Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
Учреждение высшего образования
«Севастопольский государственный университет»

Отчет

По лабораторной работе

Исследование адаптивного линейного элемента

Дисциплина «Основы нейронных сетей»

Выполнил студент:
Ольховская А.С
Группа: ГНКЗ – 7

Севастополь
2023

1 ЦЕЛЬ РАБОТЫ

Углубление теоретических знаний в области разработки нейронных сетей с линейной активационной функцией, исследование свойств квадратичной целевой функции и LMS-алгоритма обучения, приобретение практических навыков обучения однослойной сети линейных адаптивных элементов при решении задачи классификации и адаптивной фильтрации.

2 ПОСТАНОВКА ЗАДАЧИ

2.1. Повторите теоретический материал, относящийся к архитектуре и правилам обучения адаптивного линейного элемента [1, 4, 5].

2.2. Даны четыре класса, каждый из которых представлен 2-мя примерами (столбцами матрицы P), указанными в таблице 3.2 к лабораторной работе №3 (используйте свой вариант). Необходимо:

- разработать структурную схему классификатора на основе АЛЭ, распознающего эти 4 класса;

- выполнить предварительный анализ задачи, изобразив точки четырех классов и построив графически возможные границы решений;

- задать ту же целевую матрицу T, которая использовалась при выполнении задания 3.3 в лабораторной работе №3, заменив все нули на -1;

- полагая, что все входные векторы p равновероятны, написать программу, вычисляющую корреляционную матрицу C, собственные числа гессиана целевой функции A=2C и максимальное устойчивое значение параметра αmax LMS-алгоритма;

- изучить встроенные функции ann_ADALINE и ann_ADALINE_online пакета NeuralNetworks 3.0, реализующие блочный и последовательный варианты LMS-алгоритма;

- используя указанные функции, написать программу, которая:

- отображает диаграмму размещения входных точек (примеров) из P на плоскости с координатами (p1, p2);

- обучает АЛЭ правильному распознаванию входных классов с использованием 2-х указанных функций обучения при разных значениях параметра α;

- строит кривые обучения - зависимости СКО от номера эпохи для 2-х указанных функций обучения (для этого необходимо модифицировать встроенные функции ann_ADALINE и ann_ADALINE_online);

- накладывает на диаграмму входных точек данных границы решения после обучения слоя АЛЭ;

- выполняет тестирование полученного решения для всех заданных входных данных;

- сравнить получаемые границы решения слоя АЛЭ с границами решения персептрона, полученными в лабораторной работе №3, обратив внимание на равноудаленность границ от точек соседних классов для случая слоя АЛЭ;

- сравнить кривые обучения слоя АЛЭ двумя модифицированными функциями ann_ADALINE и ann_ADALINE_online для случая, когда параметр α LMS-алгоритма значительно меньше αmax и когда он близок к αmax .

Таблица 1 – Данные

Вариант	Класс 1 P(:,1), P(:,2),	Класс 2 P(:,3), P(:,4),	Класс 3 P(:,5), P(:,6),	Класс 4 P(:,7), P(:,8),
6	[1; 2], [2; 2]	[-1; 2], [-2; 2]	[-1; -1], [-2; -1]	[1; -1], [2; -1]

2.3. Исследуйте адаптивный линейный предсказатель (рисунок 4.4):

- сгенерируйте входной y(k) и желаемый t(k)=y(k) сигналы адаптивного предсказателя. При этом параметры генератора выбирайте в соответствии с вариантом.

Таблица 2 – Данные

Вариант	Число составляющих L	Основная частота F	Частота среза Fc
6	5	0.02	F*5

- запишите развернутые выражения для всех элементов (C,h,c) целевой функции предсказателя, заданной в виде СКО (4.12);

- вычислите конкретные значения матрицы C, вектора h и константы c для сгенерированного входного сигнала y(k);

- вычислите собственные значения и собственные векторы матрицы Гессе целевой функции предсказателя, точку минимума целевой функции, постройте линии контуров равных уровней целевой функции;

- вычислите максимальное устойчивое значение скорости обучения αmax для LMS-алгоритма;

- напишите программу, обучающую предсказатель с использованием встроенной функции ann_ADALINE_predict пакета NeuralNetworks 3.0;

- модифицируйте функцию ann_ADALINE_predict таким образом, чтобы по результатам её работы можно было построить кривую обучения предсказателя и траекторию движения вектора параметров предсказателя на диаграмме контуров равных уровней;

- постройте в одном графическом окне графики входного процесса и его предсказанных значений, кривую обучения, траекторию движения вектора параметров на диаграмме контуров;

- убедитесь, что алгоритм обучения сходится, если α_{max};

- убедитесь, что при малых α траектория движения вектора параметров при использовании LMS алгоритма аппроксимирует в среднем траекторию движения вектора параметров алгоритма наискорейшего спуска.

