Министерство науки и высшего образования Российской Федерации
Филиал Южно-Уральского государственного университета
в г. Нижневартовске
Кафедра «Гуманитарные, естественно-научные и технические дисциплины»

Ю.А. Захарова

Современные финансовые технологии

Методические указания для выполнения практических работ по дисциплине «Современные финансовые технологии» для всех форм обучения и направлений подготовки

НИЖНЕВАРТОВСК
2025

ББК 32.818.1
С 56

Одобрено
редакционно-издательским советом филиала

Современные финансовые технологии: методические указания для выполнения практических работ по дисциплине «Современные финансовые технологии» для всех форм обучения и направлений подготовки /сост. Ю.А. Захарова. – Нижневартовск, 2025.– 44 с.

Практические работы по разделу «Искусственный интеллект и большие данные. Основные понятия нейронных сетей. Генерация и обучение НС в матричной лаборатории SciLab. Применение НС в цифровой экономике» составлены для седьмого семестра обучения всех форм обучения и направлений подготовки для формирования компетенций, предусмотренных РПД.

СОДЕРЖАНИЕ:

ПРАКТИЧЕСКИЕ РАБОТЫ ПО РАЗДЕЛУ «ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И БОЛЬШИЕ ДАННЫЕ. ОСНОВНЫЕ ПОНЯТИЯ НЕЙРОННЫХ СЕТЕЙ.

ГЕНЕРАЦИЯ И ОБУЧЕНИЕ НС В МАТРИЧНОЙ ЛАБОРАТОРИИ SCILAB. ПРИМЕНЕНИЕ НС В ЦИФРОВОЙ ЭКОНОМИКЕ»

Практическая работа - Введение в матричную лабораторию Scilab: основные элементы языка, работа с матрицами

Упражнение № 1 (Приоритет операторов) Каков результат вычисления следующих выражений (проверьте свои ответы с помощью Scilab)?

```
2 * 3 + 4
2 + 3 * 4
2 / 3 + 4
2 + 3 / 4
```

Упражнение № 2 (Скобки) Каков результат вычисления следующих выражений (проверьте свои ответы с помощью Scilab)?

```
2 * (3 + 4)
(2 + 3) * 4
(2 + 3) / 4
3 / (2 + 4)
```

Упражнение № 3 (Экспоненциальная запись чисел) Каков результат вычисления следующих выражений (проверьте свои ответы с помощью Scilab)?

```
1.23456789 d10
1.23456789 e10
1.23456789e -5
```

Упражнение № 4 (Функции) Каков результат вычисления следующих выражений (проверьте свои ответы с помощью Scilab)?

```
sqrt (4)
sqrt (9)
sqrt ( -1)
sqrt ( -2)
exp (1)
log ( exp (2))
exp ( log (2))
10^2
log10 (10^2)
10^ log10 (2)
sign (2)
sign ( -2)
sign (0)
```

Упражнение № 5 (Тригонометрические функции) Каков результат вычисления следующих выражений (проверьте свои ответы с помощью Scilab)?

```
cos (0)
sin (0)
cos ( %pi )
sin ( %pi )
cos ( %pi /4) - sin ( %pi /4)
```

Упражнение № 6 (Плюс 1) С помощью одной инструкции получите вектор (x_1 + 1, x_2 +1, x_3 + 1, x_4 + 1), если x равен

x = 1 : 4;

Упражнение № 7 (Векторизованное умножение) Получите вектор (x_1y_1, x_2y_2, x_3y_3, x_4y_4), если x и y равны

x = 1 : 4;

y = 5 : 8;

Упражнение № 8 (Вектор обратных величин) Получите вектор (1/ x_1 , 1/x_2 , 1/x_3 , 1/x_4), если x равен

x = 1 : 4;

Упражнение № 9 (Векторизованное деление) Получите вектор $\left(\frac{x_1}{y_1}, \frac{x_2}{y_2}, \frac{x_3}{y_3}, \frac{x_4}{y_4}\right)$, если x и y равны

x = 12 * (6 : 9);

y = 1 : 4;

Упражнение № 10 (Векторизованное возведение в степень) Получите вектор (x_1^2, x_2^2, x_3^2, x_4^2), если x = 1,2,3,4.

Упражнение № 11 (Применение функции к вектору) Получите вектор ($sin(x_1)$, $sin(x_2)$, . . . , $sin(x_{10})$) для 10 значений, равномерно распределенных на отрезке [0, π].

Упражнение № 12 (Векторизованные функции) Вычислите значения $y = f(x)$ функции f, определяемой уравнением

$$f(x) = \log10\ (r/10^x + 10^x)$$

если r = 2.220.10^{-16}, для 100 равноотстоящих значений из отрезка [−16,0].

Практическая работа – Операторы ветвления и цикла в Scilab

1. *Первое и второе задания вашего варианта выполнить с использованием цикла For.*
2. *Третье задание вашего варианта выполнить с использованием цикла While.*
3. *Четвертое и пятое задание выполните с использованием оператора «:».*

Вариант № 1

1. Написать программу подсчета суммы S первых N членов гармонического ряда:

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{N}$$

2. Напишите программу проверки тождества: $1+2+3+...+n=n*(n+1)/2$

3. Составить программу, вычисляющую сумму квадратов натуральных чисел, начиная с 1, до тех пор, пока, она не окажется число кратным 8 или 12.

4. Вычислить и вывести на экран значение функции:

$$y=\frac{a^4}{(a+x)^2}, при \quad -1\le x\le 1, \quad с \quad шагом \quad dx=0{,}2.$$

5. Вычислить значение функции F ($x\in[a;b]$ с шагом 0,5):

$$F=\begin{cases} 0, при & x\le 0 \\ x^2-x, при & 0<x\le 1 \\ x^2-\sin\pi x^2, при & x>1 \end{cases}$$

Вариант № 2

1. Написать программу подсчета произведения P первых K сомножителей:

$$\frac{1}{1}*\frac{1}{4}*\frac{1}{9}*...*\frac{1}{K^2}$$

2. Напишите программу проверки тождества: $1+3+5+...+(2*n-1)=n^2$

3. Составить программу, вычисляющую произведение натуральных чисел, начиная с 5, до тех пор, пока, она не превысит некоторое заданное число X.

4. Вычислить и вывести на экран значение функции:

$$y=\sqrt{\left|a^2+x^2\right|}, при \quad -2\le x\le 2, \quad с \quad шагом \quad dx=0{,}3.$$

5. Вычислить значение функции F ($x\in[a;b]$ с шагом 1):

$$F=\begin{cases} \cos x, & при \quad x<1 \\ \dfrac{1}{x}, & при \quad 1\le x\le 10 \\ \sqrt{x}, & при \quad x>10 \end{cases}$$

Вариант № 3

1. Написать программу подсчета суммы S первых N членов ряда:
$1+3^2+5^2+...+(2\cdot N-1)^2$
2. Напишите программу проверки тождества:
$1^2+2^2+3^2+...+n^2=n*(n+1)*(2*n+1)/6$
3. Вычислять сумму членов геометрической прогрессии до тех пор, пока она не окажется числом кратным 8: $b_1=2,\ \ q=4,\ \ b_n=b_{n-1}*q, S_n=S_{n-1}+b_n$

4. Вычислить и вывести на экран значение функции:
$y=b^2\cdot\cos x+\frac{x}{2}, при\ \ -1\le x\le 1,\ \ с\ \ шагом\ \ dx=0{,}2.$
5. Вычислить значение функции F ($x\in[a;b]$ с шагом 0,5):

$$F=\begin{cases} -x, & при\ \ x<1 \\ -1, & при\ \ 1\le x\le 3 \\ x-4, & при\ \ x>3 \end{cases}$$

Вариант № 4

1. Написать программу подсчета произведения P первых K сомножителей:
$\frac{1}{1}*\frac{1}{8}*\frac{1}{27}*...*\frac{1}{K^3}$
2. Напишите программу проверки тождества: $1^3+2^3+3^3+...+n^3=n^2*(n+1)^2/4$
3. Составить программу, вычисляющую произведение четных, натуральных чисел, начиная с 2, до тех пор, пока, она не окажется числом кратным 10 или 12.

4. Вычислить и вывести на экран значение функции:
$y=\sin x\cdot\cos x, при\ \ 0\le x\le 1,\ \ с\ \ шагом\ \ dx=0{,}1.$

5. Вычислить значение функции F ($x\in[a;b]$ с шагом 2):

$$F=\begin{cases} 0, при\ \ x\le 0 \\ x, при\ \ 0<x\le 1 \\ x^4, в\ остальных\ \ случаях \end{cases}$$

Вариант № 5

1. Написать программу подсчета суммы S первых N членов ряда: $1+\frac{1}{4}+\frac{1}{9}+\frac{1}{16}+...+\frac{1}{N^2}$
2. Напишите программу проверки тождества: $1^2+3^2+5^2+...+(2*n-1)^2=n*(4*n^2-1)/3$

3. Составить программу, вычисляющую сумму натуральных чисел, начиная с 3, до тех пор, пока, она не превысит некоторое заданное число X.

4. Вычислить и вывести на экран значение функции:
$y=(a+x)^2+\frac{a^3}{(x+1)}, при\ \ 0\le x\le 3,\ \ с\ \ шагом\ \ dx=0{,}1.$
5. Вычислить значение функции F ($x\in[a;b]$ с шагом $\pi/10$):

$$F=\begin{cases} -1, & при \quad x<0 \\ 0, & при \quad x=0 \\ \sin x, & при \quad x>0 \end{cases}$$

Вариант № 6

1. Написать программу подсчета произведения P первых K сомножителей:
$\frac{1}{2^2} * \frac{1}{4^2} * \frac{1}{6^2} * ... * \frac{1}{(2 \cdot K)^2}$

2. Напишите программу проверки тождества: $(1+2+3+...+n)^2 = n^2 * (n+1)^2 / 4$

3. Вычислять сумму членов арифметической прогрессии до тех пор, пока она не превысит заданное число X: $a_1 = 1, \quad d = 2, \quad a_n = a_{n-1} + d, S_n = S_{n-1} + a_n$

4. Вычислить и вывести на экран значение функции:
$y = \frac{a^2}{(a + x^2)^2}, при \quad -3 \le x \le 0, \quad с \quad шагом \quad dx = 0{,}2.$

3. Вычислить значение функции F ($x \in [a;b]$ с шагом $\pi / 4$):

$$F=\begin{cases} -\frac{x^2}{2}, & при \quad x<-10 \\ |x|, & при \quad -10 \le x \le 10 \\ \sin x, & при \quad x>10 \end{cases}$$

Вариант № 7

1. Написать программу подсчета суммы S первых N членов ряда: $\frac{1}{3} + \frac{1}{6} + \frac{1}{9} + ... + \frac{1}{3 * N}$

2. Напишите программу проверки тождества: $1^2 + 3^2 + 5^2 + ... + (2 * n - 1)^2 = n * (4 * n^2 - 1) / 3$
3. Составить программу, вычисляющую сумму натуральных чисел, начиная с 1, до тех пор, пока, она не окажется числом, кратным 7 или 12.

4. Вычислить и вывести на экран значение функции:
$y = x \cdot e^{\frac{b^2}{1+x^2}}, при \quad 0 \le x \le 2, \quad с \quad шагом \quad dx = 0{,}2.$

5. Вычислить значение функции F ($x \in [a;b]$ с шагом $\pi / 4$):

$$F=\begin{cases} x, & при \quad x<-5 \\ |x+2|, & при \quad -5 \le x \le 5 \\ \sin x \cdot \cos^2 x, & при \quad x>5 \end{cases}$$

Вариант № 8

1. Написать программу подсчета произведения P первых K сомножителей:
$\frac{1}{1} * \frac{1}{3^2} * \frac{1}{5^2} * ... * \frac{1}{(2 \cdot K - 1)^2}$

2. Напишите программу проверки тождества: $1 + 2 + 3 + ... + n = n * (n+1) / 2$

3. Вычислять сумму членов геометрической прогрессии до тех пор, пока она не превысит заданное число X: $b_1 = 1,\ \ q = 2,\ \ b_n = b_{n-1} * q, S_n = S_{n-1} + b_n$

4. Вычислить и вывести на экран значение функции:
$y = x^2 + a \cdot \sin^2 x, при\ \ -1 \le x \le 4,\ \ с\ \ шагом\ \ dx = 0{,}2.$

5. Вычислить значение функции F ($x \in [a;b]$ с шагом $\pi / 4$):

$$F=\begin{cases} \dfrac{\sin x}{x}, & при \quad x > 0 \\ 1, & при \quad x = 0 \\ \dfrac{1}{x}, & при \quad x < 0 \end{cases}$$

Вариант № 9

1. Написать программу подсчета суммы S первых N членов ряда:
$1 + 3^3 + 5^3 + ... + (2 \cdot N - 1)^3$

2. Напишите программу проверки тождества: $1^3 + 2^3 + 3^3 + ... + n^3 = n^2 * (n+1)^2 / 4$

3. Составить программу, вычисляющую произведение нечетных, натуральных чисел, начиная с 1, до тех пор, пока, она не окажется числом кратным 5 или 9.

4. Вычислить и вывести на экран значение функции:

$$y = \frac{a^2}{a + \cos x^2}, при\ \ 0 \le x \le 1,\ \ с\ \ шагом\ \ dx = 0{,}1.$$

3. Вычислить значение функции F ($x \in [a;b]$ с шагом $\pi / 10$):

$$F=\begin{cases} 1 - \cos x, & при \quad x > 0 \\ 2\left(\dfrac{x}{2}\right)^2, & при \quad x = 0 \\ 1 - x, & при \quad x < 0 \end{cases}$$

Вариант № 10

1. Написать программу подсчета суммы S первых N членов ряда: $\frac{1}{1} + \frac{1}{8} + \frac{1}{27} + ... + \frac{1}{N^3}$

2. Напишите программу проверки тождества: $1 + 3 + 5 + ... + (2 * n - 1) = n^2$

3. Вычислять сумму членов арифметической прогрессии до тех пор, пока она не окажется числом, кратным 5: $a_1 = 2,\ \ d = 4,\ \ a_n = a_{n-1} + d, S_n = S_{n-1} + a_n$

4. Вычислить и вывести на экран значение функции:

$$y = \frac{\cos^2(ax)}{\sin x}, при\ \ 0{,}1 \le x \le 1,\ \ с\ \ шагом\ \ dx = 0{,}1.$$

3. Вычислить значение функции F ($x \in [a;b]$ с шагом 0.5):

$$F=\begin{cases} \ln^2 x, & при \quad x > 1 \\ x, & при \quad 0 \le x \le 1 \\ \dfrac{1}{x^2}, & при \quad x < 0 \end{cases}$$

Вариант № 11

1. Написать программу подсчета суммы S первых N членов гармонического ряда:

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{N}$$

2. Напишите программу проверки тождества: $1+2+3+...+n=n*(n+1)/2$

3. Составить программу, вычисляющую сумму квадратов натуральных чисел, начиная с 1, до тех пор, пока, она не окажется число кратным 8 или 12.

