ФГАОУ ВО «Севастопольский государственный университет»

Институт информационных технологий   
и управления в технических системах

ОТЧЁТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

«Исследование адаптивного линейного элемента»

по дисциплине «Нейрокомпьютерные технологии»

Выполнила:  
студентка группы ИС/м-21-1-з  
Ускова Екатерина Дмитриевна

Севастополь

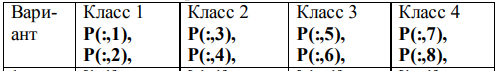
2022

# ЦЕЛЬ РАБОТЫ

Углубление теоретических знаний в области архитектуры 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.

# Ход работы

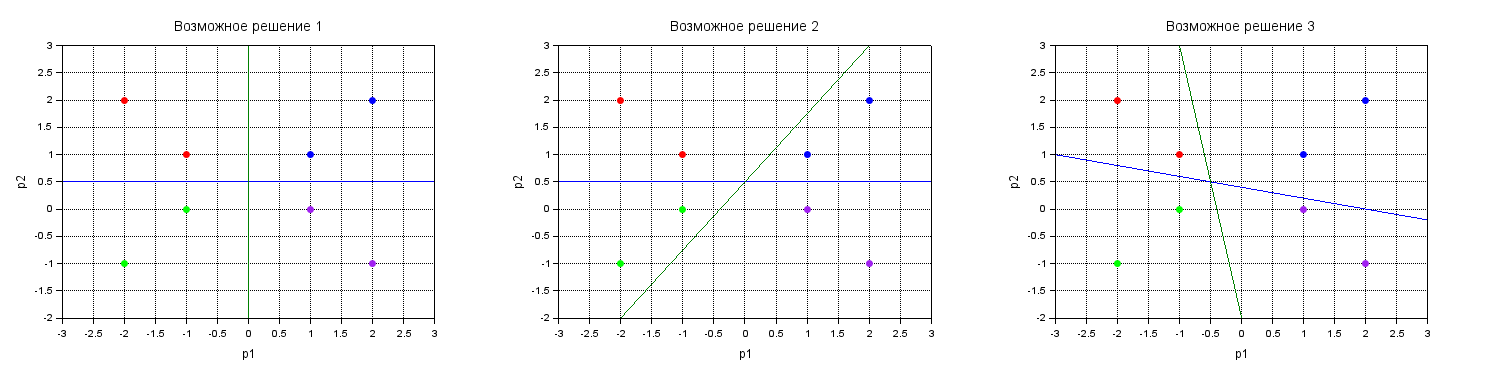
Даны четыре класса, каждый из которых представлен 2-мя точками (столбцами матрицы P), указанными в таблице.





*Разработана структурная схема классификатора на основе АЛЭ, распознающего эти 4 класса.*

Выполнен предварительный анализ задачи, изображены точки четырех классов и построены возможные границы решений.



Задана целевую матрицу T, которая использовалась при выполнении лабораторной работы №3, в которой все нули заменены на -1.

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

1 1 1 1 -1 -1 -1 -1]'

Полагая, что все входные векторы p равновероятны, написана программа, вычисляющая корреляционную матрицу R, собственные числа гессиана целевой функции A=2R и максимальное устойчивое значение параметра αmax LMS-алгоритма.

Z = [P; 1 1 1 1 1 1 1 1]

*// корреляционная матрица*

Q = size(Z, 2)

R = zeros(3, 3)

for q=1:Q

R = R + Z(:,q) \* Z(:,q)'

end

R = 1 / Q \* R

disp(R)

*// собственные числа гессиана*

A = 2 \* R

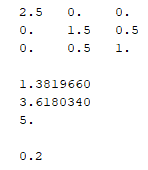
evals=spec(A)

disp(evals)

*// максимальное устойчивое значение (alpha < 1 / lambda\_max)*

alpha\_max = 1 / max(evals)

disp(alpha\_max)



Изучены встроенные функции ann\_ADALINE и ann\_ADALINE\_online пакета NeuralNetworks 2.0, реализующие блочный и последовательный варианты LMS-алгоритма.