3 ХОД РАБОТЫ

3.1 Разработана структурная схема классификатора на основе АЛЭ, распознающего 4 класса (рисунок 1). Выполнен предварительный анализ задачи, где изображены точки четырех классов и построены графически возможные границы решений. Задана та же целевая матрица T, которая использовалась при выполнении задания 3.3 в лабораторной работе №3, где заменены все нули на -1.

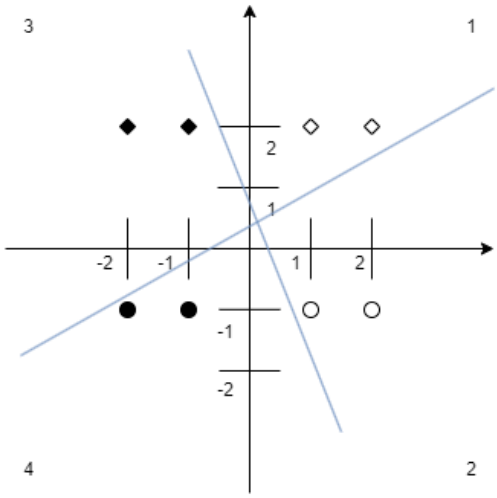

Рисунок 1 – Структурная схема

Матрица T для всех входных точек, представленных столбцами матриц данных P:

T = [-1 -1 -1 -1 1 1 1 1;

-1 -1 1 1 -1 -1 1 1]

Написана программа, вычисляющая корреляционную матрицу C, собственные числа гессиана целевой функции A=2C и максимальное устойчивое значение параметра α_{max} LMS-алгоритма (рисунок 2).

Листинг 1

```
P = [1 2;
2 2;
-1 2;
-2 2;
```

```
-1 -1;
-2 -1;
1 -1;
2 -1];

Q = size(P, 2);
R = size(P, 1) + 1;
Cor = zeros(R, R);
for q = 1:Q
 z = [P(:,q); 1];
 Cor = Cor + (z*z');
end
Cor = 1 / Q.* Cor;
evals = spec(2 * Cor);
alpha_max = 1 / max(evals);
disp("Корреляционная матрица C:");
disp(Cor);
disp("Собственные числа гессиана A = 2C:");
disp(evals);
disp("Максимальное устойчивое значение параметра alpha_max LMS-алгоритма:");
disp(alpha_max);
```

```
--> exec('C:\Users\anast\alea.sce', -1)

 "Корреляционная матрица C:"

  2.5   3.   1.5   1.  -1.5  -2.   -0.5   0.    1.5
  3.    4.   1.    0.  -2.   -3.    0.    1.    2.
  1.5   1.   2.5   3.  -0.5   0.   -1.5  -2.    0.5
  1.    0.   3.    4.   0.    1.   -2.   -3.    0.
 -1.5  -2.  -0.5   0.   1.    1.5   0.   -0.5  -1.
 -2.   -3.   0.    1.   1.5   2.5  -0.5  -1.5  -1.5
 -0.5   0.  -1.5  -2.   0.   -0.5   1.    1.5   0.
  0.    1.  -2.   -3.  -0.5  -1.5   1.5   2.5   0.5
  1.5   2.   0.5   0.  -1.   -1.5   0.    0.5   1.

 "Собственные числа гессиана A = 2C:"

 -3.582D-15
 -5.972D-16
 -1.360D-16
 -1.036D-17
  4.043D-16
  1.314D-15
  3.456D-15
  20.000000
  22.000000

 "Максимальное устойчивое значение параметра alpha_max LMS-алгоритма:"

  0.0454545
```

Рисунок 2 – Выполнение программы

Изучены встроенные функции ann_ADALINE и ann_ADALINE_online пакета NeuralNetworks 3.0, реализующие блочный и последовательный варианты LMS-алгоритма.