4. Вычислить и вывести на экран значение функции:

$$y=\frac{a^4}{(a+x)^2}, при \quad -1\le x\le 1, \quad с \quad шагом \quad dx=0{,}2.$$

5. Вычислить значение функции ($x\in[a;b]$ с шагом $\pi/10$):

$$F=\begin{cases} x, & при \quad x<-5 \\ |x+2|, & при \quad -5\le x\le 5 \\ \sin x\cdot\cos^2 x, & при \quad x>5 \end{cases}$$

Вариант № 12

1. Написать программу подсчета произведения P первых K сомножителей:

$$\frac{1}{1}*\frac{1}{8}*\frac{1}{27}*...*\frac{1}{K^3}$$

2. Напишите программу проверки тождества: $1^3+2^3+3^3+...+n^3=n^2*(n+1)^2/4$

3. Составить программу, вычисляющую произведение четных, натуральных чисел, начиная с 2, до тех пор, пока, она не окажется числом кратным 10 или 12.

4. Вычислить и вывести на экран значение функции:

$$y=\sin x\cdot\cos x, при \quad 0\le x\le 1, \quad с \quad шагом \quad dx=0{,}1.$$

5. Вычислить значение функции F ($x\in[a;b]$ с шагом $\pi/10$):

$$F=\begin{cases} \cos^2 x, & при \quad x<1 \\ \frac{1}{x}, & при \quad 1\le x\le 4 \\ \sqrt{|x-10|}, & при \quad x>4 \end{cases}$$

Практическая работа - Построение и форматирование графиков в в Scilab

Задание 1. Постройте график функции f(x) заданный в декартовых координатах, с использованием ранжированной переменной (диапазон x меняется от 1 до 10). Постройте график тремя способами используя примеры 1_1, 1_2 и 1_3. В примере 1_3 используется функция **subplot** для размещения нескольких графиков в одном графическом окне

№ варианта	**f(x)**
1	$\tan(x)$
2	$\sin(2*x)$
3	$\cos(2*x)$

Пример 1_1

```
clf();
x=1:0.1:10;                    // Инициализация
plot(x,sin(x),'r-.>')          // строит штрихпунктирную линию с
треугольниками, указывающими вправо, центрованными на каждой точке
xtitle("Название графика", "Подпись оси X", "Подпись оси Y");
```

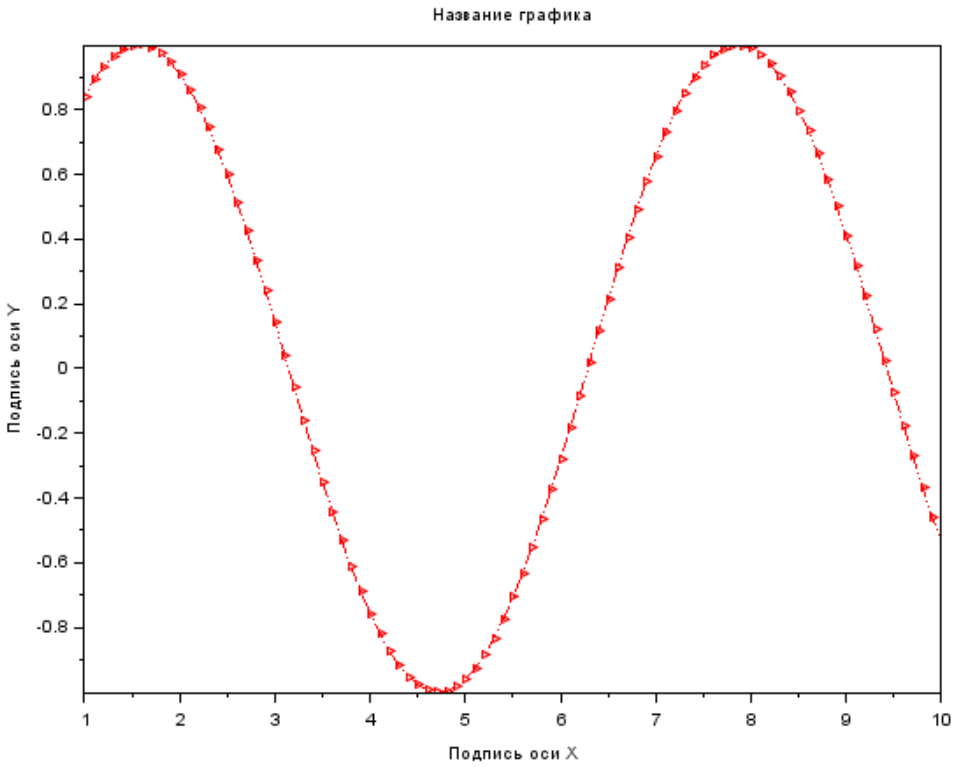


Рис 1_1. Пример 1_1

Пример 1_2

```
clf();
x=1:0.1:10;
// Если вы укажете маркер без стиля линии, то только маркеры будут
нарисованы
plot(x,sin(x),'d')
```

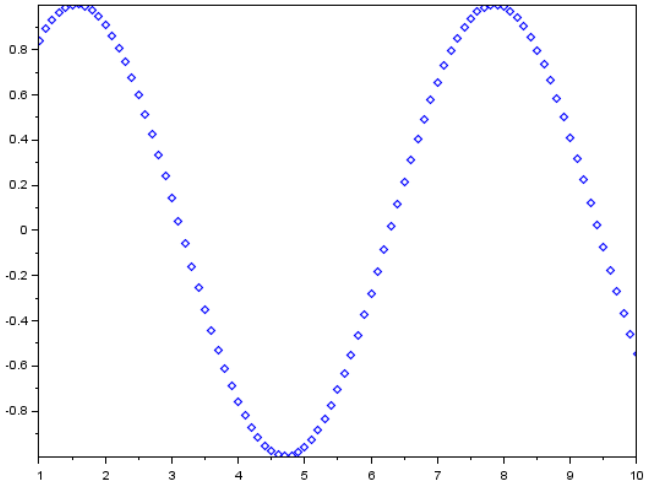


Рис 1_2. Пример 1_2

Пример 1_3

```
clf();
x = 1:0.5:10; // Инициализация
// Порядок информации о цвете, стили линии  или маркерах  не  имеет значения
// ОДНАКО информация  должна  быть однозначной
subplot(311); plot(x,sin(x),'.b-');
subplot(312); plot(x,sin(x),'b.-');
subplot(313); plot(x,sin(x),'b-.'); // Точка  принадлежит информации  о стиле линии (не маркера!)
```

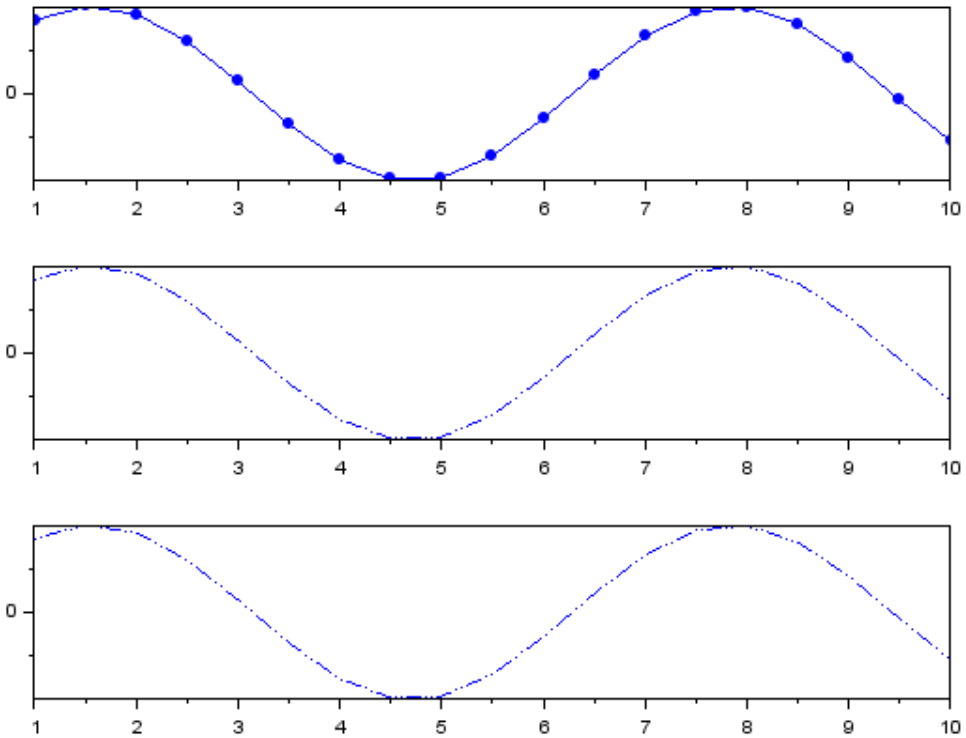


Рис 1_3. Пример 1_3

Задание 2. Постройте графики функций f(x) и g(x), заданные в декартовых координатах, с использованием ранжированной переменной $x \in [a, b]$. Постройте графики двумя способами используя примеры 3_1 и 3_2.

№ варианта	**f(x)**	**g(x)**	
1	$x+\cos^2(x)$	$x\cdot\cos^2(x)$	$x\in[-8\pi,8\pi]$
2	$x^2\cdot\sin(x)+x$	$x^2\cdot\cos^2(x)+x^2$	$x\in[-8\pi,8\pi]$
3	x^2-8x+1	$-2x^2+4x+1$	$x\in[-8,8]$

Пример 3_1. Постройте графики функций $f(x)=x+\sin(x)$, $g(x)=x\cdot\sin(x)$ заданные в декартовых координатах, с использованием ранжированной переменной $x\in[-8\pi,8\pi]$.

```
// графики  с различными видами линии (1 вариант)
clf();
x=-8*%pi:%pi/20:8*%pi;
plot(x,x+sin(x),'ro-',x,x .*cos(x),'c+');
xtitle("Название графика", "Подпись оси X", "Подпись оси Y");
```

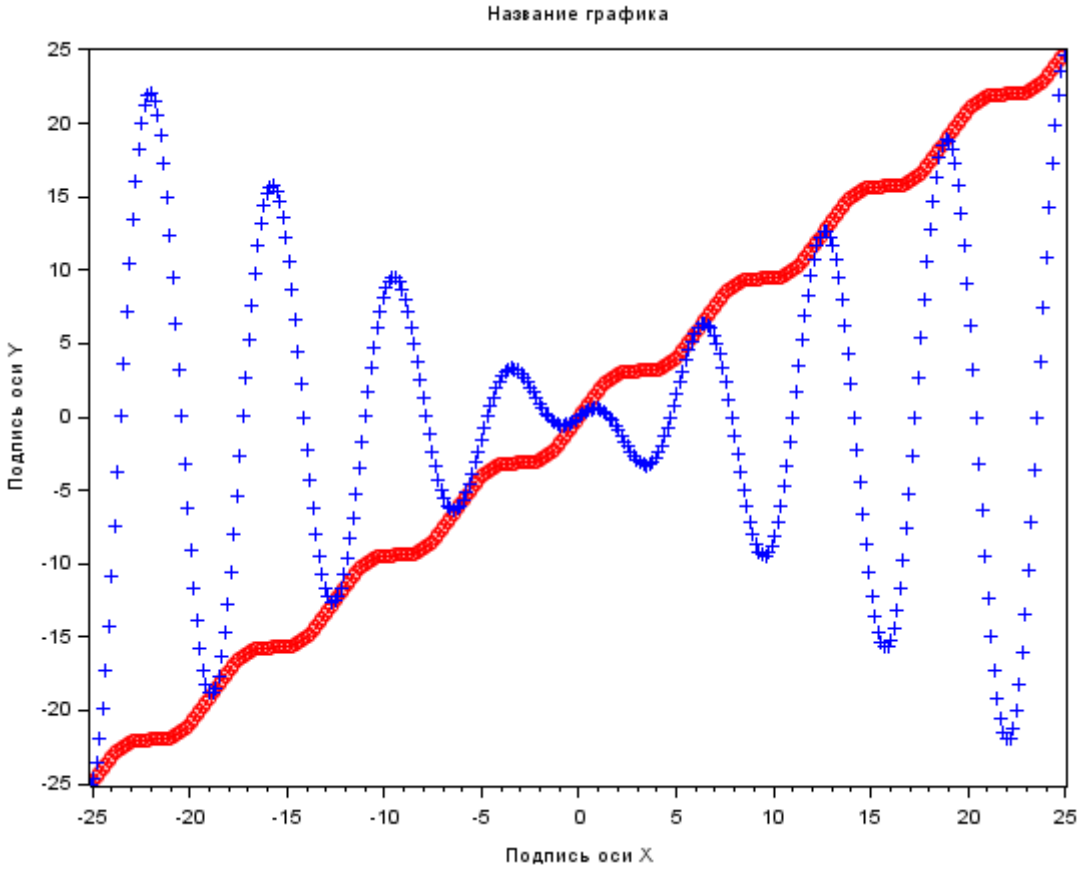


Рис 3_1. Пример 3_1

Пример 3_2.

