|  |  |  |  |
| --- | --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования  Белорусский Государственный Университет Информатики и Радиоэлектроники | | | |
| Факультет | Компьютерных сетей и систем | | |
| Кафедра | Информатики  Дисциплина: Конструирование те технологии электронных вычислительных средств | | |
|  |  | | |
| **Отчет ПО ЛАБОРАТОРНым РАБОТАМ**  по курсу «Нейросетевое моделирование и обработка данных» | | | |
| Студент:  гр. 758641  Ярош Г.И. | |  | Проверил:  Жвакина А.В. |
| Минск, 2018 | | | |

СОДЕРЖАНИЕ

1 GUI-ИНТЕРФЕЙС ДЛЯ ПАКЕТА NEURAL NETWORKS TOOLBOX ПРОГРАММНОЙ СРЕДЫ MATLAB . НАЗНАЧЕНИЕ И ОБЗОР 3

1.1 Цель 3

1.2 Ход работы 3

1.3 Вывод 6

2 ИЗУЧЕНИЕ СВОЙСТВ ЛИНЕЙНОГО НЕЙРОНА И ЛИНЕЙНОЙ НЕЙРОННОЙ СЕТИ 7

2.1 Цель 7

2.2 Ход работы 7

2.3 Вывод 13

3 ИЗУЧЕНИЕ МНОГОСЛОЙНОГО НЕЛИНЕЙНОГО ПЕРСЕПТРОНА И АЛГОРИТМА ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ 14

3.1 Цель 14

3.2 Ход работы 14

3.1 Вывод 16

4 ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ АППРОКСИМАЦИИ ФУНКЦИЙ И ПРЕДСКАЗАНИЯ ВРЕМЕННОГО ПРОЦЕССА 17

4.1 Цель 17

4.2 Ход работы 17

4.3 Вывод 21

# GUI-ИНТЕРФЕЙС ДЛЯ ПАКЕТА NEURAL NETWORKS TOOLBOX ПРОГРАММНОЙ СРЕДЫ MATLAB . НАЗНАЧЕНИЕ И ОБЗОР

## Цель

Изучить основные свойства и основы работы с GUI – интерфейсом пакета Neural Networks Toolbox в программной среде MatLab. Создать, используя графический интерфейс пользователя, нейронную сеть для аппроксимации функции:

## Ход работы

Первым этапом были заданы векторы значений , , значений функции . Векторы входным значений были объединены в матрицу входных значений (рис. 1.1).

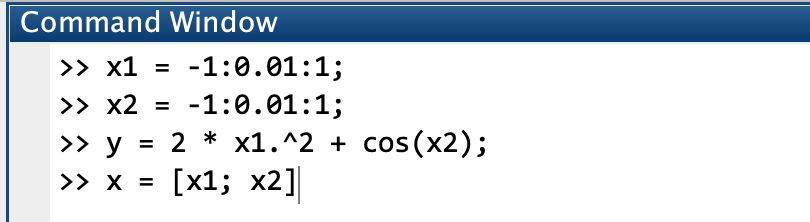


Рис. 1.1. Объявление векторов значений.

Затем была вызвана диалоговое окно для работы с нейросетями с помощью команды nntool. В нем в качестве входных значений была импортирована матрица , а целевых - (рис. 1.2).

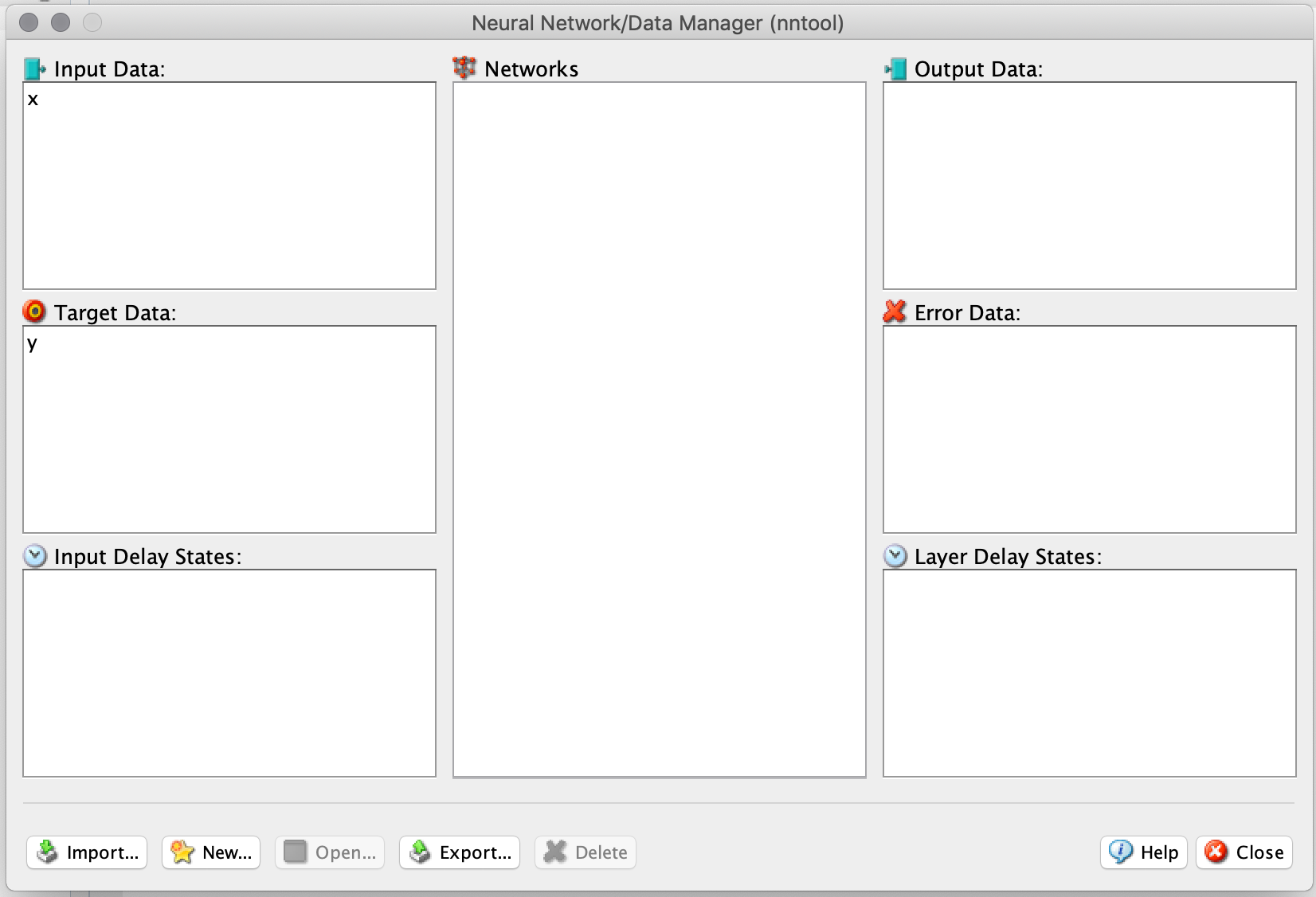


Рис. 1.2. Окно работы с нейросетями с импортированными и .

Далее мною была создана сама нейросеть. Для этого было вызвано окно создания сети. В нем в качестве типа сети был выбран тип *Feed-forward backprop*. Далее были указаны входные и целевые значения и , количество слоев установлено в 2, количество нейронов на первом слое в 2, а функции активации первого и второго слоя в линейные *PURELIN* (рис. 1.3).