Написана программа, которая использует указанные функции, а также:

- отображает диаграмму размещения входных точек (примеров) из P на плоскости с координатами (p1, p2);

- обучает АЛЭ правильному распознаванию входных классов с использованием 2-х указанных функций обучения при разных значениях параметра α;

- строит кривые обучения - зависимости СКО от номера эпохи для 2-х указанных функций обучения (для этого необходимо модифицировать встроенные функции ann_ADALINE и ann_ADALINE_online);

- накладывает на диаграмму входных точек данных границы решения после обучения слоя АЛЭ;

- выполняет тестирование полученного решения для всех заданных входных данных;

- сравнить получаемые границы решения слоя АЛЭ с границами решения персептрона, полученными в лабораторной работе №3, обратив внимание на равноудаленность границ от точек соседних классов для случая слоя АЛЭ;

- сравнить кривые обучения слоя АЛЭ двумя модифицированными функциями ann_ADALINE и ann_ADALINE_online для случая, когда параметр α LMS-алгоритма значительно меньше αmax и когда он близок к αmax .

Листинг программы 2

```
function [w, b, mse1]=ann_ADALINE1(P, T, alpha, itermax, initfunc)
rhs=argn(2); // argn(2) – возвращает число аргументов функции
if rhs < 2; error("Должно быть минимум 2 аргумента: P и T"); end
if rhs < 3; alpha = 0.01; end
if rhs < 4; itermax = 100; end
if rhs < 5; w = rand(size(T,1),size(P,1)); b = rand(size(T,1),1); end
if rhs == 5 then
 select initfunc
 case 'rand' then
```

```
 w = rand(size(T,1),size(P,1));
 b = rand(size(T,1),1);
 case 'zeros' then
 w = zeros(size(T,1),size(P,1));
 b = zeros(size(T,1),1);
 case 'ones' then
 w = ones(size(T,1),size(P,1));
 b = ones(size(T,1),1);
 else
    error("Неверное значение входного аргумента 5");
 end
end
if itermax == []; itermax = 100; end
if alpha == []; alpha = 0.01; end
itercnt = 0; // Счетчик итераций - эпох
mse1=zeros(1,itermax);
while itercnt < itermax

 n = w*P + repmat(b,1,size(P,2));
 a = ann_purelin_activ(n);
 e = T - a;

 w = w + (2*alpha*e*P')./size(P,2);
 b = b + 2*alpha*mean(e,2);

 n = w*P + repmat(b,1,size(P,2));
 a = ann_purelin_activ(n);
 e = T - a;

 itercnt = itercnt + 1;
 mse1 = mean(e.^2);
end
endfunction


function [w, b, mse2]=ann_ADALINE1_online(P, T, alpha, itermax, initfunc)
rhs=argn(2); // argn(2) – возвращает число аргументов функции
if rhs < 2; error("Должно быть минимум 2 аргумента: P и T"); end
if rhs < 3; alpha = 0.01; end
if rhs < 4; itermax = 100; end
if rhs < 5; w = rand(size(T,1),size(P,1)); b = rand(size(T,1),1); end
if rhs == 5 then
 select initfunc
```

```
 case 'rand' then
 w = rand(size(T,1),size(P,1));
 b = rand(size(T,1),1);
 case 'zeros' then
 w = zeros(size(T,1),size(P,1));
 b = zeros(size(T,1),1);
 case 'ones' then
 w = ones(size(T,1),size(P,1));
 b = ones(size(T,1),1);
 else
 error("Неверное значение входного аргумента 5");
 end
end
if itermax == []; itermax = 100; end
if alpha == []; alpha = 0.01; end

itercnt = 0; // Счетчик итераций - эпох
mse2=zeros(1,itermax);
while itercnt < itermax
 for cnt = 1:size(P,2) // Цикл по всем обучающим примерам - 1 эпоха

 e = T(:,cnt) - ann_purelin_activ(w*P(:,cnt)+b);

 w = (w + 2*alpha*e*P(:,cnt)');
 b = b + 2*alpha*e;

 e_all(cnt) = e.^2;
 end
 itercnt = itercnt + 1;
 mse2(itercnt)=mean(e_all);
end
endfunction

P = [1 2 1 2 -1 -2 -1 -2;
     2 2 -1 -1 2 2 -1 -1];


T = [-1 -1 -1 -1 1 1 1 1]


maxiter = 40;
alpha = 0.01;
[w,b,mse1] = ann_ADALINE1(P,T,alpha,maxiter,'ones');
```

```
[w2,b2,mse2] = ann_ADALINE1_online(P,T,alpha,maxiter,'ones');

clf(1, "reset");
scf(1);
x_axis=1:maxiter;
plot(x_axis, mse1(1:min(maxiter, length(mse1))), 'r', x_axis,
mse2(1:min(maxiter, length(mse2))), 'g');
xtitle('Средний квадрат ошибки', 'Эпоха','СКО');
xgrid;
legend('batch', 'online', 1, %t);
```