Постройте графики функций $y1=\sin(t), y2=\cos(t), y3=|\sin(t)|$ заданные в декартовых координатах, с использованием ранжированной переменной $t\in[0,2\pi]$.

```
//множество графиков с различными видами линии (2 вариант)
clf();
t=0:%pi/20:2*%pi;
plot(t,sin(t),'ro-',t,cos(t),'c+',t,abs(sin(t)),'--m');
```

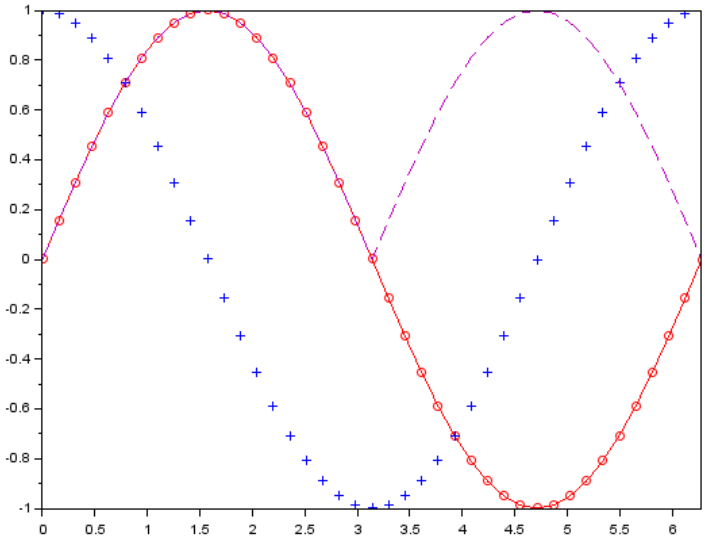


Рис 4. Пример 4

Задание 3. Выполните задание 1 с использованием функции **plot2d.** Постройте график функции f(x) заданный в декартовых координатах, с использованием ранжированной переменной (диапазон x меняется от 1 до $4 \cdot \pi$).

Пример 4

```
clf();
// вызов функции plot2d будет таким
x=1:0.1:4*%pi;
plot2d( x, ..                    // диапазон аргументов
        sin(x),      ..          // функция
        style = color('blue'), ..   // задает синий цвет линии
        leg = 'Функция синуса', ..  // задает надпись легенды
        strf='181')        // цифра 1 включает рамочки с подписями, в том числе легенду
       // цифра 8 — построитель размещает график по границам диапазона значений
                           // цифра 1 — оси рисуются слева
// Все остальное включается вручную через дескриптор
g=get('current_axes');                          // получаем дескриптор осей
g.title.text='Мой график';                      // делаем подпись графика
g.x_label.text='Ось абсцисс';           // делаем подпись для оси абсцисс
g.y_label.text='Ось ординат';           // делаем подпись для оси ординат
g.children(1).children.line_style=6;    // меняем стиль линии на штрих-пунктирный
g.grid=[0 0];                           // включаем сетку и делаем её черной
g.children(1).children.thickness=4;     // делаем линию чуть-чуть потолще
```

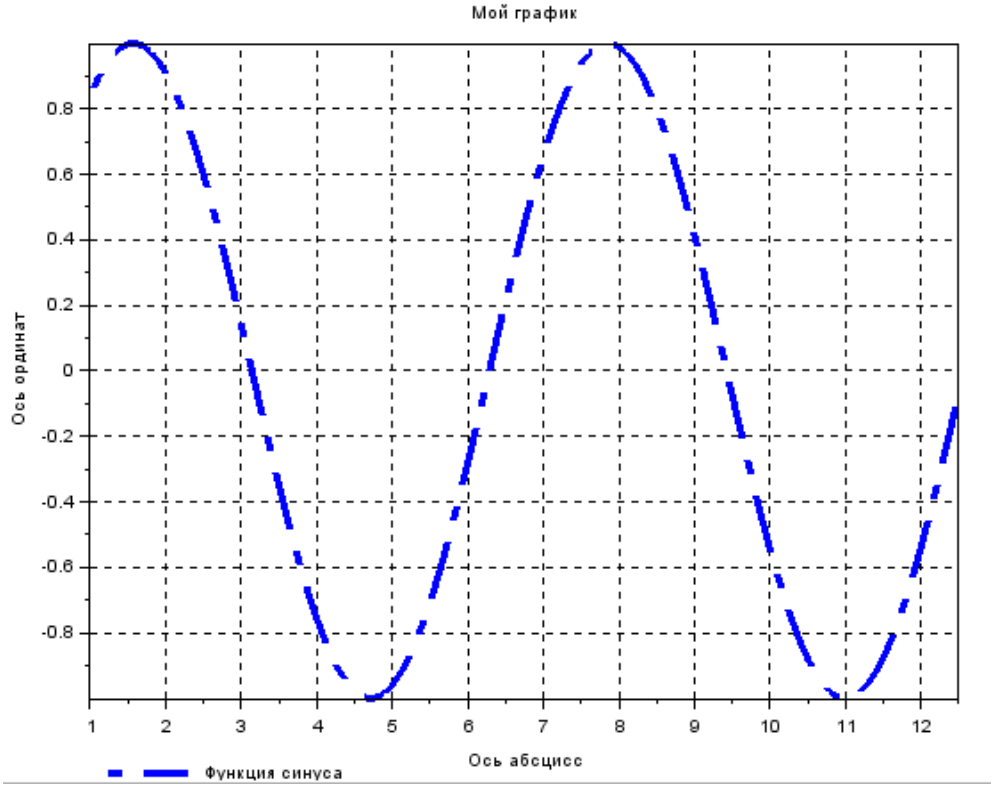


Рис 4. Пример 4

Задание 4. Изобразите кривую, заданную в полярных координатах (параметр φ – ранжированная переменная). Постройте графики двумя способами: первую кривую используя пример 4_1 и вторую кривую используя пример 4_2.

Вариант 1 – кривые под номерами 1 и 4; *Вариант 2* – кривые под номерами 2 и 5; *Вариант 3* – кривые под номерами 3 и 6.

№ кривой	ρ(φ)	
1. Улитка Паскаля	$5\cdot\cos(\varphi)+2$	$\varphi\in[0,2\pi]$
2. Семилепестковый цветок	$5\cdot\cos(7\cdot\varphi)+2$	$\varphi\in[0,2\pi]$
3. Логарифмическая спираль	$2\cdot e^{a\varphi}$	$\varphi\in[0,6\pi]$, $a=0.2$
4. Трехлепестковая роза	$10\cdot\sin(3\cdot\varphi)$	$\varphi\in[0,\pi]$
5. Лемниската	$2\sqrt{\cos(2\cdot\varphi)}$	$\varphi\in[0,2\pi]$
6. Кардиоида	$2\cdot(1+\cos(\varphi))$	$\varphi\in[0,2\pi]$

Пример 4_1. Изобразите кривую, заданную в полярных координатах, следующего вида: $\rho(\varphi)=2\cdot\sin(6\varphi), \varphi\in[0,2\pi]$.

```
t= 0:.01:2*%pi;
clf();
polarplot(t, 2.*sin(6.*t),style=1);
```

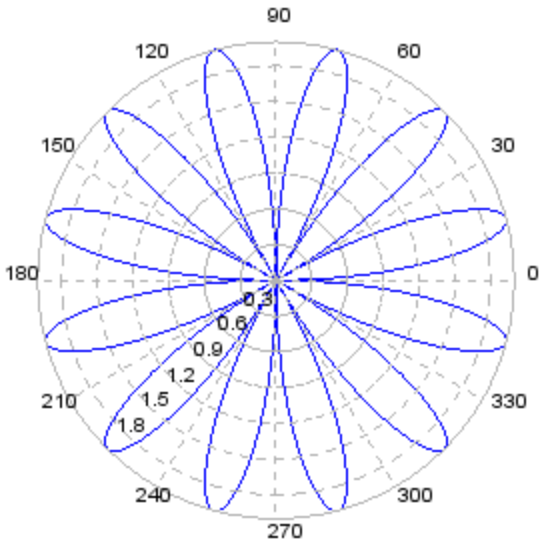


Рис 4.1. Пример 4_1

Пример 4_2. Изобразите кривую, заданную в полярных координатах, следующего вида: $\rho(\varphi) = 1 + 0.2 \cdot \sin(\varphi \text{^} 2), \varphi \in [0, 2\pi]$.

```
clf()
t=[0:0.01:2*%pi]';
rho=1+0.2*cos(t .^2)
polarplot(t,rho, style=5)
a=gca()
a.isoview='on'
a.data_bounds=[-1.2,-1.2;1.2,01.2]
```

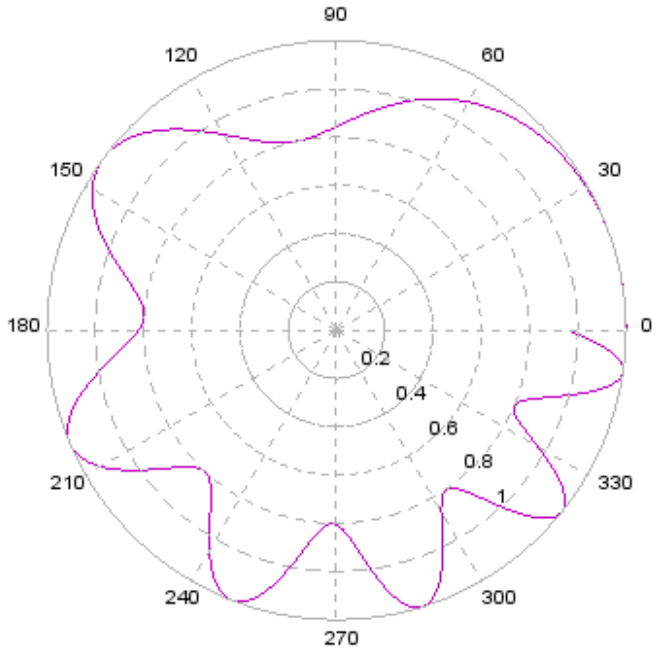


Рис 4.2. Пример 4_2

Практическая работа - генерация персертронной НС, алгоритмы обратного распространения ошибки градиентным спуском с использованием функций обучения в Scilab

Задание 1. Выполните генерацию персертронной НС для выполнения логической операции **И** с биполярными входами и выходами с использованием функций обучения **ann_PERCEPTRON** и **ann_PERCEPTRON_visualize** и функции использования сети **ann_PERCEPTRON_run.**

Синтаксис функции:

[w,b] = **ann_PERCEPTRON**(P,T)*//обучение НС*

Параметры функции:

P: Training input (обучающее входное множество – входные параметры);
T: Training target (обучающее выходное множество (целевые параметры));
w: weights for the network (веса для нейронной сети)
b: bias for the network (смещение для нейронной сети)
Синтаксис функции:

y = **ann_PERCEPTRON_run**(P,w,b) *// использование НС*

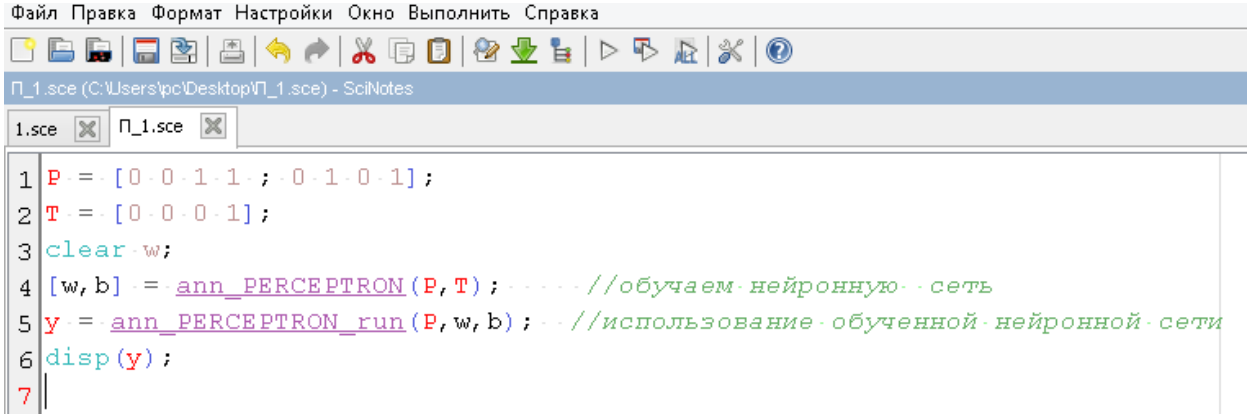


Рис 1. Обучение нейронной сети с помощью функции «**ann_PERCEPTRON**»

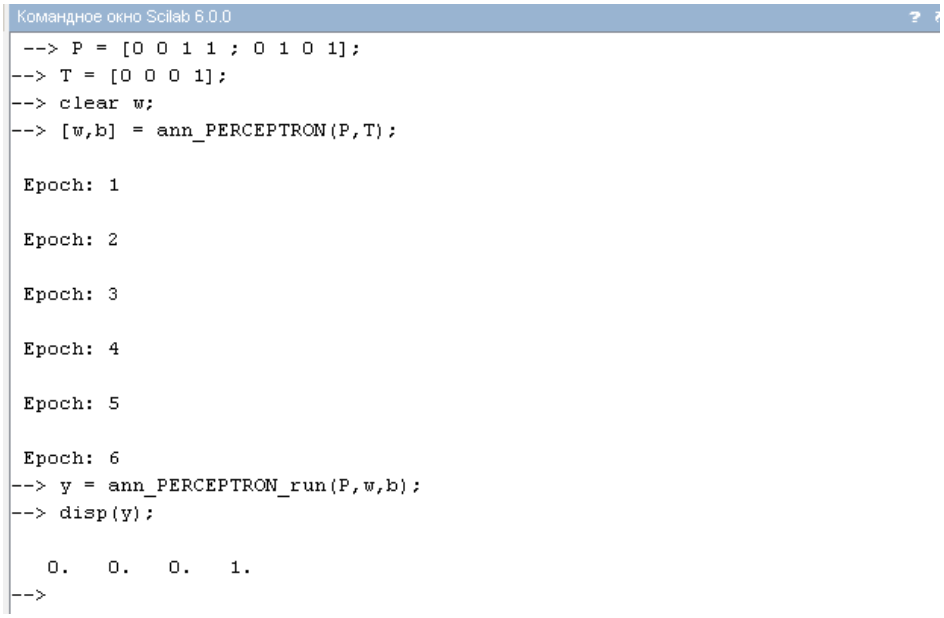


Рис 2. Результат обучения нейронной сети с помощью функции «**ann_PERCEPTRON**»

Синтаксис функции:

[w,b] = **ann_PERCEPTRON_visualize**(P,T,delay)

Параметры функции:

P: Training input (обучающее входное множество – входные параметры);
T: Training target (обучающее выходное множество (целевые параметры));
w: weights for the network (веса для нейронной сети);
b: bias for the network (смещение для нейронной сети;
delay: time delay for animation (задержка времени для анимации).

