Часть первая
Цель работы

Изучение основных свойств и основ работы с GUI – интерфейсом пакета Neural Networks Toolbox в программной среде MatLab.

Содержание

Постановка задачи

Построить и натренировать нейронную сеть для вычисления функции

$$y = x_1 * sin(x_2),\ x_1,\ x_2 \in [-1; 1]$$

Исходные данные

В качестве исходных данных с помощью написанного скрипта были сгенерированы значения для x_1 и x_2, а затем по ним вычислены соответствующие значения y:

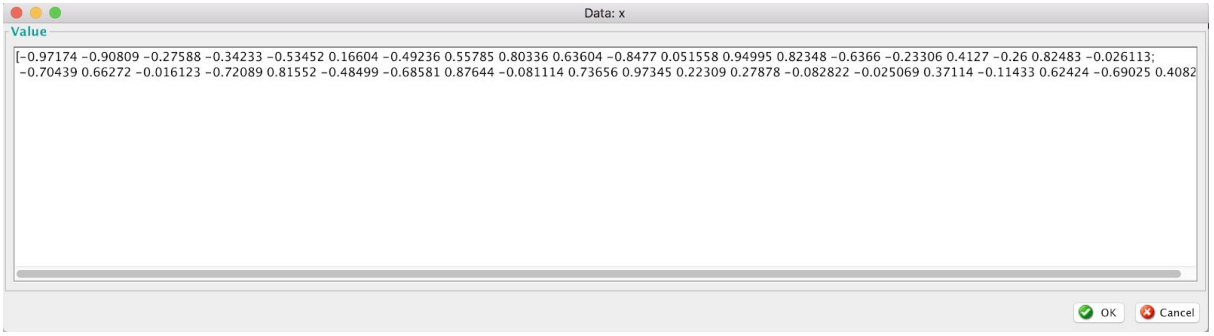
x =[-0.971744075369 -0.908085502026 -0.275879929245 -0.34233462519 -0.534516732717 0.166042363216 -0.492364276638 0.557852348271 0.803364729692 0.63603628916 -0.847704356355 0.0515582795791 0.949948818347 0.8234815387 -0.636596416475 -0.233058037265 0.412697908595 -0.259996319331 0.82482797877 -0.026113028086 ; -0.704394141359 0.662720300264 -0.0161233750095 -0.720889931336 0.815524377274 -0.48498938254 -0.685808227007 0.87643916787 -0.081113665884 0.736559535742 0.973447793537 0.223086935397 0.278783480788 -0.0828220159893 -0.0250693227568 0.371137173967 -0.114327560673 0.624237227564 -0.690246562882 0.408250344609]

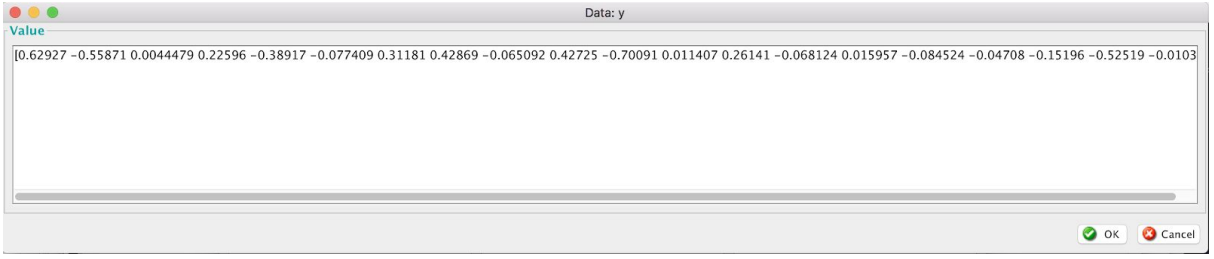
y =[0.629274528104 -0.55871195223 0.00444792283491 0.225959155974 -0.389173696763 -0.0774087781133 0.311813661833 0.428690121478 -0.0650924249649 0.427253084799 -0.700907259645 0.0114068110504 0.261412902368 -0.0681244556635 0.0159573694507 -0.0845244206891 -0.0470800263705 -0.151962264571 -0.52519051128 -0.0103669785159]

