МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания

по выполнению лабораторных работ
по дисциплине
«Основы распознавания образов»
для направления подготовки **09.03.02 Информационные системы и технологии**
направленность (профиль) **Информационные системы и технологии обработки цифрового контента**

Пятигорск
2024

ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа № 1

Основы байесовских методов классификации

Цель и содержание работы: познакомить студента с основными аспектами байесовских методов классификации.

Теоретическое обоснование

Байесовский подход является классическим в теории распознавания образов и лежит в основе многих методов. Он опирается на теорему о том, что если плотности распределения классов известны, то алгоритм классификации, имеющий минимальную вероятность ошибок, можно выписать в явном виде. Для оценивания плотностей классов по выборке применяются различные подходы.

Рассмотрим пример параметрического подхода, но сначала определим вероятностную постановку задачи классификации.

Пусть X – множество объектов, Y – конечное множество имён классов, множестве X $\times$ Y является вероятностным пространством с плотностью распределения p(x, y) = P(y)p(x|y). Вероятности появления объектов каждого из классов $P_y = P(y)$ называются априорными вероятностями классов. Плотности распределения $p_y(x) = p(x|y)$ называются функциями правдоподобия классов. Вероятностная постановка задачи классификации разделяется на две независимые подзадачи.

1. Имеется простая выборка $X^l = (x_i, y_i)_{i=1}^{l}$ (l – размер выборки) из неизвестного распределения $p(x, y) = P_y p_y(x)$. Требуется построить эмпирические оценки априорных вероятностей $\hat{P}_y$ и функций правдоподобия $\hat{p}_y(x)$ для каждого из классов $y \in Y$.

2. По известным плотностям распределения $p_y(x)$ и априорным вероятностям P_y всех классов $y \in Y$ построить алгоритм a(x), минимизирующий вероятность ошибочной классификации.

Первая задача имеет множество решений, поскольку многие распределения p(x, y) могли бы породить одну и ту же выборку X^l. Приходится привлекать различные предположения о плотностях, что и приводит к большому разнообразию байесовских методов.

В параметрическом подходе предполагается, что плотность распределения выборки известна.

Нормальный дискриминантный анализ – это специальный случай байесовской классификации, когда предполагается, что плотности всех классов $p_y(x)$, $y \in Y$ являются многомерными нормальными.

Приведём формулу n-мерного (гауссова) распределения:

$$p(x) = \frac{1}{(2\pi)^{n/2}|S|^{1/2}} \exp\left(-\frac{1}{2}(\tilde{x}-m)^T S^{-1}(\tilde{x}-m)\right), \quad (1.1)$$

где n – количество числовых признаков, $m = E[x]$ – математическое ожидание (центр), S – ковариационная матрица ($S = E[(x-m)(x-m)^T]$), |S| - детерминант S.

Пример 1. Вычислим, используя Matlab, значение гауссова распределения для $x_1 = [0.2, 1.3]^T$ и $x_2 = [2.2, -1.3]^T$, где $m = [0, 1]^T$, а $S = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

```
close('all');
clear;
m=[0 1]'; S=eye(2);
x1=[0.2 1.3]'; x2=[2.2 -1.3]';
pg1=comp_gauss_dens_val(m,S,x1)
pg2=comp_gauss_dens_val(m,S,x2)
```

Где comp_gauss_dens_val(·) определим как

```
function [z]=comp_gauss_dens_val(m,S,x)
[l,c]=size(m);
z=(1/( (2*pi)^(l/2)*det(S)^0.5) )*exp(-0.5*(x-m)'*inv(S)*(x-m));
```

Функцию нужно создать в отдельном файле Matlab (New → Function). Таким образом будет два файла Matlab, которые могут располагаться в одной папке. Чтобы Matlab понимал откуда ему загружать необходимую функцию, можно нажать в окне Current Folder на нужную папку правой мышкой и установить (Add to Path → Selected Folders and Subfolders), рисунок 1.1.

Для того, чтобы выбрать нужную папку в окне Current Folder необходимо нажать кнопку «Browse for folder».

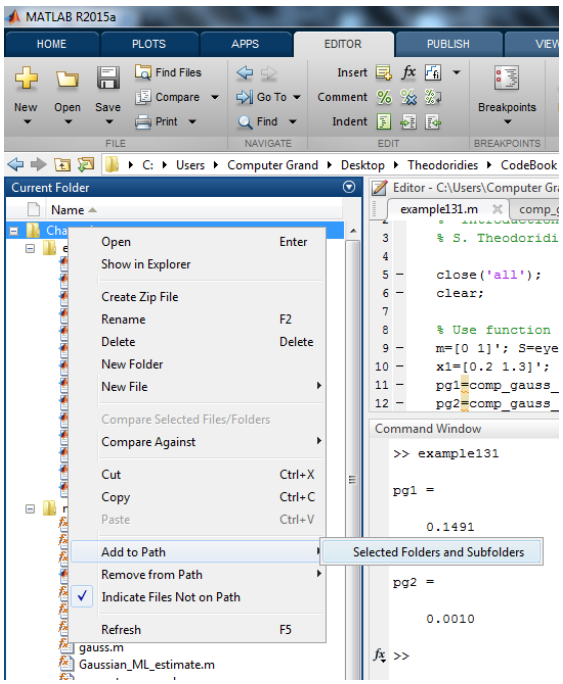


Рисунок 1.1 – Установка папки, откуда Matlab будет брать необходимые ему файлы

Функция **eye()** – создаёт единичную матрицу, запятая после квадратных скобок означает транспонирование вектора (**[]'**). Для получения справки по неизвестному объекту языка matlab, достаточно установить на него курсор и нажать F1 или написать команду help с именем объекта, допустим, **help eye**.

В результате выполнения кода получится, что pg1 = 0.1491, pg2 = 0.001.

На основании этого кода можно написать код байесовского классификатора, который определит к какому из двух классов относится входной вектор.

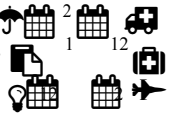
Есть два класса w_1 и w_2, имеющих гауссовы распределения $N(m_1, S_1)$, $N(m_2, S_2)$. m_1 = [1, 1]T, m_2 = [3, 3]T, S_1 = S_2 = $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Примем во внимание, что вероятность появления первого и второго классов подсчитаны заранее и равны $P(w_1) = P(w_2) = ½$. Требуется узнать к какому классу принадлежит вход $x = [1.8, 1.8]^T$.

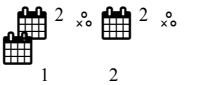
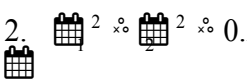
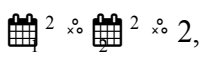
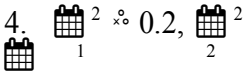
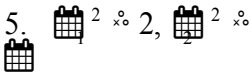
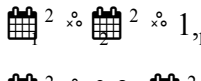
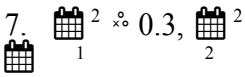
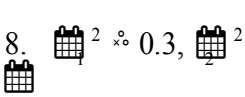
Пример 2. Напишем код байесовского классификатора, который решит эту задачу:

```
close('all');
clear;
P1=0.5;
P2=0.5;
m1=[1 1]';
m2=[3 3]';
S=eye(2);
x=[1.8 1.8]';
p1=P1*comp_gauss_dens_val(m1,S,x)
p2=P2*comp_gauss_dens_val(m2,S,x)
```

Получается, что p1 = 0.042, p2 = 0.0189, т.е. p1 > p2 , соответственно двумерный вектор принадлежит первому классу.

Далее, проведём геометрический анализ нормальной плотности, её зависимость от матрицы ковариации.

Пример 3. Рассмотрим код, который генерирует 500 двумерных точек, распределенных в соответствие с двумерным нормальным распределением с центром m = $[0, 0]^T$ и матрицей ковариации S  , где существует 8 типов этой матрицы:

1.  ;
2.  ;
3.  ;
4.  ;
5.  ;
6.  ;
7.  ;
8.  .

```
close('all');
clear;
% Генерируем точки для первого случая
randn('seed',0); % используем старый вариант вызова генератора
% случайных чисел, слово 'seed' заменяет ряд параметров для такого генератора
m=[0 0]'; % центр
S=[1 0;0 1]; % матрица ковариации
N=500; % количество точек
X = mvnrnd(m,S,N)'; % стандартная функция по генерации многомерного
% нормального случайного распределения
% Рисуем точки для первого варианта
figure(1), plot(X(1,:),X(2,:),'.');
figure(1), axis equal % устанавливаем тип стиля для осей
figure(1), axis([-7 7 -7 7])
% Рисуем точки для второго варианта
m=[0 0]';
S=[0.2 0;0 0.2];
N=500;
X = mvnrnd(m,S,N)';
figure(2), plot(X(1,:),X(2,:),'.');
figure(2), axis equal
figure(2), axis([-7 7 -7 7])
% Рисуем точки для третьего варианта
```

```
m=[0 0]';
S=[2 0;0 2];
```

```
N=500;
X = mvnrnd(m,S,N)';
figure(3), plot(X(1,:),X(2,:),'.');
figure(3), axis equal
figure(3), axis([-7 7 -7 7])
% Рисуем точки для четвертого варианта
m=[0 0]';
S=[0.2 0;0 2];
N=500;
X = mvnrnd(m,S,N)';
figure(4), plot(X(1,:),X(2,:),'.');
figure(4), axis equal
figure(4), axis([-7 7 -7 7])
% Рисуем точки для пятого варианта
m=[0 0]';
S=[2 0;0 0.2];
N=500;
X = mvnrnd(m,S,N)';
figure(5), plot(X(1,:),X(2,:),'.');
figure(5), axis equal
figure(5), axis([-7 7 -7 7])
% Рисуем точки для шестого варианта
m=[0 0]';
S=[1 0.5;0.5 1];
N=500;
X = mvnrnd(m,S,N)';
figure(6), plot(X(1,:),X(2,:),'.');
figure(6), axis equal
figure(6), axis([-7 7 -7 7])
% Рисуем точки для седьмого варианта
m=[0 0]';
S=[.3 0.5;0.5 2];
N=500;
X = mvnrnd(m,S,N)';
figure(7), plot(X(1,:),X(2,:),'.');
figure(7), axis equal
figure(7), axis([-7 7 -7 7])
% Рисуем точки для восьмого варианта
m=[0 0]';
S=[.3 -0.5;-0.5 2];
N=500;
X = mvnrnd(m,S,N)';
figure(8), plot(X(1,:),X(2,:),'.');
figure(8), axis equal
figure(8), axis([-7 7 -7 7])
```

На рисунках 1.2 – 1.9 показано сгенерированное множество точек.

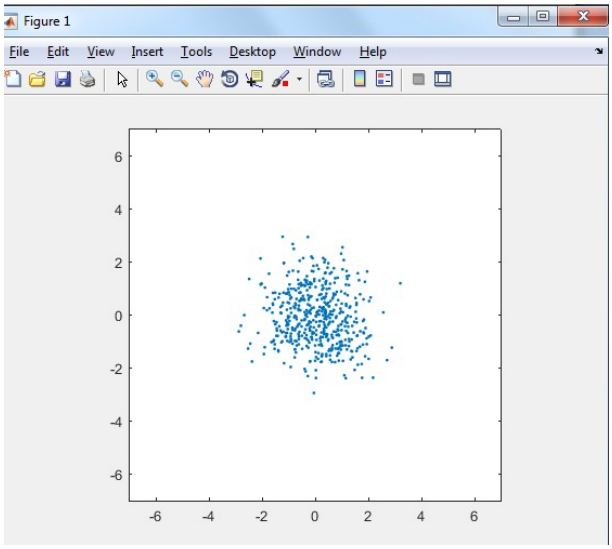


Рисунок 1.2 – Сгенерированное множество точек для случая $\sigma_1^2 = \sigma_2^2 = 1$, $r_{12} = 0$

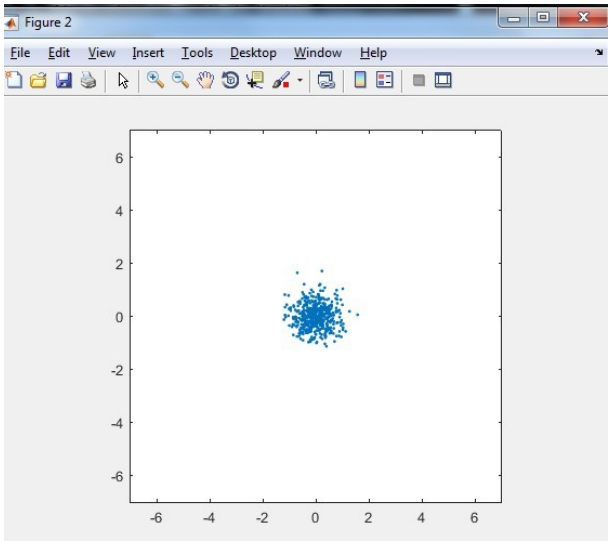


Рисунок 1.3 – Сгенерированное множество точек для случая $\sigma_1^2 = \sigma_2^2 = 0.2$, $r_{12} = 0$

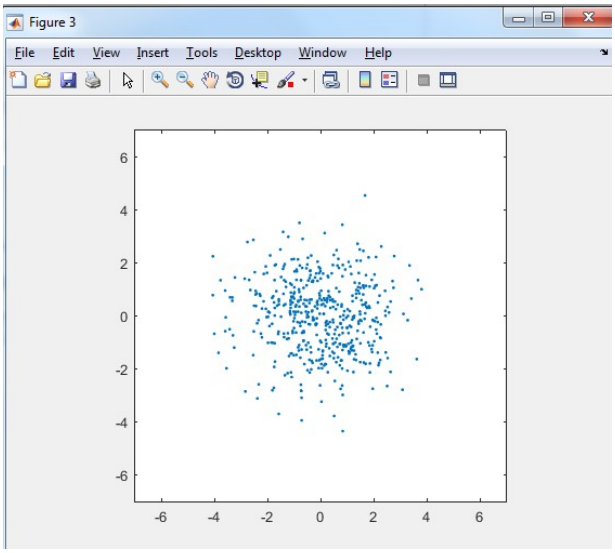


Рисунок 1.4 – Сгенерированное множество точек для случая $\sigma_1^2 = \sigma_2^2 = 2, \sigma_{12} = 0$

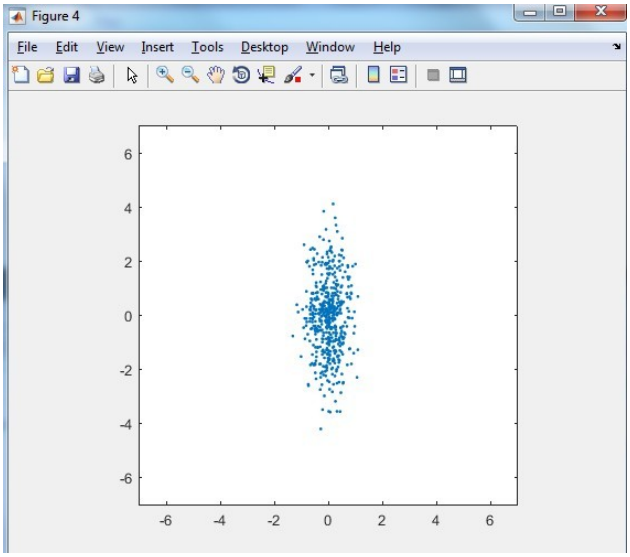


Рисунок 1.5 – Сгенерированное множество точек для случая $\sigma_1^2 = 0.2, \sigma_2^2 = 2, \sigma_{12} = 0$

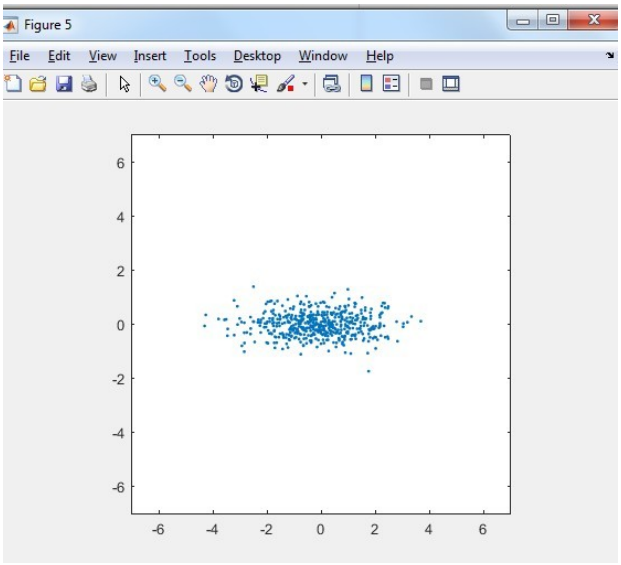


Рисунок 1.6 – Сгенерированное множество точек для случая $\sigma_1^2 = 2, \sigma_2^2 = 0.2, \sigma_{12} = 0$

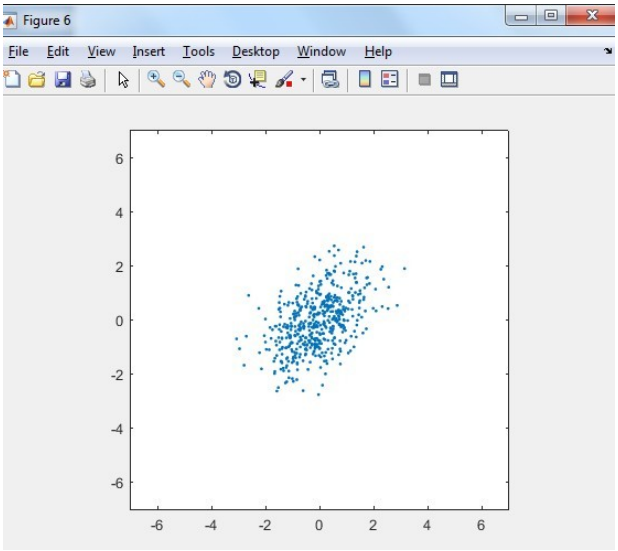


Рисунок 1.7 – Сгенерированное множество точек для случая $\sigma_1^2 = \sigma_2^2 = 1$, $\rho_{12} = 0.5$

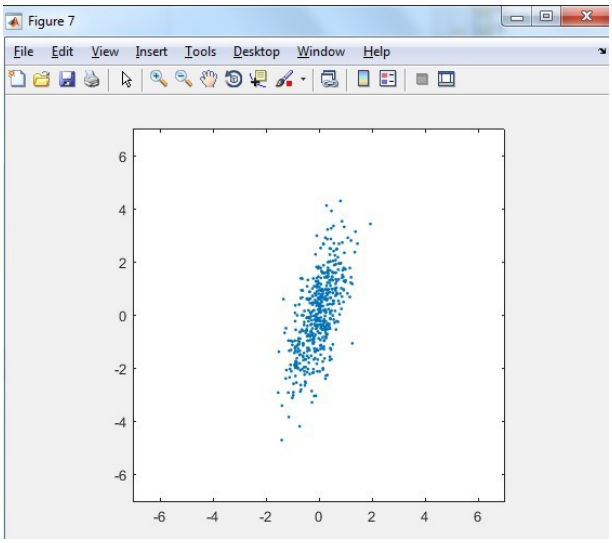


Рисунок 1.8 – Сгенерированное множество точек для случая $\sigma_1^2 = 0.3$, $\sigma_2^2 = 2$, $\rho_{12} = 0.5$

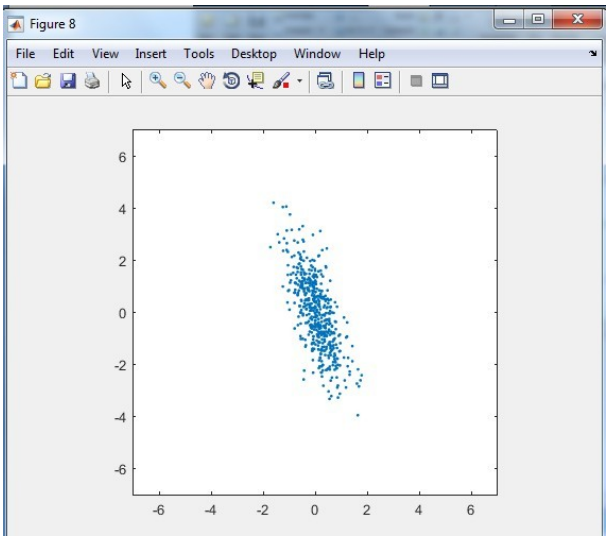


Рисунок 1.9 – Сгенерированное множество точек для случая $\sigma_1^2 = 0.3$, $\sigma_2^2 = 2$, $\rho_{12} = -0.5$

Из этих графиков можно сделать следующий вывод, если признаки некоррелированы, S = diag(σ_1^2, …, σ_n^2), то линии уровня плотности распределения имеют форму эллипсоидов с центром m и осями, параллельными линиям координат (рисунок 1.5 и 1.6). Если признаки имеют одинаковые дисперсии, то эллипсоиды являются сферами (рисунок 1.2–1.4). Если признаки коррелированны, то матрица S не диагональная и линии уровня имеют форму эллипсоидов, оси которых повернуты относительно исходной системы координат (рисунок 1.7–1.9).

Итак, в коде байесовского классификатора мы определяли класс по формуле из теории вероятности: умножали вероятность соответствующего класса на плотность распределения, причем предполагали, что плотность – нормальная, а вероятности известны.

Но можно решать задачу классификации и путём вычисления расстояний. Допустим, у нас есть некая точка в виде n-мерного вектора, и есть выборка, которую тоже можно отразить в виде набора точек в n-мерном пространстве. Тогда, если мы сначала получим некие характеристики выборки, допустим, узнаем её центр-масс, то можно определить расстояние от точки до центра масс. Если выборок много, то какое расстояние меньше, тому классу и принадлежит точка.

Пример 4. Сначала используем формулу из школьного курса: евклидово расстояние.

$$d(p,q) = \sqrt{(p_1 - \theta)^2 + (p_2 - \theta)^2 + \ldots + (p_n - \theta)^2} = \sqrt{\sum_{k=1}^{n} (p_k - \theta)^2}, \qquad (1.2)$$

где p и q n-мерные вектора.

```
close('all');
clear;
% x – входной вектор
x=[0.1 0.5 0.1]';
% m1 и m2 играют роль оценки двух неизвестных множеств: их центры масс
m1=[0 0 0]';
m2=[0.5 0.5 0.5]';
m=[m1 m2];
% передача информации в функцию
z=euclidean_classifier(m,x)


% функия в отдельном файле
function [z]=euclidean_classifier(m,X)
% вернёт индекс класса 1 или 2 к которому принадлежит точка X
% X – это матрица 1x3
% m – это матрица 3x2
% size() – возвращает координаты матрицы
[l,c]=size(m); % (l, c) = (3, 2)
[l,N]=size(X); % (l, N) = (3, 1)

for i=1:N % цикл по строкам X от 1 до 3
```

```
   for j=1:c % цикл по столбцам m от 1 до 2
      % высчитываем два евклидовых расстояния
      de(j)=sqrt((X(:,i)-m(:,j))'*(X(:,i)-m(:,j)));
   end
   % de(1) = 0.5196
   % de(2) = 0.5657
   % в пит будет содержаться минимальное значение, т.е. de(1)
   % z(i) – будет содержать его индекс, т.е. 1
   [num,z(i)]=min(de);
end
```

Однако можно использовать и более сложную формулу расстояния: расстояние Махаланобиса, которое использует не только центр масс, но и матрицу корреляции.

Если мы оценили множества и нашли его среднее значение $m = [m_1, m_2, \ldots, m_n]T$, а также матрицу ковариации S, то расстояние определится следующим образом от точки $x = [x_1, x_2, \ldots, x_n]T$ до m.

$$D_M(x) = \sqrt{(\tilde{x} - \mu)^T \Sigma^{-1} (\tilde{x} - \mu)}. \quad (1.3)$$

Тогда точка будет принадлежать тому классу для которого выполняется условие:

$$\sqrt{(\tilde{x} - \mu_i)^T \Sigma^{-1} (\tilde{x} - \mu_i)} = \min_j \sqrt{(\tilde{x} - m)_j^T S^{-1} (\tilde{x} - m)_j}, \quad i = \overline{1, l}, \quad (1.4)$$

т.е. для которого расстояние Махаланобиса минимально.

Пример 5. Напишем код для соответствующей классификации.

```
close('all');
clear;
x=[0.1 0.5 0.1]';
m1=[0 0 0]';
m2=[0.5 0.5 0.5]';
m=[m1 m2];
% для двух множеств одна и таже матрица ковариации S
S=[0.8 0.01 0.01;
  0.01 0.2 0.01;
  0.01 0.01 0.2];
z=mahalanobis_classifier(m,S,x)

% функция для расстояния Махаланобиса
function z=mahalanobis_classifier(m,S,X)
[l,c]=size(m);
[l,N]=size(X);
for i=1:N
   for j=1:c
      dm(j)=sqrt((X(:,i)-m(:,j))'*inv(S)*(X(:,i)-m(:,j)));
   end
% dm(1) = 1.1334
% dm(2) = 0.9918
   [num,z(i)]=min(dm);
end
```

% z = 2

Видим, что в этом случае точка принадлежит классу 2. Получается противоречие: евклидово расстояние показывает принадлежность к классу 1, а расстояние Махаланобиса – наоборот. В таком случае нужно принять ответ классификатора Махаланобиса, т.к. за счёт матрицы ковариации расстояние Махаланобиса учитывает ещё и форму распределения точек вокруг центра масс. По сути расстояние Махаланобиса – это просто расстояние между заданной точкой и центром масс, делённое на ширину эллипсойда в направлении заданной точки.

Рассмотрим далее принцип максимума правдоподобия, который составляет основу параметрического подхода. Пусть задано множество объектов $X m = \{x_1, ..., x_m\}$, выбранных независимо друг от друга из вероятностного распределения с плотностью φ(x;θ). Функцией правдоподобия называется совместная плотность распределения всех объектов выборки:

$$p(X m;\theta) = p(x_1, ..., x_m;\theta) = \prod_{i=1}^{m} \varphi(x_i;\theta). \qquad (1.5)$$

Значение параметра θ, при котором выборка максимально правдоподобна, то есть функция $p(X^m;\theta)$ принимает максимальное значение, называется оценкой макисмума правдоподобия.

В случае гауссовской плотности с параметрами θ = (m, S) задача максимизации правдоподобия имеет аналитическое решение:

$$m_{ML} = \frac{1}{N} \sum_{i=1}^{N} x_i, \qquad (1.6)$$

$$S_{ML} = \frac{1}{N} \sum_{i=1}^{N} (x_i - m_{ML})(x_i - m_{ML})^T. \qquad (1.7)$$

Пример 6. Рассмотрим на примере как работает алгоритм максимума правдоподобия. Сгенерируем 50 точек, имеющих по две координаты каждая, используя гауссово распределение с центром m = $[2, -2]^T$, $S = \begin{pmatrix} 0.9 & 0.2 \\ 0.2 & \end{pmatrix}$. Потом оценим параметры этого распределения по формулам (1.6) и (1.7), сравним результаты.

```
close('all');
clear;
% Генерируем 50 точек с заданным распределением гаусса
randn('seed',0);
m = [2 -2]; S = [0.9 0.2; 0.2 .3];
X = mvnrnd(m,S,50)';
% Оцениваем параметры распределения по выборке
[m_hat, S_hat]=Gaussian_ML_estimate(X)
```

Функцию можно создать в отдельном файле Matlab.

```
function [m_hat,S_hat]=Gaussian_ML_estimate(X)
% Входной параметр:
%  X:     lxN матрица, чьи колонки есть наши данные.
% Выходной аргумент:
%  m_hat: l-размерная оценка среднего вектора распределения.
%  S_hat: lxl оценка ковариационной матрицы распределения.
 [l,N]=size(X); % l = 2, N = 50
m_hat=(1/N)*sum(X')'; % получаем среднеарифметический центр масс
% создаём и инициализируем 0 квадратную матрицу 2x2
S_hat=zeros(l);
for k=1:N
   S_hat=S_hat+(X(:,k)-m_hat)*(X(:,k)-m_hat)';
end
% Высчитываем среднее
S_hat=(1/N)*S_hat;
```

По результатам получился вектор m_hat = $[2.0495, -1.9418]^T$,

$S\_hat = \begin{matrix} 0.8082 & 0.0885 \\ 0.0885 & \end{matrix}$.

Как видно, получившиеся оценки близки к оригиналу, однако, нужно учесть, что они проводились на выборке, состоящей из 50 элементов, что очень мало, поэтому эти оценки не надёжны. Для увеличения надёжности оценок нужно увеличить выборку.

Пример 7. Теперь рассмотрим последний пример в котором обобщим весь предыдущий материал. Сгенерируем два множества X и X_1 каждое содержит 999 трехмерных точек, X – обучающее множество, X_1 – тестовое. Каждое множество включает три равновероятных класса: w_1, w_2, w_3; классы создаются с помощью распределения Гаусса со средними $m_1 = [0, 0, 0]T$, $m_2 = [1, 2, 2]T$, $m_3 = [3, 3, 4]T$ и матрицами ковариации

$$S_1 = S_2 = S_3 = \begin{matrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.8 \end{matrix} = \sigma^2 I .$$

Используя X с помощью принципа максимального правдоподобия оценим такие параметры как среднее m и матрицы ковариации S_1, S_2, S_3, т.к. у нас 103 точек, то эти оценки будут надёжными. Далее, используя эти оценки проведём классификацию с помощью расстояния Махаланобиса, Евклида, а также байесовским классификатором. Для каждого способа вычислим ошибку вероятности и сравним результаты.

Сначала сгенерируем обучающее множество X.

```
close('all');
clear;
% Матрица m 3x3
```

```
m=[0 0 0; 1 2 2; 3 3 4]';
% Создаём матрицу S1 3x3 (единичную матрицу умножаем на 0.8)
S1=0.8*eye(3);
% Как переменная m содержит три вектора-центра масс, так и тут
% трехмерный массив S содержит три одинаковых матрицы ковариации
S(:,:,1)=S1;S(:,:,2)=S1;S(:,:,3)=S1;
% вероятность появления каждого из трех классов
P=[1/3 1/3 1/3]';
% количество элементов в обучающем множестве
N=1000;
% устанавливаем датчик случайных чисел с параметрами seed и 0.
randn('seed',0);
% Генерируем множество X[3;999] – сами точки, y[1, 999] – их метки
% от 1 до 333 – метка 1, от 334 до 666 – метка 2, от 667 до 999 – метка 3.
[X,y]=generate_gauss_classes(m,S,P,N);

% Функцию пишем в отдельном файле
function [X,y]=generate_gauss_classes(m,S,P,N)
[l,c]=size(m); % (l, c) = (3, 3)
X=[]; % Инициализация множеств
y=[];
for j=1:c % цикл от 1 до 3
% Генерируем трехмерные точки со средним m(:, j), где ":" – для всех строк,
% матрицей ковариации S(:, :, j) и количеством 1/3*1000, ограниченным снизу
  t=mvnrnd(m(:,j),S(:,:,j),fix(P(j)*N))';
  % сгенерированные точки присваиваем вектору X причем так, чтобы
  % они добавлялись в конец уже существующего вектора X. Сгенерировали
  % точки для j = 1, добавили их к пустому вектору, затем сгенерировали
  % точки для j = 2, добавили их в конец вектора, который уже содержит
  % точки для j = 1 и т.д.
  X=[X t]; % генерируем метки
  y=[y ones(1,fix(P(j)*N))*j];
end
```

Далее аналогично сгенерируем множество X1, но уже с другими параметрами для **randn()**.

```
% Генерируем тестовую выборку X1
randn('seed',100);
[X1,y1]=generate_gauss_classes(m,S,P,N);
```

Теперь по принципу максимума правдоподобия вычислим оценки для распределения, которое использовалось при создании множества X.

```
% извлекаем из X только те точки, которые имеют метку 1
class1_data=X(:,find(y==1));
% вычисляем центр масс и ковариационную матрицу
% функция для вычисления была разобрана ранее
[m1_hat, S1_hat]=Gaussian_ML_estimate(class1_data);
% Аналогично
class2_data=X(:,find(y==2));
[m2_hat, S2_hat]=Gaussian_ML_estimate(class2_data);
```

```
% Аналогично
class3_data=X(:,find(y==3));
[m3_hat, S3_hat]=Gaussian_ML_estimate(class3_data);
% Получаем общую оценку единой ковариационной матрицы,
% как среднеарифметическое трех уже вычисленных матриц
S_hat=(1/3)*(S1_hat+S2_hat+S3_hat);
m_hat=[m1_hat m2_hat m3_hat];
```

В итоге получаем $S\_hat = \begin{pmatrix} 0.8602 & 0.0212 & 0.0259 \\ 0.0212 & 0.8070 & 0.0134 \\ 0.0259 & 0.0134 & 0.8031 \end{pmatrix}$ и

$m\_hat = \begin{pmatrix} 0.0512 & 0.9468 & 3.0138 \\ 0.0150 & 2.0470 & 3.0130 \\ 0.0698 & 1.9889 & 3.9398 \end{pmatrix}$. Видно, что получившиеся оценки близки к тем, которые мы задавали изначально. В реальности S и m из условия задачи нам нужны были только для того, чтобы создать выборки. Предполагается, что выборки X и X_1 мы получаем откуда-то со стороны и делаем предположение, что они сформированы посредством нормального распределения. Такое предположение мы можем сделать, т.к. рисунки 1.2–1.9 демонстрируют, что гауссово распределение может моделировать широкий класс данных.

Теперь, зная характеристики выборки, точнее трёх классов, входящих в неё, можно провести классификацию точек из тестового множества X_1.

```
% высчитываем расстояние от точки X1[i], i=1..999 до точки-центра
% соответствующего класса, z_euclidean будет содержать 999 индексов классов
% с самым минимальным расстоянием
z_euclidean=euclidean_classifier(m_hat,X1);
% Аналогично
z_mahalanobis=mahalanobis_classifier(m_hat,S_hat,X1);
% Используем байесовский классификатор, однако с математической
% точки зрения, чтобы он работал корректно, мы должны подавать ему
% не вычисленные оценки множества X, а данные в условии, т.е. точные
z_bayesian=bayes_classifier(m,S,P,X1);
% Находим процент несовпадений по классам
err_euclidean = (1-length(find(y1==z_euclidean))/length(y1))
err_mahalanobis = (1-length(find(y1==z_mahalanobis))/length(y1))
err_bayesian = (1-length(find(y1==z_bayesian))/length(y1))


% Функция для классификатора байеса
function [z]=bayes_classifier(m,S,P,X)
[l,c]=size(m);
[l,N]=size(X);

for i=1:N % цикл от 1 до 999
   for j=1:c % цикл от 1 до 3
      t(j)=P(j)*comp_gauss_dens_val(m(:,j),S(:,:,j),X(:,i));
```

```
    end
    [num,z(i)]=max(t);
end
```

Получается, что во всех трех случаях ошибки похожи: 7.61%, 7.71%, 7.61%. Это не случайно, если выполняются 4 следующих условия, то все три метода классификации эквивалентны:

1. Все классы равновероятны;
2. Данные во всех классах имеют гауссовы распределения;
3. Матрицы ковариации одни и те же;
4. Ковариационные матрицы не просто равны, но и диагональные, причем все элементы на главной диагонали равны.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Задание для всех вариантов.

1. Вычислите как в примере 2 значения p1 и p2, но для следующих двух случаев: ($P(w_1)=1/6$, $P(w_2)=5/6$), ($P(w_1)=5/6$, $P(w_2)=1/6$). Дайте объяснения зависимости результатов классификации от априорных вероятностей.
2. Повторите пример 6, но для N = 500 и N = 5000 точек. Прокомментируйте результат.

В таблице 1.1 приведены индивидуальные задания.

Таблица 1.1 – Индивидуальные варианты

1	Повторите пример 7, но где $S = \begin{pmatrix} 0.8 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{pmatrix}$ и $\sigma^2 I$. Напишите вывод по получившемуся ответу.
2	Повторите пример 7, но где для генерации X и X_1 будут использованы следующие вероятности классов $P_1 = 1/2$, $P_2 = P_3 = 1/4$. Для этого случая байесовский классификатор должен показать лучший результат. Почему?
3	Повторите пример 7, но где P(w1) = P(w2) = P(w3) = 1/3 и $S = \begin{pmatrix} 0.8 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{pmatrix}$, $S_2 = \begin{pmatrix} 0.6 & 0.01 & 0.01 \\ 0.01 & 0.8 & 0.01 \\ 0.01 & 0.01 & 0.6 \end{pmatrix}$, $S_3 = \begin{pmatrix} 0.6 & 0.1 & 0.1 \\ 0.1 & 0.6 & 0.1 \\ 0.1 & 0.1 & 0.6 \end{pmatrix}$. Объясните результат.
4	Повторите пример 7, но где $S = \begin{pmatrix} 0.8 & 0.3 & 0.1 \\ 0.3 & 0.8 & 0.3 \\ 0.1 & 0.3 & 0.8 \end{pmatrix}$ и $\sigma^2 I$. Напишите вывод по получившемуся ответу.
5	Повторите пример 7, но где для генерации X и X_1 будут использованы следующие вероятности классов $P_1 = 0.8$, $P_2 = 0.15$, $P_3 = 0.05$. Прокомментируйте результат.
6	Повторите пример 7, но где P(w1) = P(w2) = P(w3) = 1/3 и $S = \begin{pmatrix} 0.9 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.9 \end{pmatrix}$, $S_2 = \begin{pmatrix} 0.6 & 0.01 & 0.01 \\ 0.01 & 0.8 & 0.01 \\ 0.01 & 0.01 & 0.6 \end{pmatrix}$, $S_3 = \begin{pmatrix} 0.6 & 0.2 & 0.1 \\ 0.1 & 0.6 & 0.1 \\ 0.1 & 0.2 & 0.6 \end{pmatrix}$. Объясните результат.
7	Повторите пример 7, но где $S = \begin{pmatrix} 0.7 & 0.2 & 0.4 \\ 0.2 & 0.7 & 0.2 \\ 0.4 & 0.2 & 0.7 \end{pmatrix}$ и $\sigma^2 I$. Напишите вывод по получившемуся ответу.
8	Повторите пример 7, но где для генерации X и X_1 будут использованы следующие вероятности классов $P_1 = 0.6$, $P_2 = P_3 = 0.2$ Для этого случая байесовский классификатор должен показать лучший результат. Почему?
9	Повторите пример 7, но где P(w1) = P(w2) = 0.2, P(w3) = 0.6 и $S = \begin{pmatrix} 0.8 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{pmatrix}$,

	^{1} 0.1 0.2 0.8 S_2 0.6 0.01 0.01 / 0.01 0.8 0.01 / 0.01 0.01 0.6, S_3 0.6 0.1 0.1 / 0.1 0.6 0.1 / 0.1 0.1 0.6. Объясните результат.
10	Повторите пример 7, но где для генерации X и X_1 будут использованы следующие вероятности классов $P_1 = 0.4$, $P_2 = P_3 = 0.3$. Для этого случая байесовский классификатор должен показать лучший результат. Почему?

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1) Название лабораторной работы и её номер.
2) ФИО и группу студента.
3) Формулировка индивидуального задания и номер варианта.
4) Ответ на задание в виде кода и результатов работы Matlab.
5) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Что такое матрица ковариации?
2. Что такое дисперсия и математическое ожидание?
3. Чем отличается классификация на основе расстояния Махаланобиса от классификации на основе расстояния Евклида?
4. Что такое байесовский классификатор?
5. Что такое принцип максимума правдоподобия?

Лабораторная работа № 2

Классификаторы, основанные на оценки функции оптимизации

Цель и содержание работы: познакомить студента с персептроном.

Задачи:

1) Разобрать принцип действия и алгоритм работы персептрона;
2) Разобрать алгоритм минимизации среднеквадратической ошибки (LMS).

Теоретическое обоснование

Техника получения оптимального байесовского классификатора основывается на оценке функции распределения Гаусса для обучающих данных. Однако, это сложная задача особенно для многомерных данных. Альтернативное решение заключается в отделении классов в многомерном пространстве с помощью разделяющих гиперплоскостей.

Разделяющая гиперплоскость может быть записана в виде

$$wTx\ w_0 = 0 . \tag{2.1}$$

Также (2.1) можно переписать в виде

$$w\,T\,x = [w^T, w_0]\,x = 0 , \tag{2.2}$$

где **w** – вектор настраиваемых параметров, w_0 – свободный член, **x** – входной вектор.

С помощью таких гиперплоскостей можно отделить линейно отделимые классы.

Алгоритм адаптации вектора весовых коэффициентов элементарного персептрона можно сформулировать следующим образом.

Если n-ый элемент x(n) обучающего множества корректно классифицирован с помощью весовых коэффициентов w(n), вычисленных на n-ом шаге алгоритма, то вектор весов не корректируется, т.е. действует следующее правило:

$$\begin{aligned} w(n+1) &= w(n), \text{ если } w^T x(n) > 0 \text{ и } x(n) \in C_1 \\ w(n+1) &= w(n), \text{ если } w^T x(n) < 0 \text{ и } x(n) \in C_2 \end{aligned} \tag{2.3}$$

В противном случае вектор весов персептрона подвергается коррекции в соответствии со следующим правилом:

$$\begin{aligned} w(n+1) &= w(n) - \eta(n)\, x(n), \text{ если } w^T(n)x(n) > 0 \text{ и } x(n) \in C_2 \\ w(n+1) &= w(n) + \eta(n)\, x(n), \text{ если } w^T(n)x(n) < 0 \text{ и } x(n) \in C_1, \end{aligned} \tag{2.4}$$

где C_1 и C_2 – линейно разделимые классы, а $\eta(n)$ – параметр скорости обучения.

Создадим функцию, которая будет обучать наш линейный нейрон (настраиваемую

гиперплоскость). Сигнатура функции будет следующей: **[w, iter, mis_clas] = perce(X, y, w_ini. rho)**, где X – это матрица размером (l+1)xN (l – размер входного вектора, N – количество паттернов для обучения), y – вектор меток для класса C_1 (+1) и C_2 (-1), w_ini – вектор начальных параметров, которые нужно настроить в процессе обучения так, чтобы гиперплоскость отделяла C_1 от C_2, rho – ━=const, т.е. скорость обучения, w – вектор, вычисленный алгоритмом, iter – количество итераций за которые был вычислен w, mis_clas – количество неправильно классифицированных векторов.

Рассмотрим пример.

Сгенерируем 4 множества X_i, каждое – это набор точек в двумерном пространстве. Каждая точка из X_i имеет метку -1, точки случайно разбросаны в квадрате [3, 5]x[3, 5], [2, 4]x[2, 4], [0, 2]x[2, 4], [1, 3]x[1, 3]. Точки, распложенные в квадрате [0, 2]x[0,2] имеют метку класса +1. Требуется визуально отобразить классы, обучить персептрон для ━=0.01 и ━=0.05, и начальном векторе параметров $[1, 1, -0.5]^T$.

Листинг функции, реализующий персептрон, приведён ниже.

```
function [w,iter,mis_clas]=perce(X,y,w_ini,rho)

[l,N]=size(X);
max_iter=20000; % Максимальное количество итераций,
% которые будет работать персептрон, после этого будет
% считаться, что решение не найдено.

w=w_ini;       % Инициализируем вектор параметров
iter=0;        % Счётчик итераций

mis_clas=N;    % Инициализируем количество неправильно
% классифицированных паттернов

% цикл while по двум условиям
while(mis_clas>0)&&(iter<max_iter)
   iter=iter+1;
   mis_clas=0;

   gradi=zeros(l,1); % Вычисление корректирующего компонента для вектора
   for i=1:N
      if((X(:,i)'*w)*y(i)<0)
         mis_clas=mis_clas+1;
         gradi=gradi+rho*(-y(i)*X(:,i));
      end
   end

   if(iter==1)
      fprintf('\n First Iteration: # Misclassified points = %g \n',mis_clas);
   end

   w=w-rho*gradi; % Обновление вектора параметров
```

```
end
```

Листинг основного кода приведён ниже.

```
close('all');
clear

rand('seed',0);

% Генерируем класс Xi
N=[100 100]; % 100 векторов для каждого класса
l=2; % Размерность входа каждого вектора

% для генерирования паттернов из X2, X3, X4 нужно вместо
% нижней строчки подставить одну из закомментированных строк
x=[3 3]';
% x=[2 2]'; для X2
% x=[0 2]'; для X3
% x=[1 1]'; для X4

% получаем 200 точек для Xi и для точек квадрата [0, 2]x[0, 2]
X1=[2*rand(l,N(1)) 2*rand(l,N(2))+x*ones(1,N(2))];
X1=[X1; ones(1,sum(N))];
y1=[-ones(1,N(1)) ones(1,N(2))]; % получаем метки

% 1. Выводим множество Xi
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'bo',...
X1(1,y1==-1),X1(2,y1==-1),'r.')
figure(1), axis equal

% 2. Запускаем алгоритм обучения персептрна для эта=0.01
rho=0.01; % Скорость обучения
w_ini=[1 1 -0.5]'; % начальные веса
[w,iter,mis_clas]=perce(X1,y1,w_ini,rho)
```

Имеем 4 ситуации X_i, i = 1..4, которые нужно отделить с помощью персептрона (рисунок 2.1 – 2.4).

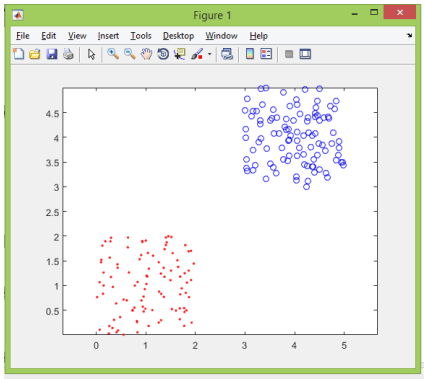


Рисунок 2.1 – Два класса для X_1

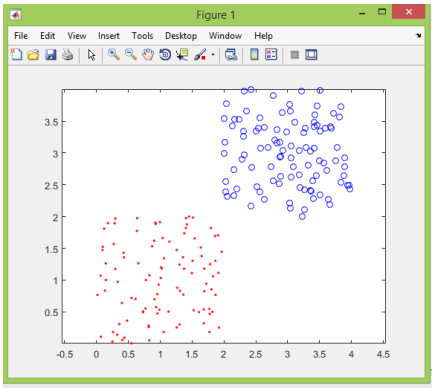


Рисунок 2.2 – Два класса для X_2

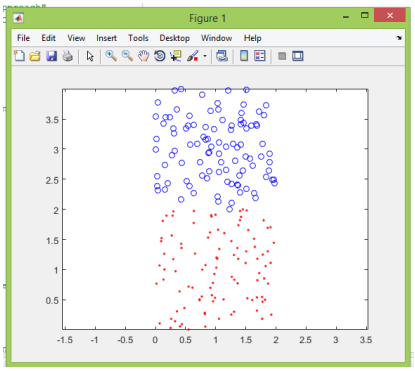


Рисунок 2.3 – Два класса для X_3

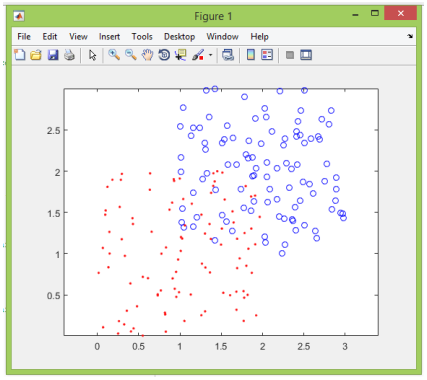


Рисунок 2.4 – Два класса для X_4

Видно, что в случае X_4 (рисунок 2.4) классы линейно не разделимы.