Используя указанные функции, написана программа, которая:

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

clf

*// отрисовка входных данных*

scatter(P(1,1:2), P(2,1:2), , "blue", 0)

scatter(P(1,3:4), P(2,3:4), , "red", 0)

scatter(P(1,5:6), P(2,5:6), , "scilabgreen2", 0)

scatter(P(1,7:8), P(2,7:8), , "purple", 0)

zoom\_rect([-3 -2 3 3])

xgrid

xtitle("Входные данные", "p1", "p2")

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

maxiter = 50

alpha = 0.19

[w1,b1,mse1] = ann\_ADALINE1(P,T,alpha,maxiter,'zeros');

[w2,b2,mse2] = ann\_ADALINE1\_online(P,T,alpha,maxiter,'zeros');

*-* строит кривые обучения - зависимости СКО от номера эпохи для 2-х указанных функций (для этого необходимо модифицировать встроенные функции ann\_ADALINE и ann\_ADALINE\_online);

clf

x\_axis=1:maxiter;

plot(x\_axis,mse1(x\_axis),'r',x\_axis,mse2(x\_axis),'g');

xtitle('Средний квадрат ошибки', 'Эпоха','СКО');

legend('batch','online')

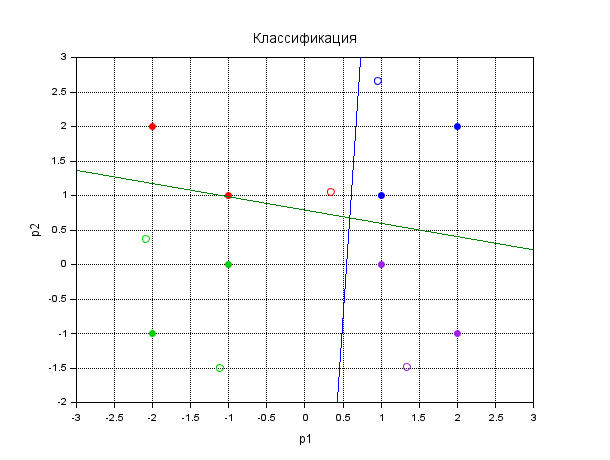
xgrid()

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

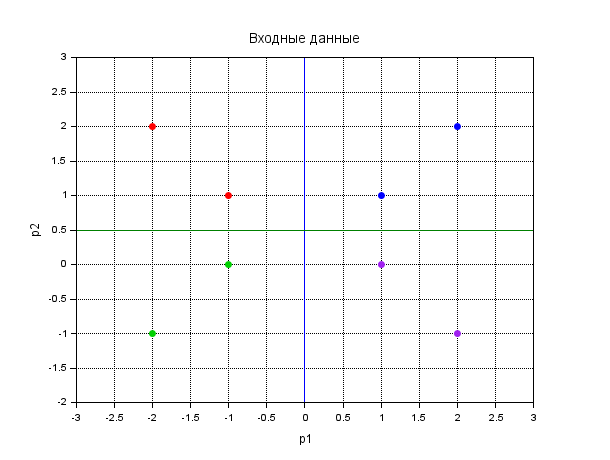
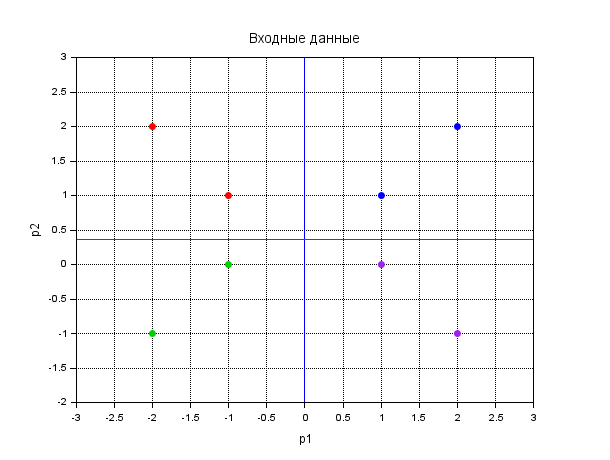
*- выполняет тестирование полученного решения для всех заданных входных данных;*