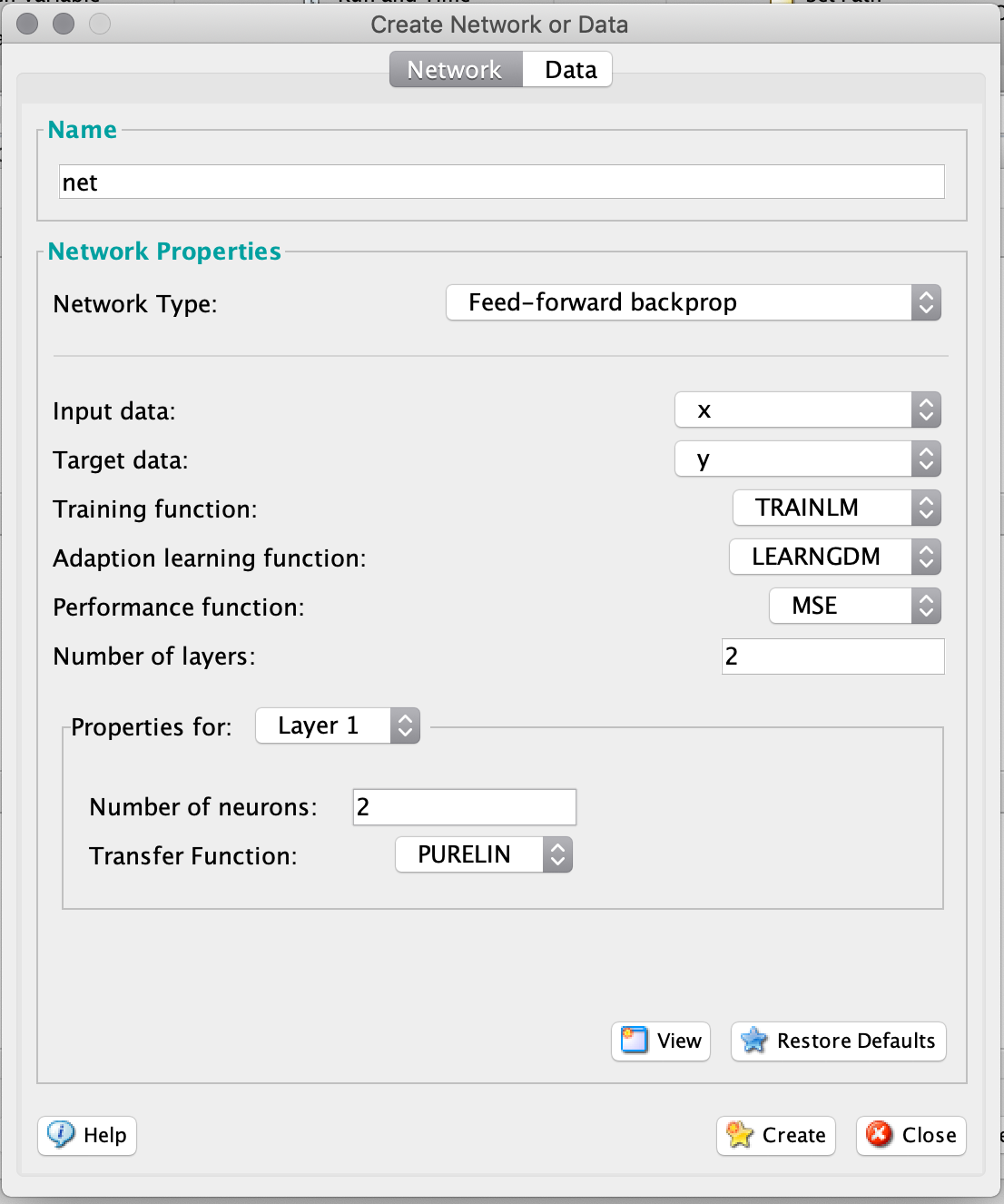


Рис. 1.3. Окно создания нейронной сети.

На вкладке *Train* были установлены значения входов и целей нейросети для обучения. Параметры обучения оставлены по умолчанию. Затем было произведено обучение созданной нейронной сети на значениях и (рис. 1.4, 1.5).

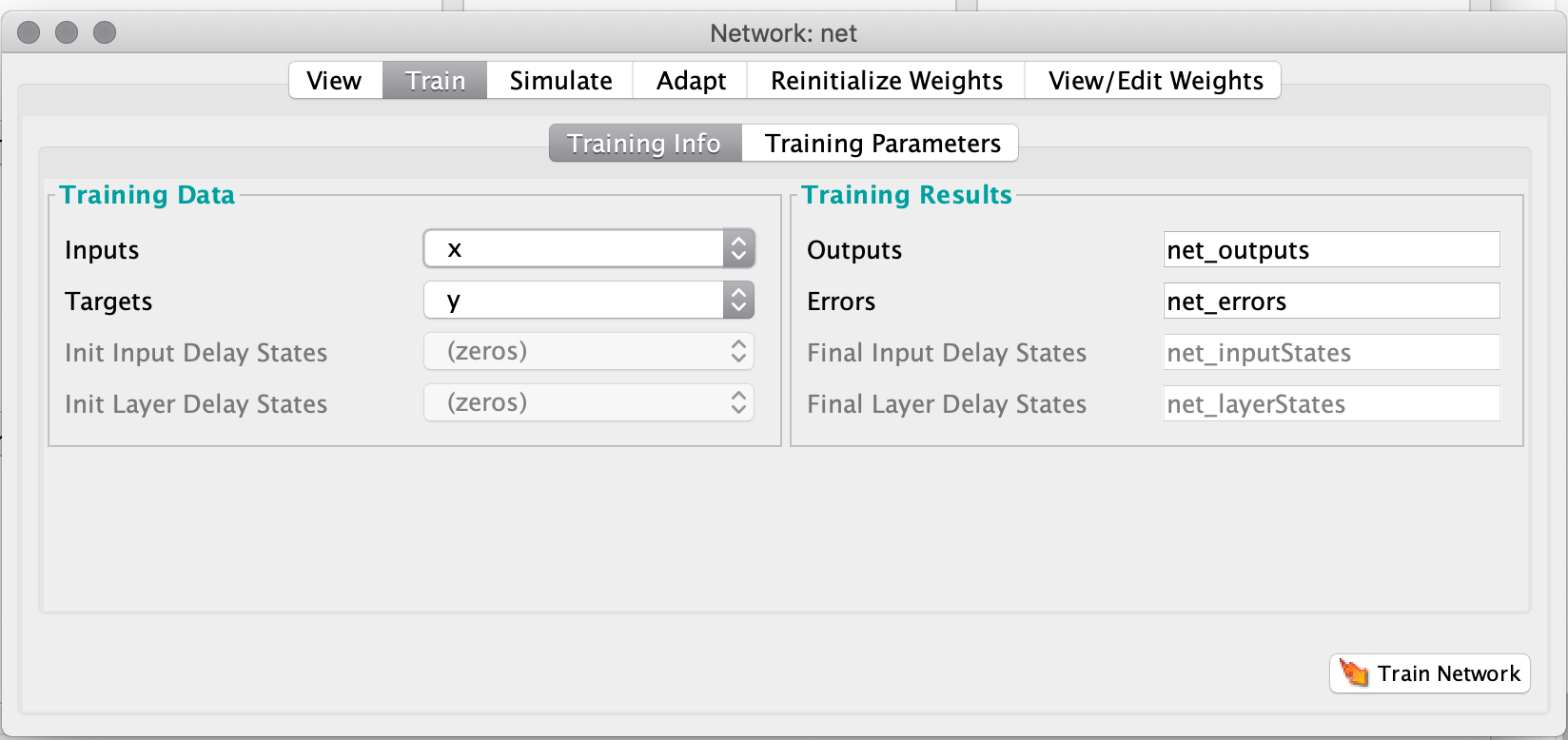


Рис. 1.4. Окно настройки параметров обучения.

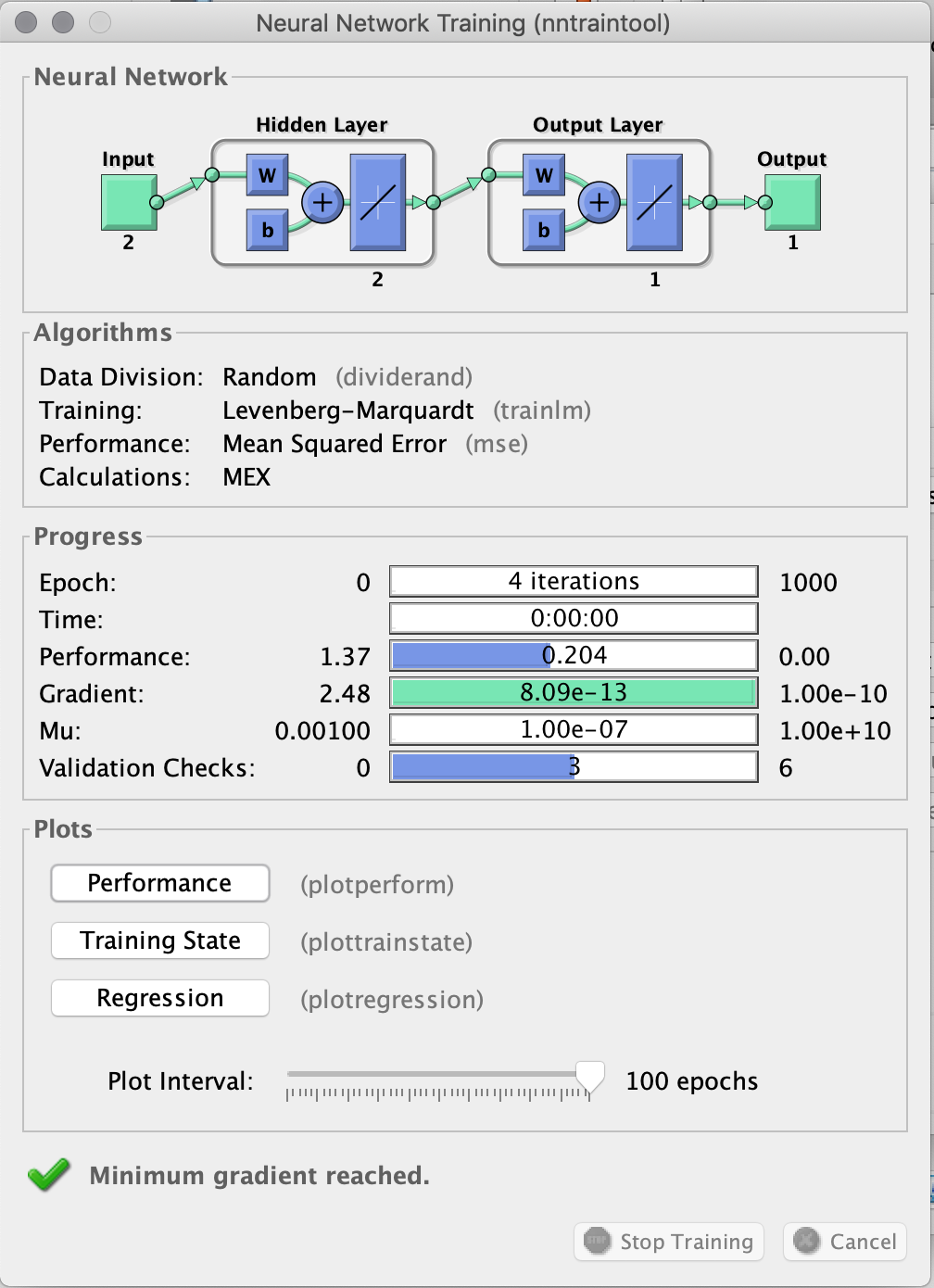


Рис. 1.5. Окно результатов обучения.

Результаты обучения приведены на рисунках 1.6 – 1.8.