Результаты обучения приведены в таблице 2.1, 2.2.

Таблица 2.1 – Количество итераций, за которые персептрон находит решение в зависимости от ━

	X 1	X 2	X 3	X 4
━ = 0.01	134	134	5441	Оптимального решения не найдено

━ = 0.05	5	5	252	Оптимального решения не найдено

Таблица 2.2 – Количество неправильно распознанных паттернов

	X 1	X 2	X 3	X 4
━ = 0.01, ━ = 0.05	0	0	0	25

Видно, что при увеличении ━ увеличивается скорость обучения персептрона. В последнем случае оптимального решения персептрон не находит, поэтому проведённая прямая отделяет не все паттерны класса.

Функция **perce(X,y,w_ini,rho)** запрограммирована для пакетного режима, т.е. вектор **w** корректируется один раз после вычисления поправок для каждого примера. Ниже приведён листинг функции, где персептрон обучается в онлайн-режиме, где корректировки вычисляются для каждого примера, но далее, они не накапливаются, а сразу применяются к вектору весов.

```
function [w,iter,mis_clas]=perce_online(X,y,w_ini,rho)

[l,N]=size(X);
max_iter=10000000; % максимальное количество итераций
% требуется, если классы расположены близко друг к другу

w=w_ini;
iter=0;

mis_clas=N;

while(mis_clas>0)&&(iter<max_iter)
   mis_clas=0;

   for i=1:N
      if((X(:,i)'*w)*y(i)<0)
         mis_clas=mis_clas+1;
         w=w+rho*y(i)*X(:,i);  % Обновляем вектор весов
      end
      iter=iter+1;
   end

   if(iter==1)
      mis_clas
   end
end
```

Вызвать эту функцию можно с помощью следующей сигнатуры:

[w,iter,mis_clas]=perce_online(X1,y1,w_ini,rho).

Если поставить предыдущий эксперимент, то получим следующие результаты, таблица 2.3, 2.4.

Таблица 2.3 – Количество итераций, за которые персептрон находит решение в зависимости от ━

	X_1	X_2	X_3	X_4
━ = 0.01	600	600	6589400	Оптимального решения не найдено
━ = 0.05	400	400	7729200	Оптимального решения не найдено

Таблица 2.4 – Количество неправильно распознанных паттернов

	X_1	X_2	X_3	X_4
━ = 0.01, ━ = 0.05	0	0	0	6

Требуется значительно больше итераций, онлайн-обучение хуже сходится, зато в практическом плане более экономное, т.к. не требует специальных структур для хранения и накопления поправок.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

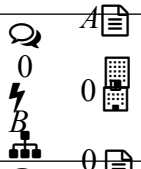
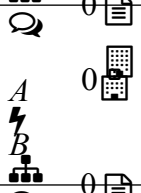
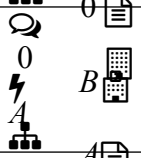
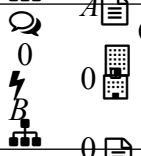
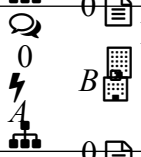
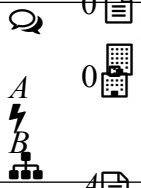
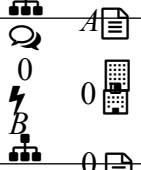
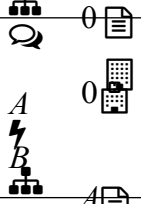
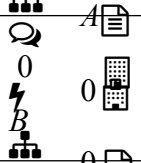
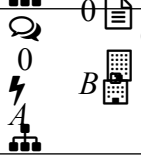
Методика и порядок выполнения работы

Создать два частично пересекающихся класса точек (100 точек для каждого класса) как

показано в таблице 2.5. Разделить их насколько это возможно с помощью персептрона. Тип

обучения персептрона онлайн или пакетный указан в таблице. Нарисовать найденное решение.

Таблица 2.5 – Индивидуальные варианты

1	 Онлайн обучение.
2	 Пакетное обучение.
3	 Пакетное обучение.
4	 Онлайн обучение.
5	 Пакетное обучение.
6	 Пакетное обучение.
7	 Онлайн обучение.
8	 Онлайн обучение.
9	 Пакетное обучение.
10	 Онлайн обучение.

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1) Название лабораторной работы и её номер.
2) ФИО и группу студента.

3) Формулировка индивидуального задания и номер варианта.
4) Ответ на задание в виде кода и результатов работы Matlab.

Вопросы для защиты работы

1. Что такое персептрон?
2. Чем персептрон отличается от сети прямого распространения?
3. Какие задачи можно решать с помощью персептрона?

Лабораторная работа № 3

Создание и обучение нейронной сети на языке высокого уровня среды Matlab

Цель и содержание работы: создать и обучить несколько нейронных сетей, решающих задачи линейного и нелинейного разделения классов, используя встроенный язык программирования Matlab.

Задачи:

- освоить базовые команды для создания и обучения простых сетей в Matlab;
- научиться создавать и обучать сети для линейного и нелинейного разделения классов;
- научиться строить графики, отражающие результаты работы сетей.

Теоретическое обоснование

Самый гибкий способ по работе с ИНС в Matlab – это использовать встроенный язык программирования для создания и обучения сетей. В данной лабораторной работе на примере линейного разделения классов и задачи XOR будет рассмотрена техника создания и обучения разных нейронных сетей.

Код Matlab будем выделять жирным шрифтом. Комментарий к коду записывается после символа «%» и идёт до конца строчки. При желании, можно получить более подробную справку о любой функции, установив курсор на нужной и нажав клавишу «F1».

Прежде, чем программировать персептрон, рассмотрим пример построения поверхности отклика для обычного нелинейного нейрона с функцией активации – гиперболический тангенс.

```
% Готовим Matlab к работе
close all, clear all, clc, format compact;
% Задаём веса нейрона
w = [4 -2];
% Задаём смещение
b = -3;
% Задаём функцию, которую хотим построить, гипербол. тангенс
func = 'tansig';
% Задаём входной вектор
p = [2 3];
% Вычисляем взвешенную сумму входа и весов
activation_potential = p*w'+b;
% Вычисляем выход нелинейной части нейрона
```

```
neuron_output = feval(func, activation_potential);
% Задаём входной вектор для диапазона от -10 до +10 с шагом 0.25
[p1, p2] = meshgrid(-10:.25:10);
% Вычисляем вектор выхода нейрона
z = feval(func, [p1(:) p2(:)]*w'+b);
% Пересчёт вещественных чисел в воксели для 3D-графика
z = reshape(z, length(p1), length(p2));
% Строим график
plot3(p1, p2, z);
% Включаем сетку
grid on;
% Подписываем оси графика
xlabel('Input 1');
ylabel('Input 2');
zlabel('Neuron output');
```

Разберём код. Команда **close all** закрывает все вспомогательные окна, которые могли быть открыты (окна различных мастеров, вроде nntool, не закрываются). **Clear all** удаляет все переменные из области workspace, **clc** – очищает область окна Command window, где будет набираться код. **Format compact** устанавливает формат отображения для чисел, так для вещественных числе будет отображаться 4 точки после запятой.

Веса задаются обычным вектором **w = [4 -2]**, как видно, значения отделяются друг от друга пробелом, указывать тип не обязательно. Функция активации задаётся строкой **'tansig'**. Далее идёт вычисление взвешенной суммы: скалярное произведение входа на веса и прибавка смещения. Вектора **p** и **w** нельзя просто так взять и перемножить, т.к. по правилам линейной алгебры один из них должен быть транспонирован, что и делается с вектором **w** с помощью **w'**. Можно посмотреть значение переменной, оно должно быть -1. Далее вычисляем функцию активации при входе равном -1. Делаем это через **feval()**. Как понятно из названия, функция вычисляет значение некоторой функции (первый параметр) от вектора входов (второй параметр). Причём, как и многие другие фунции Matlab, она перегружена, т.е. её вызов может происходить при разных способах записи второго параметра. В первом случае вектор редуцируется к скалярной величине, выход равен -0.7616.

Далее присваиваем переменным **p1** и **p2** значения от -10 до +10 с шагом 0.25. Делаем это через **meshgrid()**, которая создаёт прямоугольную сетку в специальном формате пригодном для построения графиков. Но для построения самого графика нам нужно знать значения не только **p1** и **p2**, но и выхода нейрона. Теперь вычисляем эти значения для вектора **z** с помощью **feval()**, видно, что теперь второй параметр напрямую записывается в виде скалярного произведения двух векторов плюс смещение. Двоеточие означает перевод значения в формат double (используется для научных и инженерных вычислений. Никогда

не используйте float для этих задач в таких языках как C/C++, чтобы избежать переполнения переменной).

Функция **reshape()** пересчитывает числовые значения в воксели для отображения на графике. **Plot3()** строит трёхмерный график, рисунок 3.1. **Grid on** включает сетку на этом графике, рисунок 3.2, 3.3. **Xlabel()**, **ylabel()**, **zlabel()** – подписывают оси у графика.

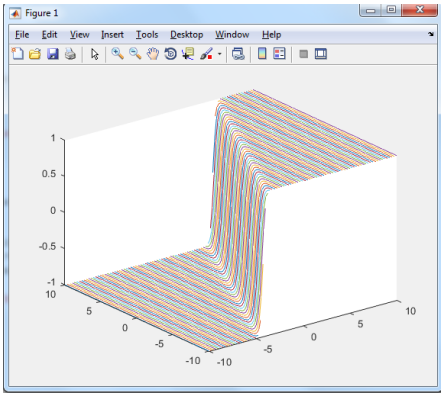


Рисунок 3.1 – Получившийся 3D-гарфик без сетки и подписей осей

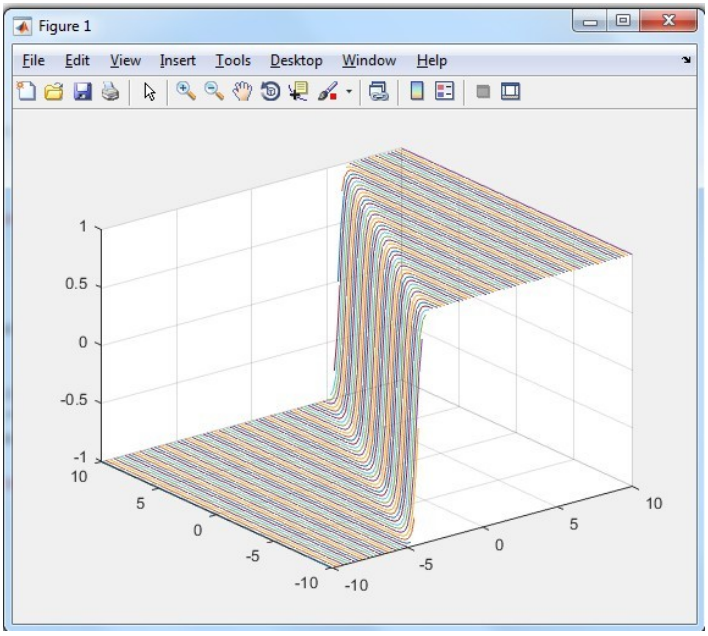


Рисунок 3.2 – Добавили сетку на фон

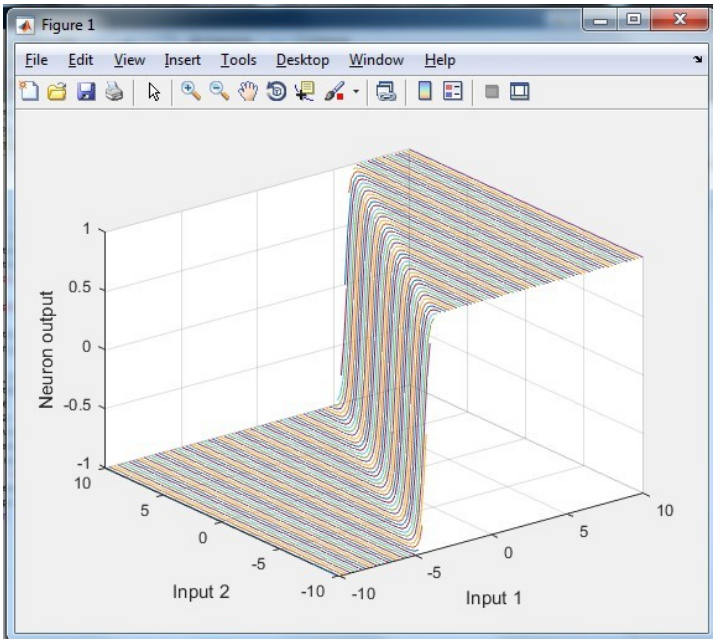


Рисунок 3.3 – Добавили подписи для осей x, y, z

Теперь создадим и обучим двухслойную сеть на одном примере.

```
close all, clear all, clc, format compact
% Входной транспонированный вектор
inputs = [1:6]'
% Выходной вектор, который должна промоделировать сеть
outputs = [1 2]';
% Задаём общую структуру сети
net = network(1, 2, [1; 0], [1; 0], [0 0; 1 0], [0 1]);
% Показываем её
view(net);
% Задаём количество промежуточных слоёв
net.layers{1}.size = 5;
% Задаём функции активации на них
net.layers{1}.transferFcn = 'logsig';
% Ещё раз отображаем сеть
view(net);
% Устанавливаем размерность входа и выхода
net = configure(net, inputs, outputs);
% Смотрим окончательный вариант сети
view(net);
% Получаем изначальный выход сети без обучения
initial_output = net(inputs);
% Устанавливаем метод обучения
net.trainFcn = 'trainlm';
% Устанавливаем функцию ошибки
net.performFcn = 'mse';
% Обучаем сеть на одном примере
net = train(net, inputs, outputs);
% Опять подаём вход, но уже на обученную сеть,
% сеть должна промоделировать выход [1 2]
final_output = net(inputs);
```

Входной вектор будет равен массиву из шести элементов от 1 до 6, т.к. диапазон записывается через «:». Функция **network()** задаёт общую структуру сети. Рассмотрим более подробно, что означает каждый параметр.

Первый параметр означает количество входов (не в смысле размерность входного вектора, а в смысле, – сколько векторов будет подано на вход сети. Допустим, в сверточных сетях обычное дело, что на вход подаётся не один, а два или три массива, каждый из которых как-то связан с дальнейшим слоем). У нас этот параметр равен 1. Второй параметр означает количество слоёв в сети. В данном случае имеем двухслойную сеть. Третий параметр – булев вектор, который означает какие нейроны будут иметь смещение, а какие – нет. Т.к. вектор [1; 0], то очевидно, что все нейроны первого слоя будут иметь смещение, а нейроны второго слоя – нет. Четвёртый параметр – это также булев вектор, который означает, – с какими слоями связан вход. В данном случае используется классическая схема, где вход связна только с последующим слоем, а следовательно, вектор [1; 0]. Пятый параметр – это матрица связей между слоями, которая означает, какой слой с каким связан. Она записывается в виде вектора. Имеем [0 0; 1 0], если записать в виде матрицы, то получится $\begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix}$. Тогда понятно, что первый слой связан со вторым слоем, т.к. единица располагается в позиции (1, 2), где 1 – это номер столбца, 2 – номер строки. Шестой параметр [0 1] – булев вектор, который показывает, с какими слоями связан выход. При таких значениях выход связан с предпоследним слоем.

При такой конфигурации получим структуру сети как на рисунке 3.4.

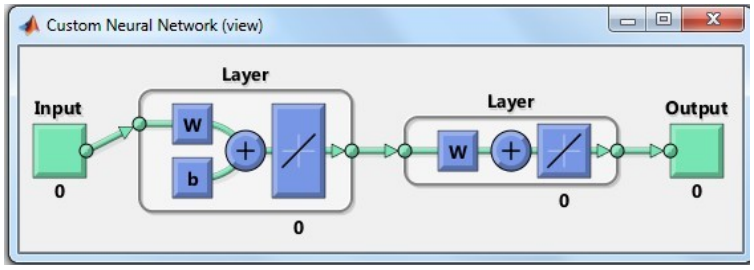


Рисунок 3.4 – Изначальная структура сети

Изменим третий параметр с [1; 0] на [1; 1], получим структуру как на рисунке 3.5.

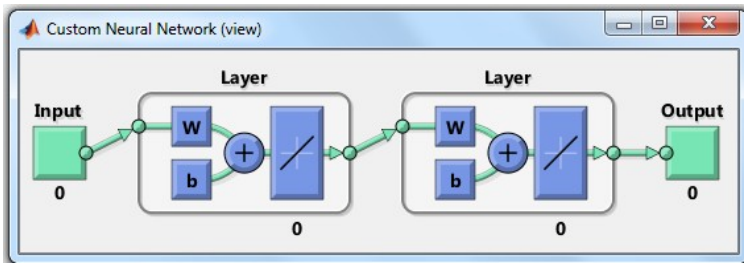


Рисунок 3.5 – Видно, что у второго слоя тоже появилось смещение

Изменим теперь и четвёртый параметр на [1; 1], получим структуру сети как на рисунке 3.6.

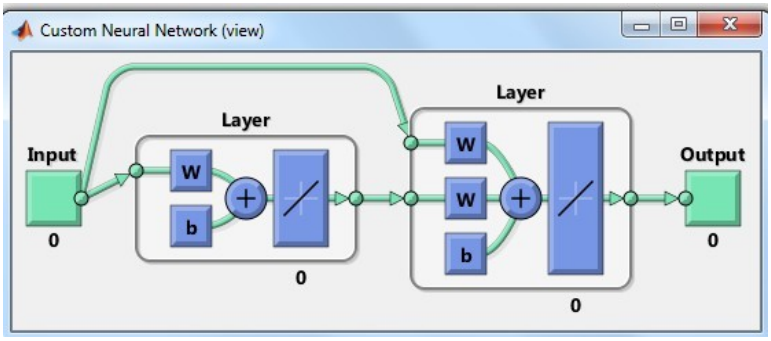


Рисунок 3.6 – Добавилась связь входа со вторым слоем

Изменим остальные два вектора на единичные, получим структуру как на рисунке 3.7.

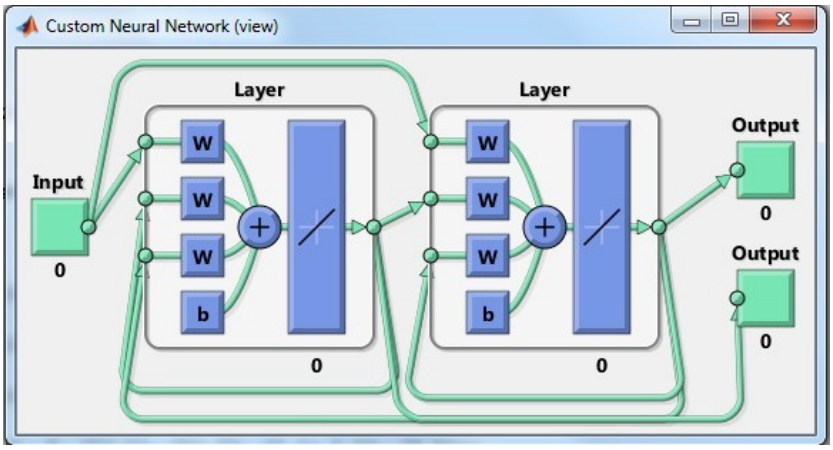


Рисунок 3.7 – Появился второй выход, связанный с первым слоем, а также каждый слой связан с самим собой и второй слой связан с первым (подаёт свои сигналы на вход первого слоя)

Net.layers{1}.size = 5 – присваиваем количество нейронов первому слою сети (индекс записывается в фигурных скобках). Видно, что обращение к данным параметрам осуществляется классическим образом для объектов: через точку. Т.е. сеть в Matlab – это объект (по терминологии объектно-ориентированного программирования (ООП)).

Небольшое примечание. Концепция ООП напрашивается для описания ИНС, т.к. вполне логично, что объект «сеть» включает в себя объекты – «слой», а те – «нейрон». Однако, нужно помнить, что при таком подходе оперативная память выделяется и группируется под конкретные объекты и в случае попыток быстрой реализации ИНС,

возможно, с использованием ассемблера, не получится быстро реализовывать матричные операции. Короче, если требуется быстрая реализация сети, то от ООП-описания придётся отказаться.

Net.layers{1}.transferFcn = 'logsig' – присваивает слоям первого слоя функцию активации сигмойд (без отрицательных значений), рисунок 3.8.

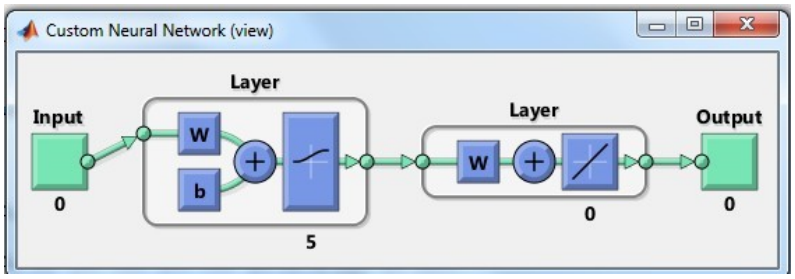


Рисунок 3.8 – Меняем функцию активации у нейронов первого слоя

Для установки входного и выходного слоя можно сконфигурировать сеть по отношению к вектору входа и выхода: **net = configure(net, inputs, outputs)**, рисунок 3.9.

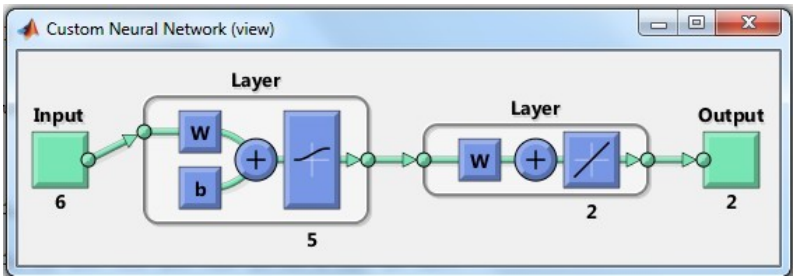


Рисунок 3.9 – Окончательно получаем общую структуру сети, где размер входа – 6, а выхода – 2

Вектор **initial_output** получит значения $[0\ 0]^T$, т.к. веса не инициализированы, т.е. сеть ничего делать не умеет. Для начала обучения нужно как минимум задать метод обучения и функцию ошибки, что и делается с помощью **net.trainFcn = 'trainlm'** и **net.performFcn = 'mse'**. **Trainlm** – это обучение по методу Левенберга-Марквардта, а **MSE** (mean square error) – это среднеквадратическая ошибка. **Train()** – запускает обучение сети и возвращает объект – обученную сеть. Процесс обучения показан на рисунке 3.10.

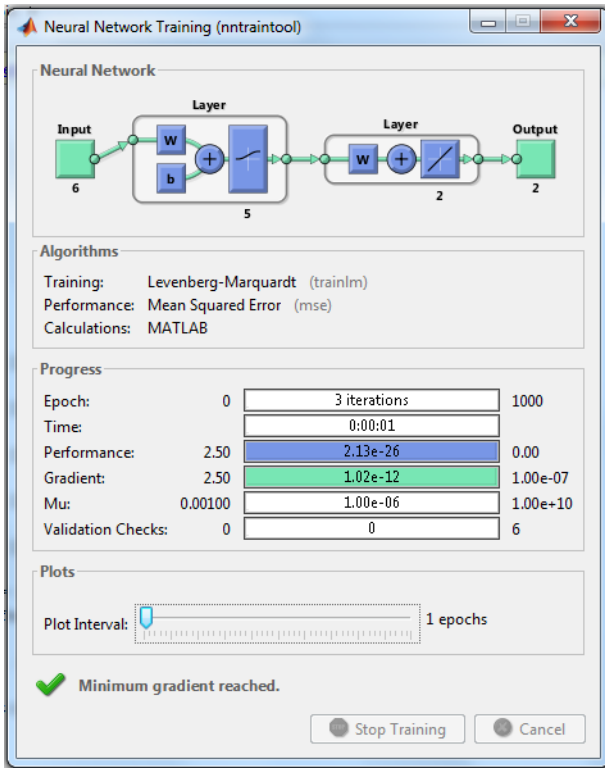


Рисунок 3.10 – Обучение сети

Позже формируем вектор **final_output**, который содержит ответ сети на вход, т.е. 1.0000 и 2.0000 (помним про формат **compact**).

Теперь решим задачу о разделении классов с помощью персептрона, подобную той, которая была решена в лабораторной работе 3.

```
close all, clear all, clc, format compact
% Задаём количество паттернов каждого класса
N = 20;
% Задаём смещение
offset = 5;
% Создаём входные значения каждого класса
x = [randn(2, N) randn(2, N)+offset];
% Создаём выходные значения для этих классов
y = [zeros(1, N) ones(1, N)];
% Открываем окно для рисования
figure(1);
% Наносим точки (паттерны)
plotpv(x, y);
% Создаём объект-сеть типа Персептрон
net = perceptron;
% Обучаем персептрон
net = train(net, x, y);
% Рисуем общую структуру сети
view(net);
% Возвращаемся в окно для рисования и рисуем прямую линию,
% которая разделит два класса
figure(1);
```

plotpc(net.IW{1}, net.b{1});

rand(2, N) – создаёт массив размером 2xN, числа которого являются случайные величины, распределённые по нормальному закону с математическим ожиданием 0 и среднеквадратическим отклонением 1. Таким образом, будут созданы две матрицы 2xN, содержащие координаты точек-паттернов в двумерном пространстве, причём координаты второй матрицы будут смещены на 5 позиций, т.е. заранее гарантируется, что классы будут линейно разделимые. Обратим внимание, что эти две матрицы на самом деле записаны в одну матрицу размером 2x2*N. **Zeros(1, N)** и **ones(1, N)** создаёт массив нулей и единиц размером N, это метки для входных паттернов.

Figure(1) – вызывает окно, содержащее канву для рисования, рисунок 3.11.

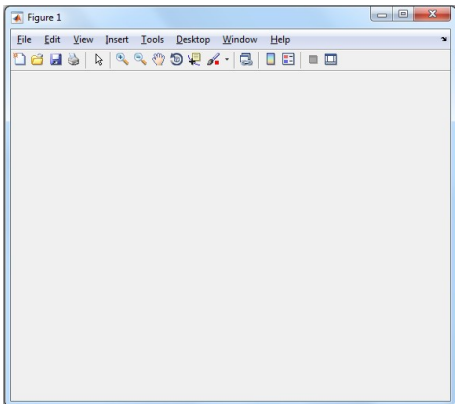


Рисунок 3.11 – Окно для рисования

Plotpv(x, y) – наносит точки на канву с координатами x и метками y, рисунок 3.12. Причём нули отображаются кружками, а единицы плюсами.

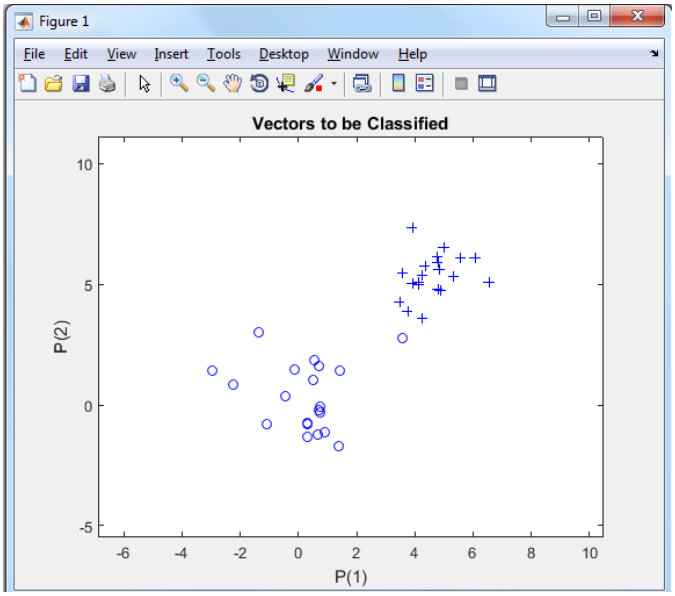


Рисунок 3.12 – Два созданных нами класса, видно, что они линейно разделимы

Net = perceptron – создаём сеть типа Персептрон, если после записи команды не ставить точку с запятой, то среда развернёт все установленные по умолчанию параметры, рисунок 3.13.

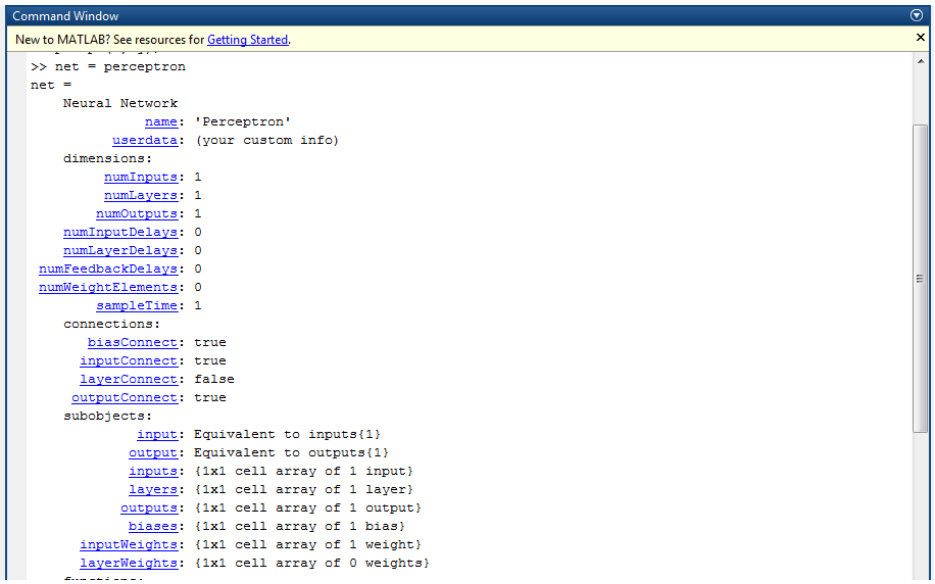


Рисунок 3.13 – Методы и свойства объекта perceptron

Далее происходит обучение сети. Общая структура сети представлена на рисунке 3.14.

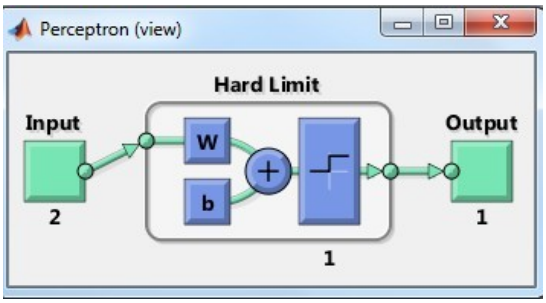


Рисунок 3.14 – Общая структура сети

Результаты обучения представлены на рисунке 3.15. Видно, что потребовалось 29 итераций.

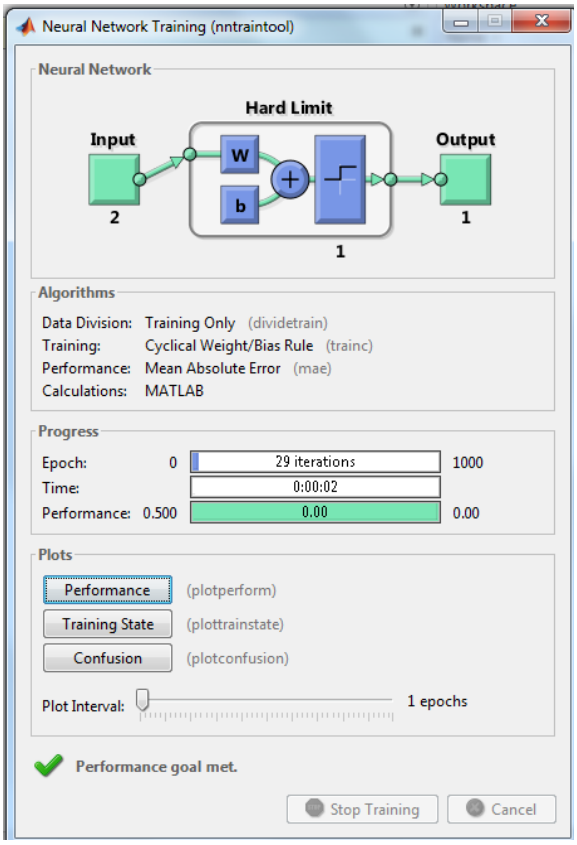


Рисунок 3.15 – Результаты обучения персептрона

На рисунке 3.16 представлен график настройки персептрона. График носит характерную зубчатую форму. После каждой эпохи высчитывается MSE между предполагаемыми ответами и реальными. Напомним, что при адаптации не используется спуск по поверхности ошибки, поэтому ждать постепенного снижения ошибки обучения не следует. Обучение идёт до тех пор, пока не будет найдена первая прямая, отделяющая один класс от другого.

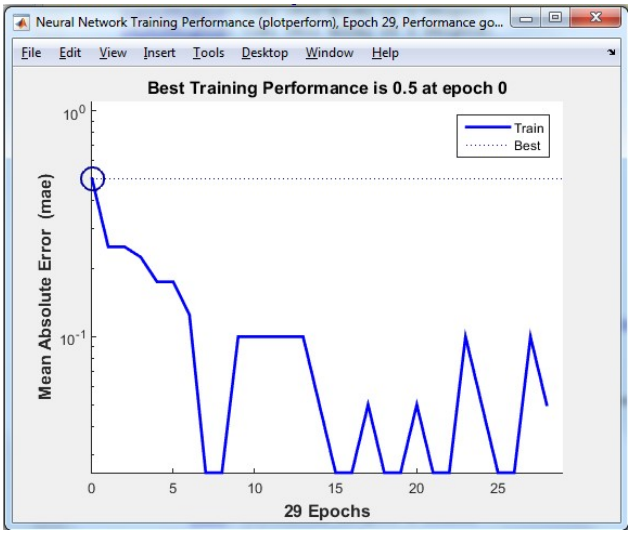


Рисунок 3.16 – График настройки весовых коэффициентов персептрона

Plotpc(net.IW{1}, net.b{1}) – строит прямую линию с соответствующими коэффициентами. **IW** – обозначает слой коэффициентов между входом и первым слоем, **b{1}** – смещение первого слоя. Итоговая прямая показана на рисунку 3.17.

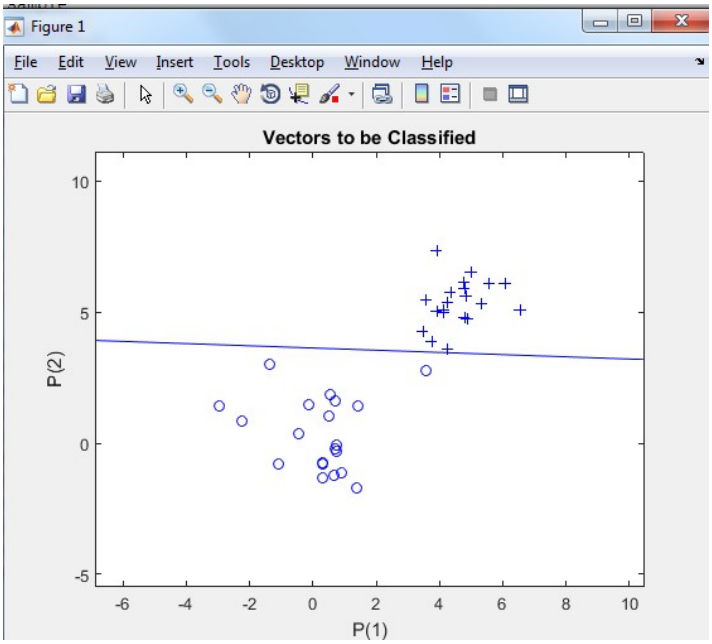


Рисунок 3.17 – Полученное решение по разделению двух классов

Теперь рассмотрим решение задачи разделения на 4 класса с помощью персептрона и обычной многослойной сети прямого распространения. Расположение классов оставим таким же, как и в задаче XOR (по углам «квадрата»), но теперь у нас не два класса, а 4, а также каждый класс представлен 30 паттернами.

Решение этой задачи с помощью персептрона.

```
close all, clear all, clc, format compact
% Каждый класс имеет 30 паттернов
K = 30;
% Смещение для классов 0.6
q = .6;
% Задаём 4 класса в виде векторов A, B, C, D (2 строки и 30 столбцов)
A = [rand(1, K)-q; rand(1, K)+q];
B = [rand(1, K)+q; rand(1, K)+q];
C = [rand(1, K)+q; rand(1, K)-q];
D = [rand(1, K)-q; rand(1, K)-q];
% Рисуем на канве точки A определённого типа (третий параметр)
plot(A(1, :), A(2, :), 'bs');
% Фиксируем канву
hold on;
grid on;
% На той же канве рисуем остальные классы
plot(B(1, :), B(2, :), 'r+');
plot(C(1, :), C(2, :), 'go');
plot(D(1, :), D(2, :), 'm*');
```

```
text(.5-q, .5+2*q, 'Class A');
text(.5+q, .5+2*q, 'Class B');
text(.5+q, .5-2*q, 'Class C');
text(.5-q, .5-2*q, 'Class D');
% Задаём кодировку классов (метки)
a = [0 1]';
b = [1 1]';
c = [1 0]';
d = [0 0]';
% Получаем общий вектор входных значений
P = [A B C D];
% Получаем общий вектор меток
T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(B)) repmat(d, 1, length(D))];
% Сеть типа «персептрон»
net = perceptron;
% Пусть среднеквадратическая ошибка не равна 0
E = 1;
% Установка параметра о том, что будет минимум один прогон
net.adaptParam.passes = 1;
% Построили изначально ответ персептрона, до обучения
linehandle = plotpc(net.IW{1}, net.b{1});
% Счётчик по циклам
n = 0;
% Цикл будет идти пока среднеквадратическая ошибка не
% станет равной нулю или пока не достигнем значения итераций в 1000
while (sse(E) & n<1000)
        n = n + 1;
        % Обучить персептрон
        [net, Y, E] = adapt(net, P,
        T);
        % Нарисовать линии
        linehandle = plotpc(net.IW{1}, net.b{1}, linehandle);
        drawnow;
end
view(net);
% Проверка работы персептрона на входном векторе
p = [0.7; 1.2];
% Выдаст класс (1; 1)
y = net(p);
```

rand(1, K) – генерирует числа из равномерного распределения в виде вектора (первый параметр) размером K, в итоге **A = [rand(1, K)-q; rand(1, K)+q]** – генерируется матрица 2xK, что соответствует координатам точек.

Plot(A(1, :), A(2, :), 'bs') – рисует на канве точки из матрицы A, переводя их в вещественный формат. Параметр **'bs'** – это цвет и форма точек: b – синий и s – квадрат. **'r+'** – красный (r) крест (+), **'go'** – зелёный (g) кружок (o), **'m*'** – пурпурная (m) звездочка (*).

Text(.5-q, .5+2*q, 'Class A') – наносит текст на канву с координатами, записанными как два первых параметра.

Общий вид отображения классов представлен на рисунке 3.18.

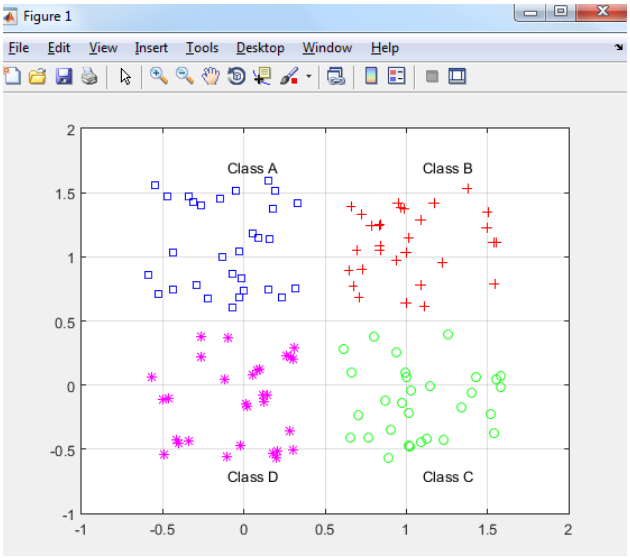


Рисунок 3.18 – Классы, которые предстоит разделить

Для получения вектора T необходимо раскопировать транспонированные вектора **a**, **b**, **c**, **d**. Для этой цели используется **repmat()**: **repmat(что копируем, копий по строкам, копий по столбцам)**. В результате вектор **T** примет вид как на рисунке 3.19: матрица из двух строк и 30 * 4 = 120 колонок.

```
>>  T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(C)) repmat(d, 1, length(D))]
T =
  Columns 1 through 17
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
  Columns 18 through 34
     0     0     0     0     0     0     0     0     0     0     0     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
  Columns 35 through 51
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
  Columns 52 through 68
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     0     0     0     0     0     0     0     0
  Columns 69 through 85
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 86 through 102
     1     1     1     1     1     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 103 through 119
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  Column 120
     0
     0
```

Рисунок 3.19 – Общий вид вектора меток T

Цикл **while()** – это цикл с предусловие, поэтому он может не выполниться ни разу, а следовательно, перед ним необходимо построить возможное решение (т.к. при создании сети **net = perceptron**) веса получают случайные значения.

[net, Y, E] = adapt(net, P, T) – процесс адаптации весов. **Net** – получит обученную сеть, **Y** – вектор ответов сети для входа **P**, **E** – ошибки сети для меток **T** и входов **P**. **Sse(E)** –

сумма квадратов этих ошибок.

На рисунке 3.20 показана структура сети, видно, что единственный слой сети содержит два нейрона, что соответствует двум прямым линиям для разделения 4-х классов.

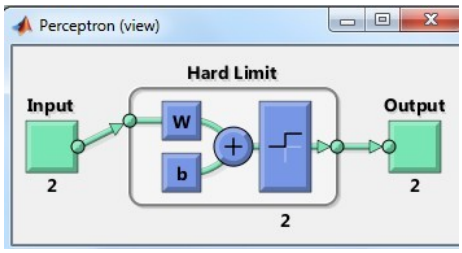


Рисунок 3.20 – Общая структура персептрона

Возможный ответ сети приведён на рисунке 3.21.

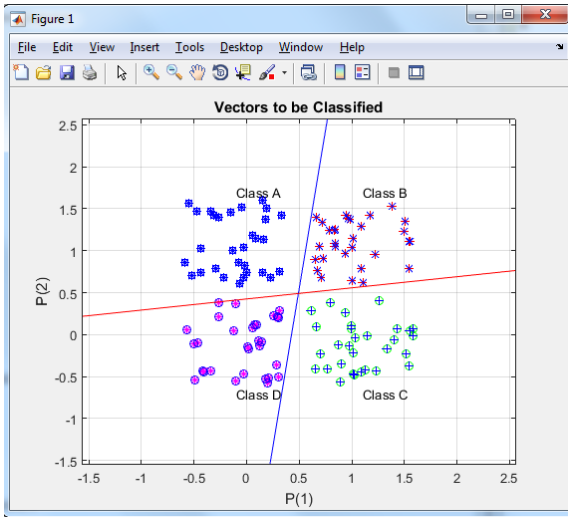


Рисунок 3.21 – Решения задачи разбития на 4 класса

Рассмотрим решение XOR-проблемы с помощью многослойной сети прямого распространения, но расширим представление каждого класса до 100 паттернов.

```
close all, clear all, clc, format compact
K = 100;
q = .6;
% Данные для обучения
%Обратите внимание на то, что randn() отделяются с помощью
%";". Если опустить этот символ, то A будет не 2x100, а 1x200.
A = [rand(1, K)-q; rand(1, K)+q];
B = [rand(1, K)+q; rand(1, K)+q];
C = [rand(1, K)+q; rand(1, K)-q];
D = [rand(1, K)-q; rand(1, K)-q];
figure(1);
% Наносим точки на канву
plot(A(1, :), A(2, :), 'k+');
hold on;
grid on;
plot(B(1, :), B(2, :), 'bd');
```

```
plot(C(1, :), C(2, :), 'k+');
plot(D(1, :), D(2, :), 'bd');
% Наносим названия классов
text(.5-q, .5+2*q, 'Class A');
text(.5+q, .5+2*q, 'Class B');
text(.5+q, .5-2*q, 'Class A');
text(.5-q, .5-2*q, 'Class B');
% Определяем метки для классов
a = -1, c = -1, b = 1, d = 1;
% Формируем входной вектор
P = [A B C D];
% Формируем вектор ответов
T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(C)) repmat(d, 1,
length(D))];
% Определяем сеть прямого распространения
net = feedforwardnet([5 3]);
% Вся выборка под обучающие примеры
net.divideParam.trainRatio = 1;
% Нет валидационной выборки
net.divideParam.valRatio = 0;
% Нет тестовой выборки
net.divideParam.testRatio = 0;
% Обучаем сеть
[net, tr, Y, E] = train(net, P, T);
view(net);
figure(2);
% График ожидаемых ответов сети
% график получается зубчатым, т.к. метки для классов
%определены как -1 и +1
plot(T', 'linewidth', 2);
hold on;
% График реальных ответов сети
plot(Y', 'r--');
grid on;
% Рисуем легенду к графикам
legend('Targets', 'Network response', 'location', 'best');
% Определяем видимый диапазон по оси y
ylim([-1.25 1.25]);
% Создаём  набор 601 числа
span = -1:.005:2;
% P1 и P2 станут матрицами размером 601x601
[P1, P2] = meshgrid(span, span);
% Транспонируем вектор резмером 601*601 = 361201
pp = [P1(:) P2(:)]';
% Получаем выходы сети
aa = net(pp);
% Возвращаемся к первой канве
figure(1);
% Рисуем трёхмерную сетку
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
% Подбираем лучшее сочетание цветов
colormap cool;
```

На рисунке 3.22 показаны паттерны входных классов.

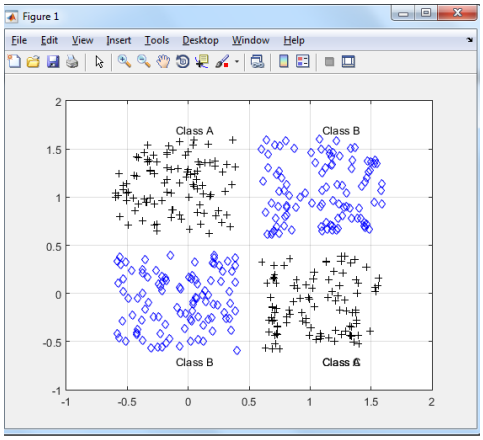


Рисунок 3.22 – Входные данные для двух классов, которые нужно разделить

Net = feedforwardnet([5 3]) – создаём сеть прямого распространения с двумя слоями, в первом будет 5 нейронов, во втором – 3. Структура сети показана на рисунке 3.23.

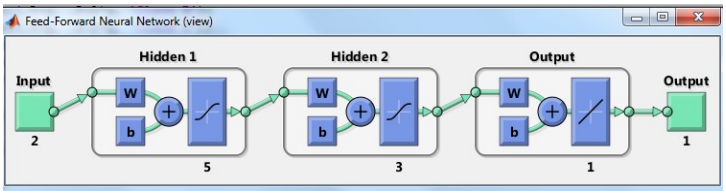


Рисунок 3.23 – Общая структура сети

Обратим внимание, что выходной нейрон имеет линейную функцию активации.

Все паттерны войдут в обучающую выборку, поэтому **net.divideParam.trainRatio = 1**, валидационную и тестовую выборку установим в 0.