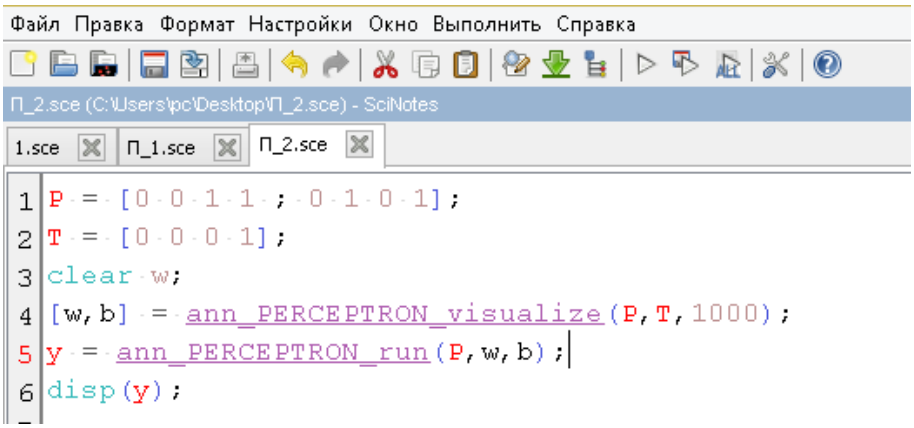


Рис 3. Обучение нейронной сети с помощью функции «**ann_PERCEPTRON_visualize**»

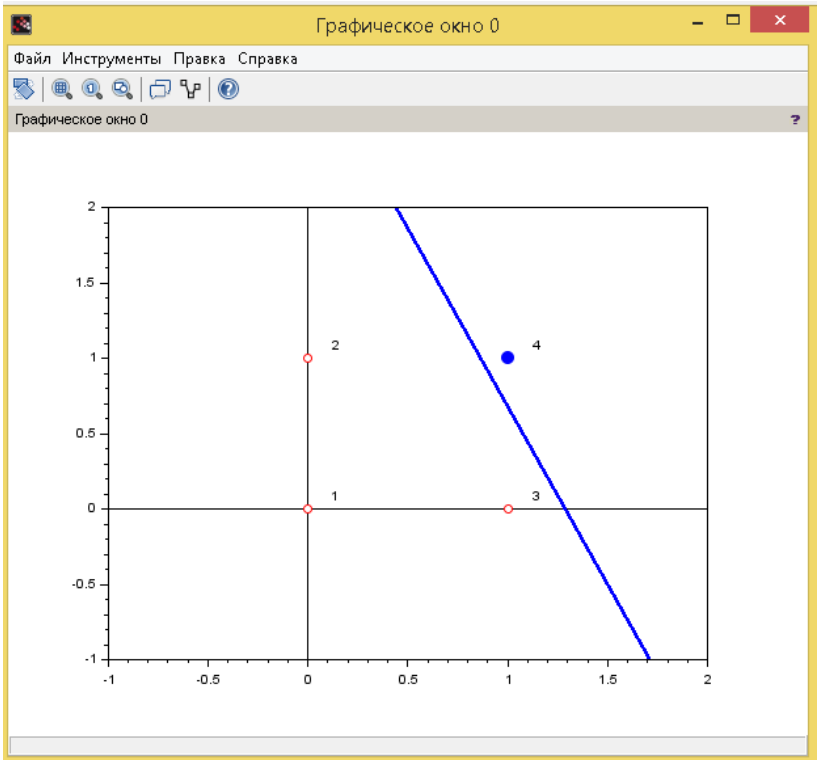


Рис 4. Графический результат обучения нейронной сети с помощью функции «**ann_PERCEPTRON_visualize**»

Задание 2. Выполните разработку нейронной сети в Scilab для сложения 2-х чисел с использованием обучающей функции алгоритма обратного распространения ошибки градиентным спуском **ann_FFBP_gd** и обучающей функции алгоритма обратного распространения ошибки градиентным спуском с импульсом ann**_FFBP_gdm**. Входные данные P должны иметь размерность не менее 15 значений в строке.

Для начала задаем входные (P) и целевые (T) параметры для обучения нейронной сети сложению чисел. Входные параметры (P) будут значения X1 и X2, а целевые параметры (T) значения Y (см. рис. 5).

Далее для обучения сети используем функцию «ann_FFBP_gd» с алгоритмом обратного распространения ошибки обучения.

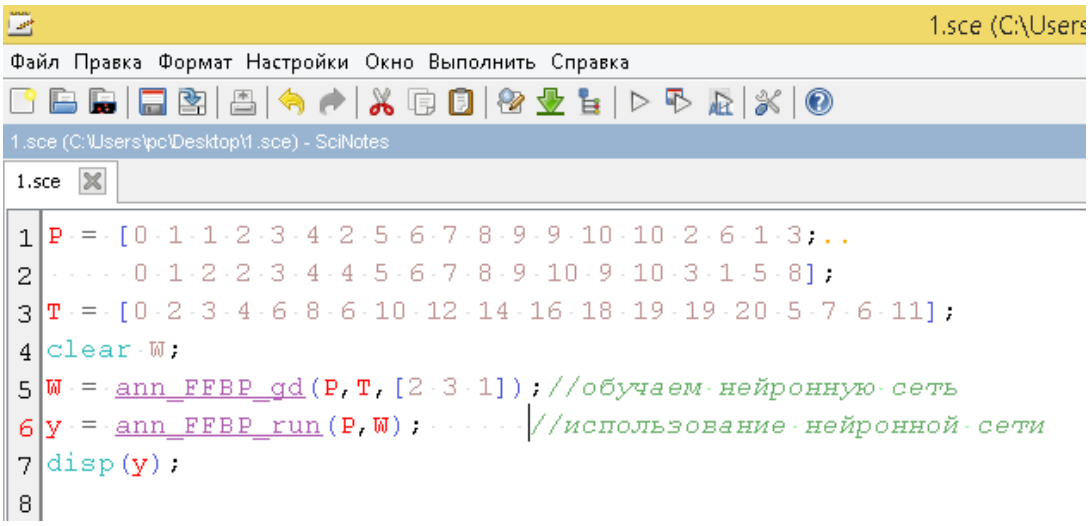


Рис 5. Обучение нейронной сети с помощью функции «**ann_FFBP_gd**»

После запуска обучения на экране появится окно Neural Network Training, это окно показывает, что обучение длится 1000 итераций и среднеквадратичную ошибку.

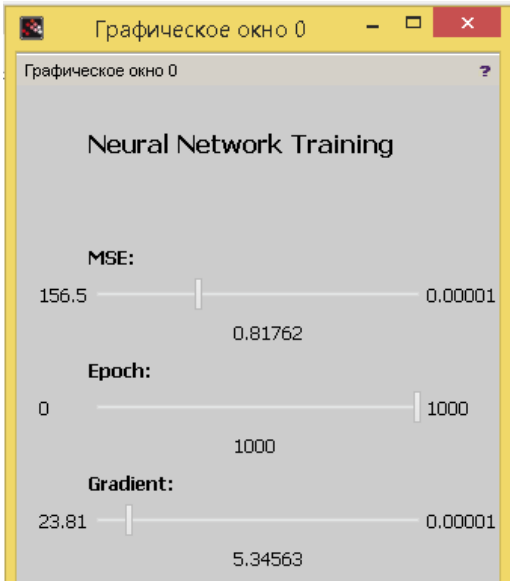


Рис 6. Окно Neural Network Training

```
Командное окно Scilab 6.0.0
 --> P = [0 1 1 2 3 4 2 5 6 7 8 9 9 10 10 2 6 1 3;..
-->      0 1 2 2 3 4 4 5 6 7 8 9 10 9 10 3 1 5 8];
--> T = [0 2 3 4 6 8 6 10 12 14 16 18 19 19 20 5 7 6 11];
--> clear W;
--> W = ann_FFBP_gd(P,T,[2 3 1]);
--> y = ann_FFBP_run(P,W);
--> disp(y);


         column 1 to 7

  -0.79154   1.48876   2.7834   4.22584   6.41164   7.67444   6.31486

         column 8 to 14

   8.57086   10.7956   15.3197   17.5406   17.9494   17.9965   17.9873

         column 15 to 19

   18.0088   5.37285   7.29362   6.2162   10.692
-->
```

Рис 7. Результат обучения нейронной сети с помощью функции «**ann_FFBP_gd**»

Синтаксис функции:

W = **ann_FFBP_gd**(P,T,N) *// Обучающая функция алгоритмом обратного распространения ошибки градиентным спуском*

W = **ann_FFBP_gd**(P,T,N,af,lr,itermax,mse_min,gd_min)

Параметры функции:

P: Training input (обучающее входное множество – входные параметры);

T: Training target (обучающее выходное множество (целевые параметры));

N: Number of Neurons in each layer, including Input and output layer (N- число нейронов каждом слое, включая входной и выходной слой (это вектор для трех слоев));

af: Activation Function from 1st hidden layer to the output layer (af - активационная функция от 1 –го слоя к выходному слою);
lr: learning rate (ir - скорость обучения);
itermax: Maximum epoch for training (максимальная эпоха для обучения);
mse_min: Minimum Error (Performance Goal) (минимальная ошибка);
gd_min: Minimum Gradient (минимальный градиент для подготовки к остановке);
W: Output Weight and bias (выходная весовая матрица и смещение);

Синтаксис функции:

[y] = ann_FFBP_run(P,W,af) *//использование НС*

Синтаксис функции:

W = **ann_FFBP_gdm**(P,T,N) *// Обучающая функция алгоритмом обратного распространения ошибки градиентным спуском с импульсом*
W = **ann_FFBP_gdm**(P,T,N,af,lr,Mr,itermax,mse_min,gd_min)
Параметры функции:

P: Training input
T: Training target
N: Number of Neurons in each layer, including Input and output layer
af: Activation Function from 1st hidden layer to the output layer
lr: Learning rate
Mr: Momentum (момент - импульс)
itermax: Maximum epoch for training
mse_min: Minimum Error (Performance Goal)
gd_min: Minimum Gradient
W: Output Weight and bias

Пример возврата значения одинаковых векторов - результат входа.

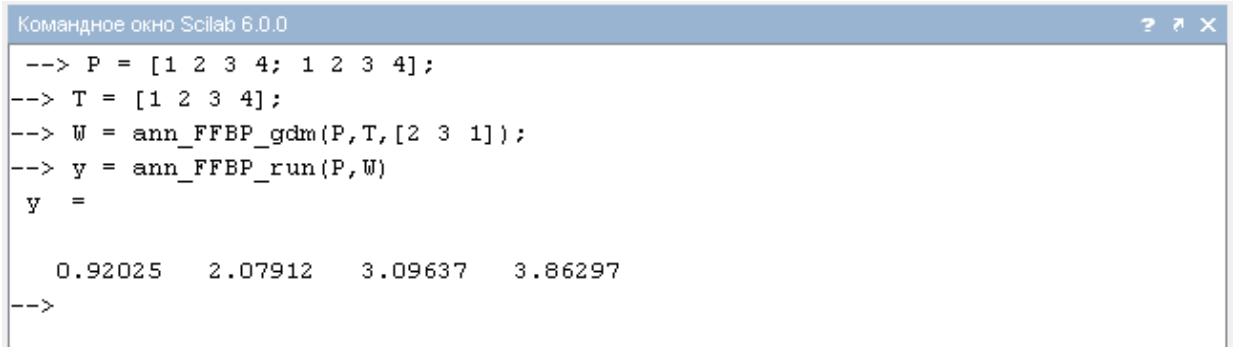


Рис 8. Результат обучения нейронной сети с помощью функции «**ann_FFBP_gd_gdm**»

Задание 3. Выполните разработку нейронной сети в Scilab для сложения значений с числом **a** с использованием обучающей функции алгоритма обратного распространения ошибки градиентным спуском из задания 2. Входные данные P должны иметь размерность не менее 10 значений в строке.

Синтаксис функции:

W = **ann_FFBP_gda**(P,T,N) *// Обучающая функция алгоритмом обратного распространения ошибки градиентным спуском с адаптивной скоростью обучения*
W =
ann_FFBP_gda(P,T,N,af,lr,lr_inc,lr_dec,itermax,mse_min,gd_min,mse_diff_max)

Параметры функции:

P: Training input;
T: Training target;
N: Number of Neurons in each layer, including Input and output layer;
af: Activation Function from 1st hidden layer to the output layer;
lr: Learning rate;
lr_inc: Learning Rate Increase Rate (увеличенная скорость обучения);
lr_dec: Learning Rate Decrease Rate (скорость снижения обучения);
itermax: Maximum epoch for training;
mse_min: Minimum Error (Performance Goal);
gd_min: Minimum Gradient;
mse_diff_max: MSE changes max in percentage, default 5% (MSE (минимальная ошибка); изменяется max в процентах, по умолчанию на 5%);
W: Output Weight and bias;

Практическая работа - Реализация Карт Кохонена в Scilab (Самоорганизующаяся карта - Self-Organizing Map)

Самоорганизующаяся карта состоит из компонентов, называемых узлами или нейронами. Их количество задаётся аналитиком. Каждый из узлов описывается двумя векторами. Первый - так называемый вектор веса **m**, имеющий такую же размерность, что и входные данные. Второй - вектор **r,** представляющий собой координаты узла на карте. Карта Кохонена визуально отображается с помощью ячеек прямоугольной или шестиугольной формы; последняя применяется чаще, поскольку в этом случае расстояния между центрами смежных ячеек одинаковы, что повышает корректность визуализации карты. Изначально известна размерность входных данных, по ней некоторым образом строится первоначальный вариант карты. В процессе обучения векторы веса узлов приближаются к входным данным. Для каждого наблюдения выбирается наиболее похожий по вектору веса узел, и значение его вектора веса приближается к наблюдению. Таким образом, в результате обучения карта Кохонена классифицирует входные данные на кластеры и визуально отображает многомерные входные данные в двумерной плоскости, распределяя векторы близких признаков в соседние ячейки и раскрашивая их в зависимости от анализируемых параметров нейронов.