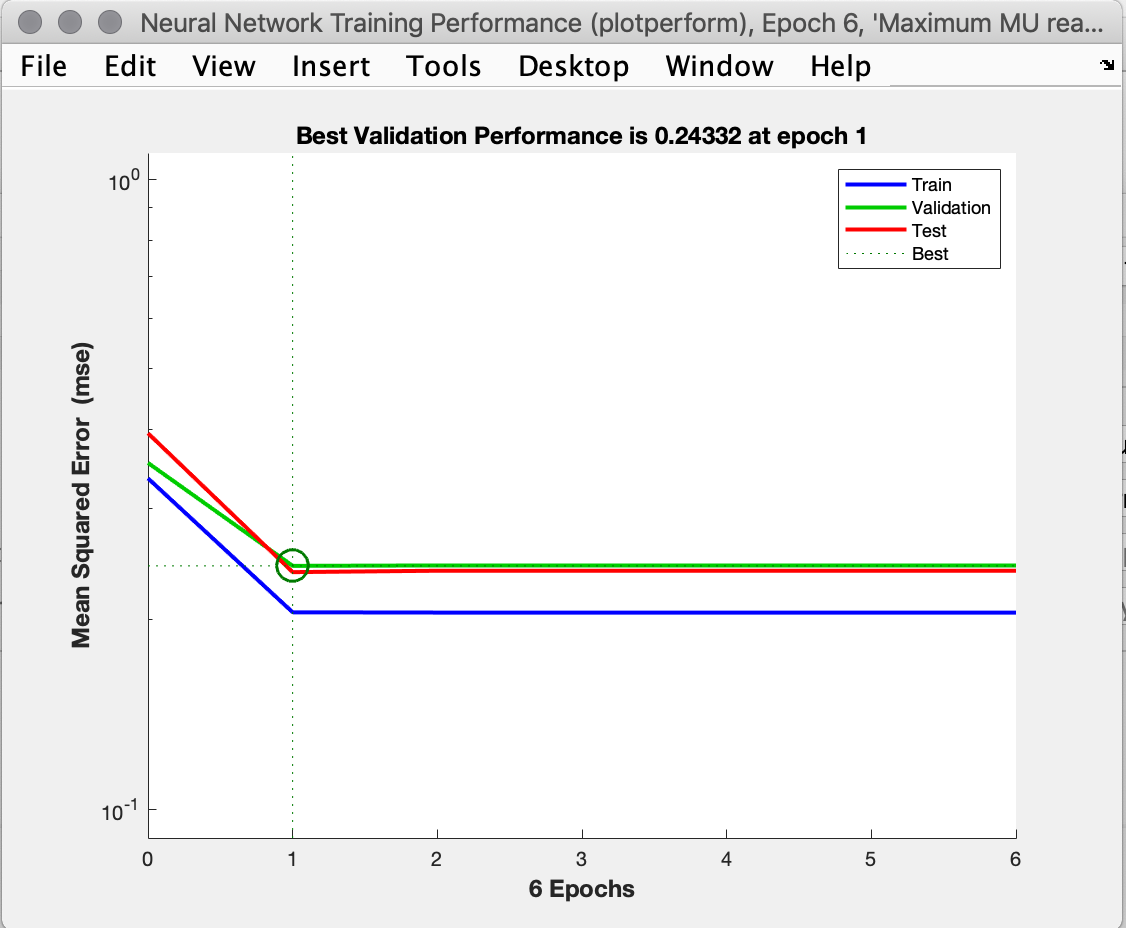


Рис. 1.6. График ошибки сети в процессе обучения.

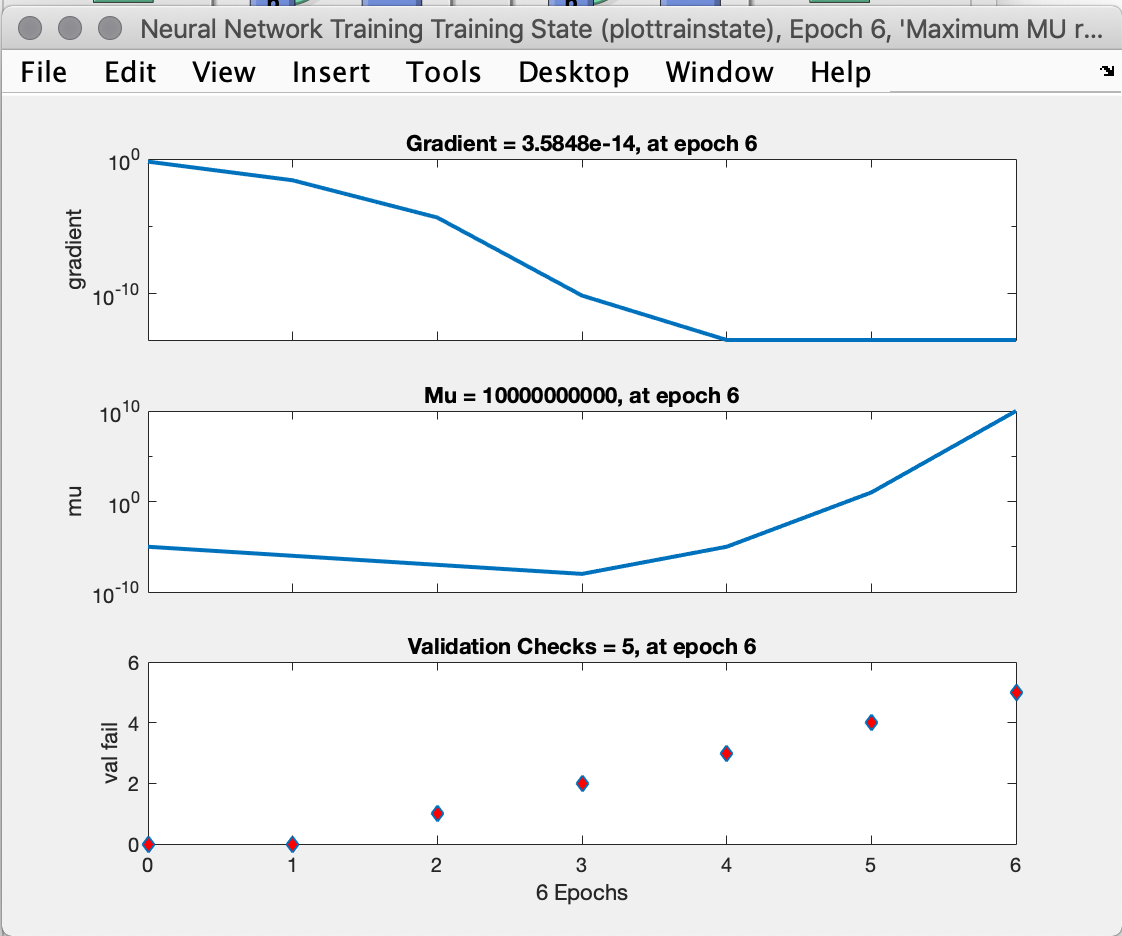


Рис. 1.7. Окно состояния основных параметров обучения.

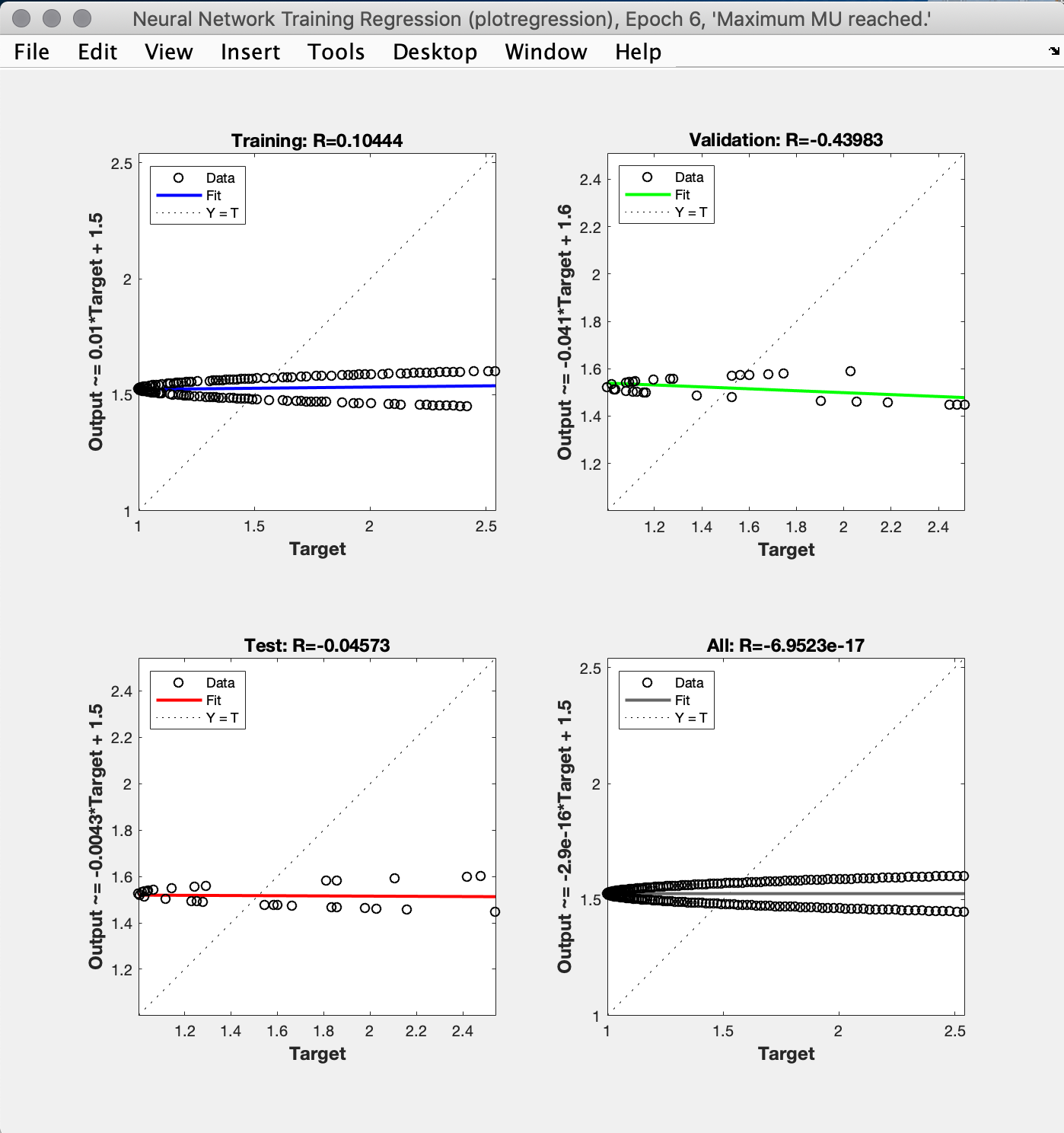


Рис. 1.8. Окно линейной регрессии.