Обучение сети осуществляется через **[net, tr, Y, E] = train(net, P, T)** – где **tr** – переменная, содержащая информацию об обучении (в среде Matlab её можно открыть и посмотреть на входящие в неё поля и их значения). Процесс обучения показан на рисунке 3.24, кривая обучения на рисунке 3.25.

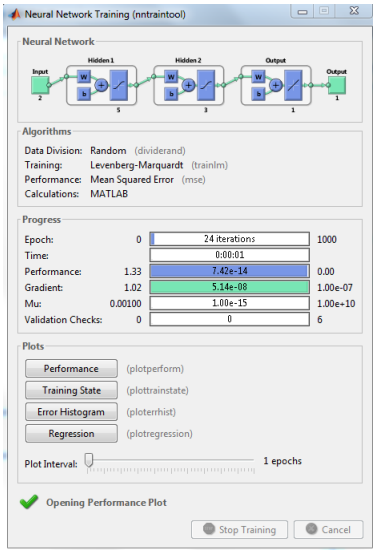


Рисунок 3.24 – Процесс обучения сети, потребовалось 24 итерации

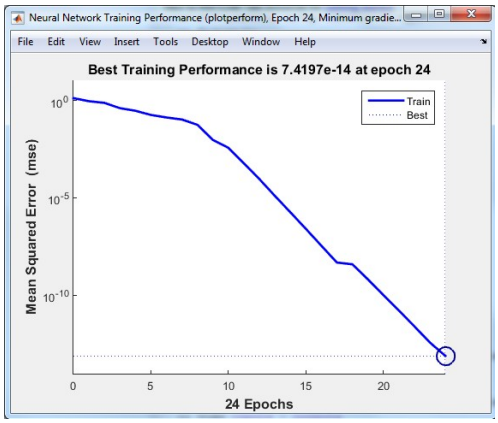


Рисунок 3.25 – Кривая обучения (MSE), лучший показатель меньше 10^{-10}

Далее отображаем на новой канве ожидаемый ответ сети, рисунок 3.26.

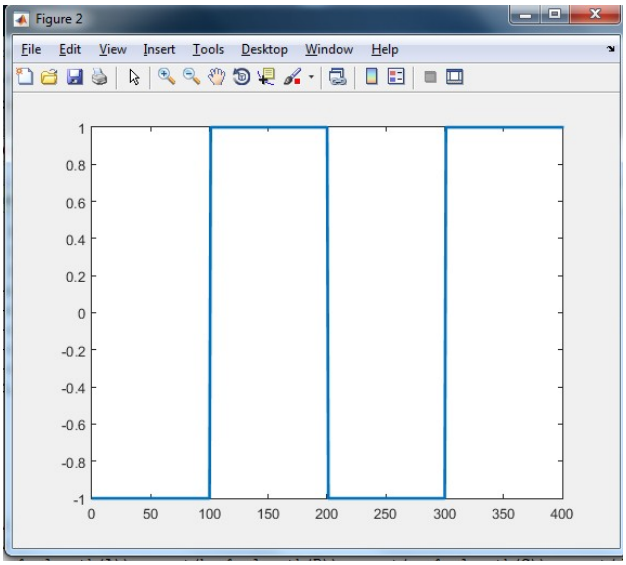


Рисунок 3.26 – Ожидаемый ответ сети (классы кодировались 1)

На той же канве отображаем реальный ответ сети, рисунок 3.27. Видно, что графики совпадают.

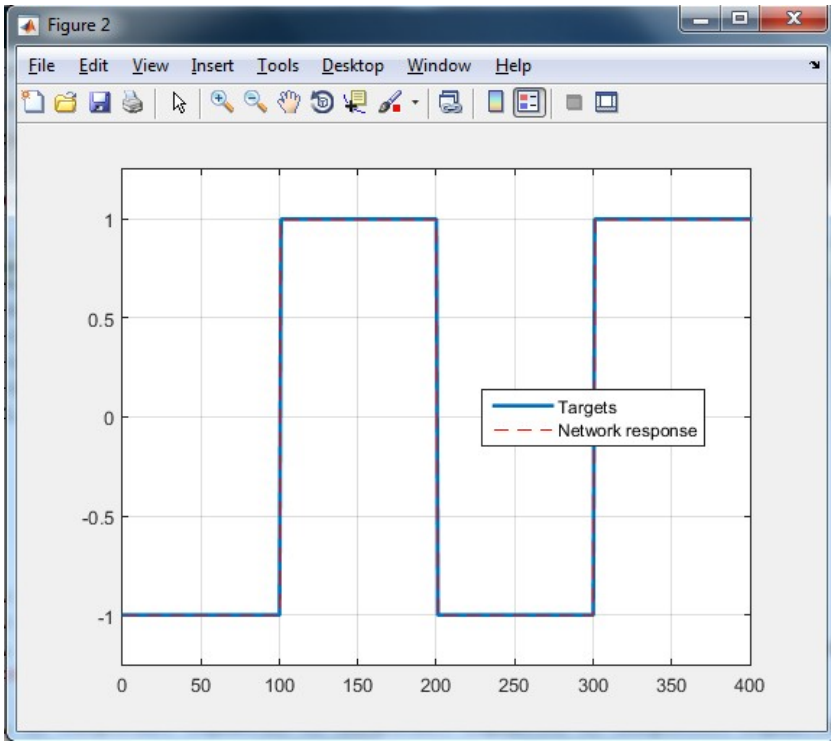


Рисунок 3.27 – Реальный ответ сети, графики совпадают

Ylim([-1.25 1.25]) – ограничивает ось OY данными значениями.

Чтобы отобразить получившееся решение уже нельзя просто построить прямые линии (как в случае с персептроном). Для этого нужно поверх первой канвы отобразить значения выходов сети на очень мелкой сетке (входные данные) и закрасить получившиеся ответы. Граница решения для разбиения двух классов окажется видна, рисунок 3.28.

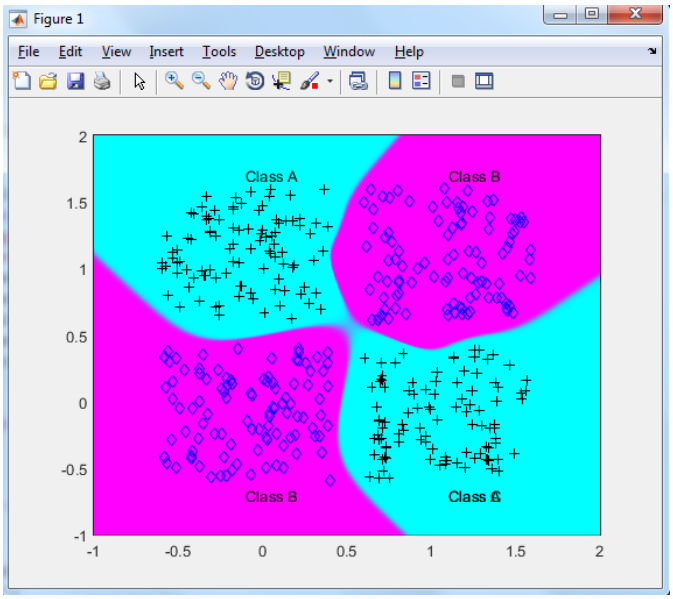


Рисунок 3.28 – Получившаяся граница решения для задачи XOR

Примечание. Сеть со структурой 2-5-3-1 явно избыточна для решения данной задачи. Можно переопределить структуру сети, как было показано в начале данной лабораторной работы:

```
net.layers{1}.size = 5;
% Получаем двухслойную сеть, где скрытый слой имеет 5 нейронов
net = configure(net, P, T);
% Меняем функцию активацию на выходном нейроне на нелинейную
net.layers{2}.transferFcn = 'tansig';
net.divideParam.trainRatio = 1;
net.divideParam.valRatio = 0;
net.divideParam.testRatio = 0;
[net, tr, Y, E] = train(net, P, T);
```

Структура такой сети показана на рисунке 3.29.

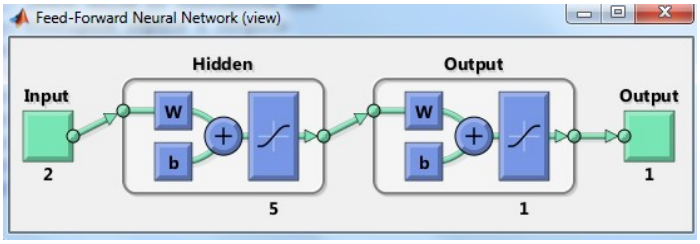


Рисунок 3.29 – Данная сеть тоже справится с задачей XOR

Однако, как видно из рисунка 3.30, ошибка обучения опустится в район 10^{-9}, что больше, чем у первой сети, что в целом хуже.

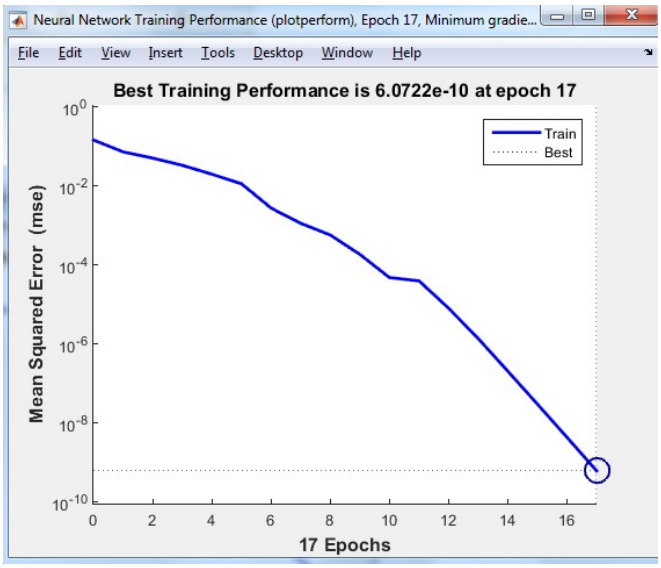


Рисунок 3.30 – Ошибка обучения для сети 2-5-1

Граница решений для этой сети показана на рисунке 3.31.

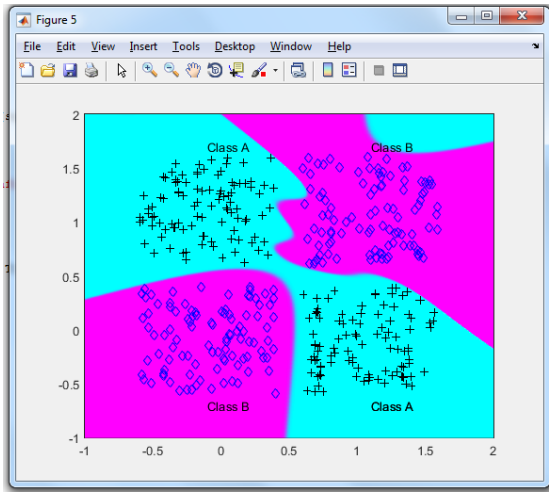


Рисунок 3.31 – Граница решений для сети с пятью скрытыми нейронами

Сеть со структурой 2-3-1 даст ошибку, показанную на рисунке 3.32.

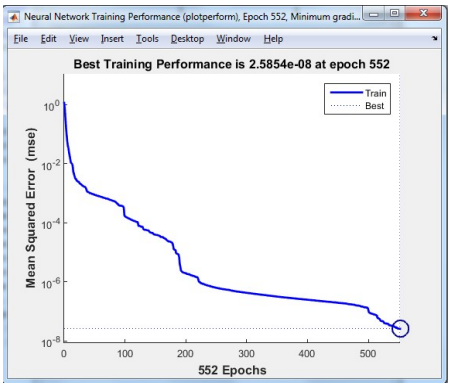


Рисунок 3.32 – Ошибка обучения для сети 2-3-1

Получившаяся граница решений для сети с тремя скрытыми нейронами показана на рисунке 3.33.

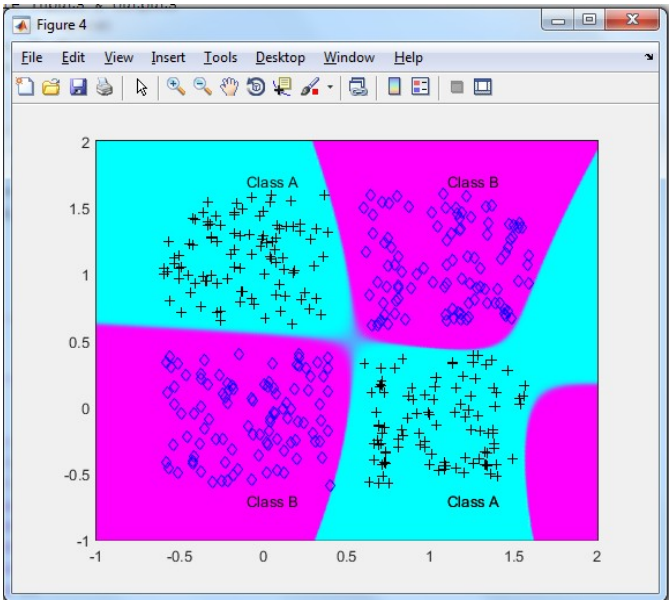


Рисунок 3.33 – Граница решений для сети из трёх нейронов в скрытом слое

Можно сделать вывод, что чем меньше количество нейронов в скрытом слое, тем грубее будут разделены классы и граница решений может в некоторых местах принимать хаотический вид (рисунок 3.31).

Конечно, такой разброс в решениях получается вследствие лёгкой разделимости классов.

Теперь применим сеть прямого распространения к разделению 4-х классов (ранее решали эту задачу с помощью персептрона).

```
close all, clear all, clc, format compact
K = 100;
q = .6;
A = [rand(1, K)-q; rand(1, K)+q];
B = [rand(1, K)+q; rand(1, K)+q];
C = [rand(1, K)+q; rand(1, K)-q];
D = [rand(1, K)-q; rand(1, K)-q];
figure(1);
plot(A(1, :), A(2, :), 'k+');
hold on;
grid on;
plot(B(1, :), B(2, :), 'b*');
plot(C(1, :), C(2, :), 'kx');
plot(D(1, :), D(2, :), 'bd');
text(.5-q, .5+2*q, 'Class A');
text(.5+q, .5+2*q, 'Class B');
text(.5+q, .5-2*q, 'Class C');
text(.5-q, .5-2*q, 'Class D');
% Кодируем классы
a = [-1 -1 -1 +1]';
b = [-1 -1 +1 -1]';
c = [-1 +1 -1 -1]';
d = [+1 -1 -1 -1]';
P = [A B C D];
T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(C)) repmat(d, 1, length(D))];
net =feedforwardnet([4 3]);
net.divideParam.trainRatio = 1;
net.divideParam.valRatio = 0;
net.divideParam.testRatio = 0;

[net, tr, Y, E] = train(net, P, T);
view(net);
% Получаем в “m” все максимальные элементы по столбцам, а в “i”
% их строковые индексы
[m, i] = max(T);
% Получаем в j индексы строк максимальных элементов
[m, j] = max(Y);
N =  length(Y);
k = 0;
```

```
% Сравниваем эти индексы, они должны совпасть, если они не
% совпадают, то увеличиваем k на 1, т.е. k будет содержать количество
% ответов ИНС, которые не совпадают с метками
if find(i - j),
        k = length(find(i-j));
end
% Высчитываем это количество в процентах
fprintf('Correct classified samples: %.if%% samples\n', 100*(N-k)/N);
figure;
% Строим верхний график на канве
subplot(211);
plot(T');
title('Targets');
ylim([-2 2]);
grid on;
% Строим нижний график на канве
subplot(212);
plot(Y');
title('Network response');
xlabel('# sample');
ylim([-2 2]);
grid on;
span = -1:.01:2;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1);
% Наносим на сетку все ответы первого нейрона для
% всей выборки, они должны занять ¼ от площади всех классов
m = mesh(P1, P2, reshape(aa(1, :), length(span), length(span))-5);
set(m, 'facecolor', [1 0.2 .7], 'linestyle', 'none');
hold on;
% Тоже самое делаем для других нейронов
m = mesh(P1, P2, reshape(aa(2, :), length(span), length(span))-5);
set(m, 'facecolor', [1 1.0 0.5], 'linestyle', 'none');
m = mesh(P1, P2, reshape(aa(3, :), length(span), length(span))-5);
set(m, 'facecolor', [.4 1.0 0.9], 'linestyle', 'none');
m = mesh(P1, P2, reshape(aa(4, :), length(span), length(span))-5);
set(m, 'facecolor', [.3 .4 0.5], 'linestyle', 'none');
view(2);
```

Создаём 4 класса, каждый представлен 100 паттернами, рисунок 3.34.

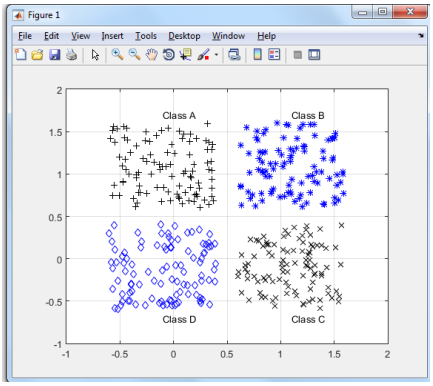


Рисунок 3.34 – 4 класса, которые нужно разделить

Далее создаём сеть со структурой 2-4-3-4 и обучаем её, рисунки 3.35, 3.36, 3.37.

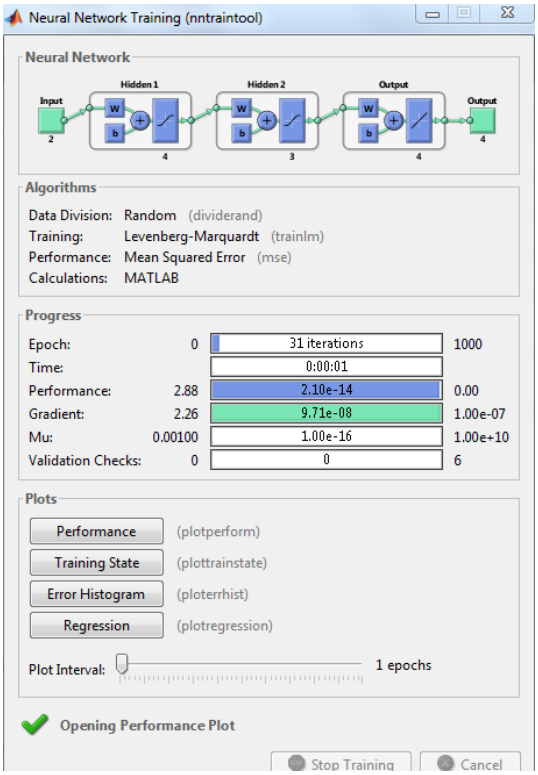


Рисунок 3.35 – Обучение сети, потребовалось 31 итерация

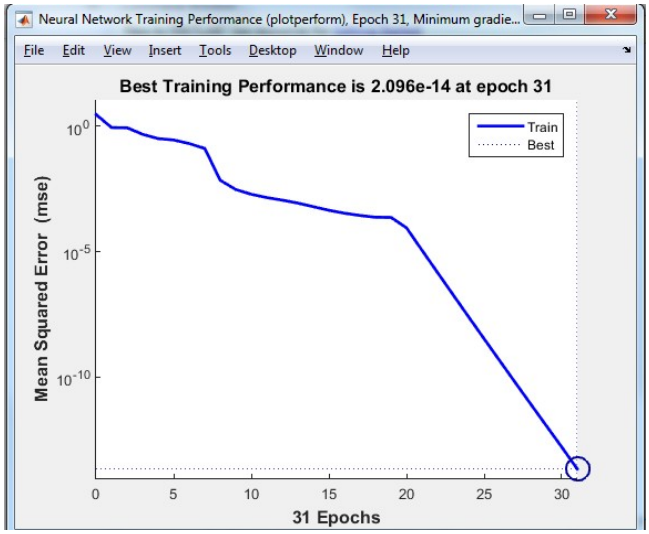


Рисунок 3.36 – Ошибка обучения очень хорошая, ушла ниже 10^{-10}

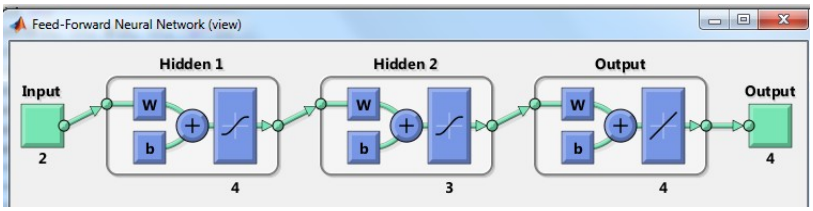


Рисунок 3.37 – Общая структура сети, которая использовалась

Разумеется, предложенная структура не единственная, на самом деле хватило бы сети с одним скрытым слоем и четырьмя нелинейными нейронами на выходном слое.

Метки и ответы сети изображены на рисунке 3.38, видно, что графики полностью совпадают, т.е. сеть точно моделирует все реакции на входные значения.

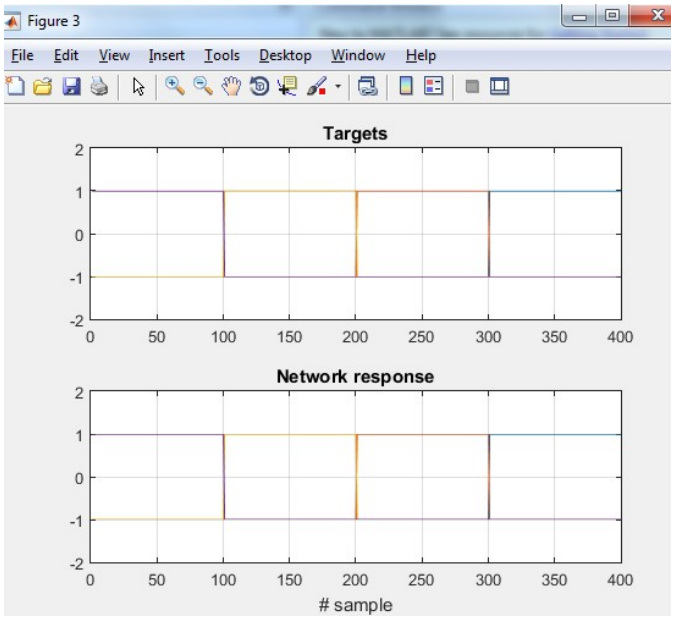


Рисунок 3.38 – График ожидаемых и реальных ответов ИНС

После построения границ решения можно видеть, что каждый класс отделён от другого, рисунок 3.39.

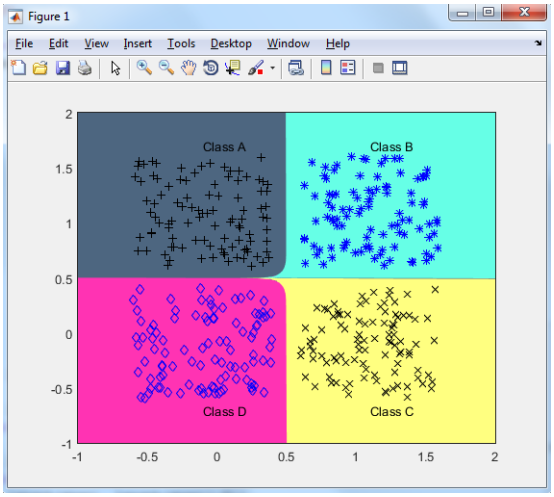


Рисунок 3.39 – Границы решения для задачи по разделению четырёх классов с помощью сети прямого распространения

Если сравнить это решение с тем, которое было достигнуто с помощью персептрона, то станет очевидным, что возможности по отделению одних классов от других у ИНС прямого распространения значительно выше, чем у персептрона, т.к. такие сети позволяют строить более сложные границы.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

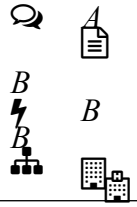
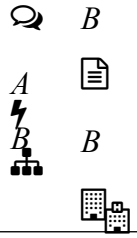
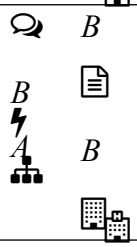
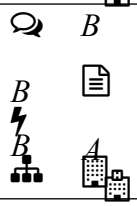
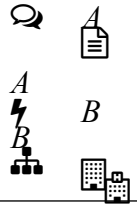
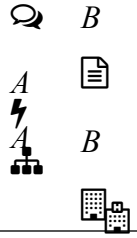
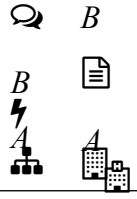
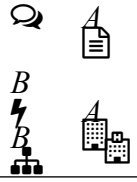
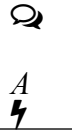
Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

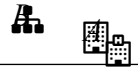
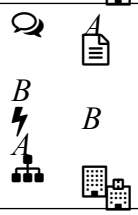
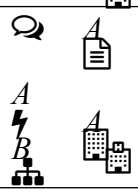
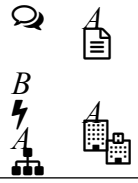
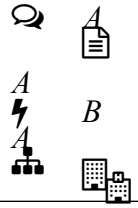
Методика и порядок выполнения работы

В индивидуальном варианте (таблица 3.1) дано расположение двух классов (A и B) по «углам» квадрата (по типу как на рисунке 3.39). Каждый «угол» представлен 200 точками. Требуется отделить класс A от B с помощью персептрона и полносвязанной

многослойной сети прямого распространения.

Таблица 3.1 – Варианты заданий

Номер варианта	Условие задачи
1	
2	
3	
4	
5	
6	
7	
8	
9	

	
10	
11	
12	
13	
14	

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1. Название лабораторной работы и её номер.
2. ФИО студента и группу.
3. Формулировка индивидуального задания.

4. Документ отчёта с Print Prtscr диалоговых окон по шагам для своего варианта по подобию того, что описано в теоретической части. Обязательно должна присутствовать раскрашенная форма с финальным разделением для сети и цветные линии, отделяющие один класс от другого для персептрона. Должен присутствовать ступенчатый график, показывающий качество работы сети.
5. Ответы на контрольные вопросы.

Вопросы для защиты работы

1) Как кодируются классы для персептронов или сетей прямого распространения?

2) Опишите последовательность действий для нанесения входных значений на канву, чтобы визуально было видно расположение классов относительно друг друга.

3) Опишите последовательность действий для построения на канве границ решения задачи классификации.

4) Приведите пример сети прямого распространения с одним скрытым слоем, с помощью которой можно решить задачу о разделении 4-х классов в рассмотренной лабораторной работе.

5) Почему с помощью сетей прямого распространения можно лучше справляться с задачами классификации, чем с помощью персептронов?

Лабораторная работа № 4

Классификаторы, основанные на оценки функции оптимизации.

Машины опорных векторов и нейронные сети прямого распространения

Цель и содержание работы: познакомить студента с машинами опорных векторов и нейронной сетью прямого распространения.

Задачи:

1) Разобрать принцип действия и алгоритм работы машины опорных векторов;
2) Разобрать принцип действия нейронной сети прямого распространения.

Теоретическое обоснование

Машина опорных векторов (**S**upport **V**ector **M**achine) основывается на понятии разницы. Рассмотрим линейный классификатор (4.1)

$$w^T x + w_0 \geq 0 \quad (4.1)$$

,

который, по сути представляет собой разделяющую гиперплоскость (4.2).

$$\begin{cases} w^T x + w_0 > 0, \text{ тогда } d_i = 1 \\ w^T x + w_0 < 0, \text{ тогда } d_i = -1 \end{cases} \quad (4.2)$$

Для данного вектора весов **w** и порога w_0 расстояние между гиперплоскостью, задаваемой уравнением (4.1), и ближайшей точкой из набора данных называется границей разделения и обозначается как ρ. Основной целью SVM является поиск конкретной гиперплоскости, для которой границе разделения будет максимальной. При этом условии поверхность решения называется оптимальной гиперплоскостью.

Общую идею можно понять из рисунка 4.1.

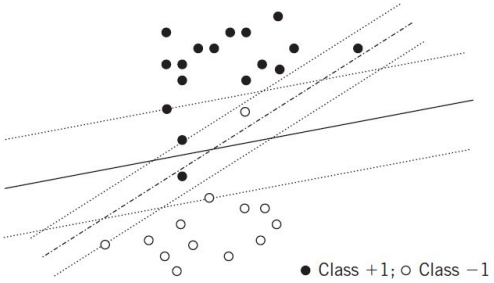


Рисунок 4.1 – Два способа отделения двух линейно разделимых классов

На рисунке 4.1 изображены два линейно отделимые класса: черные точки и белые точки. Представим, что черные и белые точки – это дома в деревне. Через деревню нужно

проложить дорогу так, чтобы разрушить минимальное количество домов и при этом, чтобы она была максимально широкая. На рисунке видны два возможных варианта прокладки дорог: дорога с разделительной пунктирной линией, идущая из левого нижнего угла в верхний правый, и дорога со сплошной разделительной линией. Обе дороги ломают по 5 домов, но очевидно, что более широкая дорога выгоднее, т.к. более удобная, чем узкая, поэтому нужно строить именно широкую дорогу. Средняя сплошная разделительная линия и будет оптимальной разделяющей гиперплоскостью.

Почему это лучше для задач распознавания образов? Дело в том, что обычные нейронные сети просто удовлетворяются от нахождения любой разделяющей гиперплоскости, они не выискивают оптимальные гиперплоскости, а просто находят первую попавшуюся в процессе обучения. Но потом, когда сеть начинает функционировать не на обучающих данных, а на тестовых, которые не участвовали в процессе обучения, то оказывается, что найденная гиперплоскость может давать много ошибок (узкую дорогу легко «перейти» и в результате паттерн из одного класса может оказаться на стороне, где распложены паттерны другого класса). Широкая дорога, т.е. оптимальная гиперплоскость, увеличивает вероятность, что сеть справится с задачей лучше.

На рисунке 4.2 представлена оптимальная гиперплоскость для линейно разделимых образов.

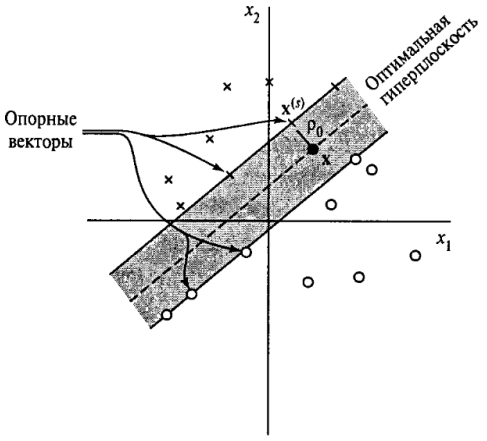


Рисунок 4.2 – Оптимальная гиперплоскость для линейно разделимых образов

Данное решение ведёт к следующей математической формулировки. Возьмем множество обучающих точек x_i с соответствующими им метками $y_i = \{\pm 1\}$, i = 1..N для задачи классификации двух классов. Вычисление оптимальной разделяющей гиперплоскости обозначим как минимизацию J():

$$J(w, w_0, \xi) = \frac{1}{2} w^2 + C \sum_{i=1}^{N} \xi_i, \qquad (4.3)$$

$$\begin{cases} w^T x_i + w_0 \geq 1 - \xi_i, & \text{if } x_i \in w_1 \\ w^T x_i + w_0 \leq -1 + \xi_i, & \text{if } x_i \in w_2 \end{cases} \qquad (4.4)$$

$$\xi_i \geq 0$$

где разница между областями равна 2 / ||w||, величины ошибок ξ_i не отрицательные, они равны 0 для точек, лежащих за пределами разделяющей области и на правильной стороне, и положители для точек, лежащих внутри области или вне области и на неправильной стороне. C – это константа, определяемая пользователем.

Решение будет представлено в следующем виде:

$$w = \sum_{i=1}^{N} \lambda_i y_i x_i, \qquad (4.5)$$

где коэффициент λ_i – это множители Лагранжа для оптимизационной задачи, и они равны 0 для всех точек, лежащих за пределами разделяющей области и на правильной стороне классификатора. Эти точки не предоставляют информацию для постройки оптимальной разделяющей гиперплоскости, остальные точки называются опорными точками или опорными векторами.

Для создания линейной SVM будет использован функция SMO2. Сигнатура данной функции имеет следующий вид: **[alpha, w0, w, evals, stp, glob] = SMO2(X′, y′, kernel, kpar2, C, tol, steps, eps, method)**.

Список формальных параметров: матрица X' включает точки данных, каждая строка – это координаты точки в многомерном пространстве, метки содержатся в y', тип ядерной функции – линейный. Два параметра ядра kra1 и kra2 в линейном случае устанавливаются всегда как 0. Пользовательский параметр C, параметр tol, максимальное число итераций алгоритма, порог eps (очень маленькое число обычно в районе 10^{-10}) используется для сравнения двух чисел (если их разница меньше, чем порог, то они считаются равными). И последний параметр – тип оптимизационного метода, который мы используем (0 – метод Платта, 1 – модификация Кири 1, 2 – модификация Кири 2).

Список возвращаемых значений: alpha – вектор, содержащий множители Лагранжа, w_0 – порог, w – вектор, содержащий параметры гиперплоскости.

Пример 1.

В двумерной плоскости есть два равновероятных класса, каждый представлен распределением Гаусса со средним $m_1 = [0, 0]T$ и $m_2 = [1.2, 1.2]T$ и ковариационными матрицами $S_1 = S_2 = 0.2I$, где I – это единичная матрица 2x2.

1. Сгенерировать и нарисовать обучающее множество X_1, содержащее 200 точек для каждого класса (всего 400 точек). Сгенерировать другое множество X_2, которое будет тестовым (также содержит по 200 точек для каждого класса).

2. Используя X_1 и метод Платта сгенерировать 6 SVM, которые бы разделяли два класса, используя пользовательскую константу со значением C = 0.1, 0.2, 0.5, 1, 2, 20; константа tol = 0.001. Вычислить ошибку обучения и обобщения, подсчитать количество опрных векторов, вычислить разницу между двумя областями (2 / ||w||), нарисовать решение.

Листинг задачи приведён ниже, полный листинг функции SMO2, svcplot_book и CalcKernel приведён в приложении 1.

```
close('all');
clear;

% Генерируем и рисуем X1
randn('seed',50)
m=[0 0; 1.2 1.2]'; % среднее векторов
S=0.2*eye(2); % матрица ковариации
points_per_class=[200 200];
X1=mvnrnd(m(:,1),S,points_per_class(1))';
X1=[X1 mvnrnd(m(:,2),S,points_per_class(2))'];
y1=[ones(1,points_per_class(1)) -ones(1,points_per_class(2))];

figure(1), plot(X1(1,y1==1),X1(2,y1==1),'r.', X1(1,y1==-1),X1(2,y1==-1),'bo')

% Генерируем X2
randn('seed',100)
X2=mvnrnd(m(:,1),S,points_per_class(1))';
X2=[X2 mvnrnd(m(:,2),S,points_per_class(2))'];
y2=[ones(1,points_per_class(1)) -ones(1,points_per_class(2))];

% Генерируем
SVM
kernel='linear';
kpar1=0;
kpar2=0;
C=0.1;
% Тут можно выбирать нужную константу
% C=0.2;
% C= 0.5;
% C=1;
% C=2;
% C=20;
tol=0.001;
steps=100000;
eps=10^(-10);
method=0;
[alpha, w0, w, evals, stp, glob] = SMO2(X1', y1',kernel, kpar1, kpar2, C, tol, steps, eps, method);
```

```
% Вычисляем ошибку обучения
Pe_tr=sum((2*(w*X1-w0>0)-1).*y1<0)/length(y1)

% Вычисляем ошибку обобщения
Pe_te=sum((2*(w*X2-w0>0)-1).*y2<0)/length(y2)

% Рисуем оптимальную разделяющую гиперплоскость
global figt4
figt4=2;
svcplot_book(X1',y1',kernel,kpar1,kpar2,alpha,-w0)

% Вычисляем количество опорных векторов
sup_vec=sum(alpha>0)

% Вычисляем разницу для разделительной области
marg=2/sqrt(sum(w.^2))
```

На рисунке 4.3 представлены исходные данные для обучения.

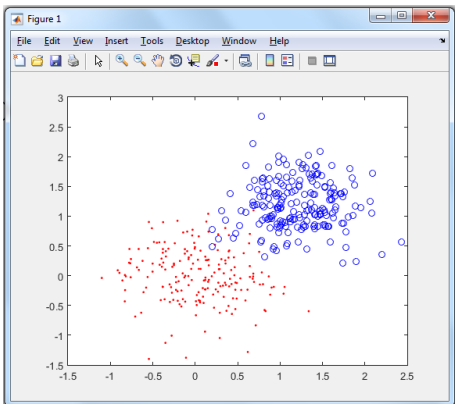


Рисунок 4.3 – Исходные данные для обучения

На рисунке 4.4. представлена нарисованная оптимальная гиперплоскость, которая отделяет один класс от другого

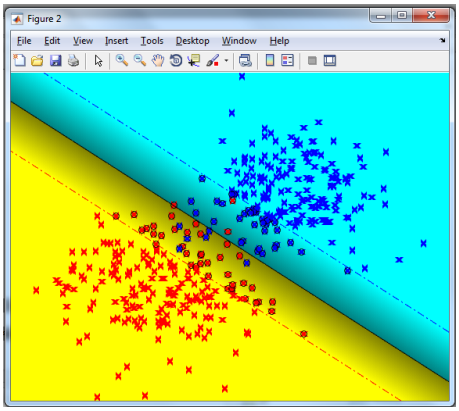


Рисунок 4.4 – Решение на обучающей выборке

В таблице 4.1 представлены результаты работы алгоритма для различных

пользовательских констант.

Таблица 4.1 – Сравнительные результаты для различных пользовательских констант C

	C=0.1	C=0.2	C=0.5	C=1	C=2	C=20
Количество поддерживающих векторов	82	61	44	37	31	25
Ошибка обучения	2.25%	2.00%	2.00%	2.25%	3.25%	2.50%
Ошибка на тестовом множестве	3.25%	3.00%	3.25%	3.25%	3.50%	3.50%
Величина зазора (margin)	0.9410	0.8219	0.7085	0.6319	0.6047	0.3573

Из таблицы видно, что ширина разделяющей области увеличивается, когда уменьшается константа C. Это естественно, потому что уменьшение C делает последнее слагаемое в (4.3) более важным. Лучшее решение, т.е. минимальная ошибка обобщения получена для C = 0.2.

Далее рассмотрим многослойную сеть прямого распространения. Общий вид такой сети представлен на рисунке 4.5.

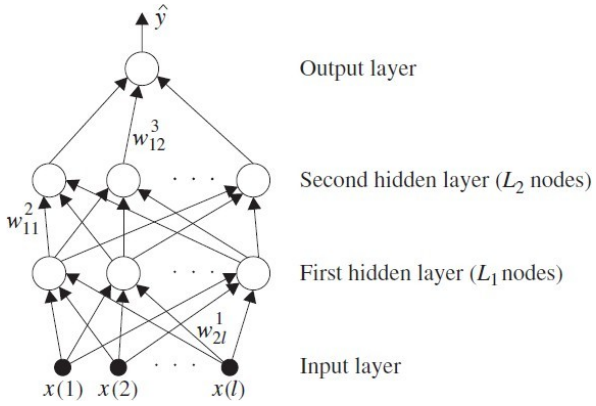


Рисунок 4.5 – Общий вид полносвязанной сети прямого распространения

Общая ошибка обучения вычисляется как

$$J = \sum_{i=1}^{N} (\tilde{y} - \hat{y}_i)^2 \qquad (4.6)$$

,

где y_i – предполагаемый ответ сети, а $\hat{y}_i$ – фактический ответ сети. Поиск минимума по поверхности ошибок функции J и есть обучение нейронной сети.

Веса сети обновляются по формуле (4.7):

$$w(\text{new}) \leftarrow w(\text{old}) + \Delta w \qquad (4.7)$$

Δw находим исходя из (4.8):

$$\Delta w = -\eta \frac{\partial J}{\partial w}, \qquad (4.8)$$

где η - скорость обучения. Формулу (4.8) можно немного модифицировать, чтобы получилось обучение с инерцией:

$$\Delta w(\text{new}) \leftarrow \alpha \Delta w(\text{old}) - \eta \frac{\partial J}{\partial w}, \qquad (4.9)$$

где α - коэффициент инерции, также как и η вводится пользователем и принимает значения от 0 до 1.

Для обучения нейронной сети будем использовать функцию NN_training со следующей сигнатурой: **[net, tr] = NN_training(X, y, k, code, iter, par_vec)**.

Список формальных параметров: X содержит обучающие вектора по колонкам, y – содержит метки для соответствующих классов, code – параметр для определения типа алгоритма обучения (1 – стандартный алгоритм обратного распространения, 2 – обратное распространение с моментом (с инерцией), 3 – обратное распространение с адаптивной скоростью обучения), iter – максимально допустимое количество итераций, par_vec – 5 размерный вектор, содержащий в себе скорость обучения, момент, три значения для третьего алгоритма обучения с адаптивной скоростью обучения.

Список возвращаемых значений: net – структура сети, tr – структура, используемая MATLAB, куда помещаются различные параметры сети.

Пример 2.

Рассмотрим задачу классификации двух классов на плоскости. Точки первого (второго) класса имеют метки +1 (-1) соответственно. Точки происходят от трех (четырех) Гауссовых распределений с одинаковыми вероятностями: $[-5, 5]^T$, $[5, -5]^T$, $[10, 0]^T$ ($[-5, -5]^T$, $[0, 0]^T$, $[5, 5]^T$, $[15, -5]^T$). Ковариационная матрица для каждого распредления равна $\sigma^2 I$, где $\sigma^2 = 1$, а I – это единичная матрица размером 2x2.

1. Сгенерировать и нарисовать множество X_1 (обучающее множество), содержащее 60 точек для класса +1 (приблизительно по 20 точек для каждого распределения) и 80 точек для класса -1 (также по 20 точек для каждого распределения). Аналогичные предписания и для формирования X_2 (тестового множества).

2. На X_1 обучить две двухслойный сети прямого распространения с двумя и четырьмя нейронами в скрытом слое. Для всех нейронах на скрытых слоях использовать функцию активации гиперболический тангенс, на выходном слое – линейную функцию

активации. Запустить стандартный алгоритм обратного распространения ошибки для 9000 итераций со скоростью обучения 0.01. Вычислить ошибку обучения и обобщения, и нарисовать регионы решений.

3. Повторить шаг 2, используя скорость обучения 0.0001 для стандартного алгоритма обратного распространения ошибки.

4. Повторить шаг 2, применив адаптивный алгоритм (тип 3) с 6000 итераций, со скоростью обучения 0.0001 и $r_i = 1.05$, $r_d = 0.7$, $c = 1.04$.

Листинг программы рассмотрен ниже.

```
close('all');
clear;

randn('seed',0);
% 1. Генерируем X1
l=2; % Размерность
m1=[-5 5; 5 -5; 10 0]'; % Центроиды
m2=[-5 -5; 0 0; 5 5; 15 -5]';
[l,c1]=size(m1); % Номер гауссиана для каждого класса
[l,c2]=size(m2);

P1=ones(1,c1)/c1; % Веса для смешенной модели
P2=ones(1,c2)/c2;
s=1; % разница

% Генерируем данные для первого класса
N1=60; % Количество точек для первого класса
for i=1:c1
   S1(:,:,i)=s*eye(l);
end
sed=0; % Генератор случайных чисел, параметр для него
[class1_X,class1_y]=mixt_model(m1,S1,P1,N1,sed);

% Генерируем точки для второго класса
N2=80; % Количество точек для второго класса
for i=1:c2
   S2(:,:,i)=s*eye(l);
end
sed=0;
[class2_X,class2_y]=mixt_model(m2,S2,P2,N2,sed);

% Форма X1
X1=[class1_X class2_X]; % Вектор данных
y1=[ones(1,N1) -ones(1,N2)]; % Вектор меток
figure(1), hold on
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'r.',X1(1,y1==-1),X1(2,y1==-1),'bx')

% Генерируем тестовое множество X2

% Данные первого класса
```

```
sed=100; % Используем генератор случайных чисел
[class1_X,class1_y]=mixt_model(m1,S1,P1,N1,sed);

% Данные для второго класса
sed=100; % Используем генератор случайных чисел
[class2_X,class2_y]=mixt_model(m2,S2,P2,N2,sed);

% Объединяем данные в единый вектор и с метками также
X2=[class1_X class2_X]; % Для данных
y2=[ones(1,N1) -ones(1,N2)]; % Для меток


% 2. Шаг 2

rand('seed',100)
randn('seed',100)
iter=9000; % Количество итераций
code=1; % Выбираем тип алгоритма для обучения нейронной сети
k=4; % число нейронов в скрытом слое
lr=.01; % скорость обучения
par_vec=[lr 0 0 0 0];
[net,tr]=NN_training(X1,y1,k,code,iter,par_vec);

% Вычисляем ошибку обучения и обобщения
pe_train=NN_evaluation(net,X1,y1)
pe_test=NN_evaluation(net,X2,y2)

% Рисуем данные
maxi=max(max([X1'; X2']));
mini=min(min([X1'; X2']));
bou=[mini maxi];
fig_num=2;
resolu=(bou(2)-bou(1))/100; % разрешения рисунка
plot_NN_reg(net,bou,resolu,fig_num); % рисуем регион решения
figure(fig_num), hold on % рисуем обучающее множество
figure(fig_num), plot(X1(1,y1==1),X1(2,y1==1),'r.', X1(1,y1==-1),X1(2,y1==-1),'bx')

figure(3), plot(tr.perf)
pause

% Шаг 4
iter=6000; % Количество итераций
code=3; % Выбираем алгоритм обучения
k=4; % количество нейронов в скрытом слое
lr=.0001; % скорость обучения
par_vec=[lr 0 1.05 0.7 1.04]; % вектор параметров для алгоритма code = 3
[net,tr]=NN_training(X1,y1,k,code,iter,par_vec);

% Вычисляем ошибку обучения и обобщения
pe_train=NN_evaluation(net,X1,y1)
pe_test=NN_evaluation(net,X2,y2)
```

Первоначальные данные представлены на рисунке 4.6.

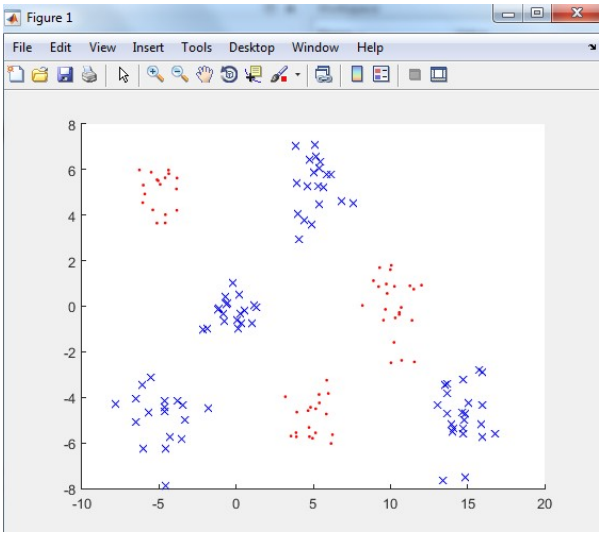


Рисунок 4.6 – Данные, которые необходимо разделить

Получившееся решение представлено на рисунке 4.7, видно, что сеть проведя разделяющие границы успешно отделяет один класс от другого.