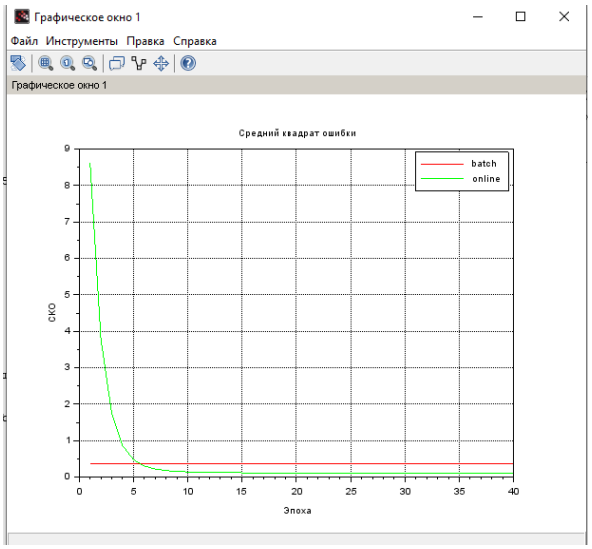

Рисунок 3 – Кривые обучения при a = 0.01

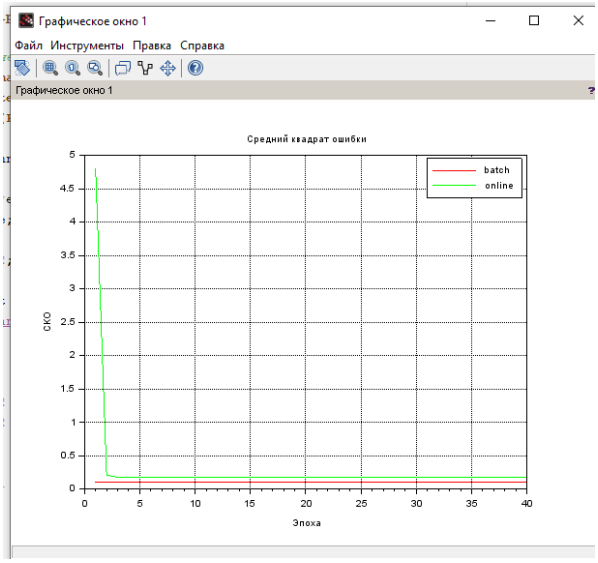

Рисунок 4 – Кривые обучения при a = 0.04545

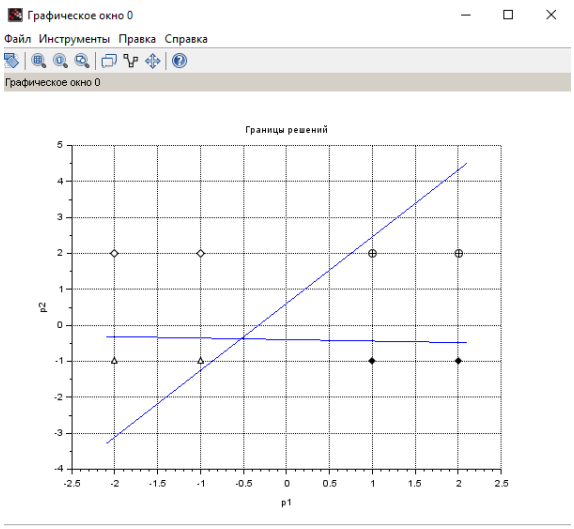

Рисунок 5 – Отображение размещения входных точек из P на плоскости

3.2 Часть 1.