Алгоритм решения

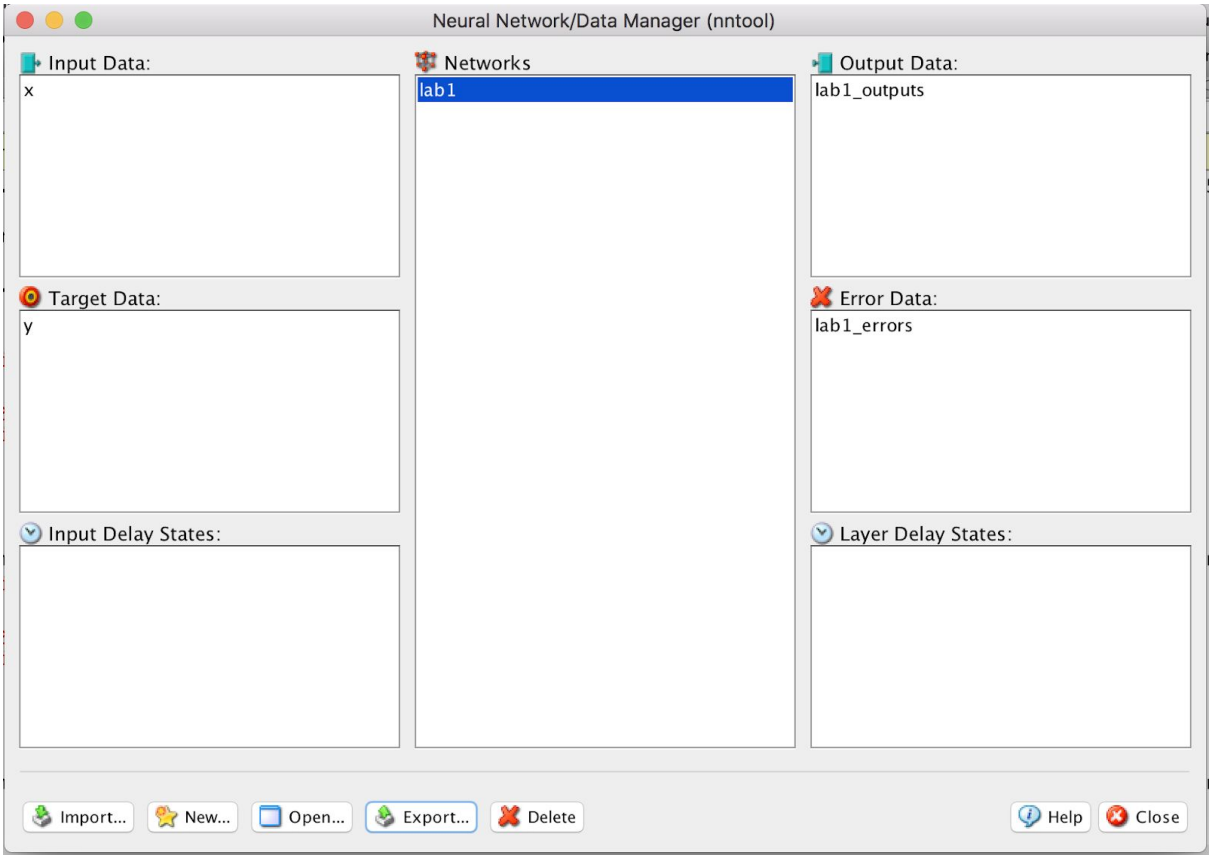
Построение сети

1. С помощью функции **nntool** откроем основное окно интерфейса и сформируем последовательность входов и целей в рабочей области.
2. Внесем исходные данные (*Input Data* и *Target Data)* в окно *Neural Network/Data Manager*.

