоритма обучения CMAC ее автор [1] исходил из нейрофизиологического подхода. Согласно этому подходу, каждый входной сигнал возбуждает определенную область мозжечка, суммарная энергия которой соответствует значению запоминаемой функции. В нейронной сети CMAC предполагается, что каждый входной сигнал (аргумент функции) возбуждает, или делает активными, ровно ρ^* ячеек памяти, суммарное содержание которых равно значению запоминаемой функции. Следовательно, даже в случае скалярного аргумента $x = x_1$ ему соответствует ρ^*-мерный вектор номеров активных ячеек памяти. Параметр ρ^* играет очень важную роль. В литературе он известен как обобщающий параметр (generalization parameter), его значение определяет разрешающую способность CMAC и требуемый объем памяти.

Одномерный случай

Каждому значению скалярного аргумента $x_l = \overline{1, x_{\max}^{(l)}}$ соответствует ровно ρ^* активных ячеек памяти с номерами

$$x_1, x_1 + 1, x_1 + 2, ..., x_1 + \rho^* - 1, \quad (3)$$

так что максимальному значению аргумента $x_1 = x_{\max}^{(1)}$ соответствуют ячейки памяти с номерами $x_{\max}^{(1)}, x_{\max}^{(1)} + 1, x_{\max}^{(1)} + 2, ..., x_{\max}^{(1)} + \rho^* - 1$, откуда следует, что число ячеек памяти, необходимых для хранения функции одной переменной в CMAC, равно

$$M^{(1)} = x_{\max}^{(1)} + \rho^* - 1, \quad (4)$$

что на $\rho^* - 1$ больше числа значений функции, запоминаемых нейронной сетью.

Таким образом, в одномерном случае нейронная сеть CMAC, с точки зрения требуемого объема памяти, неэффективна, однако ее эффективность проявляется при достаточно точном запоминании гладких функций по небольшому числу обучающей последовательности $n \ll M^{(l)}$.

Очевидно, что номера активных ячеек памяти, соответствующие скалярной переменной $x_1 = \overline{1, x_{\max}^{(l)}}$, можно представить ρ^*-мерным вектором $m^{(1)}$, значения компонент которого равны номерам активных ячеек. Из последующего изложения следует, что последовательность значений компонент $m_k^{(l)}$, $k = \overline{1, \rho^*}$, вектора $m^{(1)}$ в одномерном случае не существенна. В многомерном случае она, однако, приобретает принципиальное значение.

Далее приводится алгоритм формирования вектора номеров активных ячеек для одномерного случая, поскольку он служит основой построения вектора номеров активных ячеек в многомерном случае. Этот алгоритм обеспечивает (в соответствии с приведенным в работе [1] требованием) однозначное соответствие номера компоненты *k* вектора $m^{(1)}$ и ее значения $m_k^{(1)}$.

Введем функцию

$$a \overline{\bmod} b = \begin{cases} b, & a \bmod b = 0; \\ a \bmod b, & a \bmod b \neq 0, \end{cases} \quad (5)$$

где $a \bmod b$ – остаток от деления *a* на *b*, где *a* и *b* – целые числа.

Тогда значения всех ρ^* компонент $m_k^{(l)}, l = \overline{1, N}, k = \overline{1, \rho^*}$ вектора $m^{(l)}$ вычисляются согласно выражению

$$m^{(l)}_{[(x_i \overline{\mathrm{mod}}\,\rho^*+i)\overline{\mathrm{mod}}\,\rho^*]} = x_l + i, i = \overline{0, \rho^* - 1}. \quad (6)$$

Для вычисления компонент вектора $m^{(l)}$ согласно выражению (6) достаточно определить номер $x_1 \overline{\mathrm{mod}}\,\rho^*$ той компоненты, значение которой равно x_1, т. е. вычислить (6) при $i = 0$. Значения следующих компонент равны $x_1 + 1$, $x_1 + 2$ и т. д., при этом полагается, что номера компонент замкнуты в кольцо, т. е. после компоненты с номером ρ^* следует компонента с номером 1. В качестве поясняющего примера в таблице приведены выражения для вектора $m^{(1)}$ размерности $\rho^* = 8$ для ряда значений аргумента $x_1 = 22, 23, 24, 25, 26, 27, 28, 29, 30$.

Из таблицы следует, что если двум различным значениям аргумента x_1 соответствуют одинаковые номера активных ячеек, то эти номера всегда являются одними и теми же компонентами вектора $m^{(1)}$; при этом первой компонентой вектора $m^{(1)}$ оказываются такие номера ячеек, остаток от деления которых на ρ^* равен единице; второй компонентой вектора $m^{(1)}$ оказываются такие номера ячеек, остаток от деления которых на ρ^* равен двум, и т. д.

Из последовательности (3) и соотношения (6) следует, что если для двух значений аргумента x_1, равных $x_1 = c$ и $x_1 = d$, выполняется условие

$$|c - d| = \rho < \rho^* \quad (7)$$

$x_1 = 22$	$m^{(1)} = (25, 26, 27, 28, 29, 22, 23, 24)\ 22 \overline{\mathrm{mod}} 8 = 6$
$x_1 = 23$	$m^{(1)} = (25, 26, 27, 28, 29, 30, 23, 24)\ 23 \overline{\mathrm{mod}} 8 = 7$
$x_1 = 24$	$m^{(1)} = (25, 26, 27, 28, 29, 30, 31, 24)\ 24 \overline{\mathrm{mod}} 8 = 8$
$x_1 = 25$	$m^{(1)} = (25, 26, 27, 28, 29, 30, 31, 32)\ 25 \overline{\mathrm{mod}} 8 = 1$
$x_1 = 26$	$m^{(1)} = (33, 26, 27, 28, 29, 30, 31, 32)\ 26 \overline{\mathrm{mod}} 8 = 2$
$x_1 = 27$	$m^{(1)} = (33, 34, 27, 28, 29, 30, 31, 32)\ 27 \overline{\mathrm{mod}} 8 = 3$
$x_1 = 28$	$m^{(1)} = (33, 34, 35, 28, 29, 30, 31, 32)\ 28 \overline{\mathrm{mod}} 8 = 4$
$x_1 = 29$	$m^{(1)} = (33, 34, 35, 36, 29, 30, 31, 32)\ 29 \overline{\mathrm{mod}} 8 = 5$
$x_1 = 30$	$m^{(1)} = (33, 34, 35, 36, 37, 30, 31, 32)\ 30 \overline{\mathrm{mod}} 8 = 6$

то число общих ячеек памяти *j*, соответствующее этим значениям аргумента, равно $\Delta\rho = \rho^* - \rho$, т. е. чем меньше расстояние между двумя значениями аргумента, тем больше общее число ячеек памяти, соответствующих этим значениям аргумента.

Многомерный случай

В многомерном случае, когда нейронная сеть CMAC запоминает функцию *y(x)* *N* переменных $x = (x_1, x_2, ..., x_N)^T$, заданных на целочисленной сетке $X = \{x_1 = \overline{1, x^{(1)}_{\max}}; x_2 = \overline{1, x^{(2)}_{\max}}; ...; x_N = \overline{1, x^{(N)}_{\max}}\}$ входному *N*-мерному вектору *x* также ставится в соответствие ρ^*-мерный вектор *m* активных ячеек памяти. Вычисление номеров активных ячеек памяти выполняется с помощью описываемого далее алгоритма.

Каждой компоненте $x_l, l=\overline{1,N}$ вектора *x* соответствует ρ^*-мерный вектор $m^{(1)}$ активных ячеек памяти. Значения всех ρ^* – компонент вектора $m^{(1)}$ – вычисляются, как это было описано выше в одномерном случае, согласно выражению (6).

В результате этих вычислений вектору *x* ставится в соответствие промежуточная матрица M активных ячеек памяти с элементами $m_k^{(l)}, l=\overline{1,N}, k=\overline{1,\rho^*}$:

$$x=\begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{vmatrix} \rightarrow \begin{vmatrix} m_1^{(1)} & m_2^{(1)} & m_3^{(1)} & \dots & m_{\rho^*}^{(1)} \\ m_1^{(2)} & m_2^{(2)} & m_3^{(2)} & \dots & m_{\rho^*}^{(2)} \\ m_1^{(3)} & m_2^{(3)} & m_3^{(3)} & \dots & m_{\rho^*}^{(3)} \\ \dots & \dots & \dots & \dots & \dots \\ m_1^{(N)} & m_2^{(N)} & m_3^{(N)} & \dots & m_{\rho^*}^{(N)} \end{vmatrix}. \quad (8)$$

На следующем шаге матрице M размера $N\times\rho^*$ ставится в соответствие ρ^*-мерный вектор активных ячеек памяти вектора *x*. Для этого выполняется последовательное слияние (concatenation) элементов каждого *k*-го столбца матрицы M в один *k*-й элемент вектора активных ячеек памяти *m,* так что этот вектор приобретает следующий вид:

$$m=(m_1^{(1)}\, m_1^{(2)}\, m_1^{(3)} \dots m_1^{(N)}; m_2^{(1)}\, m_2^{(2)}\, m_2^{(3)} \dots m_2^{(N)}; \dots;$$
$$m_{\rho^*}^{(1)}\, m_{\rho^*}^{(2)}\, m_{\rho^*}^{(3)} \dots m_{\rho^*}^{(N)})^T,$$

каждый элемент которого

$$m_k = m_k^{(1)}\, m_k^{(2)}\, m_k^{(3)} \dots m_k^{(N)}, \quad k=\overline{1,\rho^*}, \quad (9)$$

однозначно определяет номер активной ячейки памяти CMAC. Выражение (9) можно трактовать как позиционную запись числа m_k, старший разряд которого равен $m_1^{(1)}$, а младший разряд – $m_1^{(N)}$. К особенностям чисел, образующих символьную запись числа m_k (9), относится отмеченный выше факт, заключающийся в том, что остаток от деления каждого из них на ρ^* равен *k*. Алгоритм вычисления числа m_k по значениям $m_k^{(l)}, l=\overline{1,N}$, принимает наиболее простую форму, когда максимально возможные значения компонент вектора *x* удовлетворяют следующему соотношению:

$$x_{\max}^{(l)} = (\mu^{(l)}-1)\rho^*+1, \quad l=\overline{1,N} \quad (10)$$

Согласно (4), число ячеек памяти, необходимых для хранения функций одной переменной в CMAC, оказывается равным

$$M^{(l)} = \mu^{(l)}\rho^*, \quad l=\overline{1,N}. \quad (11)$$

В этом случае алгоритм вычисления m_k, который приводится без доказательства, имеет вид

$$m_k = (m_k^{(l)}-1)\prod_{l=2}^{N}\mu^{(l)} + \text{int}((m_k^{(2)}-1)/\rho^*)\prod_{l=3}^{N}\mu^{(l)} + \dots \quad (12)$$
$$\dots + \text{int}((m_k^{(N-1)}-1)/\rho^*)\mu^{(N)} + \text{int}((m_k^{(N)}-1)/\rho^*)+1, \quad k=\overline{1,\rho^*},$$

где $\text{int}(a/b)$ – функция целочисленного деления *a* на *b*.

В качестве иллюстративного примера рассмотрим сеть со следующими параметрами: размерность N входного вектора равна 2, параметр $\rho^* = 4$, $\mu^{(1)} = \mu^{(2)} = 3$, так что согласно (10) $x_{\max}^{(1)} = x_{\max}^{(2)} = 9$ и, следовательно, $X = \{\overline{1,9}; \overline{1,9}\}$ (2).

Найдем номера ячеек памяти CMAC согласно выражениям (6), (8), (9) и (12), которые соответствуют входному вектору $x^{(1)} = (6;8)^T$:

$$x[1] = \begin{vmatrix} 6 \\ 8 \end{vmatrix} \to M = \begin{vmatrix} 9 & 6 & 7 & 8 \\ 9 & 10 & 11 & 8 \end{vmatrix} \to m[1] = (9,9;6,10;7,11;8,8)^T.$$

Пересчитывая символические номера 9,9; 6,10; 7,11; 8,8 активных ячеек в соответствии с алгоритмом (12), получим

$$9,9 \to 8*3+2+1=27,$$
$$6,10 \to 5*3+2+1=18,$$
$$7,11 \to 6*3+2+1=23,$$
$$8,8 \to 7*3+1+1=23,$$

так что вектор номеров активных ячеек, соответствующий вектору $x^{(1)} = (6;8)^T$, равен

$$m[1] = (27,18,21,23)^T.$$

Не приводя промежуточных результатов, заметим, что вектор $x^{(2)} = (7;6)^T$ соответствует вектору номеров активных ячеек в символическом $m[2] = (9,9;10,6;7,7;8,8)^T$ и в десятичном $m[2] = (27,29,20,23)^T$ представлениях, и, следовательно, ячейки памяти с номерами 23 и 27 активируются как вектором, так и вектором $x[2]$.

Отметим здесь, что общее число ячеек памяти для рассматриваемой сети $M = 36$, тогда как возможное число точек запоминаемой функции равно $\overline{M} = 81$.

Объем памяти

Необходимое число ячеек памяти M нейронной сети CMAC следует из формулы (12), когда все числа $m_k^{(l)}, l = \overline{1,N}$, принимают свои максимально возможные значения, т. е. когда $m_k^{(l)} = M^{(l)} = \mu^{(l)}\rho^*,\ l = \overline{1,N}$.

При этом сомножители $\mathrm{int}((m_k^{(l)} - 1)/\rho^*), l = \overline{1,N}$, входящие в формулу (12), оказываются равными

$$\mathrm{int}((m_k^{(l)} - 1)/\rho^*) = \mu^{(l)} - 1,\ l = \overline{1,N}, \tag{13}$$

а величина M задается выражением

$$M = (\mu^{(1)}\rho^* - 1)\prod_{l=2}^{N}\mu^{(l)} + (\mu^{(2)} - 1)\prod_{l=3}^{N}\mu^{(l)} + \ldots \tag{14}$$

$$\ldots + (\mu^{(N-1)} - 1)\mu^{(N)} + (\mu^{(N)} - 1) + 1 = \rho^*\prod_{l=1}^{N}\mu^{(l)}.$$

Соотношения (10) и (14) позволяют выразить объем памяти CMAC в функции $x_{\max}^{(l)}$, $l = \overline{1,N}$, где $x_{\max}^{(l)}$ — наибольшее значение, которое принимает аргумент x_l:

$$M = \rho^* \prod_{l=1}^{N} \mu^{(l)} = \rho^* \prod_{l=1}^{N} \frac{(\mu^{(l)}\rho^* - \rho^* + 1) + \rho^* - 1}{\rho^*} = \tag{15}$$

$$= \rho^* \prod_{l=1}^{N} \frac{x_{\max}^{(l)} + \rho^* - 1}{\rho^*} = \rho^* \prod_{l=1}^{N} \frac{M^{(l)}}{\rho^*},$$

где $M^{(l)}$ определено соотношением (11).

Если бы функция многих переменных *y(x)* хранилась в памяти обычным способом, когда значение функции записывается в одну ячейку памяти, то для этого, очевидно, понадобилось бы

$$\overline{M} = \prod_{l=1}^{N} x_{\max}^{(l)} \tag{16}$$

ячеек памяти, что следует также из формулы (15) при $\rho^* = 1$, когда в CMAC каждый входной сигнал возбуждает только одну ячейку памяти. Из сравнения выражений (15) и (16) следует, что CMAC уменьшает объем памяти по сравнению с обычным способом хранения приблизительно в $(\rho^*)^{(N-1)}$ раз, так что эффективность его с точки зрения требуемого объема памяти растет с увеличением размерности входного вектора.

Пользуясь выражением (15), можно исследовать влияние параметра ρ^* на объем памяти CMAC при фиксированных характеристиках входного вектора *x*. Наиболее удобно сделать это, когда функция *y(x)* задана на гиперкубе, т. е. когда все значения $x_{\max}^{(l)}$, $l = \overline{1, N}$, одинаковы и равны

$$x_{\max}^{\circ} = 2^D + 1. \tag{17}$$

Полагая

$$\rho^*(r) = 2^r, r = \overline{0, D}, \tag{18}$$

найдем $\mu^{\circ} = \mu^{(l)}(10)$ и $M^{\circ} = M^{(l)}(11)$:

$$\mu^{\circ} = 2^{D-r} + 1; M^{\circ} = \mu^{\circ} \cdot \rho^*(r) = 2^D + 2^r. \tag{19}$$

Объем памяти CMAC согласно (15) в этом случае равен

$$M = \frac{(2^D + 2^r)^N}{2^{r(N-1)}}. \tag{20}$$

Нетрудно показать, что объем памяти *M* (20) с ростом параметра ρ^* убывает и достигает минимально возможного значения

$$M_{\min} = \frac{(2^D + 2^D)^N}{2^{D(N-1)}} = 2^N \cdot 2^D. \tag{21}$$

при $\rho^*(r) = 2^D$. Таким образом, коэффициент выигрыша объема памяти η, определяемый как отношение объема памяти $\overline{M}$ (16) при обычном способе хранения информации к объему памяти CMAC $M_{\min}$, принимает наибольшее значение при $\rho^*(r) = 2^D$ и оказывается равным

$$\eta = \frac{\overline{M}}{M_{\min}} = \frac{(2^D + 1)^N}{2^N \cdot 2^D}.$$

При $D \geq 5$ выражение для коэффициента η с большой точностью можно принять равным $\eta = 2^{DN-D-N}$. В частности, если хранить изображение, заданное на двумерной сетке $(2^{10} + 1)(2^{10}$

+ 1) обычным способом, то понадобится один мегабайт памяти, тогда как если это изображение запомнить в CMAC при максимально возможном значении коэффициента $\rho^*(r) = 2^{10}$, то можно ограничиться только четырьмя килобайтами памяти.

Выход нейронной сети CMAC

Значение запомненной в CMAC функции $\hat{y}(x)$ равно сумме содержимого активных ячеек памяти, Для формализации понятия «выход сети» введем следующие переменные:

$w[n]$ – *M*-мерный вектор памяти сети, вычисленный на *n*-м шаге обучения, каждая *j*-я компонента которого $w_j[n],\ j = \overline{1, M}$, соответствует содержимому *j*-й ячейки памяти CMAC;

$a(x)$ – *M*-мерный ассоциативный вектор, однозначно связанный с вектором аргументов *x* посредством вектора *m* (9) номеров активных ячеек памяти по следующему правилу: элементы вектора $a_j,\ j = \overline{1, M}$, номера которых совпадают с номерами активных ячеек памяти, равны единице, все остальные элементы вектора *a* равны нулю.

Следовательно, в векторе *a* всегда ρ^* элементов равны единице. Тогда в соответствии с правилом функционирования нейронной сети CMAC ее выход $\hat{y}(x, n)$ при заданном входном векторе *x* равен

$$\hat{y}(x, n) = a^T(x) w[n], \tag{22}$$

т. е. выход равен сумме содержимого активных ячеек памяти сети. Такое представление выхода сети было введено в работе Милитцера и Паркса [13]; оно позволило понять природу этой нейронной сети, исследовать процесс обучения в ней и найти пути повышения ее эффективности.

Обучение нейронной сети CMAC

Процесс формирования *M*-мерного вектора памяти *w*, направленный на запоминание функции *y(x)* *N* переменных $x = (x_1, x_2, ..., x_N)^T$, заданных на целочисленной сетке $X = \{x_1 = \overline{1, x_{\max}^{(1)}}; x_2 = \overline{1, x_{\max}^{(2)}}; ...; x_N = \overline{1, x_{\max}^{(N)}}\}$ называется *обучением*. Алгоритм обучения нейронной сети CMAC был предложен Альбусом в работе [1]. Этот алгоритм, имеющий рекуррентную природу, функционирует следующим образом.

Пусть после $(n-1)$-го измерения значений запоминаемой функции *y(x)* и соответствующих значений вектора аргументов $x(x[i], y[i], i = \overline{1, n-1})$ был вычислен вектор памяти $w[n-1]$. Тогда на следующем *n*-м шаге после измерения значения функции $y[n]$ при известном значении аргумента $x[n]$ сначала с помощью «алгоритма нелинейного преобразования аргументов» вычисляются номера активных ячеек памяти, т. е. вычисляется соответствующий вектору $x[n]$ вектор $m[k]$ (9, 12), далее вычисляется предсказываемое нейронной сетью значение функции $\hat{y}[k]$, равное сумме содержимого активных ячеек памяти. Вычисляются ошибка предсказания $\varepsilon[n] = y[n] - \hat{y}[n]$ и величина коррекции $\Delta w[k] = \varepsilon[n] / \rho^*$, которая прибавляется к содержимому активных ячеек памяти. Неактивные ячейки коррекции не подвергаются.

Если воспользоваться ассоциативным вектором *a,* однозначно связанным с вектором *m* (9) номеров активных ячеек памяти по следующему правилу: элементы вектора $a_j,\ j = \overline{1, M}$, номера которых совпадают с номерами активных ячеек памяти, равны единице, все остальные элементы вектора *a* равны нулю, то алгоритм обучения CMAC можно записать в следующей форме:

$$w_i[n] = w_i[n-1] + (y[n] - \sum_j w_j[n-1]) / \rho^*; \quad (23)$$

$$\forall i\, a_i[n] = 1; \forall j\, a_j[n] = 1;$$

$$w_i[n] = w_i[n-1],\ \forall i\, a_i[n] = 0. \quad (24)$$

Последовательность $w[n]$ (23), (24) устроена так, что после коррекции вектора памяти $w[n-1]$ оценка функции $y[n]$, которая является суммой содержимого активных ячеек, совпадает с ее значением, т. е.

$$y[n] = \sum_j w_j[n],\ \forall j\, a_j[n] = 1 \quad (25)$$

Это утверждение следует непосредственно из суммирования правой и левой частей уравнения (23) по активным номерам ячеек памяти.

Альбус исследовал сходимость алгоритма только экспериментально. Теоретическое обоснование сходимости последовательности (23), (24) впервые было дано в работе Милитцера и Паркса [13, 30], которые показали, что алгоритм обучения CMAC есть итеративный процесс решения системы линейных уравнений относительно неизвестных значений компонент вектора памяти *w*. Вопросы сходимости процедуры (23), (24) обсуждались также в работе [31]. В работе [13] было показано, что в зависимости от вида запоминаемой функции и параметров CMAC последовательность $w[n]$ (23), (24) сходится либо в точку, либо в некоторую область, которую авторы назвали *зоной притяжения*. Основой доказательства сходимости последовательности $w[n]$ послужило представление алгоритма (23), (24) в эквивалентной векторной форме:

$$w[n] = w[n-1] + \frac{y[n] - a^T(x[n])w[n-1]}{a^T(x[n])a(x[n])} a(x[n]), \quad (26)$$

$$w[0] = 0,\ n = 1, 2, \ldots .$$

В работе [14] было отмечено, что предложенный Альбусом алгоритм обучения в форме (26) является известным алгоритмом Качмажа [16, 17] для итеративного решения системы линейных уравнений вида

$$a^T(x[n])w = y[n],\ n = 1, 2, \ldots, \quad (27)$$

и поэтому в дальнейшем алгоритм (23), (24) обучения нейронной сети CMAC будем называть *алгоритмом Альбуса—Качмажа*. Представление алгоритма обучения в форме (26) позволило в работе [14] предложить и исследовать алгоритм обучения, обладающий повышенной скоростью сходимости, который будет описан во второй части статьи. С помощью такого представления также удалось повысить точность восстановления запоминаемых функций [32].

Система уравнений (27) обладает рядом особенностей: в каждом уравнении системы ровно ρ^* коэффициентов равны единице, все остальные коэффициенты равны нулю; как правило, размерность *M* вектора *w* существенно больше числа ненулевых элементов вектора, число которых равно ρ^*, так что матрица системы (27) является сильно разреженной матрицей; поскольку максимально возможное число линейно независимых уравнений системы (27) равно числу неизвестных, равному размерности *M* вектора *w*, а максимальное число уравнений равно числу точек $\overline{M} = \prod_{l=1}^{N} x_{\max}^{(l)}$ целочисленной сетки

$$X = \{x_1 = \overline{1, x_{\max}^{(1)}}; x_2 = \overline{1, x_{\max}^{(2)}}; \ldots; x_N = \overline{1, x_{\max}^{(N)}}\}$$

и, как правило, $\overline{M} \gg M$ (напомним, что $M = \prod_{l=1}^{N} (x_{\max}^{(l)} + \rho^* + 1)/(\rho^*)^{N-1}$ (15)), то эта система уравнений обычно несовместна; система уравнений (27) всегда имеет единственное решение в случае, когда параметр $\rho^* = 1$; при этом матрица системы равна единичной матрице или легко преобразуется к ней путем перенумерования последовательности $y[n]$. Этот случай соответствует обычному способу хранения информации, когда в каждой ячейке памяти хранится значение функции, а номер ячейки памяти однозначно связан с аргументом функции.

Несовместность системы уравнений (27) приводит к тому, что последовательность $w[n]$ (26) сходится в зону решения («зону притяжения» [13]) и совершает там движения в соответствии с алгоритмом обучения сети (26) и законом формирования входного вектора $x[n]$. За решение принимается одно из значений вектора $w[n]$, которое принадлежит зоне притяжения.

Последовательность выполнения работы

1. Получить у преподавателя функцию двух переменных
2. Используя файл cmac.mcd обучить нейронную сеть в MathCAD воспроизводить заданную функцию двух переменных.

Содержание отчета

1. Название лабораторной работы, цель и программа работы.
2. Коды Mathcad, описывающие структуру сети CMAC и ее обучение с пояснениями и промежуточными результатами (для массивов большой размерности все значения изображать необязательно).
3. Графики оригинальной функции в заданной области определения и функции воспроизводимой сетью CMAC.
4. График изменения абсолютной и относительной ошибки обучения сети. Итоговая ошибка обучения CMAC. 5. Выводы об информационной емкости, структуре, объеме памяти по сравнению с классическими методами табличной реализации заданной функции, скоростью обучения, способностью обобщения результатов обучения и других свойств CMAC.

Контрольные вопросы к защите лабораторной работы

1. Пояснить процесс формирования виртуальных и физических адресов сети CMAC.
2. На чем основывается способность нейронной сети обучаться с использованием предыдущего опыта?
3. За счет чего сокращается объем памяти сети CMAC по сравнению с классическим способом хранения функций табличной арифметики?
4. Показать зависимость информационной емкости сети от ее параметров.

Лабораторная работа № 2 «Многослойные нейронные сети»

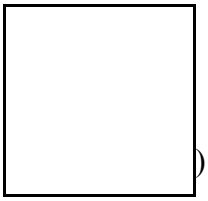
Цель работы

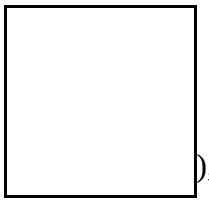
Изучить метод обучения обратного распространения ошибки многослойных нейронных сетей при использовании сети для различных прикладных задач

Программа работы

1. Исследовать возможности многослойных нейронных сетей при решении задач аппроксимации функции многих переменных.
2. Исследовать возможности многослойных нейронных сетей при решении задач прогноза значений временного ряда
3. Исследовать возможности многослойных нейронных сетей при решении задач классификация векторных данных
4. Составить и защитить отчет по результатам исследований.

Краткие сведения из теории

Архитектура многослойной нейронной сети представлена на рисунке 1. Все нейроны сети объединены в группы, называемые слоями. За каждый такт дискретного времени сигнал передается от слоя к слою, пока не достигнет выходного (последнего) слоя, на котором фиксируется реакция сети. Входной (нулевой) слой не выполняет какой-либо вычислительной функции и служит исключительно для распределения входного векторного сигнала $n = (n_1, n_2, \ldots,$  $)$ на нейроны первого слоя. В многослойной нейронной сети сигнал распространяется только в прямом направлении: обратные связи и связи между нейронами одного слоя отсутствуют. В связи с этим представленная на рис. 1 сеть называется многослойной нейронной сетью прямого распространения (feed-forward layered neural network).

Рабочие слои сети, расположенные между нулевым (распределительным) и выходным слоями, называются скрытыми (hidden layers). Состояние q-го слоя характеризуется вектором s^q, $q = \overline{0, K}$. В соответствии с обозначением входного векторного сигнала справедливо $s^0 = n$. Число нейронов в q-ом слое принимается равным N_q, так что $s^q = (s^q_1, s^q_2, \ldots,$  $)$, $q = \overline{0, K}$. Здесь s^q_i, $q = \overline{0, K}$, $i = \overline{1, N_q}$ означает состояние i- го нейрона q-го слоя.

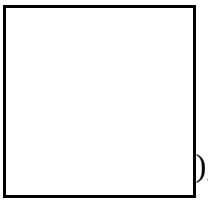
Функционирование каждого слоя сети определяется значениями матрицы W синаптических коэффициентов и вектора $-b$ смещений нейронов. В q-ом слое, $q = \overline{1, K}$, матрица W^q имеет размерность $[N_q, N_{q-1}]$, а вектор $b^q = (b^q_1, b^q_2, \ldots, b^q_{N_q})$. Расчет прохождения сигнала в сети прямого распространения выполняется по следующим формулам:

$$\begin{cases} s_i^q = f_q(h_i^q), \\ h_i^q = \sum\limits_{j=1}^{N_{q-1}} w_{ij}^q s_j^{q-1} - b_i^q, \end{cases} \qquad (2.1)$$

$$q = \overline{1, K},\ i = \overline{1, N_q},$$

где h^q_i — потенциал i-го нейрона q-го слоя; $f_q(h)$ — функция активации нейронов q-го слоя (предполагается, что все нейроны одного слоя имеют одинаковые функции активации). К функциям активации $f_q(h), q = \overline{1, K}$, предъявляется требование непрерывности и дифференцируемости. "Жесткие" пороговые функции активации не допускаются. Под w^q_{ij} в формуле (2.1) понимается элемент матрицы W^q синаптических коэффициентов q-го слоя.

Удобно преобразовать математическую модель (2.1) путем введения одного дополнительного нейрона в каждый слой МНС. Отметим эти нейроны нулевым индексом и положим их состояние тождественно равным 1, так что $s^q_0 = 1$ для $q = \overline{0, K}$. За счет введенных нейронов естественно расширяются вектора s^q, $q = \overline{0, K}$, состояний нейронов в слоях:

$\bar{s}^q = (s^q_0, s^q_1, s^q_2, \ldots,$  $)$, $q = \overline{0, K}$.

(2.2)

Расширим также матрицу синаптических коэффициентов в q-ом слое ($q = \overline{1, K}$), дополнив ее нулевым столбцом, содержащим смещения нейронов q-го слоя:

$$\overline{W}^q = \begin{pmatrix} -b_1^q & w_{11}^q & w_{12}^q & \dots w_{1N_{q-1}}^q \\ -b_2^q & w_{21}^q & w_{22}^q & \dots w_{2N_{q-1}}^q \\ . & . & . & . \\ -b_{N_q}^q & w_{N_q1}^q & w_{N_q2}^q & \dots w_{N_qN_{q-1}}^q \end{pmatrix} \tag{2.3}$$

Обозначим $\overline{w}_{ij}^q$, $i = \overline{1, N_q}$, $j = \overline{0, N_{q-1}}$, матрицы $\overline{W}^q$. В новых обозначениях математическая модель (2.1) преобразуется к следующему виду:

$$\begin{cases} s_0^q = 1, \quad s_i^q = f_q(h_i^q), \\ h_i^q = \sum_{j=0}^{N_{q-1}} w_{ij}^q s_j^{q-1} - b_i^q, \end{cases} \tag{2.4}$$

$$q = \overline{1, K},\ i = \overline{1, N_q}.$$

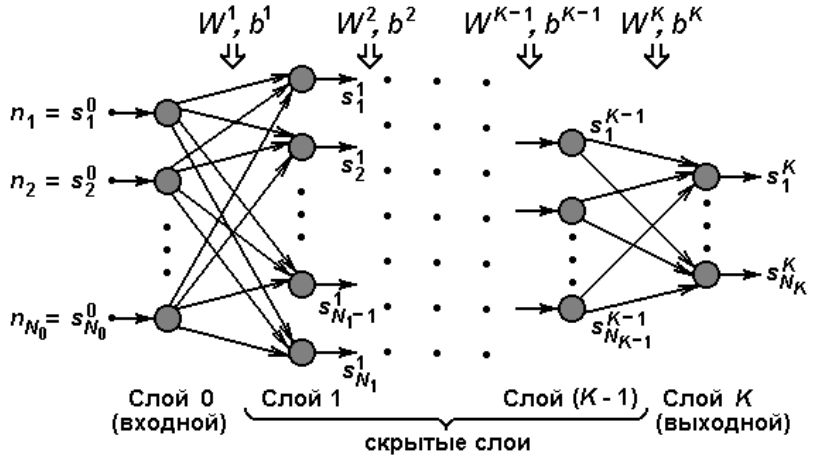


Рисунок 2.1 – Архитектура многослойной нейронной сети

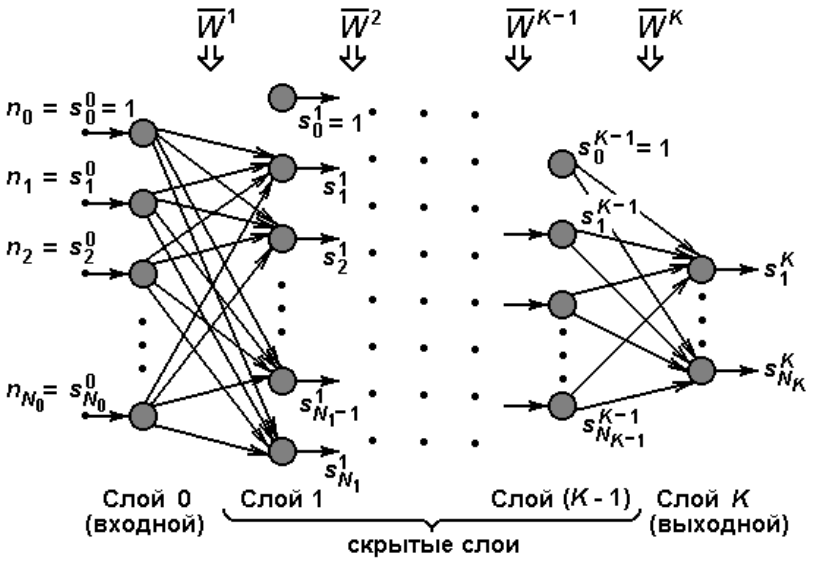


Рисунок 2.2 – Эквивалентное преобразование многослойной нейронной сети

Формулам (2.4) соответствует схема функционирования МНС, представленная на рисунке 2.2.