Написан код программы для генерирования входной y(k) и желаемый t(k)=y(k) сигналы адаптивного предсказателя: Программа:

- записывает развернутые выражения для всех элементов (C,h,c) целевой функции предсказателя, заданной в виде СКО (4.12);

- вычисляет конкретные значения матрицы C, вектора h и константы c для сгенерированного входного сигнала y(k);

- вычисляет собственные значения и собственные векторы матрицы Гессе целевой функции предсказателя, точку минимума целевой функции, постройте линии контуров равных уровней целевой функции;

- вычисляет максимальное устойчивое значение скорости обучения αmax для LMS-алгоритма

Листинг программы 3

```
L= 5;
F= 0.02;
Fc= F*5;
//Генератор полигармонического входного сигнала;
td=1/(20*F);
t=0:td:2/F;
fi=(2*%pi*F).*t;
X=0;
for i=1:L
 X=X+(Fc)/(Fc+2*%pi*F*i)*sin(fi*i);
```

```
end
Y=X(2:$) // Y задержен на такт - вход предсказателя
T=X(1:$-1); // желаемый выход предсказателя
D = 2; // кол-во элементов задержки предсказателя
P = [];
for cnt = 1:D // для каждого выхода линии задержки
 // формируем строки матрицы P из отсчетов Y
 // очередная строка P – сдвинутая на один отсчет копия предыдущей строки
P = [P; Y(cnt:$-D+cnt-1)];
end
//формируем вектор целевых значений с длиной, равной длине строки из P
T1 = T(1:$-D+1);
// Построение графика формирования матрицы входных данных
figure;
plot(P');
title('Формирование матрицы входных данных');
xlabel('Отсчеты');
ylabel('Значения');
legend('p1', 'p2');
```

Результат:

```
Y
 Y  =

         column 1 to 12

    0.7689632   0.8280926   0.4591588   0.3077139   0.3707923   0.2887279
0.1128166   0.0916198   0.1187326  -2.012D-16  -0.1187326  -0.0916198

         column 13 to 24

   -0.1128166  -0.2887279  -0.3707923  -0.3077139  -0.4591588  -0.8280926
-0.7689632   1.989D-15   0.7689632   0.8280926   0.4591588   0.3077139

         column 25 to 36

    0.3707923   0.2887279   0.1128166   0.0916198   0.1187326   6.809D-16
-0.1187326  -0.0916198  -0.1128166  -0.2887279  -0.3707923  -0.3077139

         column 37 to 40

   -0.4591588  -0.8280926  -0.7689632   3.977D-15
```

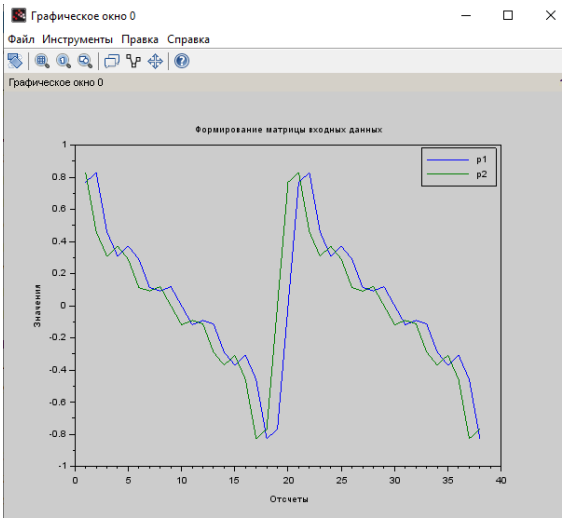

Рисунок 6 – Формирование матрицы входных данных

Часть 2

Написана программа, обучающая предсказатель с использованием встроенной функции ann_ADALINE_predict пакета NeuralNetworks 3.0.

Модифицирована функция ann_ADALINE_predict таким образом, что по результатам её работы можно построить кривую обучения предсказателя и траекторию движения вектора параметров предсказателя на диаграмме контуров равных уровней;

Построены в одном графическом окне графики входного процесса и его предсказанных значений, кривую обучения, траекторию движения вектора параметров на диаграмме контуров (рисунок 7).