## Вывод

В результате работы были изучены основы создания нейронных сетей с помощью Neural Netrowork Toolbox пакета Matlab. Была создана и обучена нейросеть для аппроксимации функции.

# ИЗУЧЕНИЕ СВОЙСТВ ЛИНЕЙНОГО НЕЙРОНА И ЛИНЕЙНОЙ НЕЙРОННОЙ СЕТИ

## Цель

Изучить свойства линейного нейрона. Изучить возможности нейрона выполнять логические функции двух пере- менных: И-НЕ (NAND), ИЛИ (OR), ИЛИ-НЕ (NOR), равнозначность (XOR), неравнозначность (NXOR).

## Ход работы

Сперва мною были заданы векторы входных значений x1 и x2, а также векторы результата логических операций над ними (рис. 2.1).

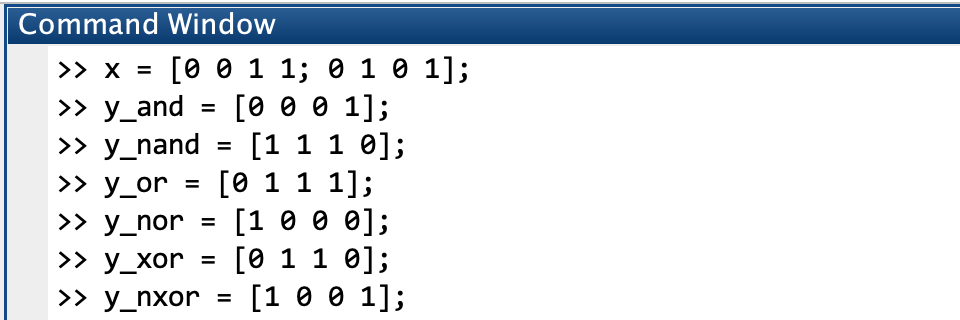


Рис. 2.1. Определение данных.

Затем командой net = newp([0 1; 0 1], 1); создается нейронная сеть с одним нейроном и диапазонов входа от 0 до 1.

Далее, с помощью команды net = train(net, x, y\_<операция>); производится обучение нейронной сети на входных данных для логической операции. С помощью команды sim(net, x) симулируется работа обученной нейронной сети. После завершения обучения над текущей операцией сеть реиницилизировалась с помощью команды net = init(net);.

Результаты обучения для операции AND приведены на рисунках 2.2 – 2.3. Нейронная сеть успешно справилась с моделированием логической операции AND.

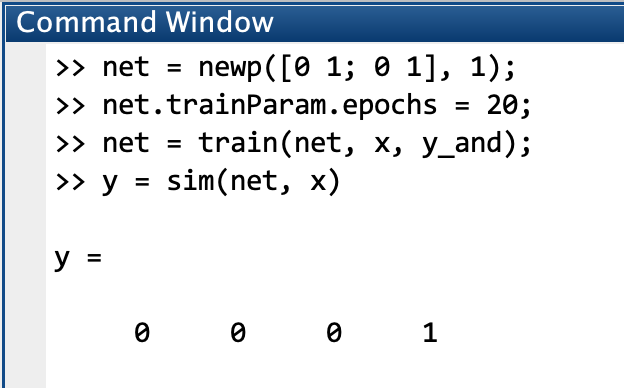


Рис. 2.2. Результат моделирования логической операции AND.

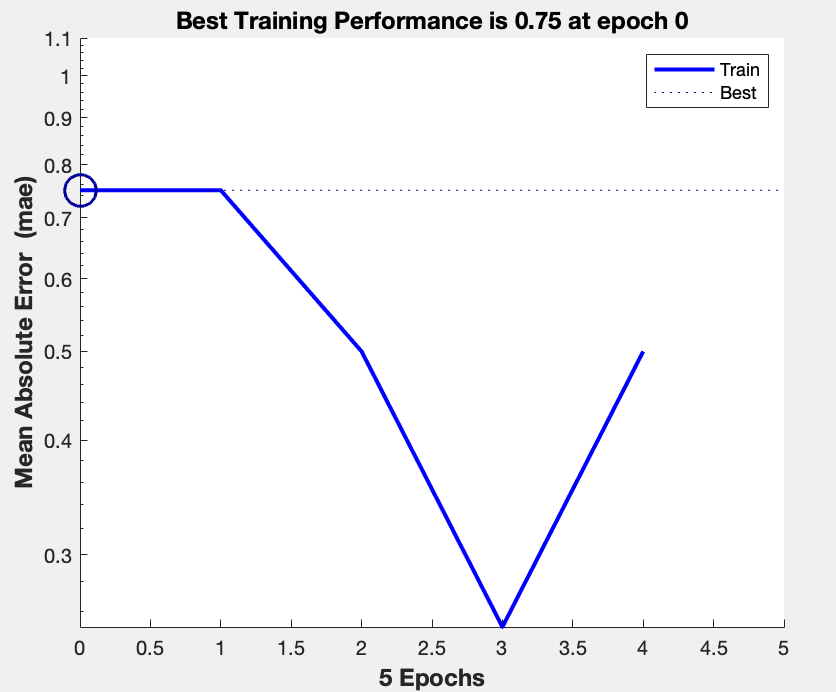


Рис. 2.3. График ошибки в процессе обучения.

Результаты обучения для операции NAND приведены на рисунках 2.4 – 2.5. Нейронная сеть успешно справилась с моделированием логической операции NAND.

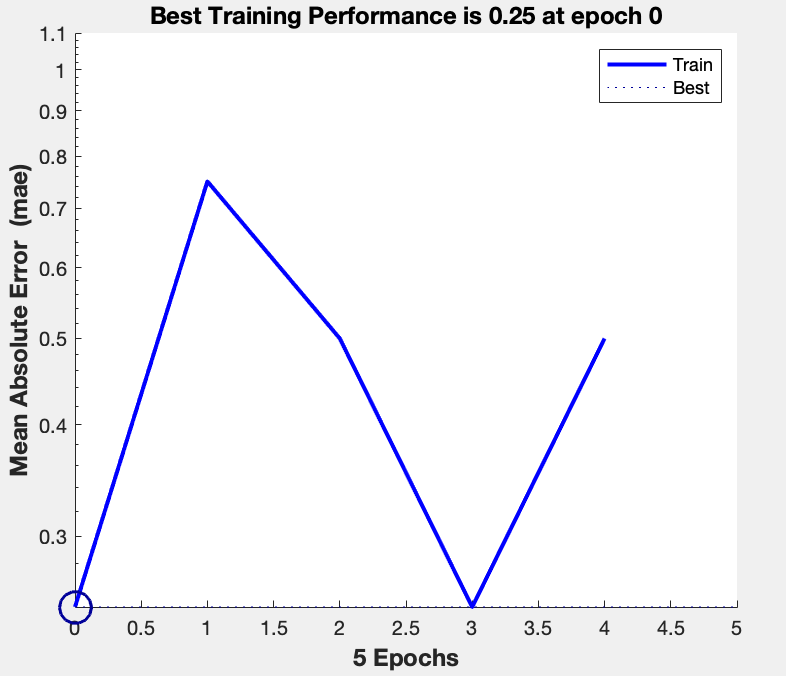


Рис. 2.4. График ошибки в процессе обучения.

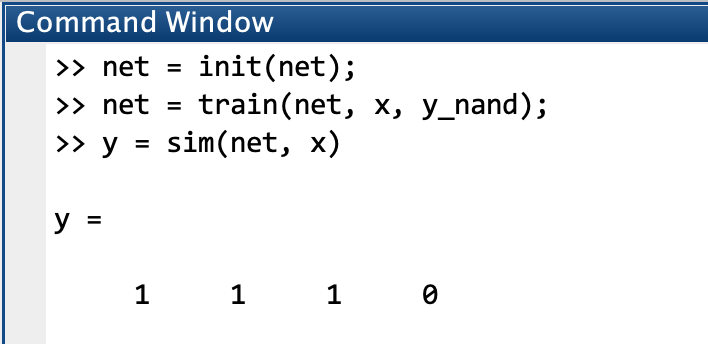


Рис. 2.5. Результат моделирования логической операции NAND.

Результаты обучения для операции OR приведены на рисунках 2.6 – 2.7. Нейронная сеть успешно справилась с моделированием логической операции OR.