Применяя векторно-матричные операции над данными в слоях МНС, перейдем от уравнений (2.4) к следующей математической модели:

$$\begin{cases} s_0^q = 1,\ s_i^q = f_q(h_i^q), \\ h^l = \bar{s}^{q-1}(\overline{W}^q)^T, \\ q=\overline{1,K},\ i=\overline{1,N_q}, \end{cases} \tag{2.5}$$

где $h^q=(h^q{}_1, h^q{}_2, \ldots, h^q_{N_q})$ — вектор потенциалов нейронов q-го слоя, верхний индекс ^{T} — знак транспонирования вектора или матрицы.

Постановка задачи обучения МНС

Многослойная нейронная сеть осуществляет преобразование входного вектора n размерности $N_0 = M$ в выходной вектор s^K размерности $N_K = M$: $s^K = \varphi(n)$. Это функциональное преобразование для выбранной архитектуры сети (числа слоев, распределения нейронов по слоям и активационных характеристик нейронов) зависит от значений синаптических коэффициентов и смещений всех нейронов. Эти параметры сети собраны в матрицы $\overline{W}^q$, $q = \overline{1,K}$. Даже в достаточно простых практических приложениях число параметров сети может достигать нескольких десятков тысяч. Например, в задаче распознавания изображения, представленного матрицей значений яркости размером 10×10 (M = 100), с помощью сети, содержащей два рабочих слоя (скрытый слой N_1 = 50 и выходной слой $N_2 = N$ =2), число параметров превышает 5000. Если при решении практической задачи разумно подобрана архитектура нейронной сети, то настройкой параметров можно добиться близости фактического функционального преобразования, реализуемого нейронной сетью, к желаемому преобразованию в решаемой задаче. Процесс настройки параметров нейронной сети называется ее обучением.

Для обучения МНС используются данные обучающей выборки. Выборка состоит из образцов (примеров), содержащих желаемую реакцию $\sigma^p = (\sigma^p{}_1, \sigma^p{}_2, \ldots, \sigma^p{}_N)$ сети на входное воздействие $n^p = (n^p{}_1, n^p{}_2, \ldots, n^p{}_M)$:

$$\{ n^p, \sigma^p \},\ p = \overline{1,P}, \tag{2.6}$$

где P — объем обучающей выборки.

Обозначим фактическую реакцию рассматриваемой МНС на воздействие n^p через s^{pK} (верхний индекс K соответствует номеру выходного слоя). Тогда разность (σ^p- s^{pK}) характеризует ошибку преобразования входного сигнала сетью, а показатель

$$D = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{N}(\sigma_m^p - s_m^{pk})^2 \tag{2.7}$$

может служить критерием качества настройки параметров нейронной сети. Задача обучения нейронной сети сводится в такой постановке к достижению

$$\min_{\overline{W}^q,\ q=\overline{1,K}} D(\overline{W}^1, \overline{W}^2, \ldots, \overline{W}^K). \tag{2.8}$$

Поскольку при такой формализации задачи обучения предполагаются известными желаемые реакции МНС на входные сигналы из заданной выборки, что равносильно присутствию “учителя” в процессе обучения, сам процесс обучения называется “обучением с учителем”.

Градиентный поиск оптимальных параметров МНС

Определим закон, по которому эволюционируют значения параметров $\overline{w}_{ij}^q$, $q = \overline{1,K}$, $i = \overline{1,N_q}$, $j = \overline{0,N_{q-1}}$, в процессе минимизации показателя D, следующим равенством:

$$\frac{d}{dt}\overline{w}_{ij}^q = -\varepsilon\frac{\partial}{\partial\overline{w}_{ij}^q}D \tag{2.9}$$

где ε — параметр закона обучения, время t процедуры настройки полагается непрерывным. Если объединить все настраиваемые параметры МНС в один вектор, то закон (2.9) определяет, что в процессе эволюции этот вектор изменяется в каждый момент времени в направлении антиградиента критерия D. Несложно показать, что закон обучения (2.9) обеспечивает невозрастание показателя D в процессе эволюции:

$$\frac{d}{dt}D=\sum_{q=1}^{K}\sum_{i=1}^{N_q}\sum_{j=1}^{N_{q-1}}\frac{\partial}{\partial\overline{w}_{ij}^{q}}D\frac{d}{dt}\overline{w}_{ij}^{q}=-\varepsilon\sum_{q,i,j}\left(\frac{\partial}{\partial\overline{w}_{ij}^{q}}D\right)^2\le 0.$$

Таким образом, следуя закону (2.9) настройки параметров МНС, из любого начального значения параметров осуществляется спуск к экстремальной точке. В связи с тем, что показатель *D*, рассматриваемый как функция настраиваемых параметров, может содержать много локальных минимумов, нет гарантии спуска из произвольного начального значения вектора параметров в точку глобального минимума. В практических приложениях рекомендуется реализовать процедуру градиентного спуска неоднократно из различных начальных положений для выбора лучшего из полученных решений.

Численная реализация закона (2.9) требует перехода от непрерывного к дискретному времени $t = 0, 1, 2, \ldots$. В этом случае уравнения эволюции синаптических коэффициентов записываются в следующей форме:

$$\overline{w}_{ij}^{q}(t+1)=\overline{w}_{ij}^{q}(t)-\alpha\frac{\partial}{\partial\overline{w}_{ij}^{q}}D(t), \qquad (2.10)$$

где α — параметр закона обучения. От выбора параметра α зависит скорость обучения, форма переходного процесса по настраиваемым параметрам, а также сам факт сходимости процесса обучения. Чем меньше значение параметра α, тем менее различаются уравнения (2.9) и (2.10), реализующие поиск в непрерывном и дискретном времени. Следовательно, тем выше вероятность сходимости дискретной поисковой процедуры (гарантированное невозрастание критерия *D* в процессе эволюции доказано выше только для непрерывного времени). В то же время чрезмерное уменьшение параметра α затягивает переходные процессы, увеличивает необходимое время вычислений. Обычно в программных средствах предусматриваются различные возможности адаптивной подстройки значения α в процессе поиска экстремума, а выбор начального значения этого параметра определяется пользователем.

Для реализации закона обучения (2.10) необходимо определить алгоритм вычисления частных производных критерия *D* по искомым параметрам: $\frac{\partial}{\partial\overline{w}_{ij}^{q}}D$, $q=\overline{1,K}$, $i=\overline{1,N_q}$, $j=\overline{0,N_{q-1}}$. Этому вопросу посвящены следующие два подраздела.

Алгоритм обучения однослойной нейронной сети с непрерывной передаточной функцией

Рассмотрим частный случай $K = 1$, когда сеть содержит один рабочий слой. Опустим верхний индекс номера слоя в обозначениях настраиваемых синаптических коэффициентов $\overline{w}_{ij}^{q}$, $i=\overline{1,N}$ $(N = N_1)$, $j=\overline{1,M}$ $(M = N_0)$, а также векторов потенциалов h_i, $i=\overline{1,N}$, и состояний нейронов s_i, $i=\overline{1,N}$, выходного слоя. Опустим также индекс номера слоя в обозначении передаточной функции нейронов $f(h)$.

В принятых обозначениях критерий точности *D* на основании выражений (1.7) и (1.4) может быть представлен в следующем виде:

$$D=\frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{N}(\sigma_m^p-s_m^{pk})^2=\frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{N}(\sigma_m^p-f(h_m^p))^2 \qquad (2.11)$$

где $h_m^p\sum_{j=0}^{M}\overline{w}_{mj}n_j^p$, $(n_j=s_j^0,\ j=\overline{1,M})$; n_j^p — j-я составляющая входного вектора n в p-м примере обучающей выборки.

Вычислим с использованием выражения (2.11) производную показателя *D* по аргументу $\overline{w}_{ij}$, $i=\overline{1,N}$, $j=\overline{1,M}$:

$$\frac{\partial}{\partial\overline{w}_{ij}}D=-\sum_{p=1}^{P}(\sigma_i^p-f(h_i^p))\frac{\partial}{\partial\overline{w}_{ij}}f(h_i^p)=$$

$$= -\sum_{p=1}^{P} (\sigma_i^p - f(h_i^p))\, f'(h_i^p) n_j^p \qquad (2.12)$$

В выводе выражения (2.12) было учтено, что в сумме слагаемых по индексу m в выражении (2.11) только одно слагаемое, соответствующее значению $m = i$, зависит от $\overline{w}_{ij}$. В связи с этим окончательное выражение (2.12) содержит лишь сумму по примерам обучающей выборки. Введем обозначение:

$$\Delta_i^p = (\sigma_i^p - f(h_i^p))\, f'(h_i^p) = (\sigma_i^p - s_i^p)\, f'(h_i^p),$$
$$i = \overline{1, N}. \qquad (2.13)$$

Тогда выражение (1.12) преобразуется к следующей краткой форме:

$$\frac{\partial}{\partial \overline{w}_{ij}} D = -\sum_{p=1}^{P} \Delta_i^p n_j^p,\ i = \overline{1, N}, j = \overline{0, M}. \qquad (2.14)$$

Заметим, что в силу предположения о непрерывности и дифференцируемости активационной характеристики нейронов МНС производная $f'(h)$ существует (см. выражения (2.12) и (2.13)). В частности, для $f(h) = \text{th}(\beta h)$

$$f'(h) = \beta [1 - (f(h))^2],$$

а для логистической функции $f(h) = 1 / (1 + \exp(-\beta h))$

$$f'(h) = \beta f(h) (1 - f(h)).$$

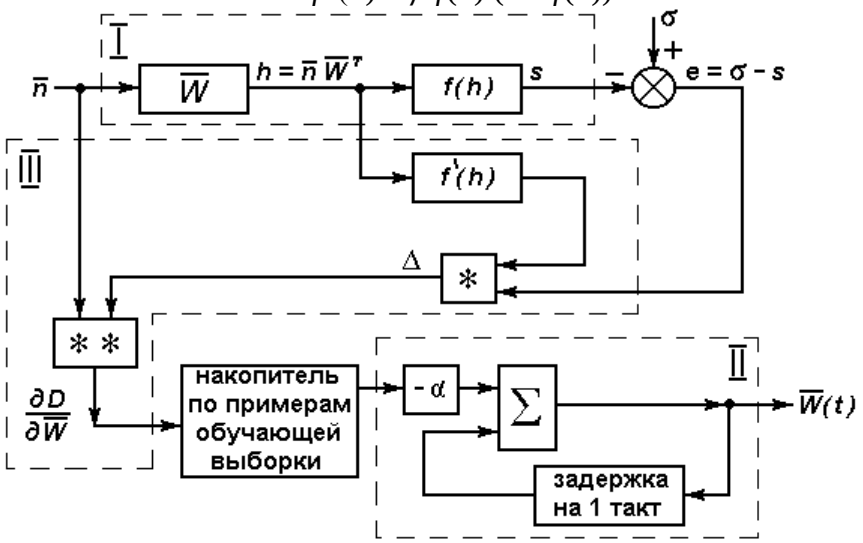


Рисунок 2.3 – Схема настройки параметров однослойной нейронной сети

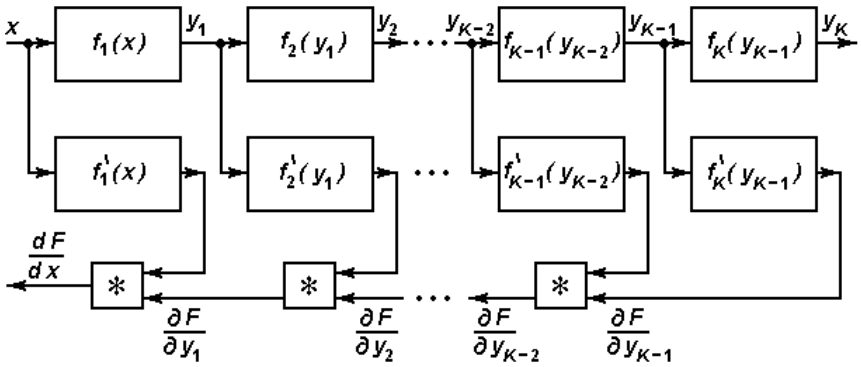


Рисунок 2.4 – Схема вычисления производной при последовательности соединений нелинейных преобразователей

На рисунке 2.3 представлена схема вычислений в соответствии с формулами (2.10), (2.14), определяющими алгоритм настройки параметров нейронной сети для рассматриваемого случая $K = 1$. В целях упрощения схемы применяются векторно-матричные обозначения для переменных. Блок I представляет собой сеть прямого распространения входного сигнала. Блок II отражает правило эволюции матрицы $\overline{W}$ в процессе градиентного спуска. Блок III служит для вычисления мат-

рицы частных производных $\frac{\partial}{\partial \overline{w}_{ij}} D,\ i = \overline{1, N}, j = \overline{0, M}$. В составе блока III звено, отмеченное знаком "*", функционирует согласно выражению (2.13), а звено, отмеченное знаком "**", вычисляет все парные произведения $\Delta_i\, n_j$ элементов входных векторов $\overline{n}$ и Δ ($i = \overline{1, N}, j = \overline{0, M}$).

Перед началом процесса обучения устанавливаются некоторые произвольные значения элементов матрицы $\overline{W}(0)$. Эта операция называется инициализацией нейронной сети. Обычно для этих целей используется датчик случайных чисел, распределенных равномерно на отрезке [-*c*; *c*], где *c* — параметр, устанавливаемый пользователем. Параметр *c* должен быть достаточно малым. В противном случае нейрон может оказаться в зоне насыщения сигмоидальной характеристики, когда он нечувствителен к любым малым изменениям синаптических коэффициентов. Эволюция соответствующих коэффициентов прекращается, хотя их значения далеки от оптимальных. Этот эффект называется "параличом" сети.

После установки значений $\overline{W}(0)$ блоки I и III функционируют *P* раз в соответствии с объемом обучающей выборки, что позволяет вычислить и накопить сумму (2.14), определяющую значение $\frac{\partial}{\partial \overline{w}_{ij}} D$ во всех обучающих примерах. Этот цикл обычно называют эпохой. Далее в блоке II реализуется подстройка элементов матрицы $\overline{W}$, в результате чего формируется $\overline{W}(1)$:

$$\overline{w}_{ij}(1) = \overline{w}_{ij}(0) - \alpha \frac{\partial D(0)}{\partial \overline{w}_{ij}}.$$

Это значение устанавливается в блоке I, после чего реализуется следующая эпоха и производится следующий такт подстройки коэффициентов в блоке III. Процедура повторяется до тех пор, пока не будет достигнут заданный уровень ошибки *D*, которая вычисляется в соответствии с выражением (2.11) в процессе обучения сети. Если требуемый уровень по ошибке *D* не достигается, то это может быть обусловлено несколькими причинами. Возможно, в рамках выбранной архитектуры сети принципиально невозможно достигнуть заданной точности. Другая причина — "паралич" сети. Возможно также, что текущее значение матрицы $\overline{W}$ соответствует положению точки в области обширного "плоскогорья" на поверхности функции $D(\overline{W})$ (эта поверхность обычно называется адаптивным рельефом), когда продвижение точки сильно замедлено. В этом случае можно увеличить значение параметра α, чтобы ускорить продвижение к "обрыву" на адаптивном рельефе. В программных пакетах, реализующих общую идею обучения, обычно применяют различные модификации градиентного спуска, ускоряющие процесс поиска минимума показателя *D*.

Метод обратного распространения ошибки для обучения МНС

Метод обратного распространения ошибки (error backpropagation), который является одним из самых распространенным в практических приложениях нейронных сетей, был сформулирован независимо друг от друга несколькими русскими и зарубежными учеными в 80-е годы.

Для иллюстрации излагаемого далее принципа рассмотрим следующий пример [8]. Пусть нелинейный преобразователь $F(x)$ представляет собой последовательное соединение нелинейных элементов $f_1, f_2, \ldots, f_K$ (см. рис. 4). В соответствии со схемой преобразователя можно записать $F(x)= f_K(f_{K-1}(\ldots f_1(x)))$.

Поставим задачу вычисления производной $\frac{d}{dx} F$. Согласно правилу дифференцирования сложной функции

$$\frac{d}{dx} F(x) = f'_K(y_{K-1})\Big|_{y_{K-1}(x)}\ f'_{K-1}(y_{K-2})\Big|_{y_{K-2}(x)} \ldots\ f'_1(x)$$

На рисунке 2.4 показано, что этот результат формируется на выходе дополнительной цепочки, в которой последовательно соединены блоки перемножения. В этой цепочке реализуется обратное движение сигнала с использованием результата прямого распространения входного воздействия x.

Этот пример наводит на мысль о возможности вычисления частных производных, необходимых для обучения МНС по закону (2.10), с использованием известных правил дифференцирова-

ния сложной функции и реализации обратного распространения сигнала ошибки (роль F играет показатель D точности обучения сети).

В качестве первого шага рассмотрим выражение для частных производных $\frac{\partial}{\partial \overline{w}_{ij}^{K}} D$ функционала D по настраиваемым параметрам последнего слоя:

$$\frac{\partial}{\partial \overline{w}_{ij}^{K}} D = \frac{\partial}{\partial \overline{w}_{ij}^{K}} \left(\frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{N=N_k} (\sigma_m^p - s_m^{pK})^2 \right) =$$

$$= -\sum_{p=1}^{P} \sum_{m=1}^{N=N_k} (\sigma_m^p - s_m^{pK}) \frac{\partial}{\partial \overline{w}_{ij}^{K}} s_m^{pK} \qquad (2.15)$$

$$i = \overline{1, N_K}, \; j = \overline{0, N_{K-1}}.$$

Вычисление частной производной $\frac{\partial}{\partial \overline{w}_{ij}^{K}} s_m^{pK}$ опирается на уравнение функционирования K-го слоя МНС:

$$s^{pK}{}_m = f_K(s_m^{pK} = f_K \sum_{r=0}^{N_{K-1}} \overline{w}_{mr}^{K} s_r^{p(K-1)}, \; m = \overline{1, N_k} \qquad (2.16)$$

Из уравнения (2.16) следует, что

$$s^{pK}{}_m = \frac{\partial}{\partial \overline{w}_{ij}^{K}} s_m^{pK} = \begin{cases} 0, & m \neq i \\ f'(h_i^{pK}) s_j^{p(K-1)}, & m = i' \end{cases} \qquad (2.17)$$

$$i = \overline{1, N_K}, \; j = \overline{0, N_{K-1}}.$$

После подстановки выражения (2.17) в (2.15) получим:

$$\frac{\partial}{\partial \overline{w}_{ij}^{K}} D = -\sum_{p=1}^{P} (\sigma_i^p - s_i^{pK}) \, f_K'(h_i^{pK}) s_j^{p(K-1)}$$

С использованием обозначения

$$\Delta_i^{pK} = (\sigma_i^{pK} - s_i^{pK}) \, f_K'(h_i^{pK}), \quad i = \overline{1, N_K}, \qquad (2.18)$$

последнее выражение для частной производной преобразуется к следующему виду:

$$\frac{\partial}{\partial \overline{w}_{ij}^{K}} D = -\sum_{p=1}^{P} \Delta_i^{pK} s_j^{p(K-1)}, \quad i = \overline{1, N_K}, \; j = \overline{0, N_{K-1}}. \qquad (2.19)$$

Перейдем к рассмотрению (K-1)-го слоя с настраиваемыми коэффициентами $\overline{w}_{ij}^{K-1}$, $i = \overline{1, N_{K-1}}$, $j = \overline{0, N_{K-2}}$:

$$\frac{\partial}{\partial \overline{w}_{ij}^{K}} D = -\sum_{p=1}^{P} \sum_{m=1}^{N=N_K} (\sigma_m^p - s_m^{pK}) \, f_K'(h_m^{pK}) \frac{\partial}{\partial s_i^{p(K-1)}} h_m^{pK} \frac{\partial}{\partial \overline{w}_{ij}^{K-1}} s_i^{p(K-1)} =$$

$$= -\sum_{p=1}^{P} \sum_{m=1}^{N=N_K} (\sigma_m^p - s_m^{pK}) \; f_K'(h_m^{pK}) \; \overline{w}_{mi}^{K} \; f_{K-1}'(h_i^{p(K-1)}) \; s_j^{p(K-2)} =$$

$$= -\sum_{p=1}^{P} \Delta_i^{p(K-1)} s_j^{p(K-2)} \qquad (2.20)$$

где использовано обозначение:

$$\Delta_i^{p(K-1)} = \sum_{m=1}^{N=N_K} (\sigma_m^p - s_m^{pK}) \, f_K'(h_m^{pK}) \overline{w}_{mi}^{K} \; f_{K-1}'(h_i^{p(K-1)}) =$$

$$= \left(\sum_{m=1}^{N=N_K} \Delta_m^{pK} \; \overline{w}_{mi}^{K} \right) f_{K-1}'(h_i^{p(K-1)}), \; i = \overline{1, N_{K-1}}. \qquad (2.21)$$

В последнем выражении была применена формула (2.18).

Для последующих (с конца) слоев (*K*-2), (*K*-3), … вычисления частных производных функционала *D* по элементам расширенной матрицы синаптических коэффициентов в слое выполняются аналогичным образом. В итоге таких вычислений получается следующая общая формула:

$$\frac{\partial}{\partial \overline{w}_{ij}^{q}} D = -\sum_{p=1}^{P} \Delta_i^{pq} s_j^{p(q-1)}, \; q = \overline{1, K}, \; i = \overline{1, N_q}, \; j = \overline{0, N_{q-1}}, \qquad (2.22)$$

где

$$\Delta_i^{pq} = \left(\sum_{m=1}^{N=N_{q+1}} \Delta_m^{p(q+1)} \; \overline{w}_{mi}^{q+1} \right) f_q'(h_i^{pq}), \quad q = \overline{1, K-1}, \qquad (2.23)$$

$$\Delta_i^{pK} = (\sigma_i^{p} - s_i^{pK}) \, f_K'(h_i^{pK}), \quad i = \overline{1, N_K}.$$

Переменные Δ_i^{pq}, $q = \overline{1, K}$, $i = \overline{1, N_q}$, получили название двойственных по отношению к потенциалам нейронов h_i^{pq}, $q = \overline{1, K}$, $i = \overline{1, N_q}$, в сети прямого распространения входного сигнала.

На рисунке 2.5 представлена схема вычислений в соответствии с формулами (2.22), (2.23). Схема содержит цепь обратного распространения , которая возбуждается сигналом ошибки $e = \sigma - s^K$. В целях упрощения обозначений в схеме опущен индекс p примера обучающей выборки. Цепь обратного распространения формирует двойственные переменные Δ^q, $q = \overline{1, K}$, являющиеся векторами размерности N_q. Знаками “*” и ”**” отмечены звенья, которые осуществляют то же преобразование данных, что и в схеме на рисунке 2.3.

Для управления процессом настройки параметров МНС согласно системе уравнений (2.10) следует активизировать вычисления (см. рисунок 2.5) P раз по числу обучающих примеров и провести накопление результатов подобно тому, как это показано на рисунке 2.3 для простейшего случая однослойной сети.

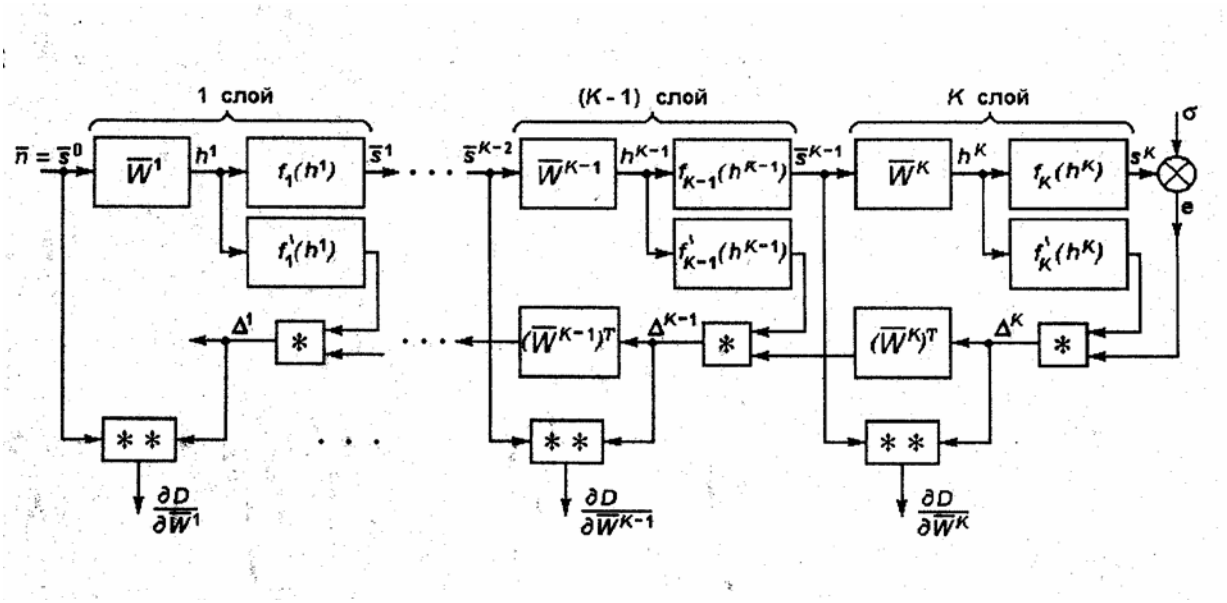


Рисунок 2.5 – Схема сетей прямого распространения и обратного распространения ошибки

Сопоставление схем прямого и обратного распространения в их полном (а не сжатом векторно-матричном) представлении показывает, что схема обратного распространения может быть построена по заданной прямой схеме путем применения к последней следующих правил:

1. Направление стрелок, указывающих прохождение сигнала, меняется на обратное.
2. Фрагменты схемы с активационными характеристиками нейронов $f_q(h_q^i)$, $q = \overline{1, K}$, $i = \overline{1, N_q}$ заменяются нелинейным преобразователем $f_q'(h_q^i)$ и блоком перемножения.
3. Матрицы синаптических коэффициентов $\overline{W}^q$, $q = \overline{1, K}$, транспонируются.
4. Сумматоры заменяются точками разветвления, а точки разветвления – сумматорами.

Сформулированные правила построения схемы формирования двойственных переменных справедливы и в том случае, когда нейронная сеть содержит прямые связи не только рядом расположенных слоев, но и более удаленных (связь "перепрыгивает" несколько ближайших слоев).

Практические исследования показывают, что обученная МНС обладает высокой робастностью: при установке значений синаптических коэффициентов, отличающихся от оптимальных (ошибки реализации), сеть продолжает выполнять свою функциональную задачу. Даже разрыв некоторых синаптических связей (технический отказ отдельных элементов вычислительной сети) может не приводить к потере работоспособности сети (отказоустойчивость). Следует заметить, что указанные свойства проявляются только в том случае, когда нейронная сеть обладает некоторой информационной "избыточностью" по отношению к решаемой задаче.

Принципиальной является способность МНС к обобщению, то есть способность формировать "разумную" реакцию на входные воздействия, которых не было в составе обучающей выборки. Именно благодаря этому свойству МНС успешно применяется для интерполяции функций многих переменных, экстраполяции временных рядов, классификации объектов по их признакам и в других практических приложениях.

Последовательность выполнения работы

1. Определение типа решаемой задачи. В соответствии с вариантом задания (таблица 2.1) сделать заключение, что нейросеть будет применяться для решения задачи: аппроксимации функций, прогноза значений временного ряда или классификации.

Таблица 2.1 – Варианты заданий

№ варианта	Задача	№ варианта	Задача
1	Аппроксимация var1	12	Классификация var4
2	Прогноз var1	13	Аппроксимация var5
3	Классификация var1	14	Прогноз var5
4	Аппроксимация var2	15	Классификация var5
5	Прогноз var2	16	Аппроксимация var6
6	Классификация var2	17	Классификация var6
7	Аппроксимация var3	18	Классификация var7
8	Прогноз var3	19	Классификация var8
9	Классификация var3	20	Классификация var9
10	Аппроксимация var4	21	Классификация var10
11	Прогноз var4	22	Классификация var11

2. Визуализация данных в пакете MATLAB. Загрузить MATLAB (6.x) и указать путь к М-файлу лабораторной работы № 6, например, используя команду addpath (пример: addpath c:\neuro\lab7). Набрать в командной строке имя М-файла mnnapps. Указать тип решаемой задачи (аппроксимация функций, прогноз временных рядов или классификация). В трёх верхних полях указать пути и имена файлов с обучающей, тестовой и контрольной выборками последовательно. Визуализировать указанные выборки, используя кнопку View. В выпадающих списках, соответствующих каждой координатной оси, укажите название столбца, содержащего данные выборки (пример: ось X – in1, ось Y – out1, select – in2 from 0 to 2 – построить график функции out1(in1) при условии, что 0<in2<2). Зарисовать графики функции в различных проекциях (для задач аппроксимации), временной ряд (для задач прогноза) или геометрическое расположение классов (для задач классификации).

3. Анализ данных обучающей выборки.

Ответить на вопросы:

в задаче аппроксимации:

1) имеются ли выбросные значения функции (сильно отличающиеся от соседних значений)?
2) наблюдается ли линейная зависимость значений функции от какого-либо аргумента?
3) наблюдается ли колебательная зависимость значений функции от какого-либо аргумента?

в задаче классификации:

1) есть ли области перекрытия классов?

2) наблюдаются ли вложенные классы?
3) сколько прямых нужно провести, для того чтобы представить границу между классами?

в задаче прогноза временных рядов:

1) имеются ли выбросные значения ряда (сильно отличающиеся от соседних значений)?
2) наблюдается ли линейный тренд?
3) наблюдается ли колебательная зависимость значений ряда от временных отсчетов?

Сделайте вывод о том, сколько слоев должна содержать сеть и сколько нейронов должно быть в каждом слое для оптимального решения задачи (выполнение указанной функции при точности 10% и максимального выполнения свойства генерализации).

4. Создание и обучение МНС. В пакете NNTool или из командной строки MatLab создайте новый проект. В структуру новой сети введите оцененные в п.3 параметры архитектуры. Задайте точность сети. Обучите нейросеть простым градиентным методом. Зафиксируйте значение ошибки на обучающей выборке. Сохраните результаты обработки обучающей выборки в текстовый файл.

5. Тестирование МНС. Зафиксируйте результаты обработки сетью тестовой выборки и сохраните их в текстовый файл.

6. Определение работы сети для контрольных примеров. Зафиксируйте результаты обработки сетью контрольной выборки. Сохраните результаты обработки контрольной выборки в текстовый файл.

7. Визуализация результатов обработки данных нейросетью.

Укажите в трёх нижних полях пути и имена сохранённых ранее текстовых файлов с результатами обработки сетью обучающей, тестовой и контрольной выборками последовательно. Визуализируйте указанные выборки, используя кнопку View. В выпадающих списках, соответствующих каждой координатной оси, укажите название столбца, содержащего данные выборки (см. п. 1).

8. Упрощение МНС. Сократите число синапсов МНС. Получите результаты обработки сетью обучающей, тестовой и контрольной выборок (см. пп. 5, 6). Визуализируйте полученные результаты (см. п. 7). Сделайте вывод о том, как повлияло упрощение на точность работы сети и ее способность к обобщению, используя результаты тестирования выборок и визуализацию результатов работы сети на рабочей выборке.

9. Выбор лучшей архитектуры для решения поставленной задачи. Из полученных в пп. 4 и 8 архитектур выберите наиболее удачную или предложите и обучите новую. Опишите, по каким критериям происходило рассмотрение лучшей архитектуры и что привело к созданию новой архитектуры.

Содержание отчета

1. Название лабораторной работы, цель и программа работы.
2. Графики с изображением обучающей, тестовой и контрольной выборок.
3. Выводы о структуре сети. Архитектура выбранной нейронной сети.
4. Коды MatLab формирования и обучения нейронной сети. Графики изменения ошибки обучения. Результаты тестирования нейронной сети. Результаты обработки контрольной выборки.
5. Коды MatLab формирования и обучения сокращенной нейронной сети. Графики изменения ошибки обучения. Результаты тестирования нейронной сети. Результаты обработки контрольной выборки.
6. Выводы по использованию архитектуры сети.

Контрольные вопросы к защите лабораторной работы

1. Объясните содержание задачи аппроксимации функции многих переменных на многослойной нейронной сети.
2. В чем состоит этап подготовки данных для обучения многослойной нейронной сети в задаче аппроксимации функции многих переменных?
3. Объясните содержание задачи прогноза временных рядов на многослойной нейронной сети.
4. В чем состоит этап подготовки данных для обучения многослойной нейронной сети в задаче прогноза временных рядов?
5. Объясните содержание задачи классификации данных на многослойной нейронной сети.
6. В чем состоит этап подготовки данных для обучения многослойной нейронной сети в задаче

классификации данных?
7. Какой критерий используется при решении задачи классификации данных с помощью многослойной нейронной сети?
8. Какие правила останова процесса обучения МНС применяются в практических приложениях?
9. В чем состоит функциональная задача дополнительного блока принятия решения на выходе МНС при использовании ее в качестве классификатора данных?
10. Почему чрезмерное увеличение числа нейронов скрытых слоев в МНС при решении задачи аппроксимации может ухудшить точностные показатели сети? Какое свойство нейронной сети имеется в виду?
11. Почему на задачи классификации данных и прогноза временного ряда распространяются те же рекомендации по выбору архитектуры МНС, что и в задаче аппроксимации функции многих переменных?
12. Какие Вы можете предложить нейросетевые решения задачи прогноза временного ряда одновременно на несколько последовательных временных тактов?