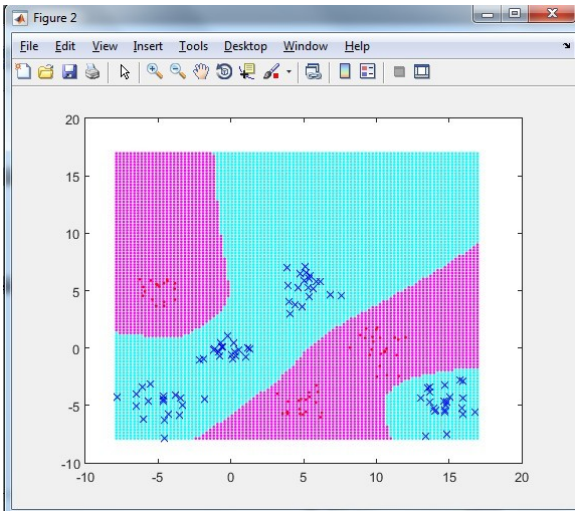


Рисунок 4.7 – Получившееся решение

Зависимость ошибки обучения от эпохи приведена на рисунке 4.8.

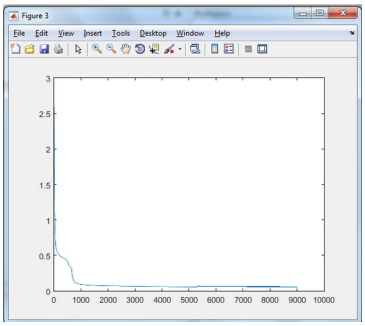


Рисунок 4.8 – Зависимость ошибки обучения от эпохи

После 2000-ой эпохи обучение практически останавливается.

Общие параметры обучения приведены на рисунке 4.9.

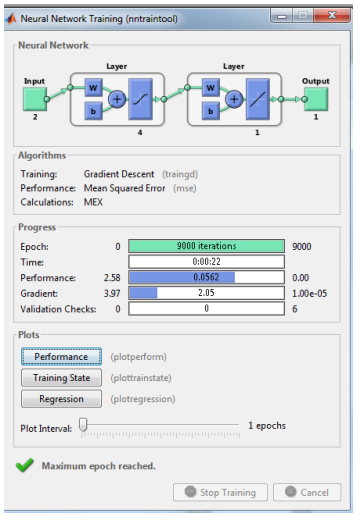


Рисунок 4.9 – Окно среды Matlab, отражающее процесс обучения нейронной сети

В таблице 4.2 приведены результаты обучения сети для стандартного алгоритма обратного распространения ошибки со скоростью обучения 0.01 и 9000 итераций. В таблице 4.2 приведены результаты для адаптивного алгоритма обратного распространения ошибки, скорости обучения 0.0001 и 6000 итераций.

Таблица 4.2 – Применение стандартного алгоритма обратного распространения ошибки

	Два узла	Четыре узла
Ошибка обучения	29.29%	0
Ошибка обобщения	30.71%	0

Таблица 4.3 – Применение адаптивного алгоритма обратного распространения ошибки

	Два узла	Четыре узла
Ошибка обучения	29.29%	0
Ошибка обобщения	32.14%	0

Функции, использующиеся в листинге, приведены ниже.

Первая функция.

```
function [net,tr]=NN_training(X,y,k,code,iter,par_vec)

rand('seed',0);  % Датчик случайных чисел
randn('seed',0);

% Список методов обучения для ИНС, они встроены в Matlab
methods_list={'traingd'; 'traingdm'; 'traingda';'trainlm'; 'traingdx'};
```

```
% Ограничение для области, где лежат данные
limit=[min(X(:,1)) max(X(:,1)); min(X(:,2)) max(X(:,2))];
% Определение нейронной сети
net=newff(limit,[k 1],{'tansig','purelin'}, methods_list{code,1}); %  'traingda')

% Инициализация нейронной сети
net=init(net);

% Установка параметров
net.trainParam.epochs=iter;
net.trainParam.lr=par_vec(1);
if(code==2)
   net.trainParam.mc=par_vec(2);
elseif(code==3)
   net.trainParam.lr_inc=par_vec(3);
   net.trainParam.lr_dec=par_vec(4);
   net.trainParam.max_perf_inc=par_vec(5);
end

% Обучение нейронной сети
[net,tr]=train(net,X,y);
```

Вторая функция.

```
function [X,y]=mixt_model(m,S,P,N,sed)

rand('seed',sed);
[l,c]=size(m);

P_acc=P(1);
for i=2:c
   t=P_acc(i-1)+P(i);
   P_acc=[P_acc t];
end

X=[];
y=[];
for i=1:N
   t=rand;
   ind=sum(t>P_acc)+1;
   X=[X; mvnrnd(m(:,ind)',S(:,:,ind),1)];
   y=[y ind];
end
X=X';
```

Третья функция.

```
function pe=NN_evaluation(net,X,y)

y1=sim(net,X);

pe=sum(y.*y1<0)/length(y);
```

Четвертая функция.

```
function plot_NN_reg(net,bou,resolu,fig_num)

t=round(((bou(2)-bou(1))/resolu)+1);
t_vec=bou(1):resolu:bou(2);
t_vec=t_vec';
X1=[];
for i=bou(1):resolu:bou(2)
   X1=[X1; i*ones(t,1) t_vec];
end
X1=X1';
out_vec=2*(sim(net,X1)>0)-1;
figure(fig_num),
plot(X1(1,out_vec>0),X1(2,out_vec>0),'m.',X1(1,out_vec<0),X1(2,out_vec<0),'c.')
```

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Решить собственный вариант по образцу примера 1 и 2. Индивидуальные варианты представлены в таблице 4.4.

Таблица 4.4 – Индивидуальные варианты

1	Первая задача. Рассмотреть задачу из примера 1 с параметрами: $m_1 = [0, 0]T$ и $m_2 = [1, 1]T$ и ковариационными матрицами $S_1 = S_2 = 0.5I$, количество паттернов 400, а не 200. Вторая задача. Рассмотреть задачу из примера 2 с параметрами: точки происходят

	от трех (четырех) Гауссовых распределений с одинаковыми вероятностями: [-3, 3]^T, [2, -3]^T, [7, 1]^T ([-7, -7]^T, [0, 0]^T, [10, 10]^T, [10, -7]^T).
2	Первая задача. Рассмотреть задачу из примера 1 с параметрами: m_1 = [0, 0]T и m_2 = [1.7, 1.5]T и ковариационными матрицами $S_1 = S_2 = 0.1I$, количество паттернов 200. Вторая задача. Рассмотреть задачу из примера 2 с параметрами: точки происходят от трех (четырех) Гауссовых распределений с одинаковыми вероятностями: [-1.5, 2.5]^T, [3.5, -4.5]^T, [10, -3]^T ([-6, -4]^T, [0, 0]^T, [6, 6]^T, [17, -8]^T).
3	Первая задача. Рассмотреть задачу из примера 1 с параметрами: m_1 = [-1, -1]T и m_2 = [1.5, 1.5]T и ковариационными матрицами $S_1 = S_2 = 0.7I$, количество паттернов 200. Вторая задача. Рассмотреть задачу из примера 2 с параметрами: точки происходят от трех (четырех) Гауссовых распределений с одинаковыми вероятностями: [-5, 5]^T, [7, -5]^T, [10, 1]^T ([-5, -6]^T, [1, 1.5]^T, [5.6, 5.9]^T, [15, -10]^T).
4	Первая задача. Рассмотреть задачу из примера 1 с параметрами: m_1 = [3, 3]T и m_2 = [1.5, 1.5]T и ковариационными матрицами $S_1 = S_2 = 0.5I$, количество паттернов 400. Вторая задача. Рассмотреть задачу из примера 2 с параметрами: точки происходят от трех (четырех) Гауссовых распределений с одинаковыми вероятностями: [-10, 5]^T, [5, -10]^T, [20, 0]^T ([-10, -10]^T, [0, 0]^T, [7, 5]^T, [15, -6]^T).
5	Первая задача. Рассмотреть задачу из примера 1 с параметрами: m_1 = [0, 0]T и m_2 = [1.8, 1.8]T и ковариационными матрицами $S_1 = S_2 = 0.1I$, количество паттернов 400. Вторая задача. Рассмотреть задачу из примера 2 с параметрами: точки происходят от трех (четырех) Гауссовых распределений с одинаковыми вероятностями: [-2, 2]^T, [2, -2]^T, [7, 0]^T ([-5, -5]^T, [1, 1.5]^T, [6, 6.5]^T, [19, -5.5]^T).
6	Первая задача. Рассмотреть задачу из примера 1 с параметрами: m_1 = [0, 0]T и m_2 = [1.5, 1.4]T и ковариационными матрицами $S_1 = S_2 = 0.3I$, количество паттернов 200. Вторая задача. Рассмотреть задачу из примера 2 с параметрами: точки происходят от трех (четырех) Гауссовых распределений с одинаковыми вероятностями: [-5.5, 5]^T, [6.7, -4.5]^T, [10, 0]^T ([-4, -4]^T, [1.1, 1.5]^T, [5, 5]^T, [20, -5]^T).
7	Первая задача. Рассмотреть задачу из примера 1 с параметрами: m_1 = [1.1, 0]T и m_2 = [1.5, 1.9]T и ковариационными матрицами $S_1 = S_2 = 0.2I$, количество паттернов 200. Вторая задача. Рассмотреть задачу из примера 2 с параметрами: точки происходят от трех (четырех) Гауссовых распределений с одинаковыми вероятностями: [-5, 5]^T, [5, -7]^T, [10, 0]^T ([-5, -5]^T, [1, 1]^T, [8, 8]^T, [15, -9]^T).
8	Первая задача. Рассмотреть задачу из примера 1 с параметрами: m_1 = [0, 0]T и m_2 = [1.6, 1.6]T и ковариационными матрицами $S_1 = S_2 = 0.4I$, количество паттернов 400. Вторая задача. Рассмотреть задачу из примера 2 с параметрами: точки происходят от трех (четырех) Гауссовых распределений с одинаковыми вероятностями: [-5, 6]^T, [6, -5]^T, [12, 2]^T ([-6, -6]^T, [1, 1.4]^T, [5, 5]^T, [19, -5]^T).

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1) Название лабораторной работы и её номер.
2) ФИО и группу студента.
3) Формулировка индивидуального задания и номер варианта.

4) Ответ на задание в виде кода и результатов работы Matlab.

Вопросы для защиты работы

1. Чем SVM отличается от сети прямого распространения?
2. Чем SVM отличается от персептрона?
3. Что такое оптимальная разделяющая гиперплоскость?

Лабораторная работа № 5

Пример создания и обучения нейронных сетей для задач классификации в среде Matlab. Часть 1

Цель и содержание работы: создать и обучить несколько нейронных сетей, решающих задачи классификации на примере задачи ирисов Фишера и двух спиралей.

Задачи:

- разобрать решение задачи об ирисах Фишера;
- научиться обучать сети с разной архитектурой и отбирать из них наилучшие на примере задачи об ирисах;
- разобрать решение задачи двух спиралей;
- научиться подбирать ряд параметров для архитектуры нейронных сетей на основе задаче о двух спиралях;
- научиться создавать сети с помощью разных функций Matlab и обучать их с помощью разных алгоритмов на основе задачи о двух спиралях;
- разобрать код обратного распространения ошибки для задачи о двух спиралях.

Теоретическое обоснование

В предыдущей лабораторной работе рассматривались очень лёгкие задачи классификации, причём тестовое и валидационное множество отсутствовали. В этой лабораторной работе будут рассмотрены два реальных примера задачи классификации.

Первая задача – это задача ирисов Фишера. Ирисы Фишера состоят из данных о 150 экземплярах ириса, по 50 экземпляров из трёх видов – Ирис щетинистый (Iris setosa), Ирис виргинский (Iris virginica), Ирис разноцветный (Iris versicolor), рисунок 5.1.

Рисунок 5.1 – Слева направо: Iris versicolor, Iris virginica, Iris setosa

Каждый экземпляр того или иного ириса характеризовался четырьмя числами, которые выражали определённые характеристики цветков: длина наружной доли околоцветника (sepal length), ширина наружной доли околоцветника (sepal width), длина

внутренней доли околоцветника (petal length), ширина внутренней доли околоцветника (petal width); всё в сантиметрах. На основании этого набора данных требуется построить правило классификации, определяющее вид растения по данным измерений. Это задача многоклассовой классификации, т.к. имеются три вида ириса.

Все данные для данной задачи приведены в таблице 5.1. Именно в подобном виде записывается большинство задач классификации. Стоит обратить внимание, что данные приведены в вещественном формате, где разделителем служит запятая, а не точка. Это сделано для того, чтобы Excel правильно интерпретировал вещественные числа, т.к. в этой среде разделитель – запятая, а точка обычно используется для записи даты.

Таблица 5.1 – Данные для задачи ирисов Фишера

№	Sepal	Sepal	Petal	Petal	Species	№	Sepal	Sepal	Petal	Petal	Species
1	5,1	3,5	1,4	0,2	*I, setosa*	76	6,6	3,0	4,4	1,4	*I. versicol*
2	4,9	3,0	1,4	0,2	*I, setosa*	77	6,8	2,8	4,8	1,4	*I. versicol*
3	4,7	3,2	1,3	0,2	*I, setosa*	78	6,7	3,0	5,0	1,7	*I. versicol*
4	4,6	3,1	1,5	0,2	*I, setosa*	79	6,0	2,9	4,5	1,5	*I. versicol*
5	5,0	3,6	1,4	0,2	*I, setosa*	80	5,7	2,6	3,5	1,0	*I. versicol*
6	5,4	3,9	1,7	0,4	*I, setosa*	81	5,5	2,4	3,8	1,1	*I. versicol*
7	4,6	3,4	1,4	0,3	*I, setosa*	82	5,5	2,4	3,7	1,0	*I. versicol*
8	5,0	3,4	1,5	0,2	*I, setosa*	83	5,8	2,7	3,9	1,2	*I. versicol*
9	4,4	2,9	1,4	0,2	*I, setosa*	84	6,0	2,7	5,1	1,6	*I. versicol*
10	4,9	3,1	1,5	0,1	*I, setosa*	85	5,4	3,0	4,5	1,5	*I. versicol*
11	5,4	3,7	1,5	0,2	*I, setosa*	86	6,0	3,4	4,5	1,6	*I. versicol*
12	4,8	3,4	1,6	0,2	*I, setosa*	87	6,7	3,1	4,7	1,5	*I. versicol*
13	4,8	3,0	1,4	0,1	*I, setosa*	88	6,3	2,3	4,4	1,3	*I. versicol*
14	4,3	3,0	1,1	0,1	*I, setosa*	89	5,6	3,0	4,1	1,3	*I. versicol*
15	5,8	4,0	1,2	0,2	*I, setosa*	90	5,5	2,5	4,0	1,3	*I. versicol*
16	5,7	4,4	1,5	0,4	*I, setosa*	91	5,5	2,6	4,4	1,2	*I. versicol*
17	5,4	3,9	1,3	0,4	*I, setosa*	92	6,1	3,0	4,6	1,4	*I. versicol*
18	5,1	3,5	1,4	0,3	*I, setosa*	93	5,8	2,6	4,0	1,2	*I. versicol*
19	5,7	3,8	1,7	0,3	*I, setosa*	94	5,0	2,3	3,3	1,0	*I. versicol*
20	5,1	3,8	1,5	0,3	*I, setosa*	95	5,6	2,7	4,2	1,3	*I. versicol*
2	5,	3,4	1,7	0,2	*I,*	9	5,7	3,0	4,2	1,2	*I.*

1	4				*setosa*	6					*versicol*
22	5,1	3,7	1,5	0,4	*I, setosa*	97	5,7	2,9	4,2	1,3	*I. versicol*
23	4,6	3,6	1,0	0,2	*I, setosa*	98	6,2	2,9	4,3	1,3	*I. versicol*
24	5,1	3,3	1,7	0,5	*I, setosa*	99	5,1	2,5	3,0	1,1	*I. versicol*
25	4,8	3,4	1,9	0,2	*I, setosa*	100	5,7	2,8	4,1	1,3	*I. versicol*
26	5,0	3,0	1,6	0,2	*I, setosa*	101	6,3	3,3	6,0	2,5	*I. virginic*
27	5,0	3,4	1,6	0,4	*I, setosa*	102	5,8	2,7	5,1	1,9	*I. virginic*
28	5,2	3,5	1,5	0,2	*I, setosa*	103	7,1	3,0	5,9	2,1	*I. virginic*
29	5,2	3,4	1,4	0,2	*I, setosa*	104	6,3	2,9	5,6	1,8	*I. virginic*
30	4,7	3,2	1,6	0,2	*I, setosa*	105	6,5	3,0	5,8	2,2	*I. virginic*
31	4,8	3,1	1,6	0,2	*I, setosa*	106	7,6	3,0	6,6	2,1	*I. virginic*
32	5,4	3,4	1,5	0,4	*I, setosa*	107	4,9	2,5	4,5	1,7	*I. virginic*
33	5,2	4,1	1,5	0,1	*I, setosa*	108	7,3	2,9	6,3	1,8	*I. virginic*
34	5,5	4,2	1,4	0,2	*I, setosa*	109	6,7	2,5	5,8	1,8	*I. virginic*
35	4,9	3,1	1,5	0,2	*I, setosa*	110	7,2	3,6	6,1	2,5	*I. virginic*

36	5,0	3,2	1,2	0,2	*I, setosa*	111	6,5	3,2	5,1	2,0	*I. virginic*
37	5,5	3,5	1,3	0,2	*I, setosa*	112	6,4	2,7	5,3	1,9	*I. virginic*
38	4,9	3,6	1,4	0,1	*I, setosa*	113	6,8	3,0	5,5	2,1	*I. virginic*
39	4,4	3,0	1,3	0,2	*I, setosa*	114	5,7	2,5	5,0	2,0	*I. virginic*
40	5,1	3,4	1,5	0,2	*I, setosa*	115	5,8	2,8	5,1	2,4	*I. virginic*
41	5,0	3,5	1,3	0,3	*I, setosa*	116	6,4	3,2	5,3	2,3	*I. virginic*
42	4,5	2,3	1,3	0,3	*I, setosa*	117	6,5	3,0	5,5	1,8	*I. virginic*
43	4,4	3,2	1,3	0,2	*I, setosa*	118	7,7	3,8	6,7	2,2	*I. virginic*
44	5,0	3,5	1,6	0,6	*I, setosa*	119	7,7	2,6	6,9	2,3	*I. virginic*
45	5,1	3,8	1,9	0,4	*I, setosa*	120	6,0	2,2	5,0	1,5	*I. virginic*
46	4,8	3,0	1,4	0,3	*I, setosa*	121	6,9	3,2	5,7	2,3	*I. virginic*
47	5,1	3,8	1,6	0,2	*I, setosa*	122	5,6	2,8	4,9	2,0	*I. virginic*
48	4,6	3,2	1,4	0,2	*I, setosa*	123	7,7	2,8	6,7	2,0	*I. virginic*
49	5,3	3,7	1,5	0,2	*I, setosa*	124	6,3	2,7	4,9	1,8	*I. virginic*
50	5,0	3,3	1,4	0,2	*I, setosa*	125	6,7	3,3	5,7	2,1	*I. virginic*
51	7,0	3,2	4,7	1,4	*I, versicol*	126	7,2	3,2	6,0	1,8	*I. virginic*
52	6,4	3,2	4,5	1,5	*I, versicol*	127	6,2	2,8	4,8	1,8	*I. virginic*
53	6,9	3,1	4,9	1,5	*I, versicol*	128	6,1	3,0	4,9	1,8	*I. virginic*
54	5,5	2,3	4,0	1,3	*I, versicol*	129	6,4	2,8	5,6	2,1	*I. virginic*
55	6,	2,8	4,6	1,5	*I, versicol*	130	7,2	3,0	5,8	1,6	*I. virginic*

	5										
56	5,7	2,8	4,5	1,3	*I, versicol*	131	7,4	2,8	6,1	1,9	*I. virginic*
57	6,3	3,3	4,7	1,6	*I, versicol*	132	7,9	3,8	6,4	2,0	*I. virginic*
58	4,9	2,4	3,3	1,0	*I, versicol*	133	6,4	2,8	5,6	2,2	*I. virginic*
59	6,6	2,9	4,6	1,3	*I, versicol*	134	6,3	2,8	5,1	1,5	*I. virginic*
60	5,2	2,7	3,9	1,4	*I, versicol*	135	6,1	2,6	5,6	1,4	*I. virginic*
61	5,0	2,0	3,5	1,0	*I, versicol*	136	7,7	3,0	6,1	2,3	*I. virginic*
62	5,9	3,0	4,2	1,5	*I, versicol*	137	6,3	3,4	5,6	2,4	*I. virginic*
63	6,0	2,2	4,0	1,0	*I, versicol*	138	6,4	3,1	5,5	1,8	*I. virginic*
64	6,1	2,9	4,7	1,4	*I, versicol*	139	6,0	3,0	4,8	1,8	*I. virginic*
65	5,6	2,9	3,6	1,3	*I, versicol*	140	6,9	3,1	5,4	2,1	*I. virginic*
66	6,7	3,1	4,4	1,4	*I, versicol*	141	6,7	3,1	5,6	2,4	*I. virginic*
67	5,6	3,0	4,5	1,5	*I, versicol*	142	6,9	3,1	5,1	2,3	*I. virginic*
68	5,8	2,7	4,1	1,0	*I, versicol*	143	5,8	2,7	5,1	1,9	*I. virginic*
69	6,2	2,2	4,5	1,5	*I, versicol*	144	6,8	3,2	5,9	2,3	*I. virginic*
70	5,6	2,5	3,9	1,1	*I, versicol*	145	6,7	3,3	5,7	2,5	*I. virginic*
71	5,9	3,2	4,8	1,8	*I, versicol*	146	6,7	3,0	5,2	2,3	*I. virginic*
72	6,1	2,8	4,0	1,3	*I, versicol*	147	6,3	2,5	5,0	1,9	*I. virginic*
73	6,3	2,5	4,9	1,5	*I, versicol*	148	6,5	3,0	5,2	2,0	*I. virginic*
74	6,1	2,8	4,7	1,2	*I, versicol*	149	6,2	3,4	5,4	2,3	*I. virginic*
75	6	2,9	4,3	1,3	*I,*	150	5,	3,0	5,1	1,8	*I.*

5	, 4				*versicol*	0	9				*virginic*

Данная выборка по ирисам является одной из классических задач классификации. Чтобы оценить сложность разделимости классов, можно отобразить точки в пространстве признаков. Так как каждый цветок характеризуется четырьмя числами, то искомая точка лежит в 4-х мерном пространстве, отобразить нельзя, но можно отобразить от двух и от трёх параметров. На рисунке 5.2 приведено расположение классов в зависимости от двух параметров, на рисунке 5.3 – от трёх параметров.

Рисунок 5.2 – Расположение экземпляров классов ирисов Фишера в зависимости от двух параметров: petal width и petal length

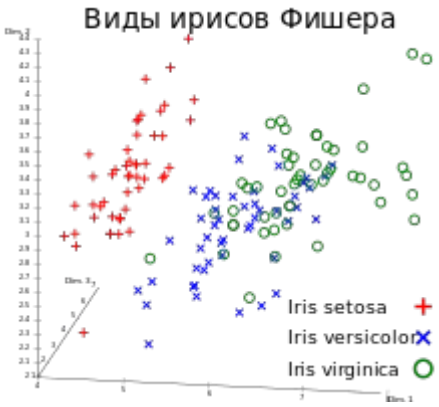


Рисунок 5.3 – Расположение экземпляров классов ирисов Фишера в зависимости от трёх параметров: petal width, petal length, sepal width

Из рисунков видно, что Iris setosa (на рисунке 5.2 – синие крестики, на рисунке 5.3 – красные крестики) линейно отделим от двух других классов, которые не слишком сильно смешаны между собой. Т.е. задача не слишком сложная, каждый класс имеет ярко выраженный собственный центр в пространстве признаков.

Вторая задача – это задача о спиралях, которые нужно разделить. Обычно имеется в виду два класса, экземпляр каждого класса – это точка в двумерном пространстве, каждый класс – это одна из непересекающихся спиралей, рисунок 5.4. Соответственно нужно отделить одну спираль от другой.

Рисунок 5.4 – Две непересекающиеся спирали

Эта задача – пример синтетических (сгенерированных человеком) данных. Есть разные варианты этой задачи: 3 спирали, 5 спиралей и т.д.

Пример пяти спиралей приведён на рисунке 5.5.

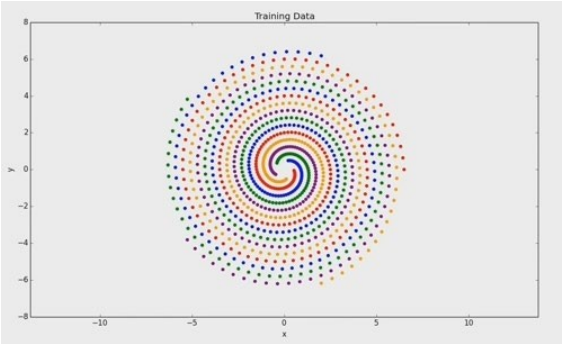


Рисунок 5.5 – Пять спиралей

Соответственно решение для подобных задач будет представлять из себя N спиралевидных поверхностей из сигмойд, на вершине каждой лежит нужный класс, а в ложбине все остальные классы. Пример такой поверхности для пяти спиралей показан на рисунке 5.6 (поверхность отображается в данном случае через ответы сети, покрашенные в разный цвет).

Рисунок 5.6 – Разделение пяти спиралей

На рисунке 5.7 показан пример поверхности отклика для двух спиралей, синий цвет – ложбина для первой спирали, красный цвет – возвышенность для второй спирали.

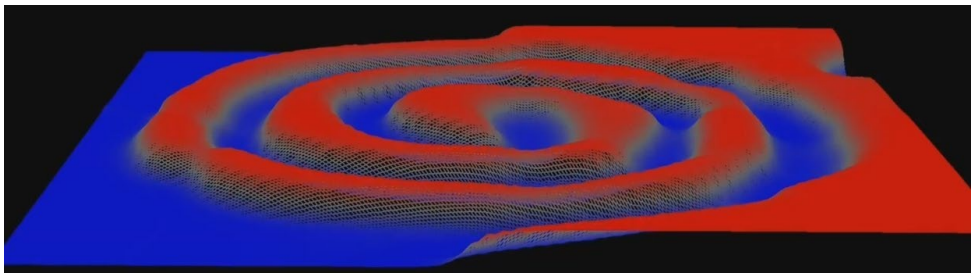


Рисунок 5.7 – Поверхность отклика для задачи о разделении двух спиралей

Для того, чтобы детально разобраться как происходит построение поверхности отклика смотрите приложение 1.

В отличие от задач первого класса, которые встречаются в реальной «природе», такие задачи для ИНС сложны и созданы главным образом для тестирования мощности той или иной сети (Так, для пяти спиралей требуется больше 16000 эпох обучения).

Задача о двух спиралях появилась впервые в [1]. В статье также приведён код на C по генерированию этого набора данных.

В таблице 5.2 приведены данные для задачи двух спиралей.

Таблица 5.2 – Данные для задачи о разделении двух спиралей

№	x	y	Class	№	x	y	Class
1	6,5	0	1	98	3,5	0,00008	-1
2	-6,5	0	-1	99	-3,37143	-0,6707	1
3	6,3138	1,2559	1	100	3,37143	0,6707	-1
4	-6,3138	-1,2559	-1	101	-3,11806	-1,29163	1
5	5,88973	2,4396	1	1	3,1180	1,2916	-1

		1		02	6	3	
6	-5,88973	-2,43961	-1	103	-2,7542	-1,84039	1
7	5,24865	3,50704	1	104	2,7542	1,84039	-1
8	-5,24865	-3,50704	-1	105	-2,29804	-2,29815	1

9	4,41941	4,41943	1	106	2,29804	2,29815	-1
10	-4,41941	-4,41943	-1	107	-1,77082	-2,65035	1
11	3,43758	5,14473	1	108	1,77082	2,65035	-1
12	-3,43758	-5,14473	-1	109	-1,19581	-2,88715	1
13	2,34392	5,65877	1	110	1,19581	2,88715	-1
14	-2,34392	-5,65877	-1	111	-0,59739	-3,00367	1
15	1,18272	5,94601	1	112	0,59739	3,00367	-1
16	-1,18272	-5,94601	-1	113	0,00008	-3	1
17	-0,00002	6	1	114	-0,00008	3	-1
18	0,00002	-6	-1	115	0,57315	-2,88104	1
19	-1,15837	5,82341	1	116	-0,57315	2,88104	-1
20	1,15837	-5,82341	-1	117	1,10029	-2,65612	1
21	-2,24829	5,42778	1	118	-1,10029	2,65612	-1
22	2,24829	-5,42778	-1	119	1,5626	-2,33847	1
23	-3,22928	4,8329	1	120	-1,5626	2,33847	-1
24	3,22928	-4,8329	-1	121	1,9446	-1,94449	1
25	-4,06589	4,06584	1	122	-1,9446	1,94449	-1
26	4,06589	-4,06584	-1	123	2,23462	-1,49303	1
27	-4,729	3,15978	1	124	-2,23462	1,49303	-1

28	4,729	-3,15978	-1	125	2,42521	-1,00447	1
29	-5,19684	2,15256	1	126	-2,42521	1,00447	-1
30	5,19684	-2,15256	-1	127	2,51328	-0,49985	1
31	-5,45563	1,08515	1	128	-2,51328	0,49985	-1
32	5,45563	-1,08515	-1	129	2,5	0,00007	1
33	-5,5	-0,00004	1	130	-2,5	-0,00007	-1
34	5,5	0,00004	-1	131	2,39065	0,4756	1
35	-5,33301	-1,06085	1	132	-2,39065	-0,4756	-1
36	5,33301	1,06085	-1	133	2,19419	0,90894	1
37	-4,96584	-2,05696	1	134	-2,19419	-0,90894	-1
38	4,96584	2,05696	-1	135	1,92273	1,28482	1
39	-4,41716	-2,95151	1	136	-1,92273	-1,28482	-1
40	4,41716	2,95151	-1	137	1,59094	1,59104	1
41	-3,71228	-3,71234	1	138	-1,59094	-1,59104	-1
42	3,71228	3,71234	-1	139	1,21525	1,81888	1
43	-2,88198	-4,31328	1	140	-1,21525	-1,81888	-1
44	2,88198	4,31328	-1	141	0,81314	1,96327	1
45	-1,9612	-4,7349	1	142	-0,81314	-1,96327	-1
46	1,9612	4,7349	-1	143	0,40231	2,02288	1
47	-0,98759	-4,9652	1	14	-0,4023	-2,02288	-1

		4		4	1		
48	0,98759	4,96524	-1	145	-0,00007	2	1
49	0,00006	-5	1	146	0,00007	-2	-1
50	-0,00006	5	-1	147	-0,37805	1,90026	1
51	0,96331	-4,84262	1	148	0,37805	-1,90026	-1
52	-0,96331	4,84262	-1	149	-0,71759	1,73225	1
53	1,86564	-4,50389	1	150	0,71759	-1,73225	-1
54	-1,86564	4,50389	-1	151	-1,00702	1,507	1
55	2,67373	-4,00141	1	152	1,00702	-1,507	-1
56	-2,67373	4,00141	-1	153	-1,23748	1,23739	1
57	3,3588	-3,35871	1	154	1,23748	-1,23739	-1
58	-3,3588	3,35871	-1	155	-1,40314	0,93748	1
59	3,89755	-2,60418	1	156	1,40314	-0,93748	-1

60	-3,89755	2,60418	-1	157	-1,50133	0,62181	1
61	4,27297	-1,76985	1	158	1,50133	-0,62181	-1
62	-4,27297	1,76985	-1	159	-1,53249	0,30477	1
63	4,47485	-0,89004	1	160	1,53249	-0,30477	-1
64	-4,47485	0,89004	-1	161	-1,5	-0,00006	1
65	4,5	0,00007	1	162	1,5	0,00006	-1
66	-4,5	-0,00007	-1	163	-1,40987	-0,28049	1
67	4,35222	0,86578	1	164	1,40987	0,28049	-1
68	-4,35222	-0,86578	-1	165	-1,27031	-0,52624	1
69	4,04195	1,6743	1	166	1,27031	0,52624	-1
70	-4,04195	-1,6743	-1	167	-1,09128	-0,72923	1
71	3,58567	2,39595	1	168	1,09128	0,72923	-1
72	-3,58567	-2,39595	-1	169	-0,88385	-0,88392	1
73	3,00515	3,00525	1	170	0,88385	0,88392	-1
74	-3,00515	-3,00525	-1	171	-0,6597	-0,9874	1
75	2,32639	3,48182	1	172	0,6597	0,9874	-1
76	-2,32639	-3,48182	-1	173	-0,43048	-1,03938	1
77	1,5785	3,81103	1	174	0,43048	1,03938	-1
78	-1,5785	-3,81103	-1	175	-0,20724	-1,04209	1

79	0,79248	3,98445	1	176	0,20724	1,04209	-1
80	-0,79248	-3,98445	-1	177	0,00004	-1	1
81	-0,00007	4	1	178	-0,00004	1	-1
82	0,00007	-4	-1	179	0,18293	-0,91948	1
83	-0,76824	3,86183	1	180	-0,18293	0,91948	-1
84	0,76824	-3,86183	-1	181	0,33488	-0,80838	1
85	-1,48297	3,58	1	182	-0,33488	0,80838	-1
86	1,48297	-3,58	-1	183	0,45143	-0,67555	1
87	-2,11817	3,16994	1	184	-0,45143	0,67555	-1
88	2,11817	-3,16994	-1	185	0,53035	-0,53031	1
89	-2,6517	2,6516	1	186	-0,53035	0,53031	-1
90	2,6517	-2,6516	-1	187	0,57165	-0,38193	1
91	-3,06609	2,0486	1	188	-0,57165	0,38193	-1
92	3,06609	-2,0486	-1	189	0,57744	-0,23915	1
93	-3,34909	1,38716	1	190	-0,57744	0,23915	-1
94	3,34909	-1,38716	-1	191	0,5517	-0,10971	1
95	-3,49406	0,69493	1	192	-0,5517	0,10971	-1
96	3,49406	-0,69493	-1	193	0,5	0,00002	1
97	-3,5	-0,00008	1	194	-0,5	-0,00002	-1

Рассмотрим на примере решения задачи ирисов Фишера общий алгоритм подбора

нейронной сети к ранее неизвестной задаче.

Не существует надёжных математических формул, с помощью которых можно однозначно подобрать оптимальную структуру ИНС к задаче (а те, что есть – для простых сетей и не всегда дают качественный результат). Поэтому желательно отталкиваться от уже готовых решений этой или похожей задач. Допустим, получить начальную структуру сети для задачи об ирисах можно с помощью команды nnstart, и выбора данной выборки из стандартных примеров Matlab (в разделе классификации). Но предположим, что нам не

удалось получить никаких идей о том, какая должна быть сеть. Тогда единственный путь методом перебора: обучить некое количество сетей и из них выбрать сеть с оптимальной структурой.

Общий план работы будет такой:

1) Разбиваем искомую выборку на три подмножества: обучающее, тестовое, валидационное (тот случай, когда это подмножество будет полезным).

2) Перебираем 25^2 сетей общей структуры 4-i-j-3, где i, j = 1..25. Два скрытых слоя – это конечно излишество для данной задачи, но они приведены для демонстрации более полного перебора. В переборе используем для контроля от переобучения только валидационную выборку. Запоминаем сеть с наилучшими i, j (таких сетей может быть несколько, поэтому запоминаем ещё и с наименьшими i и j). «Наилучшесть» сети определяется по количеству распознанных паттернов на валидационной выборке: чем больше распознали, тем лучше.

3) Далее объединяем обучающее и валидационное множество в новое обучающее, и учим нашу выбранную сеть ($4\text{-}i_{const}\text{-}j_{const}\text{-}3$). В качестве контроля используем тестовое множество, которое теперь формально будет новым валидационным.

Как видно, тестовое множество не используется в процессе подбора структуры сети, это сделано для того, чтобы это множество не стало артефактом обучения, а осталось объективной мерой оценки качества обучения. Если же структура сети известна сразу или количество возможных структур, из которых происходит выбор, будет небольшим, то можно валидационную выборку не использовать, а сразу бить все данные на обучающую и тестовую выборки.

Листинг программы с комментариями приведён ниже. Программа составлена без использования какой-либо модульности (процедур, функций, библиотек-модулей).

```
% загрузка выборки
load fisheriris;
% рисуем паттерны классов от sepal length и sepal width
gscatter(meas(:, 1), meas(:, 2), species, 'rgb', 'osd');
xlabel('Sepal length');
ylabel('Sepal width');

% бьём искомую выборку на три части: тестовое, валидационное и
% обучающее множество
colIndx = 1:4;
trainSeto = meas(1:35, colIndx);
trainVers = meas(51:85, colIndx);
trainVirg = meas(101:135, colIndx);
valSeto = meas(36:43, colIndx);
valVers = meas(86:93, colIndx);
valVirg = meas(136:143, colIndx);
```

```
testSeto = meas(44:50, colIndx);
testVers = meas(94:100, colIndx);
testVirg = meas(144:150, colIndx);
% формируем ответы учителя
a = [-1 -1 +1]';
b = [-1 +1 -1]';
c = [+1 -1 -1]';
% Заново объединяем все множества в одно, но паттерны уже
% располагаются в нужном нам порядке и оно транспонировано
initSet = [trainSeto' trainVers' trainVirg' valSeto' valVers' valVirg' testSeto' testVers'
testVirg'];
T = [repmat(a, 1, length(trainSeto)) repmat(b, 1, length(trainVers)) repmat(c, 1,
length(trainVirg)) repmat(a, 1, length(valSeto)) repmat(b, 1, length(valVers)) repmat(c, 1,
length(valVirg)) repmat(a, 1, length(testSeto)) repmat(b, 1, length(testVers)) repmat(c, 1,
length(testVirg))];
% оформляем ответы и транспонируем три подмножества
trainSet = [trainSeto' trainVers' trainVirg'];
trainT = [repmat(a, 1, length(trainSeto)) repmat(b, 1, length(trainVers)) repmat(c, 1,
length(trainVirg))];
valSet = [valSeto' valVers' valVirg'];
valT = [repmat(a, 1, length(valSeto)) repmat(b, 1, length(valVers)) repmat(c, 1,
length(valVirg))];
testSet = [testSeto' testVers' testVirg'];
testT = [repmat(a, 1, length(testSeto)) repmat(b, 1, length(testVers)) repmat(c, 1,
length(testVirg))];
% инициализируем некоторые начальные переменные перед обучением
size = 25;
trainCor = zeros(size, size);
valCor = zeros(size, size);
MAX = 0;
max_i = 0;
max_j = 0;
% перебираем возможные структуры сетей и проводим обучение
for i = 1:size,
    for j = 1:size,
        net = newff(initSet, T, [i j], {'tansig' 'tansig' 'tansig'}, 'trainbfg');
        net = init(net);
          % бить будем через диапазоны индексов
        net.divideFcn = 'divideind';
        % для обучающего множества
        net.divideParam.trainInd = 1:105;
        % для валидационного
        net.divideParam.valInd = 106:129;
        % максимальное количество ошибок на валидационном
        % множестве, имеется ввиду, что ошибка обучения падает,
        % а ошибка на валидационном множестве растёт
        net.trainParam.max_fail = 2;
          % обучаем сеть
        [net, tr] = train(net, initSet, T);
          % получаем ответы сети на обучающей части
        Y = sim(net, trainSet);
          % получаем ответы сети на валидационной части
```

```
        Z = sim(net, valSet);
          % высчитываем процент правильных ответов
        col = 0;
        for k = 1:length(trainSet)
          % процент правильных ответов определяется через евклидово
          % расстояние между ответом учителя и ответом сети,
          % величина расхождения – не более 0.01
            if norm(trainT(k)-Y(k)) < 0.01
              col = col + 1;
            end
        end
        trainCor(i, j) = 100 * col / length(trainT);

        col = 0;
        for k = 1:length(valT)
            if norm(valT(k)-Z(k)) < 0.01
              col = col + 1;
            end
        end
        valCor(i, j) = 100 * col / length(valT);
         % определяем максимальное количество ответов, причём
         % сохраняем сеть с минимальной архитектурой (в условии
         % используем “>”, а не “>=”
        if valCor(i, j) > MAX
          MAX   =   valCor(i,
          j); max_i = i;
          max_j = j;
        end
    end
end

figure
% рисуем получившиеся матрицы
surf(1:size, 1:size, trainCor);
% в 3-D
view(3)
% в 2-D
view(2)
figure
surf(1:size, 1:size, valCor);
view(2);
% учим выбранную сеть до тех пор, пока количество правильных
% ответов на всех ирисах не станет больше 90%
Q = 10;
while (Q < 90)
        net = newff(initSet, T, [max_i max_j], {'tansig' 'tansig' 'tansig'}, 'trainbfg');
        net = init(net);
        net.divideFcn = 'divideind';
          % объединяем обучающее и валидационное множество
        net.divideParam.trainInd = 1:129;
          % новое валидационное множество теперь то, что было
          % зарезервировано под тестовое
```

```
    net.divideParam.valInd = 130:150;
    net.trainParam.max_fail = 3;
    [net, tr] = train(net, initSet, T);
    Z1 = sim(net, initSet);
    col = 0;
    for k = 1:length(T)
       if norm(T(k)-Z1(k)) < 0.01
            col = col + 1;
       end
    end
    Q = 100 * col / length(T);
en
d
```

Сначала выведем распределение классов, результат показан на рисунке 5.8.

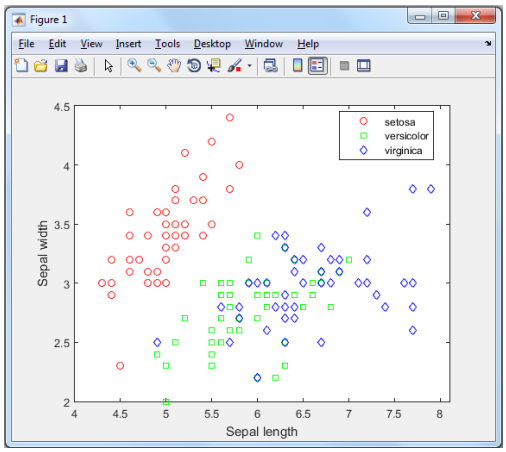


Рисунок 5.8 – Распределение паттернов классов в зависимости от двух параметров: sepal width и sepal length

Затем разобьём исходное множество на три выборки: валидационное, тестовое, обучающее. Каждый класс представлен 50 паттернами, 35 из них оставим для обучающей выборки, 8 – для валидационной и 7 – для тестовой.

Стоит отметить, что разбитие выборки на три множества – серьёзное дело. В учебной задаче можно обойтись простыми диапазонами не углубляясь в то, какие данные попадут в эти диапазоны, в реальных же задачах так бить нельзя. В одни диапазоны могут попасть крупные выбросы (нетипичные значения), в другие – усреднённые данные. В результате может получиться, что, допустим, валидационная и тестовая выборки будут составлены некорректно.

Когда мы загрузили выборку (**load fisheriris**), то массив species стал содержать названия ирисов, но ИНС работает только с числами, поэтому нам необходимо создать ответы учителя. Классов три, поэтому и выходов нейронов – три. Правильный нейрон кодируется +1, неправильный -1.

При создании сети с помощью **newff()** необходимо обозначить, что выходной нейрон

будет обладать нелинейной функцией активации, иначе по умолчанию, среда создаст сеть с нейронами, имеющими на выходе функцию **purelin**, рисунок 5.9.

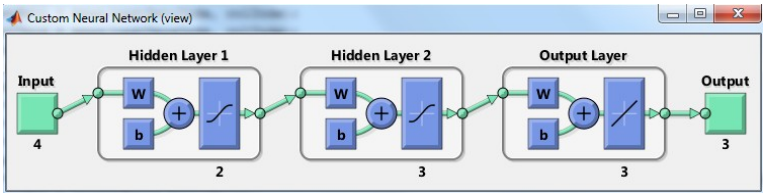


Рисунок 5.9 – Сеть, которая создастся по умолчанию, если в **newff()** прямо не указать, что нужны функции **tansig**

Процент правильно распознанных паттернов из обучающей выборки приведён на рисунке 5.10 (массив trainCor).

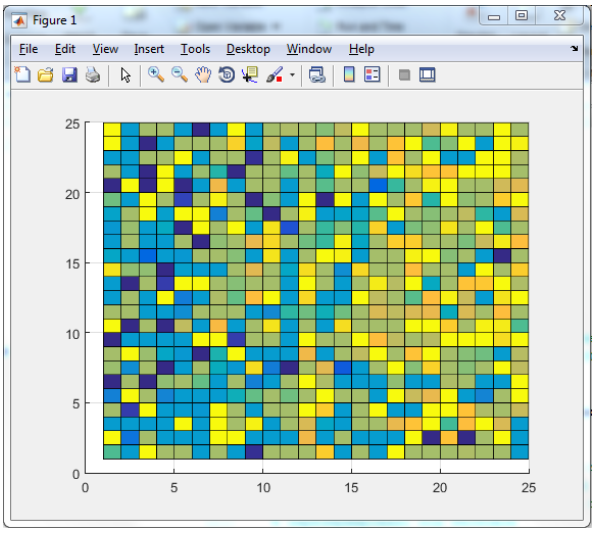


Рисунок 5.10 – Массив trainCor, желтые цвета соответствуют высокому проценту распознанных паттернов, синие - низкому

На рисунке 5.11 приведён массив valCor, именно по нему определяется первый максимально правильный ответ. В данном случае max = 100, max_i = 1, max_j = 3.

Рисунок 5.11 – Массив valCor

Можно наблюдать высокую корреляцию между двумя массивами, что говорит о хорошем разбиении искомого множества на валидационную и обучающую выборки. Также видно, что правый верхний угол матрицы значительно желтее, чем нижний левый, что говорит о том, что с ростом скрытых слоёв качество решения задачи улучшается.

Пример обучения сети со структурой 4-25-25-3 приведён на рисунке 5.12.

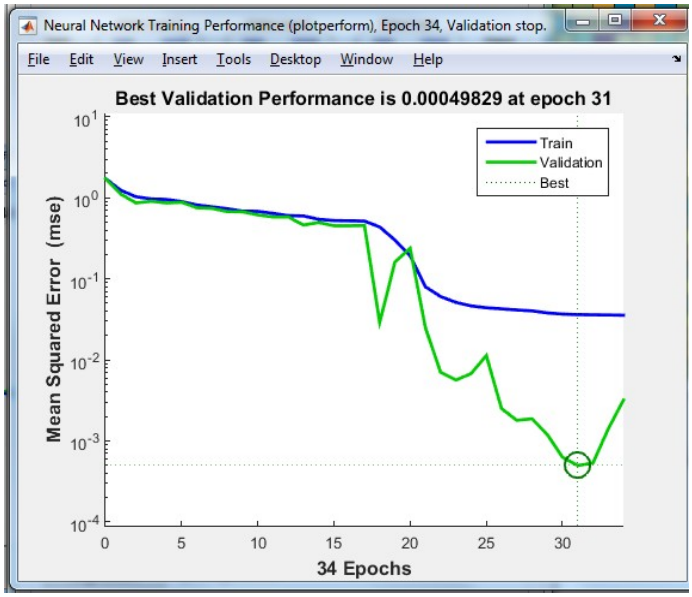


Рисунок 5.12 – Видно, что на 31 эпохе на валидационной выборки был достигнут лучший результат, потом произошёл рост ошибки, а так как значение max_fail установлено в 2, то очень быстро обучение было прервано, чтобы сеть не переобучилась и не превратилась в память

После выбора нужной структуры (4-1-3-3) необходимо обучить данную сеть. В коде использовался критерий оценки количества правильных результатов на всей

первоначальной выборки, но также можно использовать критерий оценки только на тестовой выборке. Стоит обратить внимание также на то, что первоначальная обучающая и валидационная выборки были объединены в новую обучающую выборку, а искомая тестовая выборка стала новой валидационной.