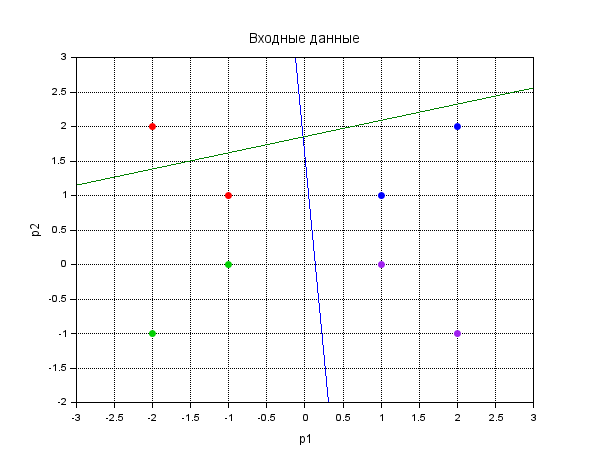
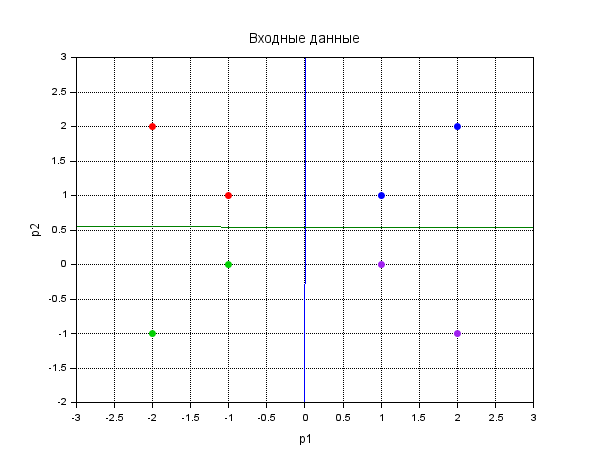
Выполнено сравнение получаемых границ решения слоя АЛЭ с границами решения персептрона, полученными в лабораторной работе №3.

Границы решения персептрона:



Границы решения слоя АЛЭ (слева alpha = 0.02, справа 0.19 и 0.12. Верх batch, низ online:

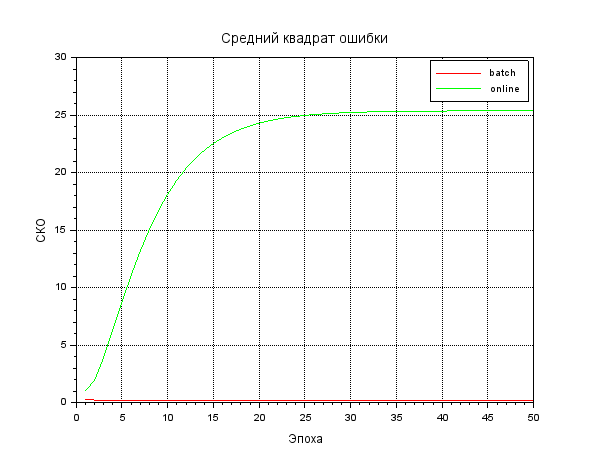
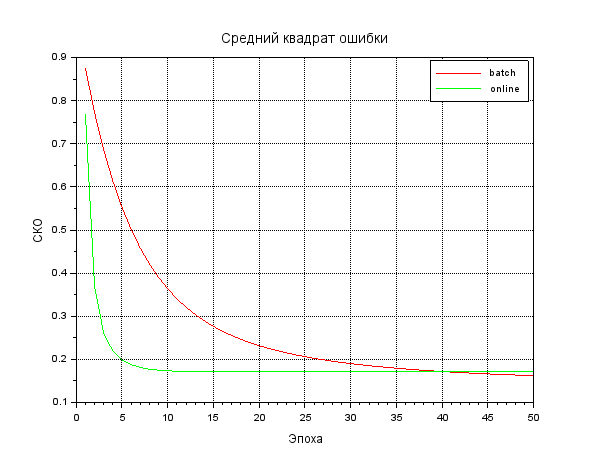