Лабораторная работа № 3 «Математические модели искусственных нейронных сетей Хэмминга»

Цель работы

Изучить математическую модель нейронной сети Хэмминга

Программа работы

1. Исследовать нейронную сеть Хэмминга с сетью MAXNET прямого распространения сигнала.
2. Исследовать нейронную сеть Хэмминга с рекуррентной сетью MAXNET.
3. Составить и защитить отчет по результатам исследований, в котором должны быть приведены архитектуры исследованных нейронных сетей, результаты моделирования нейронных сетей, выводы по обобщающей способности и скорости функционирования сетей Хэмминга.

Краткие сведения из теории

В практических приложениях часто встречаются задачи обработки данных, которые относятся к классу задач распознавания образов. К подобным задачам относятся:

• распознавание рукописных букв независимо от их масштаба, угла поворота и особенностей почерка;

• диагностика заболевания пациента по данным медицинских анализов;

• определение типа сопровождаемой цели по известным характеристикам отраженного от цели радиолокационного сигнала;

• оценка уровня экологического загрязнения водоема по характеристикам отраженного лазерного луча.

Во всех перечисленных примерах исследуемый объект характеризуется совокупностью признаков, образующих вектор $x=(x_1,x_2,\ldots,x_M)$. Объект может принадлежать одному из K известных классов. Так, при распознавании русских рукописных букв заданы 32 класса. На рисунке 3.1 представлена иллюстрация к задаче распознавания образов, в которой объект характеризуется двумерным вектором признаков $x=(x_1,x_2)$, заданным в прямоугольнике допустимых значений $x_1 \in [x_1`, x_1``]$, $x_2 \in [x_2`, x_2``]$. Пространство признаков разделено на три непересекающиеся области, каждая из которых объединяет объекты, принадлежащие одному классу. Эти области обозначены на рисунке Ω_1, Ω_2, Ω_3 . Сложность решения задачи распознавания образов в практических приложениях связана с тем, что границы между классами неизвестны, их уравнения не заданы и простая логическая проверка принадлежности текущего объекта x к одной из областей Ω_1, Ω_2 и Ω_3 невозможна.

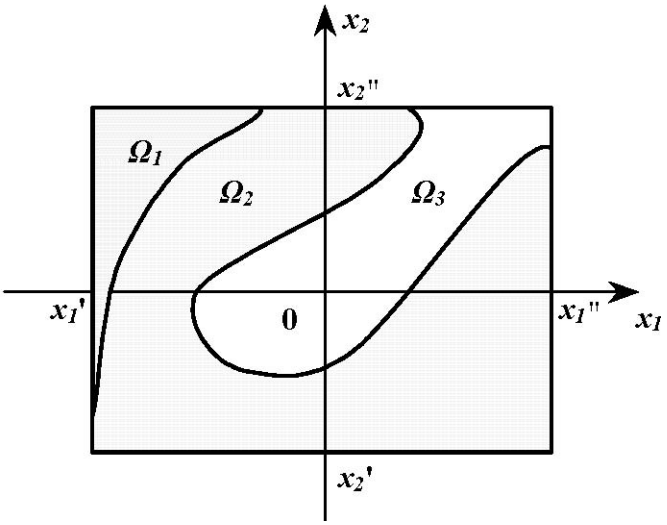


Рисунок 3.1 – Иллюстрация к задаче распознавания образов

Разные математические постановки задачи распознавания образов связаны, как правило, с разным объемом и качеством располагаемой информации о решаемой задаче. В классической постановке задачи распознавания образов используются статистические данные относительно векторов признаков объектов в разных классах и вводится статистический критерий, позволяющий с определенной надежностью отнести объект к одному из классов. Такой подход требует предварительной статистической обработки располагаемых данных и опирается на некоторые допущения о типах распределений вероятностей, которые могут быть неточными.

В последние годы широко распространился нейросетевой подход к решению задачи распознавания образов, который опирается исключительно на экспериментальные примеры, образующие обучающую выборку. С помощью обучающей выборки осуществляется такая настройка параметров нейронной сети (синаптических коэффициентов и смещений нейронов), чтобы при предъявлении сети вектора признаков объекта x она в качестве реакции формировала указание на класс принадлежности объекта. Например, можно потребовать при настройке параметров сети, чтобы при наличии K классов она возбуждала тот из K выходных нейронов, номер которого совпадает с номером класса принадлежности объекта.

Задача распознавания образов может решаться на нейронных сетях разной архитектуры. В настоящей лабораторной работе используется простейшая модель нейронной сети, в которой каждый из нейронов выполняет четко определенную функциональную задачу и потому легко контролируется при проведении исследований.

Целью работы является изучение математической модели технического нейрона и нейронной сети, ознакомление с особенностями функционирования нейронных сетей прямого распространения и рекуррентных нейронных сетей, исследование работоспособности нейронной сети при изменении ее внутренних характеристик.

Математическая постановка задачи

Для начального изучения модели технического нейрона и структуры нейронной сети рассматривается сеть Хемминга (Hamming). Эта сеть предназначена для распознавания класса принадлежности объекта, заданного вектором x биполярных признаков (возможные значения признаков +1 и -1) размерности M. Если представить $M = n_1 \cdot n_2$, то строка биполярных признаков может быть преобразована к прямоугольной матрице с n_1 строками и n_2 столбцами. Тогда возможна графическая интерпретация объекта (рисунке 3.2), в которой клетки матрицы, соответствующие признакам со значением +1, представлены черным цветом, а -1 – белым цветом. Предполагается, что имеются K классов, каждый из которых характеризуется своим эталонным представителем – объектом $x^{(k)}$, $k = 1,2, \ldots, K$.

Сеть Хемминга принимает на M входов биполярные признаки объекта и после выполненной обработки данных активизирует один из K выходов, который указывает на класс принадлежности предъявленного на входе объекта.

Критерием отнесения объекта x к классу является квадрат расстояния между векторами x и

$x^{(k)}$, k = 1,2, ... , K:

$$R\left(x, x^{(k)}\right) = \sum_{j=1}^{M}\left(x_j - x_j^{(k)}\right)^2, \tag{3.1}$$

где x_j – j – ый биполярный признак, j = 1, 2, …, M.

Сеть относит объект x к классу k^*, если

$$\min_{k=K} R\left(x, x^{(k)}\right) = R\left(x, x^{(k^*)}\right). \tag{3.2}$$

Раскроем выражение (3.1) для $R(x, x^{(k)})$, учитывая, что признаки являются биполярными:

$$R\left(x, x^{(k)}\right) = \sum_{j=1}^{M}\left(x_j^2 + \left(x_j^{(k)}\right)^2 - 2x_j x_j^{(k)}\right) =$$

$$= 2\sum_{j=1}^{M}\left(1 - x_j x_j^{(k)}\right) = 2\left(M - \sum_{j=1}^{M} x_j x_j^{(k)}\right). \tag{3.3}$$

Перейдем от критерия минимизации $R(x, x^{(k)})$ к критерию максимизации выражения $I_k = -\frac{1}{2}R\left(x, x^{(k)}\right) + 2M$, для которого достигается максимум при $k = k^*$:

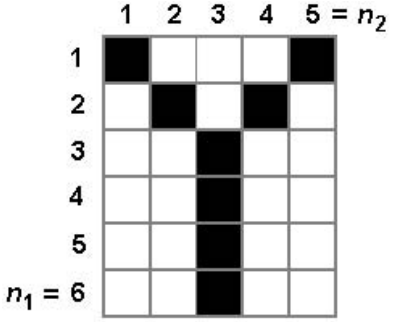


Рисунок 3.2 – Графическая интерпретация вектора признаков объекта

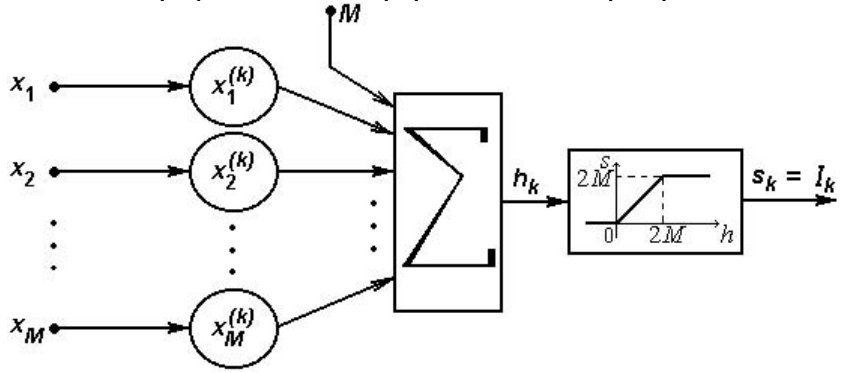


Рисунок 3.3 – Схема k-го нейрона в рабочем слое Хемминга, k = 1, 2, …, K

$$\arg\max_{k=\overline{1,K}} I_k = \arg\max_{k=\overline{1,K}}\left(\sum_{j=1}^{M} x_j x_j^{(k)} + M\right) = k^*. \tag{3.4}$$

Элементарный анализ показывает, что значение $R \in [0, 4M]$, поэтому $I_k \in [0, 2M]$.

Для вычисления значений I_k, $k = \overline{1,K}$, применяется рабочий слой, состоящий из K нейро-

нов со следующими характеристиками:

$$w_{kj} = x_j^{(k)},\ b_k = -M,\ k = \overline{1,K}. \qquad (3.5)$$

Потенциал h_k k-го нейрона определяется с учетом (3.5) выражением:

$$h_k = \sum_{j=1}^{M} w_{kj} x_j - b_k = \sum_{j=1}^{M} x_j x_j^{(k)} + M = I_k. \qquad (3.6)$$

Передаточная функция (активационная характеристика) нейронов полагается линейной с ограничениями, связанными с областью практических возможных значений потенциала $h_k = I_k$ (рисунок 3.3).

Для вычисления аргумента $k = k^*$, для которого достигается $\max_k I_k$, рабочий слой нейронов сети Хемминга дополняется группой нейронов, решающих задачу максимизации. Эта группа нейронов образует нейронную сеть, которая далее называется MAXNET. На рисунке 3.4 показана структура сети Хемминга.

В лабораторной работе рассматривается два варианта построения сети MAXNET:
• сеть MAXNET прямого распространения (feed-forward MAXNET),
• рекуррентная сеть MAXNET (recurrent MAXNET).

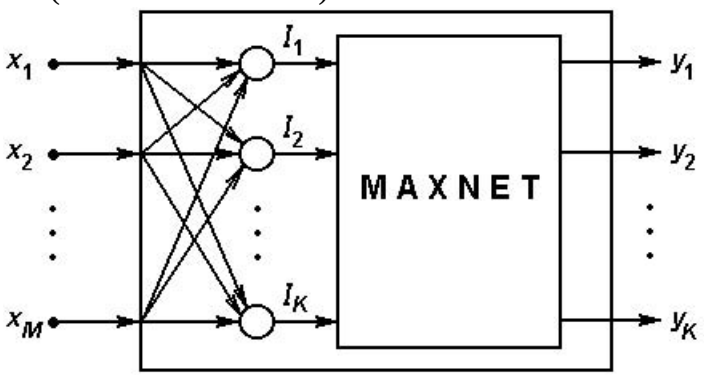


Рисунок 3.4 – Структурная схема сети Хемминга

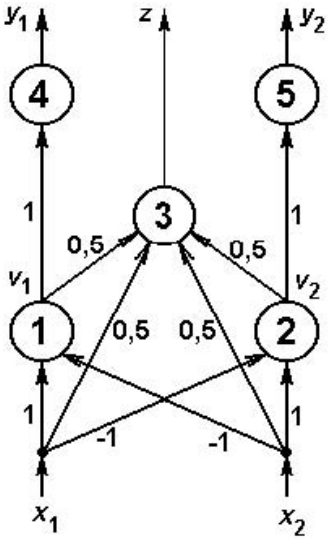


Рисунок 3.5 – Схема нейросетевого компаратора на 2 входа

Математическая модель сети feed-forward MAXNET

В основе сети MAXNET прямого распространения лежит нейросетевой компаратор на два входа (см. рисунок 3.5). Нейроны 1, 2 и 3 имеют нулевое смещение и линейную активационную характеристику с ограничениями (рисунок 3.6). Значение параметра c (уровня ограничения сверху) определяется условиями технической реализации. Предполагается, что практические значения

переменных x_1 и x_2 таковы, что значения $h > c$ невозможны. Таким образом, выходы нейронов 1 и 2 могут быть представлены выражениями:

$$v_1 = \begin{cases} x_1 - x_2, x_1 > x_2 \\ 0, x_1 < x_2 \end{cases},$$

$$v_2 = \begin{cases} 0, x_1 > x_2 \\ x_2 - x_1, x_1 < x_2 \end{cases}.$$

Заметим, что значения x_1 и x_2 предполагается несовпадающими. Нейрон 3 в любой из ситуаций $x_1 > x_2$ и $x_1 < x_2$ формирует выходе $z = \max(x_1, x_2)$.

Для указания того, по какому из входов пришел максимальный сигнал, служат нейроны 4 и 5. Они имеют пороговую активационную характеристику, представленную на рисунок 3.7.

Если требуется построить MAXNET на несколько входов, используется несколько компараторов, соединенных в иерархическую сеть, и выходные нейроны, играющие роль логических элементов.

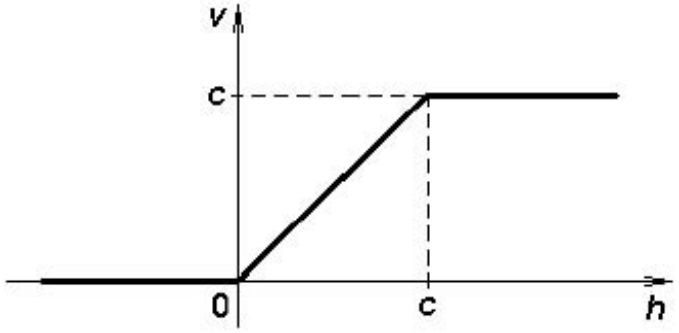


Рисунок 3.6 – Активационная характеристика нейронов 1 и 2

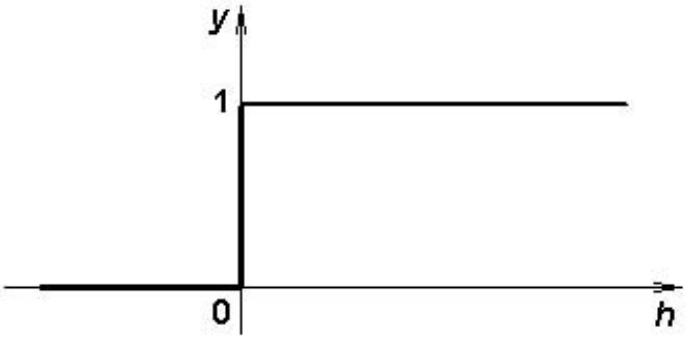


Рисунок 3.7 – Активационная характеристика нейронов 4 и 5

На рисунке 3.8 представлена архитектура MAXNET на 8 входов. Выходные нейроны, формирующие сигналы y_1, y_2, …, y_8, имеют единичные синаптические коэффициенты, нулевые смещения и пороговую активационную характеристику, представленную на рисунке 3.9. На рисунке 3.8 темными кружочками представлены нейроны, имеющие пороговые активационные характеристики и выполняющие логические функции.

Число слоев сети MAXNET прямого распространения растет пропорционально $\log_2 K$, где K – число входов. Соответственно растет и время поиска максимума, так как каждый слой обрабатывает данные за один временной такт.

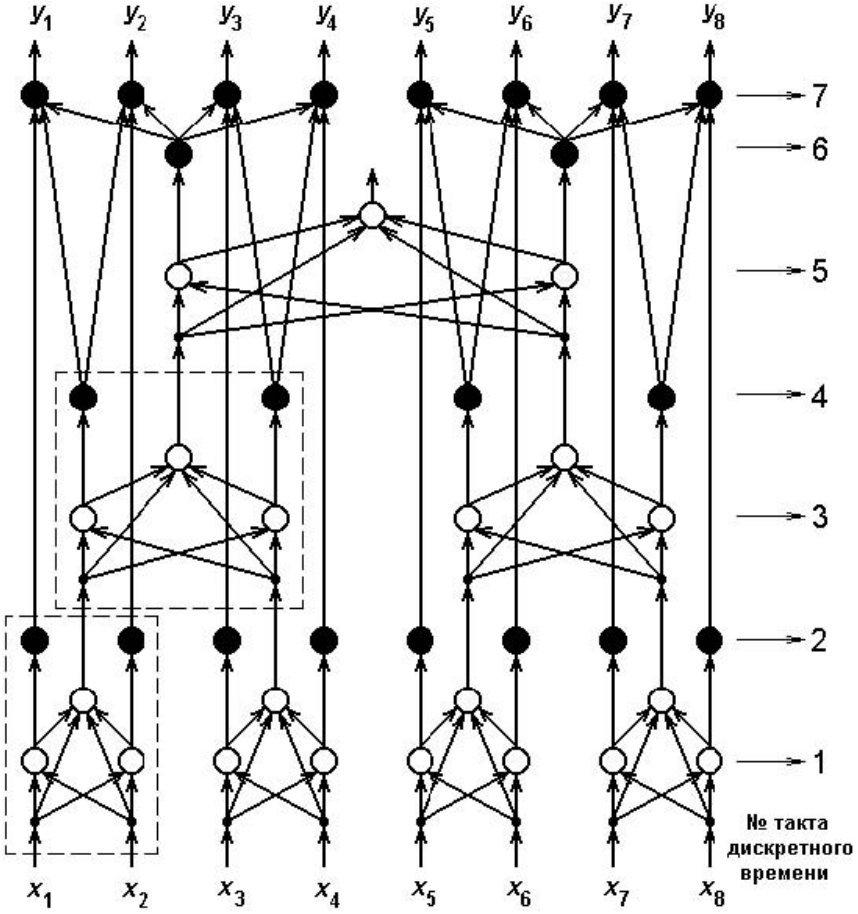


Рисунок 3.8. Схема сети MAXNET на восемь входов

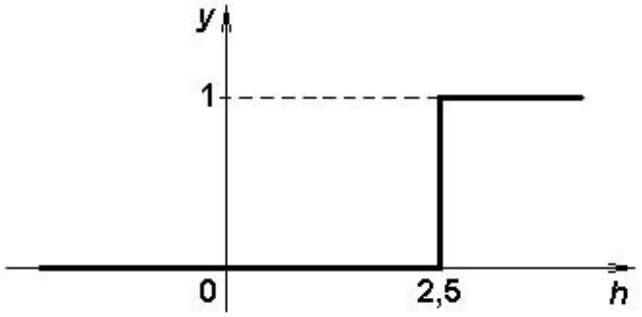


Рисунок 3.9. Пороговая активационная характеристика выходных нейронов сети MAXNET

Математическая модель рекуррентной сети MAXNET

Предполагается, что входные сигналы рекуррентной сети MAXNET положительны и не совпадают друг с другом. Обозначим их $x_1, x_2, \ldots, x_K$. Рассматриваемая нейронная сеть содержит K нейронов, каждый из которых имеет линейную активационную характеристику с ограничениями (рисунок 3.6). Сеть содержит полные связи и функционирует во времени в синхронном режиме, когда на каждом временном такте срабатывают все нейроны. Рассмотрим математическую модель i-го нейрона. Его смещение равно нулю, а синаптические коэффициенты определяются выражением:

$$w_{ij} = \begin{cases} 1, i = j \\ -\varepsilon, i \neq j, \varepsilon \leq \dfrac{1}{K} \end{cases}$$

Выход y_i i-го нейрона на такте (t+1) дискретного времени рассчитывается по следующей итерационной формуле:

$$y_i(t+1) = f\left(y_i(t) - \varepsilon \sum_{j \neq i} y_j(t) \right), \qquad (3.7)$$

где $f(h)$ – активационная характеристика нейрона.

В качестве начальных условий для эволюции рассматриваемой динамической системы рассматриваются

$$y_i(0) = x_i, i = \overline{1, M}\ .$$

Заметим, что

$$\left[\varepsilon \sum_{j \neq i} y_j(t) \right] \leq \varepsilon (K-1) y_{\max}(t) \leq y_{\max}(t).$$

Отсюда следует, что для нейрона, на котором достигается максимальное значение среди всех выходов $y_i(t)$, $i = \overline{1, K}$, аргумент функции f в выражении (3.7) всегда положителен и, следовательно, на следующем такте выход этого нейрона не может иметь нулевого значения. Все другие нейроны имеют более мощную отрицательную обратную связь со стороны нейрона - "победителя" с максимальным входным сигналом. Со временем выходные сигналы этих нейронов обнуляются, так как их потенциалы становятся отрицательными. Отличным от нуля остается только выходной сигнал нейрона - "победителя". Описанная выше схема называется схемой с латеральным (боковым) торможением, в которой реализуется принцип "победитель берет все" ("winner takes all").

В данной нейросетевой схеме расчета максимума время вычисления заранее не может быть предсказано, так как существенно зависит от входных данных. На длительность переходного процесса влияет выбор значения параметра ε: чем больше значение ε ($\varepsilon \leq 1/K$), тем быстрее он завершается.

Описание программы "Классификатор Хемминга"

Программа позволяет определять класс принадлежности объектов, представленных вектором признаков длиной 30. Число классов фиксировано и равно 3. Каждый класс задается своим эталонным представителем. Для классификации используется сеть Хемминга. В программе предоставлена возможность выбора типа модуля максимизации MAXNET: *прямого распространения* или *рекуррентный*.

Программа запускается из командной строки системы MATLAB по команде '`hamming`'. Стартовое окно программы содержит краткие сведения о решаемой задаче (см. рисунок 3.10). Управление классификатором осуществляется из главного окна программы, которое показано на рисунке 3.20. Рассмотрим важные компоненты главного окна.

Для удобства ввода и визуального восприятия векторов признаков объекта и эталонов классов предусмотрены специальные панели с матрицами черных и белых клеток размера 6×5. Панель ввода для объекта находится в левой части экрана и обозначена «объект X». Панели эталонных представителей классов расположены в правой части экрана, пронумерованы и обозначены «эталоны классов». Для ввода векторов признаков требуется нажатиями мыши на клетки матриц (цвет клетки после однократного нажатия инвертируется) установить нужные изображения объекта и эталонов, после чего включить флаг «Готово» внизу окна. В результате устанавливаются новые значения весов связей нейронов.

Тип модуля MAXNET устанавливается переключателями в нижней части экрана. Соответственно меняется изображение MAXNET в центре экрана (см. рисунки 3.11 и 3.12). Для ввода весов связей и смещения каждого нейрона рабочего слоя Хемминга предусмотрено специальное диалоговое окно (см. рисунок 3.13), которое вызывается нажатием мыши на изображение соответствующего нейрона слоя. Левая часть этого окна предназначена для ввода весов связей.

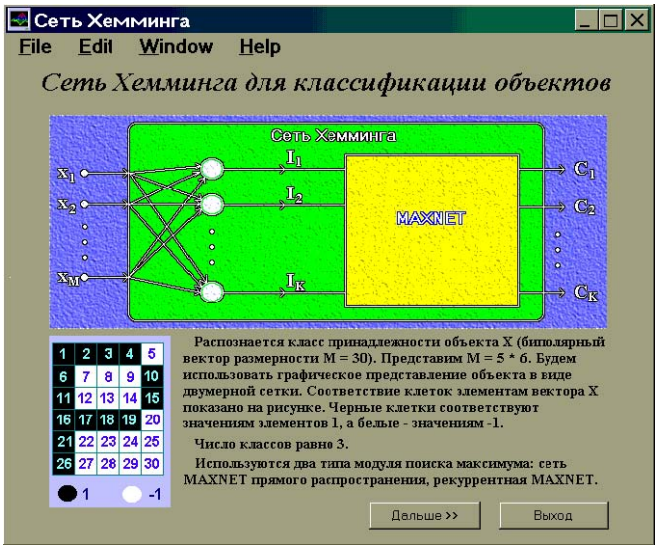


Рисунок 3.10 – Стартовое окно

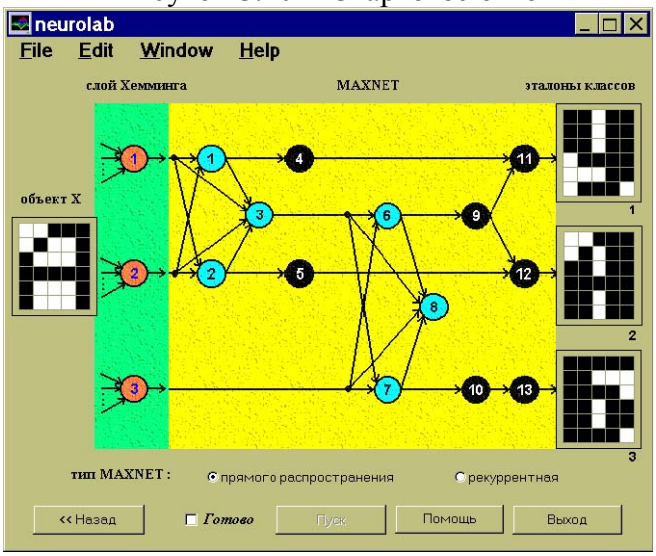


Рисунок 3.11 – Главное окно. Выбрана MAXNET прямого распространения

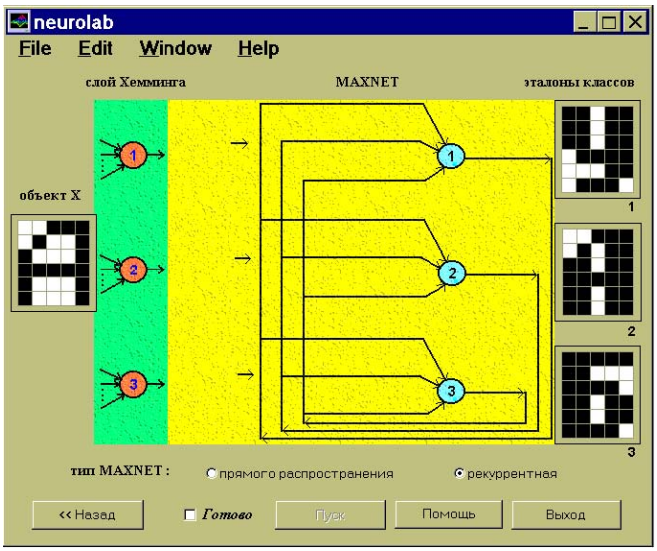


Рисунок 3.12 – Главное окно. Выбрана рекуррентная MAXNET

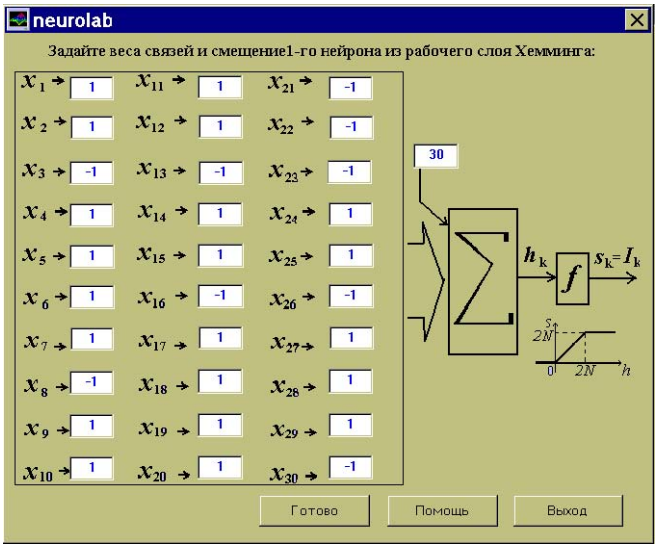


Рисунок 3.13 – Окно ввода весов и смещения 1-го нейрона рабочего слоя Хемминга

Для того чтобы просмотреть установленные или ввести новые значения весов связей сети MAXNET, нужно нажать мышью на изображение MAXNET в главном окне. В зависимости от того, какой тип MAXNET установлен в данный момент, появится окно ввода весов связей либо сети прямого распространения (см. рисунок 3.14), либо рекуррентной сети (см. рисунок 3.15).

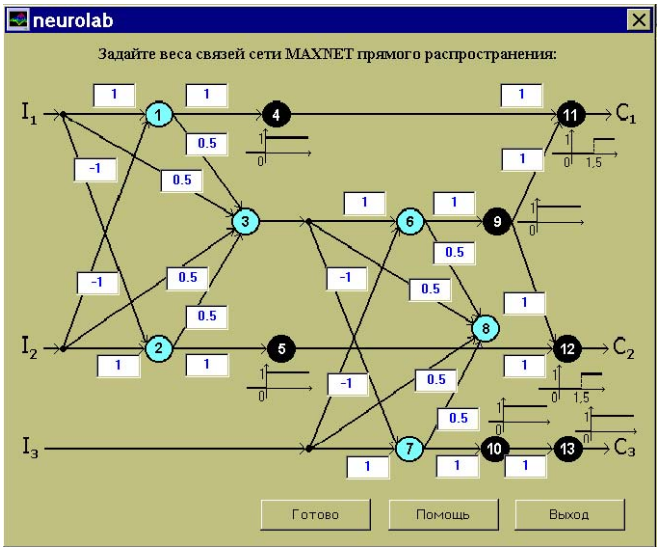


Рисунок 3.14 – Окно ввода весов связей MAXNET прямого распространения

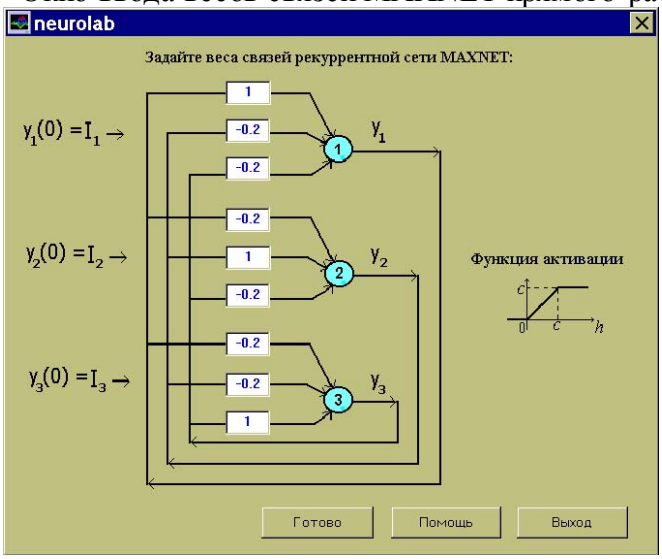


Рисунок 3.15 – Окно ввода весов связей рекуррентной MAXNET

После установки или проверки всех изменяемых параметров (с помощью описанных средств) можно запускать классификатор. Для этого нужно установить флаг «Готово». В результате этого кнопка «Пуск» внизу экрана становится доступной. Классификатор запускается по нажатии кнопки «Пуск».

Отметим, что после установки флага «Готово» автоматическая переустановка весов связей всех нейронов рабочего слоя происходит только в том случае, если за время от последней установки этого флага вносились модификации в любой из эталонных представителей классов.

В процессе классификации на экран в области изображения сети Хемминга рядом с изображениями нейронов выводятся их выходные значения. Также на экране отображается текущий номер такта функционирования MAXNET. По окончании процесса классификации рамка панели эталонного представителя класса, к которому был отнесен предъявленный объект, выделяется цветом.

Для повторного запуска классификатора достаточно снова установить флаг «Готово» (в результате информация о предыдущей работе классификатора пропадает) и нажать кнопку «Пуск».

Последовательность действий, необходимых для классификации объекта в программе «Классификатор Хемминга», проиллюстрирована на рисунок 3.16.

1. Ввести вектора признаков эталонных представителей классов, инвертируя нажатиями мыши цвет клеток на панелях, обозначенных «эталоны классов» и помеченных соответствующими номерами.
2. Аналогичным способом ввести вектор признаков объекта, подлежащего классификации. Соответствующая панель с черно-белыми клетками помечена «объект X».
3. Для каждого из трех нейронов рабочего слоя Хемминга проконтролировать автоматически установленные веса связей и смещения (либо установить другие значения). Для этого нажатием на изображение соответствующего нейрона вызвать диалоговое окно настройки параметров этого нейрона.
4. Выбрать тип MAXNET, установив соответствующий переключатель внизу окна.
5. Проконтролировать автоматически установленные веса связей (либо установить другие значения) сети MAXNET выбранного типа. Для этого нажатием на изображение MAXNET вызвать диалоговое окно установки весов.
6. Установить флаг «Готово». Нажать кнопку «Пуск».