Далее рассмотрим решение задачи о спирали. Эта задача считается сложной, т.к. даже не разбивая искомую выборку на тестовое и обучающее множество тяжело заставить сеть построить спиральную поверхность отклика. Решим эту задачу с помощью сетей разной структуры и разных алгоритмов обучения.

Как видно из рисунка 5.7 поверхность отклика должна быть спиральной, причём один класс должен лежать на возвышенности этой поверхности, а другой – в низменности. Первый скрытый слой нейронов моделирует сигмойды, а второй объединяет их в более сложные поверхности. Становится очевидным, что для данной задачи достаточно сети со структурой 2-N-1, где собственно выходной нейрон и объединит сигмойды (сориентирует их так в пространстве), чтобы они объединились в спираль. Необходимо теперь выбрать значение N. Как уже отмечалось, точных формул нет, есть примерные формулы. Для сети с одним скрытым слоем формула имеет вид:

$$N \approx \sqrt{I * O}, \tag{5.1}$$

где N – количество нейронов в скрытом слое, I, O – количество нейронов во входном и выходном слоях соответственно.

Для сети с двумя скрытыми слоями будет так:

$$r \approx \sqrt[3]{\frac{I}{O}} \tag{5.2}$$

$$N \approx O * r^2$$

$$N_1 \approx O * r, \tag{5.3}$$

где N_1 и N_2 – количество нейронов в первом и втором слоях соответственно.

Как нетрудно убедиться, формула (5.1) для задачи о спирали не работает, т.к. $N \approx 2 \approx 1$, что, разумеется, не даст решения задачи. Поэтому остаётся либо применять постепенный рост сети и смотреть на результат, либо искать в литературе примерный размер. Из [3] становится ясно, что $N \geq 40$. Учитывая, что мы отнимем часть выборки на тестовое множество, которое не будет участвовать в обучении, т.е. усложним и без того сложную задачу, возьмём N=60.

Ниже приведён код решения данной задачи.

% Загружаем данные из Excel

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
% Отображаем данные на графике
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
% Транспонируем множество и метки к нему
initSet = initSet';
T = T';
% Создаём сеть с одним скрытым слоем, на нём и на выходном
% слое устанавливаем ф.а. как гиперболический тангенс
% способ обучения trainlm – Левенберга-Марквардта
net = newff(initSet, T, [60], {'tansig' 'tansig'}, 'trainlm');
% Максимальное количество эпох
net.trainParam.epochs = 5000;
net = init(net);
% Способ разбиения на тестовое и обучающее множество – случайный
net.divideFcn = 'dividerand';
% Обучающее множество 85% = 165 паттернов
net.divideParam.trainRatio = 0.85;
% Тестовое множество 15% = 29 паттернов
net.divideParam.testRatio = 0.15;
net.divideParam.valRatio = 0;
% Минимальный уровень нормы для вектора градиента
net.trainParam.min_grad = 1e-9;
[net, tr, Y, E] = train(net, initSet, T);
% Определяем квадратную область вокруг спирали, тут
% по оси X и Y будет от -12 до 12, т.к. изначально вся спираль
% умещалась от -8 до 8 по X и от -6 до 6 по Y.
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
% вектор для ответов сети на этом квадрате
pp = [P1(:) P2(:)]';
% получаем ответы сети
aa = net(pp);
% возвращаемся к исходному рисунку со спиралью
figure(1)
% включаем режим добавления новой информации на этот рисунок,
% без него точки будут стёрты нашей раскраской
hold on;
% рисуем сетку
grid on;
% рисуем и закрашиваем сетку.
% -5 тут необходим для того, что без него только половина меток
% отобразится поверх раскраски, нам нужно понизить значения
% сетки до отрицательных величин, и тогда вся спираль будет
% видна поверх новой раскраски рисунка
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

На примере этой задачи рассмотрим, как загружать данные из Excel. До этого момента данные либо генерировались Matlab, либо, как в случае с задачей об ирисах Фишера, извлекались из стандартных баз данных Matlab. Обычно же данные содержатся в табличной форме во внешнем файле. Для унификации будем всегда полагать, что этот внешний файл – файл Excel.

Сначала необходимо перенести данные из таблицы 5.2 в Excel, помня, что в Excel разделитель для вещественных чисел – это запятая, а не точка. В ячейках необходимо установить числовой формат. Пример части данных, уже перенесённых в Excel, приведён на рисунке 5.13. В данном случае данные располагаются по столбцам.

	A	B	C	D
1	№	X	Y	Class
2	1	6,5	0	1
3	2	-6,5	0	-1
4	3	6,3138	1,2559	1
5	4	-6,3138	-1,2559	-1
6	5	5,88973	2,43961	1
7	6	-5,88973	-2,43961	-1
8	7	5,24865	3,50704	1
9	8	-5,24865	-3,50704	-1
10	9	4,41941	4,41943	1
11	10	-4,41941	-4,41943	-1
12	11	3,43758	5,14473	1

Рисунок 5.13 – Пример данных в файле Excel

К этой лабораторной работе прилагается файл «DataForSpiral.xlsx», который содержит всю искомую выборку.

Сама же подгрузка столбцов будет осуществляться с помощью функции **xlsread()**, где первый параметр – это путь к файлу, а третий – диапазон по столбцам. Эта функция перегружена, поэтому, чтобы узнать другие способы её вызова можно обратиться к справке Matlab (F1). Входные данные поместим в initSet, а метки в T. Стоит сказать, что для данной задачи на выходе достаточно одного нейрона, принимающего значения от -1 до 1, хотя можно было бы и закодировать избыточно (разряженно) с помощью двух нейронов: [-1 +1] – для первого класса, [+1 -1] – для второго класса.

Вывод точек осуществляем с помощью функции **gscatter()**, результат показан на рисунке 5.14.

Рисунок 5.14 – Вывод точек спиралей для двух классов

Далее создаём сеть с помощью функции **newff(initSet, T, [60], {'tansig' 'tansig'}, 'trainlm')**. Мы задаём количество скрытых нейронов, функции активации и способ обучения. С помощью **net.trainParam.min_grad = 1e-9** мы устанавливаем минимальное значение нормы вектора градиента, достигнув которого, процесс обучения остановится. Дело в том, что алгоритм Левенберга-Марквардта (trainlm) быстро достигнет локального минимума, поэтому это значение можно и уменьшить, но тогда сеть может переобучиться, растеряв часть своих навыков.

Результат закрашивания квадратной области, в пределах которой находятся точки спирали, показан на рисунке 5.15.

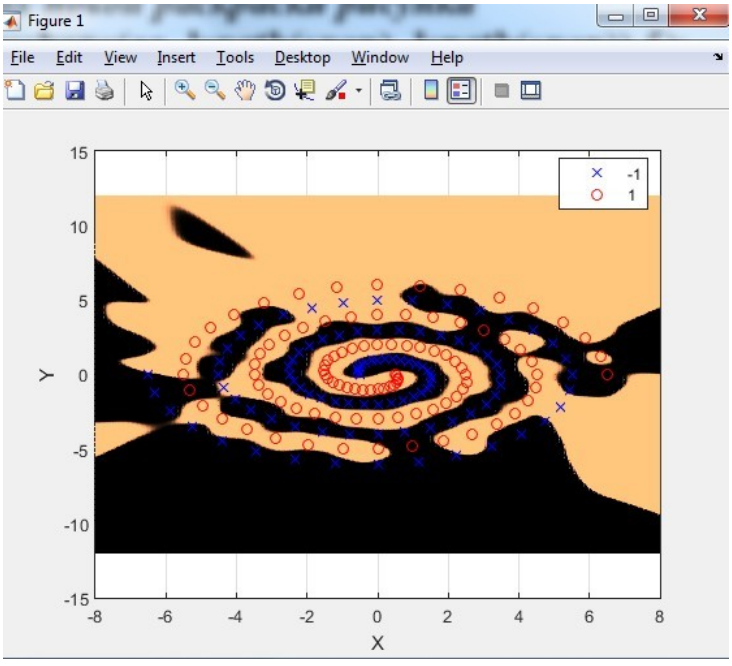


Рисунок 5.15 – Результат обучения сети

Видно, что не все представители классов двух спиралей попали на свой цвет, что и не удивительно, ведь в процессе обучения использовалась не вся спираль, а только её часть (напомним, что задачу о спиралях часто используют для тестирования алгоритмов обучения: насколько хорошо будет построена в процессе достижения локального или глобального минимума на поверхности ошибок искомая спиральная поверхность отклика. Бить на обучающее и тестовое множество не обязательно, задача и так сложна).

Поведение сети на тесте и на обучающем множестве показано на рисунке 5.16.

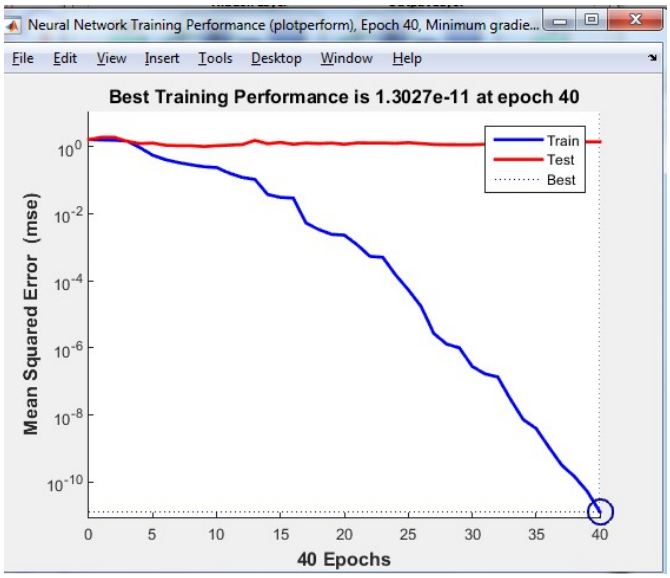


Рисунок 5.16 – Ошибка обучения и обобщения для 40 эпох

Из графиков видно, что снижать норму вектора градиента для продолжения обучения опасно, т.к. красный график (ошибка обобщения) потихоньку начинает ползти вверх.

В данной задаче тестовое множество специально не выделялось в коде, но индексы (29 штук), сгенерированные средой хранятся в **tr.testInd**, поэтому для получения конкретных ответов сети на тестовом множестве, можно написать **Q = sim(net, initSet(tr.testInd))**. Можно сравнить эти значения с ответами учителя: **T(tr.testInd)**. Из результатов сравнения становится видно, что даже несмотря на малую ошибку для обучающего множества сеть с большим трудом моделирует данную спиральную зависимость для тестового множества. Добиться для данной задачи необходимой поверхности отклика значительно проще, чем нужного ответа для тестового множества. С чем это связано?

В реальных задачах экземпляры классов обычно располагаются кучно друг по отношению к другу. Они образуют кластеры в пространстве признаков. Связано это с тем, что за каждым классом стоит некий инвариант, который в той или иной степени

проявляется в каждом экземпляре. Поэтому, если обучающая и тестовая выборка составлены репрезентативно, то низкая ошибка на обучающем множестве в целом должна приводить к низкой ошибке на тестовом множестве, т.к. гиперплоскости должны быть настроены и отделять одни кластеры от других в процессе обучения. В данной же задаче поверхность кластеров не выпуклая, более того, кластер одного класса глубоко заходит в кластер другого класса. В таких условиях угадать какому кластеру принадлежит экземпляр очень сложно. Этот пример демонстрирует ту особенность, что если зависимость сложная или в данных много хаоса (не шума), то хорошая ошибка обучения вообще ничего не скажет о том, как поведёт себя сеть на тестовом множестве. Вот почему очень сложно строить модели для предсказаний на финансовых рынках или модели для задачи классификации вроде рассмотренной.

Далее сравним различные алгоритмы обучения на примере задачи о спирали. Разбивать на тестовую и обучающую выборки уже не будем. Задача упрощается: смоделировать поверхность отклика для сети (рисунок 5.7), соответственно критерием качества будет моделируемая поверхность, а также количество эпох, которое потребовалось на это.

Подробную справку по алгоритмам обучения можно получить через команду **help nntrain**.

Рассмотрим следующие алгоритмы: **trainlm** (алгоритм Левенберга-Марквардта)**, trainbfg** (метод BFGS)**, traingd** (обычный алгоритм обратного распространения)**, traingdm** (алгоритм обратного распространения с моментом)**, trainrp** (алгоритм RPROP).

Для более подробного ознакомления рекомендуется [4–6].

Обучения ИНС – это поиск минимума на поверхности ошибок. Все методы обучения можно разбить на два класса: локальные и глобальные. Глобальные пытаются найти глобальный минимум, самый известный класс из таких методов – это генетические алгоритмы. Локальные методы ориентированы на пошаговый спуск из некоторой точки на поверхности ошибок к локальному минимуму. Локальные методы можно разбить на три класса в зависимости от того какая информация ими используется для спуска: методы нулевого порядка (не используют производные), методы первого порядка (используют первые производные, - обычно для получения вектора градиента), методы второго порядка (используют информацию от вторых производных (матрица Гесса, матрица Якоби)). В целом зависимость такая: чем выше порядок метода, тем он считается эффективнее для спуска, но в тоже время требует больше вычислений и памяти для хранения дополнительных структур (матриц). Если сети большие, то придётся отказаться от методов второго порядка, т.к. потребуется слишком много памяти и времени на обучение.

Глобальные методы часто используются для преднастройки сети перед использованием локальных методов.

Алгоритмы **trainlm** и **trainbfg** – алгоритмы второго порядка, все остальные – первого порядка. Между рассматриваемыми алгоритмами можно выделить следующую взаимосвязь, таблица 5.3.

Таблица 5.3 – Взаимосвязь между рассматриваемыми алгоритмами

№	Тип алгоритма	Скорость, эффективность	Память
1	Trainlm	Самый быстрый, во многих функциях стоит по умолчанию	Расходует больше всех памяти
2	Trainbfg	Более медленный, чем Trainlm	Расходует меньше памяти, чем Trainlm
3	Trainrp	Более медленный, чем Trainbfg	Расходует меньше памяти, чем Trainbfg
4	Traingdm	Очень медленный	Расходует меньше памяти, чем Trainrp
5	Traingd	Очень медленный	Расходует меньше памяти, чем Trainrp, отличий от Traingdm практически нет

Листинг по применению алгоритма trainlm приведён ниже. Стоит обратить внимание, что здесь использована практически полная форма вызова функции **newff()**, а также установлен параметр **net.trainParam.epochs = 60.** Дело в том, что если не устанавливать этот параметр, то алгоритм остановит свою работу, когда модуль вектора градиента станет близким к нулю, но, если при этом изучить график ошибки обучения (синий цвет), то окажется, что при использовании этого алгоритма минимум достигается очень быстро (в среднем 50-60 эпох), а остальные силы (порядка 150 эпох) идут на «топтание на месте» (приближение к асимптоте). В результате этого процесса сеть всё больше становится похожа на память и всё больше теряет обобщающие способности, поэтому имеет смысл остановить этот процесс. Получившаяся поверхность отклика представлена на рисунке 5.17.

Переменная **pref** содержит результирующую ошибку обучения (разницу между метками учителя и реальными ответами сети).

initSet = xlsread('C:\Users\Computer Grand\Desktop\Для

```
студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для
```

```
студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
% Вместо learngd можно применить learngdm, вместо mse
% msereg – с регуляризацией, crossentropy – кросс-энтропия
net = newff(initSet, T, [60], {'tansig' 'tansig'}, 'trainlm', 'learngd', 'mse', {}, {}, 'dividerand');
net.divideParam.trainRatio = 1;
net.divideParam.testRatio = 0;
net.divideParam.valRatio = 0;
net.trainParam.epochs = 60;
net = init(net);
[net, tr] = train(net, initSet, T);
% один из способов получения ответов сети
y = net(initSet);
% высчитываем разницу между ожидаемыми и реальными ответами
% PREF ≈ 0.0014
perf = perform(net, T, y);
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

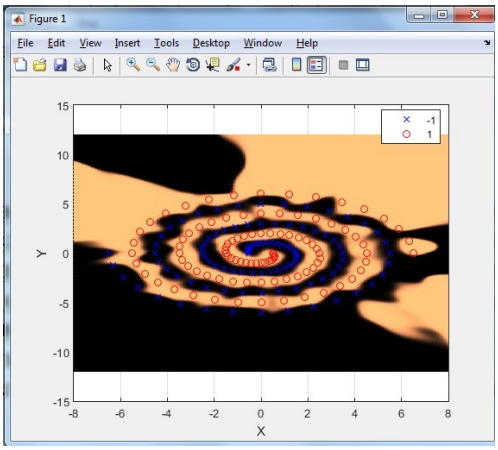


Рисунок 5.17– Поверхность отклика после применения алгоритма trainlm

Ниже приведён листинг для использования алгоритм **trainbfg**. Как видно, было изменено количество эпох, которые примерно потребуются для нахождения решения. Это количество выросло. Количество нейронов в скрытом слое также было снижено до 50.

Найденное решение показано на рисунке 5.18. Изменение ошибки обучения

показано на рисунке 5.19.

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
net = newff(initSet, T, [50], {'tansig' 'tansig'}, 'trainbfg', 'learngdm', 'mse', {}, {}, 'dividerand');
net.divideParam.trainRatio = 1;
net.divideParam.testRatio = 0;
net.divideParam.valRatio = 0;
net.trainParam.epochs = 250;
net = init(net);
[net, tr] = train(net, initSet, T);
y = net(initSet);
% PREF == 7.4596e-08;
perf = perform(net, T, y);
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

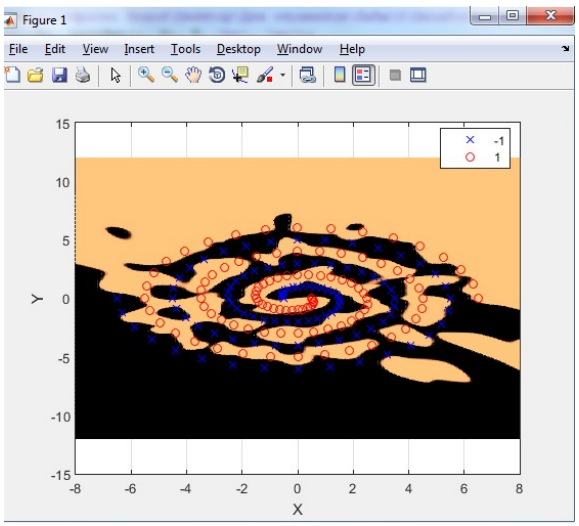


Рисунок 5.18 – Найденное решение с использованием алгоритм trainbfg

Рисунок 5.19 – Изменение величины ошибки обучения за 250 эпох

Далее рассмотрим применение алгоритма **trainrp**, но вместо функции **newff()** будем использовать **feedforwardnet()**. В пакете matlab очень часто существует иерархия функций, которые выполняют, по сути, одно и тоже, но с разной степенью детализации для пользователя. Для создания сетей прямого распространения можно выделить условно следующую иерархию: network()✉newff()✉feedforwardnet()✉patternnet().

Ниже приведён листинг для обучения с помощью алгоритма trainrp (RPROP). Стоит обратить внимание, что количество эпох уже исчисляется тысячами. Сеть создавали с помощью функции **feedforwardnet()**, которая получает в качестве входных параметров количество нейронов в скрытом слое и алгоритм обучения. У данного алгоритма обычно настраивают такие параметры обучения как **net.trainParam.delt_inc, net.trainParam.delt_dec, net.trainParam.delta0, net.trainParam.deltamax, net.trainParam.lr**, однако, учитывая, что для осмысленного регулирования этих параметров необходимо знание тонкостей работы алгоритмы, мы их оставили настроенными по умолчанию. Для ознакомления с этими параметрами можно использовать команду **help trainrp**. Полученное решение показано на рисунке 5.20, а изменение значения ошибки обучения на рисунке 5.21.

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
net = feedforwardnet(60, 'trainrp');
net = configure(net, initSet, T);
net.layers{2}.transferFcn = 'tansig';
net.divideParam.trainRatio = 1;
```

```
net.divideParam.testRatio = 0;
net.divideParam.valRatio = 0;
net.trainParam.epochs = 6000;
net = init(net);
[net, tr] = train(net, initSet, T);
y = net(initSet);
perf = perform(net, T, y);
% PREF == 1.7276e-05.
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

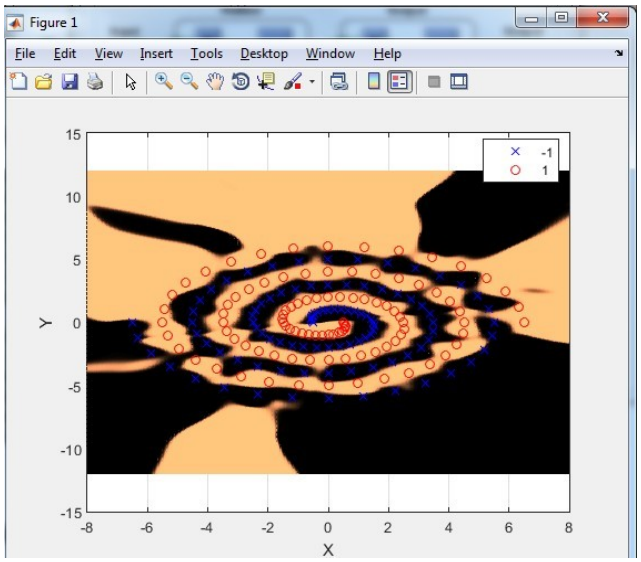


Рисунок 5.20 – Найденное решение с использованием алгоритма trainrp

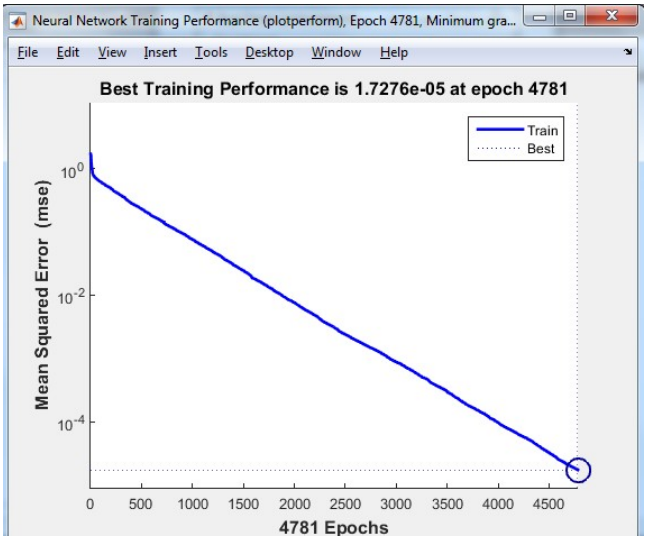


Рисунок 5.21 – Изменение величины ошибки обучения за 5000 эпох

Далее рассмотрим сеть с двумя скрытыми слоями для решения задачи о спиралях.

Формулы (5.2) и (5.3) тут также не подойдут для выбора размеров скрытых слоёв. Ориентируясь на различные источники выберем структуру 2-10-10-1.

Листинг обучения такой сети приведён ниже.

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
net = feedforwardnet([10 10], 'trainlm');
net = configure(net, initSet, T);
net.layers{3}.transferFcn = 'tansig';
net.divideParam.trainRatio = 0.85;
net.divideParam.testRatio = 0.15;
net.divideParam.valRatio = 0;
net.trainParam.epochs = 6000;
net.trainParam.min_grad = 1e-10;
net = init(net);
[net, tr] = train(net, initSet, T);
y = net(initSet);
perf = perform(net, T, y);
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
Q = sim(net, initSet(tr.testInd))
T(tr.testInd)
```

Подсчитав количество совпадений между векторами Q и T найдём, что качество работы сети выросло на неизвестных примерах с 28% до 50%.

Теперь создадим пользовательскую сеть, которая решает задачу о спирали. Структура для этой сети позаимствована из [1], рисунок 5.22.

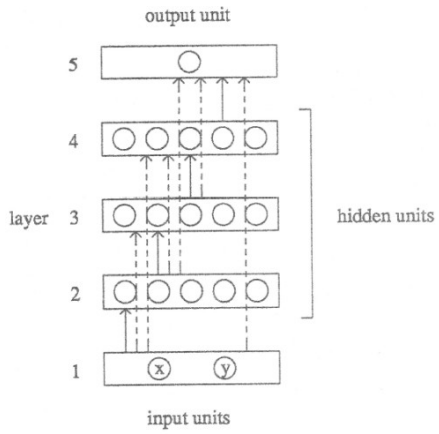


Рисунок 5.22 – Сверху структура предлагаемой сети в Matlab, снизу она же в виде обобщённой схемы

Как пишут авторы статьи, такая структура была выбрана потому, что они допустили, что каждый синапс-вес может с лёгкостью запомнить-хранить 1.5 бит информации (судя по всему предполагается, что вещественное число может в частности принять три значения: два концевых, допустим, 0 и 1, и одно промежуточное, допустим, 0.5. Таким образом, это не два бита информации, но и не один). Всего в выборке 194 паттерна и для их хранения им потребуется 194 / 1.5 = 130 весов. Далее они подбирали сеть с несколькими слоями, чтобы она содержала примерно 130 весов. Сеть на рисунке 5.22 имеет 136 параметров. Можно это число получить сразу (после создания сети набрать **net.numWeightElemetns**) или подсчитать вручную: первый слой имеет два нейрона и связан с тремя другими слоями ((5+5)*3), второй слой связан с тремя слоями, два из них по 5 нейронов и один содержит один нейрон, сам же этот первый скрытый слой имеет тоже 5 нейронов (25+25+5), третий слой связан с двумя слоями (25+5), и для четвёртого слоя – 5. Также учитываем количество смещений для всех скрытых слоёв и выходного слоя – 16. Получаем 120+16=136. Что касается связей через слой, то они нужны, чтобы частично скомпенсировать затухание градиента. Если их убрать и оставить только связи от последовательных слоёв, то невязка, распространяемая с выходного слоя, дойдя до входного, совсем затухнет (примет маленькие значения), и обучение будет неэффективным. Поэтому связи через один-два слоя

– вынужденная мера. Стоит отметить, что вообще с затуханием градиента борются через введение общих весов (сверточные сети).

Листинг программы с подробными комментариями приведён ниже. Стоит обратить внимание на применение функции **network()**. В лабораторной работе 4 объяснялись её параметры.

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
% создание пользовательской сети
net = network(1, 4, [1; 1; 1; 1], [1; 1; 1; 0], [0 0 0 0; 1 0 0 0; 1 1 0 0; 1 1 1 0], [0 0 0 1]);
% размеры скрытых слоёв
net.layers{1}.size = 5;
net.layers{2}.size = 5;
net.layers{3}.size = 5;
% функции активации на скрытых слоях
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';
net.layers{3}.transferFcn = 'tansig';
% функция активации на выходном слое
net.layers{4}.transferFcn = 'tansig';
% способ разделения обучающего множества: случайно
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 1;
net.divideParam.testRatio = 0;
net.divideParam.valRatio = 0;
% алгоритм обучения RPROP, нужно учесть, что чем сложнее
% структура пользовательской сети, тем более осторожно
% нужно применять мощные алгоритмы, т.к. они могут не сработать
% в силу отсутствия реализации некоторых тонкостей настройки
% этих алгоритмов к пользовательским архитектурам.
% trainrp, traingd, traingdm – как раз простые алгоритмы, которые
% подойдут практически ко всему
net.trainFcn = 'trainrp';
% функция ошибки – mse, ставить, допустим, crossentropy удастся
% не всегда, т.к. многие алгоритмы рассчитаны на mse или msereg
net.performFcn = 'mse';
% максимальное количество эпох, ставим побольше
net.trainParam.epochs = 35000;
net.trainParam.goal = 0;
% начальная скорость обучения
net.trainParam.lr = 0.01;
net.trainParam.max_fail = 4;
% настройки RPROP
net.trainParam.delt_inc = 1.2;
```

```
net.trainParam.delt_dec = 0.5;
net.trainParam.delta0 = 0.07;
net.trainParam.deltamax = 50.0;
% минимальный размер модуля градиента для остановки алгоритма
% для более мощных алгоритмов можно установить 1e-10, для слабых
% от 1e-5 до 1e-7
net.trainParam.min_grad = 1e-7;
% подключаем график изменения ошибки обучения
net.plotFcns = {'plotperform'};
% чтобы на пользовательской архитектуре работал алгоритм
% обратного распространения ошибки, нужно вручную для слоёв
% установить инициализацию весов и правило обновления-обучения
% установка правила обучения для весов
net.adaptFcn = 'adaptwb';
% net.layerWeights{i, j} – это матрица 4x4, показывающая какой слой
% с каким связан, в ней заполнены все элементы ниже главной
% диагонали, кроме элемента {4, 4}, поэтому для них для всех нужно
% установить правило обучения
net.layerWeights{1}.learnFcn = 'learngdm';
net.layerWeights{2}.learnFcn = 'learngdm';
net.layerWeights{3}.learnFcn = 'learngdm';
net.layerWeights{3,2}.learnFcn = 'learngdm';
net.layerWeights{4}.learnFcn = 'learngdm';
net.layerWeights{4,2}.learnFcn = 'learngdm';
net.layerWeights{4,3}.learnFcn = 'learngdm';
% тоже, но для смещений
net.biases{1}.learnFcn = 'learngdm';
net.biases{2}.learnFcn = 'learngdm';
net.biases{3}.learnFcn = 'learngdm';
net.biases{4}.learnFcn = 'learngdm';
% для входного слоя
net.inputWeights{1}.learnFcn = 'learngdm';
% этот параметр сообщает, что при применении configure(), веса
% будут инициализированы с помощью выбранного нами параметра,
% мы выбрали initnw, т.е. инициализация по методу Нгуена-Видроу
net.initFcn = 'initlay';
net.layers{1}.initFcn = 'initnw';
net.layers{2}.initFcn = 'initnw';
net.layers{3}.initFcn = 'initnw';
net.layers{4}.initFcn = 'initnw';
net.biases{1}.initFcn = 'initnw';
net.biases{2}.initFcn = 'initnw';
net.biases{3}.initFcn = 'initnw';
net.biases{4}.initFcn = 'initnw';
% устанавливаем величину момента
net.trainParam.mc = 0.5;
% конфигурируем сеть, т.е. применяем к ней все наши настройки
net = configure(net, initSet, T);
% обучаем сеть
[net, tr, Y, E] = train(net, initSet, T);
y = net(initSet);
% вычисляем производительность
```

```
perf = perform(net, T, y);
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
% выводим полученное решение
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

На рисунках 5.23–5.25 приведены результаты обучения этой сети.

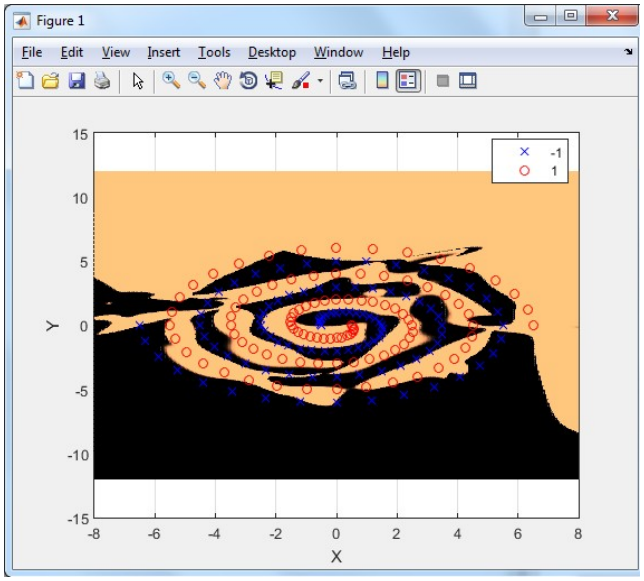


Рисунок 5.23 – Полученная поверхность отклика для пользовательской сети

Рисунок 5.24 – Результирующие параметры обучения

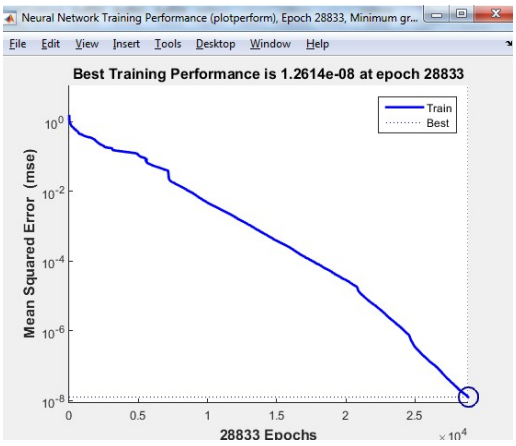


Рисунок 5.25 – Изменение величины ошибки обучения

Изменив **net.divideParam.trainRatio = 0.85, net.divideParam.testRatio = 0.15** можно

ещё раз обучить сеть, подсчитать число совпадений между векторами Q и T (как в первом листинге). Получилось порядка 65%.

Теперь можно сделать общие выводы по всем трём сетям. Будем оценивать ресурсы, обобщающую способность, аппроксимационную способность. Под ресурсами понимается количество синаптических весов, т.е. количество настраиваемых параметров. Так как мы проводили эксперименты с разным количеством нейронов для первой сети с одним скрытым слоем, то для определённости пусть будет 50 нейронов в скрытом слое. Под аппроксимационной способностью подразумевается способность построить необходимую поверхность отклика. Результаты приведены в таблице 5.4.

Таблица 5.4 – Сравнение трёх сетей с различной структурой

Структура сети	Ресурсы net.numWeightElements	Справилась с задачей	Обобщающая способность
2-50-1	201	да	30%
2-10-10-1	151	да	50%
2-5-5-5-1	136	да	65%

Обобщающую способность можно было бы оценить более точно, если провести не один эксперимент, а целую серию и усреднить результаты, подсчитав, допустим, доверительные интервалы. Что видно из этих результатов?

Во-первых, введение промежуточных слоёв позволяет уменьшить количество настраиваемых параметров.

Во-вторых, все три сети без проблем справились с постройки правильной поверхности отклика, т.е. без проблем раскрасили спирали.

В-третьих, то, что они хорошо строят поверхность отклика, не говорит ничего про их обобщающую способность! Первые две сети не справились с заданием вообще, т.к. 50% - это метод «тыка» при двух классах. Это говорит о том, что для сложных задач сложная система нелинейности (много слоёв) подходит лучше, чем простая, т.е. такие сети могут лучше справляться с задачами классификации. Более подробно про доказательства этого результата можно почитать в [7], где рассматриваются неглубокие архитектуры (shallow architecture) и глубокие архитектуры (deep architecture). Стоит также отметить, что для того, чтобы ИНС качественно предсказывали какой спирали может принадлежать точка, нужно увеличить обучающую выборку. Для двух спиралей обучающая выборка должна быть в среднем в районе 400 точек.

В приложении 1 приведён код алгоритма обратного распространения ошибки для

построения разделяющей поверхности к данным, представленным на рисунке 5.26. Это две полуспирали.

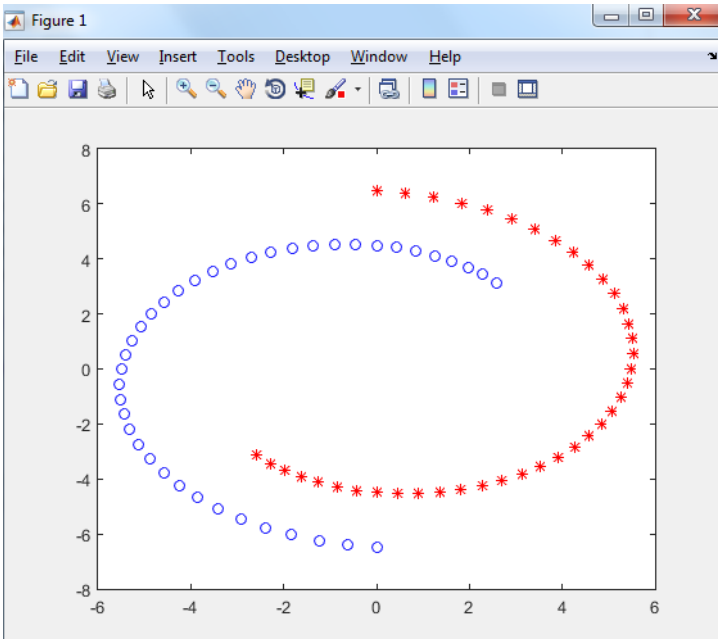


Рисунок 5.26 – Две полуспирали, которые необходимо разделить

В данном лабораторном практикуме везде используются высокоуровневые функции Matlab для обучения и создания нейронных сетей. В приложении 1 показано, как самому запрограммировать обучение сети.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

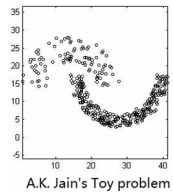
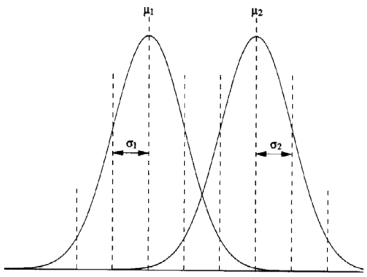
Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

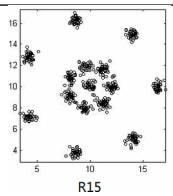
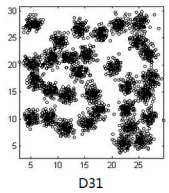
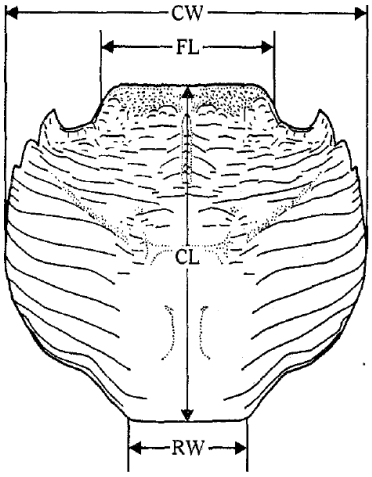
В индивидуальном варианте (таблица 5.5) дано условие задачи на классификацию. Все задачи подобраны так, что предобработки данных в целом не требуется (хотя в

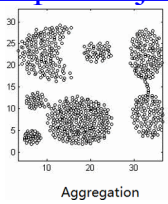
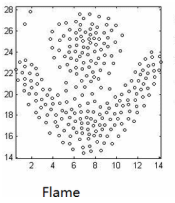
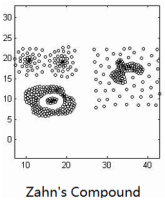
некоторых задачах её можно было бы сделать), приемлемая точность на тестовой выборке везде отмечена как 70%, что в целом не много. Достижение более большой точности распознавания оценивается в дополнительных баллах.

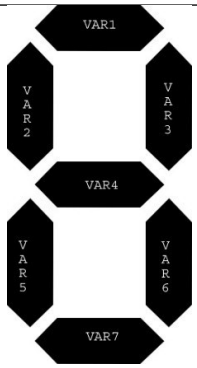
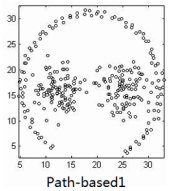
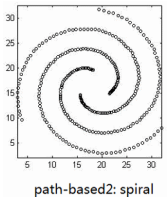
Таблица 5.5 – Варианты заданий

№	Условие задачи
1	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  A.K. Jain's Toy problem Необходимо перевести выборку в формат Excel. Затем решить задачу классификации: создать нейронную сеть, которая принимая координаты точки, относит её к тому или иному классу. Разбитие на тестовую и обучающую выборку осуществляет сам студент, исходят из задачи. Можно ориентироваться на 80%-85% для обучающей выборки и 15%-20% для тестовой. В выборках обязательно учесть степень представленности классов. Если один класс представлен 20 паттернами, а другой 5, то очевидно, что в обучающей и тестовой выборке окажутся разное количество паттернов из этих классов. Постараться получить классификатор с качеством правильных ответов не ниже 70% для тестовой выборки. Правильным ответом считается ответ сети, при котором евклидово расстояние между идеальным требуемым ответом (учитель) и ответом сети не превышает по модулю 0.1.
2	Задача о двух многомерных перекрывающихся нормальных распределениях.  Даны два двадцатимерных нормальных распределения (7400 паттернов). Одно кодируется классом 0, другое – 1. Класс 1 имеет среднее (a, a, a,…, a), класс 2 – (-a, -a, -a, …, -a), где $a \approx 2/20$. Необходимо на тестовом множестве (можно использовать 300 паттернов) добиться правильной классификации в 70%. Ошибка примерно 25% - нормальна.
3	Дана синтетическая выборка, в которой паттерны принадлежат двум кластерам в форме банана. Каждый паттерн задаётся двумя точками и типом класса: -1 или 1. Требуется разбить выборку на два множества: обучающее и тестовое и добиться правильной классификации на тестовом множестве более 70%. В этой задаче полезно для студента вывести данные на канву и оценить степень репрезентативной представленности каждого класса.
4	Синтетические данные на классификацию. Постоянный адрес задачи:

	https://cs.joensuu.fi/sipu/datasets/.

	 R15 Комментарий к варианту 1.
5	Задача про фонемы. Даны два класса: один описывает назальный звук (класс 0), другой – оральный (класс 1). Класс 0 содержит 3818 паттернов, а класс 1 – 1586 паттернов. Каждый класс описывается 5 фонемами: Aa, Ao, Dcl, Iy, Sh. Фонемы изменяются в следующих вещественных диапазонах: Aa [-1.7, 4.107], Ao [-1.327, 4.378], Dcl [-1.823, 3.199], Iy [-1.581, 2.826], Sh [-1.284, 2.719]. В физический смысл этих фонем и диапазонов вникать не нужно. Необходимо произвести классификацию и добиться на тестовом множестве качества распознавания не ниже 70%.
6	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  D31 Комментарий к варианту 1.
7	Задача о крабах genus Leptograpsus. Имеется выборка по крабам двух видов. Каждый вид представлен 50 самцами и 50 самками, т.е. вся выборка 200 измерений. Первая колонка – пол. Вторая колонка – индекс от 1 до 50, далее 5 колонок – это измерения тела краба (как в задачах об ирисах Фишера) в мм: FL, RW, CL, CW, BD. Схема измерений представлена ниже. В [8] можно получить подробную информацию об этих животных, а также о смысле эксперимента.  Последняя колонка это вид краба. Выборка бьётся на тестовую и обучающую. Для тестовой выборки подойдёт по 5 экземпляров из каждого класса: 5 самцов типа 1, 5 самок типа 1, 5 самцов типа 2, 5 самок типа 2. Всё остальное – обучающая выборка, хотя студент может предложить своё разбиение. На вход сети подаётся 6 параметров (индексы, разумеется, подавать не надо, они нужны для разбиения на обучающее и тестовое множество, а также для анализа ответов). Требуется создать и обучить сеть, которая правильно определяет тип краба на тестовой выборке с вероятностью более 70% (чем больше, тем лучше). Что такое правильный ответ написано в варианте №1, а также разобрано в задаче об ирисах Фишера.

8	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Комментарий к варианту 1.
9	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Комментарий к варианту 1.
10	Задача о классификации стекла. Имеется выборка из 214 записей. Состоит из 10 колонок (+ одна колонка индексов): RI – индекс преломления стекла, далее 8 колонок, которые обозначают содержание в процентах того или иного элемента (Na, Mg, Al, Si, K, Ca, Ba, Fe). Последняя колонка – тип стекла: 1 – window float glass, 2 – window non-float glass, 3 – vehicle glass, 4 – vehicle non-float glass (**данный класс отсутствует в данной выборке**), 5 – containers, 6 – tableware, 7 – vehicle headlamp glass. Из-за отсутствующего одного класса получается 6 типов стекла. Выборку необходимо разбить на две части: обучающую и тестовую. В данной выборке классы представлены различным количеством элементов, поэтому для тестовой каждый класс вносит не одинаковое количество паттернов: 1 – 10 паттернов, 2 – 10, 3 – 3, 5 – 3, 6 – 2, 7 – 4. Всё остальное идёт в обучающую выборку. Добиться классификации с точностью на тестовой выборке более 70%. Дополнительную информацию вместе со статьями, в которых использовалась эта выборка, можно получить по адресу: https://archive.ics.uci.edu/ml/datasets/Glass+Identification/. В Matlab можно набрать команду **nprtool**, нажать **Next**, далее **Load Example Dataset** и выбрать **Types of Glass** и почитать информацию об этой задаче. Однако, стоит учесть, что в Matlab интегрирована упрощенная задача, где на выходе имеем всего 2 класса.
11	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Комментарий к варианту 1.
12	Задача о цифрах.

	 Цифра на калькуляторе кодируется переменными VAR1..VAR7, расположенными так, как показано выше. Переменная может принимать значение 0 (ZERO), если часть табло, которая связана с этой переменной не работает, или 1 (ONE), если часть табло работает. В выборке всего 500 записей, каждая запись состоит из 7 значений переменных и класса цифры. Не все переменные правильно кодируют свой класс, т.к. табло неисправно. Необходимо составить классификатор, который на тестовом множестве покажет выше 70% правильных ответов. Можно использовать следующую структуру сети: 7-5-10, количество эпох 500, начальная скорость обучения 0.01. Разбитие на обучающую и тестовую выборки можно производить исходя из соотношений 85%-15%, степень равномерной представленности каждого класса должен сделать сам студент.
13	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Комментарий к варианту 1.
14	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Комментарий к варианту 1.

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1. Название лабораторной работы и её номер.
2. ФИО студента и группу.
3. Формулировка индивидуального задания.
4. Документ отчёта с Print Prtscr диалоговых окон по шагам для своего варианта по подобию того, что описано в теоретической части. Должна присутствовать информация о количестве классов, количестве паттернов в каждом классе, о размерах тестовой и

обучающей выборки, о представленности каждого класса в тестовой и обучающей выборке, об используемых алгоритмах обучения нейронных сетей, а также результирующие графики и процент правильно распознанных паттернов на тестовой выборке. Необходимо также указать структуру сети и обосновать почему выбрана именная такая сеть.

5. Ответы на контрольные вопросы.

Вопросы для защиты работы

1) Какие алгоритмы обучения нейронных сетей вы знаете, чем они отличаются?
2) Для чего используются в сетях прямого распространения связи через слой?
3) В каких случаях для задачи классификации выгоднее использовать рбф-нейрон, чем сигмоидальный нейрон?
4) Какой ответ сети считать правильным?
5) Чем нужно руководствоваться, разбивая выборку на тестовую и обучающую?
6) Какие функции в Matlab можно использовать для создания сети и в чём их отличие?
7) Какая структура данных содержит информацию об процессе обучения сети и как её получить?

Лабораторная работа № 6

Пример создания и обучения нейронных сетей для задач классификации в среде Matlab. Часть 2

Цель и содержание работы: создать и обучить несколько нейронных сетей, решающих задачи классификации, рассмотреть предобработку данных.

Задачи:

- рассмотреть различные способы нормировки входных данных;
- научиться кодировать категориальные переменные;
- научиться работать с вещественными переменными разных диапазонов.