Как видно из графиков, LMS-алгоритм находит границы решения, равноудаленные от центров соседних классов, и получаемое решение не зависит от способа инициализации весов слоя АЛЭ. В то же время слой персептронов формировал различные границы решения при разных начальных значениях весов.

Выполнено сравнение кривых 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.

Сравнение кривых обучения: слева α=0,02; справа α=0,19



Блочный LMS-алгоритм при малых α требует для обучения большего числа эпох, чем последовательный алгоритм, но в итоге дает более точное решение. Последовательный LMS-алгоритм при малых α обучается быстрее, но при значении α = αmax теряет устойчивость.

Исследован адаптивный линейный предсказатель.

- сгенерирован входной y(k) и желаемый t(k)=y(k) сигналы адаптивного предсказателя. При этом параметры генератора выбраны в соответствии с вариантом (4).

*//Значения параметров генератора L, F, Fc выбирайте из таблицы 4.1;*

*//Параметры:*

L=20;

F=0.01;

Fc=F\*8;

*//Генератор полигармонического входного сигнала;*

td=1/(20\*F);

t=0:td:2/F;

fi=(2\*%pi\*F).\*t;

Y=0;

for i=1:L

Y=Y+(Fc)/(Fc+2\*%pi\*F\*i)\*sin(fi\*i);

end

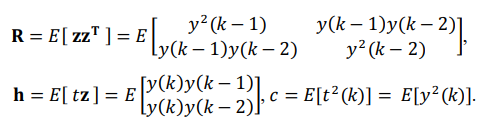
T=Y;





- записаны развернутые выражения для всех элементов (R,h,c) целевой функции предсказателя, заданной в виде СКО.

Чтобы применить функции



нужно сначала получить значения векторa p, элементы которого соответствуют выходам линии задержки.

Для формирования матрицы P, каждый столбец которой равен очередному входному вектору p АЛЭ, использована начальная часть кода встроенной функции ann\_ADALINE\_predict, предназначенной для обучения адаптивного линейного предсказателя в последовательном режиме на основе LMS-алгоритма:

*// Получение значений вектора p*

T=Y;

D=2;

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 c длиной, равной длине строки из P*

T = T(1:$-D+1);

- вычислены конкретные значения матрицы R, вектора h и константы с для сгенерированного входного сигнала y(k) ;

Q = size(P, 2)

R = zeros(2, 2)

for q=1:Q

R = R + P(:,q) \* P(:,q)'

end

R = 1 / Q \* R

printf('R: ')

disp(R)

h = zeros(2, 1)

for q=1:Q-1

h = h + P(:,q) \* P(1,q+1)

end

h = 1 / Q \* h

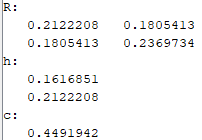
printf('h: ')

disp(h)

c = 1 / Q \* sum(P .^ 2)

printf('c: ')

disp(c)



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

*// собственные числа и собственные векторы гессиана*

A = 2 \* R

[evals, diagevals]=spec(A)

printf('evals: ')

disp(evals)

printf('diagevals: ')

disp(diagevals)