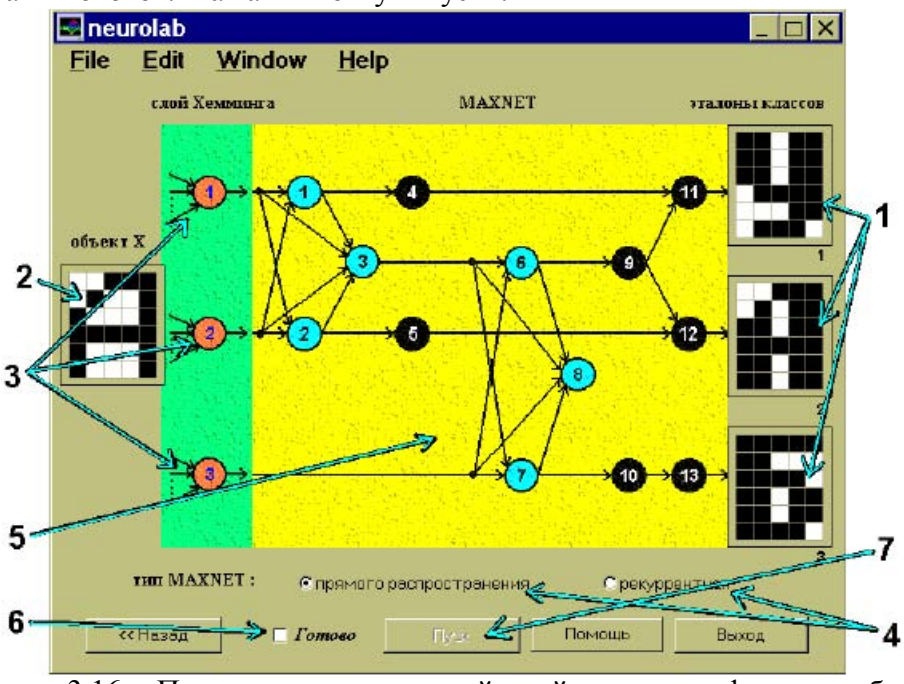


Рисунок 3.16 – Последовательность действий при классификации объекта

Последовательность выполнения работы

Классифицируемые в данной работе объекты характеризуются биполярным вектором размерности M=30. Для визуального представления объектов используется матрица размера 6×5 (n_1 = 6, n_2 = 5, $M = n_1 \cdot n_2$). Представленный для классификации объект может принадлежать одному из

трех классов (K = 3). Студенту задаются изображения эталонных представителей классов на биполярной матрице размерности 6×5.

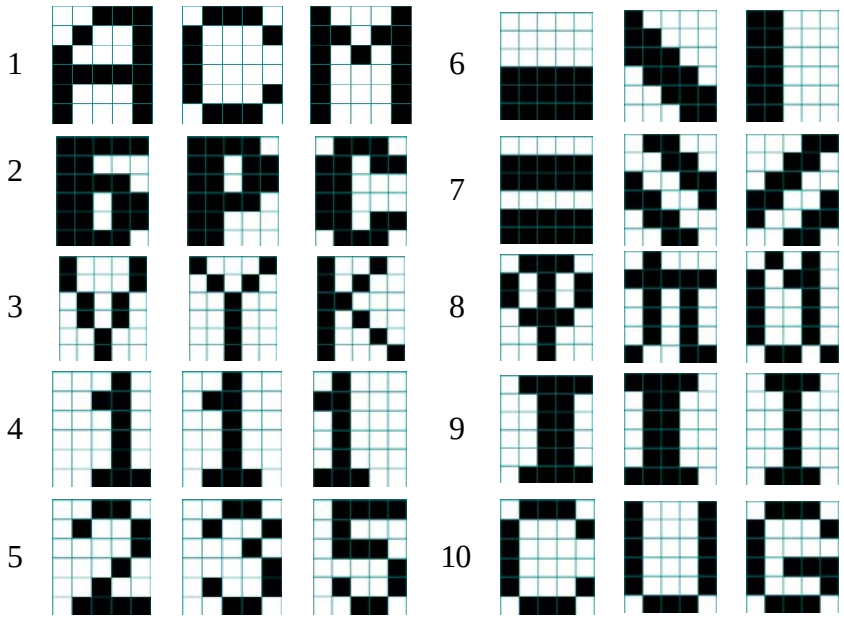
1. Подготовить данные и осуществить запуск сети Хемминга с использованием сети MAXNET прямого распространения.

1.1. Ввести эталонные изображения представителей классов.

1.2. Ввести изображение текущего классифицируемого объекта.

1.3. Проверить правильность установки синаптических коэффициентов всех нейронов рабочего слоя сети Хемминга.

Таблица 3.1 – Варианты заданий



1.4. Проконтролировать значения синаптических коэффициентов сети MAXNET прямого распространения.

1.5. Осуществить запуск сети. Проанализировать результат классификации и значения выходов нейронов рабочего слоя и сети MAXNET.

1.6. Повторить пункты 1.1-1.5 для разных вариантов задания текущего объекта.

2. Провести исследование возможности классификации объекта в зависимости от степени отличия его признаков от эталонного представителя класса.

Для этого проанализировать работу сети Хемминга для нескольких входных объектов, соответствующих разным уровням искажения эталонного представителя класса. Зафиксировать, при каком минимальном количестве несовпадений признаков текущего и эталонного объектов выполняется ошибочная классификация. Эксперимент провести для всех трех классов.

3. Провести исследование устойчивости классификации объекта при изменениях значений синаптических коэффициентов рабочего слоя и сети MAXNET. При проведении этого эксперимента могут устанавливаться произвольные действительные значения синаптических коэффициентов. Текущие объекты предполагаются близкими к эталонным представителям (отличия в двух – трех признаках). Изменением синаптических коэффициентов добиться ситуации, в которой сеть неправильно классифицирует объект, и зафиксировать уровень искажения значений синаптических коэффициентов в процентах от номинальных и число введенных искажений.

4. Подготовить данные и осуществить запуск сети Хемминга с использованием рекуррентной сети MAXNET.

4.1. Ввести эталонные изображения представителей классов.

4.2. Ввести изображение текущего классифицируемого объекта.

4.3. Проверить правильность установки синаптических коэффициентов всех нейронов рабочего слоя сети Хемминга.

4.4. Проконтролировать значения синаптических коэффициентов рекуррентной сети MAXNET.

4.5. Осуществить запуск сети. Проанализировать результат.

4.6. Повторить пункты 4.1-4.5 для разных вариантов задания текущего объекта.

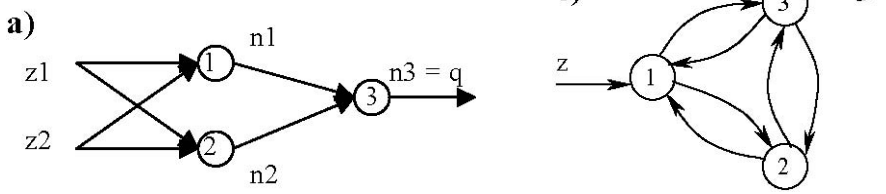
5. Проанализировать время принятия решения о принадлежности объекта классу в зависимости от значения параметра ε и различия значений потенциалов нейронов рабочего слоя. Для этого воспользоваться соответствующей модификацией синаптических коэффициентов рекуррентной сети MAXNET и текущих анализируемых объектов.

Содержание отчета

1. Название лабораторной работы, цель и программа работы.
2. Структуры исследуемых нейронных сетей и результаты моделирования.
3. Выводы по обобщающей способности нейронных сетей и скорости функционирования.

Контрольные вопросы к защите лабораторной работы

1. Какие особенности функционирования биологического нейрона учтены при построении математической модели технического нейрона?
2. Напишите уравнение функционирования технического нейрона.
3. Что называется активационной характеристикой (передаточной функцией) нейрона? Приведите примеры.
4. Какими свойствами обладают сигмоидальные нелинейные преобразователи? Приведите примеры.
5. Что называется синаптическими коэффициентом w_{ij} математической модели нейронной сети? v_{ij}?
6. Напишите уравнения, характеризующие динамику нейронной сети в дискретном времени (скалярная форма).
7. Напишите векторно-матричные уравнения, характеризующие динамику нейронной сети в дискретном времени.
8. Как записываются начальные условия для уравнений динамики нейронной сети?
9. Постройте математическое описание нейронной сети, представленной на рисунке а.
10. Постройте математическое описание нейронной сети, представленной на рисунке б.



11. Какая нейронная сеть называется рекуррентной?
12. Объясните особенности функционирования нейронной сети прямого распространения.
13. Какими особенностями обладает математическое описание многослойной нейронной сети?
14. Объясните принцип функционирования стохастического нейрона.
15. Какая активационная характеристика должна быть использована в детерминированном нейроне, который воспроизводит среднее значение выхода стохастического нейрона?
16. В чем состоит задача распознавания образов? Какая информация должна быть представлена об объекте при решении задачи распознавания образов?
17. Какую задачу решет сеть Хемминга?
18. В какой форме представлены характеристики объектов при решении задачи распознавания образов с помощью сети Хемминга?
19. Каким способом заданы классы при решении задачи распознавания образов с помощью сети Хемминга?
20. Сформулируйте критерий оптимальности отнесения объекта к одному из классов при применении сети Хемминга.
21. Какое содержание имеют выходные значения нейронов рабочего слоя сети Хемминга?
22. Сколько нейронов содержит рабочий слой сети Хемминга? Каковы их активационные характеристики?
23. Какими способами можно решать задачу поиска максимума среди K значений при построении сети Хемминга?
24. Нарисуйте схему нейросетевого компаратора на два входа и объясните его работу. Какие активационные характеристики нейронов используются в схеме нейросетевого компаратора?

25. Объясните принцип построения сети MAXNET прямого распространения при произвольном числе входов с использованием нейросетевого компаратора на два входа.
26. Нарисуйте схему рекуррентной сети MAXNET. Какие начальные условия устанавливаются на сети?
27. Какой параметр рекуррентной сети MAXNET может влиять на ее быстродействие?
28. Объясните принцип работы рекуррентной сети MAXNET.

Лабораторная работа № 4 «Сеть Хопфилда»

Цель работы

В лабораторной работе исследуются свойства нейронной сети Хопфилда, которая рассматривается как модель ассоциативной памяти, позволяющей восстановить объект по ограниченному набору зашумленных признаков. Сеть Хопфилда относится к классу рекуррентных нейронных сетей. В асинхронном режиме работы нейронов сеть Хопфилда из произвольного начального состояния за конечное число тактов дискретного времени приходит в состояние устойчивого равновесия (аттрактор). Чисел различных аттракторов сети определяет ее объем «памяти» (число запомненных образов).

Экспериментальное исследование свойств сети Хопфилда выполняется в нейроэмуляторе, реализованном в системе MATLAB. Изучается динамика переходного процесса, методом статистического моделирования рассчитывается число аттракторов и размеры бассейнов аттракторов. Исследуется устойчивость решений при зашумлении данных.

Программа работы

4. Исследовать процесс эволюции сети Хопфилда от начального состояния к устойчивому состоянию (динамика).
5. Определить множество устойчивых состояний сети, размер бассейнов аттракторов (аттракторы).
6. Исследовать способность сети сохранять в качестве аттракторов некоторое множество образов и релаксировать к сохраненному образу при подаче на вход этого же образа с искажениями (ассоциативная память).
7. Исследовать способность сети сохранять свойство ассоциативной памяти после искажения матрицы синаптических связей (робастность).
8. Составить и защитить отчет по результатам исследований.

Краткие сведения из теории

При организации вычислений в компьютере вызов из его “памяти” необходимых данных производится по их адресам. Имеется в виду, что требуемые данные строго локализованы в некоторой фиксированной области памяти. Человеческая память организована принципиально иначе. Вызов нужной информации обеспечивается, как правило, некоторыми ассоциациями. Здесь могут быть полезными сведения, которые уточняют особенности вспоминаемого объекта и связанные с ним события.

Таким образом, можно классифицировать устройства памяти по признакам “адресно-ориентированного поиска информации” или “поиска данных по их содержанию” (“address-oriented memory” или “content-addressable memory”). Память, в которой хранение и поиск информации основаны на ее содержании, носит название ассоциативной (associated memory). Ассоциативная память позволяет восстановить информацию в случае ее частичной потери или при значительном искажении входного запроса.

Если память позволяет восстановить поданный на ее вход искаженный неполный образ, то она называется автоассоциативной. Если при подаче на вход одного образа память реагирует извлечением другого, семантически связанного с запросом, то такая память называется гетероассоциативной (heteroassociative). Например, при отгадывании слов в кроссворде реализуются задачи автоассоциативной памяти, когда на входе заданы несколько букв слова, а память восстанавливает полное слово. При поиске марки стали по набору ее прочностных характеристик память проявляет себя как гетероассоциативная.

Сформулируем математическую задачу, связанную с функционированием автоассоциативной памяти. Предположим, что память предназначена для хранения P образов (patterns). Каждый

образ представлен вектором признаков , $x^p = \left(x_1^p, x_2^p, ..., x_N^p\right)$, $p = \overline{1,P}$. На вход автоассоциативной памяти предъявляется некоторый входной образ с набором признаков $n = \left(n_1, n_2, ..., n_N\right)$. Требуется найти среди хранящихся в памяти такой образ x^λ, который наиболее близок к *n* с точки зрения евклидовой меры, т. е. для которого достигается

$$\min_{p=\overline{1,P}} H_p = \min_{p=\overline{1,P}} \sum_{i=1}^{N} \left(n_i - x_i^p\right)^2 . \qquad (4.1)$$

Если признаки являются бинарными (возможные значения 0 и 1), то показатель H_p близости образов *n* и x^p характеризуется расстоянием Хэмминга между двумя *N*–разрядными бинарными последовательностями, т. е. числом несовпадающих бинарных разрядов.

Хопфилд (Hopfield) предложил следующую идею реализации автоассоциативной памяти для образов, характеризующихся векторами бинарных признаков. Рассмотрим нейронную сеть с бинарными нейронами (пороговая активационная характеристика) и полными связями. При начальном возбуждении вектором *n* сеть динамически изменяет свое состояние в дискретном времени. Пусть сеть является устойчивой и за конечное число тактов приходит к одному из своих состояний равновесия. Это предельное состояние и будет представлять собой образ, ассоциированный с входным возбуждением *n*. Число устойчивых состояний сети (аттракторов) представляет собой число хранящихся в сети образов. Сама нейронная сеть выполняет при этом функцию ассоциативной памяти.

Хопфилд сконструировал нейронную сеть, обладающую указанными свойствами. В дальнейшем были разработаны различные модификации этой модели и статистическая интерпретация ее работы.

Рассмотрим полносвязную нейронную сеть, содержащую *N* нейронов по числу признаков образов. На рисунке 4.1 представлена схема такой сети. Каждый нейрон сети представляет собой биполярный элемент (s_i = ±1, $i = \overline{1,N}$), динамика которого в дискретном времени *t* = 0, 1, 2, … описывается системой уравнений:

$$s_i\left(t+1\right) = \begin{cases} \operatorname{sgn} h_i\left(t\right), h_i\left(t\right) \neq 0; \\ s_i\left(t\right), h_i\left(t\right) = 0; \quad i = \overline{1,N}. \end{cases} \qquad (4.2)$$

В (2.2) приняты следующие обозначения:

$h_i\left(t\right) = \sum_{j=1}^{N} w_{ij} s_j\left(t\right) - b_i$ – потенциал *i*–го нейрона; (4.3)

b_i – смещение *i*–го нейрона;

$$\operatorname{sgn}\left(z\right) = \begin{cases} 1, z > 0; \\ -1, z \leq 0; \end{cases}$$

w_{ij} – коэффициент синаптической связи *j*-го нейрона с *i*-м, $i, j = \overline{1,N}$.

В некоторых приложениях может оказаться более удобным применение бинарных нейронов с допустимыми значениями $\tilde{s}_i$, равными 0 и 1. Переход от биполярных нейронов к бинарным осуществляется с помощью линейного преобразования $s_i = 2\tilde{s}_i - 1$.

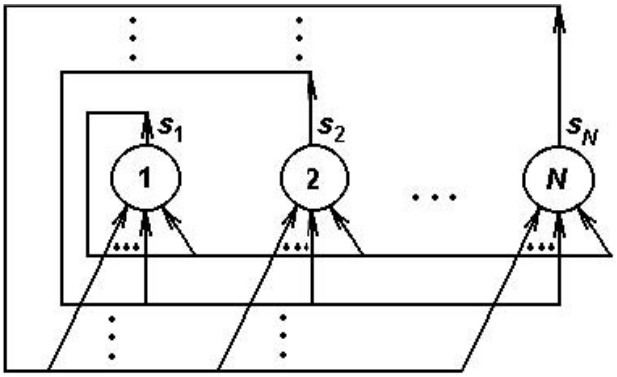


Рисунок 4.1 – Полносвязная нейронная сеть

Пусть матрица W синаптических связей размерности [N x N] является симметричной и имеет нулевые диагональные элементы (отсутствует непосредственная обратная связь с выхода нейрона на собственный вход), т.е. выполняются равенства: $w_{ii} = 0$, , $i \neq j$, $w_{ij} = w_{ji}$, $i, j = \overline{1, N}$.

Кроме того, положим смещения нейронов равными нулю: $b_i = 0$, $i = \overline{1, N}$.

Возможны два способа изменения состояний нейронов в соответствии с уравнениями (2.2). Первый способ предполагает, что в текущий момент времени t фиксируется вектор $s(t) = (s_1(t), s_2(t), ..., s_N(t))$ и одновременно (параллельно) изменяются состояния всех нейронов согласно уравнениям (4.2). Таким образом, за один такт дискретного времени происходит переход к новому вектору $s(t+1) = (s_1(t+1), s_2(t+1), ..., s_N(t+1))$. Такое функционирование нейронной сети получило название синхронного.

Другой способ организации вычислений в сети состоит в последовательном изменении состояний нейронов согласно уравнениям (4.2). На каждом такте дискретности фиксируется один нейрон и вычисляется его новое состояние. Порядок опроса нейронов в такой последовательной процедуре может быть произвольным. (Хопфилд рассматривал схему случайного равновероятного выбора нейронов для расчета их новых состояний). Описанная последовательная динамика нейронной сети получила название асинхронной. В дальнейшем динамика сети полагается асинхронной, т. е. на одном такте дискретности может измениться состояние только одного нейрона.

Рассмотрим геометрическую интерпретацию состояний нейронной сети. N признаков (N нейронов) образуют N–мерное пространство. Поскольку для признаков допустимыми значениями являются –1 и +1, любой вектор состояния нейронной сети направлен в вершину гиперкуба, имеющего центр в начале координат и ребра длины 2, параллельные осям координат (см. рисунок 4.2). При асинхронной динамике сети вектор состояния за один такт дискретности может переместиться в одну из вершин гиперкуба, непосредственно прилегающих к текущей вершине (расстояние равно длине ребра).

Пусть заданы P образов своими векторами признаков x^p, $p = \overline{1, P}$. Определим синаптические коэффициенты , $i \neq j$, w_{ij}, $i, j = \overline{1, N}$, следующим выражением:

$$w_{ij} = k \sum_{p=1}^{P} x_i^p x_j^p \text{ , } k > 0. \qquad (4.4)$$

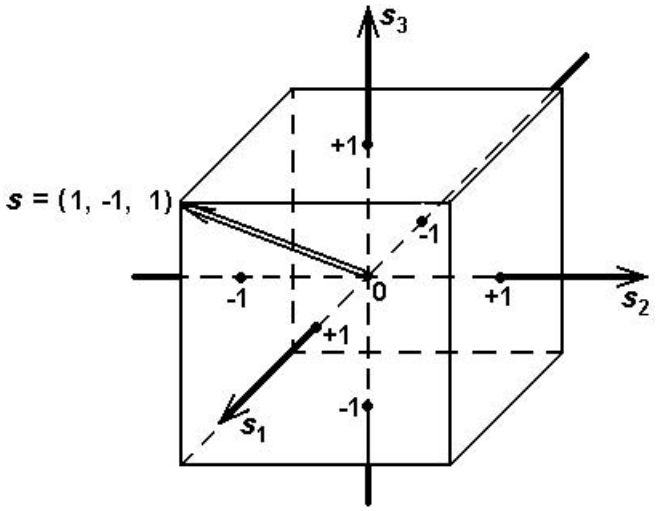


Рисунок 4.2 – Геометрическая интерпретация состояний сети Хопфилда

Динамические свойства сети не зависят от значения положительного параметра k. Это следует из того, что согласно уравнениям (4.2) состояние $s_i(t + 1)$ нейрона на следующем такте дискретного времени определяется только знаком его потенциала $h_i(t)$. Поскольку в соответствии с выражением (4.3) для потенциала $h_i(t)$ параметр $k > 0$ не влияет на его знак, этот параметр не влияет и на динамику сети в целом. В частности, если положить $k = 1/P$, то значение w_{ij} служит оценкой статистической связи i-го и j-го признаков рассматриваемых образов x^p, $p = \overline{1, P}$, для заданной выборки.

Текущее состояние динамической нейронной сети характеризуется следующим энергетическим функционалом:

$$E(t) = -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij} s_i(t) s_j(t). \tag{4.5}$$

Исследуем, какие изменения претерпевает энергетический функционал в процессе эволюции состояния сети Хопфилда. Режим функционирования сети полагается асинхронным. Пусть на такте $(t + 1)$ проведен "опрос" k-го нейрона и его состояние в соответствии с формулами (4.2) и (4.3) изменилось:

$$s_k(t+1) = s_k(t) + \Delta s_k(t), \tag{4.6}$$

где $\Delta s_k(t) \neq 0$. Для остальных нейронов в соответствии с правилами асинхронного функционирования выполняется равенство:

$$s_i(t+1) = s_i(t),\ i \neq ki = \overline{1, N}. \tag{4.7}$$

Рассмотрим изменение энергетического функционала:

$$\Delta E(t+1) = E(t+1) - E(t) =$$

$$= -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij} s_i(t+1) s_j(t+1) + \frac{1}{2}\sum_{i,j=1}^{N} w_{ij} s_i(t) s_j(t) =$$

$$= -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij} \left(s_i(t) + \Delta s_i(t)\right)\left(s_j(t) + \Delta s_j(t)\right) + \frac{1}{2}\sum_{i,j=1}^{N} w_{ij} s_i(t) s_j(t).$$

После раскрытия скобок в последнем выражении и приведения подобных членов получим:

$$\Delta E(t+1) = -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij} s_i(t)\Delta s_j(t) - \frac{1}{2}\sum_{i,j=1}^{N} w_{ij} s_j(t)\Delta s_i(t) -$$

$$-\frac{1}{2}\sum_{i,j=1}^{N} w_{ij}\Delta s_i(t)\Delta s_j(t). \qquad (4.8)$$

Согласно выражениям (4.6), (4.7) последняя сумма содержит только одно слагаемое с отличным от нуля парным произведением $\Delta s_i(t)\Delta s_j(t)$ при $i = j = k$, но при этом множитель $w_{ij} = w_{kk}$ равен нулю (диагональный элемент матрицы W синаптических коэффициентов). Таким образом, в выражении (4.8) остаются только первые две суммы. В связи с тем, что $w_{ij} = w_{ji}$, эти две суммы совпадают и потому справедливо равенство:

$$\Delta E(t+1) = -\sum_{i,j=1}^{N} w_{ij} s_j(t)\Delta s_i(t) = -\Delta s_k(t)\sum_{j=1}^{N} w_{kj} s_j(t) =$$

$$= -\Delta s_k(t) h_k(t). \qquad (4.9)$$

Рассмотрим два возможных случая: $h_k(t) > 0$ и $h_k(t) < 0$ ($h_k(t) \neq 0$, т. к. в противном случае не может обеспечиваться условие $\Delta s_k(t) \neq 0$ в выражении (4.6).

Если $h_k(t) > 0$, то $s_k(t + 1) = 1$. Это означает, что $s_k(t) = -1$ и $\Delta s_k(t) = s_k(t+1) - s_k(t) = 2$. Отсюда следует, что $\Delta E(t+1) = -2h_k(t) < 0$.

Если $h_k(t) < 0$, то $s_k(t + 1) = -1$. Это означает, что $s_k(t) = 1$ и $\Delta s_k(t) = s_k(t+1) - s_k(t) = -2$. Отсюда следует, что $\Delta E(t+1) = 2h_k(t) < 0$.

Таким образом, изменение состояния нейрона сети в режиме ее асинхронного функционирования приводит к уменьшению энергетического функционала. В силу ограниченности снизу значения энергетического функционала $E(t)$ для конечного числа нейронов сети и приращений $\Delta E \leq 0$ через конечное число тактов энергетический функционал достигает одного из своих локальных минимумов, который является состоянием устойчивого равновесия сети.

Полученный результат и составляет содержание теоремы о конечности переходного процесса в сети Хопфилда.

В теории динамических систем $E(t)$ называется функцией Ляпунова. Состояние устойчивого равновесия нейронной сети (как и любой другой динамической системы) называется ее аттрактором. Сеть Хопфилда может иметь множество аттракторов, являющихся точками локальных минимумов энергетического функционала. Начиная свое движение из состояния $s(0)$, сеть Хопфилда "сваливается" в ближайший локальный минимум через некоторое число временных тактов. На рисунке 4.3 дана иллюстрация этого свойства. В целях упрощения различные возможные состояния сети указаны вдоль горизонтальной оси.

Обозначим $s(\infty)$ состояние устойчивого равновесия сети Хопфилда, а $h(\infty)$ — соответствующий вектор потенциалов нейронов. Согласно выражению (4.3) $h_i(\infty) = \sum_{j=1}^{N} w_{ij} s_j(\infty)$, $i = \overline{1, N}$. Это позволяет записать выражение для энергетического функционала в следующей форме:

$$E(\infty) = -\sum_{i,j=1}^{N} w_{ij} s_i(\infty) s_j(\infty) = -\sum_{i=1}^{N} h_i s_i(\infty). \qquad (4.10)$$

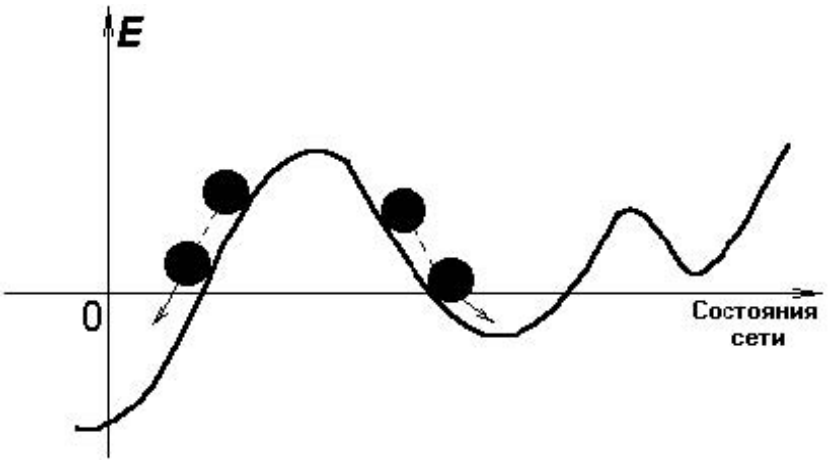


Рисунок 4.3 – Энергия сети Хопфилда

В состоянии устойчивого равновесия знак $h_i(\infty)$ согласно уравнениям динамики (4.2) совпадает со знаком $s_i(\infty)$, $i=\overline{1,N}$, или $h_i(\infty) = 0$. В противном случае наблюдались бы изменения состояний нейронов и режим не был бы установившимся. Следовательно, для всех $i=\overline{1,N}$ произведения $h_i(\infty)s_i(\infty)\geq 0$. Применение этого неравенства к (4.10) позволяет заключить, что $E(\infty)\leq 0$.

Анализ устойчивых состояний сети Хопфилда начнем с частного случая, когда $P = 1$, т. е. в сети хранится единственный образ x^1. В этом случае справедливо следующее выражение для синаптических коэффициентов:

$$w_{ij}=\begin{cases}0, i=j,\\ kx_i^1x_j^1, i\neq j, k>0.\end{cases} \tag{4.11}$$

Проверим, является ли x^1 устойчивым состоянием нейронной сети. Предположим, что в результате эволюции сеть из некоторого начального состояния пришла в состояние x^1 в момент времени t, т. е. $s(t) = x^1$. Тогда ее состояние на следующем такте определяется выражением:

$$s_i(t+1)=\begin{cases}\operatorname{sgn}h_i(t), h_i(t)\neq 0;\\ s_i(t), h_i(t)=0;\quad i=\overline{1,N},\end{cases} \tag{4.12}$$

где $h_i(t)=\sum_{j=1}^{N}w_{ij}s_j(t)=k\sum_{\substack{j=1\\ i\neq j}}^{N}x_i^1x_j^1s_j(t)=k\sum_{\substack{j=1\\ i\neq j}}^{N}x_i^1\left(x_j^1\right)^2=\left(k\sum_{\substack{j=1\\ i\neq j}}\left(x_j^1\right)^2\right)x_i^1$, $i=\overline{1,N}$

Заметим, что в связи с положительностью значения $\left(k\sum_{\substack{j=1\\ i\neq j}}^{N}\left(x_j^1\right)^2\right)$ в полученном выражении для $h_i(t)$ знак $h_i(t)$ (при $h_i(t)\neq 0$) определяется знаком x_j^1.

В обоих случаях согласно выражению (4.12) $s_i(t+1) = s_i(t) = x_i^1,\ i = \overline{1,N}$. Таким образом, состояние $s(t) = x^1$ не изменяется на следующем такте времени и является, следовательно, аттрактором сети.

Анализ показывает, что аттрактором рассматриваемой сети при $P = 1$ является и инверсное состояние $(-x^1)$. Студентам предлагается самостоятельно доказать это утверждение.

Рассматриваемому в сети Хопфилда биполярному (значения +1 и –1) вектору признаков x^p, равно как и биполярному вектору состояний нейронов $s(t)$ можно дать простую графическую интерпретацию (см. рисунок 4.4). Рассматривается прямоугольник, содержащий $N = n_1 \times n_2$ черно-белых клеток. Каждая клетка поставлена в соответствие определенной координате вектора признаков или нейрону сети Хопфилда. Клетка зачернена, если соответствующее значение координаты вектора равно 1, и остается белой в противном случае (значение –1). В такой графической интерпретации доказанное выше свойство сети Хопфилда может быть сформулировано следующим образом: если сеть рассчитана на хранение одного образа (изображения), то этот образ и его негатив являются аттракторами сети.

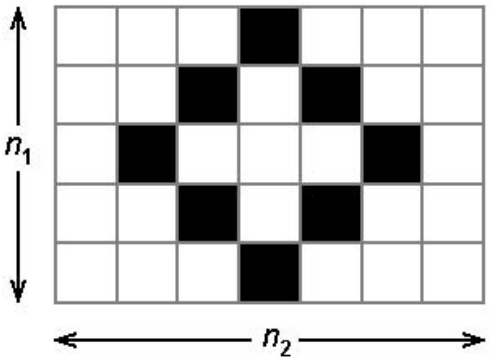


Рисунок 4.4 – Графическая интерпретация вектора состояний нейронов

Аттрактор динамической системы обладает тем свойством, что к этому состоянию сеть "сваливается", если находится в некоторой его окрестности. Оценим ширину этой окрестности для рассматриваемого случая $P = 1$. Допустим, что в момент времени t состояние сети $s(t)$ не полностью совпадает с аттрактором x^1. Предположим для определенности, что различие имеет место для m координат векторов $s(t)$ и x^1:

$$s_i(t) = \begin{cases} -x_i^1, i = \overline{1,m}; \\ x_i^1, i = \overline{m+1,N}. \end{cases} \tag{4.13}$$

Рассчитаем в этом случае состояние сети $s_i(t+1),\ i = \overline{1,N}$. Для этого следует оценить значение потенциала $h_i(t),\ i = \overline{1,N}$:

$$h_i(t) = k\sum_{\substack{j=1 \\ i \neq j}}^{N} x_i^1 x_j^1 s_j(t) = k\left[-\sum_{\substack{j=1 \\ i \neq j}}^{m} x_i^1 \left(x_j^1\right)^2 + \sum_{\substack{j=m+1 \\ i \neq j}}^{N} x_i^1 \left(x_j^1\right)^2\right] =$$

$$= kx_i^1\left[-M_{1i} + M_{2i}\right], \tag{4.14}$$

где $M_{1i} = \sum_{\substack{j=1 \\ i \neq j}}^{m} x_i^1 \left(x_j^1 \right)^2$, $M_{2i} = \sum_{\substack{j=m+1 \\ i \neq j}}^{N} x_i^1 \left(x_j^1 \right)^2$. Заметим, что в зависимости от значения индекса i M_{1i} может принимать значения $(m - 1)$ или m, а M_{2i} — $(N - m)$ или $(N - m - 1)$ соответственно. В обоих случаях $-M_{1i} + M_{2i} \geq N - 2m - 1$. Пусть число различий векторов $s(t)$ и x^1 таково, что $N - 2m - 1 > 0$. Тогда $-M_{1i} + M_{2i} > 0$ и в соответствии с выражением (4.14) знак $h_i(t)$ определяется знаком x_i^1. Следовательно, для любого $i = \overline{1, N}$, на такте $(t + 1)$ установится значение $s_i(t+1) = \operatorname{sgn} h_i(t) = x_i^1$.

Это означает, что за N тактов дискретного времени все нейроны гарантированно будут приведены к состояниям x_i^1, $i = \overline{1, N}$. Таким образом, даже при 50% различии входного образа сети (состояния $s(0)$) от хранимого в сети образа x^1 (число различий m должно удовлетворять неравенству $m < (N - 1) / 2$) сеть дает правильную ассоциацию.