Теоретическое обоснование

Рассмотрим 2 задачи, уделив особое внимание предобработки данных. Под предобработкой будет пониматься работа с категориальными, ординальными (порядковыми) и обычными числовыми переменными, и нормализация данных.

Первая задача. Оценка влияния социальных факторов на выживание пассажиров потерпевшего аварию судна «Титаник».

В интернете есть различные варианты данной выборки. Одна из наиболее старых и больших, это выборка с Delve Project [1]. Выборка состоит всего из 2201 наблюдений. Каждое наблюдение состоит из трёх параметров и результата. Первая колонка – класс, которым плыл человек на «Титанике» (0 – член команды, 1 – первый класс, 2 – второй класс, 3 – третий класс), вторая колонка – возраст (1 – взрослый, 0 - ребёнок), третья колонка – пол (1 – мужчина, 0 - женщина), четвёртая колонка – результат, по сути, - класс (1 – выжил после крушения, 0 – нет). Фрагмент выборки приведён в таблице 6.1.

Таблица 6.1 – Фрагмент выборки для задачи о «Титанике»

№	Class	Age	Sex	Survived
1	2	1	0	1
2	1	0	1	1
3	1	1	1	0
4	0	1	1	0
5	0	1	0	1
6	0	1	1	1

Выборка содержится в файле «DataForTitanic_1.xlsx». Однако данная выборка не подойдёт нам. Дело в том, что входной вектор в этом варианте выборки содержит только

категориальные и ординальные переменные и их слишком мало: Пол, Класс, Возраст (в виде категории). К чему это приводит? К тому, что данные становятся противоречивыми: слишком много одинаковых входных векторов будут иметь разные выходные классы (выжил, не выжил). Если бы мы решали данную задачу не с помощью нейронных сетей, где требуется получить один классификатор, который и принимает решение, а с помощью, допустим, random forest, то проблемы бы не было. Нам же требуется «разбавить» эти отобранные главные компоненты более второстепенной информацией. Поэтому за основу была взята выборка, содержащаяся в файле «DataForTitanic_2.xlsx», она была преобразована к финальной выборке «DataForTitanic.xlsx». Нужно учесть, что несмотря на то, что теперь данная задача стала адекватной для нейронной сети, однако серьёзных результатов ждать не приходится, т.к. на выживаемость больше всего влияют именно класс, пол и возраст, поэтому то, что мы «разбавили» входную информацию, ещё не означает, что мы сняли логические противоречия. В замаскированном виде они все равно присутствуют в выборке, т.е. данная задача неудобна для нейронной сети.

Теперь сеть имеет отличающиеся входные вектора там, где раньше они были похожи для двух разных классов, но самые влияющие на выживаемость компоненты этих векторов всё равно одинаковые.

Были оставлена такие столбцы как class (каким классом плыл), survived (выжил, не выжил), sex (пол), age (возраст в виде числа, а не в виде категории), sibsp (количество родственников второго порядка: муж, жена, братья), parch (количество родственников первого порядка: мать, отец, дети), fare (цена билета), embarked (порт посадки: 0 – Cherbourg, 1 – Queenstown, 2 – Southampton). Колонка Age содержала 263 пропуска, они были заполнены медианой по всему столбцу от этой выборки (28 лет). Колонка Fare содержала один пропуск, он также был заполнен медианой (14.45). Колонка Embarked содержала два пропуска, они были заполнены значениями «2». Именно на этой выборке будет обучаться нейронная сеть. Выборка содержит 1309 паттернов.

Вторая задача. Классификация грибов.

Имеется выборка Шлиммера в которой описаны 23 вида грибов из семейств Agaricus и Lepiota. Всего в выборке 8124 записей, каждый гриб описывается 22 параметрами (однако, один из параметров: stalk-root, может содержать пропущенные значения, поэтому он исключается из оригинальной выборки Шлиммера и остаётся 21 параметр). Необходимо классифицировать гриб как съедобный или не съедобный. Ниже приведены номера столбцов и их значения (ботанические термины переведены приближённо).

1. Форма шляпки (cap-shape: bell (колоколообразная) = b, conical (коническая) = c, convex (выпуклая) = x, flat (плоская) = f, knobbed (шарообразная) = k, sunken (впалая) = s).

2. Поверхность шляпки (cap-surface: fibrous (жилистая) = f, grooves (с выемками) = g, scaly (чешуйчатая) = y, smooth (гладкая) = s).

3. Цвет шляпки (cap-color: brown (коричневый) = n, buff (цвета буйволовой кожи) = b, cinnamon (цвета корицы) = c, gray (серый) = g, green (зелёный) = r, pink (розовый) = p, purple (пурпурный) = u, red (красный) = e, white (белый) = w, yellow (жёлтый) = y).

4. Наличие вмятин (bruises?: bruises = t, no = f).

5. Запах (odor: almond (миндаль) = a, anise (анис) = l, creosote (креозот) = c, fishy (рыбный) = y, foul (отталкивающий) = f, musty (запах плесени) = m, none (нет запаха) = n, pungent (острый) = p, spicy (запах пряностей) = s).

6. Наличие гимениальной пластинки, т.е. пластинки у гриба на нижней стороне шляпки (gill-attachment: attached (прикреплена) = a, descending (нисходящая) = d, free (свободно размещена) = f, notched (пилообразна) = n).

7. Тип разбивки на гимениальной пластинке (gill-spacing: close (близкая) = c, crowded (тесная) = w, distant (на некоторой дистанции) = d).

8. Размер гимениальной пластинки (gill-size: broad (широкий) = b, narrow (узкий) = n).

9. Цвет гимениальной пластинки (gill-color: black (черная) = k, brown (коричневая) = n, buff (цвета буйволовой кожи) = b, chocolate (шоколадная) = h, gray (коричневая) = g, green (зелёная) = r, orange (оранжевая) = o, pink (розовая) = p, purple (пурпурная) = u, red (красная) = e, white (белая) = w, yellow (жёлтая) = y).

10. Форма ножки гриба (stalk-shape: enlarging (расширяющаяся) = e, tapering (суживающаяся) = t).

11. Поверхность ножки гриба выше кольца (stalk-surface-above-ring: ibrous () = f, scaly (чешуйчатая) = y, silky (шелковистая) = k, smooth (гладкая) = s).

12. Поверхность ножки гриба ниже кольца (stalk-surface-below-ring: ibrous = f, scaly = y, silky = k, smooth = s).

13. Цвет ножки гриба выше кольца (stalk-color-above-ring: brown = n, buff = b, cinnamon (корицы) = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y).

14. Цвет ножки гриба ниже кольца (stalk-color-below-ring: brown = n, buff = b, cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y).

15. Тип раскраски (veil-type: partial (частичный) = p, universal (всеобщий) = u).

16. Цвет раскраски (veil-color: brown = n, orange (оранжевый) = o, white = w, yellow = y).

17. Количество колец (ring-number: none (нет) = n, one (одно) = o, two (два) = t).

18. Типы колец (ring-type: cobwebby (затянутое) = c, evanescent (суживающееся) = e, flaring (яркое) = f, large (крупное) = l, none (нет) = n, pendant (по типу юбки) = p, sheathing (обволакивающее) = s, zone (зональное) = z).

19. Цвет спор (spore-print-color: black = k, brown = n, buff = b, chocolate = h, green = r, orange = o, purple = u, white = w, yellow = y).

20. Популяция (population: abundant (изобильная) = a, clustered (пучочная, растущая пучками) = c, numerous (равномерная) = n, scattered (редкая, рваная) = s, several (малая) = v, solitary (одиночная) = y).

21. Место распространения (habitat: grasses (трава) = g, leaves (листва) = l, meadows (луга) = m, paths (тропы) = p, urban (город) = u, waste (кустарник) = w, woods (леса) = d).

Класс для съедобного гриба edible = e, класс для несъедобного гриба poisonous = p.

Данная выборка находится в файле «DataForMushrooms.xlsx». Скачать выборку можно с сайта DELVE Project. Более подробная информация по выборке приведена в [2].

Для начала приведём краткую полезную информацию по предобработке данных: по кодированию входных переменных, а также по нормировке. Более сложные методы, такие как выбеливание, анализ главных компонент и т.д. не рассматриваются, т.к. эти методы больше относятся к большим объёмам данным (направление data mining) и для тренировочных задач избыточны.

Нейронные сети могут работать только с числами, поэтому вся входная информация должна быть представлена в числовом формате. Если параметр нечисловой, то обычно его можно отнести к одному из двух классов: категориальная переменная, ординальная переменная. Ординальные переменные отличаются от категориальных только тем, что их можно ранжировать, допустим, плохо – хорошо – отлично. Пример категориальных переменных красный, синий, зелёный и т.д.

Ординальные переменные более близки числовому ряду (т.к. есть порядок), поэтому их можно закодировать числами, допустим, очень медленно (A = 0) – медленно (A = 0.25) – средне (A = 0.5) – быстро (A = 0.75) – очень быстро (A = 1). Однако, нужно учитывать, что это не всегда разумное решение, т.к. тогда мы вводим отношения, которых изначально не было, допустим, при таких A получается, что «быстро» в 3 раза больше, чем «медленно». 0.75 – конкретное число, но ведь «быстро» может быть больше и в 4 раза, чем «медленно», это ведь абстрактные понятия, поэтому чтобы не вводить лишнюю информацию, лучше кодировать эти переменные также как и категориальные.

Примечание. В рассмотренном примере со скоростями значение A выбиралось произвольно человеком, более общий способ это подсчитать количества очень медленных, медленных, средних, быстрых, очень быстрых скоростей, и делить это количество на общую их сумму. Тогда для очень медленной скорости получится (A = количество очень медленных скоростей / общее количество скоростей) и т.д.

Категориальные переменные кодируют через задание численных эквивалентов

классов (двоичное кодирование). Обычно используют два возможных варианта: код n→n и код n→m (m < n).

N→n означает, что ели у нас 5 состояний, которые может принимать переменная, то используется 5 регистров-нейронов для её кодирования. Очень медленно (10000), медленно (01000), средне (00100), быстро (00010), очень быстро (00001).

N→m означает, что количество регистров-нейронов будет меньше, чем количество состояний. Обычно используются два варианта: «температурное» кодирование и классический двоичный код. Рассмотрим «температурное» кодирование. Очень медленно (0000), медленно (1000), средне (1100), быстро (1110), очень быстро (1111). Видно, что состояний 5, а регистров-нейронов уже 4. Второй тип кодирования – это классическое бинарное. У нас 5 состояний, тогда нам нужно m регистров-нейронов, чтобы $2^m \geq 5$, т.е. m = 3. Очень медленно (001), медленно (010), средне (100), быстро (101), очень быстро (110).

В чём особенность каждого типа и когда оно применяется? Кодирование n✉n применяется тогда, когда количество значений каждого типа примерно равно. Т.е. пусть P_i – это количество значений переменной по скорости (A), i = 1..5, где P_1 – количество «очень медленных» скоростей, а P_5 – количество «очень быстрых» скоростей. Если $P_1 \approx P_2 \approx P_3 \approx P_4 \approx P_5$, то можно применять кодирование из n→n, если это условие не выполняется хотя бы для одного P_i, то лучше применить из n→m. Почему? Предположим, что P_1 значительно больше, чем $P_2..P_5$. Тогда, те связи, которые выходят из этого нейрона (первого из пяти) чаще активируются в процессе обучения, тогда как остальные – простаивают. Получается ситуация, что связи есть, а их обучения – нет.

Также стоит упомянуть ещё один момент, что изменения в процессе обучения весов первого слоя зависит от входных значений. И если во входном слое очень много нулей, то обучения просто не будет, процесс растянется. Поэтому во многих случаях лучше кодировать не 0 и 1, а 💼1.

Теперь рассмотрим числовые входы, где значения только числа. В таком случае размер этих входов обычно не меняется. Если, к примеру, у нас есть 10 входных нейронов, из них 7 нейронов получают числовые входы, а не категории, то эти 7 нейронов так и останутся 7-ю нейронами, тогда, как оставшиеся 3, скорее всего, будут преобразованы в q > 3 нейронов по какому-нибудь рассмотренному выше способу кодирования. Однако, бывают и исключения, допустим, круговое кодирование, когда некий нейрон должен получать градусы от 0 до 360 или дни в году. Если мы оставим такой вход, то получится, что между 359 и 0 градусами очень большая разница, а они, тем не менее, стоят рядом на круговой шкале. Чтобы не вводить не существующих отношений можно этот нейрон заменить двумя нейронами, один будет принимать sin(X), другой cos(X). Но обычно числовые входы по

количеству не меняются. Главная проблема числовых входов – различные диапазоны значений. Один нейрон может принимать числа от 0 до 1, а другой от 10^3 до 10^4. Можно такие входы и оставить, но тогда сеть будет вынуждена для больших входов подбирать меньшие значения весов, чтобы скомпенсировать разницу, т.е. появляется дополнительная сложность в обучении, которая может вообще затормозить весь процесс. Поэтому все числовые входы должны быть отнормированы к некоторому стандартному диапазону.

Существуют множество типов нормировок. Тут будут рассмотрены три типа нормировки: линейная, использующая среднеквадратическое отклонение, и сигмоидальная.

Для начала рассмотрим формулы этих нормировок. (6.1) – линейная нормировка, (6.2) – нормировка, использующая среднеквадратическое отклонение, (6.3) – сигмоидальная нормировка.

$$X_{new} = \frac{\tilde{X} - X_{min}}{X_{max} - X_{min}} \cdot (b - a) + a, \quad (6.1)$$

где X_{new} – новое значение входа, X – старое значение входа, X_{min} – минимальное значение из всех значений входа, X_{max} – максимальное значение из всех значений входа, b – максимальное значение нового диапазона в который переводим старое значение, a – минимальное значение нового диапазона.

$$X_{new} = \frac{\tilde{X} - X_{cp}}{\sigma}, \quad (6.2)$$

где X_{cp} – среднеарифметическое значение (6.5) всех значений конкретного входа, σ - среднеквадратическое отклонение (6.6).

$$X_{new} = \frac{1}{1 + e^{-x}}, \quad (6.3)$$

$$x = \frac{X - X_{cp}}{\sigma}, \quad (6.4)$$

$$X_{cp} = \frac{1}{n} * \sum_{i=1}^{n} x_i, \quad (6.5)$$

где n – количество значений конкретного входа, x_i – конкретное значение входа.

$$\sigma = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (x^2)_c - x_{cp}^2}, \quad (6.6)$$

Исходя уже из одних формул можно сделать вывод, что все нормировки используют

какие-нибудь статистики, заранее извлечённые из нужных диапазонов. Линейная использует минимальное и максимальное значения, другие – среднеквадратическое отклонение и т.д. А это означает, что не все нормировки применимы к режиму реального времени. Ведь подразумевается, что обучающая выборка не изменится в размерах, поэтому можно заранее в процессе предобработки извлечь нужные статистики и потом их всегда использовать. Если же обучающая выборка может пополняться в процессе обучения, а это очень вероятно для реальных задач, то нужно либо отказаться от сложных и качественных нормировок (за исключением линейной), либо использовать ускорительные техники для быстрого пересчёта значений и статистик, что не всегда возможно.

Рассмотрим код, демонстрирующий все три типа нормировки, а затем кратко сравним их.

```
% Двести значений
x=5:0.005:6;
% получаем некий числовой ряд с помощью тригонометрических
% функций
y1=exp(x)+25*sin(100*x)+43*randn(1,201)+36*cos(10*x);
% делаем в этом ряде два резких выброса, т.е. нетипичных значения
y1(50)=y1(50)+986;
y1(150)=y1(150)+1286;
% высчитываем статистики
% среднее арифметическое
mean1 = sum(y1)/length(y1);
% среднеквадратическое отклонение
std1 =sqrt((norm(y1)^2)/length(y1)-mean1);
% альфу
alpha = (y1 - mean1)/std1;
% минимум и максимум
min1=min(y1);
max1=max(y1);
% задаём новый диапазон для линейной нормировки
min2=-1; max2=1;
% линейно нормируем
y2=((y1-min1) / (max1-min1)) * (max2-min2)+min2;
% нормируем с помощью среднеквадратического отклонения
y3=(y1-mean1)/std1;
% сигмоидальная нормировка
y4 = (1-exp(-alpha))./(1+exp(-alpha));
% Выводим 4 графика, при желании их можно
% увеличить с помощью zoom+
figure(4)
subplot (221), plot(y1(1:200)), xlabel('Y1');
subplot (222), plot(y2(1:200)), xlabel('Y2. Linear normalization');
subplot (223), plot(y3(1:200)), xlabel('Y3. Zscore normalization');
subplot (224), plot(y4(1:200)), xlabel('Y4. Sigmoidal normalization');
```

Рисунок 6.1 – Исходные значения (Y1) и три типа нормировки: Y2 – линейная, Y3 – с использованием среднеквадратического отклонения, Y4 - сигмоидальная

Исходя из рисунка 6.1 можно сделать следующие выводы: линейная нормировка совершенно не справляется с размазыванием значений между положительными и отрицательными (а это очень важно, чтобы отрицательных и положительных значений было примерно поровну в отнормированных данных. Это ускоряет процесс обучения), зато мы сами можем задать интересующий нас диапазон; нормировка с использованием среднеквадратического отклонения справляется с размазыванием лучше, однако нет гарантии интервала от -1 до +1; наконец сигмоидальная нормировка лучше остальных размазывает значения между положительными и отрицательными величинами, а также неплохо борется с выбросами данных, помещая их в единичный диапазон.

Конечный выбор нормировки остаётся всегда за разработчиком.

Рассмотрим первую задачу о «Титанике».

Рисунок 6.2 – Крушение «Титаника»

Прежде чем решать задачу необходимо провести предварительный анализ входных данных путём ответов на ряд основных вопросов. Тогда понимание задачи значительно вырастет. При практическом решении задачи ряд пунктов предварительного анализа могут быть пересмотрены.

Вопрос 1. Если задача классификации, то сколько есть классов?

Ответ очевиден, в данной задаче 2 класса: выжил–не выжил.

Вопрос 2. Сколькими паттернами представлен каждый класс (степень равномерной представленности классов)?

Для ответа на этот вопрос нужно провести небольшой анализ. Удобно его провести либо силами Excel (фильтры, функции), либо написать небольшой код в Matlab. В данной лабораторной работе будет использован Excel. В файле «DataForTitanic.xlsx» можно в свободной ячейки написать «=СЧЁТЕСЛИ(H2:H1310; 1)» или «=СЧЁТЕСЛИ(H2:1310; 0)», что вернёт количество классов выжил и не выжил, соответственно. Получится первый класс (1) представлен 500 паттернами, второй класс (0) представлен 809 паттернами, т.е. классы представлены неравномерно (погибших больше, чем выживших).

Вопрос 3. Сколькими элементами описывается паттерн каждого класса?

В данной задаче каждый класс до предобработки описывается вектором из семи элементов.

Вопрос 4. Какой тип каждой переменной-элемента (категориальная, ординальная, числовая) и её возможный числовой диапазон?

1-ая переменная (Class) – ординальная переменная: 1 – первый класс, 2 – второй класс, 3 – третий класс.

2-ая переменная (Sex) – категориальная переменная: 0 – женщина, 1 – мужчина.

3-ая переменная (Age) – обычная числовая из диапазона [0.17..80].

4-ая переменная (Sibsp) – обычная числовая из диапазона [0..8].

5-ая переменная (Parcn) – обычная числовая из диапазона [0..9].

6-ая переменная (Fare) – обычная числовая из диапазона [0..512.33].

7-ая переменная (Embarked) – категориальная: 0 – Cherbourg, 1 – Queenstown, 2 – Southampton.

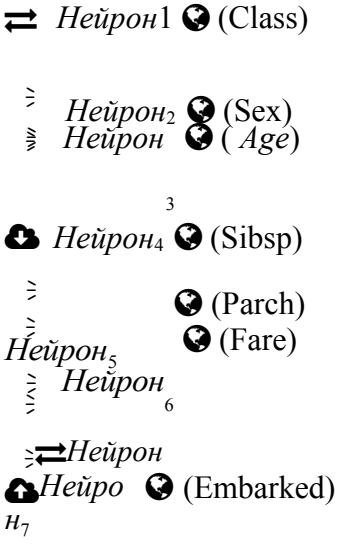
Ответ: 0 – не выжил, 1 – выжил.

Вопрос 5. Требуется ли изменять размерность входного вектора?

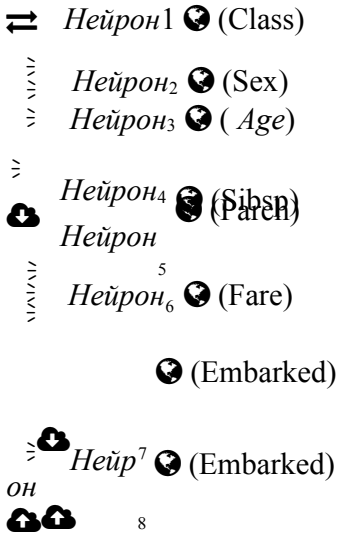
Есть категориальные и ординальные переменные, поэтому размерность придётся менять.

Переменную Class мы можем как перекодировать, так и нет, т.к. она ординальная и если мы оставим значения 1, 2, 3, то мы не введём не существующих отношений. Переменную Sex тоже менять не будем.

Перекодируем категориальную переменную Embarked. Для того, чтобы узнать, как это сделать, необходимо сначала проанализировать степень равномерной представленности значений этого компонента входного вектора. Анализ осуществлялся с помощью Excel («=СЧЁТЕСЛИ(B2:B2202; 0)», «=СЧЁТЕСЛИ(G2:G1310; 0)», «=СЧЁТЕСЛИ(G2:G1310; 1)», «=СЧЁТЕСЛИ(G2:G1310; 2)»). Получились следующие значения: 0 – 272 штук, 1 – 123 штук, 2 – 914 штук. Пусть P_i – количество соответствующих значений для данного значения-переменной входного вектора, где i = 0..2, тогда получается, что $P_2 >> P_0 > P_1$, а, следовательно, нужно использовать классическое бинарное кодирование. $2^m \geq 3$ откуда m = 2. Т.е. вместо седьмого входного нейрона необходимо сделать два входных нейрона. Был вход такой:



, а теперь стал такой:



.

Ответ на пятый вопрос будет не полным, если не определиться с кодировкой входных значений. Если мы выберем кодировку 0/1, тогда Embarked закодируется, допустим, так 0 – (1, 1), 1 – (1, 0), 2 – (0, 1) (Учитывая расположение нейронов, далее везде в лабораторной работе предполагается, что 0 – (1, 1) это на самом деле 0 – $(1, 1)^T$). Это приемлемо, но в целом не нужно забывать, что обучение первого слоя нейронной сети зависит от входных значений, и если тот или иной нейрон будет иметь слишком много нулей в качестве своих входных значений, то обучение его связей с первым скрытым слоем

затормозится. Процесс обучения ускорится при использовании кодировки 💼1.

Приходим к следующему результату: Embarked (0 → (-1, 1), 1 → (1, -1), 2 → (1, 1)).

Количество портов посадки с индексом 0 и 1 меньше отличается друг от друга, чем с индексом 2, поэтому им назначаем симметричные коды.

Вопрос 6. Как кодировать класс, который нужно определить?

Мы уже определились, что у нас два класса, которые нужно классифицировать. Это простейший случай. Для двух классов достаточно одного нейрона в выходном слое с сигмоидальной или линейной функцией активации. Соответственно, для одного класса нейрон должен активироваться в зоне «плюс», для другого класса – в зоне «минус». Однако, более правильно сделать два нейрона. В задачах классификации обычно количество выходных нейронов соответствует количеству классов, которые нужно распознать. Такой подход упрощает процесс обучения, т.к. веса каждого такого нейрона решают свою локальную задачу классификации (отделения своего класса от всех остальных), а не пытаются решить две прямо противоположные задачи для разных классов (активировать нейрон в симметричных зонах). Исходя из этого, имеет смысл перекодировать класс Survived: 0 → (1, -1), 1 → (-1, 1).

Вопрос 7. К какому диапазону приводить остальные значения?

Мы перекодировали вход Embarked и выходной класс. Есть ещё одна категориальная переменная Sex, её также необходимо перекодировать в 💼1: 0 → (-1), 1 → (1). Теперь остались обычные числовые значения и один ординарный вход. Их всех необходимо как можно более равномернее «размазать» между положительными и отрицательными значениями. Можно либо писать свой код для нормировки того или иного входа нейронной сети, как в листинге выше, а можно воспользоваться стандартными средствами Matlab, прежде всего, **mapminmax()** и **mapstd()**. **Mapminmax()** реализует нормировку, которая соответствует формуле (6.1), **mapstd()** – формуле (6.2). Можно также воспользоваться **fixunknowns()** для работы с пропущенными значениями. Более подробное применение этих функций приводится в листинге ниже.

Вопрос 8. Какая пропорция берётся для обучающей и тестовой выборки?

В тех задачах, где выборка заранее разбита этот вопрос не имеет смысла, если же дана одна искомая неразбитая выборка, то пользователь сам должен решать вопрос о разбитии. В данной задаче мы возьмём 80% (1047 паттернов) для обучающей выборки и 20% (262) для тестовой. Также популярны разбития 70%/30%. В целом, чем сложнее задача, тем больше паттернов должно идти в обучающую выборку и тем грамотнее они должны отбираться.

Вопрос 9. Как формировать обучающую и тестовую выборки?

Этот вопрос тесно связан с ответом на второй вопрос. Если бы в нашей выборке каждый класс был равномерно представлен, то достаточно было бы просто случайным

образом отобрать нужное количество паттернов для теста и для обучения. Но тут ситуация такая, что выжило 500, а погибло 809 человек, и если мы при таком раскладе начнём отбирать случайно, то может получиться, что, допустим, в обучающей выборке будет слишком мало паттернов класса «Выжил», что неправильно. Тогда перед нами стоит задача отобрать в тестовую и обучающую выборки количество паттернов каждого класса, учитывая общую пропорцию классов «Выжил» и «Погиб». Найдём сначала, сколько процентов занимает класс «Выжил» от «Погиб».

$\begin{cases} 809 \ 100\% \\ 500 \ x \end{cases} \Rightarrow x = \frac{500 * 100}{809} \approx 62\%$, теперь зная это, найдём количество паттернов классов «Выжил» и «Погиб» для обучающей выборки:

$\begin{cases} t_1 + t_2 = 1047 \\ t_2 = 0.62 * t_1 \end{cases} \Rightarrow \begin{cases} t_1 = 646 \\ t_2 = 401 \end{cases}$, где t_1 – количество паттернов класса «Погиб» для обучающей выборки, t_2 – количество паттернов класса «Выжил» для обучающей выборки, учитывая общее соотношение этих классов в первоначальной выборке.

Решим аналогичную систему для тестовой выборки:

$\begin{cases} z_1 + z_2 = 262 \\ z_2 = 0.62 * z_1 \end{cases} \Rightarrow \begin{cases} z_1 = 162 \\ z_2 = 100 \end{cases}$, где z_1 и z_2 – количество паттернов класса «Погиб» и «Выжил» для тестовой выборке соответственно. Однако, не нужно забывать, что решая эти две системы мы округляли проценты до 0.62, поэтому возможны небольшие неточности. Проверим результат. Общее количество выживших людей $z_2 + t_2 = 100 + 401 = 501$, общее количество погибших $z_1 + t_1 = 162 + 646 = 808$. Ответ же на второй вопрос при анализе выборки показал, что количество выживших людей составило 500, а погибших 809 людей, т.е. есть небольшая неточность, хотя по суммам всё сходится (808+501 = 809+500). Получается отличие в два человека. Вручную подкорректировав результат, отняв от одного уравнения и прибавив одного человека к другому уравнению в рамках одной из двух систем, окончательно получим $\begin{cases} t_1 = 647 \\ t_2 = 400 \end{cases} \begin{cases} z_1 = 162 \\ z_2 = 100 \end{cases}$.

Теперь мы точно знаем, сколько паттернов и какого типа отбирать в обучающую выборку и в тестовую. После отбора выборки нужно перемешать.

Вопрос 10. Сколько слоёв сети требуются для решения данной задачи?

Требуется три слоя: входной, скрытый, выходной.

Вопрос 11. Сколько нейронов в каждом слое?

Как уже неоднократно отмечалось на этапе предварительного анализа точно

ответить на этот вопрос невозможно, нужна практика.

Для входного слоя 8 нейрона (после расширения), для выходного – 2 нейрона. Для скрытого q 8 4 нейрона (Лабораторная работа №5). Как было замечено выше в этой задаче главная проблема – противоречие в данных, поэтому повысить качество через два скрытых слоя или серьёзное увеличение нейронов в скрытом слое не удастся всё равно.

Вопрос 12. Какие связи между слоями?

Сеть полносвязанная: каждый нейрон слоя связан со всеми нейронами последующего слоя. В большинстве задач, однако, полносвязанность почти всегда избыточное условие, которое накладывается в силу того, что мало что известно о внутренней структуре «черного ящика» (ИНС). Поэтому почти всегда удаётся улучшить результат, вводя дополнительную информацию в структуру сети, как это обычно происходит в сверточных сетях.

Вопрос 13. Какие функции активации?

В скрытом слое нейроны с гиперболическим тангенсом, значения которого изменяются от -1 до +1 (т.к. вход закодирован в 1), нейроны выходного слоя могут быть также такими же или линейными. Если выбирается гиперболический тангенс, то можно ставить функции активации на выходном слое вида A*tanh(B*x), где A и B – специальные константы. Это делается потому, что tanh(x) имеет 1 в зонах насыщения, этих значений тяжело достичь, а при константе A функция вытягивается вдоль оси OY и 1 оказываются в линейной части.

Вопрос 14. Какая функция ошибки?

Среднеквадратическая ошибка (MSE).

Приведённый анализ должен помочь пользователю лучше понять задачу. Ряд других вопросов лучше решать на практике: какой алгоритм использовать при обучении, какие результаты и как отображать, и т.д.

```
% Загружаем выборку
initSet = xlsread('C:\Users\Computer Grand\Desktop\МАТЕРИАЛЫ\К лабе 6\DataForTitanic.xls', 1, 'A:H');
% Транспонируем выборку
initSet = initSet';
% Находим индексы тех элементов, где в последней строчке стоит 0,
% т.е., где человек утонул
ind_zero = find(initSet(8, :)==0);
% находим индексы 1
ind_one = find(initSet(8, :)==1);
% Формируем из них обучающий вектор с нужным балансом 0 и 1
ind_for_training = [ind_zero(1:647) ind_one(1:400)];
% Аналогично для тестового вектора
ind_for_test = [ind_zero(648:809) ind_one(401:500)];
```

```
% Но теперь они не перемешаны, индексы нужно перемешать
IndForTest = ind_for_test(randperm(length(ind_for_test)));
IndForTraining = ind_for_training(randperm(length(ind_for_training)));
% Очищаем оперативную память от громоздких матричных данных,
% которые уже не нужны
clear ind_for_test, clear ind_for_training, clear ind_one, clear ind_zero;
% Создаём заготовку для всей выборке, куда будут помещаться уже
% перемешанные данные
initSetPerm = [repmat([0 0 0 0 0 0 0 0]', 1, length(initSet))];
% Заполняем нашу заготовку, теперь она и есть наша выборка
initSetPerm(:, 1:1047) = initSet(:, IndForTraining(:));
initSetPerm(:, 1048:1309) = initSet(:, IndForTest(:));
clear IndForTest, clear IndForTraining, clear initSet;
% Создаём ещё одну заготовку, но уже для выборки с изменённым размером
% входов теперь станет 8, а не 7 и выход с одного нейрона расширится до 2-х
ExtendedInitSet = [repmat([0 0 0 0 0 0 0 0 0 0]', 1, length(initSetPerm))];
% осуществляем все необходимые замены
proba = find(initSetPerm(8, :)==0);
for i=1:length(proba)
    ExtendedInitSet(9:10, proba(i)) = [1 -1]';
end
proba = find(initSetPerm(8, :)==1);
for i=1:length(proba)
    ExtendedInitSet(9:10, proba(i)) = [-1 1]';
end
proba = find(initSetPerm(7, :)==0);
for i=1:length(proba)
    ExtendedInitSet(7:8, proba(i)) = [-1 1]';
end
proba = find(initSetPerm(7, :)==1);
for i=1:length(proba)
    ExtendedInitSet(7:8, proba(i)) = [1 -1]';
end
proba = find(initSetPerm(7, :)==2);
for i=1:length(proba)
    ExtendedInitSet(7:8, proba(i)) = [1 1]';
end
proba = find(initSetPerm(2, :)==0);
for i=1:length(proba)
    ExtendedInitSet(2, proba(i)) = -1;
end
proba = find(initSetPerm(2, :)==1);
for i=1:length(proba)
    ExtendedInitSet(2, proba(i)) = 1;
end
% копируем данные, которые не требуют на этом этапе нормировки,
% они будут отнормированы на этапе обучения сети
% ExtendedInitSet(1, :) = initSetPerm(1, :);
% ExtendedInitSet(3, :) = initSetPerm(3, :);
% ExtendedInitSet(4, :) = initSetPerm(4, :);
% ExtendedInitSet(5, :) = initSetPerm(5, :);
% ExtendedInitSet(6, :) = initSetPerm(6, :);
```

```
% однако их можно и в принудительном порядке отнормировать, как
% показано ниже
ExtendedInitSet(1, :) = mapminmax(initSetPerm(1, :));
ExtendedInitSet(3, :) = mapstd(initSetPerm(3, :));
ExtendedInitSet(4, :) = mapstd(initSetPerm(4, :));
ExtendedInitSet(5, :) = mapstd(initSetPerm(5, :));
ExtendedInitSet(6, :) = mapstd(initSetPerm(6, :));

clear i, clear initSetPerm, clear proba;
% Бьём выборку на входы и метки
input_data = ExtendedInitSet(1:8, :);
output_data = ExtendedInitSet(9:10, :);

% Создаём сеть для решения задачи, использовали 8 нейронов в скрытом слое,
% можно попробовать сеть с двумя скрытыми слоями, но результат не улучшится
net = newff(input_data, output_data, [8], {'tansig', 'tansig'}, 'trainlm', 'learngd', 'mse', {}, {},
'divideind');
% Отмечаем диапазон для обучения и валидационной выборки (теста)
net.divideParam.trainInd = 1:1047;
net.divideParam.valInd = 1048:1309;
net.trainParam.max_fail = 2;
% Устанавливаем тип нормировки
% net.inputs{1}.processFcns = {'mapstd'};
net = init(net);
[net, tr] = train(net, input_data, output_data);
% выделяем тест для проверки
testSet = input_data(1:8, 1048:1309);
testT = output_data(1:2, 1048:1309);
Y = sim(net, testSet);
col = 0;
for i = 1:length(testSet)
% сравниваем ответ сети с идеальным
    if norm(testT(i)-Y(i)) < 0.3
        col = col + 1;
    end
end
% выводим процент правильных ответов на тесте
percent = 100 * col / length(testT)
```

В результате работы этого кода, можно добиться точности распознавания до 65%, что довольно мало. Какой вывод следует из решения данной задачи? Если нейронная сеть получает противоречивые данные или данные, где противоречие находится не в явном виде, то никакая архитектура нейронной сети не сможет качественно решить задачу. Что тогда можно сделать? Можно, во-первых, постараться изменить обучающую выборку, что и было сделано. Во-вторых, нужно на обучающее множество отвести как можно больше паттернов. В-третьих, использовать сеть с несколькими скрытыми слоями. В-четвёртых, смягчить критерий правильного ответа. Так в коде правильный ответ считается тот, чей вектор отличается менее, чем на 0.3 от идеального. Но последняя техника возможна не

всегда. Альтернатива всем этим действиям – выделить нейронной сети много настраиваемых параметров и превратить её в память, но тогда потеряются все свойства обобщения.

Рассмотрим вторую задачу о грибах.

Рисунок 6.3 – Гриб из семейства Agaricus

Также проведём предварительный анализ данной задачи.

Вопрос 1. Если задача классификации, то сколько есть классов?

Два класса: съедобен (edible), ядовит (poisons).

Вопрос 2. Сколькими паттернами представлен каждый класс (степень равномерной представленности классов)?

Съедобен – 4208 (51.8%), ядовит – 3916 (48.2%). Фактически это равномерная представленность классов.

Вопрос 3. Сколькими элементами описывается паттерн каждого класса?

Каждый паттерн описывается вектором из 21 элементов-параметров.

Вопрос 4. Какой тип каждой переменной-элемента (категориальная, ординальная, числовая) и её возможный числовой диапазон?

Все переменные категориальные.

Вопрос 5. Требуется ли изменять размерность входного вектора?

Прежде, чем отвечать на этот вопрос в учебных целях проведём анализ входных переменных. На основе этого анализа сделаем ряд предварительных выводов. Результаты анализа представлены в таблице 6.2.

Таблица 6.2 – Анализ значений входных переменных

№	Входная переменная	Значения входной переменной (количество каждого значения)	Вывод
1	Cap-shape	b(452), c(4), x(3656), f(3152), k(828), s(32)	Неравномерное распределение

			значений, кодирование бинарное
2	Cap-surface	f(2320), g(4), y(3244), s(2556)	Неравномерное распределение значений, кодирование бинарное
3	Cap-colour	n(2284), b(168), c(44), g(1840), r(16), p(144), u(16), e(1500), w(1040), y(1072)	Неравномерное распределение значений, кодирование бинарное
4	Bruises?	t(3376), f(4748)	Равномерное распределение, но всего два значения, поэтому остаётся один нейрон
5	Odor	a(400), l(400), c(192), y(576), f(2160), m(36), n(3528), p(256), s(576)	Неравномерное распределение значений, кодирование бинарное
6	Gill-attachment	a(210), d(0), f(7914), n(0)	Неравномерное распределение значений, кодирование бинарное, два значения в реальной выборке отсутствуют, поэтому значений не 4, а 2
7	Gill-spacing	c(6812), w(1312), d(0)	Неравномерное распределение значений, кодирование бинарное, значений не 3, а 2

8	Gill-size	b(5612), n(2512)	Неравномерное распределение значений, всего два значения, поэтому остаётся один нейрон
9	Gill-colour	k(408), n(1048), b(1728), h(732), g(752), r(24), o(64), p(1492), u(492), e(96), w(1202), y(86)	Неравномерное распределение значений, кодирование бинарное
10	Stalk-shape	e(3516), t(4608)	Равномерное распределение

			значений, но их всего два, поэтому остаётся один нейрон
11	Stalk-surface-above-ring	f(552), y(24), k(2372), s(5176)	Неравномерное распределение значений, кодирование бинарное
12	Stalk-surface-below-ring	f(600), y(284), k(2304), s(4936)	Неравномерное распределение значений, кодирование бинарное
13	Stalk-colour-above-ring	n(448), b(432), c(36), g(576), o(192), p(1872), e(96), w(4464), y(8)	Неравномерное распределение значений, кодирование бинарное
14	Stalk-colour-below-ring	n(512), b(432), c(36), g(576), o(192), p(1872), e(96), w(4384), y(24)	Неравномерное распределение значений, кодирование бинарное
15	Veil-type	p(8124), u(0)	Данный вход по факту есть константа, достаточно одного нейрона
16	Veil-colour	n(96), o(96), w(7924), y(8)	Неравномерное распределение значений, кодирование бинарное
17	Ring-number	n(36), o(7488), t(600)	Неравномерное распределение значений, кодирование бинарное
18	Ring-type	c(0), e(2776), f(48), l(1296), n(36), p(3968), s(0), z(0)	Неравномерное распределение значений,

			кодирование бинарное, значений не 8, а 5
19	Spore-print-colour	k(1872), n(1968), b(48), h(1632), r(72), o(48), u(48), w(2388), y(48)	Неравномерное распределение значений, кодирование бинарное
20	Population	a(384), c(340), n(400), s(1248), v(4040), y(1712)	Неравномерное распределение

			значений, кодирование бинарное
21	Habitat	g(2148), l(832), m(292), p(1144), u(368), w(192), d(3148)	Неравномерное распределение значений, кодирование бинарное

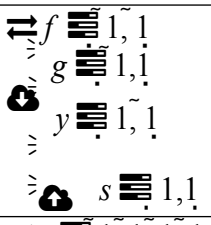
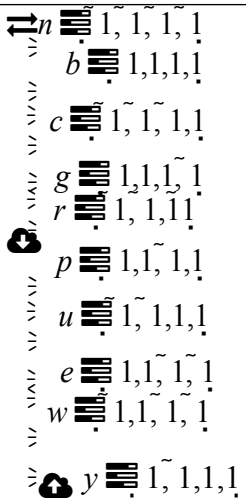
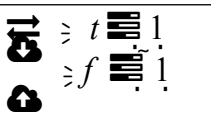
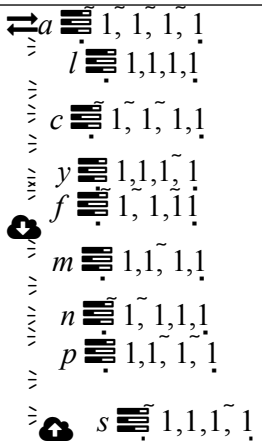
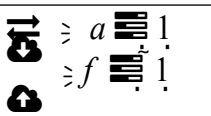
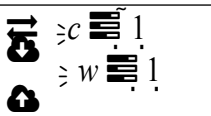
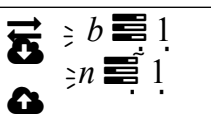
Как видно из таблицы анализ помог выявить, что часть значений у некоторых переменных вообще не встречается, хотя они описаны в условии задачи, а также почти везде был выбран самый компактный тип кодирования категориальных переменных: классическое бинарное кодирование.

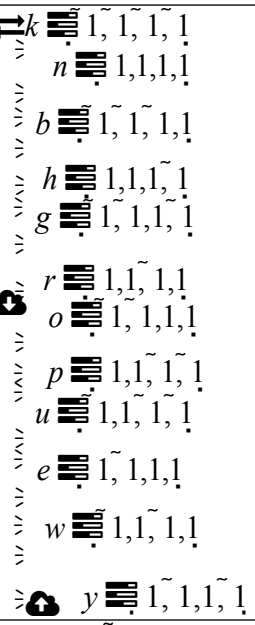
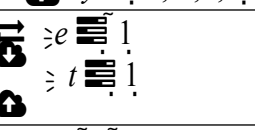
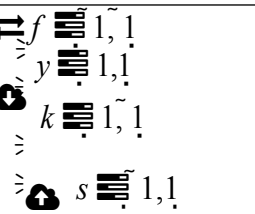
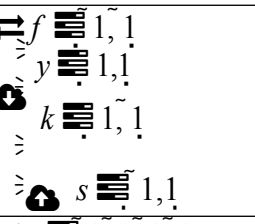
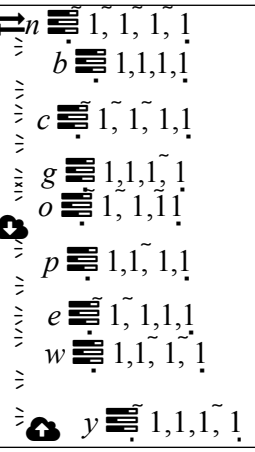
В отличие от первой задачи тут размерность входа достаточно большая (21 переменная), поэтому не нужно опасаться, что входной вектор будет близок к нулевому во многих случаях, он будет просто разряженным. Тут уместно оставить кодировку 0/1, но можно использовать и ☛1. Выберем ☛1 из-за скорости обучения.

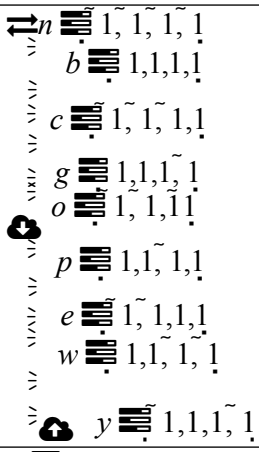
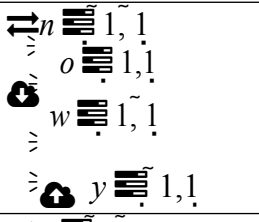
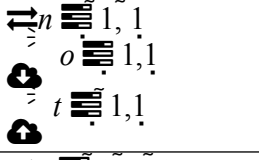
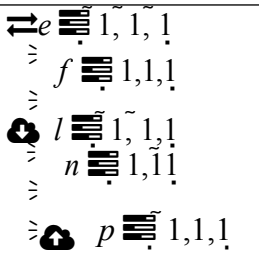
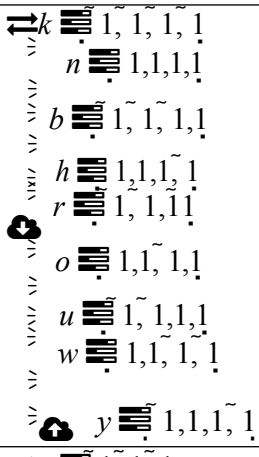
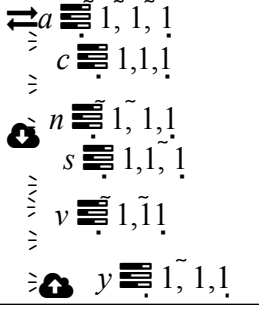
Прежде, чем кодировать входные значения необходимо учесть, что может быть такая ситуация, что, допустим, некоторая переменная принимает 5 значений, тогда для того, чтобы закодировать все значения бинарной кодировкой нужно 3 нейрона, но три нейрона могут максимум закодировать 8 значений (2^3). Получается, что у нас есть 3 лишних кода, тогда встаёт вопрос о том, какие значения бинарной кодировки использовать для кодирования? Лучше использовать такие значения из возможных 8, чтобы сохранялась симметрия между 0 и 1 или ☛1. Т.е., если мы закодируем просто 5 значений кодом без учёта симметрии, то получим что-то вроде следующего: 0 → (000), 1 → (001), 2 → (010), 3 → (011), 4 → (100). Видно, что нулей в таком коде больше единиц, а с учётом симметрии будет так: 0 → (000), 1 → (111), 2 → (001), 3 → (110), 4 → (100). Учтём этот момент при кодировке. Результат кодировки представлен в таблице 6.3.

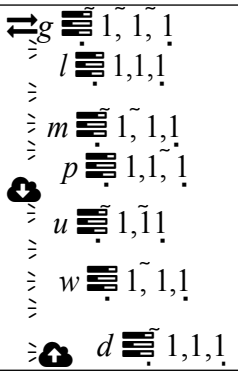
Таблица 6.3 – Кодировка входных значений

№	Переменная	Кодировка значений
1	Cap-shape	⇄*b* 1,1,1 *c* 1,1,1 *x* 1,1,1 *f* 1,1,1 *k* 1,1,1 *s* 1,1,1

2	Cap-surface	
3	Cap-colour	
4	Bruises?	
5	Odor	
6	Gill-attachment	
7	Gill-spacing	
8	Gill-size	

9	Gill-colour	
10	Stalk-shape	
11	Stalk-surface-above-ring	
12	Stalk-surface-below-ring	
13	Stalk-colour-above-ring	

14	Stalk-colour-below-ring	
15	Veil-type	
16	Veil-colour	
17	Ring-number	
18	Ring-type	
19	Spore-print-colour	
20	Population	

21	Habitat	

Таким образом, размерность входного вектора сильно изменилась от первоначальных 21 переменных к 52. И теперь возвращаемся к вопросу 5. Мы видим, что входной вектор сильно вырос. По возможности, нужно стараться не расширять вектор входных значений, а сужать его, т.к. существует такая проблема, как «проклятие размерности», когда с ростом входа количество требуемых данных для обучения растёт экспоненциально. Конечно, в данном случае рост входа компенсируется тем, что добавленные нейроны принимают только два значения. Также учтём, что все входные переменные категориальные, а это означает, что в этой задаче мы можем и не расширять входной вектор.