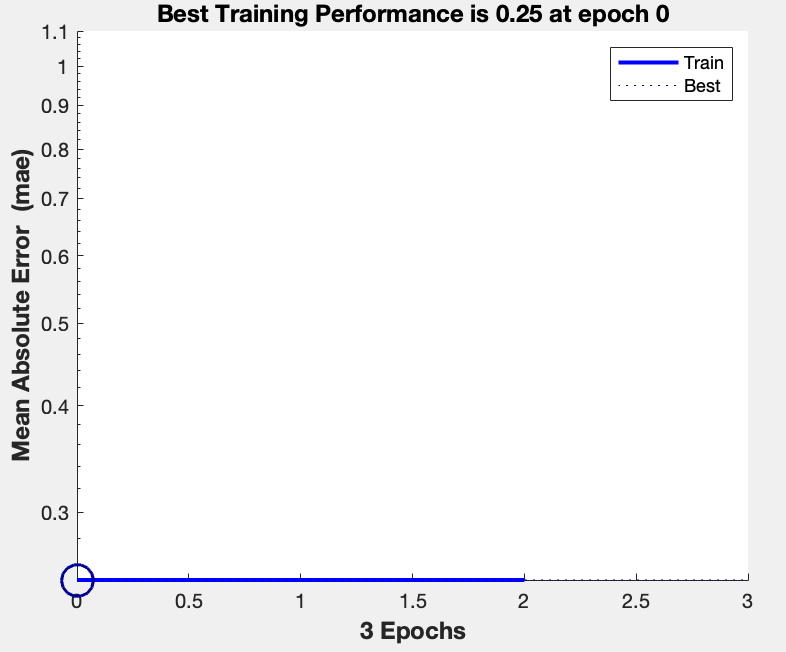


Рис. 2.6. График ошибки в процессе обучения.

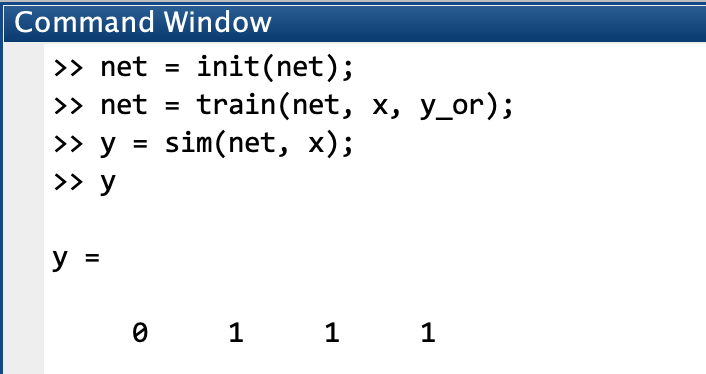


Рис. 2.7. Результат моделирования логической операции OR.

Результаты обучения для операции NOR приведены на рисунках 2.8 – 2.9. Нейронная сеть успешно справилась с моделированием логической операции OR.

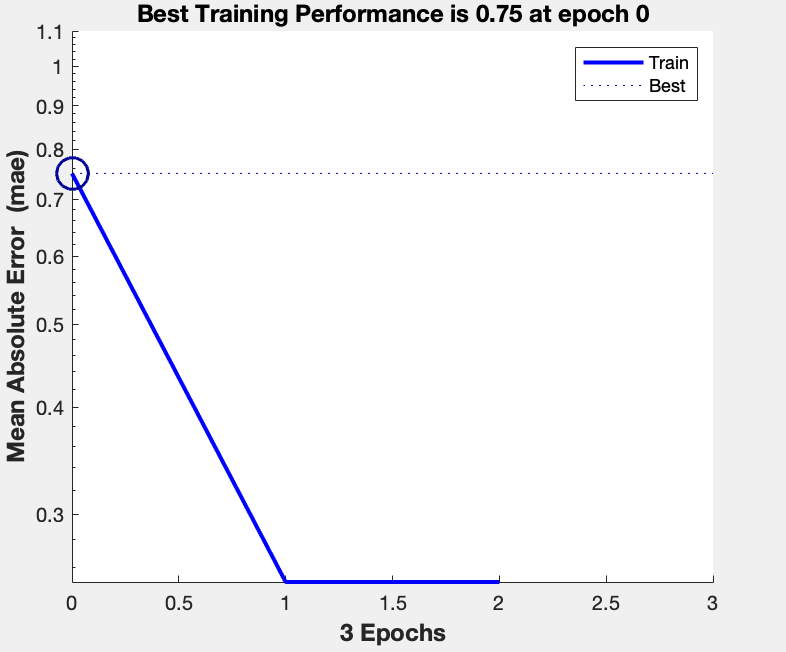


Рис. 2.8. График ошибки в процессе обучения.

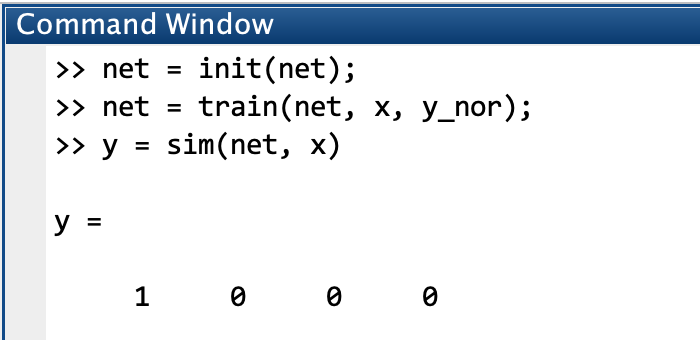


Рис. 2.9. Результат моделирования логической операции NOR.

Результаты обучения для операции XOR приведены на рисунках 2.10 – 2.11.

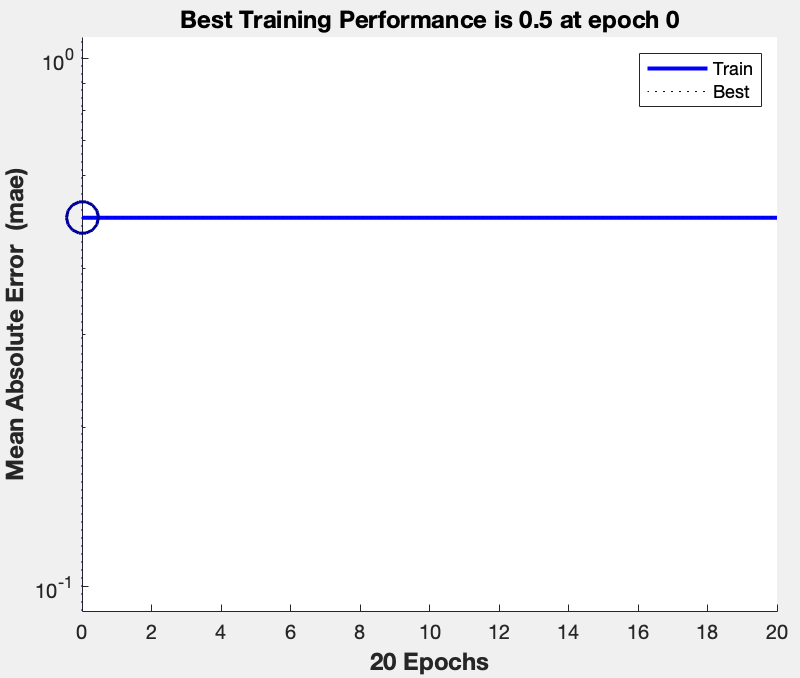


Рис. 2.10. График ошибки в процессе обучения.

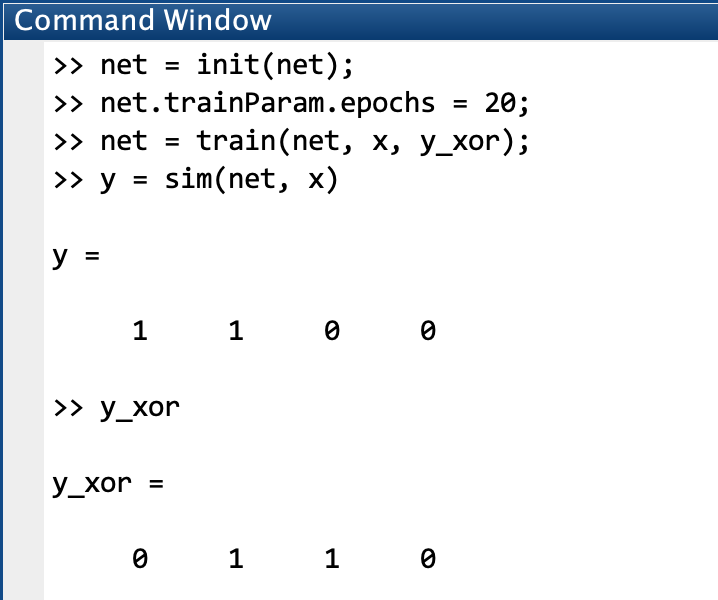


Рис. 2.11. Результат моделирования логической операции XOR.

Ошибка при обучении не уменьшалась и держалась на уровне 0,5. При симуляции работы нейрона не были получен корректный вектор значений операции XOR. Следовательно, логическая операция XOR не является линейной и не может быть успешно смоделирована с помощью линейного нейрона. Данный факт отображен на рисунке 2.12, где видно, что область значений операции XOR линейно несеперабельна.

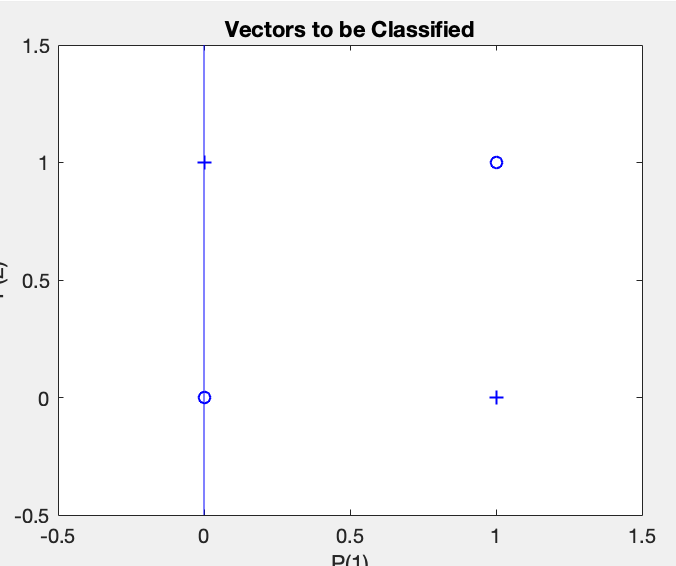


Рис. 2.12. График результатов операции XOR и линии, разделяющей результаты.