Перейдем к рассмотрению общего случая $P > 1$. Зафиксируем один из образов x^p, $p = \overline{1, P}$, на основании которых конструируется сеть, обозначив его x^λ. Проверим, является ли x^λ аттрактором сети. Анализ будем проводить по той же схеме, что применялась в случае $P = 1$. Допустим, что на временном такте t установилось значение $s(t) = x^\lambda$, и рассчитаем значение $s(t + 1)$. Расчет $s(t + 1)$ предполагает необходимость вычисления вектора потенциалов нейронов $h(t)$:

$$h_i(t) = \sum_{j=1}^{N} w_{ij} s_j(t) = k \sum_{p=1}^{P} \sum_{\substack{j=1 \\ i \neq j}}^{N} x_i^p x_j^p x_j^\lambda, \; i = \overline{1, N}. \tag{4.15}$$

В выражении (4.15) выделим из суммы по индексу p одно слагаемое, для которого $p = \lambda$:

$$h_i(t) = k \left[\sum_{\substack{j=1 \\ i \neq j}}^{N} x_i^\lambda \left(x_i^\lambda \right)^2 + \sum_{\substack{p=1 \\ p \neq \lambda}}^{P} \sum_{\substack{j=1 \\ i \neq j}}^{N} x_i^p x_j^p x_i^\lambda \right] = k' \left(x_i^\lambda + r_i \right), \; i = \overline{1, N}, \tag{4.16}$$

где $k' = k\,(N - 1) > 0$,

$$r_i = \frac{1}{N-1} \sum_{\substack{p=1 \\ p \neq \lambda}}^{P} \sum_{\substack{j=1 \\ i \neq j}}^{N} x_i^p x_j^p x_i^\lambda. \tag{4.17}$$

Из выражения (4.16) следует, что знак $h_i(t)$ определяется суммой $x_i^\lambda + r_i$. Если образы x^λ и x^p, $p \neq \lambda$, $p = \overline{1, P}$, некоррелированы (ортогональны), т. е.

$$\sum_{j=1}^{N} x_j^p x_j^\lambda = 0, \tag{4.18}$$

то согласно (4.17) r_i близко к нулю (отличие от нуля может быть связано лишь с исключением из суммы по индексу j в (4.17) слагаемого, для которого $i \neq j$). Таким образом, в этом случае член r_i в (4.16) в подавляющем большинстве случаев не влияет на знак $h_i(t)$ и можно считать, что знаки x_i^λ и $h_i(t)$ совпадают. Следовательно, $s(t + 1)$ не будет отличаться от $s(t) = x^\lambda$, т. е. x^λ является аттрактором сети.

Рассмотрим теперь общий случай, когда свойство ортогональности (4.18) не выполняется. В этом случае уместен статистический анализ значения r_i. В связи с предполагаемым разнообразием предъявленных для обучения сети образов x^p, $p = \overline{1,P}$, можно допустить, что отдельные слагаемые $x_i^p x_j^p x_i^\lambda$ в выражении (4.17) для r_i представляют собой независимые случайные величины с равновозможными значениями +1 и –1. Это означает, что математическое ожидание каждого слагаемого равно нулю, а дисперсия равна единице. Из центрированности каждого слагаемого в выражении для r_i следует, что и случайная величина r_i в целом является центрированной. Дисперсия r_i вычисляется с применением свойства независимости отдельных слагаемых, имеющих дисперсию, равную единице:

$$D(r_i) = \frac{P-1}{N-1}. \qquad (4.19)$$

Для больших значений P и N можно написать следующее приближенное выражение:

$$D(r_i) \cong \frac{P}{N}. \qquad (4.20)$$

Полученные статистические характеристики величины r_i позволяют построить для нее доверительный интервал $\left[-2.5\sigma[r_i]; +2.5\sigma[r_i]\right]$, где $\sigma[r_i] = \sqrt{\frac{P}{N}}$. Согласно выражению (4.16) знак $h_i(t)$ с высоким уровнем надежности не будет отличаться от x_i^λ, если абсолютное значение r_i не превосходит единицы, т. е. границы доверительного интервала для r_i удовлетворяют неравенству:

$$2.5\sigma[r_i] = 2.5\sqrt{\frac{P}{N}} < 1.$$

Отсюда следует, что должно выполняться неравенство:

$$P < 0.16\,N. \qquad (4.21)$$

Выполнение условия (4.21) обеспечивает высокую вероятность того, что образ x^λ, достигнутый в сети на такте t, не изменится на следующем такте дискретности: $s(t+1) = x^\lambda$. Иначе говоря, в этом случае x^λ является одним из аттракторов сети Хопфилда или “запомненным” сетью образом. Неравенство (4.21) можно также интерпретировать как следующее утверждение: число образов, которые может запомнить сеть Хопфилда, зависит от числа содержащихся в ней нейронов и составляет около $0.16\,N$. Если образы, используемые для обучения сети, сильно коррелированы, то число устойчивых состояний (или запомненных образов) будет ниже $0.16\,N$. В этом случае вполне возможно, что несколько разных образов обучающей выборки сходятся к одному и тому же аттрактору. Сами же аттракторы могут не совпадать с предъявленными сети образами.

Рассмотренные выше свойства сети Хопфилда позволяют разделить множество аттракторов любой сети на три группы. К первой группе отнесем те аттракторы, которые «притягивают» образы обучающей выборки. Именно благодаря этим аттракторам сеть Хопфилда является ассоциативной памятью. Вторую группу аттракторов образуют “негативы” (или “зеркальные отображения”) аттракторов первой группы. В третью группу включены аттракторы, которые не соответствуют ни одному примеру обучающей выборки x^p, $p = \overline{1,P}$. Эти аттракторы называют “ложной памятью” (spurious memory).

При использовании сети Хопфилда как ассоциативной памяти аттракторы третьей группы представляют собой нежелательные состояния, которые следует подавить. В то же время некоторые авторы склонны рассматривать эти аттракторы как сгенерированные сетью новые образы, акцентируя внимание на “творческом начале”, демонстрируемом сетью [11].

Применение сети Хопфилда для кластеризации данных

Допустим, что имеется представительная выборка данных, которые должны быть разделены на группы (классы) в соответствии с принципом похожести внутри группы и различия групп. Предположим также, что априорная информация о количестве групп и их отличительных признаках отсутствует. Процедура группирования данных иначе называется кластеризацией. Имеется множество различных формализованных постановок задач кластеризации. В них предлагаются критерии качества решения и различные вычислительные процедуры для достижения цели.

Задача кластеризации может быть решена и с помощью сети Хопфилда. Поставим в соответствие группе данных один аттрактор сети Хопфилда. Этот аттрактор может рассматриваться как эталонный представитель группы, или прототип группы. Следует отметить, что прототип группы может не содержаться в обучающей выборке — "ложная память". Число аттракторов в сети Хопфилда в такой постановке задачи кластеризации представляет собой число сформированных сетью групп.

Известен алгоритм, который приводит к решению задачи кластеризации с помощью сети Хопфилда. Этот алгоритм является многошаговым и, как установлено экспериментально, быстро сходится. На первом шаге по данным обучающей выборки рассчитывается сеть Хопфилда и анализируются ее аттракторы. Поиск всех аттракторов осуществляется путем генерации случайных образов и использования их в качестве начального возбуждения построенной сети. При этом фиксируются все устойчивые состояния сети. Может оказаться, что некоторые аттракторы не притягивают ни одного примера из заданной обучающей выборки и поэтому не могут рассматриваться как прототипы реальных групп данных. Эти аттракторы фактически обозначают пустые кластеры.

Следующий шаг процедуры кластеризации состоит в добавлении к обучающей выборке аттракторов, соответствующих пустым классам. Основанием для такого расширения примеров для обучения является предположение, что обучающая выборка недостаточно представительна и полученные в сети аттракторы могут быть реальными данными, которые прежде не наблюдались. После расширения обучающей выборки все действия по расчету сети Хопфилда и поиску аттракторов повторяются. Как правило, число аттракторов сокращается. Уменьшается и число пустых кластеров.

Процедура расширения обучающих примеров за счет аттракторов пустых кластеров повторяется до тех пор, пока в сети не будет пустых кластеров. Число шагов процедуры зависит от числа нейронов в сети и фактической сгруппированности экспериментальных данных.

Новые обучающие примеры, сгенерированные в процессе кластеризации, обычно вливаются в группы, содержащие примеры обучающей выборки. В этом случае они могут трактоваться как возможные образы группы. В то же время допустима ситуация, когда кластер полностью состоит из добавленных примеров. Такой кластер может рассматриваться как предсказание новой возможной категории данных.

Применение сети Хопфилда для классификации объектов по заданному вектору признаков

Для построения классификатора данных, который настраивается по примерам, необходимо использовать информацию о классе принадлежности каждого из примеров обучающей выборки. Таким образом, полное множество обучающих примеров может быть представлено объединением подмножеств, каждое из которых содержит примеры одного класса. Пусть, например, решается задача распознавания буквы русского алфавита по черно–белому изображению на прямоугольной сетке. Один класс в этом примере включает в себя различные начертания определенной буквы, а общее число классов равно 32.

Свяжем с каждым классом отдельную сеть Хопфилда, синаптические коэффициенты которой рассчитываются только по тем обучающим примерам, которые принадлежат рассматриваемому классу. Разные аттракторы сети Хопфилда, настроенной на один класс, соответствуют различным модификациям объектов класса (например, различным типам начертания одной и той же буквы в задаче распознавания русских букв).

Для определения класса принадлежности некоторого объекта, который не входил в состав обучающей выборки, вектор его признаков предъявляется всем сетям классов. Каждая сеть под воздействием этого начального возбуждения эволюционирует к одному из своих аттракторов. Далее оцениваются среднеквадратические расстояния от вектора признаков объекта до аттракторов

разных классов. «Побеждает» тот класс, для которого расстояние оказалось минимальным. Вместо расстояния можно использовать время переходного процесса от начального возбуждения до достижения сетью аттрактора.

Описанный способ классификации привлекателен благодаря своей наглядности и простоте модификации классификатора при добавлении новых обучающих примеров.

Последовательность выполнения работы

Данные для работы сети представляют собой набор биполярных векторов (элементы вектора могут иметь одно из двух возможных значений: {–1; +1}) одинаковой заданной длины N, которые в используемой на лабораторном занятии программе для удобства зрительного восприятия визуализируются в виде прямоугольной матрицы с заданным числом строк n_1 и столбцов n_2 ($n_1 n_2 = N$). Элементам вектора, равным –1, ставится в соответствие белая клетка, равным +1 — черная (рисунок 4.5). Для создания и редактирования подобных наборов образов в программном обеспечении лабораторного занятия предусмотрен графический редактор, который сохраняет вектора и параметры n_1 и n_2 в текстовых файлах специального формата с расширением pat. Подробнее о программе смотрите в п. «Описание программного обеспечения».

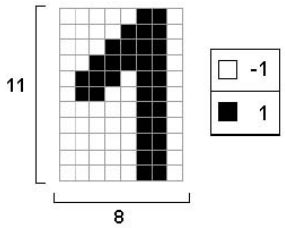


Рисунок 4.5 – Графическая интерпретация образа. Вектор из 88 элементов представлен в виде матрицы с 11 строками и 8 столбцами

Каждому из четырех указанных выше аспектов модели Хопфилда в лабораторной работе посвящено экспериментальное исследование. Рассмотрим содержание этих экспериментов.

Эксперимент «Динамика». Целью эксперимента является определение средней длительности процесса эволюции сети Хопфилда к устойчивому состоянию, длительности переходного процесса в зависимости от числа различий входа сети и устойчивого состояния, проверка свойств энергетического функционала.

На вход сети в качестве начального состояния нейронов подается некоторый образ. После этого запускается функционирование сети в соответствии с уравнениями динамики модели Хопфилда (в асинхронном режиме). При этом за один такт (цикл) *последовательно* в случайном порядке вычисляется новое состояние каждого из N нейронов сети. Порядок обработки нейронов выбирается перед каждым тактом с помощью генератора случайных чисел по равномерному закону. После каждого такта выполняется проверка идентичности достигнутого состояния сети состоянию сети после предыдущего такта. Когда эта проверка устанавливает факт достижения устойчивого состояния, динамический процесс останавливается.

Эволюция сети сопровождается изменением энергетического функционала. Текущее значение энергии вычисляется для начального состояния сети и далее пересчитывается после обработки каждого нейрона. По результатам вычислений автоматически строится график зависимости энергии сети от времени асинхронного функционирования.

Эксперимент «Аттракторы». В данном исследовании требуется найти устойчивые состояния сети Хопфилда, определить их количество, тип (истинная/ложная память), размер бассейна.

Число всех возможных состояний сети Хопфилда, содержащей N нейронов, равно 2^N. Некоторые из них являются устойчивыми, причем множество аттракторов могут составлять как те образы, по которым рассчитана синаптическая матрица сети, так и другие состояния. Каждый аттрактор характеризуется *размером бассейна*. Бассейном аттрактора называется множество состояний сети, таких что при подаче любого из этих состояний на вход сети Хопфилда сеть в результате динамического процесса сходится к этому аттрактору. Узнать точное число аттракторов сети и размеры соответствующих бассейнов можно только перебрав множество всех возможных состояний сети (подавать каждое в качестве начального состояния и фиксировать аттрактор — достигну-

тое конечное состояние). Такой полный перебор потребовал бы слишком много времени на вычисления.

В лабораторном исследовании используется метод поиска аттракторов и оценивания относительного размера бассейнов, основанный на случайном поиске. Выбирается заданное число начальных состояний (случайным образом), и из каждого из них осуществляется релаксация к аттрактору. По результатам определяется число различных найденных аттракторов и относительное число попаданий в каждый из аттракторов (от общего числа начальных состояний), которое является оценкой относительного размера бассейна. Аттракторы, характеризующиеся большим значением относительного размера бассейна, считаются «более мощными» по сравнению с аттракторами, имеющими меньший бассейн («более слабые»).

Известно, что объем памяти сети Хопфилда, в том числе ложной, зависит от того, насколько коррелированы образы, по которым рассчитана сеть. Поэтому для удобства анализа результатов поиска аттракторов автоматически вычисляются коэффициенты корреляции образов (когда в сеть «записано» более одного образа).

Эксперимент «Ассоциативная память». В данном эксперименте требуется определить средний уровень искажения образа, подаваемого на вход сети, допускающий правильное восстановление исходного образа на выходе. Под искажением понимается инвертирование элементов вектора (замена –1 на 1 или наоборот), мерой искажения является расстояние Хемминга (число отличающихся элементов в исходном и искаженном векторах). Поскольку для ассоциативного восстановления разных векторов, по которым рассчитана сеть, допустимый уровень шума может быть существенно различным, требуется определить среднее допустимое искажение для образа, являющегося мощным аттрактором, и для образа — слабого аттрактора.

Эксперимент «Робастность». Требуется экспериментально проверить свойство робастности сети и определить максимальный уровень искажения синаптической матрицы (максимальную амплитуду шума), допускающий правильную работу ассоциативной памяти. Для определенности под правильной работой ассоциативной памяти в рамках данного эксперимента будем понимать способность сети правильно восстанавливать записанный образ с приблизительно 50% искажений от найденного максимально допустимого уровня в предыдущем эксперименте.

Искажение синаптической матрицы сети состоит в данной работе во внесении аддитивной помехи ξ:

$$W = W + \xi ,$$

где W — матрица коэффициентов связей (размерности N x N), ξ_{ij} — число, распределенное по равномерному закону на отрезке $[0, A]$, $i, j = 1, 2, \ldots, N$. В таком случае уровень искажения синаптической матрицы характеризуется параметром A, который далее в лабораторной работе будем называть *амплитудой* шума. При таких условиях в результате добавления помехи нарушается симметричность синаптической матрицы, и не выполняется условие $w_{ii} = 0$, $i = 1, 2, \ldots, N$ (равенство нулю элементов матрицы связей по главной диагонали). Программное обеспечение позволяет искусственно поддерживать выполнение этих условий одновременно или по отдельности.

Поскольку уровень аддитивной помехи задается ее амплитудой, в программе при визуализации искажения синаптической матрицы для общности матрица характеризуется так же, как и шум, своими минимальным и максимальным элементами. Разумеется, приведенная форма задания шума и условия эксперимента являются одними из возможных для наблюдения проявлений свойства робастности.

Варианты индивидуальных заданий

Индивидуальный вариант задания и стандартный файл с образами для экспериментов выдаются непосредственно на лабораторном занятии в соответствии с номером варианта (например, номер в списке учебной группы). Стандартное множество образов предназначено для использования в 1 - 3 пп. задания (*см. порядок выполнения работы*). Для п. 4 задания требуется с помощью редактора, входящего в состав программного обеспечения лабораторного занятия, создать новый файл с множеством образов, описанным в индивидуальном задании. Ниже приведен пример индивидуального варианта:

Создать сеть Хопфилда для каждой из 4–х групп образов, выполнить для каждой сети эксперименты, указанные в описании лабораторной работы (*см. порядок выполнения работы*):
1. Цифра «3»;
2. Цифры «1», «3», «4»;
3. Цифры «1», «2» , «3», «4», «5», «6», «7», «8», «9»;
4. 10 искаженных версий цифры «3».

Для сетей 1 – 3 использовать стандартное множество образов, для 4–й сети создать новое множество образов самостоятельно.

На рисунок 4.6 показано стандартное множество образов для этого варианта задания, на рис. 7 – один из возможных примеров множества образов для 4-го пункта задания.

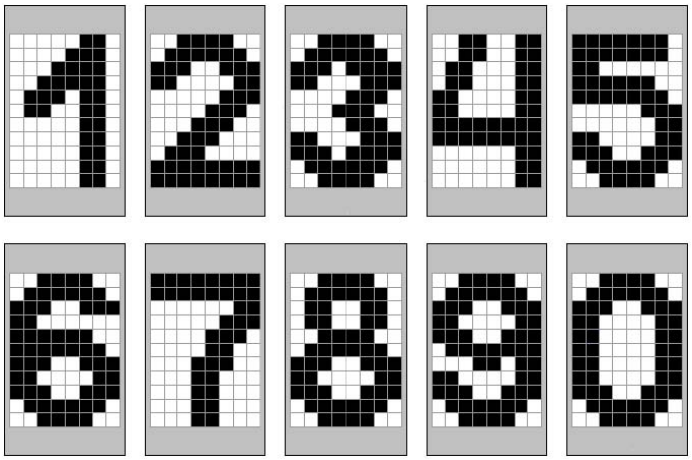


Рисунок 4.6 – Стандартное множество образов, упомянутое в приведенном примере варианта задания

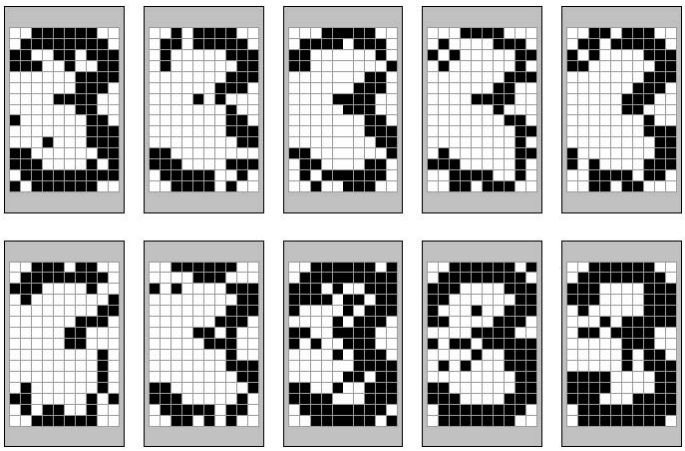


Рисунок 4.7 – 10 искаженных версий цифры «3», полученных в «Редакторе образов» с помощью автоматического внесения искажений

Порядок выполнения работы

Экспериментальные исследования выполняются в следующей последовательности:

1) Создать сеть Хопфилда для одного единственного образа, указанного в 1–м п. индивидуального варианта.

1.1) Провести исследование динамики рассчитанной сети, экспериментально определить:

• является ли устойчивым состоянием записанный в сеть образ (для этого подать на вход сети в качестве начального состояния записанный образ и сравнить начальное и конечное состояние сети);

• за сколько тактов в среднем сеть от случайного начального состояния сходится к устойчивому состоянию (провести 5 – 7 экспериментов).

1.2) Провести исследование аттракторов сети, экспериментально оценить:
• количество и тип (истинная/ложная память) устойчивых состояний сети;
• относительный размер бассейнов аттракторов.
Количество начальных случайных стимулов, подаваемых на вход сети при поиске аттракторов, взять равным 1500 – 2000.
1.3) Исследовать свойство сети – ассоциативную память, экспериментально определить:
• максимальный уровень искажения записанного в сеть образа, допускающий правильное распознавание этого образа сетью (расстояние Хэмминга между исходным и искаженным образами) на основании 5 – 7 экспериментов.
1.4) Исследовать робастность сети, экспериментально определить:
• максимальную амплитуду аддитивной помехи, вносимой в синаптическую матрицу, допускающую правильную работу сети в качестве ассоциативной памяти. Уровень искажения образа поддерживается при этом постоянным и равным приблизительно 50% от максимально допустимого, полученного в п. 1.3. (ассоциативная память).
2) Создать сеть Хопфилда для группы образов, указанной в п. 2 индивидуального варианта.
2.1) Как в п. 1.1. для всех записанных в сеть образов.
2.2) Как в п. 1.2.
2.3) Как в п. 1.3. для двух записанных в сеть образов (первый из образов должен являться более мощным аттрактором, второй – более слабым).
2.4) Как в п. 1.4. для двух записанных в сеть образов (первый из образов должен являться более мощным аттрактором, второй – более слабым, те же образы, что в п. 2.3.).
3) Создать сеть Хопфилда для группы образов, указанной в п. 3 индивидуального варианта.
3.1) Как в п. 1.1. для всех записанных в сеть образов.
3.2) Как в п. 1.2.
3.3) Как в п. 1.3. для двух записанных в сеть образов (первый из образов должен являться более мощным аттрактором, второй – более слабым).
3.4) Как в п. 1.4. для двух записанных в сеть образов (первый из образов должен являться более мощным аттрактором, второй – более слабым, те же образы, что в п. 3.3.).
4) Создать сеть Хопфилда для группы образов, указанной в п. 4 индивидуального варианта (несколько искаженных версий одного образа).
Провести исследование аттракторов сети (как в п. 1.2).

Описание программного обеспечения

Лабораторная работа выполняется с помощью пакета программ «Сеть Хопфилда», разработанного для среды MATLAB 5.0. Программа «Сеть Хопфилда» позволяет загружать из текстового файла специального формата множество образов, рассчитывать синаптическую матрицу сети по одному или нескольким образам текущего множества и, после того как сеть рассчитана, можно выполнять экспериментальные исследования (описанные в п. «Содержание работы» четыре эксперимента). Кроме того, можно сохранять в текстовом файле результаты выполненных экспериментов. Для подготовки новых наборов образов или редактирования имеющихся в программу «Сеть Хопфилда» включен специальный графический редактор. Программа запускается по команде

```
hopfnet
```

из командной строки системы MATLAB. При запуске программы появляется окно, показанное на рисунок 4.8. В выпадающем меню в нижней части стартового окна требуется выбрать номер индивидуального варианта, после чего нажать кнопку «OK».

В результате появляется главное окно программы, вид которого показан на рисунок 4.9. Непосредственно после запуска программы недоступен пункт «Эксперименты» главного меню и четыре кнопки в правой части окна (дублирующие подпункты меню «Эксперименты»), поскольку нейронная сеть Хопфилда пока не рассчитана.

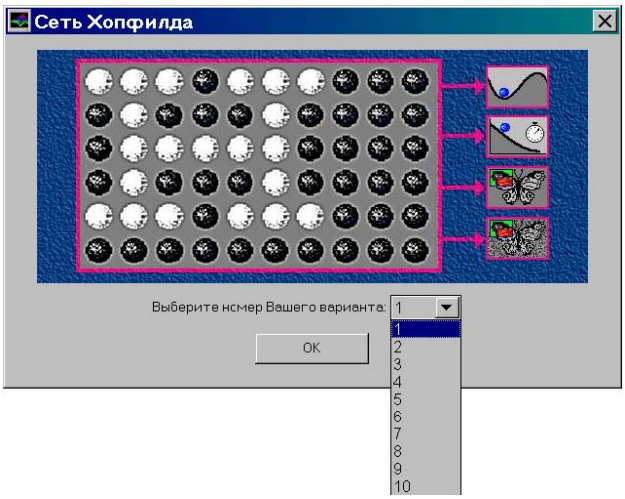


Рисунок 4.8. Это окно появляется при запуске программы «Сеть Хопфилда»

Кратко остановимся на содержании пунктов главного меню. Меню «Файл» позволяет сохранить результаты работы и выйти из программы. Меню «Параметры» включает два подпункта: «Образы» и «Сеть», которые дублируют одноименные кнопки в левой части окна. О содержании меню «Эксперименты» уже было сказано. Меню «Помощь» помимо информации о программе включает пункт «Задание», которому соответствует одноименная кнопка в нижней части главного окна.

При запуске программы «Сеть Хопфилда» загружается текст индивидуального задания с указанным номером и соответствующий варианту стандартный набор образов. Для того, чтобы просмотреть текст задания, нужно нажать на кнопку «Задание». Появляется окно, показанное на рисунок 4.10 (текст в окне отличается для разных вариантов).

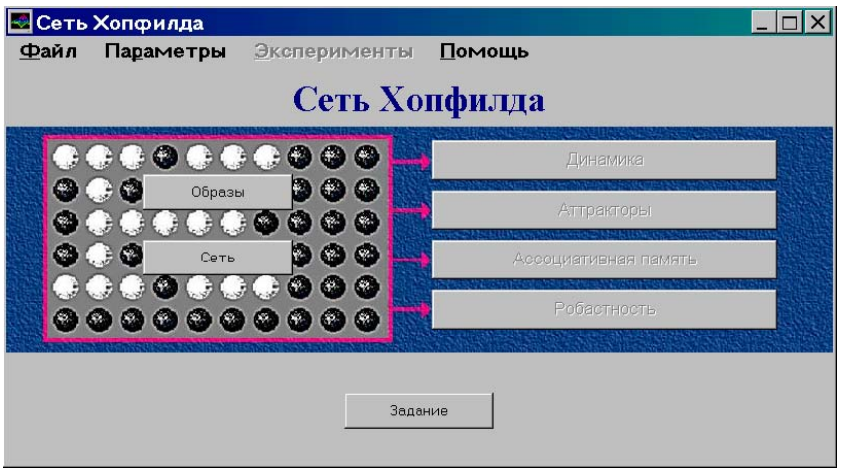


Рисунок 4.9 – Главное окно программы «Сеть Хопфилда» непосредственно после запуска

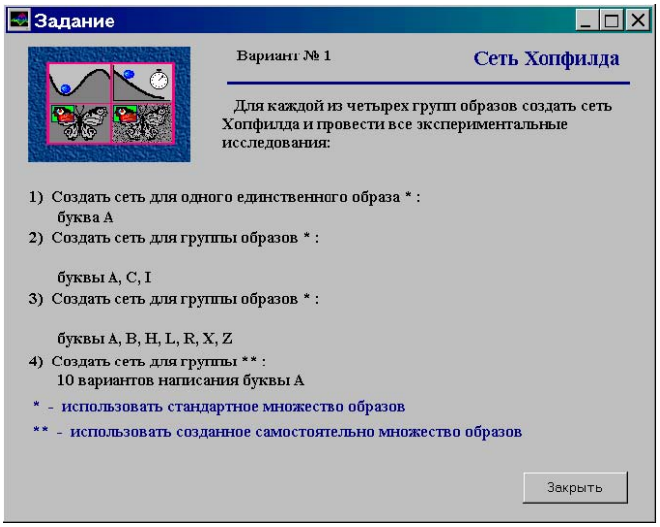


Рисунок 4.10 – Окно «Задание»

Для того, чтобы внести изменения в текущее множество образов или загрузить новое, нужно нажать на кнопку «Образы» главного окна или выбрать пункт «Образы» меню «Параметры». Появляется окно, показанное на рисунке 4.11. Центральная часть окна занята изображениями образов из текущего набора (в виде черно–белых диаграмм). Под каждым изображением выведено имя соответствующего образа. Однократное нажатие мыши на диаграмме устанавливает соответствующий образ текущим, пространство вокруг образа выделяется цветом. В нижней части окна «Образы» выводится информация о размерности образов и количестве образов в текущем наборе. Кнопка «Открыть файл» предназначена для загрузки нового множества образов из файла с расширением `pat`. Для того, чтобы вернуться к главному окну, загрузив новое множество образов, требуется нажать кнопку «Применить».

При нажатии на кнопку «Редактор» в окне «Образы», запускается программа «Редактор образов», вид главного окна которой показан на рисунке 4.12.

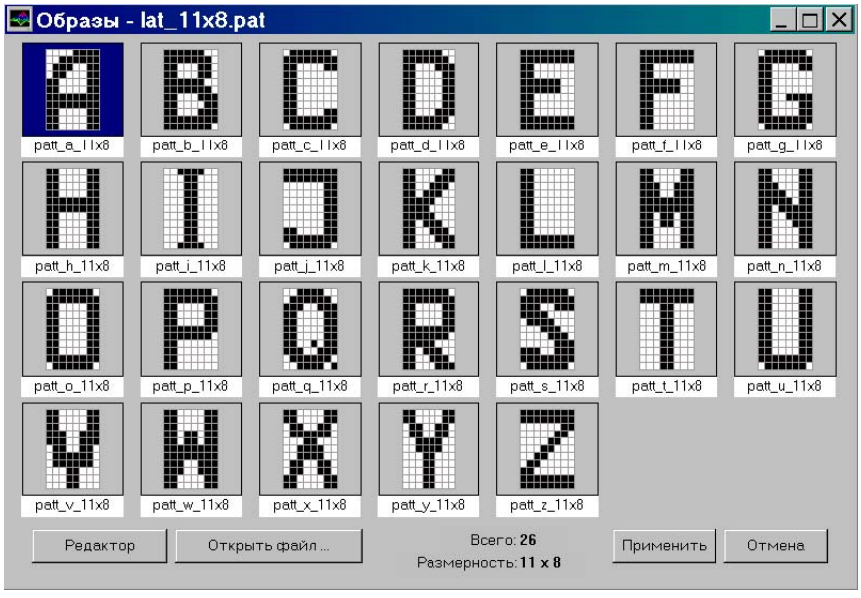


Рисунок 4.11 – Окно «Образы»

Программа «Редактор образов» является частью программного пакета «Сеть Хопфилда» и работает в среде MATLAB 5.0. Программа предназначена для создания и редактирования наборов биполярных образов, которые сохраняются в текстовых файлах специального формата. Все образы набора (множества) должны иметь одинаковую размерность, т. е. одинаковую длину вектора признаков и одинаковое число строк и столбцов для графического представления в виде черно–белой диаграммы.

Как видно на рисунке 4.12, один из образов набора изображен в центре экрана. Этот образ

называется «текущим». В результате однократного щелчка кнопки мыши на клетке диаграммы текущего образа цвет этой клетки устанавливается выбранным из «палитры» (белый или черный) в левом нижнем углу главного окна «Редактора образов». На рисунке 4.12 видно, что текущим цветом является черный. Для выбора цвета рисования нужно щелкнуть кнопкой мыши на черном или белом прямоугольнике палитры. Операции редактирования применяются к текущему образу. Для того, чтобы сделать текущим другой образ из набора используйте список в правой части окна. Заголовок окна начинается с имени файла из которого загружено множество образов. В нижней части экрана выводится информация о размерности и количестве образов в наборе.

Рассмотрим главное меню программы «Редактор образов». Меню «Файл» включает команды «Новый», «Открыть …», «Сохранить», «Сохранить как …» и «Выход». Команда «Новый» предназначена для создания нового множества образов. При этом требуется в специальном диалоговом окне задать размерность (число строк и столбцов) нового множества образов. Команда «Открыть …» используется для того, чтобы открыть имеющийся файл с множеством образов. Команды «Сохранить» и «Сохранить как …» предназначены для сохранения текущего множества образов под текущим и новым именем соответственно.

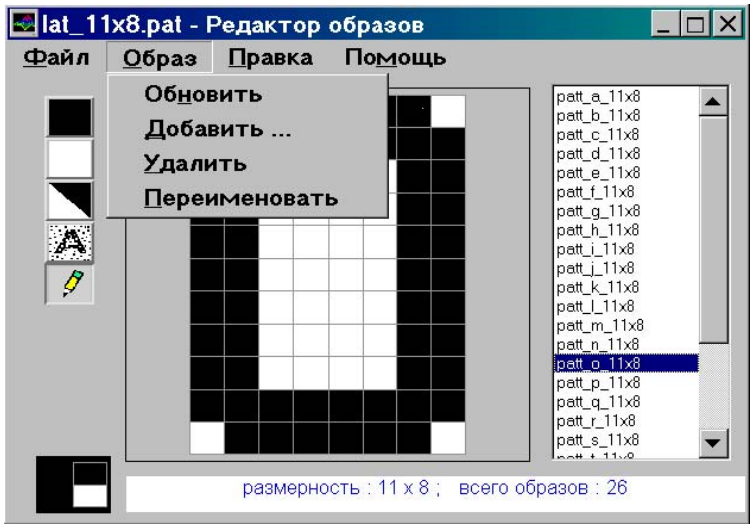


Рисунок 4.12 – Главное окно программы «Редактор образов»

Рисунок 4.13 – Окно «Зашумить»

Меню «Образ» (рисунок 4.12) главного окна программы «Редактор образов» содержит подпункты «Обновить», «Добавить …», «Удалить» и «Переименовать». Команда «Обновить» должна быть выполнена для сохранения результата редактирования перед сменой текущего образа. Команда «Добавить …» предназначена для добавления нового образа в набор. При этом в появляющемся специальном окне нужно задать имя добавляемого образа. Команда «Удалить» использует-

ся для удаления текущего образа из набора. С помощью команды «Переименовать» можно изменить имя текущего образа.

Меню «Правка» включает команды редактирования «Инвертировать», «Закрасить черным», «Закрасить белым», «Зашумить» (дублируются кнопками главного окна «Редактора образов», расположенными слева) и команды «Отменить», «Повторить» (результат последней выполненной по отношению к текущему образу команды редактирования). Команды редактирования меняют цвет клеток диаграммы с белого на черный или наоборот (и, соответственно, значения элементов вектора признаков с –1 на +1 или с +1 на –1). Команда инвертировать меняет цвет всех клеток диаграммы. Команды «Закрасить черным» и «Закрасить белым» устанавливают один и тот же цвет для всех клеток. По команде «Зашумить» появляется специальное окно, показанное на рисунке 4.13.