Листинг программы 4

```
function [w, b, y, ee,mse,W]=ann_ADALINE1_predict(P, T, alpha, itermax, D, initfunc)
rhs=argn(2); // argn(2) - возвращает число аргументов функции
if rhs < 2; error("Должно быть минимум 2 аргумента: P и T"); end
if rhs < 3; alpha = 0.01; end
if rhs < 4; itermax = 100; end
if rhs < 5; D = 1; end
if rhs < 6; w = rand(size(T,1),D); b = rand(size(T,1),1); end
//Инициализация весов и смещений
if rhs == 6 then
 select initfunc
 case 'rand' then
```

```
 w = rand(size(T,1),D);
 b = rand(size(T,1),1);
 case 'zeros' then
 w = zeros(size(T,1),D);
 b = zeros(size(T,1),1);
 case 'ones' then
 w = ones(size(T,1),D);
 b = ones(size(T,1),1);
 else
 error("Неверное значение входного аргумента 5");
 end
end
if itermax == []; itermax = 100; end
if alpha == []; alpha = 0.01; end
if D == []; D = 1; end
// Формирование выходов линии задержки: реформатирование X в матрицу P
P = [];
for cnt = 1:D // для каждого выхода линии задержки
 //формируем строки матрицы P из отсчетов X
 // очередная строка P – сдвинутая на один отсчет копия предыдущей строки
 P = [P; X(cnt:$-D+cnt-1)];
end
//формируем вектор целевых значений
T = T(1:$-D+1);
//2. ====== Реализация правил обучения ==========================
mse=zeros(1,itermax); // создаем вектор СКО
W=[w]; //матрица, каждая строка которой – вектор весов на очередном шаге
itercnt = 0; // Счетчик итераций - эпох
while itercnt < itermax
 for cnt = 1:size(P,2) // Цикл по всем обучающим примерам из P (1 эпоха)
 n = w*P(:,cnt)+b; // Сетевая функция АЛЭ
 a = ann_purelin_activ(n);
 e = T(:,cnt) - a; // Ошибка
 y(cnt) = a; // Запоминаем выход АЛЭ
 ee(cnt) = e; // Запоминаем ошибку
 //Реализуем правило обучения Уидроу-Хоффа
 //для каждого примера P(:,cnt) вычисляем обновление w и b
 w = (w + 2*alpha*e*P(:,cnt)');

 // Вычисляем и запоминаем квадраты текущих ошибок
 e_all(cnt) = e.^2;
W=[W;w];
 end
```

```
 itercnt = itercnt + 1;
 mse(itercnt)=mean(e_all);
end
endfunction
D = 2; // кол-во элементов задержки предсказателя
P = [];

L = 5;
F = 0.02;
Fc= F*5;

//Генератор полигармонического входного сигнала;
td=1/(20*F);
t=0:td:2/F;
fi=(2*%pi*F).*t;
X=0;
for i=1:L
    X=X+(Fc)/(Fc+2*%pi*F*i)*sin(fi*i);
end

Y=X(2:$) // Y задержен на такт - вход предсказателя
T=X(1:$-1); // желаемый выход предсказателя

for cnt = 1:D // для каждого выхода линии задержки
 //формируем строки матрицы P из отсчетов Y
 // очередная строка P – сдвинутая на один отсчет копия предыдущей строки
 P = [P; Y(cnt:$-D+cnt-1)];
end
//формируем вектор целевых значений с длиной, равной длине строки из P
T1 = T(1:$-D+1);

Q = size(P, 2); // кол-во обучающих примеров
R = size(P, 1); // кол-во входов АЛЭ
Cor = zeros(R, R); // корреляционная матрица
c = 0;
h = 0;
for q = 1:Q
 p = P(:, q); // q-ый входной вектор
 Cor = Cor + (p * p').*1 / Q;
 c = c + T1(q)^2 * 1 / Q;
 h = h + T1(q) * p * 1 / Q;
end
[evals, diagevals] = spec(2 * Cor);
```

```
xstar = inv(Cor) * h; //решение Винера
function z=f(w1, w2, c, h, Cor)
 x = [w1; w2];
 z = c - 2 * x' * h + x' * Cor * x;
endfunction
x = linspace(-20, 20, 100); // диапазон по оси x
y = linspace(-20, 20, 100); // диапазон по оси y
z = feval(x, y, f); //вычисляем значения высот целевой функции f на сетке x,y
clf(1);
figure(1);
subplot(1, 2, 1);
surf(x, y, z'); //строим 3D поверхность целевой функции
subplot(1, 2, 2);
contour2d(x, y, z, 30); //отображаем линии контуров равных уровней
xset("fpf", " ") //подавляет отображение чисел над линиями контуров
xtitle("Контуры равных уровней квадратичной функции", "w1", "w2");
xgrid;
plot(xstar(1), xstar(2), '*b'); //отображаем точку решения минимума СКО (4.15
// Создание и обучение предсказателя по значениям P и T
// Используем модифицированную функцию предсказателя

[w, b, y, ee, mse, W] = ann_ADALINE1_predict(P, T, 0.1 * alpha_max, 40, D,'ones');
plot2d(W(:, 1), W(:, 2), 5);
```