Результаты обучения для операции NXOR приведены на рисунках 2.13 – 2.14. Аналогично операции XOR, не является линейной и не может быть успешно смоделирована с помощью линейного нейрона. Данный факт отображен на рисунке 2.15, где видно, что область значений операции XOR линейно несеперабельна.

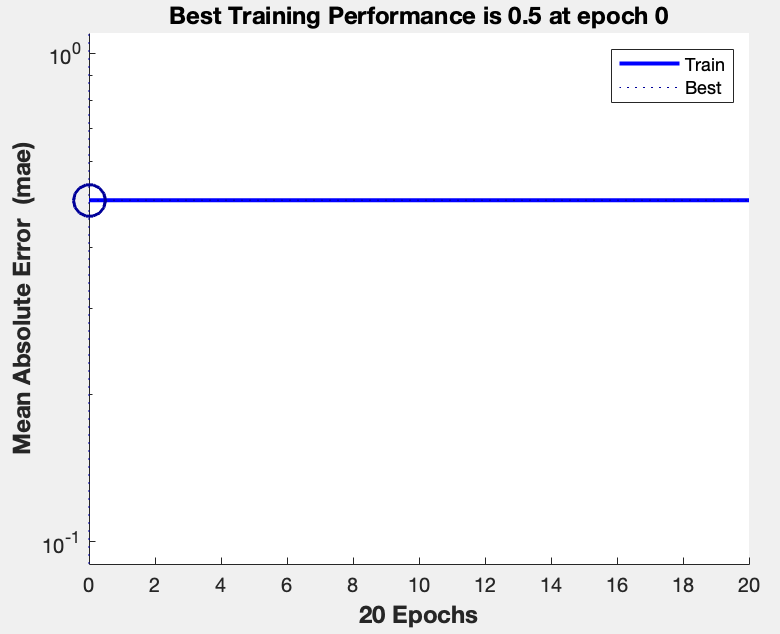


Рис. 2.13. График ошибки в процессе обучения.

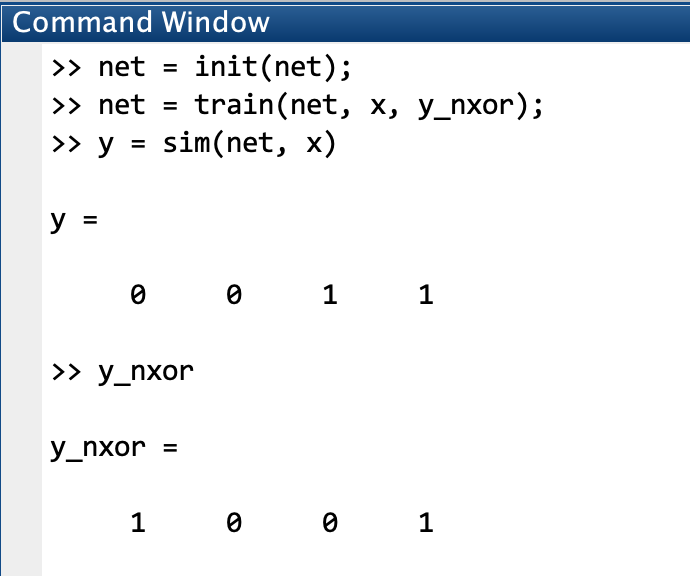


Рис. 2.11. Результат моделирования логической операции XOR.

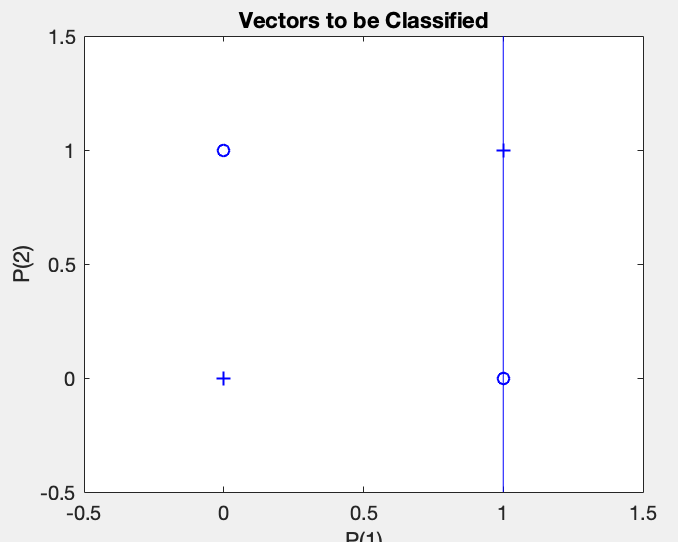


Рис. 2.12. График результатов операции NXOR и линии, разделяющей результаты.

## Вывод

В результате работы были изучены возможности линейного нейрона при моделировании логических операций. Оказалось, что логические операции AND, NAND, OR, NOR успешно моделируются линейным нейроном, т.к. области значений этих операций можно разделить линейно. Операции XOR и NXOR не моделируются с помощью линейного нейрона, т.к. являются нелинейными и область их значений не может быть разделена прямой.

# ИЗУЧЕНИЕ МНОГОСЛОЙНОГО НЕЛИНЕЙНОГО ПЕРСЕПТРОНА И АЛГОРИТМА ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

## Цель

Изучить возможности многослойного персептрона как универсального аппроксиматора и классификатора. Реализовать алгоритм обратного распространения ошибки.

## Ход работы

Изначально мною был реализован алгоритм обратного распространения ошибки для одного входа. Он был оформлен в виде функции, которая принимает значение сигнала, выхода, скорости обучения, а также векторы весов и пороговых значений для каждого слоя. Код функции приведен ниже, комментариями обозначены ключевые шаги:

function [out, w1, b1, w2, b2] = ...

backpropagate(p, t, w1, b1, w2, b2, lr)

% вычисление пр€мого распространени€ сигнала

n1 = w1 \* p + b1;

a1 = logsig(n1);

n2 = dot(w2, a1) + b2;

a2 = purelin(n2);

% вычисление обратного распространени€ сигнала

e = t - a2;

s2 = -2 \* dpurelin(n2, a2) \* e;

s1 = dlogsig(n1, a1) .\* w2 .\* s2;

% вычисление новых весов и задержек

w1 = w1 - lr \* s1 \* p;

b1 = b1 - lr \* s1;

w2 = w2 - lr \* s2 \* a1;

b2 = b2 - lr \* s2;

out = a2;

end

Затем, для отладки, данная функция была применена для значений из примера Backpropogation Calculation. Код:

% инициализаци€ исходных данных

p = 1;

t = 1 + sin(p \* pi / 4);

w1 = [-0.270 -0.410];

b1 = [-0.480 -0.130];

w2 = [0.090 -0.170];

b2 = 0.48;

lr = 0.1;

% вызов функции обратного распространени€ ошибки

[out1, w1, b1, w2, b2] = ...

backpropagate(p, t, w1, b1, w2, b2, lr);

% вывод результата

fprintf('out = '); disp(out1);

fprintf('w1 = '); disp(w1);

fprintf('b1 = '); disp(b1);

fprintf('w2 = '); disp(b1);

fprintf('b2 = '); disp(b1);

Результат совпал со значениями из примера:

out = 0.4463

w1 = -0.2651 -0.4200

b1 = -0.4751 -0.1400

w2 = -0.4751 -0.1400

b2 = -0.4751 -0.1400

Для дальнейшей проверки работоспособности реализованной функции мною были сгенерирован вектор значений сигналов и значений функции. Затем было произведено обучение нейросети с помошью реализованного алогиритма:

% инициализация исходных данные

p = -10:0.1:10;

t = 1 + sin(p \* pi / 4);

out = zeros(length(p));

% цикл по количеству эпох

for epoch = 1:100

% цикл по вектору сигналов и значений функции

for i = 1:length(p)

% вызов функции обратного распространения ошибки

[out(i), w1, b1, w2, b2] = ...

backpropagate(p(i), t(i), w1, b1, w2, b2, lr);

end

end

% построение графиков функции и ее аппроксимации

plot(p, t, p, out)

В результате получилась нейросеть, способная аппроксимировать исходную функцию. Результат аппроксимации приведен на рисунке 3.1.

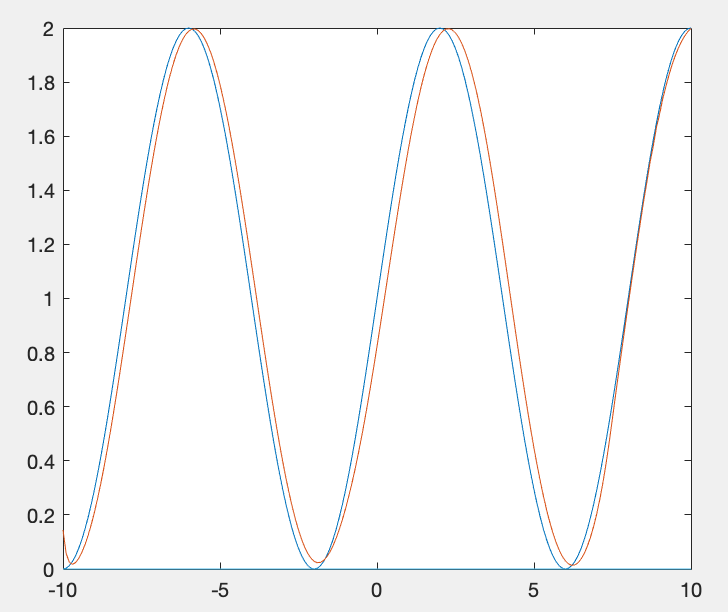


Рис. 3.1. Графики исходной функции и ее аппроксимации.