Окно «Зашумить» предназначено для автоматического внесения искажений в текущий образ. В левой верхней части окна изображается искаженный результат, на который будет заменен текущий образ «Редактора образов» после нажатия кнопки «Применить». В центре имеется шкала с ползунком, который управляется мышью, для указания уровня зашумления в процентах. Установленный процент искажения выводится слева от шкалы (на рисунке 4.13 уровень шума равен 25%). С помощью переключателя выбирается один из трех режимов искажения: изменить цвет, закрасить белым, закрасить черным (на рисунке 4.13 выбран режим «закрасить белым»). Автоматическое искажение выполняется по такому закону: каждая клетка диаграммы изменяется в соответствии с выбранным режимом с вероятностью, равной заданному уровню шума.

При выходе из «Редактора образов» появляются диалоговые окна с предложением сохранить в файле отредактированное множество образов и загрузить это множество в программу «Сеть Хопфилда».

Для создания (расчета) сети Хопфилда, над которой можно проводить экспериментальные исследования, предназначено окно «Сеть» (см. рисунок 4.14), которое вызывается по нажатии одноименной кнопки главного окна программы «Сеть Хопфилда» или по команде «Сеть» меню «Параметры». Основную часть окна занимают черно-белые изображения образов, слева имеется список имен всех образов текущего множества.

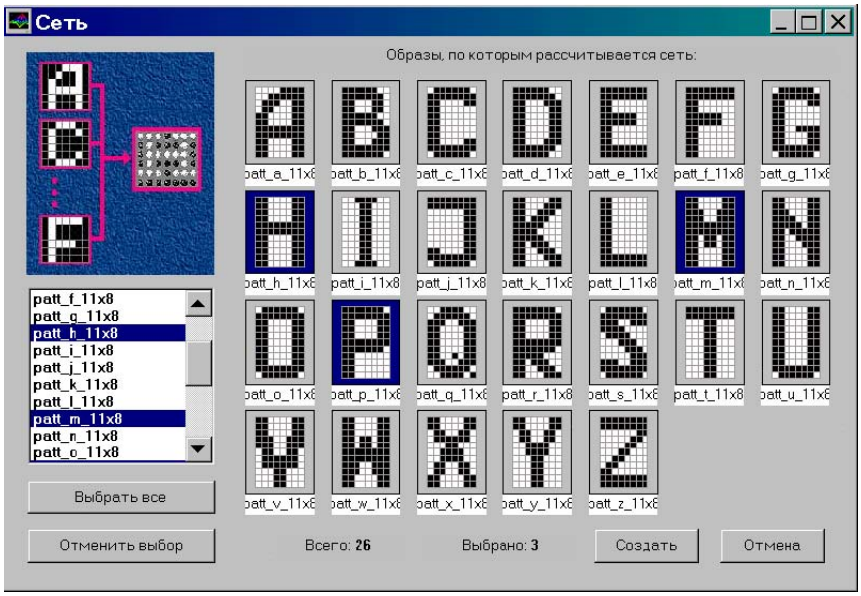


Рисунок 4.14 – Окно «Сеть»

Образ выделяется цветом фона при нажатии мыши на изображении или при указании в списке. Общее число образов и число выделенных выводится внизу окна. Для расчета сети нужно выделить нужные образы и нажать кнопку «Создать». В окне, изображенном на рисунке 4.14, выбраны три образа, по которым будет рассчитана синаптическая матрица сети. Последующие иллюстрации к описанию программы «Сеть Хопфилда» соответствуют именно этой сети.

На рисунке 4.15 показано окно эксперимента «Динамика» сразу после открытия. Для выполнения эксперимента нужно задать начальное состояние сети и выбрать режим просмотра, после чего нажать кнопку «Пуск». Начальное состояние может быть либо случайным вектором, сге-

нерированным автоматически, либо одним из тех векторов, по которым рассчитана сеть (способ задания начального состояния нужно указать с помощью переключателя). Предусмотрено три режима просмотра: непрерывный, пауза после каждого такта, пауза после обработки каждого нейрона. Вектор начального состояния сети можно редактировать щелчком мыши на клетках диаграммы (в режиме инвертирования). Как видно на рисунке, заданному начальному состоянию соответствует значение энергии сети 0,15909. На рис. 16 показано окно «Динамика» после завершения переходного процесса в сети Хопфилда. В центре изображено конечное состояние сети, число тактов (равно трем), за которое оно было достигнуто, соответствующие значение энергетического функционала (–80,5) и график зависимости энергии сети от времени. На графике по вертикальной оси отложено значение энергии, а по горизонтальной — номер обработанного нейрона. Для того, чтобы снова наблюдать переходный процесс, нужно нажать кнопку «Повторить эксперимент».

На рисунке 4.17 представлено окно эксперимента «Аттракторы». Поскольку сеть рассчитана более, чем по одному образу (по трем образам, соответствующим буквам латинского алфавита «Н», «М» и «Р»), имеет смысл при анализе результатов эксперимента учитывать корреляцию образов.

В левой части окна для пары образов, выбранной из списка отображается коэффициент корреляции. Единственным начальным условием эксперимента «Аттракторы» является число начальных состояний, из которых осуществляется поиск. Поиск начинается по нажатии кнопки «Пуск». В программе предусмотрено средство для принудительной остановки процесса поиска.

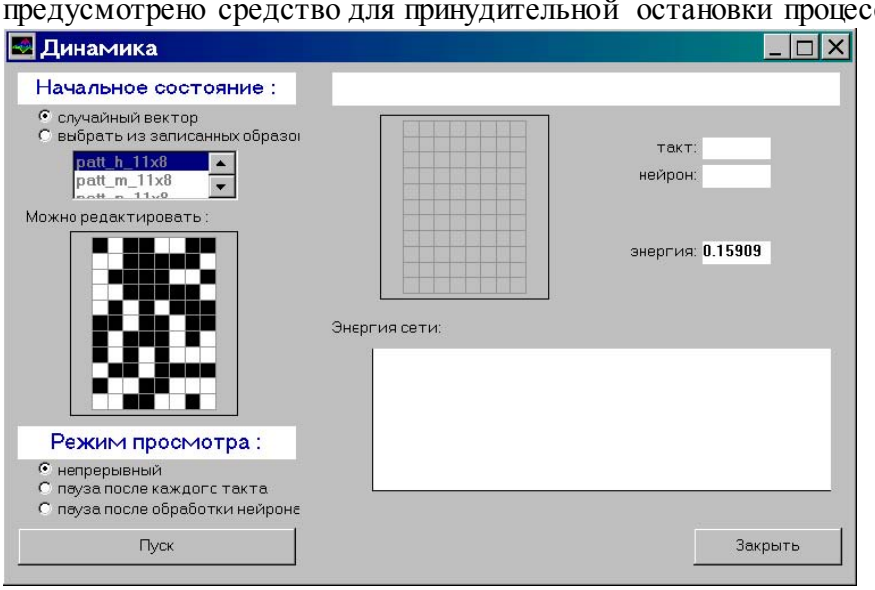


Рисунок 4.15 – Окно «Динамика»

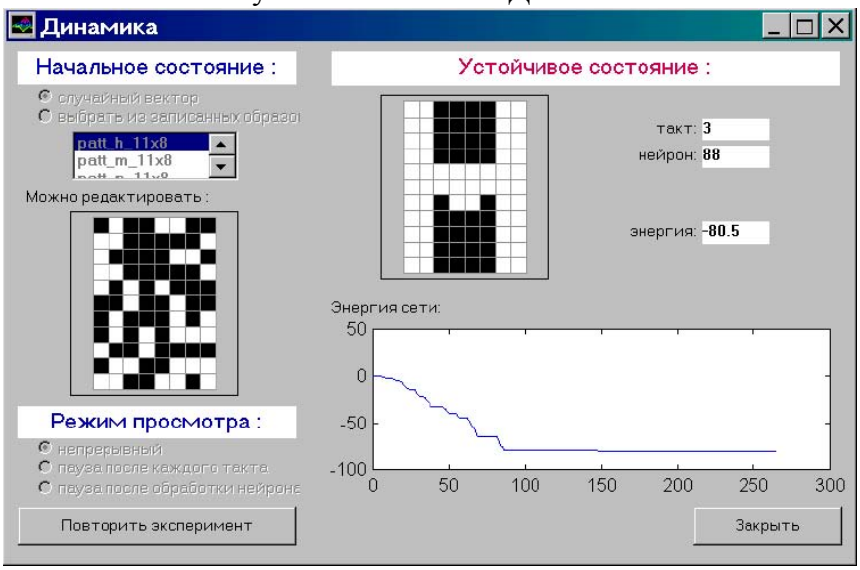


Рисунок 4.16 – Окно «Динамика» после завершения переходного процесса в сети Хопфилда

На рисунке 4.18 показан результат поиска аттракторов. Вверху окна отображено число найденных устойчивых состояний и фактическое число начальных состояний, которые проверялись во время поиска. В центре окна находится круговая диаграмма оценки относительных размеров бассейнов. Синим цветом выделен сектор, соответствующий аттрактору, изображенному справа от круговой диаграммы и выделенному в списке справа. Под изображением аттрактора выведена оценка относительного размера бассейна с точностью до двух знаков после запятой. Для того, чтобы просмотреть другой найденный аттрактор, нудно либо один раз щелкнуть мышью на изображении одного из секторов круговой диаграммы, закрашенных серым цветом, либо выбрать другой пункт в списке найденных аттракторов. Найденные аттракторы можно сохранить в текстовом файле на диске в формате набора образов с расширением `pat`. Для этого предназначена кнопка «Сохранить аттракторы …». В появляющемся диалоговом окне можно выбрать один из трех вариантов сохранения: добавить к текущему множеству образов, добавить к другому множеству образов, сохранить в отдельном файле.

Окно эксперимента «Ассоциативная память» показано на рисунке 4.19. Начальным условием эксперимента является начальное состояние сети, которое задается следующим образом. Сначала нужно один из образов выбрать в списке записанных в сеть (в левой части окна). Этот образ изображается на диаграмме в центре окна. Это изображение можно редактировать мышью в режиме инвертирования и с использованием кнопок, расположенных над диаграммой.

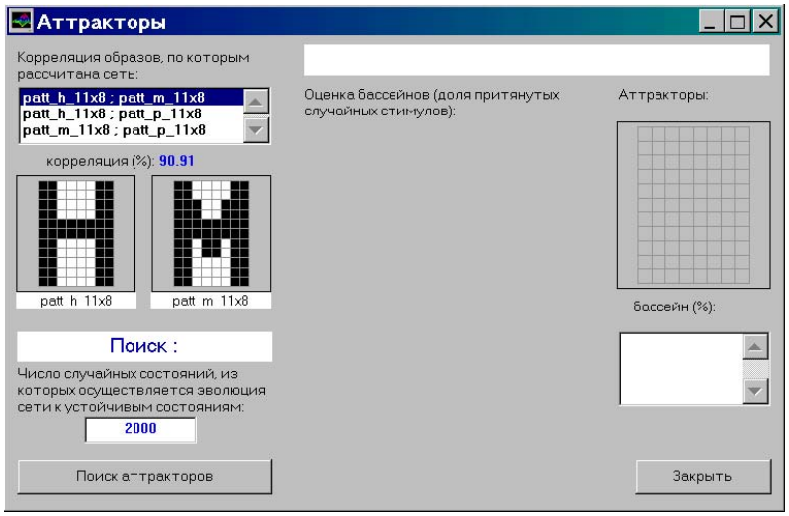


Рисунок 4.17 – Окно «Аттракторы»

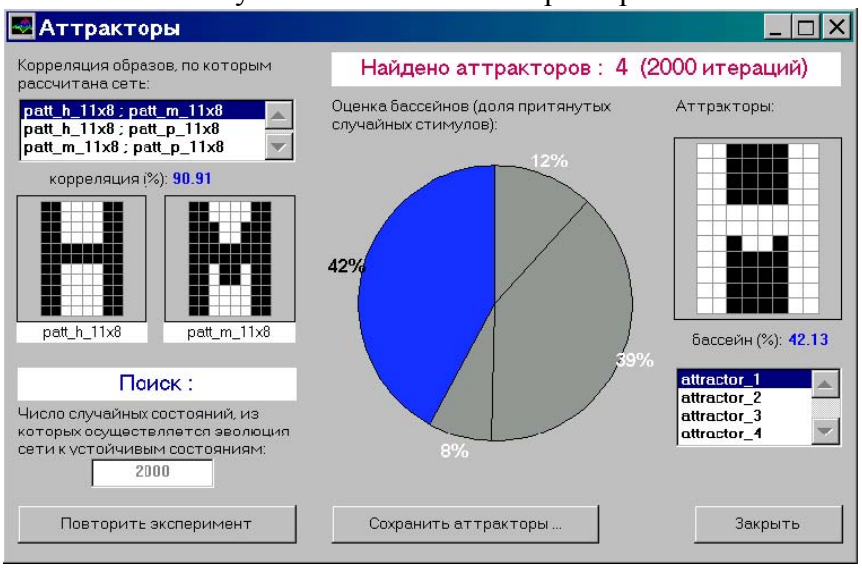


Рисунок 4.18 – Окно «Аттракторы». Результаты поиска

Эти кнопки функционально эквивалентны кнопкам «Редактора образов». По мере изменения исходного образа пересчитывается и выводится в нижней части окна расстояние Хемминга между исходным вектором и текущим состоянием (на рисунке 4.19 оно равно 16). Когда начальное состояние задано, следует нажать кнопку «Пуск». На рисунке 4.20 показан вид окна после

срабатывания ассоциативной памяти. В правой части окна изображено конечное состояние сети. Видно, что оно совпадает с исходным образом. Это считается правильным распознаванием и диаграмма конечного состояния выделяется зеленым цветом фона. На рисунке 4.21 показан результат другого эксперимента с тем же исходным образом. Видно, что начальное состояние сильнее отличается от исходного образа (расстояние Хемминга равно 28). Конечное состояние не совпадает с исходным вектором. Неправильность распознавания образа подчеркивается красным цветом фона диаграммы конечного состояния сети.

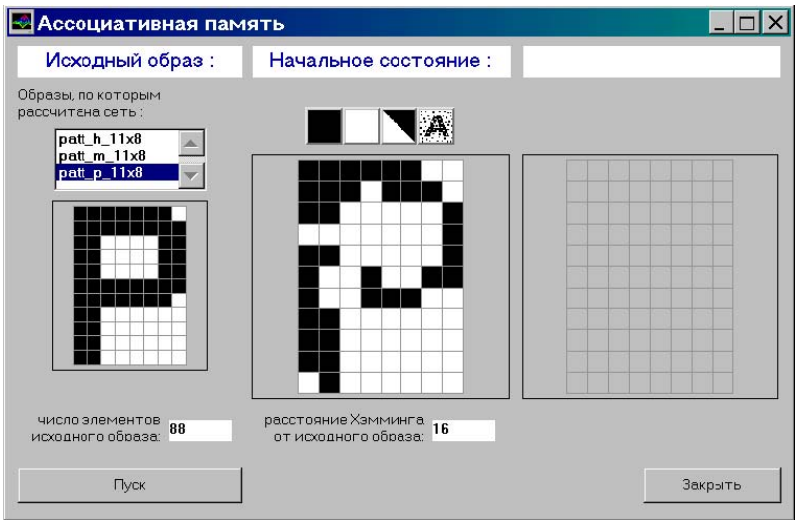


Рисунок 4.19 – Окно «Ассоциативная память»

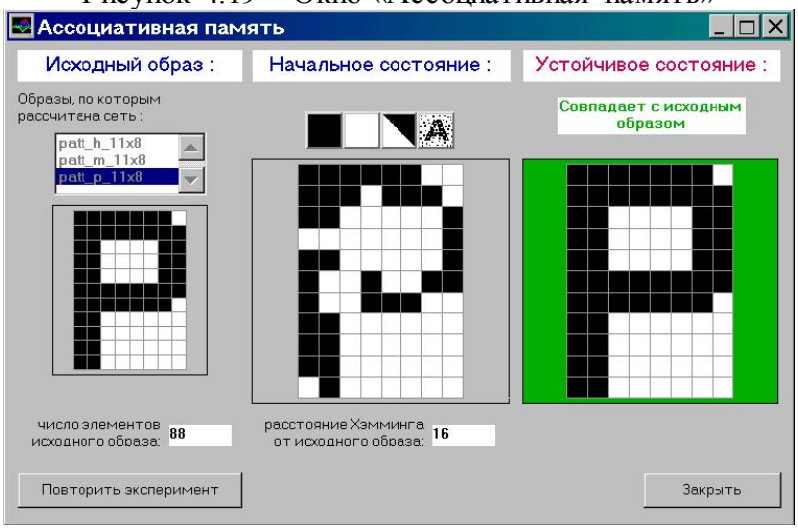


Рисунок 4.20 – Окно «Ассоциативная память». Правильное распознавание

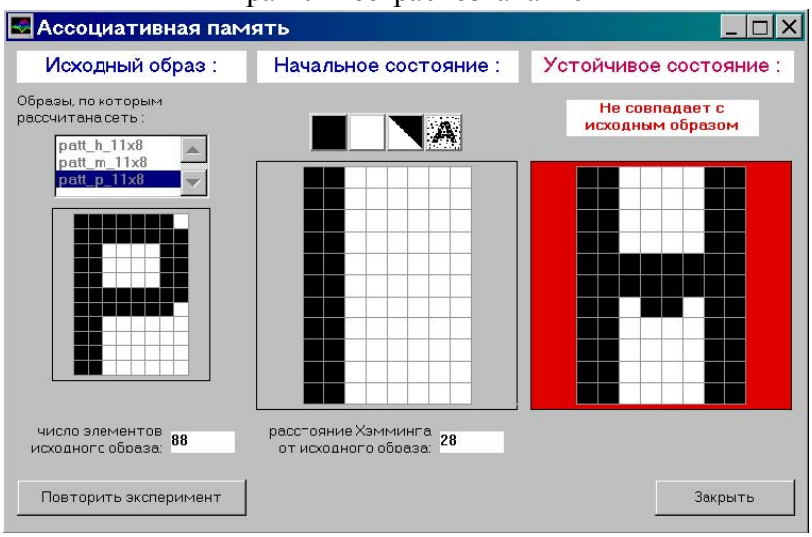


Рисунок 4.21 – Окно «Ассоциативная память».

Неправильное распознавание

На рисунке 4.22 показано окно эксперимента «Робастность». Правая часть окна почти совпадает функционально с элементами окна «Ассоциативная память». Точно так же, как и в предыдущем эксперименте, нужно выбрать исходный образ из списка и задать начальное состояние сети, зашумив исходный образ. Левая и центральная часть окна предназначена для задания параметров автоматического искажения синаптической матрицы нейронной сети. Итак, начальными условиями эксперимента являются входной образ и текущая матрица коэффициентов связей. Для искажения синаптической матрицы нужно выполнить следующие действия: задать амплитуду шума (в редактируемой строке), выставить находящийся рядом флаг «Применить», выставить желаемым образом флаги «сохранять симметричность» и «сохранять нули по главной диагонали», нажать кнопку «Зашумить исходную». Результаты этих действий изображаются на условной диаграмме в виде прямоугольников. Вид окна «Робастность» с заданными условиями эксперимента приведен на рисунке 4.23. Теперь можно нажать кнопку «Пуск». Результат срабатывания сети выводится в правом нижнем углу окна и интерпретируется аналогично эксперименту «Ассоциативная память». На рисунке 4.24 и 4.25 показаны результаты распознавания для различных начальных состояний.

Начальные условия и результаты всех экспериментов для текущей сети сохраняются в процессе работы с программой. Эти данные могут быть записаны в текстовый файл на диске по команде «Сохранить последний результат» или «Сохранить все результаты» меню «Файл» главного окна. В первом случае в файл записывается последний результат каждого из четырех экспериментов.

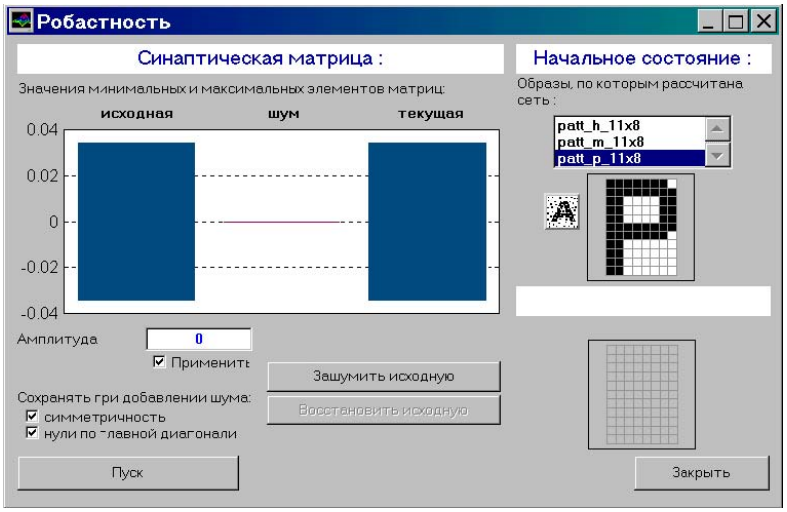


Рисунок 4.22 – Окно «Робастность»

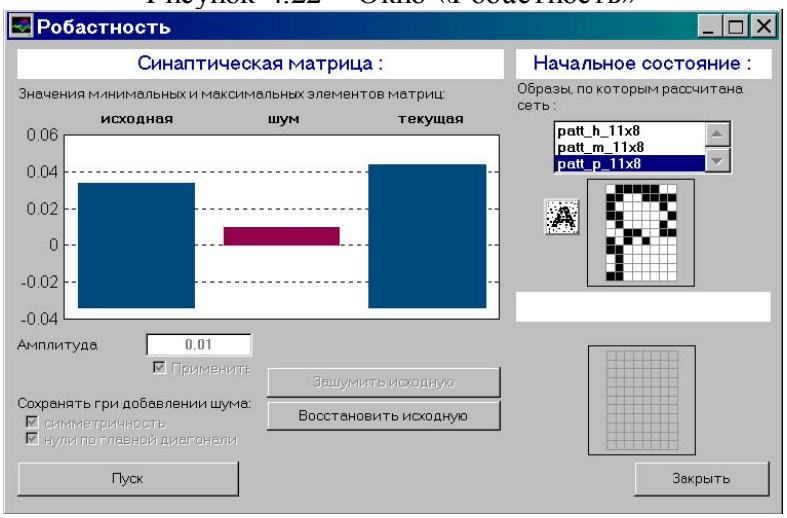


Рисунок 4.23 – Окно «Робастность». Начальные условия эксперимента

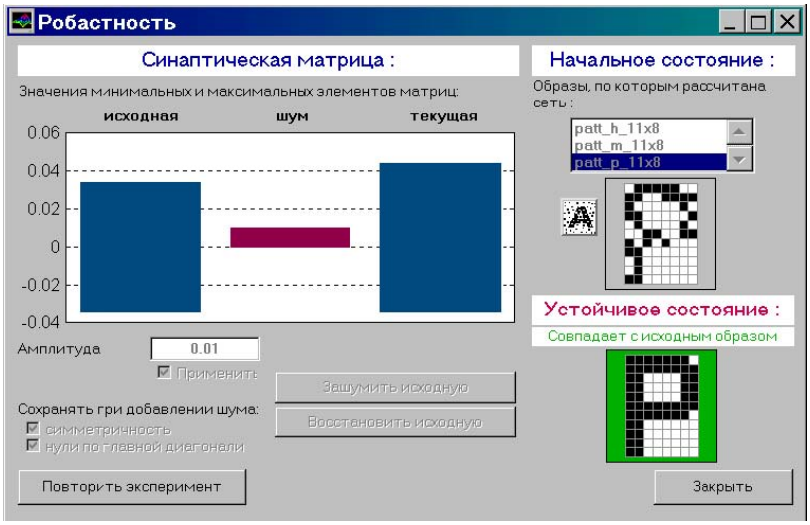


Рисунок 4.24 – Окно «Робастность». Правильное распознавание

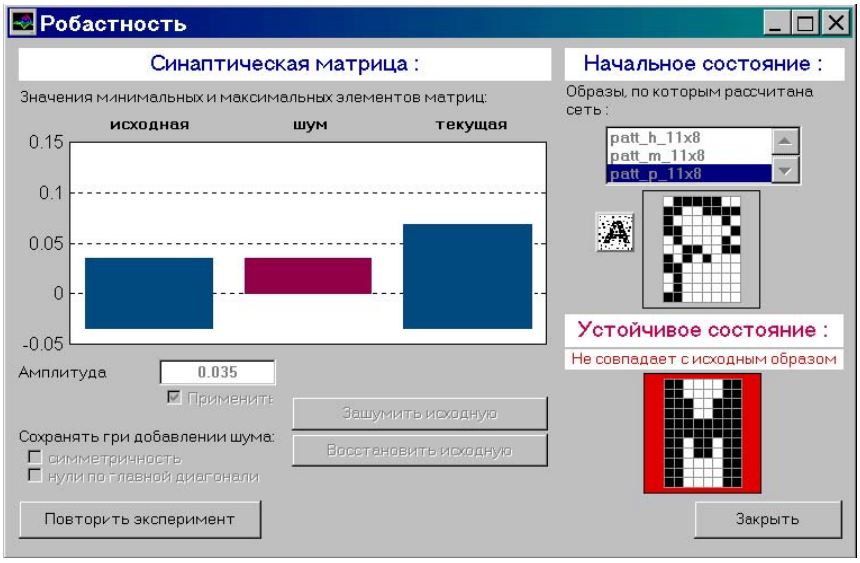


Рисунок 4.25 – Окно «Робастность». Неправильное распознавание

Содержание отчета

1. Название лабораторной работы, цель и программа работы.
2. Для каждой сети — список образов, на основании которых она рассчитана.
3. Для каждой сети – начальные условия и результаты всех экспериментов, указанных в п. «Порядок выполнения работы»
4. Анализ полученных результатов (ответить на вопрос, почему для четырех исследованных сетей в одних и тех же экспериментах получены разные результаты).

Контрольные вопросы к защите лабораторной работы

1. Что называется автоассоциативной памятью? Приведите пример.
2. Что называется аттрактором динамической системы? Объясните принцип применения динамических систем с множеством аттракторов для построения ассоциативной памяти.
3. Напишите уравнения динамики сети Хопфилда.
4. Какова активационная характеристика нейронов сети Хопфилда?
5. Чему равно начальное состояние нейронов сети Хопфилда?
6. Объясните различие синхронного и асинхронного режимов функционирования рекуррентной нейронной сети. Какой из режимов функционирования используется в сети Хопфилда?
7. Каким выражением определен энергетический функционал в процессе работы сети Хопфилда?
8. Почему время достижения сетью Хопфилда одного из аттракторов из произвольного начального состояния конечно?
9. Как рассчитывается матрица синаптических коэффициентов сети Хопфилда? Какими свойствами она обладает?

10. Как приближенно оценивается объем памяти сети Хопфилда?
11. На какие типы можно разделить множество аттракторов сети Хопфилда? Что такое “ложная память”?
12. Как применяется сеть Хопфилда для кластеризации данных? Как интерпретируются кластеры?
13. Как устанавливается класс принадлежности вектора признаков при использовании нескольких (по числу классов) сетей Хопфилда, “настроенных” на разные классы?
14. Как организован в лабораторной работе процесс оценивания размеров бассейнов аттракторов сети Хопфилда?

Лабораторная работа № 5 Изучение свойств линейного нейрона и линейной нейронной сети

1 ЦЕЛЬ РАБОТЫ

Изучить свойства линейного нейрона и линейной нейронной сети.

2 СВЕДЕНИЯ ИЗ ТЕОРИИ

2.1 Модель нейрона

Искусственные нейронные сети (НС) представляют собой простейшие математические модели мозга. Понять основные принципы построения НС можно, рассматривая их как совокупность (сеть) отдельных структур (нейронов). Очень грубо структуру биологического нейрона можно описать следующим образом. Нейрон имеет сому – тело, дерево входов – дендриты, выход – аксон. На соме и на дендритах располагаются окончания аксонов других нейронов, называемых синапсами. Принятые синапсами входные сигналы стремятся либо возбудить нейрон, либо затормозить. Когда суммарное возбуждение достигает некоторого порога, нейрон возбуждается и посылает по аксону сигнал другим нейронам. Каждый синапс обладает уникальной синаптической силой, которая пропорционально своему значению изменяет передаваемый на нейрон входной сигнал. В соответствии с приведенным описанием математическая модель нейрона представляет собой суммирующий пороговый элемент (рис. 1).

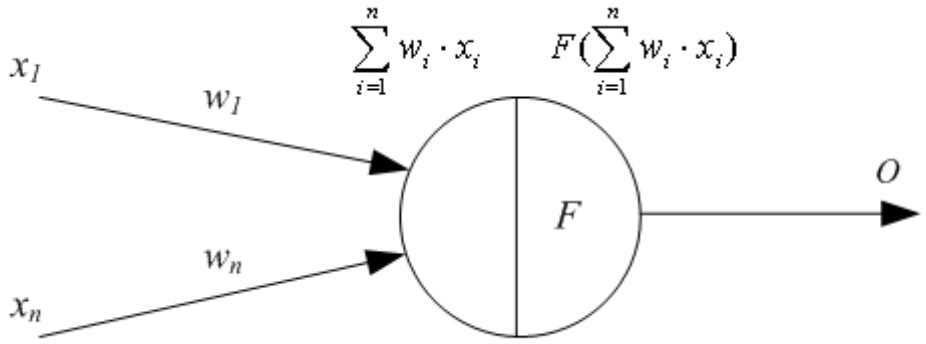


Рис. 1. Искусственный нейрон

Формула срабатывания нейрона:

$$O = F(< W^T , X >) = F(\sum_{i=1}^{n} w_i \cdot x_i),$$

$$O = \begin{cases} 1, (< W^T , X >) \geq 0, \\ 0, \textit{иначе.} \end{cases}$$

2.2 Алгоритм обучения по дельта-правилу

Обучение НС происходит на некоторой обучающей выборке, для каждого образца которой определяются и сравниваются с желаемыми значениями все текущие выходы. Если разница недопустима, то веса изменяются. Окончанием обучения считается ситуация, когда общая ошибка на всех образцах допустима.

Все алгоритмы обучения нейросетей являются разновидностями алгоритма обучения по методу коррекции ошибки, которая осуществляется по-разному. Идея изменения весов НС сводится к нахождению общей меры качества сети, в качестве которой обычно выбирают функцию ошибки сети. Тогда, чтобы подобрать нужные веса, необходимо минимизировать функцию ошибки. Самым распространенным методом поиска минимума является метод градиентного спуска. Для случая функции с одной переменной веса изменяются в направлении, противоположном производной, т. е. справедлива формула

$$W^{n+1} = W^n - h \cdot F'(W),$$

где h – некоторый уровень обучения, шаг изменения;
$F'(W)$ – производная функции качества НС для одной переменной.

Для функции F от n переменных и единичного вектора e в пространстве R^n $||e|| = 1$, $e \in R^n$, дифференциал выражается формулой

$$\partial_e F(W) = \frac{\lim_{t\to 0}(F(W+et) - F(W))}{t}.$$

Для случая $e = (0, 0...1...0)$ определим частный дифференциал

$$\partial_i F(W) = \frac{\lim_{t\to 0}(F(w_1, w_2, ..., w_{i+et}, ..., w_n) - F(W))}{t}.$$

Таким образом, антиградиент – это набор следующих дифференциалов:

$$\partial F(W) = ((-\partial F(w_1), -\partial F(w_2), ..., -\partial F(w_i), ..., -\partial F(w_n))^T.$$

Для определения обобщенной функции ошибки рассмотрим обучающую выборку $\{(x^k, y^k)\}$, где $k = 1, ..., K$. Накопленная по всем эпохам ошибка

$$E = \sum_{k=1}^{K}(E^k) = \sum_{k=1}^{K}\left(\sum_{i=1}^{m} 1/2\left\|O_i - Y_i\right\|^2\right).$$

Формула модификации весов НС

$$W^{n+1} = W^n - h \cdot \partial E / \partial W$$

уточняется для различных видов функции активации. Для линейной функции $F(t) = t$, НС формирует каждый выход как скалярное произведение весов на вектор входов: $O_i = < W_i, X_i >$ и градиент будет равен:

$$\partial E / \partial W = -(Y_i - O_i) \cdot X,$$

где Y_i – желаемый выход; O_i – полученный выход; X – вектор выхода.

Таким образом, получаем формулу изменения весов

$$W^{n+1} = W^n - h \cdot (Y_i - O_i) \cdot X.$$

Если значением δ назвать разницу $(Y_i - O_i)$, то получим формулу

$$W^{n+1} = W^n - h \cdot \delta \cdot X,$$

что является *алгоритмом обучения по* δ*-правилу*.

2.3 Описание основных функций

Для работы с нейронными сетями необходимо инсталлировать MATLAB и обладать первоначальными знаниями относительно языка системы. Поскольку MATLAB представляет собой интерпретатор, то обучение инструментарию нейронных сетей заключается в основном в изучении функций и их параметров. Узнать возможности нейрона как классификатора простых линейно сепарабельных задач можно путем проведения экспериментов с моделью одного линейного нейрона.

Функция *newp*. Для того чтобы создать нейрон, используют функцию newp, имеющую следующий синтаксис:

$$net = newp(PR, S, TF, LF),$$

где PR – матрица минимальных и максимальных R входных элементов; S – количество нейронов (при создании одного нейрона S=1); TF – функция активации (transfer function); LF – имя функции обучения нейрона.