*// максимальное устойчивое значение (alpha < 1 / lambda\_max)*

*//alpha\_max = 1 / max(evals)*

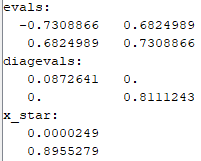
*//disp(alpha\_max)*

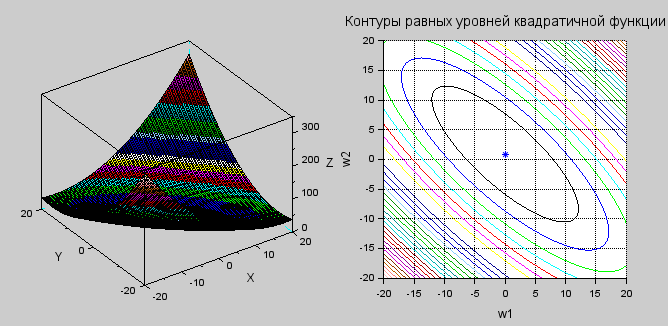
*// точка минимума*

x\_star = inv(R) \* h

printf('x\_star: ')

disp(x\_star)





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

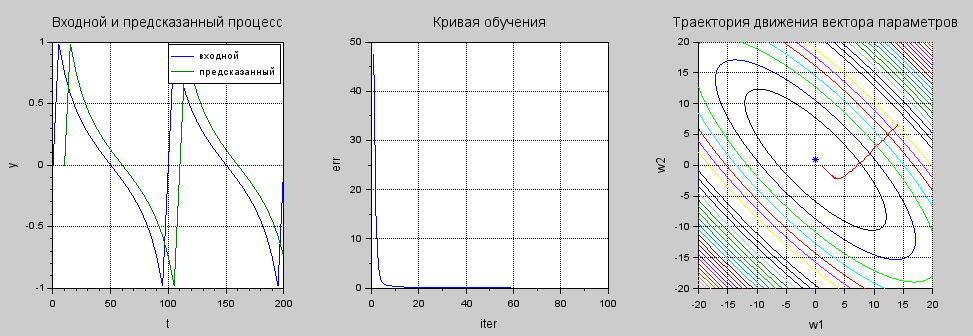
alpha\_max = 1 / max(evals)



- написана программа, обучающая предсказатель с использованием встроенной функции ann\_ADALINE\_predict пакета NeuralNetworks 2.0;

- модифицирована функция 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;

- построены графики входного процесса и его предсказанных значений, кривая обучения, траектория движения вектора параметров на диаграмме контуров;

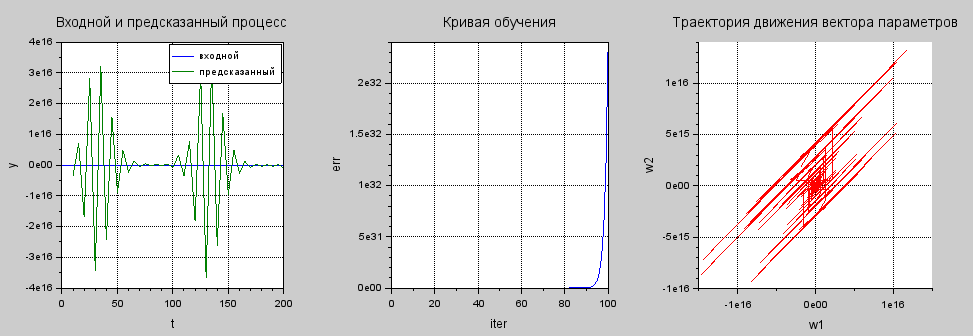


- алгоритм обучения сходится, если α<αmax и нестабилен когда α>αmax;

Alpha=0.71:



Если 0.75:



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

# Вывод

Выполняя эту работу, я углубила теоретические знания в области архитектуры нейронных сетей с линейной активационной функцией, исследовала свойства квадратичной целевой функции и LMS-алгоритма обучения, приобрела практические навыки обучения однослойной сети линейных адаптивных элементов при решении задачи классификации и адаптивной фильтрации.