Задание 1. Выполните генерацию НС с использованием функций обучения **ann_SOM**, **ann_SOM_visualize2d** и функции использования сети **ann_SOM _run.** Задайте одно из множеств значений кластера 1 с центром в точке **(2;0)** и разбросом от центра **rand**

(1,30). Затем задайте множество значений для второго кластера с центром **(-2;0).** Далее задайте третий и четвёртый кластеры с центрами (**0;2**) и **(0;-2)** (см. рис. 1).

```
x1=2+rand(1,30);
y1=0+rand(1,30);

x2=-2+rand(1,30);
y2=0+rand(1,30);

x3=0+rand(1,30);
y3=2+rand(1,30);

x4=0+rand(1,30);
y4=-2+rand(1,30);

```

Рис. 1 Создания множества в SciNotes

Далее для того чтобы отобразить наши кластеры на графике нужно написать код по созданию фигуры

```
//clf();
line_width=3;
figure(1);
plot (x3,y3,'.g','linewidth',line_width)
plot (x4,y4,'.c','linewidth',line_width)
plot (x1,y1,'.b','linewidth',line_width)
plot (x2,y2,'.r','linewidth',line_width)
g=set (gca(),"grid",[1,1]);// получаем дескриптор осей

```

Рис. 2 Создание фигуры в SciNotes

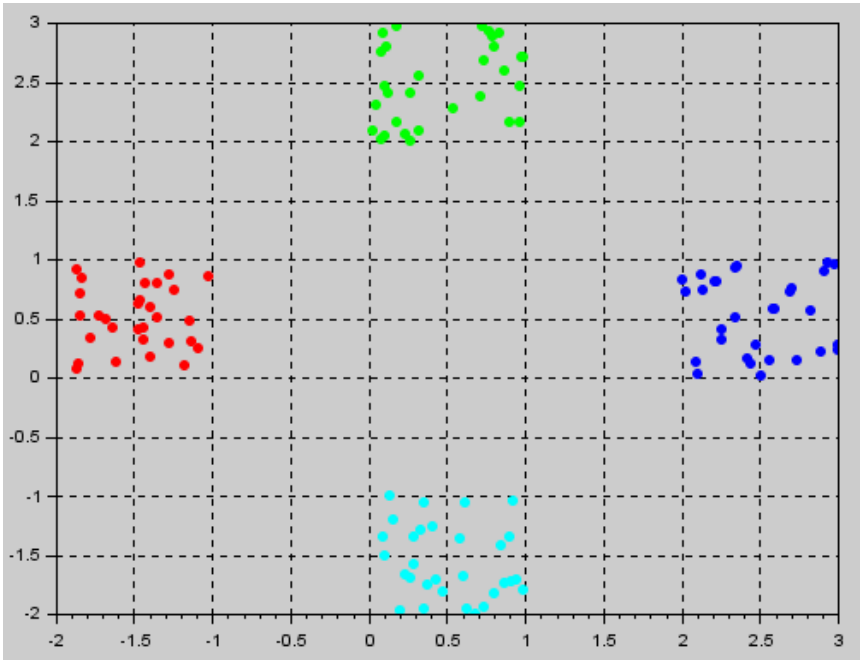


Рис. 3 Вектора данных для кластеризации

Затем нужно соединить все *x* и все *y* и присвоить их переменной *z*, которая является матрицей из векторов столбцов для обучения сети Кохонена. После этого мы выбираем сетку карты Кохонена гексагональной 2x2 (см. рис. 4).

```
g=set (gca(),"grid",[1,1]);// получаем дескриптор осей


x=[x1 x2 x3 x4];
y=[y1 y2 y3 y4];
z=[x;y];
```

Рис. 4 Задание гексагональной сетки карты Кохонена

Синтаксис функции:

W = **ann_SOM**(P)
W = **ann_SOM**(P,N,itermax,steps,NS,topfcn,distfcn)

Параметры функции
P: Training input (Входные данные для обучения);
N: Structure of Feature Map (структура карты);
itermax: Maximum epoch for training (Максимальная эпоха обучения);
steps: steps of the NS decrease (шаги уменьшения NS);
NS: nitial neighbourhood size (начальный размер окрестности);
topfcn: Topology Function (функция топологии);
distfcn: Distance Function (функция расстояния);

Синтаксис функции:

[y,classes] = **ann_SOM**_run(W,P)

Параметры функции

W: weights for the network (веса для сети);
P: Testing input (вход для тестирования);
y: Winning Neuron (Побеждающий нейрон);
classes: Classes for the input (Классы для ввода);

```
clear W;
W = ann_SOM(z,[2 2]); // обучаем НС
[y,classes] = ann_SOM_run(W,z) // использование обученной НС
```

Рис 5. Обучение нейронной сети с помощью функции **«ann_SOM»**

Синтаксис функции:

W = **ann_SOM_visualize2d** (P)
W = **ann_SOM_visualize2d** (P,N,itermax,steps,NS,topfcn,distfcn)

```
clear W;
//W = ann_SOM(z,[2 2]); // обучаем НС
//[y,classes] = ann_SOM_run(W,z)// использование обученной НС

W = ann_SOM_visualize2d(z,[2 2]);
[y,classes] = ann_SOM_run(W,z)
```

Рис 6. Обучение нейронной сети с помощью функции **«ann_SOM_visualize2d»**

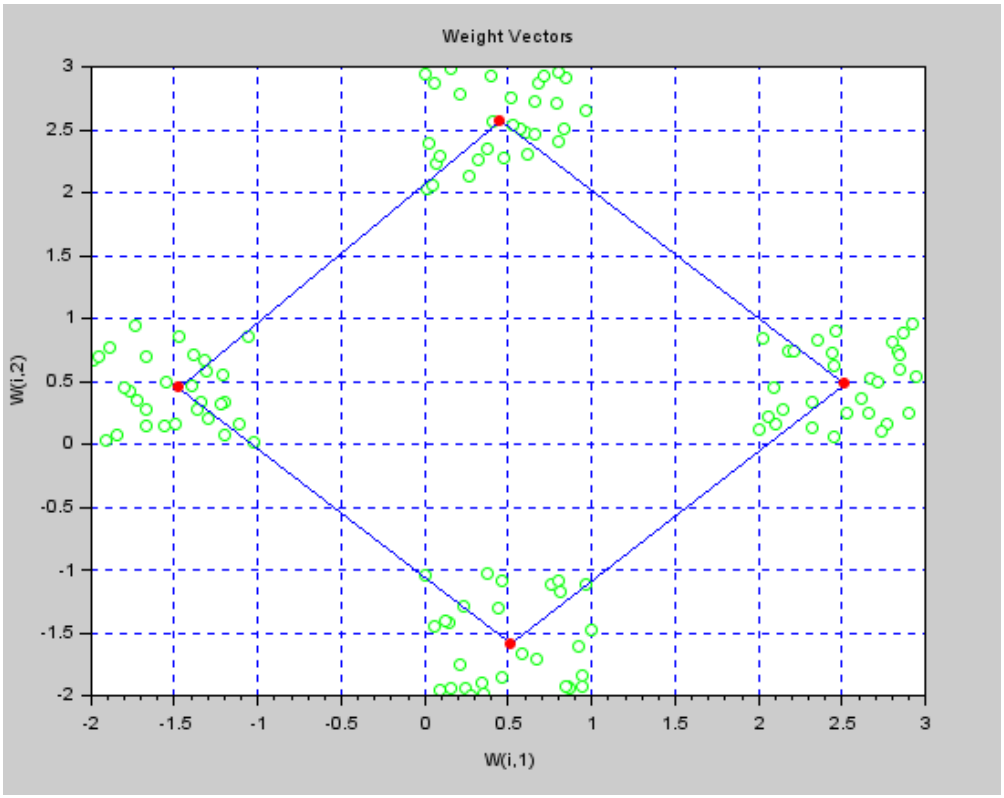


Рис 7. Результат работы функции «ann_SOM_visualize2d» – вектора весов

Пример 1.

```
x = rand(2,10);
x(:,1:5) = x(:,1:5) + 1;
W = ann_SOM_visualize2d(x,[2 2]);
[y,classes] = ann_SOM_run(W,x)
```

Практическая работа - Генерация Adaline НС с использованием функций обучения в Scilab.

Адаптивный линейный элемент (Адаптивный линейный нейрон или ADALINE) - частный случай линейного классификатора или искусственной нейронной сети с одним слоем. Был предложен Видроу и Хоффом в 1960 году, развивая математическую модель нейрона МакКаллока–Питтса.

Схема работы ADALINE несколько напоминает работу биологического нейрона:

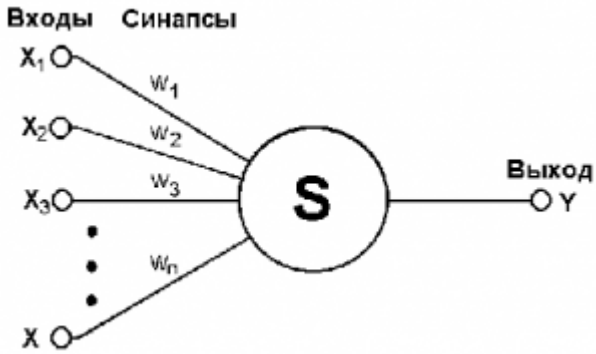


Рис 1. модель работы нейрона

На вход подаётся вектор импульсов x_n, состоящий из *n* числовых признаков. Внутри нейрона импульсы складываются с некоторыми весами $w_j, j = 1..n$, и, если суммарный импульс $S(x) = \sum_{i=1}^{n} w_j \cdot x_j$ превысит порог активации w_0, то нейрон возбуждается и выдаёт некоторое значение $a(x) = S(x) - w_0$.

Если добавить фиктивный импульс-признак $x_0 \equiv -1$ и ему сопоставить вес w_0 – порог активации, то формулу выхода Y(x) можно выписать более компактно: a(x) =(w,x) , где (w,x) - скалярное произведение, w и x - векторы весов и импульсов-признаков соответственно.

Обучение ADALINE

Пусть дана обучающая выборка: множество входных значений X и множество выходящих значений Y, такие что каждому входу x_i соответствует y_i - выход, i = 1..m. Необходимо по этим данным построить ADALINE, которая допускает наименьшее количество ошибок на этой обучающей выборке. Обучение ADALINE заключается в подборе "наилучших" значений вектора весов w. Какие значение весов лучше определяет функционал потерь. В ADALINE используется функционал, предложенный Видроу и Хоффом, $L(a, x) = (a - y)^2$. Таким образом необходимо минимизировать функционал Q(w):

$$Q(w) = \sum_{i=1}^{m} (a(x_i, w) - y_i)^2 \to \min_w$$

Применим метод градиентного спуска, тогда следующее значение будет иметь вид: $w = w - \eta \Delta Q(w) = w - \eta((w, x) - y) \cdot x$, где η - темп обучения.

Схема обучение ADALINE

Вход:

X^m - обучающая выборка из m элементов ; η - темп обучения

λ - параметр сглаживания функционала Q

Выход:

Вектор весов w

Тело:

Инициализировать веса $w_j, j = 0..n;$

Инициализировать начальную оценку функционала: $Q(w) = \sum_{i=1}^{m} (a(x_i, w) - y_i)^2;$

Повторять:

1. Выбрать объект x_i из X^j (например, случайным образом);
2. Вычислить ошибку: $\varepsilon_i = ((x_i, w) - y)^2;$
3. Сделать шаг градиентного спуска: $w := w - \eta((x_i, w) - y_i) \cdot x_i;$
4. Оценить значение функционала: $Q := (1 - \lambda)Q + \lambda \cdot \varepsilon_i;$
5. Пока значение Q не стабилизируется и/или веса w не перестанут изменяться.

Задание 1. Выполните генерацию НС Adaline (адаптивного линейного элемента) для выполнения логической операции **ИЛИ** с биполярными входами и выходами с использованием функций обучения **ann_ADALINE** и **ann_ ADALINE_predit** и функции использования сети **ann_ ADALINE_run.**

Синтаксис функции:

[w,b] = **ann_ADALINE**(P,T) *) //обучение НС*

[w,b] = **ann_ADALINE**(P,T, alpha, itermax, initfunc)

Параметры функции:

P: Training input (обучающее входное множество – входные параметры);
T: Training target (обучающее выходное множество (целевые параметры));
alpha: Learning Rate (скорость обучения);
itermaxs: Maximum Training Iteration (максимальная итерация обучения);
initfunc: Weight and Bias Initialize functions, either 'rand', 'zeros', or 'ones' (вес и смещение инициализации функций, «rand», «нулей» или «единиц»);
w: weights for the network (веса для нейронной сети);
b: bias for the network (смещение для нейронной сети);

Синтаксис функции:

y = **ann_ ADALINE_run**(P,w,b) *// использование НС*

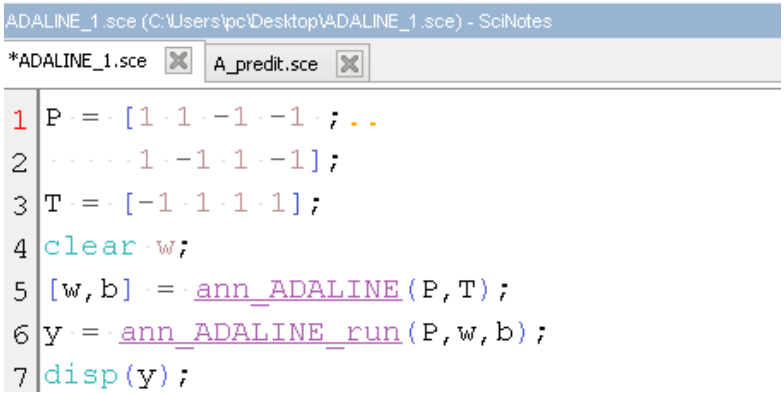


Рис 1. Обучение нейронной сети с помощью функции «**ann_ ADALINE**»

Задание 2. Выполните генерацию НС Adaline (адаптивного линейного элемента) для прогнозирования с использованием функции обучения и использования сети **ann_ ADALINE_predit.**

Синтаксис функции:

[w,b] = **ann_ADALINE_predict**(P,T,alpha) *// Создание и обучение НС с адаптивным линейным нейроном с инкрементным (проходящим через данные по одному) алгоритмом обучения и отложенной задержкой.*