В случае если параметры функции *newp* не заданы, их значения определяются посредством ввода значений в диалоговые окна. Построенный нейрон характеризуется функциями весов (weight function), входов сети (net input function) и определенной функцией активации. Функция весов – это умножение весов на входной сигнал, функция входов сети – их сумма. Веса задаются как для входов нейрона, так и для фиксированного входа, задающего порог срабатывания (bias). Вектор весов инициализируется нулями. Для обучения используются функции, рассмотренные ниже.

Функция *learnp* настраивает веса нейрона. Синтаксис функции обучения довольно сложен:

$$[dW, LS] = learnp(W, P, Z, N, A, T, E, gW, gA, D, LP, LS),$$
$$[db, LS] = learnp(b, ones(1, Q), Z, N, A, T, E, gW, gA, D, LP, LS),$$
$$info = learnp(code).$$

Функция *learnp (W, P, Z, N, A, T, E, gW, gA, D, LP, LS)* имеет несколько входов, где вектор W – вектор весов; P – вектор входов; Z – вектор взвешенных входов; N – вектор сети; A – вектор выхода; T – вектор желаемых выходов; E – вектор ошибок; gW – вектор изменения весов; gA – изменения выходов. Функция возвращает значения: dW – изменения матрицы весов; LS – новый уровень обученности.

Функция *learnp* может быть использована с параметрами по умолчанию:

$$dW = learnp([\,], p, [\,], [\,], [\,], e, [\,], [\,], [\,], [\,]).$$

Использование пустого списка [ ] означает параметр по умолчанию.

Функция *learnp* вычисляет изменение весов dW для заданного нейрона в соответствии с правилом обучения персептрона:

$$dW = \begin{cases} 0, \textit{если ошибка } e = 0, \\ p', \textit{если } e = 1, \\ -p', \textit{если } e = -1, \end{cases}$$

т. е. $dW = e \cdot p'$.

Функция *learnpn* настраивает нормализованные веса:

$$[dW, LS] = learnpn(W, P, Z, N, A, T, E, gW, gA, D, LP, LS).$$

Функция *learnpn* вычисляет изменение весов *dW* для данного нейрона и его входа *P* и ошибки *E* в соответствии с нормализованным правилом обучения персептрона:

$$pn = p / \sqrt{(1 + p(1)^2 + p(2)^2 + ... + p(R)^2)},$$

$$dW = \begin{cases} 0, \textit{если ошибка } e = 0, \\ pn', \textit{если } e = 1, \\ -pn', \textit{если } e = -1, \end{cases}$$

т. е. $dW = e \cdot pn'$.

Линейный нейрон имеет одно существенное ограничение. Входные векторы должны быть линейно разделимы. Если векторы невозможно отделить прямой или гиперплоскостью, то персептрон не способен решить задачу классификации.

Функция *adapt* адаптирует НС к условиям задачи:

$$[net, Y, E, Pf, Af] = adapt(net, P, T, P_i, A_i).$$

Параметры функции *adapt*: *net* – имя сети; *P* – входы сети; *T* – желаемый выход; P_i – исходные условия задержки; A_i – исходные условия задержки для слоя. Функция возвращает параметры адаптированной сети *net.adaptParam*: *net* – измененная сеть; *Y* – выход сети; *E* – ошибки сети; *Pf* – условия задержки входов; *Af* – условия задержки слоя. Параметры P_i и *Pf* необязательные и необходимы только для сетей, имеющих задержки на входах и слое.

Функция *train* также обучает НС и использует следующий синтаксис:

$$[net, tr] = train(net, P, T, P_i, A_i).$$

Функция *train* имеет следующие параметры: *net* – сеть; *P* – входы сети; *T* – желаемый выход; P_i – исходные условия задержки входа; A_i – исходные условия задержки слоя.

Функция *sim* имитирует нейронную сеть:

$$[Y, Pf, Af] = sim(net, P, P_i, A_i),$$

где *net* – сеть; *P* – входы сети; P_i — исходные условия задержки входов сети; A_i – исходные условия задержки слоя. Функция возвращает *Y* – выходы сети; *Pf* – окончательные условия задержки входов; *Af* – окончательные условия задержки слоя.

Функции активации. Ниже представлены назначения этих функций.

Функция	Назначение
hardlim	Возвращает 1, если на входе положительное число и 0 в противном слу-

	чае.
tansig	Вычисляет гиперболический тангенс от входа.
purelin	Вычисляет выход слоя от сетевого входа.

Функции графического интерфейса и вспомогательные функции. Назначение этих функций представлено ниже.

Функция	Назначение
axis([X_{min} X_{max} Y_{min} Y_{max})	Устанавливает диапазоны координатных осей.
title('строка')	Выводит в графическое окно рисунков заголовок графика.
rand(M, N)	Возвращает матрицу размерности *M* на *N* со случайными значениями.
xlabel('строка') *ylabel('строка')*	Подписывают наименование координатных осей.
cla reset	Очищает координатную сетку в окне рисунков.
hold on *hold off*	Включают и отключают режим добавления графиков на координатную сетку.
text(X, Y, 'строка')	Выводит строку, начиная с указанных координат в поле рисунков.
pause (n)	Ожидает пользовательского ответа *n* секунд.
plot(X, Y, 'цвет и символ')	Изображает на координатной сетке точки с координатами, заданными векторами *X, Y,* с помощью указанного символа и цвета.
plotpv(P, V)	Изображает точки *P* указанными маркерами *T,* где *P* – матрица входных векторов размерностью *R* на *Q* (*R* должен быть 3 или меньше), *T* – матрица двоичных векторов размерностью 5 на *Q (S* должен быть 3 или меньше).
plotes (WV, BV, ES, V)	Изображает поверхность ошибки на отдельном входе, где *WV* – вектор строк значений весов *W* размерности *N, BV* – вектор строк значений порогов *B* размерности *M, ES* – матрица ошибки размерности *M* на *N, V* – угол зрения по умолчанию [-37, 5, 30].
plotsom(POS)	Изображает позицию нейрона красной точкой, связывая синей линией нейроны, находящиеся друг от друга на расстоянии 1. *POS* – матрица *S N*-размерных нейронов.
ind2vec *vec2ind*	Позволяют представить индексы либо собственно значениями индексов, либо векторами, строки которых содержат 1 в позиции индекса.
full	Преобразует разреженную матрицу в полную.
maxlinlr(P)	Функция возвращает максимальный уровень обученности линейного слоя без *bias,* который обучался только на векторе *P.*
trainlm	Выполняет обучение многослойной НС методом Левенберга-Марквардта.
netprod	Входная сетевая функция, которая вычисляет выход сетевого слоя, умножая входной вектор на веса и прибавляя *bias.*
init	Итеративно инициализирует НС.

Структура данных описания нейронных сетей. Структура данных *net* – это описание обученной НС. Обучение осуществляется в соответствии со следующими параметрами, значения которых либо устанавливаются пользователем, либо по умолчанию.

Структура данных	Комментарий
net.trainParam. epochs 100	Максимальное количество эпох обучения.
net. trainParam.goal 0	Целевое значение ошибки.
net.trainParam.max_fail 5	Максимальное значение ошибки.
net.trainParam.mem reduc 1	Фактор оптимизации процесса обучения: оптимизация использования памяти или времени процессора.

net. trainParam. min_grad 1e-10	Минимальное значение градиента.
net.trainParam.show 25	Количество эпох между показами.
net.trainParam.time inf	Максимальное время обучения в секундах.
TR	Структура данных, содержащая значения об обученности НС в текущую эпоху.
TR. epoch	Номер эпохи.
TR.perf	Уровень обученности *(Trainingperformance).*
TR.vperf	Степень качества *(Validation performance).*
TR.tperf	Результативность обработки теста *(Testperformance).*
TR.mu	Значение адаптивности.

Структура данных описания адаптированной НС *net.adaptfcn* включает в себя следующие поля *net.adapt.param: NET* – адаптированная НС; *Y* – выходы НС; *E* – ошибки НС; *Pf* – окончательные входные значения задержек; *Af* – окончательные выходные задержки; *TR* – результат обучения (эпохи и целевая ошибка). Проведем в среде *Matlab toolbox* эксперименты, используя рассмотренные функции.

3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

ПРИМЕР 1 Создание нейронов, реализующих функции логического И и логического ИЛИ

Создадим нейрон с одним двухэлементным входом (интервалы первого и второго элементов [0; 1]). Определим два первых параметра функции *newp*, а в качестве значений третьего и четвертого параметра (типа функции активации и имени процедуры обучения) воспользуемся значениями по умолчанию.

```
% создание нейрона с одним двухэлементным входом (интервал
% первого элемента [0; 1] и интервал второго элемента [-2; 2]
net = newp([0 1; -2 2], 1);
```

Для того чтобы исследовать поведение нейрона, необходимо имитировать его работу с помощью функции *sim*. Для определения последовательности значений входа создадим последовательность *P1*.

```
% создание последовательности значений входа
P1 = {[0; 0] [0; 1] [1; 0] [1; 1]};
% имитация работы нейрона net на последовательности входов P
% желаемых выходов - T1, которая позволит нам провести адаптацию
% нейрона (обучить его) через 20 проходов.
Y = sim (net, P1);
% создание последовательности выходов
T1 = {0, 0, 0, 1};
% установка количества проходов (циклов) адаптации
net.adaptParam.passes = 20;
% адаптация нейрона net для обучающей выборки <P1; T1>
net = adapt (net, P1, T1);
% симуляция работы нейрона net на последовательности входов  P1
Y = sim (net, P1);
```

В результате мы получим нейрон, выполняющий функцию логического И.

Для переобучения нейрона на выполнение функции ИЛИ переопределим входы *P* и выходы *T*.

```
% создание последовательности входов
P2 = [0 0 1 1; 0 1 0 1];
% создание последовательности выходов (реакций) для нейрона,
% выполняющего функцию логического ИЛИ
T2 = [0, 1, 1, 1];
% Инициализируем нейрон, обучим его на 20 проходах (эпохах)
% инициализация нейрона net
net = init (net);
% имитация работы нейрона net на последовательности входов P2
Y = sim (net, P2);
% установка количества проходов
net.trainParam.epochs = 20;
% обучение нейрона net на обучающей выборке <P2, T2>
net = train(net, P2, T2);
% имитация работы нейрона net на последовательности входов P2
Y = sim (net, P2);
```

Функция train выводит график обучения нейрона (рис. 2).

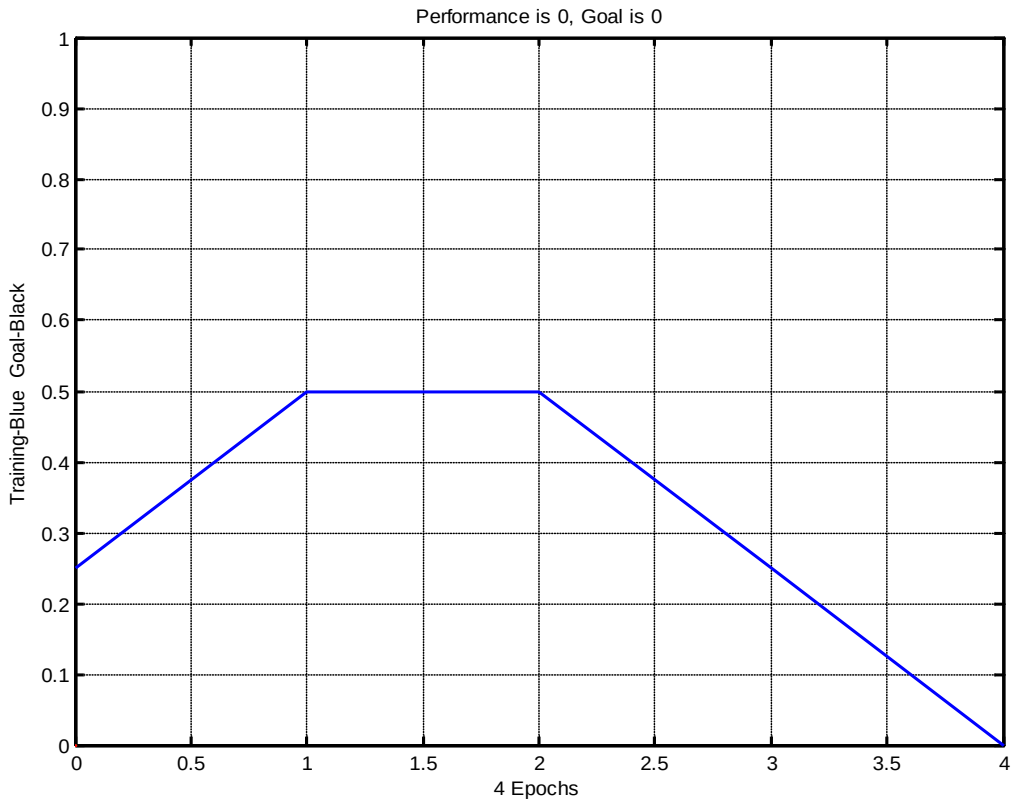


Рис. 2. График обучения нейрона

Для случайного изменения весов и порога срабатывания используем функцию *init*. По умолчанию для создаваемого нейрона указана функция *hardlim*.

ПРИМЕР 2 Обучение нейрона классификации векторов на две категории

Начнем с классификации векторов на основе двухвходового нейрона. Будем использовать функцию *newp* для создания нейрона, *sim* для имитации его работы, *adapt для* адаптации (обуче-

ния) нейронной сети. Обучим двухвходовый нейрон классифицировать входные векторы на две категории.

```
% определение четырех двухэлементных входов
P = [ -0.5 -0.5 0.4 -0.2; -0.5 0.5 -0.4 1.0];
% зададим желаемые выходы нейрона для определенных векторов
T = [1 1 0 0];
% изобразим входные векторы (точки) на плоскости:
plotpv(P, T);
```

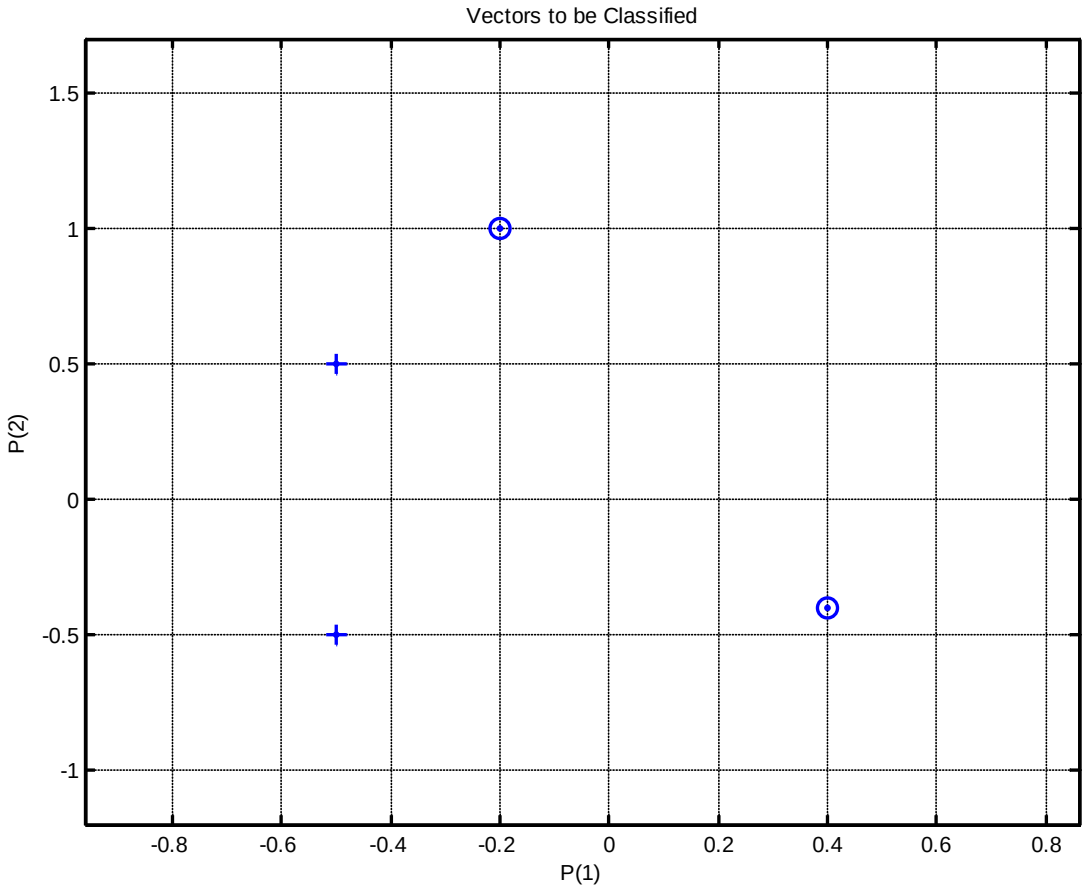


Рис. 3. Исходные векторы, предназначенные для классификации нейроном

Каждый из четырех входных векторов на плоскости *P* определяется двумя координатами, представленными как двухэлементные столбцы в матрице *P* (рис. 3).

Создадим один линейный нейрон с двумя входами, значения которых лежат в интервале [-1, 1].

```
% создание линейного нейрона с двумя входами из интервала [-1, 1]
net = newp([-1 1; -1 1], 1);
```

Нейрон по умолчанию имеет функцию активации *hardlim* и такой нейрон разделяет входные векторы прямой линией.

Определим координаты линии классификации: веса *(IW)* и порог срабатывания нейрона *(b)*.

```
% получение управляющей структуры linehandle для изображения
% разделяющей линии в координатах весов (IW) и порога
% срабатывания нейрона (b)
```

```
linehandle = plotpc (net.IW{1}, net.b{1});
% изображение разделяющей прямой
plotpc(net.IW{1}, net.b{1});
```

Если исходным весам задать нулевые значения, то любые входы дадут одинаковые выходы, и линия классификации не будет видна на плоскости. Проведем обучение:

```
% очистка координатных осей
cla;
% изображение входных векторов двух категории, категория задается
% элементами вектора T
plotpv(P, T);
% получение управляющей структуры linehanctle
linehandle = plotpc(net.IW{1}, net.b{1});
% присвоение начального значения ошибки
E = 1;
% инициализация нейрона
net = init (net);
% получение управляющей структуры linehandle
linehandle = plotpc (net.IW{1}, net.b{1});
% организация цикла пока ошибка не равна 0
while (mse(E))',
% адаптация нейрона net на обучающей выборке <P, T>,
% функция возвращает адаптированный нейрон net,
% выход Y, ошибку E
[net, Y, E] = adapt (net, P, T);
% изображение разделяющей прямой нейрона после адаптации
linehandle = plotpc(net.IW{1}, net.b{1}, linehandle);
% очистка окна графиков
drawnow;
% конец цикла while
end;
```

Функция *adapt* возвращает новый объект – сеть, которая выполняет классификацию, выход сети и ошибку (рис. 4).

Проведем классификацию нового вектора с помощью обученного нейрона на основе функции *sim*. Определим новый вектор *P*.

```
% определение вектора P
P = [0, 6; 1, 1];
% имитация работы нейрона net, получение отклика нейрона a
a = sim (net, P);
% изображение входа P, отнесенного нейроном к категории a
plotpv (P, a);
```

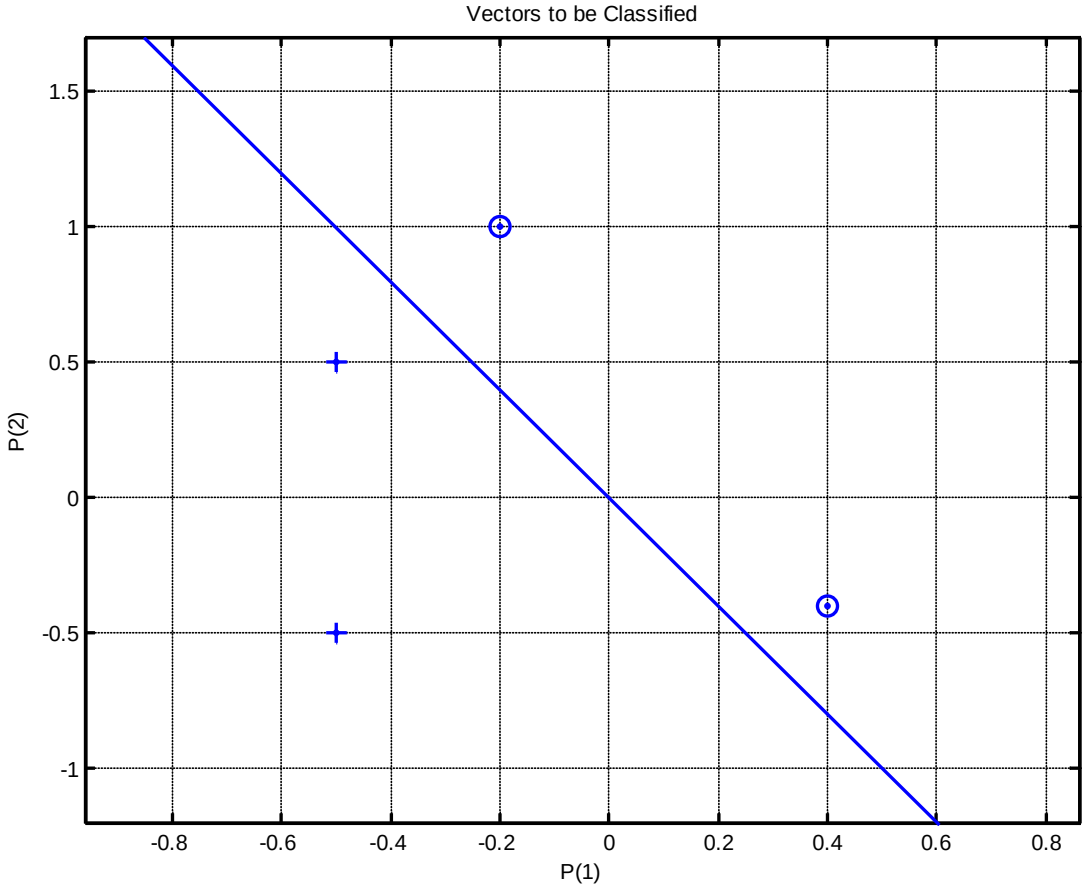


Рис. 4. Прямая, разделяющая исходные векторы на классы

Обученный нейрон можно использовать для классификации любого вектора:

```
% включить режим добавления графиков в графическом окне
hold on;
% изображение входных точек в соответствии с категориями T
plotpv(P, T);
% изображение разделяющей поверхности
plotpc (net.IW{1}, net.b{1});
% отключение режима добавления графиков
hold off;
```

Нейрон классифицирует новую точку, как принадлежащую категории «0» (представлена кружком), а не категории «1» (представлена +).

ПРИМЕР 3 Создание слоя линейных нейронов

Рассмотрим последовательность из 10 шагов (для выхода *T1*, который известным образом зависит от входов *P1):*

```
% последовательность входов
P1 = {-1 0 0 0 1 1 -1 0 -1 1};
% последовательность выходов
T1 = {-1 -1 1 0 1 2 0 -1 -1 1};
```

Используем функцию *newlin*, чтобы создать нейрон со значениями входа в интервале [-1; 1], задержками входа от 0 до 1 и уровнем обучения 0,1.

```
% создание линейного слоя из одного нейрона со значениями входа
% в интервале [-1; 1], задержками входа от 0 до 1
% и уровнем обучения 0,1.
net = newlin ([-1 1], 1, [0 1], 0.1);
```

Адаптируем нейрон к задаче одним проходом через последовательность входа. Измерим среднюю квадратичную ошибку с помощью функции *mse(e)*.

```
% адаптация нейрона к последовательности P1
[net, y, e, pf] = adapt (net, P1, T1);
% измерение ошибки
mse (e)
```

Получим довольно большую ошибку. Вновь адаптируем сеть на 10 шагах последовательности, используя предыдущее значение *pf* как новое исходное значение задержки:

```
P2 = {1 -1 -1 1 1 -1 0 0 0 1};
T2 = {2 0 -2 0 2 0 -1 0 0 1};
% адаптация с начальным вектором задержки pf
[net, y, e, pf] = adapt (net, P2, T2, pf);
mse (e)
```

Адаптируем сеть на 100-разовом прогоне последовательности:

```
% формирование новой последовательности входов
P3 = [P1 P2];
% формирование новой последовательности выходов
T3 = [T1 T2];
% установка количества проходов
net.adaptParam.passes = 100;
% адаптация нейрона
[net, y, e] = adapt (net, P3, T3);
```

Получим приемлемую ошибку, значит сеть обучена зависимости входов от выходов.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что представляет собой математическая модель нейрона?Как выглядит формульное представление алгоритма обучения по дельта-правилу?
2. Какое ограничение имеет линейный нейрон?
3. Можно ли обучить линейный нейрон выполнять логическую функцию исключающего ИЛИ?
4. Какие функции используются для настройки весов персептрона в среде MATLAB? В чем их отличие?

Лабораторная работа № 6 Изучение многослойного нелинейного персептрона и алгоритма обратного распространения ошибки

1 ЦЕЛЬ РАБОТЫ

Изучить возможности многослойного персептрона как аппроксиматора и классификатора.

2 СВЕДЕНИЯ ИЗ ТЕОРИИ

2.1 Алгоритм обратного распространения ошибки

Пусть определена трехслойная нейронная сеть с n входами, m выходами и l скрытыми между ними элементами, тогда необходимо рассмотреть и построить два слоя весов: от входов к скрытым элементам и к выходу, т. е. (W_1, W_2).

Назначение алгоритма обратного распространения ошибки – настройка всех слоев многослойной структуры. Рассмотрим работу алгоритма на примере сети с одним скрытым слоем и одним выходом (рис. 5). Преобразования входных сигналов, задаваемые нейронной сетью, определяются следующими формулами:

$$F(< W, X >) = 1/(1+\exp(-W^T X));$$
$$O_1 = 1/(1+\exp(-W_1^T X));$$
$$O_2 = 1/(1+\exp(-W_2^T O_1)).$$

Общая функция ошибки зависит от весов всех слоев, в нашем случае от вектора W_2 и от матрицы W_1:

$$E(W_1, W_2) = 1/2(Y - 1/(1+\exp(-W_2^T O_1)))^2,$$

где Y – выход, который задан в обучающей выборке.

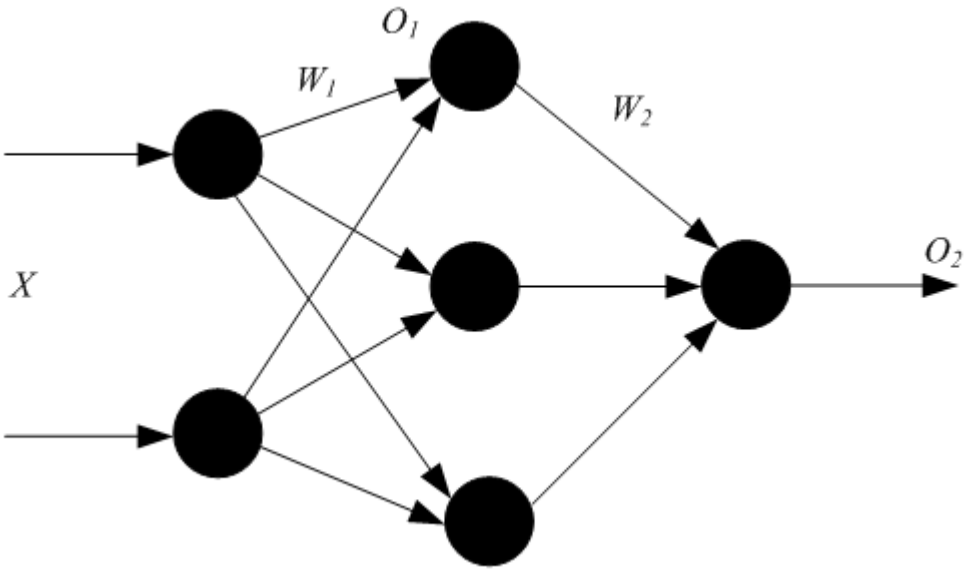


Рис. 5. Многослойный персептрон

Теперь необходимо определить приращение каждого веса с помощью частных производных:

$$\frac{\partial E(W_2, W_1)}{\partial W_2};$$
$$\frac{\partial E(W_2, W_1)}{\partial W_1}.$$

Для многослойной архитектуры частные производные ошибки по матрице весов каждого слоя определяются по формуле сложной производной. В случае униполярной сигмоиды правило изменения весов будет следующим:

$$W_2 = W_2 + h \cdot (Y - O_2) \cdot O_2 \cdot (1 - O_2) \cdot O_1;$$
$$\partial = (Y - O_2) \cdot O_2 \cdot (1 - O_2);$$
$$W_1 = W_1 + h \cdot \partial \cdot W_1 \cdot (1 - O_1) \cdot O_1 \cdot X.$$

Таким образом, метод обратного распространения ошибки позволяет изменять веса промежуточных слоев, хотя желаемые значения на промежуточных слоях не заданы.

2.2 Описание основных функций

Функция *newff* создает нейронную сеть прямого распространения сигнала, обучаемую с помощью алгоритма обратного распространения ошибки:

net = newff(PR, [S1 S2 SNl], {TF1 TF2 TFNl), BTF, BLF, PF).

Рассмотрим параметры функции *newff*: *PR* – матрица интервалов значений для *R* входных элементов, задаваемых минимальным и максимальным значениями; S_i – размер *i*-го слоя, для *N* слоев; TF_i – функция активации *i*-го слоя, по умолчанию используется функция *tansig* – гиперболический тангенс; *BTF* – функция обучения сети методом обратного распространения ошибки, по умолчанию используется функция *traingdx*; *BLF* – функция изменения весов при обучении, по умолчанию используется *learngdm*; *PF* – функция измерения ошибки, по умолчанию *mse*. Функция *newff* возвращает многослойную нейронную сеть прямого и обратного распространения сигнала и ошибки соответственно. Функции активации могут быть выбраны из следующего перечня: гиперболический тангенс *tansig*, логистическая сигмоида *logsig* или линейная функция *purelin*.

3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

ПРИМЕР 1 Создание и обучение нейронной сети с помощью алгоритма обратного распространения ошибки

Зададим с помощью графика исходную функцию:

```
% входы НС
P = [0 1 2 3 4 5 6 7 8];
% желаемые реакции НС
T = [0 0.44 0.88 0.11 -0.66 -0.95 -0.45 0.18 0.92];
% изображение аппроксимируемой функции
plot(P, T, 'o');
```

Используем функцию *newff*, чтобы создать двухслойную сеть прямого распространения. Пусть сеть имеет входы с интервалом значений от 0 до 8, первый слой с 10 нелинейными сигмоидальными, второй – с одним линейным нейронами. Используем для обучения алгоритм обратного распространения ошибки (backpropagation) Левенберга – Марквардта (рис. 6).

```
% создание двухслойной НС прямого распространения с интервалом
% значений входов от 0 до 8, причем первый слой содержит
% 10 нелинейных сигмоид, а второй — один линейный нейрон.
% Для обучения используется алгоритм обратного распространения
% ошибки (backpropagation).
net = newff([0 8], [10 1], {'tansig' 'purelin'},'trainlm');
```

```
% имитация работы необученной НС
yl = sim (net, P);
% изображение результатов работы необученной НС
plot(P, T, 'o', P, yl, 'x') ;
% Обучим сеть на 100 эпохах с целевой ошибкой 0.01:
% установка количества проходов
net.trainParam.epochs = 50;
% установка целевого значения ошибки
net.trainParam.goal = 0.01;
% обучение НС (рис. 6)
net = train(net, P, T) ;
% имитация работы обученной НС
y2 = sim(net,  P);
% изображение результатов работы НС (рис. 7)
plot(P, T, 'o', P, yl, 'x', P, y2, '+');
```

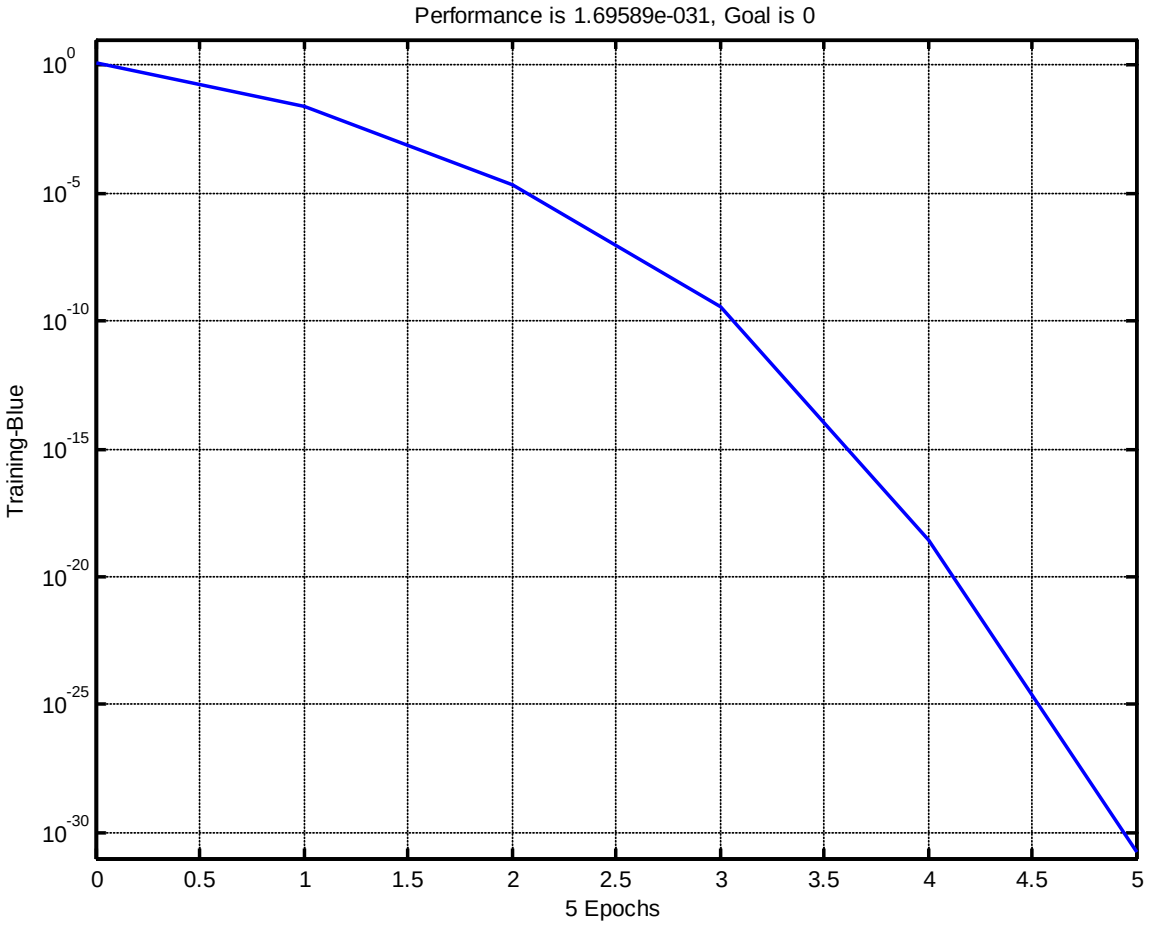


Рис. 6. График обучения двухслойного персептрона

Для исследования работы алгоритма обратного распространения ошибки воспользуемся примером, встроенным в Matlab toolbox, набрав команду demo.