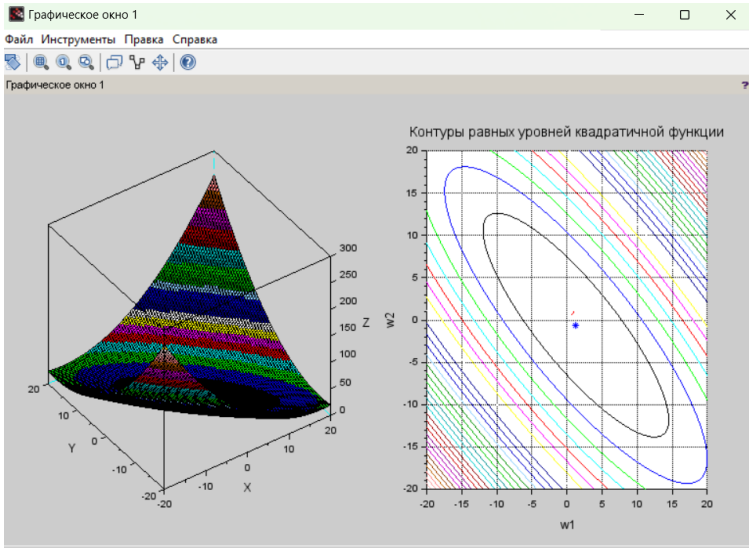

Рисунок 7 – Поверхность целевой функции, линии контуров и траектория движения вектора параметров для LMS-алгоритма

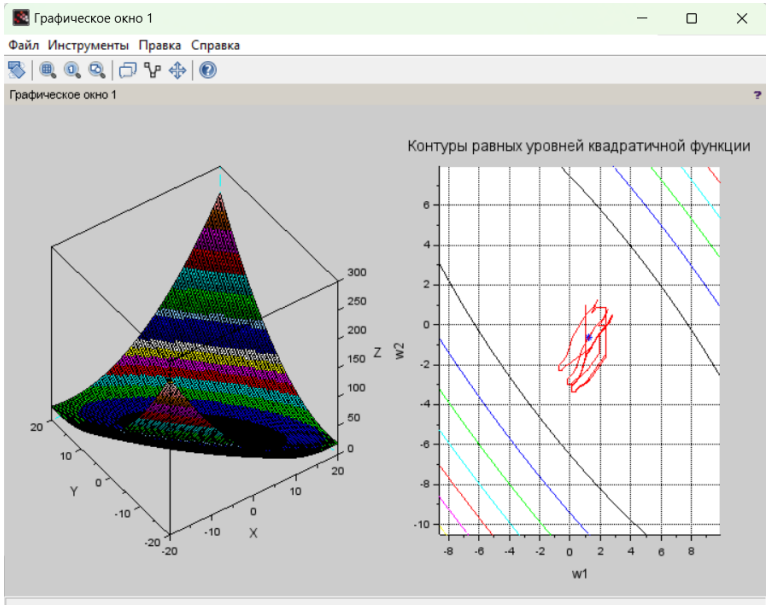

Рисунок 8 – Поверхность целевой функции, линии контуров и траектория движения вектора параметров для LMS-алгоритма при alpha = 0.96

4 ВЫВОД

В ходе лабораторной работы были углублены теоретические знания в области разработки нейронных сетей с линейной активационной функцией, исследование свойств квадратичной целевой функции и LMS-алгоритма обучения. Были приобретены практические навыки обучения однослойной сети линейных адаптивных элементов при решении задачи классификации и адаптивной фильтрации.

На графике (рисунок 8) происходит колебательные движения около точки минимума функции. Это обусловлено тем, что значение α имеет большое значение, которое мы не уменьшаем.