[w,b] = **ann_ADALINE_predict**(P,T,alpha,itermax,D,initfunc)

Параметры функции:

P: Training input (обучающее входное множество – входные параметры);
T: Training target (обучающее выходное множество (целевые параметры));
alpha: Learning Rate (скорость обучения);
itermaxs: Maximum Training Iteration (максимальная итерация обучения);
D: number of delays (количество задержек);
initfunc: Weight and Bias Initialize functions, either 'rand', 'zeros', or 'ones' (вес и смещение инициализации функций, «rand», «нулей» или «единиц»);
w: weights for the network (веса для нейронной сети);
b: bias for the network (смещение для нейронной сети);
y: Predicted output (прогнозируемый результат);
ee: Error between T and y (ошибка между T и y);

```
A_predit.sce (C:\Users\pc\Desktop\A_predit.sce) - SciNotes
ADALINE_1.sce | *A_predit.sce
1  x = 0:0.02:20;
2  P = sin(x);
3  T = 2.*sin(x - 0.2);
4  plot(x,P,x,T);
5
6  // Create and train a network to preduct T from P
7  // Создание и обучение сети для прогнозирования T из P
8  Delay = 3;
9  [w,b,y,ee] = ann_ADALINE_predict(P,T,0.2,1,Delay);
10 figure(); plot(T); plot(y,'r');
```

Рис 2. Создание и обучение нейронной сети с помощью функции «**ann_ ADALINE_predict»**

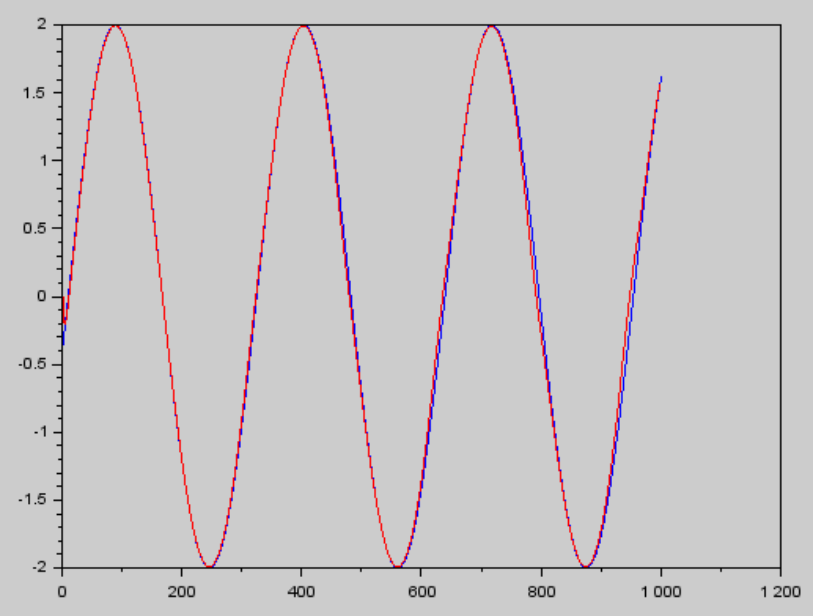


Рис 3. Результат работы функции «**ann_ ADALINE_predict»**

Задание 3. Выполните создание и обучение НС Adaline для прогнозирования функции $\cos(x)$, если желаемый результат прогнозирования функция $2\cdot\cos(x+\frac{\pi}{20})$, $x\in[0;10]$. Выполните генерацию НС Adaline для прогнозирования с использованием функции обучения и использования сети **ann_ ADALINE_predit.**

Задание 4. Выполните отладку и тестирование генерации Adaline нейронной сети с биполярными входами и выходами с использованием предложенного ниже алгоритма программы обучения сети.

Отладку произвести путем изменения ошибки **e** от 0.9 до 1.01 (взять два значения из диапазона). Такие большие значения ошибки взяты, так как алгоритм имеет очень низкую скорость сходимости.

```
// Generate OR function with bipolar inputs and targets using Adaline net
// Truth table for OR gate (Таблица истиности для ИЛИ)
// X1 X2 Y
// -1 -1 -1
// -1  1  1
//  1 -1  1
//  1  1  1           (Bipolar (1 ,-1))

clc ;
clear ;
disp ('Adaline network for OR function Bipolar inputs and targets');
// введите значения векторов входов x1 и x2
x1 =[1 1 -1 -1];
x2 =[1 -1 1 -1];

// bias pattern (смещенный вектор x3)
x3 =[1 1 1 1];
// target vector (вектор выхода)
t =[1 1 1 -1];
```

```
// initial weights and bias (задание начальных значений весов и смещения)
w1 =0.1; w2 =0.1; b =0.1;
// initialize learning rate (задание скорости обучения)
alpha =0.1;
// error convergence  (сходимость алгоритма по ошибке)
e =2;
// change in weights and bias (измение в весах и смещении)
delw1 =0; delw2 =0; delb =0;
epoch =0;     //задание начального значения для эпохи обучения
while (e > 1.01)
    epoch = epoch +1;
    e =0;
    for i =1:4
        nety(i)=w1*x1(i)+w2*x2(i)+b;
        // net input calculated and target(вычисление входов сети и выхо
        nt =[nety(i) t(i)];
        delw1 = alpha .*(t(i)-nety(i)).*x1(i);
        delw2 = alpha .*(t(i)-nety(i)).*x2(i);
        delb = alpha .*(t(i)-nety(i)).*x3(i);
```

```
        delb = alpha .*(t(i)-nety(i)).*x3(i);
        // weight changes (изменение весов)
        wc =[delw1 delw2 delb];
        // updating of weights and bias (обновление весов и смещения)
        w1=w1+delw1;
        w2=w2+delw2;
        b=b+delb ;
        //new weights (новые веса)
        w=[w1 w2 b];
        // input pattern (входные значения векторов)
        x=[x1(i) x2(i) x3(i)];
        // printing the results obtained (печать результат)
        disp ([x nt wc w]);
    end
    for i =1:4
        nety(i)=w1 .*x1(i)+w2 .*x2(i)+b;
        e=e+(t(i)-nety(i)) .^ 2;
    end
end
disp(e);
disp(epoch);
```

Рис 4. Программный код алгоритма обучения Adaline

```
            column 1 to 7

   - 1.   - 1.     1.   - 0.3602517   - 1.     0.0639748     0.0639748

            column  8 to 11

   - 0.0639748     0.4893198     0.5204044     0.4575480
-->disp(e);

    1.0093303
-->disp(epoch);

    5.
```

Рис 5. Результат работы программы алгоритма обучения Adaline с ошибкой больше 1 и с числом эпох - 5

Практическая работа - Генерация конкурентоспособной НС с использованием функций обучения в Scilab

Конкурентное обучение - это форма неконтролируемого обучения в искусственных нейронных сетях, где узлы конкурируют за право реагировать на подмножество входных данных. Вариант обучения Hebbian, конкурентное обучение работает за счет увеличения специализации каждого узла в сети. Он хорошо подходит для поиска кластеров в данных.

Модели и алгоритмы, основанные на принципе конкурентного обучения, включают векторное квантование и самоорганизующиеся карты (карты Кохонена).

Существует три основных элемента правила конкурентного обучения:

- Набор нейронов, которые все одинаковы, за исключением некоторых случайно распределенных синаптических весов, и которые поэтому реагируют по-разному на заданный набор входных паттернов;
- Ограничение, накладываемое на «прочность» каждого нейрона;
- Механизм, позволяющий нейронам конкурировать за право реагировать на заданное подмножество входных данных, так что только один выходной нейрон (или только один нейрон на группу) является активным (т.е. «включено») одновременно. Нейрон, который выигрывает соревнование, называется нейроном «победитель получает все».

Соответственно, отдельные нейроны сети учатся специализироваться на ансамблях сходных паттернов и таким образом становятся «детекторами признаков» для разных классов входных паттернов.

Тот факт, что конкурентные сети перекодируют множества коррелированных входных данных в один из нескольких выходных нейронов, по существу устраняет избыточность в представлении, которая является неотъемлемой частью обработки в биологических сенсорных системах.

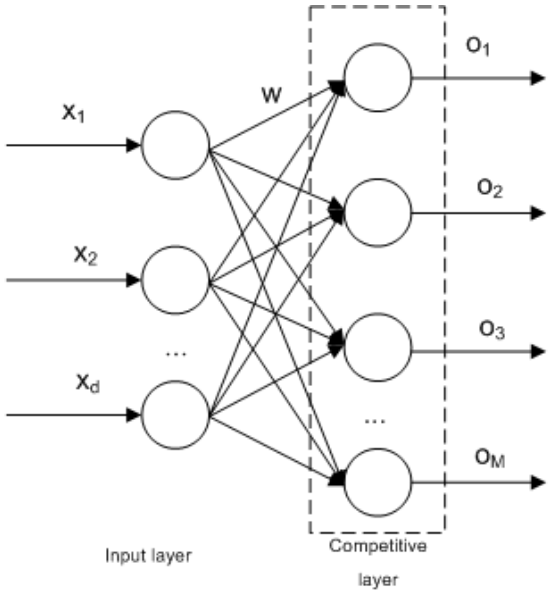


Рис. 1 Архитектура конкурентной нейронной сети

Архитектура и реализация

Конкурентное обучение обычно реализуется с помощью нейронных сетей, которые содержат скрытый слой, который обычно называют «конкурным уровнем». Каждый конкурентный нейрон описывается вектором весов $w_i = (w_{i1},...,w_{id})^T, i = 1,...,M$ и вычисляет меру подобия между входными данными $x^n = (x_{n1},...,x_{nd}) \in R^d$ и весовой вектор w_i.

Для каждого входного вектора конкурирующие нейроны «конкурируют» друг с другом, чтобы увидеть, какой из них наиболее похож на этот вектор. Выход нейрона m победителя $o_m = 1$ и выход всех другие нейроны устанавливают $o_i = 1, i = 1,...,M, i \neq m.$

Обычно для изменения подобия используется обратная величина евклидового расстояния: $\|x - w_i\|$ между выходным вектором x^n и вектором весов w_i.

Вот простой конкурентный алгоритм обучения, чтобы найти три кластера в рамках некоторых входных данных.

1. (Настройка). Пусть все датчики подключены к трем различным узлам, поэтому каждый узел подключен к каждому датчику. Пусть веса, которые каждый узел дает своим датчикам, устанавливаются случайным образом между 0 и 1. Пусть выход каждого узла будет суммой всех датчиков, причем сила сигнала каждого датчика умножается на его вес.
2. Когда в сети задают входы узлов, то узел с наибольшим входом считается победителем.
3. Нейрон – победитель обновляет каждый из своих весов, перемещая вес от соединений, что дает более слабые сигналы соединением, которые дают ему более сильные сигналы.

Таким образом, по мере поступления большого количества данных, каждый узел сходится к центру кластера, который он стал представлять и активирует сильнее сигналы для входов в этом кластере и более слабо для входов в другие кластеры.

Задание 1. Выполните генерацию НС с использованием функции обучения **ann_ COMPET,** и функции использования сети **ann_COMPET _run.** Исходные параметры для моделирования сети: обучающее входное множество ***x*** состоит из 10 значений, заданных датчиком случайных чисел; количество нейронов в конкурентном слое – 4 (кол-во выходных классов). Выполните вывод построенных точек в графическое окно, с помощью функции **plot**.

Синтаксис функции:

[W,b] = **ann_ COMPET** (P,N) *) //обучение НС*

[W,b] = **ann_ COMPET** (P,N,lr,lr_c,itermax)

Параметры функции:

P: Training input (обучающее входное множество – входные параметры);

N: Number of neurons in competition layer (Number of classes) (кол-во нейронов в конкурентном слое (кол-во классов))

lr: Learning Rate (скорость обучения);

lr_c: Learning Rate for bias (Conscience learning rate) (скорость обучения для смещения);

itermax: Maximum epoch for training (максимальная эпоха обучения)

W: Output Weight (Выходная весовая матрица)

b: Output bias (Выходной вектор смещения)

Синтаксис функции:

[y,classes] = **ann_COMPET_run**(W,P) *// использование НС*

```
clear w;
x = rand(2,10);
x(:,1:5) = x(:,1:5) + 1;
plot(x(1,:),x(2,:),'.');
[W,b] = ann_COMPET(x,4);close;
plot(W(:,1), W(:,2), 'or');
```

Рис 1. Обучение нейронной сети с помощью функции «**ann_ COMPET**» и вывод построенных точек в графическое окно

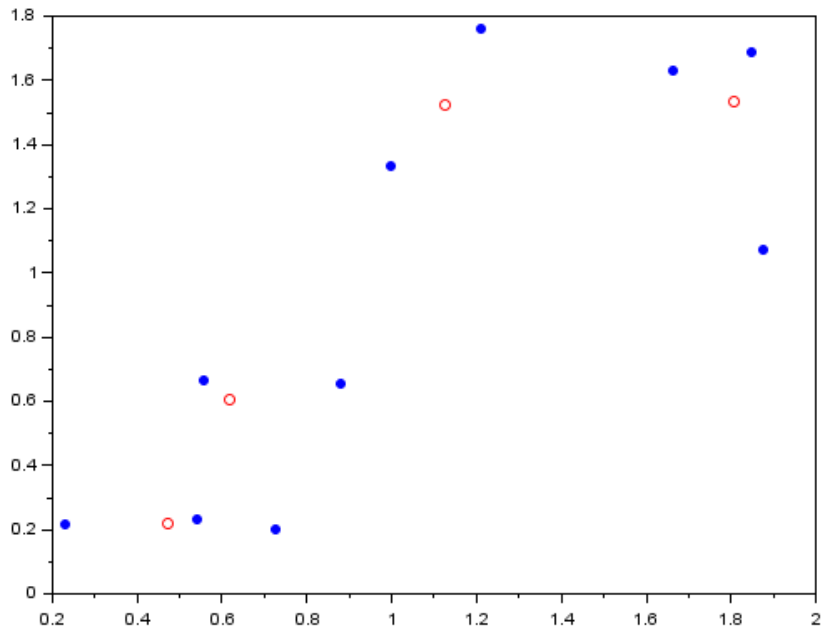


рис 2. Графический результат обучения нейронной сети с помощью функции «**ann_ COMPET**»

```
clear w;
x = rand(2,10);
x(:,1:5) = x(:,1:5) + 1;
[W,b] = ann_COMPET(x,4);
[y,classes] = ann_COMPET_run(W,x)
```

Рис 3. Обучение нейронной сети с помощью функции «**ann_ COMPET**»

```
Командное окно Scilab 6.0.0
 --> clear w;
--> x = rand(2,10);
--> x(:,1:5) = x(:,1:5) + 1;
--> [W,b] =  ann_COMPET(x,4);
--> [y,classes] = ann_COMPET_run(W,x)
 classes  =

   1.   3.   3.   3.   1.   4.   2.   4.   4.   2.
 y  =

   1.   0.   0.   0.   1.   0.   0.   0.   0.   0.
   0.   0.   0.   0.   0.   0.   1.   0.   0.   1.
   0.   1.   1.   1.   0.   0.   0.   0.   0.   0.
   0.   0.   0.   0.   0.   1.   0.   1.   1.   0.
-->
```

рис 4. Результат обучения нейронной сети с помощью функции «**ann_ COMPET**»

Задание 2. Выполните генерацию НС с использованием функции обучения **ann_ COMPET,** и функции использования сети **ann_COMPET _run.** Исходные параметры для моделирования сети: обучающее входное множество ***x*** состоит из 24 значений, заданных датчиком случайных чисел; количество нейронов в конкурентном слое – 6 (кол-во выходных классов). Выполните вывод построенных точек в графическое окно, с помощью функции **plot**.