Вопрос 6. Как кодировать класс, который нужно определить?

Так как мы приняли решение не расширять входной вектор, то выход можно просто отнормировать к диапазону -1..+1 вместо 0 и 1.

Если всё же расширять вход, то учитывая больший размер входного паттерна, ещё более необходимо каждый класс представить своим нейроном, а не использовать один нейрон на два класса: e → (-1, 1), p → (1, -1).

Вопрос 7. К какому диапазону приводить остальные значения?

Все 21 входных значений можно отнормировать к диапазону -1..+1 для увеличения скорости обучения.

Вопрос 8. Какая пропорция берётся для обучающей и тестовой выборки?

90% - обучающая выборка, 10% - тестовая выборка, 10% - валидационная выборка.

Вопрос 9. Как формировать обучающую и тестовую выборки?

Классы в искомой выборке представлены равномерно (51.8% и 48.2%, что почти одно и то же). Тогда можно случайным образом (средствами Matlab) сформировать обучающую и тестовую выборки соответствующих размеров, не заботясь сколько конкретно паттернов каждого типа будет в получившихся выборках.

Вопрос 10*. Сколько слоёв сети требуются для решения данной задачи?*

Требуется три слоя: входной, скрытый, выходной.

Вопрос 11. Сколько нейронов в каждом слое?

Во входном слое 21 нейрон, в выходном 1 нейрона, в скрытом можно взять 10 нейронов.

Вопрос 12. Какие связи между слоями?

Сеть полносвязанная.

Вопрос 13. Какие функции активации?

На скрытом слое – гиперболический тангенс, на выходном слое – аналогично или линейная функция активации.

Вопрос 14. Какая функция ошибки?

Среднеквадратическая ошибка (MSE).

Листинг программы приведён ниже. Использовалась выборка «DataForMushrooms_num.xls», в которой символы были заменены числами от 0 до количества переменных в системе из таблицы 6.3 для соответствующей строки.

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\МАТЕРИАЛЫ\К лабе 6\DataForMushrooms_num.xls', 1, 'B:W');
 initSet = initSet';
% Приводим выходы сети к диапазону -1..+1, ниже в коде есть альтернатива
initSet(22, :) = mapminmax(initSet(22, :));
input_data = initSet(1:21,:);
output_data = initSet(22,:);
net = newff(input_data, output_data, [10], {'tansig', 'tansig'}, 'trainlm', 'learngd', 'mse', {}, {}, 'dividerand');
% Ограничиваем обучение по количеству эпох
net.trainParam.epochs = 100;
net.trainParam.max_fail = 2;
% Разбиваем выборку
net.divideParam.trainRatio = 0.8;
net.divideParam.testRatio = 0.1;
 net.divideParam.valRatio = 0.1;
% Устанавливаем номировку одну для всех входов
net.inputs{1}.processFcns = {'mapminmax'};
% Можно альтернативно отнормировать выходы сети
% net.outputs{2}.processFcns = {'mapminmax'};
% инициализируем сеть нашими параметрами
net = init(net);
[net, tr] = train(net, input_data, output_data);
testSet = input_data(:, tr.testInd(:));
testT = output_data(:, tr.testInd(:));
Y = sim(net, testSet);
 col = 0;
for i = 1:length(testSet)
    if norm(testT(i)-Y(i)) < 0.1
       col = col + 1;
    end
end
% выводим процент правильных ответов на тесте
```

percent = 100 * col / length(testT)

Как видно из рисунка 6.4 обучение осуществляется почти идеально. Если мы не отнормируем выходные значения, оставив их в диапазоне 0 и 1, а входы переведём в диапазон -1..+1, то процесс обучения сильно затормозится, как показано на рисунке 6.5. Нам потребуется не 12, а 100 эпох.

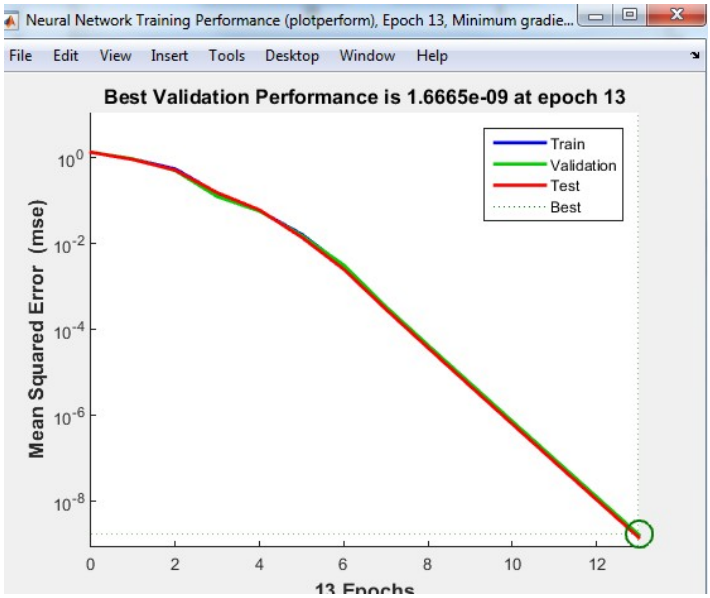


Рисунок 6.4 – Обучение сети со всеми отнормированными данными к диапазону -1..+1

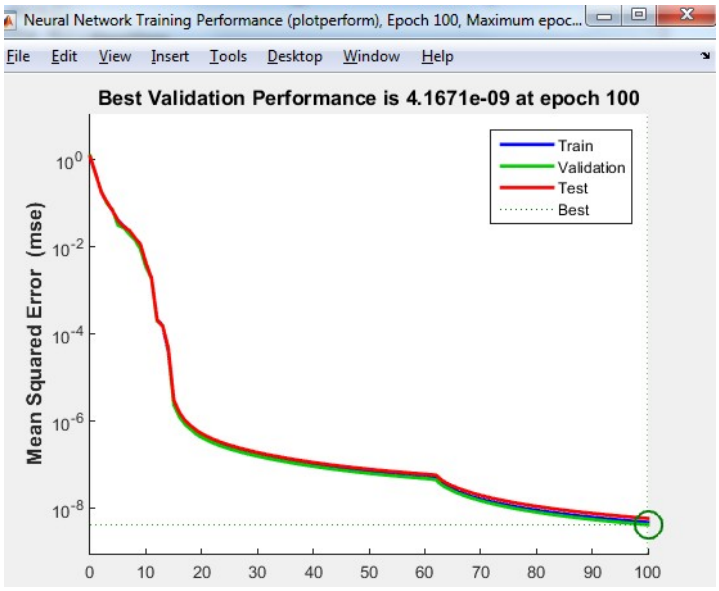


Рисунок 6.5 – Обучение сети, где отнормированы только входные данные

На тестовом множестве мы распознаём все паттерны. Студент может попробовать самостоятельно решить данную задачу, изменив ошибку обучения с MSE на CrossEntropy, тогда количество итераций при использовании алгоритма trainrp (RPROP) составит в среднем 6, а не 12.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

В индивидуальном варианте (таблица 6.4) дано условие задачи на классификацию. Приемлемая точность на тестовой выборке везде отмечена как 70%, что в целом не много. Достижение более большой точности распознавания оценивается в дополнительных баллах.

Таблица 6.4 – Варианты заданий

№	Условие задачи
1	Решить задачу о грибах с расширением входного вектора и выходного так, как это описано в разборе второй задачи. Сравнить результаты с обучением на рисунке 6.4.
2	В США в Аризоне существует племя индейцев Пима (https://ru.wikipedia.org/wiki/Пима). Особенность этого племени в том, что у его представителей самый высокий в мире процент заболеваемости сахарным диабетом 2-го типа. Среди женщин этого племени было проведено исследование (спонсируемое Всемирной Организацией Здравоохранения; речь идёт об исследовании 1988 г., т.к. были и другие). Одним из результатов данного исследование было то, что была составлена выборка, где отмечали наличие или отсутствие диабета в зависимости от разных факторов (адрес выборки: https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes/). Выборка состоит из 768 записей. Всего 9 столбцов, 8 из них – показатели, а 1 – флаг (наличие или отсутствие диабета). Классы представлены не равномерно: женщины у которых нет диабета 500 штук, у которых есть – 268 штук. Первый столбец – количество беременностей, второй – концентрация глюкозы в плазме в течении двух часов при использовании орального теста на концентрацию глюкозы, третий – кровяное давление, четвёртый – толщина сгибающей трёхглавой мышцы (Triceps skin fold thickness), пятый - 2-Hour serum insulin, шестой – индекс массы тела, седьмой - Diabetes pedigree function, восьмой – возраст, девятый – обозначение класса (0 – нет диабета, 1 – есть диабет). Для нейронной сети (многослойный персептрон) достигали результата в 76.4% распознаваний.
3	Буквы. Изначально были английские заглавные буквы, каждая буква была записана 20 различными шрифтами и к каждой такой записи применялись ещё случайные

	искажения. Всего было 20000 записей английских заглавных букв, все они располагались в рамке определённого размера. Потом через эту рамку проводилось ряд прямых, подсчитывалось количество пересечений этих прямых с буквой и результат записывался в вектор, также в вектор записывалась базовая информация о размерах рамки, количестве пикселей и т.д. Таким образом, удалось перейти от пиксельного представления буквы к её признаковому представлению (осуществить сжатие) в виде вектора размером в 16 позиций. Т.е. каждая английская буква определённого шрифта и с определённым искажением описывается таким вектором. Требуется по входному вектору классифицировать букву. Подробную информацию о сети и выборке можно получить из статьи Hussein Salim Qasim “Letter Recognition Data Using Neural Network” (https://www.researchgate.net/publication/289129248_Letter_Recognition_Data_Using_Ne ural_Network) или найти информацию в интернете по запросу «letter dataset» . На примере этой задачи видно, как сильно изменились нейросетевые технологии классификации. Раньше отдельным этапом осуществляли предобработку входных данных, причём часто этот этап был весьма нетривиальным. Теперь стандарт – подача на вход сети пиксельного изображения. Похожая задача рассматривается в варианте 10.
4	Классификация типов узлов скрещивания в цепочках ДНК (http://www.cs.toronto.edu/~delve/data/splice/spliceDetail.html). В данной задаче рассматриваются цепочки ДНК, состоящих из последовательностей аминокислот (A, C, T и G) длиной 60 каждая. В каждом примере середина цепочки является кандидатом на узел скрещивания (рассматриваются 3 класса - 0 - переход типа "интрон-эксон", 0.5 - незначимый переход, 1 - переход "эксон-интрон"). Аминокислоты закодированы целыми числами A - 0, C - 1, T - 2, G - 3. Требуется построить нейросетевую систему классификации. Данные (3175 примеров) содержатся в файле. 2500 или 2000 первых примеров используются для обучения, остальные - для тестирования. Требуется построить нейросетевую классификационную систему с наименьшей ошибкой тестирования. 60 входных значений обычно переводятся в двоичный код, и вход расширяется либо до 180, либо до 240 значений. Возможные распределения для тестовой и обучающей выборки: Обучение, 1–(464, 23.20%), 2–(485, 24.25%), 3–(1051, 52.55%); Тест, 1–(303, 25.55%), 2–(280, 23.61%), 3–(603, 50.84%).
5	Задача об оценки эффективности преподавания учителя на основе данных обучения 3 регулярных семестров и двух летних в статистическом отделении университета Висконсин-Мэдисон. Выходной класс – оценка (1 – низкая, 2 – средняя, 3 - высокая). Входной вектор: Native (английский родной – 1, не родной - 2), Instructor (25 категорий для инструктора курса), Course (26 категорий), Semester (1 – летний, 2 - регулярный), Size (числовой вход – размер класса). Всего имеется 151 запись, пропущенных значений нет.
6	Из переписи данных США 1994 взята небольшая выборка о взрослых людях. Она содержит 14 признаков (более подробно они описаны в файле к заданию). Требуется предсказать получает ли этот человек в год более 50000 долларов или нет. Выборка имеет пропущенные данные.
7	Шкала баланса. Имеются весы с двумя плечами, однако длины левого и правого плеча могут меняться. К каждому плечу подвешивался груз разной массы. От испытуемого требовалось установить будут ли весы сбалансированы или отклонятся влево, или вправо. На основании этого психологического

	эксперимента составлена выборка. Имеются 4 входа: вес для левого плеча, его длина, вес для правого плеча и его длина. Есть три выходных класса: B – баланс (когда произведение левого плеча

	на левый груз равно произведению правого плеча на правый груз), R – отклонение вправо, L – отклонение влево. Пропущенных данных нет. Более подробная информация в файле задания.
8	Нарушение печени. Имеются 5 входных параметров, которые анализируют работу печени. Требуется определить является ли человек алкоголиком или нет. Более подробная информация в файле задания.
9	Набор данных о выживании Хабермана. Этот набор данных содержит случаи из исследования, которое проводилось между 1958 и 1970 годами в больнице Биллингса Университета Чикаго по выживанию пациентов, перенесших операцию по удалению рака из молочной железы. Задача состоит в том, чтобы определить, выжил ли пациент в течении последующих 5 лет или более (положительный результат), или нет (отрицательный результат). Пропущенных данных нет. Более подробная информация в файле задания.
10	Из силуэта транспортного средства извлекли ряд признаков. По ним требуется классифицировать тип транспортного средства. Задача похожа на задачу из варианта 3. Более подробная информация в файле задания.
11	Имеется ряд параметров, измеряющих работу щитовидной железы. Требуется установить является ли человек нормальным (1) или страдает гипертиреозом (2), или гипотиреозом (3). Более подробная информация в файле задания.
12	Набор данных дрожжей. Эта база данных содержит информацию о наборе дрожжевых клеток. Задача состоит в том, чтобы определить локализацию каждой клетки среди 10 возможных вариантов. Более подробная информация в файле задания.

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1. Название лабораторной работы и её номер.
2. ФИО студента и группу.
3. Формулировка индивидуального задания.
4. Документ отчёта с Print Prtscr диалоговых окон по шагам для своего варианта по подобию того, что описано в теоретической части. Должна присутствовать информация о количестве классов, количестве паттернов в каждом классе, о размерах тестовой и обучающей выборки, о проделанной предобработке, о представленности каждого класса в тестовой и обучающей выборке, об используемых алгоритмах обучения нейронных сетей, а также результирующие графики и процент правильно распознанных паттернов на тестовой выборке. Необходимо также указать структуру сети и обосновать почему выбрана именная такая сеть.
5. Ответы на контрольные вопросы.

Вопросы для защиты работы

1) С какими типами переменных может работать нейронная сеть?

2) Какими способами можно предобработать входные и выходные данные для нейронной сети?
3) Чем ординарные данные отличаются от категориальных?
4) На какие ключевые вопросы нужно дать ответы для этапа предобработки данных?
5) Как нужно разбить выборку для теста и обучения если имеются два неравномерно представленных класса?
6) Как нужно разбить выборку для теста и обучения если имеются два равномерно представленных класса?
7) Какие типы нормировок вы знаете?

Лабораторная работа № 7

Пример создания и обучения нейронных сетей для задач регрессии в среде Matlab.

Цель и содержание работы: создать и обучить нейронную сеть для задачи регрессии.

Задачи:

- понять каким образом нейронные сети решают задачу регрессии;
- рассмотреть постобработку данных;
- создать и обучить нейронную сеть для задачи регрессии.

Теоретическое обоснование

В предыдущих лабораторных работах рассматривалось, как нейронные сети могу осуществлять классификацию входных паттернов, однако нейронные сети можно использовать и для осуществления предсказаний значений (задача регрессии). Предсказание отличается от задачи классификации тем, что на выходе сеть должна получить некоторое значение, а не некоторую фиксированную метку класса. Тем не менее, процесс обучения сети остаётся тем же. Каждый входной паттерн и выходную метку можно рассматривать как некоторую координату в многомерном пространстве, тогда задача обучения сводится к тому, чтобы аппроксимировать эти данные некоторой нелинейной моделью, так чтобы ещё появились интерполирующие свойства. Сеть добивается для каждой такой точки получать наиболее близкий ответ к учителю, и за счёт этого, по факту, сама сеть превращается в некоторую аппроксимирующую модель, на рисунке 7.1 – это кривая, проходящая через точки обучающего набора. Однако через набор таких точек можно провести бесконечно много кривых, на рисунке 7.1 вверху и внизу проходят разные кривые. Но нетрудно заметить, что нижняя кривая не имеет предсказательной способности (говорят, что сеть переобучилась), а верхняя кривая вполне подходит для предсказания ответа на тестовых данных (кружок на рисунке 7.1).

Рисунок 7.1 – Две кривые аппроксимирующие один и тот же набор данных

Интерполяционная способность достигается ростом объёма обучающей выборки до момента описания выборкой всех статистических свойств генеральной совокупности и контролем размера и свойств сети. Хорошая аппроксимация ещё ничего не говорит про ошибку обобщения.

Нейронные сети, как уже отмечалось выше, строят нелинейные модели. Самая же простая модель для предсказания – это линейная регрессия. В файле «DataForRegression.xls» на листе «Пример» в столбцах A и B приведена простая зависимость количества предметов от месяца. Если ось OX будет обозначать номер месяца, а ось OY будет обозначать количество предметов, то легко можно построить график (синий цвет на рисунке 7.2).

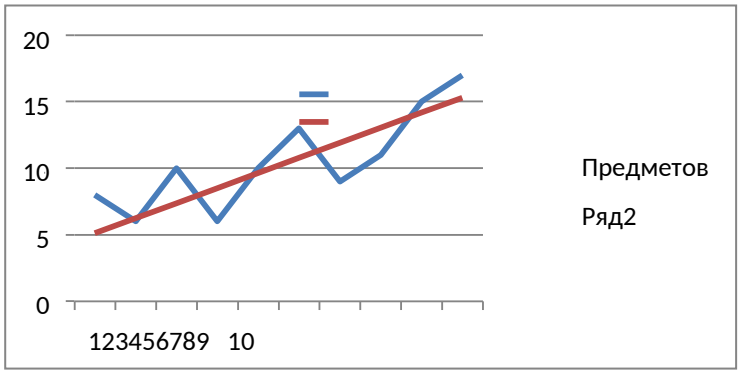


Рисунок 7.2 – Исходные данные и регрессионная модель

С помощью средств Excel мы можем, отталкиваясь от этих данных, построить линейную регрессионную модель, т.е. в данном случае обычную прямую, но такую, которая

так бы проходила около или через эти точки, что давала бы наиболее близкие ответы к исходным данным. Коэффициенты этой прямой выделены жёлтым цветом в столбце F. Получилось следующее уравнение прямой: y=1.13*x+3.97. Это и есть красная прямая на рисунке 7.2. Значения, предсказанные этой прямой для соответствующих месяцев, приведены в столбце A и B. В данном примере мы можем легко визуально построить модель, но в реальности, если входных значений будет много, то мы не сможем визуально отобразить нашу многомерную прямую. Так на листе «Задача» в столбцах A-L даются данные о стоимости домов в одном из округов США в 2014-2015 годах. Price – цена дома в долларах, train – метка для обучающей или тестовой выборки (1 – паттерн для обучающей выборки, 0 – для тестовой, они выделены красным цветом), bathrooms – количество ванных комнат, sqft_living – жилая площадь, waterfront – близко ли дом к воде, возможные значения [0, 1], view – характеристика вида возле дома в некоторой шкале возможных значений [0, 1, 2, 3, 4], condition – состояние дома в некоторой шкале возможных значений [1, 2, 3, 4, 5], grade – общая оценка дома по некоторой шкале [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13], yr_built – год постройки, zipcode - индекс из адреса дома, lat – широта из координат дома на карте, long – долгота из координат дома на карте. Соответственно требуется с помощью линейной регрессии предсказать цену дома. В столбцах P-Y представлены результаты подбора коэффициентов для данной регрессионной задачи. Получается уравнение 10-мерной прямой. Коэффициенты для этой прямой выделены жёлтым цветом, а в столбце N представлены ответы данной модели на входные данные, т.е. предсказанная цена.

Так вот, нейронная сеть также строит некоторую, но уже нелинейную модель, отсюда её предсказательные способности ещё больше, но получить готовую модель сложнее из-за риска переобучения. Переобучение наступает тогда, когда сеть имеет избыточные ресурсы (много настраиваемых параметров по отношению к обучающим данным), тогда сеть моделирует слишком сложную зависимость вместо более простой.

При использовании нейронной сети для предсказаний возникает задача обработать не только входные, но и выходные данные. Обычно выходные данные также приводятся к диапазону 0..1 или -1..1, а когда сеть тестируют на тестовой выборке, то выходные значения вновь расширяют к искомому диапазону. Если, допустим, изначально выходные значения были в диапазоне [A..B], то для обучения мы сжимаем их по формулам нормировки (лабораторная работа 6) к диапазону [-1..+1], а при тесте ответ сети, данные из диапазона [-1..+1], опять разжимаем к диапазону [A..B], чтобы получился осмысленный ответ в терминах данных задачи. Решим задачу предсказания стоимости недвижимости в районах Бостона.

Прогноз стоимости недвижимости в районах Бостона.

Исходные данные содержатся в файле «DataForBoston.doc», однако, учитывая, что это одна из задач-примеров в Matlab, исходные данные можно загрузить и через команду.

Требуется построить прогностическую систему для оценки стоимости жилья в Бостоне, основываясь на следующих 13 параметрах, в скобках приведены диапазоны, которые может принимать параметр:

1. CRIM – уровень преступности на душу населения [0.0063, 88.98).
2. ZN – доля жилых массивов на площади 25.000 футов^2 [0, 100].
3. NDUS – доля предприятий, не связанных с розничной торговлей [0.46, 27.74].
4. CHAS – близость к реке Charles River (1 – если район граничит с рекой, 0 – в противном случае).
5. NOX – концентрация окисей азота (в долях 1/10 миллионов) [0.385, 0.871).
6. RM – среднее число комнат в жилище [3.561, 8.78].
7. AGE – доля частных владений, построенных до 1940 г [2.9, 100].
8. DIS – взвешенное расстояние до 5 центров сосредоточения работы в Бостоне [1.1296, 12.1265].
9. RAD – индекс доступности радиальных автострад [1..24].
10. TAX – полный налог на недвижимость в расчёте на $10.000 [187, 711].
11. PTRATIO – доля школьных учителей в районе [12.6, 22].
12. B – коэффициент $10^3*(Bk-0.63)^2$, где Bk – доля чернокожего населения [0.32, 396.9].
13. LSTAT – процент населения ниже черты бедности [1.73, 37.97].

MEDV – параметр, который нужно предсказать: средняя цена частного жилья в $1000 [5, 50].

Всего выборка содержит 506 записей.

Выборку можно также скачать с сайта DELVE Project или с http://mlearn.ics.uci.edu/MLRepository.html. Более подробную информацию по выборке можно посмотреть в [1].

Приведём листинг решения с комментариями.

```
% Загрузка искомой выборки
load house_dataset;
% Входные значения
x = houseInputs;
% Выходные значения
t = houseTargets;
% Алгоритм обучения сети
trainFcn = 'trainlm';
% Количество нейронов в промежуточном слое
```

```
hiddenLayerSize = 15;
% Создание сети для предсказаний
net = fitnet(hiddenLayerSize,trainFcn);
% Обрабатывать возможные пропуски во входных данных
% Используя линейную нормировку сжать все данные к диапазону [-1..+1]
net.input.processFcns = {'removeconstantrows','mapminmax'};
% Аналогично предобработать выходы
net.output.processFcns = {'removeconstantrows','mapminmax'};
% Случайно разбить выборку на обучающую и валидационную
net.divideFcn = 'dividerand';
net.divideMode = 'sample';
%  В обучающей 85% всех паттернов
net.divideParam.trainRatio = 85/100;
% В валидационной 15%
net.divideParam.valRatio = 15/100;
% В тестовой ничего нет
net.divideParam.testRatio = 0/100;
net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate','ploterrhist',  'plotregression', 'plotfit'};
% Обучить сеть
[net,tr] = train(net,x,t);
% Получить ответы сети для валидационного множества
y1 = net(x(tr.valInd(1,:)));
% Получить идеальные ответы
t1=t(tr.valInd(1,:));
% Вычислить их разность
e1 = gsubtract(t1, y1);
% Посчитать количество таких разностей по модулю, которые
% меньше 10, выразить это число в процентах
col = length(find(abs(e1(1, :))<=10))/length(e1)*100;
% Получившейся ответ для задачи col = 59.21;
```

Процесс обучения и структура сети представлены на рисунке 7.3, видно, что на выходном слое находится линейная функция активации. Увеличивать количество нейронов в скрытом слое особого смысла не имеет, т.к. тогда при возрастании качества предсказания на обучающей выборки, будет падать качество предсказания на тестовой выборки. На рисунке 7.4 показан процесс изменения ошибки обучения и обобщения. На рисунке 7.5 и 7.6 представлены дополнительные графики, которые дают возможность более точно оценить степень предсказательной способности сети.

Рисунок 7.3 – Процесс обучения и структура сети для предсказания

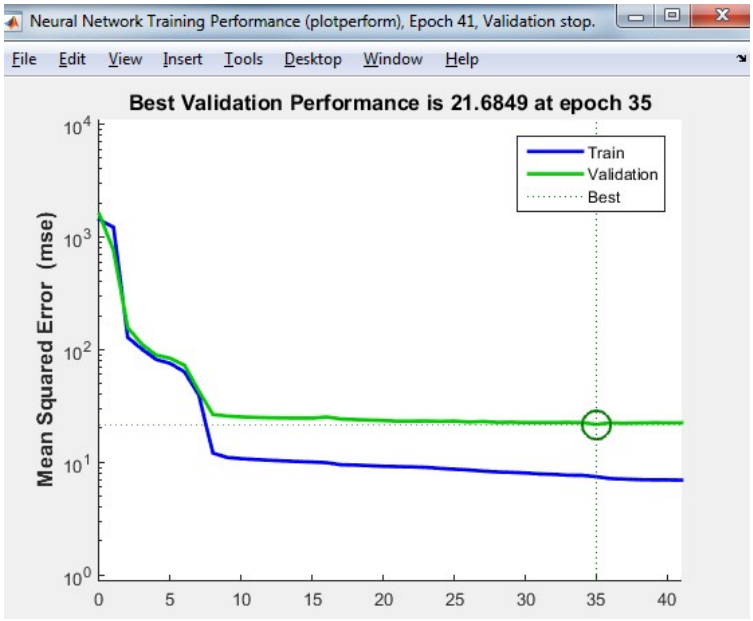


Рисунок 7.4 – Изменения ошибки обучения и ошибки обобщения

Рисунок 7.5 – Диаграмма распределения фактической ошибки при предсказании

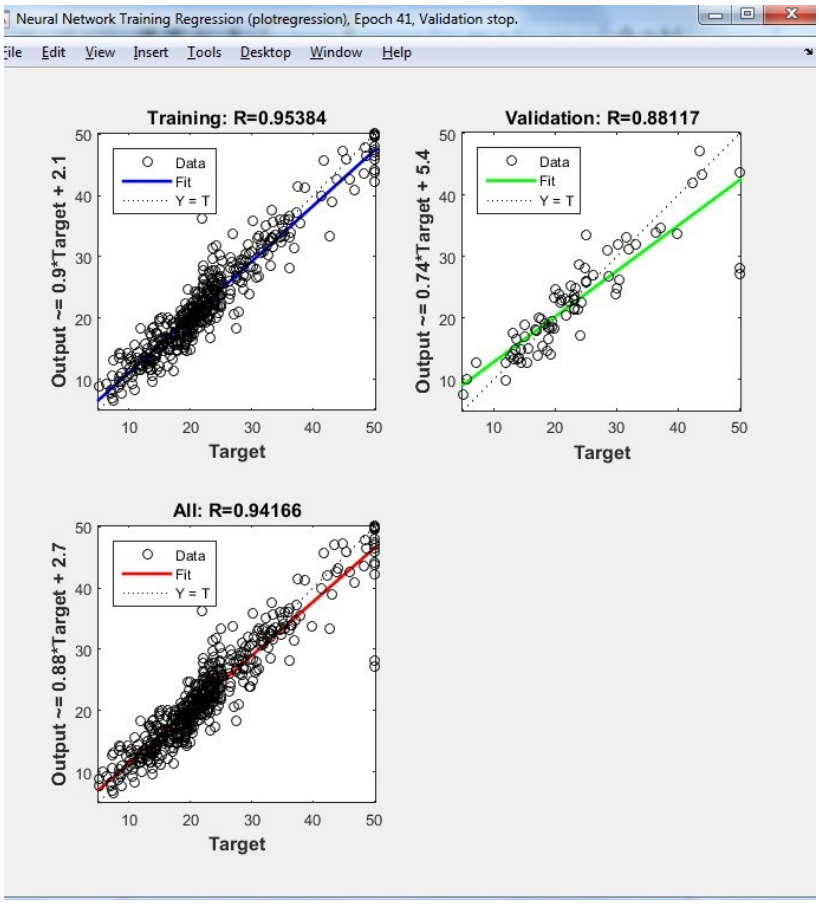


Рисунок 7.6 – Отношение данных (кружки) к получившимся моделям (линии)

В результате имеем почти 60% значений, где предсказанная цена на 10 единиц отличается от идеальной требуемой.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

В индивидуальном варианте (таблица 7.1) дано условие задачи на регрессию. Решить задачу с точностью предсказания на тестовой выборке более 60%.

Таблица 7.1 – Варианты заданий

№	Условие задачи
1	Используя данные из задачи о стоимости жилья в США составить нелинейную регрессионную модель с помощью нейронной сети. Данные содержатся в листе Data файла «DataForRegression.xls». Сравнить ошибку MSE для предсказанных данных с помощью линейной регрессионной модели и предсказанных данных нейронной сетью.
2	Предсказать решение верховного суда. Описание входов и выхода смотрите в файле задания.
3	Предсказать расход топлива по городу. Имеется один дискретный вход и 5 непрерывных. Описание входов и выходов смотрите в файле задания.
4	Предсказать расход топлива по городу. Имеется 3 дискретных входа и 5 непрерывных. Описание входов и выходов смотрите в файле задания.
5	Предсказать среднюю стоимость жилья в Калифорнии. Задача аналогичная двум рассматриваемым в теоретической части. Описание входов и выходов смотрите в файле задания.
6	Предсказать силу сопротивления сжатию для цемента. Описание входов и выходов смотрите в файле задания.
7	Предсказать угол отклонения элерона для самолёта F16. Описание входов и выходов смотрите в файле задания.
8	Предсказать уровень С-пептидов в сыворотке крови детей больных диабетом. Описание входов и выходов смотрите в файле задания.
9	Требуется предсказать недвижимость домов в США в зависимости от демографических данных района и рынка недвижимости. Описание входов и

	выходов смотрите в файле задания.
10	Предсказать искусственно сформированную зависимость Y от X1..X10. Описание входов и выходов, а также самой зависимости, смотрите в файле задания.
11	Предсказать давление пластика при определённой силе и температуре. Описание входов и выходов смотрите в файле задания.

1 2	Предсказать угловое ускорение для одной из степени свободы механической руки Puma 560. Описание входов и выходов смотрите в файле задания. Информацию по механическому манипулятору можно найти в сети Интернет.

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1. Название лабораторной работы и её номер.
2. ФИО студента и группу.
3. Формулировка индивидуального задания.
4. Документ отчёта с Print Prtscr диалоговых окон по шагам для своего варианта по подобию того, что описано в теоретической части. Должна присутствовать информация о размерах тестовой и обучающей выборки, о проделанной предобработке и постобработке, об используемых алгоритмах обучения нейронных сетей, а также результирующие графики и процент правильно распознанных паттернов на тестовой выборке. Необходимо также указать структуру сети и обосновать, почему выбрана именная такая сеть.
5. Ответы на контрольные вопросы.

Вопросы для защиты работы

1) С какими типами переменных может работать нейронная сеть?
2) Как осуществляется предсказание значений нейронной сетью?
3) В чем заключаются аппроксимирующие и интерполирующие свойства нейронных сетей?
4) Как осуществлять постобработку и для чего она нужна?
5) Нейроны с какими функциями активации можно ставить на выходной слой?
6) Чем нейронная сеть отличается от линейной регрессионной модели?
7) Что такое переобучение сети и как с ним бороться?

Приложение 1

```
function [alpha, b, w, evals, stp, glob] = SMO2(X, Y, kernel, kpar1, kpar2, C, tol, steps, eps, method)

dbstop if error
dbg = 2;

if (nargin < 10)
   method = 1;
end

if (nargin < 9)
   eps = 0.0001;
end

if (nargin < 8)
   steps = 10000;
end

if (nargin < 7)
   tol = 0.001;
end

if (nargin < 6)
   C = inf;
end

if (nargin < 5)
   kpar2 = 0;
end

if (nargin < 4)
   kpar1 = 0;
end

if (nargin < 3)
   kernel = 0;
end

if (nargin < 2)
   error('Error: At least two arguments (training points and class values) must be supplied');
else
   [n, D] = size(X);
   [n1, D1] = size(Y);
end

if (1 ~= D1)
   error('Error: Class values cannot be vectors but real numbers');
end

if (n ~= n1)
   error('Error: Number of rows of X and Y must be the same (one class value for each sample)');
end


if (method == 1)
   [alpha, b, w, evals, stp, glob] = SMO_Keerthi_modif1(X, Y, kernel, kpar1, kpar2, C, tol, steps, eps);
elseif (method == 2)
   [alpha, b, w, evals, stp, glob] = SMO_Keerthi_modif2(X, Y, kernel, kpar1, kpar2, C, tol, steps, eps);
else
   [alpha, b, w, evals, stp, glob] = SMO_Platt(X, Y, kernel, kpar1, kpar2, C, tol, steps, eps);
end

if(method==1)||(method==2)
   flag=(glob.b_up<glob.b_low-2*tol)|(stp>=steps);
else
   flag=(stp>=steps);
end

if(flag==1)
   fprintf('The algorithm has not converged. This may be due to:\n (a) the maximum number of iterations has been reached and
convergence has not, yet, been achieved or \n (b) the chosen values for the hyperparameters (C as well as the parameters that define the
kernel function) \n can not lead to a solution. \n')
end


function [alpha, b, w, evals, stp, glob] = SMO_Platt(X, Y, kernel, kpar1, kpar2, C, tol, steps, eps)
```

```
[n, D] = size(X);
%initialize alpha array to all zero
alpha = zeros(n,1);
w = zeros(1,D);
b = 0;
evals = 0;

for i=1:n
   K(:,i)=CalcKernel(X,X(i,:), kernel, kpar1, kpar2);
end


%initialize struct for temporary variables that must be global
glob = struct('ecache',[],'v_1',[],'v_2',[],'I_0',[],'ecache_f',[]);
%%initialize fcache array to all zero and its size to n
glob.ecache = zeros(n,1);
glob.ecache_f = zeros(n,1); %0->ecache value not-OK, 1->value OK
glob.v_1 = find(Y==-1);
glob.v_2 = find(Y==1);

stp = 0;
numChanged = 0;
examineAll = 1;
while ((numChanged > 0 || examineAll == 1) && stp <= steps)
   numChanged = 0;
   if (examineAll == 1)
      for i = 1 : n
         stp = stp + 1;
         if (stp > steps) break; end
         [retval, alpha, w, b, stp, evals, glob] =...
            examineExampleP(i, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, steps, stp, evals, eps,K);
         numChanged = numChanged + retval;
      end
   else
      glob.I_0 = find(alpha > 0 & alpha < C);
      k = length(glob.I_0);
      for i = 1 : k
         stp = stp + 1;
         if (stp > steps) break; end
         if (i > length(glob.I_0)) break; end %glob.I_0 changes inside loop (in examineExampleP)
         [retval, alpha, w, b, stp, evals, glob] =...
            examineExampleP(glob.I_0(i), glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, steps, stp, evals, eps,K);
         numChanged = numChanged + retval;
      end
   end
   if (examineAll == 1)
      examineAll = 0;
   elseif (numChanged == 0)
      examineAll = 1;
   end
end


function [retval, alpha, w, b, evals, glob] =...
   takeStepP(i1, i2, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, evals, eps,K)

%drawnow; %%used to give Matlab the opportunity to examine any pending ctrl+C (while in deep loops)
if (get(0,'PointerLocation')==[1 1])
   %disp('Press <ctrl+c> to stop or any key to interrupt execution temporarily ...');
   %pause;
   %disp('Type "return" to carry on');
   keyboard;
end

[n D] = size(X);
if (i1 == i2)
   retval = 0;
   return;
end
alph1 = alpha(i1);
y1 = Y(i1);
alph2 = alpha(i2);
y2 = Y(i2);
s = y1 * y2;
%Compute L, H
if (y1 ~= y2)
   L = max([0, alph2 - alph1]);
   H = min([C, C + alph2 - alph1]);
else % y1 == y2
```

```
    L = max([0, alph1 + alph2 - C]);
    H = min([C, alph1 + alph2]);
end
if (L == H)
    retval = 0;
    return;
end
%calculate E1 = SVM output in X[i1] - y1 (check in error cache)
if (glob.ecache_f(i1) == 0)
    ki1 = K(:,i1);
    evals = evals + n;
    E1 = -y1 + (ki1' * (Y .* alpha)) - b;
    glob.ecache(i1) = E1;
    glob.ecache_f(i1) = 1;
else
    E1 = glob.ecache(i1);
end
%calculate E2 = SVM output in X[i2] - y2 (check in error cache)
if (glob.ecache_f(i2) == 0)
    ki2 = K(:,i2);
    evals = evals + n;
    E2 = -y1 + (ki2' * (Y .* alpha)) - b;
    glob.ecache(i2) = E2;
    glob.ecache_f(i2) = 1;
else
    E2 = glob.ecache(i2);
end
%%computation of the derivative eta
k11 = K(i1,i1);
k12 = K(i2,i1);
k22 = K(i2,i2);
evals = evals + 3;
eta = -(2 * k12) + k11 + k22;
%%computation of new alpha(i2)
if (eta > 0)
    a2 = alph2 + (y2 * (E1 - E2) / eta);
    if (a2 < L)
        a2 = L;
    elseif (a2 > H)
        a2 = H;
    end
else %%the derivative is 0 => we have to make optimization by other means
    %%Lobj = objective function at a2=L (according to Platt)
    %%Hobj = objective function at a2=H (according to Platt)
    L1 = alph1 + (s * (alph2 - L));
    H1 = alph1 + (s * (alph2 - H));
    f1 = y1 * (E1 + b) - (alph1 * k11) - (s * alph2 * k12);
    f2 = y2 * (E2 + b) - (alph2 * k22) - (s * alph1 * k12);
    Lobj = (L1 * f1) + (L * f2) + (0.5 * k11 * L1^2) + (0.5 * k22 * L^2) + (s * k12 * L * L1);
    Hobj = (H1 * f1) + (H * f2) + (0.5 * k11 * H1^2) + (0.5 * k22 * H^2) + (s * k12 * H * H1);
    if (Lobj < Hobj - eps)
        a2 = L;
    elseif (Lobj > Hobj + eps)
        a2 = H;
    else
        a2 = alph2;
    end
end
if (abs(a2 - alph2) < (eps * (a2 + alph2 + eps)))
    retval = 0;
    return
end
% computation on new a1pha1(a1)
a1 = alph1 + (s * (alph2 - a2));
%Update threshold to reflect change in Lagrange multipliers
b_old = b;
if (a1 > L && a1 < H)
    b = E1 + (y1 * (a1 - alph1) * k11) + (y2 * (a2 - alph2) * k12) + b;
elseif (a2 > L && a2 < H)
    b = E2 + (y1 * (a1 - alph1) * k12) + (y2 * (a2 - alph2) * k22) + b;
else
    b1 = E1 + (y1 * (a1 - alph1) * k11) + (y2 * (a2 - alph2) * k12) + b;
    b2 = E2 + (y1 * (a1 - alph1) * k12) + (y2 * (a2 - alph2) * k22) + b;
    b = (b1 + b2) / 2;
end
%Update weight vector to reflect change in a1 & a2, if linear SVM
if (strcmpi(kernel, 'linear') == 1)
    w = w + (y1 * (a1 - alph1) * X(i1,:)) + (y2 * (a2 - alph2) * X(i2,:));
end
```

```
%Update ecache[i] using new Lagrange multipliers
v = find(glob.ecache_f==1);
for i = 1 : length(v)
   ki1 = K(v(i),i1);
   ki2 = K(v(i),i2);
   evals = evals + 2;
   %glob.ecache(v(i)) = glob.ecache(v(i)) + b - b_old - (y1 * (a1 - alph1) * ki1) - (y2 * (a2 - alph2) * ki2);
   glob.ecache(v(i)) = glob.ecache(v(i)) + b_old - b + (y1 * (a1 - alph1) * ki1) + (y2 * (a2 - alph2) * ki2);
end
%Store a1 and a2 in the alpha array
alpha(i1) = a1;
alpha(i2) = a2;

%%Compute updated E values for i1 and i2
%glob.ecache(i1) = E1 + b - b_old - (y1 * (a1 - alph1) * k11) - (y2 * (a2 - alph2) * k12);
glob.ecache(i1) = E1 + b_old - b + (y1 * (a1 - alph1) * k11) + (y2 * (a2 - alph2) * k12);
glob.ecache_f(i1) = 1;
%glob.ecache(i2) = E2 + b - b_old - (y1 * (a1 - alph1) * k12) - (y2 * (a2 - alph2) * k22);
glob.ecache(i2) = E2 + b_old - b + (y1 * (a1 - alph1) * k12) + (y2 * (a2 - alph2) * k22);
glob.ecache_f(i2) = 1;
% Update I_0
glob.I_0 = find(alpha > 0 & alpha < C);

retval = 1;
return


function [retval, alpha, w, b, stp, evals, glob] =...
   examineExampleP(i2, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, steps, stp, evals, eps,K)

%drawnow; %%used to give Matlab the opportunity to examine any pending ctrl+C (while in deep loops)
if (get(0,'PointerLocation')==[1 1])
   %disp('Press <ctrl+c> to stop or any key to interrupt execution temporarily ...');
   %pause;
   %disp('Type "return" to carry on');
   keyboard;
end

retval = 0;
[n D] = size(X);
y2 = Y(i2);
alph2 = alpha(i2);
if (glob.ecache_f(i2) == 1)
   E2 = glob.ecache(i2);
else
   ki2 = K(:,i2);
   evals = evals + n;
   E2 = -y2 + (ki2' * (Y .* alpha)) - b;
   glob.ecache(i2) = E2;
   glob.ecache_f(i2) = 1;
end
r2 = E2 * y2;
if ((r2 < -tol && alph2 < C) || (r2 > tol && alph2 > 0))
   if (length(glob.I_0) > 1)
      %i1 = result of second choice heuristic
      v = find(glob.ecache_f==1);
      k = length(v);
      Emax = 0;
      for i = 1 : k
         tmp = abs(glob.ecache(v(i)) - E2);
         if (tmp > Emax)
            Emax = tmp;
            i1 = v(i);
         end
      end
      stp = stp + 1;
      [retval, alpha, w, b, evals, glob] =...
         takeStepP(i1, i2, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, evals, eps,K);
      if (retval == 1)
         return;
      end
   end
   %loop over all non-zero and non-C alpha, starting at a random point
   k = length(glob.I_0);
   rand('state',2);
   r = floor(k * rand);
   for i = 1 : k
      i1 = mod(r + i, k) + 1;
      stp =stp + 1;
```

```
      [retval, alpha, w, b, evals, glob] =...
         takeStepP(i1, i2, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, evals, eps,K);
      if (retval == 1)
         return;
      end
   end
   %loop over all possible i1, starting at a random point
   k = n;
   r = floor(k * rand);
   for i = 1 : k
      i1 = mod(r + i, k) + 1;
      stp = stp + 1;
      [retval, alpha, w, b, evals, glob] =...
         takeStepP(i1, i2, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, evals, eps,K);
      if (retval == 1)
         return;
      end
   end
end
return


function [alpha, b, w, evals, stp, glob] = SMO_Keerthi_modif1(X, Y, kernel, kpar1, kpar2, C, tol, steps, eps)
[n, D] = size(X);
%initialize alpha array to all zero
%-------------------------------------------------------------------------
alpha = zeros(n,1);
w = zeros(1,D);
b = 0;
evals = 0;
%-------------------------------------------------------------------------

for i=1:n
   K(:,i)=CalcKernel(X,X(i,:), kernel, kpar1, kpar2);
end

%initialize struct for temporary variables that must be global
%-------------------------------------------------------------------------

glob = struct('fcache',[],'b_up',0,'b_low',0,'i_up',0,'i_low',0,'v_1',[],'v_2',[],...
   'I_0',[],'I_1',[],'I_2',[],'I_3',[],'I_4',[]);
%%initialize fcache array to all zero and its size to n
glob.fcache = zeros(n,1);
%initialize b_up = -1, i_up to any one index of class 1
glob.b_up = -1;
glob.v_1 = find( Y==1 );
glob.i_up = glob.v_1(1);
%initialize b_low = 1, i_low to any one index of class 2
glob.b_low = 1;
glob.v_2 = find( Y==-1 );
glob.i_low = glob.v_2(1);
%set fcache[i_low] = 1 and fcache[i_up] = -1
glob.fcache(glob.i_low) = 1;
glob.fcache(glob.i_up) = -1;

%Initialize the I_* sets
glob.I_0 = find(alpha > 0 & alpha < C);
glob.I_1 = find(alpha(glob.v_1) == 0);
glob.I_1 = glob.v_1(glob.I_1);
glob.I_2 = find(alpha(glob.v_2) == C);
glob.I_2 =  glob.v_2(glob.I_2);
glob.I_3 = find(alpha(glob.v_1) == C);
glob.I_3 =  glob.v_1(glob.I_3);
glob.I_4 = find(alpha(glob.v_2) == 0);
glob.I_4 = glob.v_2(glob.I_4);
%-------------------------------------------------------------------------


stp = 0;
numChanged = 0;
examineAll = 1;
while ( ((numChanged > 0) || (examineAll==1))&&(stp <= steps))
   numChanged = 0;
   if (examineAll==1)
      for i = 1 : n
         stp = stp + 1;
         if (stp>steps)  break;      end
         [retval, alpha, w, b, stp, evals,glob] =...
            examineExampleK(i, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, steps, stp, evals, eps,K);
```

```
            numChanged = numChanged + retval;
        end
    else
        k = length(glob.I_0);
        for i = 1 : k
            stp = stp + 1;
            if (stp > steps) break; end
            if (i > length(glob.I_0)) break; end %glob.I_0 changes inside loop (in examineExampleK)
            [retval, alpha, w, b, stp, evals, glob] =...
                examineExampleK(glob.I_0(i), glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, steps, stp, evals, eps,K);
            numChanged = numChanged + retval;
            %it is easy to check if optimality on I_0 is attained...
            if ( (glob.b_up) > ( glob.b_low - (2*tol) ) )
                %exit the loop after setting numChanged = 0
                numChanged = 0;
                break;      %EDW TA PRAGMATA ALLAZOUN AN XRHSIMOPOIHSOUME THN RETURN. SYSKEKRIMENA
MERIKES FORES ENW EXEI SYMBEI
                %TO b_up>b_low POY EINAI TO ZHTOUMENO, ME THN BREAK BGAINOUME MONO APO TO ESWTERIKO
FOR (TO PSAKSIMO STO I_0)
                %ENW ISWS (DEN EIMAI SIGOUROS) THA EPREPE NA TERMATIZEI O ALGORITHMOS. ME THN BREAK TO
PROGRAMMA SYNEXIZEI
                %KAI ALLAZOUN TA b_up KAI b_low KAI SXHMATIKA FAINETAI KALYTEROS O TAKSIMOMHTHS.
                %ME THN RETURN OMWS EINAI PIO SYNTOMOS KAI PALI FAINETAI SWSTOS. NA TO PSAKSOYME.
            end
        end
    end
    if (examineAll == 1)
        examineAll = 0;
    elseif (numChanged == 0)
        examineAll = 1;
    end
    b=(glob.b_up+glob.b_low)/2;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [alpha, b, w, evals, stp, glob] = SMO_Keerthi_modif2(X, Y, kernel, kpar1, kpar2, C, tol, steps, eps)
[n, D] = size(X);
%initialize alpha array to all zero
%------------------------------------------------------------------------
alpha = zeros(n,1);
% if (strcmpi(kernel, 'linear') == 1)
w = zeros(1,D);
% else
%     w = [];
% end
b = 0;
evals = 0;
%------------------------------------------------------------------------

for i=1:n
    K(:,i)=CalcKernel(X,X(i,:), kernel, kpar1, kpar2);
end
%initialize struct for temporary variables that must be global
%------------------------------------------------------------------------

glob = struct('fcache',[],'b_up',0,'b_low',0,'i_up',0,'i_low',0,'v_1',[],'v_2',[],...
    'I_0',[],'I_1',[],'I_2',[],'I_3',[],'I_4',[]);
%%initialize fcache array to all zero and its size to n
glob.fcache = zeros(n,1);
%initialize b_up = -1, i_up to any one index of class 1
glob.b_up = -1;
glob.v_1 = find( Y==1 );
glob.i_up = glob.v_1(1);
%initialize b_low = 1, i_low to any one index of class 2
glob.b_low = 1;
glob.v_2 = find( Y==-1 );
glob.i_low = glob.v_2(1);
%set fcache[i_low] = 1 and fcache[i_up] = -1
glob.fcache(glob.i_low) = 1;
glob.fcache(glob.i_up) = -1;

%Initialize the I_* sets
```

```
glob.I_0 = find(alpha > 0 & alpha < C);
glob.I_1 = find(alpha(glob.v_1) == 0);
glob.I_1 = glob.v_1(glob.I_1);
glob.I_2 = find(alpha(glob.v_2) == C);
glob.I_2 =  glob.v_2(glob.I_2);
glob.I_3 = find(alpha(glob.v_1) == C);
glob.I_3 =  glob.v_1(glob.I_3);
glob.I_4 = find(alpha(glob.v_2) == 0);
glob.I_4 = glob.v_2(glob.I_4);
%--------------------------------------------------------------------------


stp = 0;
numChanged = 0;
examineAll = 1;
while ((numChanged > 0 || examineAll == 1) && stp <= steps)
   numChanged = 0;
   if (examineAll == 1)
      for i = 1 : n
         stp = stp + 1;
         if (stp > steps) break; end
         [retval, alpha, w, b, stp, evals, glob] =...
            examineExampleK(i, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, steps, stp, evals, eps,K);
         numChanged = numChanged + retval;
      end
   else
      %the following loop is the only difference between the two SMO
      %modifications. Whereas, in modification 1, the inner loop selects
      %i2 from I_0 sequentially, here i2 is always set to the current
      %i_low and i1 is set to the current i_up; clearly, this corresponds
      %to choosing the worst violating pair using members of I_0 and some
      %other indices.
      inner_loop_success = 1;
      while ( ((glob.b_up)<(glob.b_low -(2*tol)))&&(inner_loop_success~=0))
         i2 = glob.i_low;
         y2 = Y(i2);
         alph2 = alpha(i2);
         F2 = glob.fcache(i2);
         i1 = glob.i_up;
         stp = stp + 1;
         if (stp > steps) break; end
         stp = stp + 1;
         [inner_loop_success, alpha, w, b, evals, glob] =...
            takeStepK(glob.i_up, glob.i_low, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, evals, eps,K);
         numChanged = numChanged + inner_loop_success;
      end
      num_changed = 0;
   end
   if (examineAll == 1)
      examineAll = 0;
   elseif (numChanged == 0)
      examineAll = 1;
   end
end
b=(glob.b_up+glob.b_low)/2;
return




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [retval, alpha, w, b, evals, glob] =...
   takeStepK(i1, i2, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, evals, eps,K)
%Much of this procedure is same as that in Platt's SMO pseudo-code.