## Вывод

В результате работы были изучены возможности многослойного перцептрона, был изучен и реализован алгоритм распространения ошибки.

# ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ АППРОКСИМАЦИИ ФУНКЦИЙ И ПРЕДСКАЗАНИЯ ВРЕМЕННОГО ПРОЦЕССА

## Цель

Выработать практические навыки при исследовании возможностей применения нейронных сетей для решения прикладных задач при помощи пакета прикладных программ MatLab.

Построить модель нейронной сечи, аппроксимирующей полином на интервале [-1, 1]:

.

Построить модель нейронной сети для аппроксимации функции на промежутке [0,10]:

## Ход работы

Для решения задачи аппроксимации полинома была создана Feed-Forward Back Propagation нейросеть с 10 нейронами на первом слое и функцией активации tansig. Второй слой состоит из одного нейрона с линейной активацией:

% инициализация входных данных

x1 = -1:0.05:1;

x2 = -1:0.05:1;

x = [x1; x2];

y = x2 .^ 2 + x1 - 2;

% создание и обучение нейронной сети

net = newff([-1 1; -1 1], [10 1], {'tansig', 'purelin'}, 'trainlm');

net.divideFcn = 'dividerand';

net = train(net, x, y);

% построение поверхности Y

y = meshgrid(y);

surf(x1, x2, y);

% соответсвующа€ поверхность, полученна€ с помошью нейросети

out = sim(net, x);

out = meshgrid(out);

surf(x1, x2, out);

Полученные результаты при обучении отображены на рисунках 4.1 – 4.3. Графики исходной функции и ее аппроксимации приведены на рисунках 4.4 – 4.5.

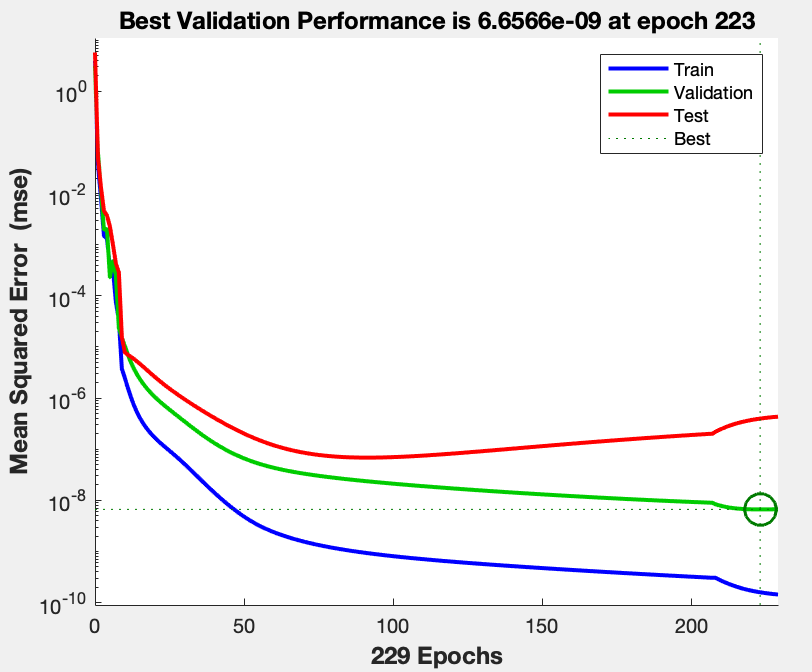


Рис. 4.1. Изменение ошибки в процессе обучения.

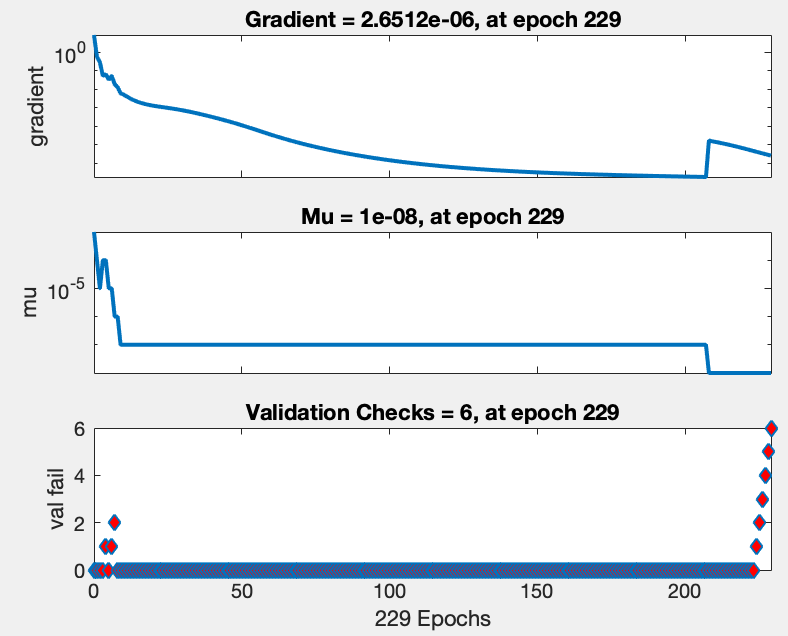


Рис. 4.2. Окно состояния обучения.

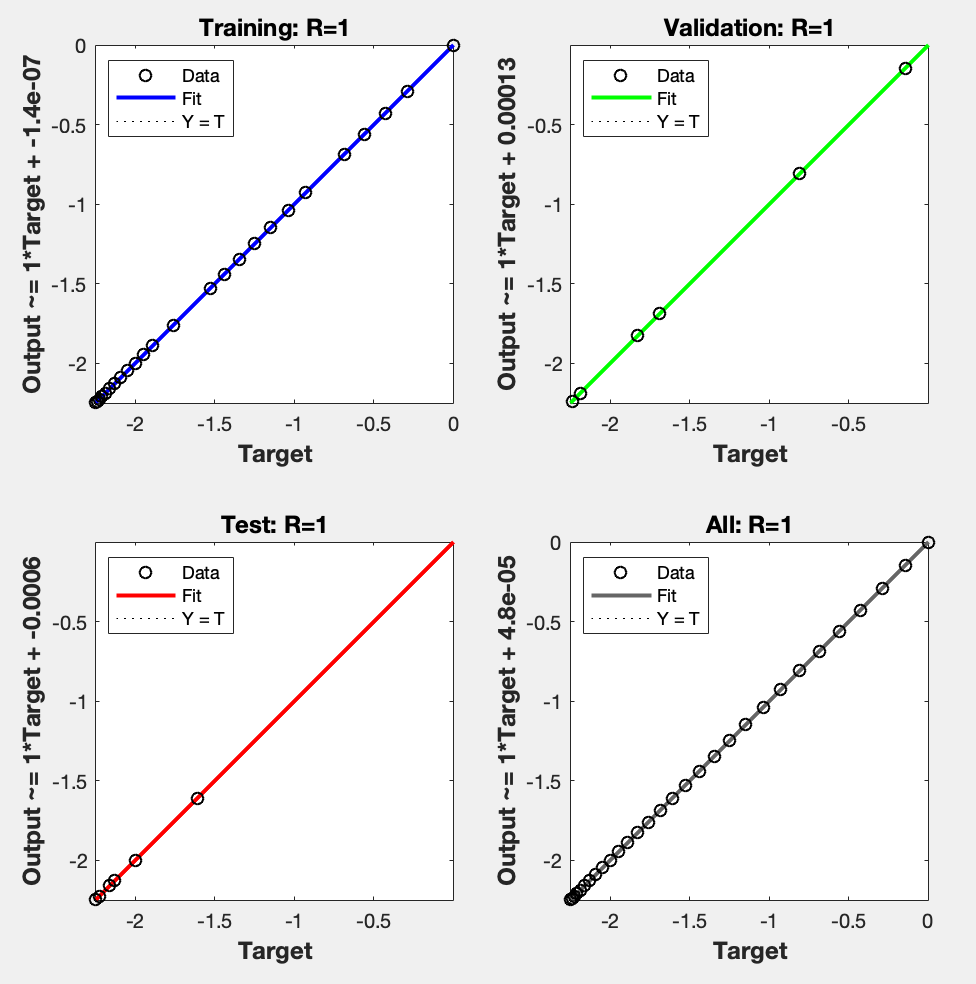


Рис. 4.3. Окно линейной регрессии между входами и выходами.

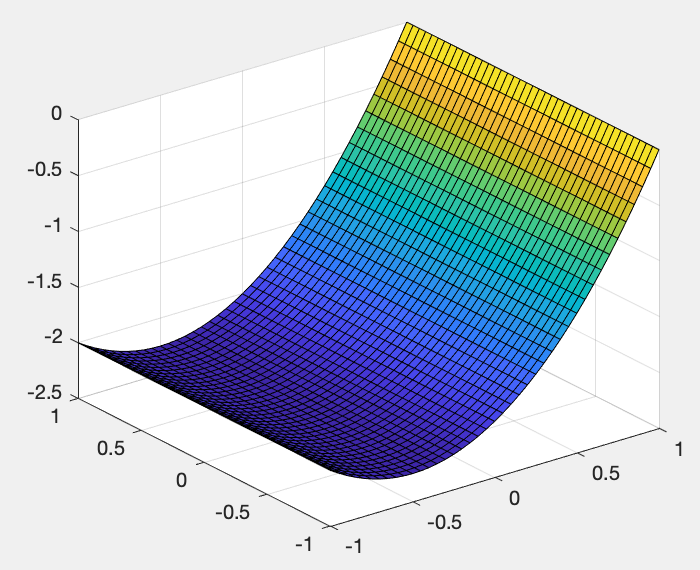


Рис. 3.4. Поверхность исходной функции.