В появившемся диалоговом окне необходимо последовательно выбирать пункты меню: Toolboxes->Neural Network->Other Demos->Other Neural Network Design textbook demos->Table of Contents->10-13->Backpropagation Calculation.

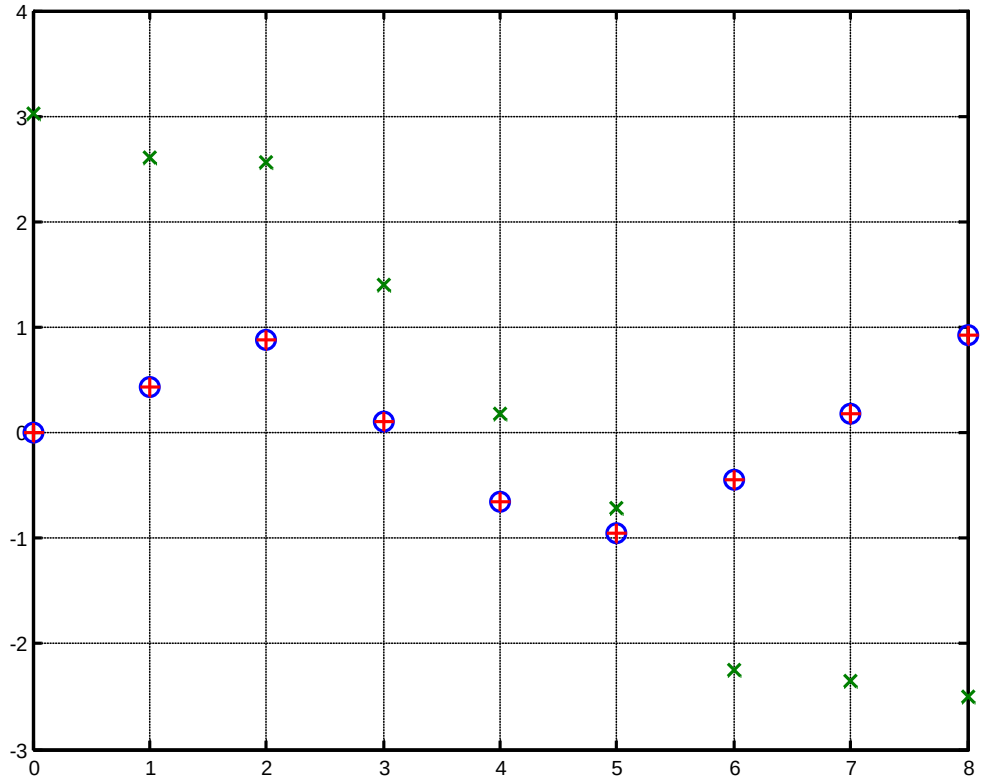


Рис. 7. Результат аппроксимации векторов двухслойным персептроном

В примере используется двухслойный персептрон с двумя нелинейными нейронами в первом слое и одним во втором. Действие алгоритма обратного распространения ошибки разбито на следующие шаги: назначение входа и желаемого выхода, прямой проход входного сигнала до выхода, обратное распространение ошибки, изменение весов. Переменные, позволяющие проследить работу алгоритма обратного распространения ошибки, обозначены следующим образом:

P – входной сигнал;

$W_1(i)$ – вектор весов первого слоя, $W_1(1)$ – вес связи, передающий входной сигнал на первый нейрон, а $W_1(2)$ – на второй;

$W_2(i)$ – вектор весов второго слоя, $W_2(1)$ – вес связи, передающий входной сигнал с первого нейрона во второй слой, а $W_2(2)$ – со второго;

$B_1(i)$ – вектор пороговых значений *(bias)* нейронов первого слоя, $i = 1, 2$;

B_2 – пороговое значение *(bias)* нейрона второго слоя;

$N_1(i)$ – вектор выходов первого слоя, $i = 1, 2$;

N_2 – выход второго слоя;

$A_1(i)$ – вектор выходных сигналов первого слоя после выполнения функции активации (сигмоиды), $i = 1, 2$;

A_2 – выход второго слоя после выполнения функции активации (линейной);

lr – коэффициент обучаемости.

Пусть входной сигнал $P = 1{,}0$, а желаемый выход $t = 1 + \sin(p * \pi / 4) = 1{,}707$.

Результаты выполнения этапов алгоритма представлены в табл. 1.

Таблица 1

Результаты поэтапного выполнения алгоритма обратного распространения ошибки

Этап	Прямое распространение входного сигнала	Обратное распространение ошибки	Изменение весов
$A_1(1)$, $A_1(2)$	$Logsig(W_1P+B_1)$ = = [0,321, 0,368]	Не выполняется	Не выполняется
A_2	$purelin(W_1P+B_1)$ = = 0,446	*То же*	*То же*
e	$t - A_2$ = 1,261	»	»
$N_1(1)$, $N_1(2)$	Не выполняется	$\frac{\partial \log sim(N_1)}{\partial N_1 \cdot W_2 \cdot N_2} =$ $= [0{,}049, 0{,}100]$	»
N_2	То же	$-2 \cdot \frac{\partial purelin(N_2)}{\partial N_2 \cdot e} = -2{,}522$	»
$W_1(1)$ $W_1(2)$	»	Не выполняется	$W_1 = W_1 - \mathrm{lr} \cdot N_1 \times P =$ $= [-0{,}265, -0{,}420]$
$B_1(1)$, $B_1(2)$	»	То же	$B_1 = B_1 - \mathrm{lr} \cdot N_1 =$ $= [-0{,}475, -0{,}140]$
B_2	»	»	$B_2 = B_2 - \mathrm{lr} \cdot N_2 = 0{,}732$
$W_2(2)$	»	»	$W_2 = W_2 - \mathrm{lr} \cdot N_2 \times N_1 =$ $= [0{,}171, 0{,}077]$

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каким алгоритмом обучают многослойные НС?
2. Из каких основных этапов состоит алгоритм обратного распространения ошибки?
3. Почему алгоритм обратного распространения ошибки относится к классу алгоритмов градиентного спуска?
4. Как влияет функция принадлежности на правило изменения весов в обратном алгоритме распространения ошибки?
5. Какая функция в среде MATLAB создает НС прямого распространения?
6. Какие функции активации могут быть назначены для нейронов НС прямого распространения?

Лабораторная работа № 7 Изучение радиальных базисных, вероятностных нейронных сетей, сетей регрессии

1 ЦЕЛЬ РАБОТЫ

Изучить модель вычислений радиального базисного нейрона, структуру и функции сетей регрессии, вероятностных нейронных сетей.

2 СВЕДЕНИЯ ИЗ ТЕОРИИ

2.1 Радиально-базисные сети. Сети регрессии. Вероятностные НС

Рассмотрим радиальный базисный нейрон с R входами. Структура нейрона представлена на рис. 8. Радиальный базисный нейрон (РБН) вычисляет расстояние между векторами входов X и вектором весов W, затем умножает его на фиксированный порог b. Функция активации РБН, полученная в среде MATLAB, представлена на рис. 10. Радиальная базисная функция имеет максимум, равный 1, когда ее входы нулевые. Следовательно, радиальный базисный нейрон действует как

детектор, который получает на выходе 1, когда вход *X* идентичен его вектору весов *W*. Фиксированный порог *b* даст возможность управлять чувствительностью нейрона. Например, если нейрон имеет порог 0,1, то выход равен 0,5 для любого входного вектора *X*, находящегося на векторном расстоянии 8,326 (8,326/*b*) от *W*.

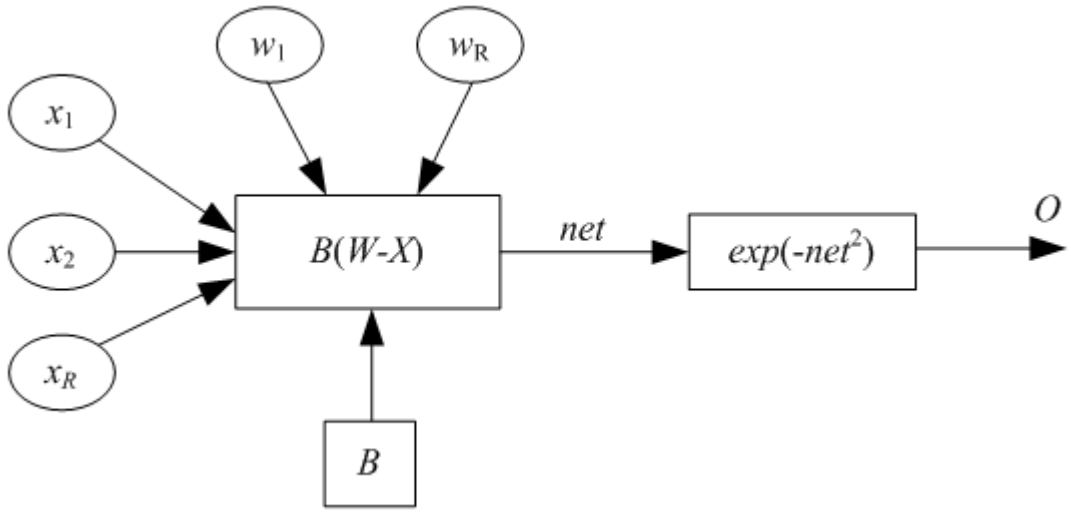


Рис. 8. Радиальный базисный нейрон

Радиальная базисная нейронная сеть (РБНС) состоит из двух слоев: скрытого радиального базисного слоя из S^1 нейронов и выходного линейного слоя из S^2 нейронов. Элементы первого слоя РБНС вычисляют расстояния между входным вектором и векторами весов первого слоя, сформированных из строк матрицы $W^{2,1}$. Вектор порогов *B* и расстояния поэлементно умножаются. Выход первого слоя можно выразить формулой

$$A^1 = radbas(\|W - X\| \cdot B),$$

где A^1 – выход первого слоя; функция *radbas* – радиально-базисная функция; *W* – матрица весов первого слоя сети; *X* – входной вектор; *B* – вектор порогов первого слоя.

Согласно формуле радиальные базисные нейроны с вектором весов, близким к *X*, сгенерируют значения, близкие к 1. Если нейрон имеет выход 1, то это значение весами второго слоя будет передано на его линейные нейроны. Фактически радиальный базисный нейрон с выходом 1 превращает выходы всех остальных нейронов в нули. Тем не менее, типичным является случай, когда несколько нейронов дают на выходах значимый результат, хотя и с разной степенью.

Радиальные базисные нейронные сети обучаются в три этапа. Опишем этапы обучения.

Первый этап – выделение центров (весов). Центры, представленные в РБН-слое, оптимизируются первыми с помощью обучения без учителя. Центры могут быть выделены разными алгоритмами, в частности обучением Кохонена. Алгоритмы должны разместить центры, отражая кластеризацию исходных данных.

Второй этап – назначение отклонений. Отклонения могут быть назначены различными алгоритмами, например алгоритмом «ближайшего соседа».

Третий этап – линейная оптимизация. Можно использовать методы обучения по дельта-правилу, обратному распространению ошибки.

Нейронные сети регрессии (НСР) имеют такой же, как и РБНС, первый слой, но второй слой строится специальным образом. Для аппроксимации функций часто используются обобщенные сети регрессии (generalized regression neuron networks). Второй слой, как и в случае РБНС, выполняет поэлементное произведение строки $W_{1,2}$ и вектора выхода первого слоя a^1. Он имеет столько нейронов, сколько существует целевых пар <входной вектор/целевой вектор>. Матрица весов *W* – это набор целевых строк. Целевое значение – это значение аппроксимируемой функции в обучающей выборке. Предположим, имеется один входной вектор x_i, который сгенерирует на

выходе первого слоя выход, близкий к 1. В результате выход второго слоя будет близок к t_i одному из значений аппроксимируемой функции, использованной при формировании второго слоя.

Сети регрессии иногда называют *Байесовскими вероятностными сетями регрессии,* или обобщенными НС регрессии. Некоторые реализации сетей регрессии имеют четыре слоя: входной, выходной, слои радиальных центров, элементов регрессии. Радиальный слой представляет собой центры-кластеры известных обучающих данных и содержит такое же количество элементов, как обучающая выборка; РБН обучаются алгоритмом кластеризации. Слой регрессии имеет только на один элемент больше, чем выходной слой, и содержит линейные элементы одного из двух типов. Элемент первого типа вычисляет условную вероятность каждого выходного атрибута, элемент второго типа вычисляет плотность вероятности. Выходной слой выполняет специальные функции деления. Каждый элемент делит выходы, ассоциированные первым типом, с помощью элементов второго типа.

Байесовские вероятностные НС используются только для проблем классификации. Они содержат четыре слоя: входной, выходной, слой РБН и элементов линейной классификации. Слои могут содержать квадратную матрицу потерь, включение которой возможно, только если третий и четвертый слои состоят из одинакового числа элементов. Радиальные базисные нейроны в таких сетях используются для хранения образцов, взятых из обучающей выборки, которая берется полностью. Следовательно, первый скрытый слой содержит такое же количество элементов, что и обучающая выборка. Так как элементы слоя классификации связаны с выходом каждого класса, можно оценить вероятность принадлежности последнему. Если используется матрица потерь, то цена решения минимальна. Такие сети обычно быстро тренируются, но медленно вычисляют из-за большого размера.

Вероятностные нейронные сети (ВНС, probabilistic neuron networks) используются для решения проблемы классификации. Первым слоем в архитектуре ВНС является слой радиальных базисных нейронов, который вычисляет расстояние и вектор индикаторов принадлежности другим входным векторам, используемым при обучении. Второй слой суммирует эти значения для каждого класса входов и формирует выходы сети, как вектор вероятностей. Далее специальная функция активации (compete) определяет максимум вероятностей на выходе второго слоя и устанавливает данный выход в 1, а остальные выходы в 0. Матрица весов первого слоя $W^{1,1}$ установлена в соответствии с обучающими парами. Блок расчета расстояний получает вектор, элементы которого показывают, насколько близок входной вектор к векторам обучающего множества. Элементы вектора умножаются на вектор порогов и преобразуются радиальной базисной функцией. Входной вектор, близкий к некоторому образцу, устанавливается в 1 в выходном векторе первого слоя. Если входной вектор близок к нескольким образцам отдельного класса, то несколько элементов выходного вектора первого слоя будут иметь значения, близкие к 1.

Веса второго слоя $W^{1,1}$ устанавливаются по матрице T целевых векторов, каждый вектор которой включает значение 1 в строке, связанной с определенным классом входов, и нули в остальных позициях. Произведения $T\ a^1$ суммируют элементы выходного вектора первого слоя a^1 для каждого из K классов. Затем функция активации второго слоя (compete) установит значение 1 в позицию, соответствующую большему элементу выходного вектора, и 0 во все остальные. Следовательно, сеть классифицирует входные векторы, назначая входу единственный класс на основе максимальной вероятности принадлежности.

2.2 Описание основных функций

Функция *newrb* создает радиальную базисную сеть и имеет следующий синтаксис:

net = newrb(P, T, goal, spread).

Радиальные базисные сети используют для аппроксимации функций. Функция *newrb* конструирует скрытый (первый) слой из радиальных базисных нейронов и использует значение средней квадратичной ошибки *(goal)*. Функция *newrb(P, T, goal, spread)* имеет следующие аргументы: P – матрица Q входных векторов размерности R на Q; T – матрица Q векторов целевых классов S на Q; *goal* – средняя квадратичная ошибка, по умолчанию 0,0; *spread* – разброс радиальной ба-

зисной функции, по умолчанию 1,0. Функция создает и возвращает в качестве объекта радиальную базисную сеть. Большое значение разброса приводит к большей гладкости аппроксимации. Слишком большой разброс требует много нейронов, для того чтобы подстроиться под быстро изменяющуюся функцию, слишком малый – для достижения гладкости аппроксимации. Подобрать значение разброса можно с помощью многократных вызовов функции *newrb*. Создадим в среде MATLAB радиальную базисную сеть:

net = newrbe(P, T, spread).

Функция *newrbe* проектирует радиальную базисную сеть с нулевой ошибкой для заданных векторов. Функция *newrbe(P, T, spread)* имеет три параметра: *P* – матрица *Q* входных векторов размерности *R* на *Q*; *T* – матрица *Q* целевых векторов – описателей класса размерности *S* на *Q*; *spread* – разброс радиальной базисной функции, по умолчанию 1,0. Функция создает радиальную базисную сеть.

Функция ***newgrnn*** проектирует НС регрессии – это вид радиальной базисной сети, которая часто используется для аппроксимации функций и быстро строит сеть для аппроксимации:

net = newgrnn(P, T, spread).

Функция *newgrnn(P, T, spread)* имеет следующие входы: *P* – матрица *Q* входных векторов размерности *R* на *Q*; *T* – матрица *Q* целевых векторов классов размерности *S* на *Q*; *spread* – разброс радиальных базисных функций, по умолчанию 1,0. Функция возвращает НС регрессии. Чем больше разброс, тем более гладкой будет функция аппроксимации. Для того чтобы настроить функцию аппроксимации на исходные данные, используют разброс меньший, чем типичное расстояние между входными векторами.

Функция *newgrnn* создает двухслойную сеть. Первый слой содержит радиальные базисные нейроны, вычисляющие расстояние между входами и весами с помощью *netprod*. Второй слой имеет нейроны с функцией активации *purelin*. Только у первого слоя существует *bias*.

Функция ***newpnn*** создает вероятностную НС, проектируемую с помощью функции

net = newpnn(P, T, spread).

Вероятностная НС – это вид радиальной базисной сети, как и НС регрессии, но данные сети используются для решения задачи классификации, а не аппроксимации. Функция *net = newpnn(P, T, spread)* имеет такие же параметры, как и вышеописанная функция *newgrnn*. Если разброс близок к нулю, вероятностная НС действует как классификатор на основе принципа выбора «ближайшего соседа», в противном случае сеть принимает в расчет несколько близких векторов.

3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

ПРИМЕР 1 Создание и обучение НС регрессии

Рассмотрим проектирование НС регрессии. Определим:

```
% входы НС регрессии
P = [1 2 3];
% выходы НС регрессии
T = [3.0 5.1 4.8];
```

Воспользуемся функцией *newgrnn* для создания НС регрессии. Зададим разброс радиальных базисных функций (переменная *spread)* меньше, чем 1, для того чтобы получить хорошую аппроксимацию данных и более гладкую функцию.

```
% установка разброса радиальных базисных функций
```

```
spread = 0.8;
% создание НС регрессии
net = newgrnn(P, T, spread);
% имитация работы НС регрессии
A = sim (net, P);
% изображение аппроксимируемой функции
plot(P, T, '.', 'markersize', 30);
% установка режима добавления графиков на координатные оси
hold on;
% изображение работы необученной НС регрессии
plot(P, A, '.', 'markersize', 30, 'color', [1 0 0]);
% очистка координатных осей
cla reset;
% установка нового входа НС регрессии
p = 4.5;
% получение отклика НС регрессии
a = sim (net, p);
% изображение аппроксимируемой функции
plot(P, T, '.', 'markersize', 30);
% установка диапазонов осей X и Y
axis ([0 9 -1 4]);
% включение режима добавления графиков
 hold on;
% изображение отклика НС регрессии на вход p
plot(p, a, '.', 'markersize', 30, 'color', [1 0 0]);
% написать заголовок графика
title('Новый входной вектор ');
% пометить ось X
xlabel('P и p');
% пометить ось Y
ylabel('T и a');
% очистить координатную сетку
cla reset;
% определить последовательность входов P2
P2 = 0: 0.2 : 9;
```

Сымитируем отклик сети для различных значений, чтобы увидеть результат аппроксимации (рис. 9).

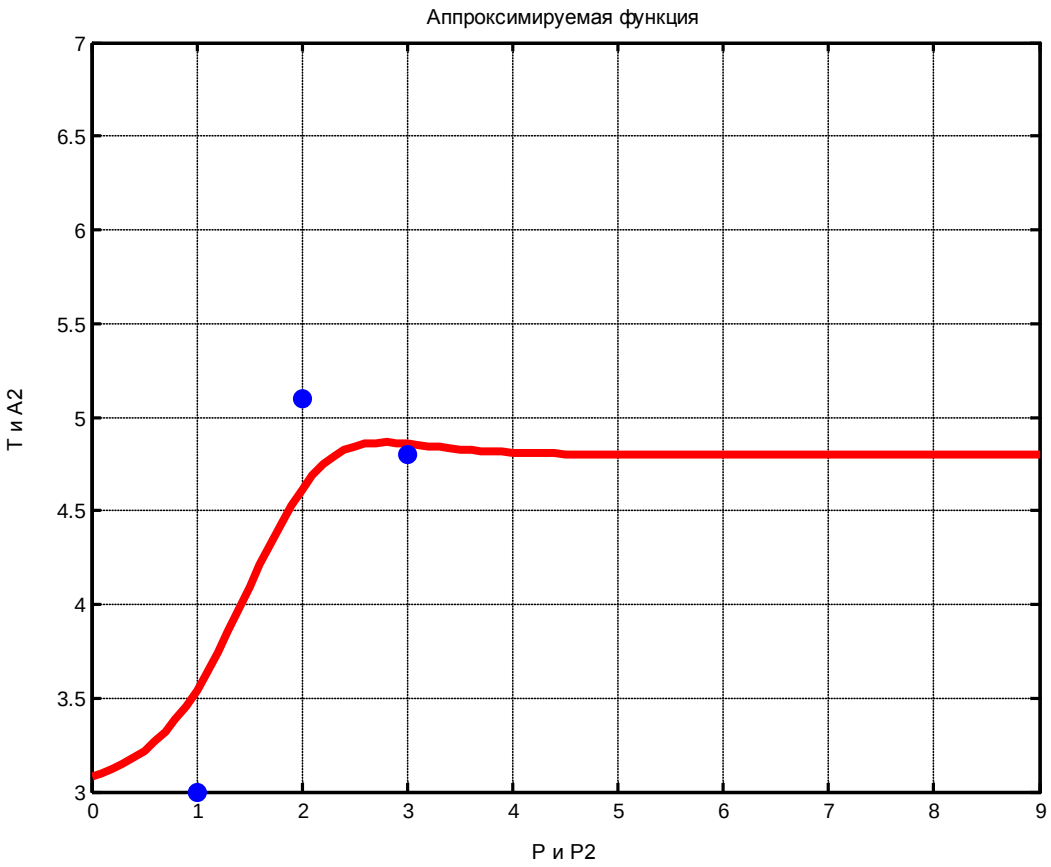


Рис. 9. Аппроксимация точек с помощью нейронной сети регрессии

```
% получить отклик НС регрессии на последовательность входов P2
A2 = sim (net, P2);
% изобразить отклик НС регрессии
plot(P2, A2, 'linewidth', 4, 'color', [1 0 0]);
% включить режим добавления графиков
hold on;
% изобразить аппроксимируемую функцию
plot(P, T, '.', 'markersize', 30);
% установить диапазон осей
axis ([0 9 3 7]);
% озаглавить график
title ('Аппроксимируемая функция');
% пометить ось X
 xlabel('P и P2');
% пометить ось Y
ylabel('T и A2');
```

ПРИМЕР 2 Использование РБНС для аппроксимации функций

Рассмотрим аппроксимацию функций на основе радиальной базисной сети.

```
% определение диапазона значений радиальной базисной функции
P = -4 : 0.1 : 4;
% вычисление радиальной базисной функции на диапазоне P
a = radbas (P);
% изображение РБФ
plot (P, a);
```

```
% озаглавить  график
title('Радиальная базисная функция');
% пометить ось X
xlabel('Вход P');
% пометить ось Y
ylabel ('Выход a');
```

На рис. 10 изображена радиальная базисная функция.

Функция *newrbe* создаст необходимую сеть. Зададим аппроксимируемую функцию как набор точек:

```
% определение последовательности аргументов
% аппроксимируемой функции P
P = -1 : 0.1 : 1;
% определение значений аппроксимируемой функции,
% соответствующих P
T = [-.6662 -.3766 -.1129 .2111 .6565 .3301 .3649 .2006...
-.1913 -.3994 -.5022 -.4531 -.1133 .0866 .3333 .4955...
.3488 .2833 -.1112 -.6685 -.3255];
```

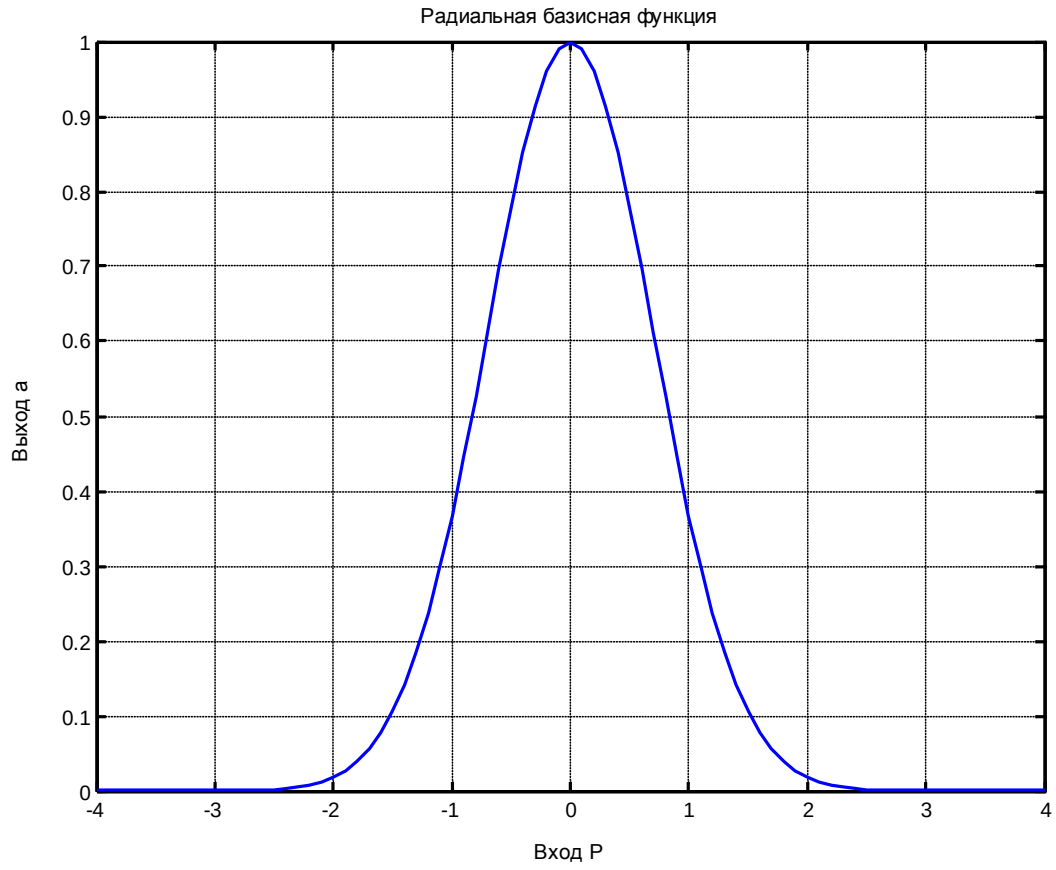


Рис. 10. Радиальная базисная функция

Изобразим график функции (рис. 11).

```
% изображение аппроксимируемой функции
plot(P, T, '*');
% озаглавить график
title('Обучающая выборка');
```

```
% пометить ось X
xlabel('Входной вектор P');
% пометить ось Y
ylabel('Вектор T');
```

Далее необходимо найти функцию, которая хорошо описывает заданные 21 точку. Функция *newrb* создает РБНС для аппроксимации:

```
% целевой среднеквадратичной ошибки
e = 0.02;
% разброса РБФ
sp = 1;
% создание РБФ
net = newrb(P,T,e,sp);
% определение вектора входов
X = -1 : 0.01 : 1;
```

Сымитируем работу сети.

```
% формирование отклика Y
Y = sim(net, X);
% включение режима добавления графика
hold on;
% изображение результатов аппроксимации
plot(X, Y);
% отключение режима добавления графика
hold off;
```

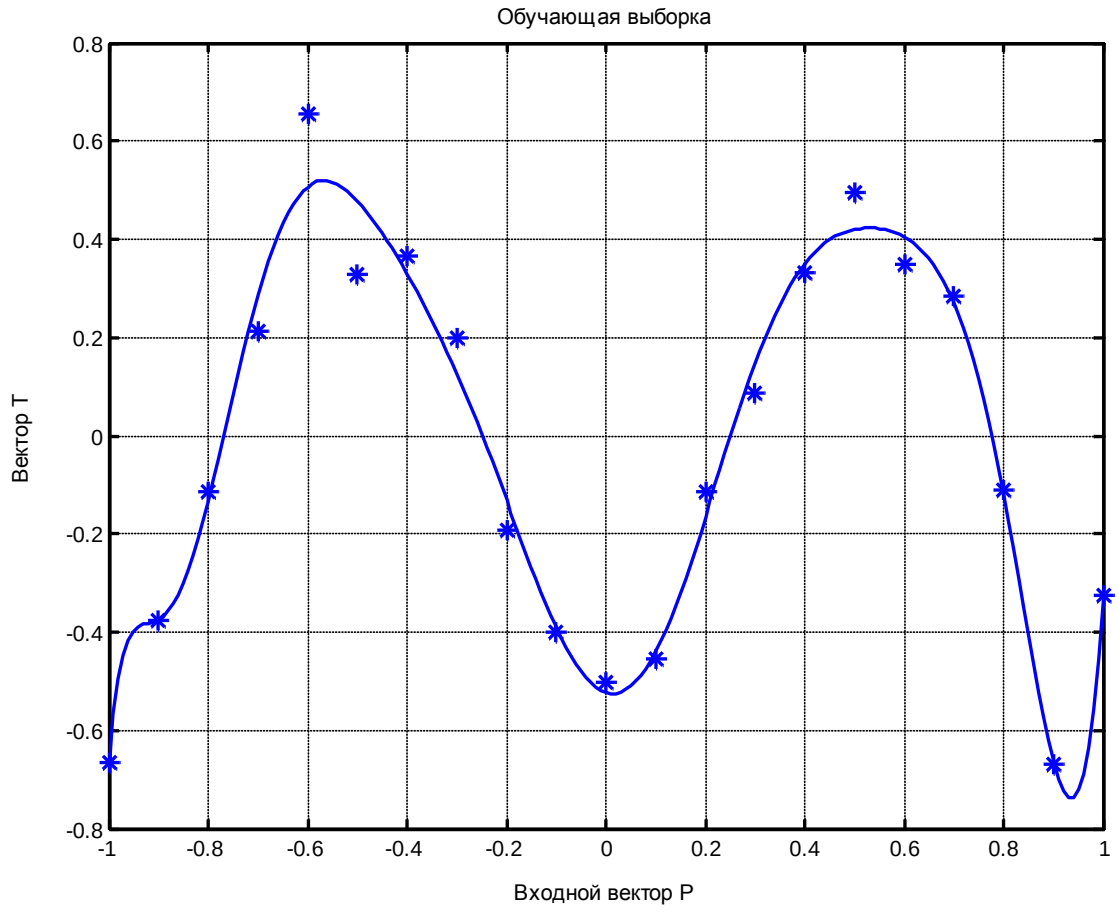


Рис. 11. Результат аппроксимации с помощью радиальной базисной нейронной сети

ПРИМЕР 3 Использование вероятностной НС для классификации векторов

Рассмотрим задачу классификации с набором входов *P* и множеством классов, обозначенным *Tc*.

```
% определение входов
P = [1 2 3 4 5 6 7];
% определение желаемых выходов
Tc =[1 1 3 3 2 1 1];
% конвертирование индексов в векторы, содержащие 1
% в индексных позициях
T = ind2vec (Tc);
% создание вероятностной НС
net = newpnn (P, T);
% имитация работы вероятностной НС
Y = sim(net, P);
% конвертирование номеров классов в векторы.
Yc = vec2ind(Y);
```

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какую функцию называют радиальной базисной функцией?
2. Из каких слоев состоит радиально-базисная НС?
3. Из каких слоев состоит НС регрессии?
4. Из каких слоев состоит вероятностная НС?
5. Какие виды НС предназначены для решения задачи аппроксимации функций, а какие – для классификации объектов?
6. Какие типы НС создаются в среде MATLAB с помощью функций *newrb*, *newrbe*, *newgrnn*, *newpnn*?