Задание 3. Выполните генерацию НС с использованием функции обучения **ann_COMPET_visualize2d** и функции использования сети **ann_COMPET _run.** Исходные параметры для моделирования сети: обучающее входное множество ***x*** состоит из 10 значений, заданных датчиком случайных чисел; количество нейронов в конкурентном слое – 4 (кол-во выходных классов).

Синтаксис функции

[W,b] = **ann_COMPET_visualize**2d(P,N) *//обучение НС*

[W,b] = **ann_COMPET_visualize2d**(P,N,lr,lr_c,itermax)

C_1.sce | C_1_1.sce | C_2d.sce

```
clear w;
x = rand(2,10);
x(:,1:5) = x(:,1:5) + 1;
[W,b] = ann_COMPET_visualize2d(x,4);
[y,classes] = ann_COMPET_run(W,x)
```

Рис 5. Обучение нейронной сети с помощью функции «**ann_COMPET_visualize2d**»

```
Командное окно Scilab 6.0.0
 --> clear w;
--> x = rand(2,10);
--> x(:,1:5) = x(:,1:5) + 1;
--> [W,b] =  ann_COMPET_visualize2d(x,4);
--> [y,classes] = ann_COMPET_run(W,x)
 classes  =

   4.    4.    1.    1.    1.    3.    3.    2.    3.    2.
 y  =

   0.    0.    1.    1.    1.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.    0.    0.    1.    0.    1.
   0.    0.    0.    0.    0.    1.    1.    0.    1.    0.
   1.    1.    0.    0.    0.    0.    0.    0.    0.    0.
-->
```

Рис 6. Результат обучения нейронной сети с помощью функции «**ann_COMPET_visualize2d**»

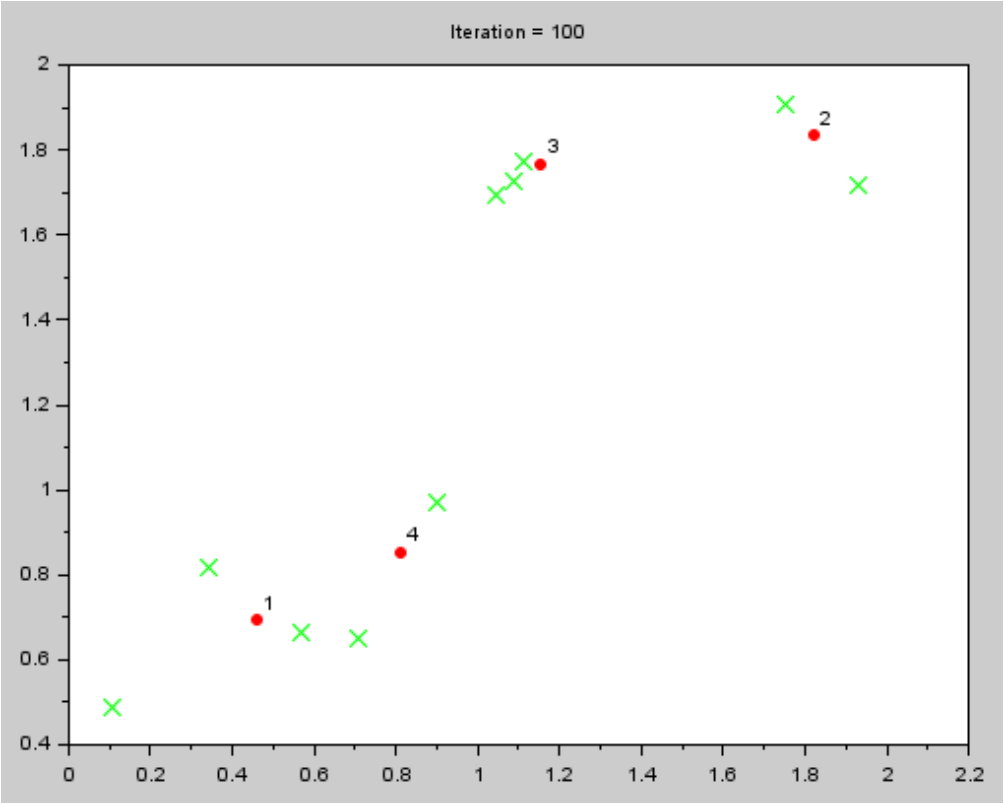


рис 6. Графический результат обучения нейронной сети с помощью функции «**ann_COMPET_visualize2d**»

Задание 4. Выполните генерацию НС с использованием функции обучения **ann_COMPET_visualize3d** и функции использования сети **ann_COMPET _run.** Исходные параметры для моделирования сети: обучающее входное множество ***x*** состоит из 10 значений, заданных датчиком случайных чисел; количество нейронов в конкурентном слое – 2 (кол-во выходных классов).

Синтаксис функции

[W,b] = **ann_COMPET_visualize**3d(P,N) *//обучение НС*

[W,b] = **ann_COMPET_visualize3d**(P,N,lr,lr_c,itermax)

```
clear w;
x = rand(3,10);
x(:,1:5) = x(:,1:5) + 1;
[W,b] = ann_COMPET_visualize3d(x,2);
[y,classes] = ann_COMPET_run(W,x)
```

Рис 7. Обучение нейронной сети с помощью функции «**ann_COMPET_visualize3d**»

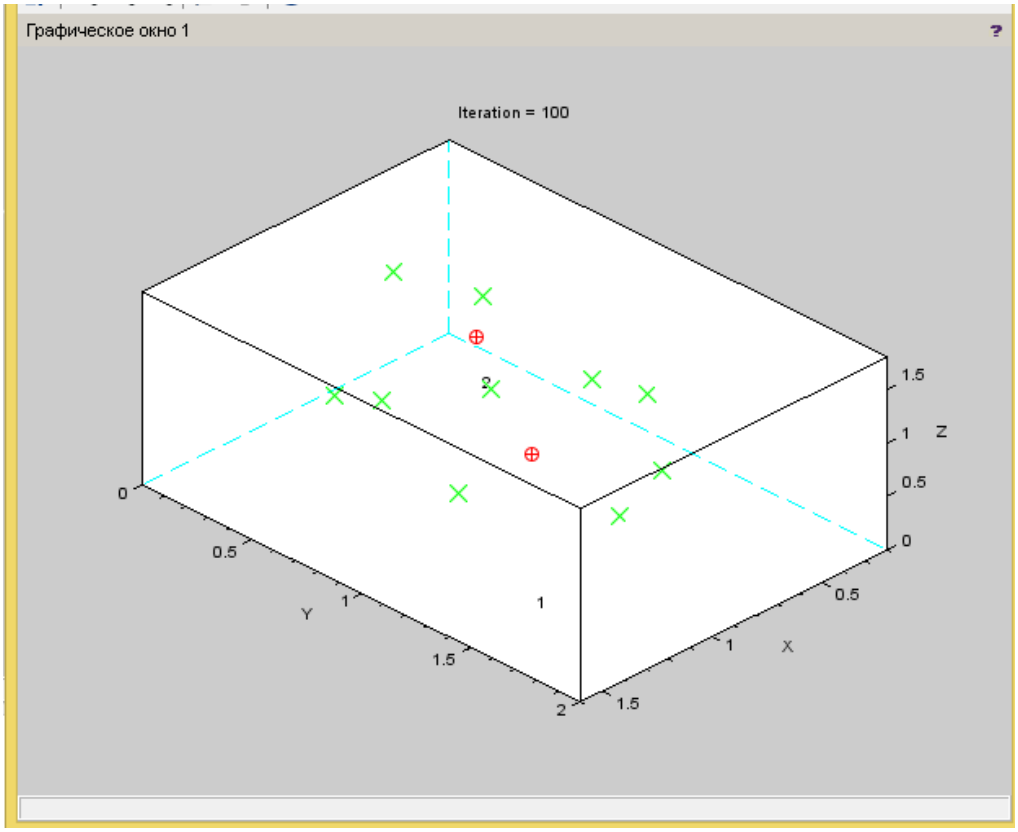


рис 8. Графический результат обучения нейронной сети с помощью функции «**ann_COMPET_visualize3d**»

Задание 5. Выполните генерацию НС с использованием функций обучения **ann_COMPET_visualize2d** и **ann_COMPET_visualize3d** и функции использования сети **ann_COMPET _run.** Исходные параметры для моделирования сети: обучающее входное множество ***x*** состоит из 12 значений, заданных датчиком случайных чисел; количество нейронов в конкурентном слое – 4 (кол-во выходных классов).

Практическая работа - Генерация сети векторного квантования, обучаемой с учителем (LVQ - Learning vector Quantization) в Scilab

Нейронная сеть Кохонена

Нейронные сети Кохонена - класс нейронных сетей, основным элементом которых является слой Кохонена. Слой Кохонена состоит из *адаптивных линейных сумматоров* («линейных формальных нейронов»). Как правило, выходные сигналы слоя Кохонена обрабатываются по правилу «*Победитель получает всё*»: наибольший сигнал превращается в единичный, остальные обращаются в ноль.

По способам настройки входных весов сумматоров и по решаемым задачам различают много разновидностей сетей Кохонена. Наиболее известные из них:

• сети векторного квантования сигналов, тесно связанные с простейшим базовым алгоритмом кластерного анализа (метод динамических ядер или *К-средних*);

• самоорганизующиеся карты Кохонена (англ. self-organising maps, SOM);

• сети векторного квантования, обучаемые с учителем (англ. learning vector quantization).

Слой Кохонена

Базовая версия

Слой Кохонена состоит из некоторого количества *n* параллельно действующих линейных элементов. Все они имеют одинаковое число входов *m* и получают на свои входы один и тот же вектор входных сигналов $x = (x_1,\ldots,x_m)$. На выходе j-го линейного элемента получаем сигнал

$$y_j = w_{j0} + \sum_{i=1}^{m} w_{ji} \cdot x_i,$$

где w_{ji} - весовой коэффициент *i-го* входа *j-го* нейрона; *i* – номер входа; *j*- номер нейрона; w_{j0} - пороговый коэффициент.

После прохождения слоя линейных элементов сигналы посылаются на обработку по правилу «победитель забирает все»: среди выходных сигналов выполняется поиск максимального y_j; его номер $y_{\max} = \arg\max_j \{y_j\}$. Окончательно, на выходе сигнал с номером $j_{\max}$ равен единице, остальные — нулю. Если максимум одновременно достигается для нескольких $j_{\max}$, то:

• либо принимают все соответствующие сигналы равными единице;

• либо равным единице принимают только первый сигнал в списке (по соглашению).

«нейроны Кохонена можно воспринимать как набор электрических лампочек, так что для любого входного вектора загорается одна из них»

Геометрическая интерпретация

Большое распространение получили слои Кохонена, построенные следующим образом: каждому (*j-му*) нейрону сопоставляется точка $W_j = (w_{j1},\ldots,w_{jm})$ в *m*-мерном пространстве (пространстве сигналов). Для входного вектора $x = (x_1,\ldots,x_m)$ вычисляются его евклидовы расстояния $p_j(x)$ до точек W_j и «ближайший получает всё» - тот нейрон, для которого это расстояние минимально, выдает единицу, остальные - нули. Следует заметить, что для сравнения расстояний достаточно вычислять линейную функцию сигнала:

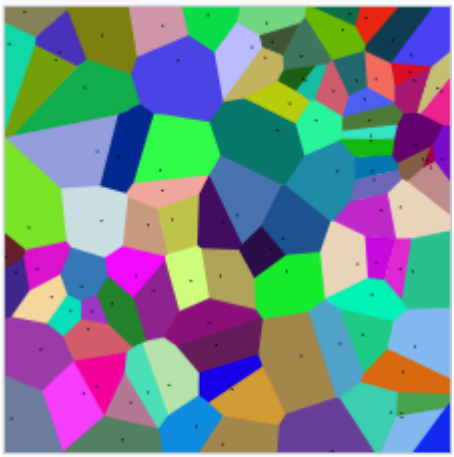
$$p_j(x)^2 = \left\|x - W_j\right\|^2 = \left\|W_j\right\|^2 - 2\cdot\sum_{i=1}^{m} w_{ji}\cdot x_i + \left\|x\right\|^2$$

(здесь $\left\|y\right\|$ - Евклидова норма вектора: $\left\|y\right\|^2 = \sum_i^m y_i^2$). Последнее слагаемое $\left\|x\right\|^2$ одинаково для всех нейронов, поэтому для нахождения ближайшей точки оно не нужно. Задача сводится к поиску номера наибольшего из значений линейных функций:

$$j_{\max} = \arg\max_j \left\{\sum_{i=1}^{m} w_{ji}\cdot x_i - \frac{1}{2}\left\|W_j\right\|^2\right\}.$$

Таким образом координаты точки $W_j = (w_{j1},\ldots,w_{jm})$ совпадают с весами линейного нейрона слоя Кохонена (при этом значение порогового коэффициента $w_{j0} = -\left\|W_j\right\|^2/2).$

Если заданы точки $W_j = (w_{j1},\ldots,w_{jm}),$ то m – мерное пространство разбивается на соответствующие многогранники Вороного - Дирихле V_j: многогранник V_j состоит из точек, которые ближе к W_j ,чем к другим $W_k\ (k \neq j).$



Разбиение плоскости на многоугольники Вороного-Дирихле для случайно выбранных точек (каждая точка указана в своём многоугольнике).