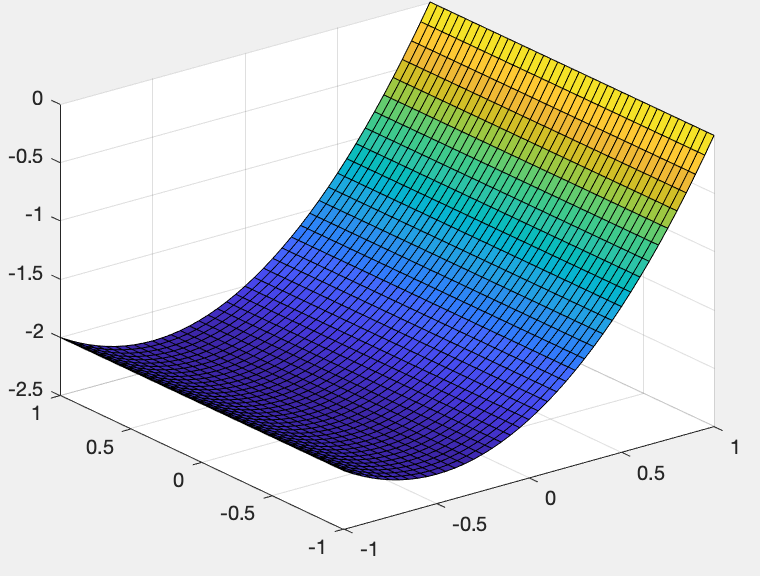


Рис. 3.5. Поверхность аппроксимации функции с помощью НС.

Для решения задачи аппроксимации функции была создана Feed-Forward Back Propagation нейросеть с 20 нейронами на первом слое и функцией активации tansig. Второй слой состоит из одного нейрона с линейной активацией:

% инициализаци€ входных данных

t = 0:0.01:10;

y = sin(t .^ 2 - 12 \* t);

% создание и обучение нейронной сети

net = newff([0 10], [20 1], {'tansig', 'purelin'}, 'trainlm');

net.divideFcn = 'dividerand';

net = train(net, t, y);

% графики функции и ее аппроксимации

out = sim(net, t);

plot(t, y, t, out);

Полученные результаты при обучении отображены на рисунках 4.6 – 4.8. Графики исходной функции и ее аппроксимации приведены на рисунке 4.9.

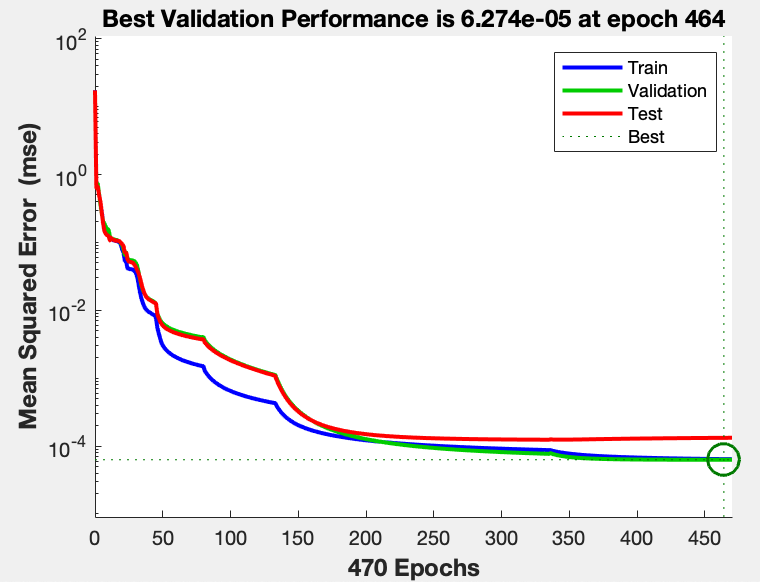


Рис. 4.6. Изменение ошибки в процессе обучения.

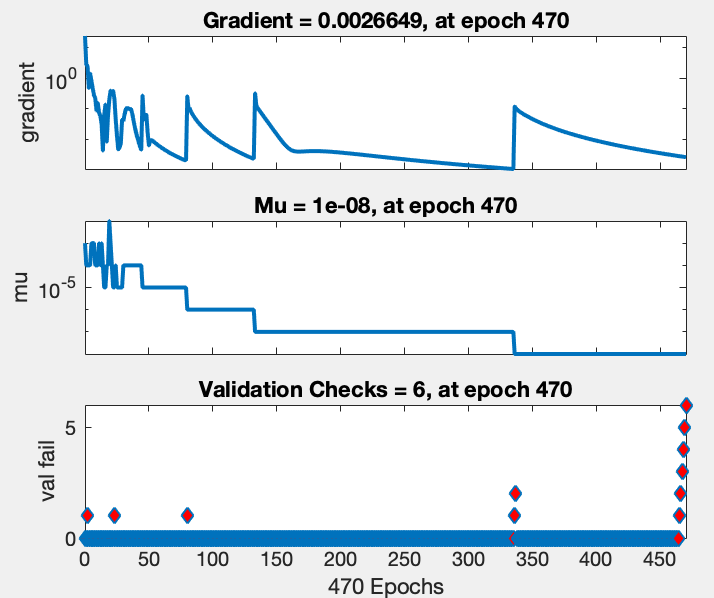


Рис. 4.7. Окно состояния обучения.

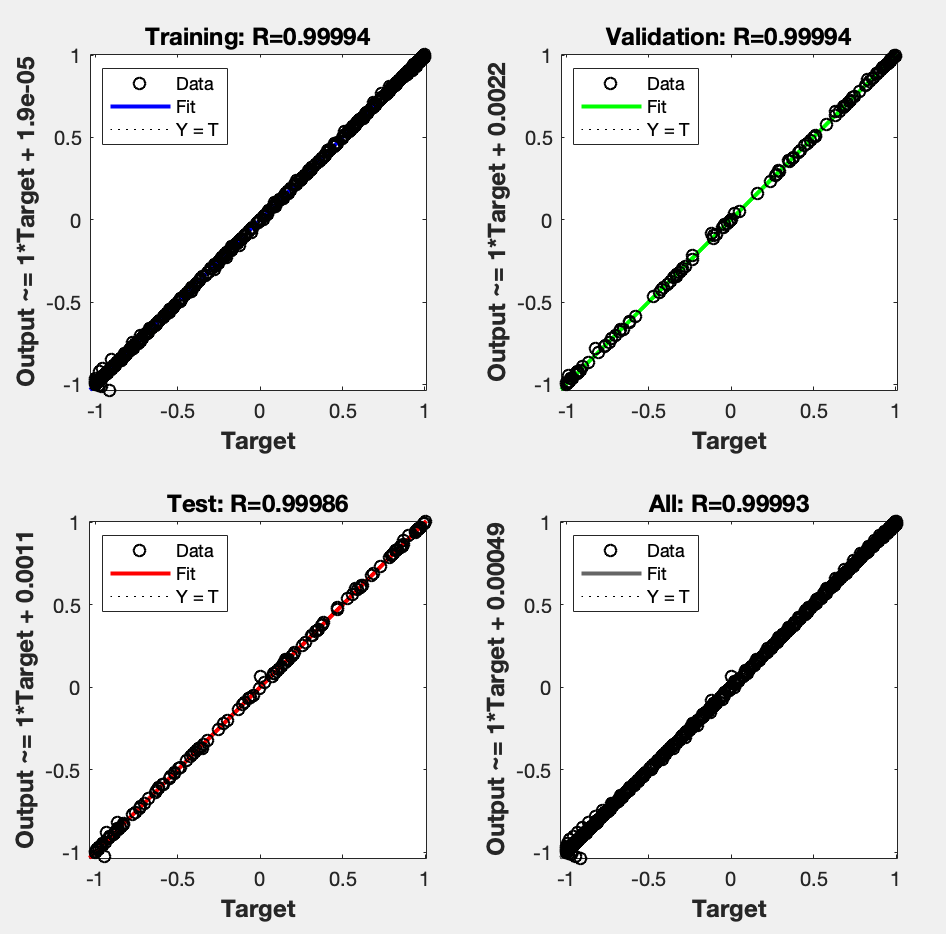


Рис. 4.8. Окно линейной регрессии.

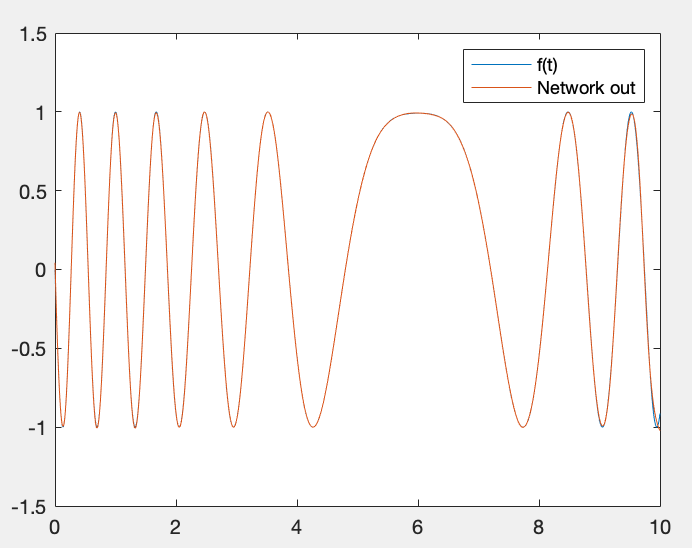


Рис 4.9. Графики исходной функции и ее аппроксимации.

## Вывод

В результате работы были построены нейросети для аппроксимации полинома и функции.