%drawnow; %%used to give Matlab the opportunity to examine any pending ctrl+C (while in deep loops)
if (get(0,'PointerLocation')==[1 1])
   %disp('Press <ctrl+c> to stop or any key to interrupt execution temporarily ...');
   %pause;
   %disp('Type "return" to carry on');
   keyboard;
end
if (i1 == i2)
   retval = 0;
   return;
end
alph1 = alpha(i1);
```

```
y1 = Y(i1);
F1 = glob.fcache(i1);
alph2  = alpha(i2);
y2 = Y(i2);
F2 = glob.fcache(i2);
s = y1 * y2;


%Compute L, H - If L=H return 0
%----------------------------------------------------------------------
if (y1 ~= y2)
   L = max([0, alph2 - alph1]);
   H = min([C, C + alph2 - alph1]);
else % y1 == y2
   L = max([0, alph1 + alph2 - C]);
   H = min([C, alph1 + alph2]);
end
if (L == H)
   retval = 0;
   return;
end
%----------------------------------------------------------------------


%%computation of the derivative eta
%----------------------------------------------------------------------
k11 = K(i1,i1);
k12 = K(i1,i2);
k22 = K(i2,i2);
evals = evals + 3;
eta = (2*k12) - k11 - k22;
%%computation of new alpha(i2)
if (eta<0)
   a2 =alph2-(y2*(F1-F2)/eta); %%HERE is different from Platt
   if (a2 < L)
      a2 = L;
   elseif (a2 > H)
      a2 = H;
   end
else %%the derivative is 0 => we have to make optimization by other means
   %%Lobj = objective function at a2=L (according to Platt)
   %%Hobj = objective function at a2=H (according to Platt)
   L1 = alph1 + (s*(alph2-L));
   H1 = alph1 + (s*(alph2-H));
   f1 = (-y1 * F1) + (alph1 * k11) + (s * alph2 * k12);
   f2 = (-y2 * F2) + (alph2 * k22) + (s * alph1 * k12);
   Lobj = (L1 * f1) + (L * f2) - (0.5 * k11 * L1^2) - (0.5 * k22 * L^2) - (s * k12 * L * L1);
   Hobj = (H1 * f1) + (H * f2) - (0.5 * k11 * H1^2) - (0.5 * k22 * H^2) - (s * k12 * H * H1);
   if (Lobj > Hobj + eps)
      a2 = L;
   elseif (Lobj < Hobj - eps)
      a2 = H;
   else
      a2 = alph2;
   end
end
%----------------------------------------------------------------------


%Calculate the change in a - if very small  return 0
%%!!!!!NOTE!! if eps not small enought it may stop the algorithm early!!!!!!
%----------------------------------------------------------------------
if (abs(a2-alph2)<eps*(a2+alph2+eps))
   retval=0;
   return
end
%----------------------------------------------------------------------


% computation on new a1pha1(a1)
%----------------------------------------------------------------------
a1 = alph1 + (s*(alph2-a2));
%----------------------------------------------------------------------


%Update weight vector to reflect change in a1 & a2, if linear SVM
%----------------------------------------------------------------------
if (strcmpi(kernel, 'linear') == 1)
   w = w + (y1 * (a1 - alph1) * X(i1,:)) + (y2 * (a2 - alph2) * X(i2,:));
```

```
end
%-----------------------------------------------------------------------


%Store a1 and a2 in the alpha array
%-----------------------------------------------------------------------
alpha(i1) = a1;
alpha(i2) = a2;
%-----------------------------------------------------------------------

%Update fcache[i] for i in I_0 using new Lagrange multipliers
%-----------------------------------------------------------------------
k = length(glob.I_0);
for i = 1 : k
   ki1 = K(glob.I_0(i),i1);
   ki2 = K(glob.I_0(i),i2);
   evals = evals + 2;
   glob.fcache(glob.I_0(i)) = glob.fcache(glob.I_0(i)) + (y1 * (a1 - alph1) * ki1) + (y2 * (a2 - alph2) * ki2);
end
%-----------------------------------------------------------------------


%The update below is simply achieved by keeping and updating information
%about alpha_i being at 0, C or in between them. Using this together with
%target[i] gives information as to which index set i belongs.
% Update I_0, I_1, I_2, I_3, I_4
%-----------------------------------------------------------------------
glob.I_0 = find(alpha > 0 & alpha < C);
glob.I_1 = find(alpha(glob.v_1) == 0);
glob.I_1 = glob.v_1(glob.I_1);
glob.I_2 = find(alpha(glob.v_2) == C);
glob.I_2 =  glob.v_2(glob.I_2);
glob.I_3 = find(alpha(glob.v_1) == C);
glob.I_3 =  glob.v_1(glob.I_3);
glob.I_4 = find(alpha(glob.v_2) == 0);
glob.I_4 = glob.v_2(glob.I_4);
%-----------------------------------------------------------------------




%Compute updated F values for i1 and i2
%-----------------------------------------------------------------------
glob.fcache(i1) = F1 + (y1 * (a1 - alph1) * k11) + (y2 * (a2 - alph2) * k12);
glob.fcache(i2) = F2 + (y1 * (a1 - alph1) * k12) + (y2 * (a2 - alph2) * k22);
%-----------------------------------------------------------------------


%Compute (i_low, b_low) and (i_up, b_up),
%using only i1, i2 and indices in I_0

%-----------------------------------------------------------------------
%--GIA TO i1 ----------------------------------------------------
v = find(glob.I_1==i1);
i1_in_I_1 = length(v);
v = find(glob.I_2==i1);
i1_in_I_2 = length(v);
v = find(glob.I_3==i1);
i1_in_I_3 = length(v);
v = find(glob.I_4==i1);
i1_in_I_4 = length(v);

%%----------------------------------------------------------------------
%--GIA TO i2 ----------------------------------------------------
v = find(glob.I_1==i2);
i2_in_I_1 = length(v);
v = find(glob.I_2==i2);
i2_in_I_2 = length(v);
v = find(glob.I_3==i2);
i2_in_I_3 = length(v);
v = find(glob.I_4==i2);
i2_in_I_4 = length(v);


% 1)First Compute i_low, i_up for I_0
%-----------------------------------------------------------------------

if size(glob.I_0)~=0 % Trying to run the smo mod1 for the alult datasets I diskovered that for small values of C there was this problem.
```

```
    [glob.b_up glob.i_up] = min(glob.fcache(glob.I_0));
    glob.i_up=glob.I_0(glob.i_up);
    if size(glob.i_up)~=1
        glob.i_up = glob.i_up(1);
    end
    [glob.b_low glob.i_low] = max(glob.fcache(glob.I_0));
    glob.i_low=glob.I_0(glob.i_low);
    if size(glob.i_low)~=1
        glob.i__low = glob.i_low(1);
    end
end



%2)Then check if i1 or i2 should replace i_up or i_low

%2a) For i1

if ( (glob.b_up>glob.fcache(i1)) && (i1_in_I_1+i1_in_I_2) )
    glob.b_up = glob.fcache(i1);
    glob.i_up = i1;
end
if ((glob.b_low <glob.fcache(i1))&&(i1_in_I_3+i1_in_I_4))
    glob.b_low = glob.fcache(i1);
    glob.i_low = i1;
end

%2b) For i2

if ((glob.b_up > glob.fcache(i2))&&(i2_in_I_1+i2_in_I_2))
    glob.b_up = glob.fcache(i2);
    glob.i_up = i2;
end
if ((glob.b_low <glob.fcache(i2))&&(i2_in_I_3+i2_in_I_4))
    glob.b_low = glob.fcache(i2);
    glob.i_low = i2;
end
retval = 1;
return
%-------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [retval, alpha, w, b, stp, evals, glob] =...
    examineExampleK(i2, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, steps, stp, evals, eps,K)

%drawnow; %%used to give Matlab the opportunity to examine any pending ctrl+C (while in deep loops)
if (get(0,'PointerLocation')==[1 1])
    %disp('Press <ctrl+c> to stop or any key to interrupt execution temporarily ...');
    %pause;
    %disp('Type "return" to carry on');
    keyboard;
end

retval = 0;
[n D] = size(X);
y2 = Y(i2);
alph2 = alpha(i2);
v = find(glob.I_0==i2);
i2_in_I_0 = length(v);
if (i2_in_I_0 > 0)
    F2 = glob.fcache(i2);
else
    ki2 = K(:,i2);
    evals = evals + n;
    F2 = -y2 + (ki2' * (Y .* alpha));
    glob.fcache(i2) = F2;
end


%Update (b_low, i_low) or (b_up, i_up) using (F2,%i2)
%-------------------------------------------------------------------------
v = find(glob.I_1==i2);
```

```
i2_in_I_1 = length(v);
v = find(glob.I_2==i2);
i2_in_I_2 = length(v);
v = find(glob.I_3==i2);
i2_in_I_3 = length(v);
v = find(glob.I_4==i2);
i2_in_I_4 = length(v);

if ((i2_in_I_1 + i2_in_I_2 > 0) && (F2 < glob.b_up))
   glob.b_up = F2;
   glob.i_up = i2;
elseif ((i2_in_I_3 + i2_in_I_4 > 0) && (F2 > glob.b_low))
   glob.b_low = F2;
   glob.i_low = i2;
end
%-----------------------------------------------------------------


%Chech optimality using current b_low and b_up and, if
%violated, find an index i1 to do joint optimization with i2
%-----------------------------------------------------------------
optimality = 1;
if ((i2_in_I_0 + i2_in_I_1 + i2_in_I_2) > 0)
   if ( (glob.b_low - F2) > (2 * tol) )
      optimality = 0;
      i1 = glob.i_low;
   end
end
if ((i2_in_I_0 + i2_in_I_3 + i2_in_I_4) > 0)
   if ((F2 - glob.b_up) > (2 * tol))
      optimality = 0;
      i1 = glob.i_up;
   end
end
if (optimality == 1)
   retval = 0;
   return;
end
%-----------------------------------------------------------------



%For i2 in I_0 choose the better i1
%-----------------------------------------------------------------

if (i2_in_I_0 > 0)
   if ((glob.b_low - F2) > (F2 - glob.b_up))
      i1 = glob.i_low;
   else
      i1 = glob.i_up;
   end
end
%-----------------------------------------------------------------



stp = stp + 1;

[retval, alpha, w, b, evals, glob] =...
   takeStepK(i1, i2, glob, alpha, w, b, X, Y, kernel, kpar1, kpar2, C, tol, evals, eps,K);
```

Функция svcplot_book.

```
function  svcplot_book(X,Y,ker,kpar1,kpar2,alpha,bias,aspect,mag,xaxis,yaxis,input)


global figt4

 color_shade = 1;

gridcellsX = 50;

gridcellsY = 50;

marg = 0.1;


if (nargin < 7 | nargin > 12)
```

```
    help svcplot
  else
    epsilon = 1e-5;
    if (nargin < 12) input = zeros(1,size(X,2));, end
    if (nargin < 11) yaxis = 2;, end
    if (nargin < 10) xaxis = 1;, end
    if (nargin < 9) mag = 0.1;, end
    if (nargin < 8) aspect = 0;, end


    xmin = min(X(:,xaxis));, xmax = max(X(:,xaxis));
    ymin = min(X(:,yaxis));, ymax = max(X(:,yaxis));
    xa = (xmax - xmin);, ya = (ymax - ymin);
    if (~aspect)
      if (0.75*abs(xa) < abs(ya))
        offadd = marg*(ya*4/3 - xa);,
        xmin = xmin - offadd - mag*marg*ya;, xmax = xmax + offadd + mag*marg*ya;
        ymin = ymin - mag*marg*ya;, ymax = ymax + mag*marg*ya;
      else
        offadd = marg*(xa*3/4 - ya);,
        xmin = xmin - mag*marg*xa;, xmax = xmax + mag*marg*xa;
        ymin = ymin - offadd - mag*marg*xa;, ymax = ymax + offadd + mag*marg*xa;
      end
    else
      xmin = xmin - mag*marg*xa;, xmax = xmax + mag*marg*xa;
      ymin = ymin - mag*marg*ya;, ymax = ymax + mag*marg*ya;
    end

    alpha_min=min(alpha);
    alpha_max=max(alpha);
    alpha_threshold = (alpha_max - alpha_min) * 0.01;
    alpha_threshold = alpha_threshold + alpha_min;

    [x,y] =  meshgrid(xmin:(xmax-xmin)/gridcellsX:xmax,ymin:(ymax-ymin)/gridcellsY:ymax);
    z = bias*ones(size(x));
    wh = waitbar(0,'Plotting...');
    for x1 = 1 : size(x,1)
      for y1 = 1 : size(x,2)
        input(xaxis) = x(x1,y1);, input(yaxis) = y(x1,y1);
        for i = 1 : length(Y)
          if (abs(alpha(i)) >= 0)
            z(x1,y1) = z(x1,y1) + Y(i)*alpha(i)*CalcKernel(input,X(i,:),ker,kpar1,kpar2);
          end
        end
      end

      waitbar((x1)/size(x,1)) ;
      drawnow
    end

    close(wh)

    figure(figt4);
```

```
fh = gcf;


set(gca,'XLim',[xmin xmax],'YLim',[ymin ymax]);
set(gca,'TickDir', 'in');
set(gca, 'XTick', [floor(xmin):ceil(xmax)]);
%       set(gca, 'XTickLabel', []); %Null list => Does not print Tick labels
set(gca, 'YTick', [floor(ymin):ceil(ymax)]);
%       set(gca, 'YTickLabel', []); %Null list => Does not print Tick labels
set(gca,'Box', 'on');
set(gca,'DataAspectRatio',[1 1 1 ]);
%eliminate borders of figure
old_gca_units = get(gca,'Units');
set(gca,'Units','Normalized');
set(gca,'Position',...
   [0.0 0.0 1.0 1.0]);
set(gca,'Units',old_gca_units);

l = (-min(min(z)) + max(max(z)))/2.0;
if (color_shade == 1)
   sp = pcolor(x,y,z);
   shading interp %has bug and does not interpolate correctly last column
   %shading flat
   set(sp,'LineStyle','none');
   set(gca,'Clim',[-1, 1])
   set(gca,'Position',[0 0 1 1])
   %       axis off
   load cmap
   colormap(colmap)
   %colormap(gray)
else
   whitebg('w')
end
hold on
for i = 1:size(Y)
   if (Y(i) == 1)
      if (color_shade == 1)
         plot(X(i,xaxis),X(i,yaxis),'rx','LineWidth',2) % Class A
      else
         plot(X(i,xaxis),X(i,yaxis),'x','LineWidth',1, 'MarkerSize', 4, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'none') % Class A
      end
   else
      if (color_shade == 1)
         plot(X(i,xaxis),X(i,yaxis),'bx','LineWidth',2) % Class B
      else
         %plot(X(i,xaxis),X(i,yaxis),'*','LineWidth',1, 'MarkerSize', 2, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'k') % Class B
         plot(X(i,xaxis),X(i,yaxis),'x','LineWidth',1, 'MarkerSize', 3, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'none') % Class B
      end
   end
   if (abs(alpha(i)) > alpha_threshold)
      if (color_shade == 1)
         plot(X(i,xaxis),X(i,yaxis),'ko','LineWidth',1, 'MarkerSize', 6) % Support Vector
      else
```

```
                plot(X(i,xaxis),X(i,yaxis),'x','LineWidth',1, 'MarkerSize', 5, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'none') % Support
Vector
            end
          end
        end
        % Plot Boundary contour

        hold on
        if (color_shade == 1)
          contour(x,y,z,[0 0],'k')
          %     contour(x,y,z,[-0.5 -0.5],'b')
          contour(x,y,z,[-1 -1],'b-.')
          contour(x,y,z,[1 1],'r-.')
          %     contour(x,y,z,[0.5 0.5],'b')
        else
          zones = 1; %how many zones to be present in [0,1]
          steps = [0 : 1/zones : 1];
          for j = 1 : zones + 1
            if ((mod(j,2))==1),
              clsp = 'k-';
            elseif ((mod(j,2))==0),
              clsp = 'k:';
            end
            if (steps(j) == 0)
              contour(x,y,z,[steps(j) steps(j)],clsp, 'LineWidth', 2)
            else
              contour(x,y,z,[steps(j) steps(j)],clsp, 'LineWidth', 1)
              contour(x,y,z,[-steps(j) -steps(j)],clsp, 'LineWidth', 1)
            end
          end
        end
        hold off
      end
```

Функция CalcKernel.

```
      function k = CalcKernel(u, v, ker, kpar1, kpar2)

      if (nargin < 3)
        error('CalcKernel needs at least 3 arguments')
      end
      if (nargin < 5)
        kpar2 = 0;
      end
      if (nargin < 4)
        kpar1 = 0;
      end

      [r1 c1] = size(u);
      [r2 c2] = size(v);
      if (r1 < 1 || r2 ~= 1)
        error('CalcKernel expect u=column of row-vectors and v a row-vector')
      end
```

```
if (c1 ~= c2)
   error('CalcKernel needs both x1 and x2 to have same num of columns')
end

switch lower(ker)
   case 'linear'
      k = u*v';
   case 'poly'
      k = (u*v' + kpar1).^kpar2;
   case 'rbf'
      k = zeros(r1,1);
      for i = 1 : r1
         k(i) = exp(-(u(i,:)-v)*(u(i,:)-v)'/(2*kpar1^2));
      end
   case 'erbf'
      k = zeros(r1,1);
      for i = 1 : r1
         k(i) = exp(-sqrt((u(i,:)-v)*(u(i,:)-v)')/(2*kpar1^2));
      end
   case 'sigmoid'
      k = zeros(r1,1);
      for i = 1 : r1
         k(i) = tanh(kpar1*u(i,:)*v'/length(u(i,:)) + kpar2);
      end
   case 'fourier'
      k = zeros(r1,1);
      for i = 1 : r1
         z = sin(kpar1 + 1/2)*2*ones(length(u(i,:)),1);
         j = find(u(i,:)-v);
         z(j) = sin(kpar1 + 1/2)*(u(i,j)-v(j))./sin((u(i,j)-v(j))/2);
         k(i) = prod(z);
      end
   case 'spline'
      k = zeros(r1,1);
      for i = 1 : r1
         z = 1 + u(i,:).*v + u(i,:).*v.*min(u(i,:),v) - ((u(i,:)+v)/2).*(min(u(i,:),v)).^2 + (1/3)*(min(u(i,:),v)).^3;
         k(i) = prod(z);
      end
   case {'curvspline','anova'}
      k = zeros(r1,1);
      for i = 1 : r1
         z = 1 + u(i,:).*v + (1/2)*u(i,:).*v.*min(u(i,:),v) - (1/6)*(min(u(i,:),v)).^3;
         k(i) = prod(z);
      end

      % - sum(u.*v) - 1;
      %       z = 1 + u.*v + (1/2)*u.*v.*min(u,v) - (1/6)*(min(u,v)).^3;
      %       k = prod(z);
      %       z = (1/2)*u.*v.*min(u,v) - (1/6)*(min(u,v)).^3;
      %       k = prod(z);

   case 'bspline'
      k = zeros(r1,1);
```

```
            for i = 1 : r1
               z = 0;
               for r = 0: 2*(kpar1+1)
                  z = z + (-1)^r*binomial(2*(kpar1+1),r)*(max(0,u(i,:)-v + kpar1+1 - r)).^(2*kpar1 + 1);
               end
               k(i) = prod(z);
            end
         case 'anovaspline1'
            k = zeros(r1,1);
            for i = 1 : r1
               z = 1 + u(i,:).*v + u(i,:).*v.*min(u(i,:),v) - ((u(i,:)+v)/2).*(min(u(i,:),v)).^2 + (1/3)*(min(u(i,:),v)).^3;
               k(i) = prod(z);
            end
         case 'anovaspline2'
            k = zeros(r1,1);
            for i = 1 : r1
               z = 1 + u(i,:).*v + (u(i,:).*v).^2 + (u(i,:).*v).^2.*min(u(i,:),v) - u(i,:).*v.*(u(i,:)+v).*(min(u(i,:),v)).^2 + (1/3)*(u(i,:).^2 +
4*u(i,:).*v + v.^2).*(min(u(i,:),v)).^3 - (1/2)*(u(i,:)+v).*(min(u(i,:),v)).^4 + (1/5)*(min(u(i,:),v)).^5;
               k(i) = prod(z);
            end
         case 'anovaspline3'
            k = zeros(r1,1);
            for i = 1 : r1
               z = 1 + u(i,:).*v + (u(i,:).*v).^2 + (u(i,:).*v).^3 + (u(i,:).*v).^3.*min(u(i,:),v) - (3/2)*(u(i,:).*v).^2.*(u(i,:)+v).*(min(u(i,:),v)).^2 +
u(i,:).*v.*(u(i,:).^2 + 3*u(i,:).*v + v.^2).*(min(u(i,:),v)).^3 - (1/4)*(u(i,:).^3 + 9*u(i,:).^2.*v + 9*u(i,:).*v.^2 + v.^3).*(min(u(i,:),v)).^4 +
(3/5)*(u(i,:).^2 + 3*u(i,:).*v + v.^2).*(min(u(i,:),v)).^5 - (1/2)*(u(i,:)+v).*(min(u(i,:),v)).^6 + (1/7)*(min(u(i,:),v)).^7;
               k(i) = prod(z);
            end
         case 'anovabspline'
            k = zeros(r1,1);
            for i = 1 : r1
               z = 0;
               for r = 0: 2*(kpar1+1)
                  z = z + (-1)^r*binomial(2*(kpar1+1),r)*(max(0,u(i,:)-v + kpar1+1 - r)).^(2*kpar1 + 1);
               end
               k(i) = prod(1 + z);
            end
         otherwise
            %k = u*v'; %linear (identity kernel)
            fprintf('CalcKernel: wrong kernel ''%s''\n',ker);
      end
```

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Основная литература

1. Курейчик, В. В. Теория эволюционных вычислений / Курейчик В. В. - Москва : ФИЗМАТЛИТ, 2012. - 260 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9221-1390-8
2. Курейчик, В. М. Основы теории эволюционных вычислений : Научная монография / Курейчик В. М. - Ростов-на-Дону : Южный федеральный университет, 2010. - 224 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9275-0799-3
3. Курейчик, В. М. Поисковая адаптация. Теория и практика / Курейчик В. М. - Москва : ФИЗМАТЛИТ, 2006. - 270 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9221-0749-6

Дополнительная литература

1. Методические указания по организации самостоятельной работы по дисциплине «Нечёткая логика и нейронные сети» : для студентов направления 38.03.05 «Бизнес-информатика» (профиль «Электронный бизнес», «Информационная бизнес-аналитика»).Учебный план 2012 г. / сост. И. Ю. Глазкова ; ФГАОУ ВПО Сев.-Кав. федер. ун-т. - Ставрополь : СКФУ, 2015. - 20 с.
2. Барский, А. Б. Нейронные сети: распознавание, управление, принятие решений : [учеб. пособие] / А.Б. Барский. - М. : Финансы и статистика, 2004. - 176 с. : ил. - (Прикладные информационные технологии). - Библиогр.: с. 170-173. - ISBN 5-279-02757-X
3. Осовский, С. Нейронные сети для обработки информации / Станислав Осовский ; пер. с польск. И. Д. Рудинского. - М. : Финансы и статистика, 2004. - 343 с. : ил. - Библиогр.: с. 3330-339. - Предм. указ.: с. 340-343. - ISBN 83-7207-187-X. - ISBN 5-279-

02567-4

4. Осовский, С. Нейронные сети для обработки информации : [учеб.-справ. изд.] / С. Осовский ; пер. с польск. И.Д. Рудинского. - М. : Финансы и статистика, 2004. - 344 с. : ил. - Библиогр.: с. 330-339. - ISBN 5-279-02567-4

Методическая литература

1. Методические рекомендации по организации самостоятельной работы студентов по дисциплине "Нейронные сети" : Направление подготовки 01.04.02 - Прикладная математика и информатика. Профиль подготовки «Математическое моделирование». Квалификация выпускника - магистр. Очная форма обучения. Изучается в 3 семестре. Учебный план 2015 г. - Ставрополь : СКФУ, 2015. - 16 с.
2. Комашинский В.И.Нейронные сети и их применение в системах управления и связи/ Д.А.Смирнов. - М:Горячая линия-Телеком,2002. - 94с. - с88
3. Нейронные сети: история развития теории : учеб. пособие для вузов / под ред. А. И. Галушкина, Я. З. Цыпкина. - М. : ИПРЖР, 2001. - 840 с. - (Нейрокомпьютеры и их применение, Кн. 5). - Гриф: Рек. МО. - Прил.: с. 826-834. - ISBN 5-93108-007-4
4. Ширяев, В. И. Финансовые рынки. Нейронные сети, хаос и нелинейная динамика : [учеб. пособие] / В.И. Ширяев. - 3-е изд. - М. : КРАСАНД, 2010. - 232 с. : ил. - На учебнике гриф: Доп.УМО. - Библиогр.: с. 210-221. - ISBN 978-5-396-00273-9, экземпляров 1

Интернет-ресурсы

1. http://algolist.manual.ru/ai/ga/ga1.php
2. http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005
3. https://basegroup.ru/community/articles/ga-math
4. https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%

D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D 0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D 1%82%D0%BC

5. http://www.prodav.narod.ru/dsp/index.html - Давыдов А.В. Цифровая обработка сигналов. Тематические лекции: Учебное пособие в электронной форме. – Екатеринбург, УГГУ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт информационных технологий и телекоммуникаций

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
К САМОСТОЯТЕЛЬНОЙ РАБОТЕ СТУДЕНТОВ
по дисциплине
«Основы распознавания образов»
направления
09.03.02 «Информационные системы и технологии»
профиль подготовки «Информационные системы и технологии обработки цифрового контента»
Бакалавриат

Пятигорск, 2024 г.

1. Цель и задачи освоения дисциплины

Цель освоения дисциплины «Основы распознавания образов» - формирование профессиональных компетенций будущего магистра по направлению подготовки 09.03.02 – Информационные системы и технологии и получение знаний, практических навыков и умений, используемых при разработке сложных систем и управлении их жизненным циклом. Задачей дисциплины является не только знакомство студентов с основными понятиями системной инженерии, международными стандартами и практиками, но и формирование системного подхода к созданию сложных систем. Целью курса является также рассмотрение перспектив развития системного подхода к созданию сложных проектов в нашей стране.

2. Место дисциплины в структуре ОП магистратуры

Дисциплина относится к вариативной части Блока 1. Ее освоение происходит в 8 семестре.

3. Связь с предшествующими дисциплинами

Для успешного освоения дисциплины «Основы распознавания образов» студент должен знать основы математики и информатики.

4. Связь с последующими дисциплинами

Дисциплина «Основы распознавания образов» включена в учебный процесс для формирования у магистрантов системного подхода к стратегии проектирования сложных информационных систем, к определению целей проектирования, к выявлению заинтересованных сторон и их требований. Дисциплина способствует формированию у бакалавров комплексного представления о современных требованиях к созданию крупных проектов и тенденциях развития информационных систем. Изучение данной дисциплины является необходимой основой для изучения дисциплин «Информационные системы в науке и производстве», «Промышленный интернет», «Анализ, моделирование и оптимизация бизнес-процессов», а также при подготовке к государственной итоговой аттестации.

5. Компетенции обучающегося, формируемые в результате освоения дисциплины

5.1. Наименование компетенций

Код	Формулировка
ПК-5	Способность разрабатывать программное обеспечение (ПО), включая проектирование, отладку, проверку работоспособности и модификацию ПО
ПК-13	способность адаптировать и модифицировать специализированное программное обеспечение, методы и алгоритмы систем искусственного интеллекта и машинного обучения в профессиональной деятельности

6. Организационно-методические рекомендации по освоению дисциплины

Самостоятельная работа студентов является важнейшим условием формирования научного способа познания. Она проводится накануне каждого лабораторного занятия и включает изучение необходимого для выполнения лабораторной работы теоретического материала, а также подготовку отчета по выполненным на лабораторном и практическом занятии заданиям.

Подготовленный материал оформляется в виде отчета. Самостоятельные занятия (СЗ) являются одной из активных форм обучения.

Самостоятельные занятия по дисциплине «Основы распознавания образов» имеют целью:

- закрепить и углубить знания, полученные студентами на лекциях и в процессе лабораторных занятий;
- привить практические навыки при проведении анализа и разработки сложных систем.

Предлагаемые методические рекомендации содержат информацию для студентов, необходимую при подготовке и проведении лабораторных и практических занятий по дисциплине «Основы распознавания образов».

7. Методические указания по выполнению самостоятельной работы студентов

Методические указания должны включать следующие разделы:

- цель работы;
- задание, которое должно быть выполнено студентом в результате проведения самостоятельной работы;
- варианты индивидуальных заданий;
- основные теоретические положения, необходимые для выполнения задания, они должны быть краткими и содержать ссылки на литературу, в которой эти положения изложены в объеме, достаточном для выполнения самостоятельной работы;
- этапы выполнения задания с указанием конкретных сроков выполнения каждого из этапов и всего задания в целом;
- требования к оформлению графической и текстовой части самостоятельной работы; пример выполнения одного из вариантов задания и оформления отчета;
- библиографический список использованных источников.

8. Содержание самостоятельной работы

Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы представлено в таблице 2.

Таблица 2 – Наименование разделов и тем для самостоятельной работы

№	Раздел (тема) дисциплины	Реализуемые компетенции	Контактная работа обучающихся с преподавателем, часов				
			Лекции	Практические занятия	Лабораторные работы	Групповые консультации	
8 семестр							
1	Основы байесовских методов классификации		2.00		2.00		
2	Классификаторы, основанные на оценки функции оптимизации		4.0		2.0		

3	Создание и обучение нейронной сети на языке высокого уровня среды Matlab		4.00		4.00		
4	Классификаторы, основанные на оценки функции оптимизации. Машины опорных векторов и нейронные сети прямого распространения		4.00		2.00		
5	Пример создания и обучения нейронных сетей для задач классификации в среде Matlab		4.00		6.00		
6	Пример создания и обучения нейронных сетей для задач регрессии в среде Matlab		2.00		4.00		
	ИТОГО за 8 семестр		20.00		20.00		68.00

Наименование тем дисциплины, цель, форма контроля, задания, требования к оформлению, перечень литературных источников представлен в таблице 3.

Таблица 3 – Наименование тем дисциплины, цель, форма контроля, задания и требования к оформлению самостоятельной работы

Название темы	**Цель**	**Форма контроля СРС**	**Задания для СРС**	**Требования к оформлению результатов СРС**	**Рекомендуемая литература**
Подготовка к лабораторным занятиям	Изучить UML	Индивидуальное собеседование	Сформулировать ответы на контрольные вопросы к лабораторным работам и практическим занятиям	Письменно ответить на контрольные вопросы к лабораторным работам	**Основная литература** 1. Курейчик, В. В. Теория эволюционных вычислений / Курейчик В. В. - Москва : ФИЗМАТЛИТ, 2012. - 260 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9221-1390-8 2. Курейчик, В. М. Основы теории эволюционных вычислений : Научная монография / Курейчик В. М. - Ростов-на-Дону : Южный федеральный
Изучение средств Matlab для работы с ГА	Изучить подходы, применяемые в системной инженерии для		Сформулировать и письменно ответить на вопрос	Краткий конспект теоретического материала и	

	управлени я жизненны м циклом		ы для контро ля владен ия компет ен циями данног о раздела	ответы на вопрос ы для контро ля владен ия компет ен циями	университет, 2010. - 224 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978- 5-9275-0799-3 3. Курейчик, В. М. Поисковая

Изучение библиотек С++ для работы с ГА	Скрещивание, селекция, ранжирование, создание популяции		программы	Краткий конспект теоретического материала и ответы на вопросы для контроля владения компетенциями	адаптация. Теория и практика / Курейчик В. М. - Москва : ФИЗМАТЛИТ, 2016. - 270 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9221- 0749-6 **Дополнительная литература** 4. Методические указания по организации самостоятельной работы по дисциплине «Нечёткая логика и нейронные сети» : для студентов направления 38.03.05 «Бизнес-информатика» (профиль «Электронный бизнес», «Информационная бизнес- аналитика»).Учебный план 2012 г. / сост. И. Ю. Глазкова ; ФГАОУ ВПО Сев.-Кав. федер. ун-т. - Ставрополь : СКФУ, 2015.- 30 с. 5. Барский, А. Б. Нейронные сети: распознавание, управление, принятие решений : [учеб. пособие] / А.Б. Барский. - М. : Финансы и статистика, 2004. - 176 с. : ил. - (Прикладные информационные технологии). - Библиогр.: с. 170-173. - ISBN 5-279-02757 6. Осовский, С. Нейронные сети для обработки информации / Станислав Осовский ; пер. с польск. И. Д. Рудинского. - М. : Финансы и статистика, 2004. - 343 с. : ил. - Библиогр.: с. 3330-339. - Предм. указ.: с. 340-343. - ISBN 83-7207- 187-X. - ISBN 5-279-02567-4 7. Осовский, С. Нейронные сети для обработки информации :
Методы проектирования информационных систем	Изучить медоды проектирования ИС				
Стандартиза ция, качество и надежность программного обеспечения (ПО)	Изучить стандарты, определяю щие качество и надеж-ность ПО				

					[учеб.-справ. изд.] / С. Осовский ; пер. с польск. И.Д. Рудинского. - М. : Финансы и статистика, 2004. - 344 с. : ил. - Библиогр.: с. 330-339. - ISBN 5-279-02567-4

					Методическая литература 8. Методические рекомендации по организации самостоятельной работы студентов по дисциплине "Нейронные сети" : Направление подготовки 01.04.02 - Прикладная математика и информатика. Профиль подготовки «Математическое моделирование». Квалификация выпускника - магистр. Очная форма обучения. Изучается в 3 семестре. Учебный план 2015 г. 9. Комашинский В.И.Нейронные сети и их применение в системах управления и связи/ Д.А.Смирнов. - М:Горячая линия-Телеком,2002. - 94с. – с. 88 10. Нейронные сети: история развития теории : учеб. пособие для вузов / под ред. А. И. Галушкина, Я. З. Цыпкина. - М. : ИПРЖР, 2001. - 840 с. - (Нейрокомпьютеры и их применение, Кн. 5). - Гриф: Рек. МО. - Прил.: с. 826-834. – ISBN 5-93108-007-4 **11.** Ширяев, В. И. Финансовые рынки. Нейронные сети, хаос и нелинейная динамика : [учеб. пособие] / В.И. Ширяев. - 3-е изд. - М. : КРАСАНД, 2010. - 232 с. : ил. - На учебнике гриф: Доп.УМО. - Библиогр.: с. 210-221. - ISBN 978-5- 396-00273-9, экземпляров 1 **Интернет-ресурсы** 12. http://algolist.manual.ru/ai/ga/ ga1.php

					13. http://rain.ifmo.ru/cat/ view.ph p/theory/unsorted/genetic-2005 14. https://basegroup.ru/ communi ty/articles/ga-math

8.1. Примерная тематика заданий для самостоятельной работы студентов

Примерная тематика заданий:

1. Классификация практик системной инженерии. Основные группы процессов. Стандарты по разработке программного обеспечения, по управлению качеством и надежностью.

2. Принципы моделирования деятельности организации. Основные типы методологий проектирования информационных систем. Методология SADT. Методологии серии IDEF. Другие методологии.

3. Документирование программных продуктов.

4. Разработка технического задания.

10. План-график выполнения СРС

№№	Название темы	Срок сдачи результатов, неделя
	8 семестр	
1	Различные методы распознавания образов	3
2	Экспорт кода Matlab в приложение, написанное на языке высокого уровня.	1 3

11. Организация контроля знаний студентов

11. 1 Формы контроля знаний студентов

Контроль и оценка знаний, умений и навыков студентов осуществляется на лабораторных и практических занятиях, консультациях, при сдаче экзамена. В ходе контроля знаний преподаватель оценивает понимание студентом содержания дисциплины «Основы распознавания образов», его способность анализировать состояние информационных систем и процессов.

Контроль знаний студентов может осуществляться в следующих формах:

- текущий контроль знаний;

- итоговый контроль знаний. Текущий контроль знаний студентов имеет целью: дать оценку работы каждого студента по усвоению им учебного материала, выявить недостатки в его подготовке и оказать практическую помощь в их устранении.

Основными формами текущего контроля знаний студентов являются:

- устный контрольный опрос;

- защита лабораторной работы;

- проверка конспектов лекций;

-проверка конспектов по теме, вынесенной на самостоятельное изучение.

Устный контрольный опрос студентов проводится на лекциях (и лабораторных занятиях). По его результатам преподаватель оценивает качество подготовки студента к занятию.

На лабораторных занятиях знания и практические навыки студентов оцениваются по 5-балльной системе. Полученные оценки выставляются в журнале.

При проверке конспектов дается анализ качества их ведения. Отмечаются допущенные ошибки, в рецензии преподавателя оценивается качество конспектирования учебного материала, даются рекомендации по улучшению качества конспектирования изучаемого материала.

11.2. Рекомендации по подготовке к экзамену Подготовка к экзамену начинается с начала изучения дисциплины. Необходимо посещать все виды занятий. Экзамен, как итоговый контроль знаний студентов имеет целью проверить и оценить учебную работу студентов, уровень полученных знаний и практических навыков.

Экзамен проводится во 2 семестре после защиты всех лабораторных работ и выполнения заданий к практическим занятиям в объеме учебной программы.

12. Рекомендации по работе с литературой и источниками

Изучение литературы и источников необходимо начинать с прочтения соответствующих глав учебных изданий, учебных пособий или литературы, рекомендованной в качестве основной или дополнительной по дисциплине «Основы распознавания образов», которые прямо или косвенно относятся к изучаемой теме.

При изучении литературы и источников студенту рекомендуется вести краткий конспект. Однако не следует переписывать все содержание изучаемой темы, нужно выписывать лишь основные идеи и главные мысли. В отдельных случаях, когда встречаются важные определения, понятия, необходимый фактический материал и примеры, статистическая информация, имеющие отношение к изучаемой теме, необходимо выписать их в виде цитат с полным указанием библиографических источников.

Конспектирование рекомендуемой литературы и источников необходимо вести с распределением собранных материалов по отдельным главам и параграфам согласно учебно-тематическому плану. Необходимо выписывать все выходные данные по используемой литературе и источникам.

Основой технологии интенсификации обучения на платформе цифровых образовательных технологий являются учебно-иллюстрационные материалы (опорный конспект) по дисциплине «Системной инженерии».

Работа с учебно-иллюстрационными материалами имеет следующие этапы.

1. Изучение теоретических основ учебного материала в аудитории: изложение преподавателем изучаемого материала студентам с объяснением по опорному конспекту;
2. Самостоятельная работа: индивидуальная работа студентов по опорному конспекту; фронтальное закрепление по блокам опорного конспекта.
3. Первое повторение - воспроизведение содержания заданной темы опорного конспекта по памяти.
4. Устное проговаривание материала опорного конспекта – необходимый этап внешне речевой деятельности при усвоении учебного материала.
5. Второе повторение – взаимоопрос и взаимопомощь студентов друг другу.

Применение учебно-иллюстрационных материалов позволяет обобщить сложный по содержанию материал, активизировать мыслительную деятельность студентов.

Необходимо помнить, что главное для студента в самостоятельной работе с рекомендуемой литературой и источниками - это формирование своего индивидуального стиля, который может стать основой в будущей профессиональной деятельности.

13. Перечень рекомендуемой литературы

Основная литература

1. Курейчик, В. В. Теория эволюционных вычислений / Курейчик В. В. - Москва : ФИЗМАТЛИТ, 2012. - 260 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9221-1390-8
2. Курейчик, В. М. Основы теории эволюционных вычислений : Научная монография / Курейчик В. М. - Ростов-на-Дону : Южный федеральный университет, 2010. - 224 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9275-0799-3
3. Курейчик, В. М. Поисковая адаптация. Теория и практика / Курейчик В. М. - Москва : ФИЗМАТЛИТ, 2006. - 270 с. - Книга находится в базовой версии ЭБС

IPRbooks. - ISBN 978-5-9221-0749-6

Дополнительная литература

4. Методические указания по организации самостоятельной работы по дисциплине «Нечёткая логика и нейронные сети» : для студентов направления 38.03.05 «Бизнес-информатика» (профиль «Электронный бизнес», «Информационная бизнес-аналитика»).Учебный план 2012 г. / сост. И. Ю. Глазкова ; ФГАОУ ВПО Сев.-Кав. федер. ун-т. - Ставрополь : СКФУ, 2015. - 20 с.
5. Барский, А. Б. Нейронные сети: распознавание, управление, принятие решений : [учеб. пособие] / А.Б. Барский. - М. : Финансы и статистика, 2004. - 176 с. : ил. - (Прикладные информационные технологии). - Библиогр.: с. 170-173. - ISBN 5-279-02757-X
6. Осовский, С. Нейронные сети для обработки информации / Станислав Осовский ; пер. с польск. И. Д. Рудинского. - М. : Финансы и статистика, 2004. - 343 с. : ил. - Библиогр.: с. 3330-339. - Предм. указ.: с. 340-343. - ISBN 83-7207-187-X. - ISBN 5-279-02567-4
7. Осовский, С. Нейронные сети для обработки информации : [учеб.-справ. изд.] / С. Осовский ; пер. с польск. И.Д. Рудинского. - М. : Финансы и статистика, 2004. - 344 с. : ил. - Библиогр.: с. 330-339. - ISBN 5-279-02567-4

Методическая литература

8. Методические рекомендации по организации самостоятельной работы студентов по дисциплине "Нейронные сети" : Направление подготовки 01.04.02 - Прикладная математика и информатика. Профиль подготовки «Математическое моделирование». Квалификация выпускника - магистр. Очная форма обучения. Изучается в 3 семестре. Учебный план 2015 г. - Ставрополь : СКФУ, 2015. - 16 с.
9. Комашинский В.И.Нейронные сети и их применение в системах управления и связи/ Д.А.Смирнов. - М:Горячая линия-Телеком,2002. - 94с. - с88
10. Нейронные сети: история развития теории : учеб. пособие для вузов / под ред. А. И. Галушкина, Я. З. Цыпкина. - М. : ИПРЖР, 2001. - 840 с. - (Нейрокомпьютеры и их применение, Кн. 5). - Гриф: Рек. МО. - Прил.: с. 826-834. - ISBN 5-93108-007-4
11. Ширяев, В. И. Финансовые рынки. Нейронные сети, хаос и нелинейная динамика : [учеб. пособие] / В.И. Ширяев. - 3-е изд. - М. : КРАСАНД, 2010. - 232 с. : ил. - На учебнике гриф: Доп.УМО. - Библиогр.: с. 210-221. - ISBN 978-5-396-00273-9, экземпляров 1

Интернет-ресурсы

12. http://algolist.manual.ru/ai/ga/ga1.php
13. http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005
14. https://basegroup.ru/community/articles/ga-math
15. https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0% BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC
16. http://www.prodav.narod.ru/dsp/index.html - Давыдов А.В. Цифровая обработка сигналов. Тематические лекции: Учебное пособие в электронной форме. – Екатеринбург, УГГУ