Сети векторного квантования (LVQ - Learning vector Quantization)

Задача векторного квантования с *k* кодовыми векторами W_j для заданной совокупности входных векторов *S* ставится как задача минимизации искажения при кодировании, то есть при замещении каждого, то есть вектора из *S* соответствующим кодовым вектором. В базовом варианте сетей Кохонена используется метод наименьших квадратов и искажение *D* вычисляется по формуле

$$D = \sum_{j=1}^{k}\sum_{x\in K_j}\left\|\mathrm{x} - W_j\right\|^2,$$

где K_j состоит из тех точек $x \in S,$ которые ближе к W_j, чем к другим $W_i, (i \neq j).$ Другими словами, K_j состоит из тех точек $x \in S,$ которые кодируются кодовым вектором W_j.

Если совокупность S задана и хранится в памяти, то стандартным выбором в обучении соответствующей сети Кохонена является метод *К-средних*. Это *метод расщепления*:

- при данном выборе кодовых векторов (они же весовые векторы сети) W_j минимизацией D находим множества K_j - они состоят из тех точек $x \in S,$ которые ближе к W_j, чем к другим W_i;
- при данном разбиении S на множества K_j минимизацией *D* находим оптимальные позиции кодовых векторов W_j - для оценки по методу наименьших квадратов это просто средние арифметические:

  $W_j = \frac{1}{|K_j|} \sum_{x \in K_j} x,$ где $|K_j|$ - число элементов K_j.

Далее итерируем. Этот метод расщепления сходится за конечное число шагов и дает локальный минимум искажения.

Если же, например, совокупность *S* заранее не задана, или по каким-либо причинам не хранится в памяти, то широко используется онлайн метод. Векторы входных сигналов *x* обрабатываются по одному, для каждого из них находится ближайший кодовый вектор («победитель», который «забирает всё») $W_{j(x)}$. После этого данный кодовый вектор пересчитывается по формуле $W_{j(x)}^{new} = W_{j(x)}^{old}(1-\theta) + x\theta,$ где $\theta \in (0,1)$ - шаг обучения. Остальные кодовые векторы на этом шаге не изменятся.

Для обеспечения стабильности используется онлайн метод с затухающей скоростью обучения: если *T* - количество шагов обучения, то полагают $\theta = \theta(T)$. Функцию $\theta(T) > 0$ выбирают таким образом, чтобы $\theta(T) \to 0$ монотонно при $T \to \infty$ и чтобы ряд $\sum_{T \to \infty}^{\infty} \theta(T)$ расходился, например, $\theta(T) = \theta_0 / T.$

Векторное квантование является намного более общей операцией, чем кластеризация, поскольку кластеры должны быть разделены между собой, тогда как совокупности K_j для разных кодовых векторов W_j не обязательно представляют собой раздельные кластеры. С другой стороны, при наличии разделяющихся кластеров векторное квантование может находить их и по-разному кодировать.

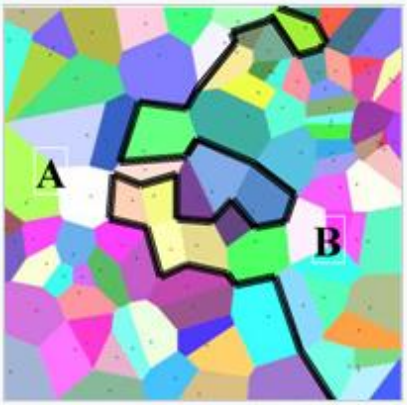
Сети векторного квантования, обучаемые с учителем

Решается задача классификации. Число классов может быть любым. Изложим алгоритм для двух классов **A** и **B**. Исходно для обучения системы поступают данные, класс которых известен. Задача: найти для класса **A** некоторое количество k_A кодовых векторов

W_j^A, а для класса **B** некоторое (возможно другое) количество k_B кодовых векторов W_i^B таким образом, чтобы итоговая сеть Кохонена с $k_A + k_B$ кодовыми векторами W_j^A, W_i^B, (объединяем оба семейства) осуществляла классификацию по следующему решающему правилу:

- если для вектора входных сигналов *x* ближайший кодовый вектор («победитель», который в слое Кохонена «забирает всё») принадлежит семейству $\{W_j^A\}$, то *x* принадлежит классу **A**; если же ближайший к *x* кодовый вектор принадлежит семейству $\{W_i^B\}$, то *x* принадлежит классу **B**.

С каждым кодовым вектором объединённого семейства $\{W_j^A\} \cup \{W_i^B\}$, связан многогранник Вороного-Дирихле. Обозначим эти многогранники V_j^A, V_i^B, соответственно Класс A в пространстве сигналов, согласно решающему правилу, соответствует объединению $\cup_j V_j^A$, а класс B соответствует объединению $\cup_i V_i^B$,. Геометрия таких объединений многогранников может быть весьма сложной (см. рисунок с примером возможного разбиения на классы).



Пример возможного разделения классов, составленного с помощью разбиения Вороного-Дирихле.

Правила обучения сети онлайн строится на основе базового правила обучения сети векторного квантования. Пусть на вход системы подаётся вектор сигналов *x*, класс которого известен. Если он классифицируется системой правильно, то соответствующий *x* кодовый вектор *W* слегка сдвигается в сторону вектора сигнала («поощрение»).

$$W^{new} = W^{old}(1-\theta) + x\theta,$$

Если же x классифицируется неправильно, то соответствующий x кодовый вектор *W* слегка сдвигается в противоположную сторону от сигнала («наказание»)

$W^{new} = W^{old}(1+\theta) - x\theta,$ где $\theta \in (0,1)$ - шаг обучения. Для обеспечения стабильности используется онлайн метод с затухающей скоростью обучения. Возможно также использование разных шагов для «поощрения» правильного решения и для «наказания» неправильного.

Это - простейшая (базовая) версия метода. Существует множество других модификаций.

Задание 1. Выполните генерацию сети векторного квантования (LVQ) с использованием функций обучения **ann_LVQ1** и функции использования сети **ann_LVQ_run.** Исходные параметры для моделирования сети: обучающее входное множество ***x*** состоит из 10 значений, заданных датчиком случайных чисел; желаемые выходные классы T разделим две группы; количество нейронов в конкурентном слое – 4.

Синтаксис функции:

[W,b] = **ann_LVQ1**(P,T,N2) *//обучение НС*

[W,b] = **ann_LVQ1**(P,T,N2,lr,itermax)

Параметры функции:

P: Training input (обучающее входное множество – входные параметры);

T: Target Classes (желаемые выходные классы)

N2: Number of neurons in competition layer (Number of subclasses) (количество нейронов в конкурентном слое (количество подклассов))

lr: Learning Rate (скорость обучения);

itermax: Maximum epoch for training (максимальная эпоха обучения);

W: weights for the network (веса для сети);

Синтаксис функции:

[y,classes] = **ann_LVQ_run**(W,P) *// использование НС*

```
clear W;
x = rand(2,10);
x(:,1:5) = x(:,1:5) + 1;
plot(x(1,:),x(2,:),'.');
T = [1 1 1 1 1 0 0 0 0 0;0 0 0 0 0 1 1 1 1 1]
[W,b] = ann_LVQ1(x,T,4);close;
[y,classes] = ann_LVQ_run(W,x)
plot(W(:,1), W(:,2),'or');
```

Рис 1. Обучение нейронной сети с помощью функции «**ann_LVQ1**» и вывод построенных точек в графическое окно

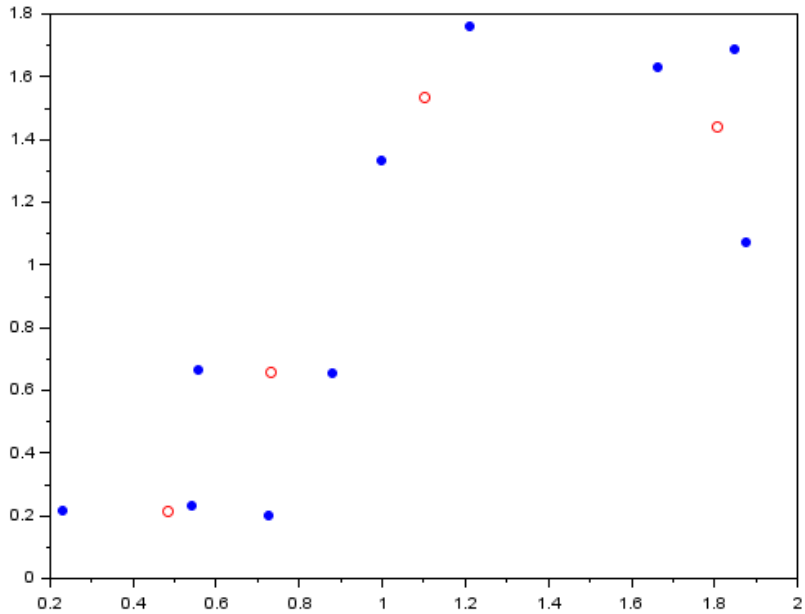


рис 2. Графический результат обучения нейронной сети с помощью функции «**ann_LVQ1**»

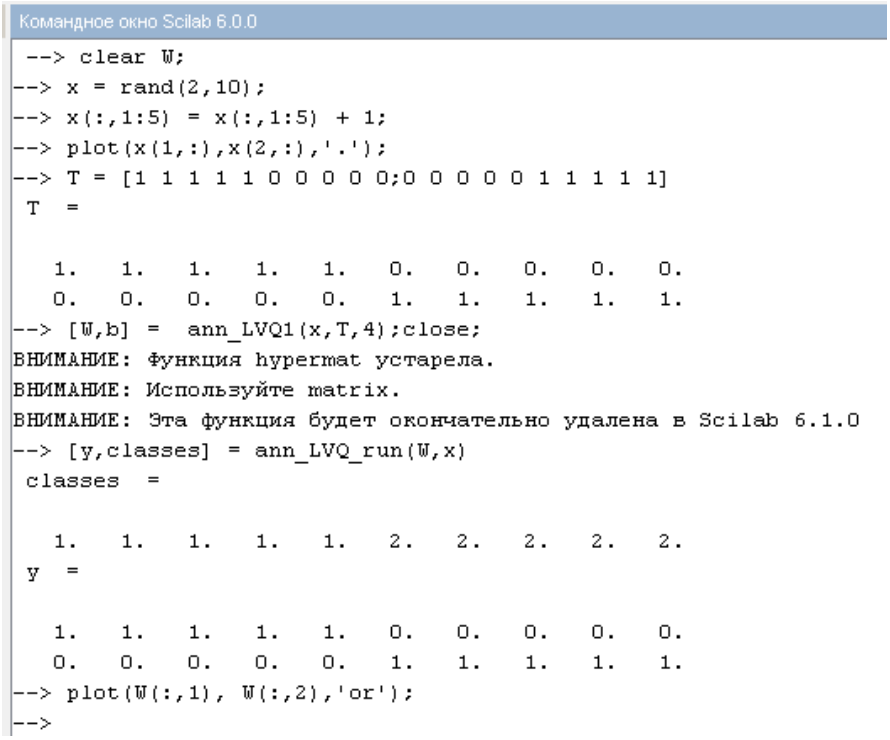


рис 3. Результат обучения нейронной сети с помощью функции «**ann_LVQ1»**

Задание 2. Выполните генерацию сети векторного квантования (LVQ) с использованием функций обучения **ann_LVQ1** и функции использования сети **ann_LVQ_run**. Исходные параметры для моделирования сети: обучающее входное множество ***x*** состоит из 12 значений, заданных датчиком случайных чисел; желаемые выходные классы T разделим три группы; количество нейронов в конкурентном слое – 6.

Задание 3. Выполните генерацию сети векторного квантования (LVQ) с использованием функций обучения **ann_LVQ1** и функции использования сети **ann_LVQ_run**. Исходные параметры для моделирования сети: обучающее входное множество ***x*** состоит из 16 значений, заданных датчиком случайных чисел; желаемые выходные классы T разделим четыре группы; количество нейронов в конкурентном слое – 16.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Основная литература

1. *Ключников, И. К. Финансы. Сценарии развития: учебник для вузов / И. К. Ключников, О. А. Молчанова. — Москва: Издательство Юрайт, 2024. — 206 с. — ISBN 978-5-9916-8768-3. — URL: https://urait.ru/bcode/538115.

Дополнительная литература

1. Чижик, В. П. Финансовые рынки и институты учеб. пособие / В.П. Чижик. — Москва: ФОРУМ: ИНФРА-М, 2022. — 384 с. — ISBN 978-5-00091-452-6. - URL: https://znanium.com/catalog/product/1945432.
2. Путилов, А. В. Коммерциализация технологий и промышленные инновации: учебное пособие / А. В. Путилов, Ю. В. Черняховская. — Санкт-Петербург: Лань, 2022. — 324 с. — ISBN 978-5-8114-3371-1. — URL: https://e.lanbook.com/book/213212.
3. Финансы [Текст]: учебник / под ред. А.М. Ковалевой.- 6-е изд., перераб. и доп.- М.: Издательство Юрайт, 2016.- 443с.- ISBN 978-59916-2806-8.
4. Финансы. В 2-х ч. Ч.1, Ч.2 [Текст]: учебник для академического бакалавриата / под ред. М.В. Романовского, Н.Г. Ивановой.-5-е изд., перераб. и доп.- М.: Издательство Юрайт, 2016.- ISBN 978-5-9916-8656-3.
5. Галушкин, А. И. Нейронные сети: основы теории: учебное пособие / А. И. Галушкин. - Москва: Горячая линия-Телеком, 2012. - 496 с. - ISBN 978-5-9912-0082-0. - Текст: электронный // Лань: электронно-библиотечная система. - URL: https://e.lanbook.com/book/5144.
6. Сайт матричной лаборатории SciLab. - URL: https://www.scilab.org/