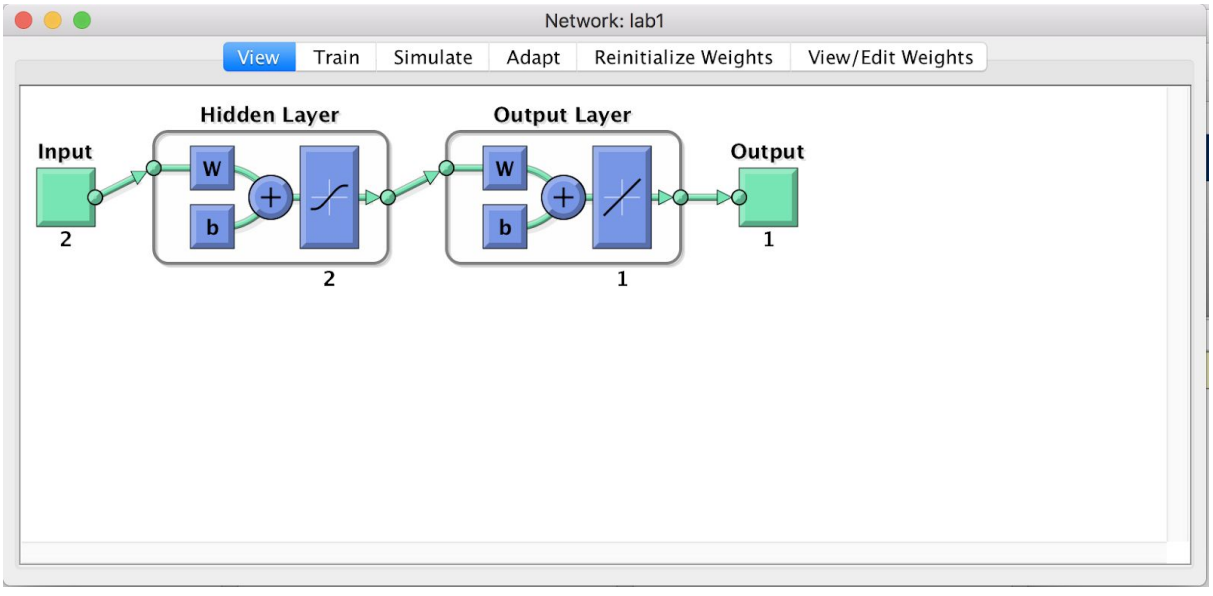




3. Добавим сеть во вкладке *Networks*

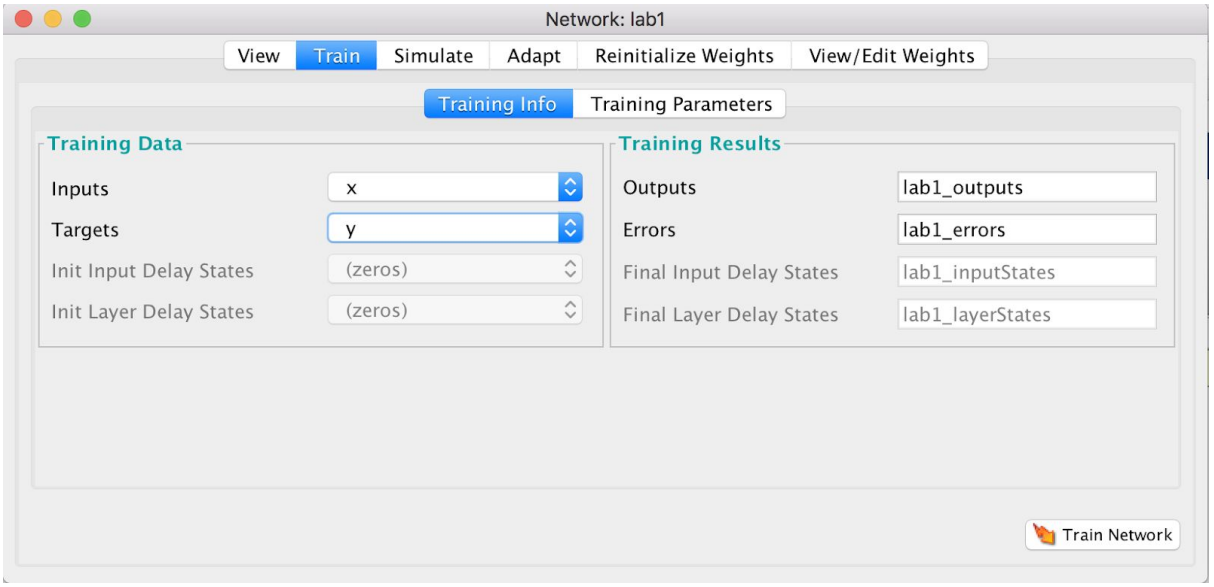


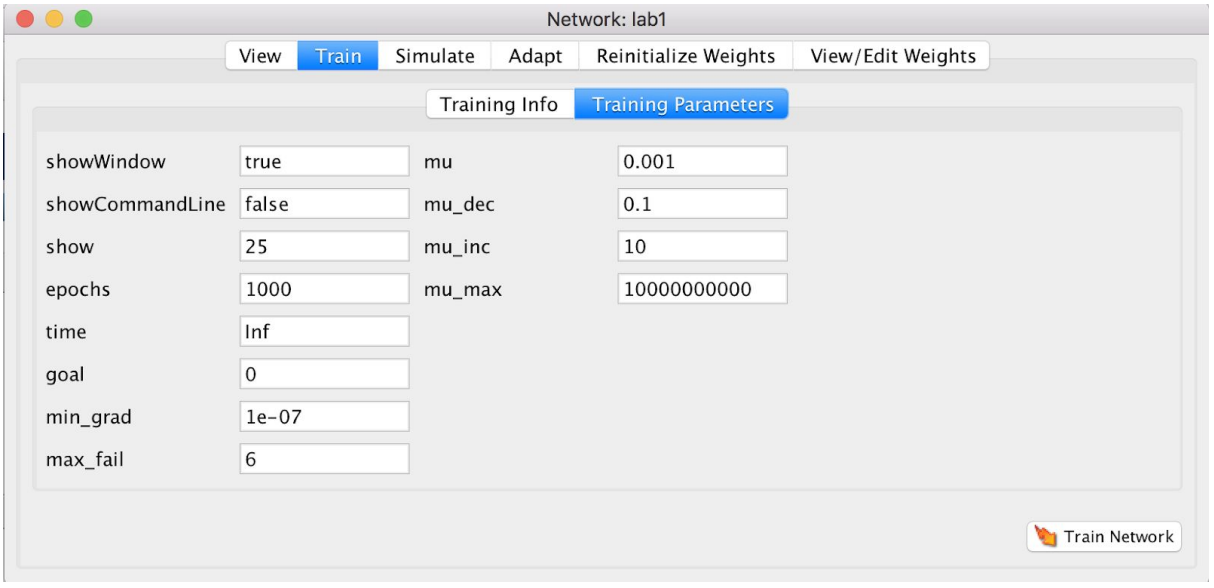
4. Зайдем в настройки сети. Выберем ее тип *Feed-forward backprop* с прямой передачей сигнала и обратным распространением ошибки.
5. Добавим сети *Input Data*, *Target Data*, количество нейронов первого слоя равно 2, функция активации во втором слое - линейная (*PURELIN*).



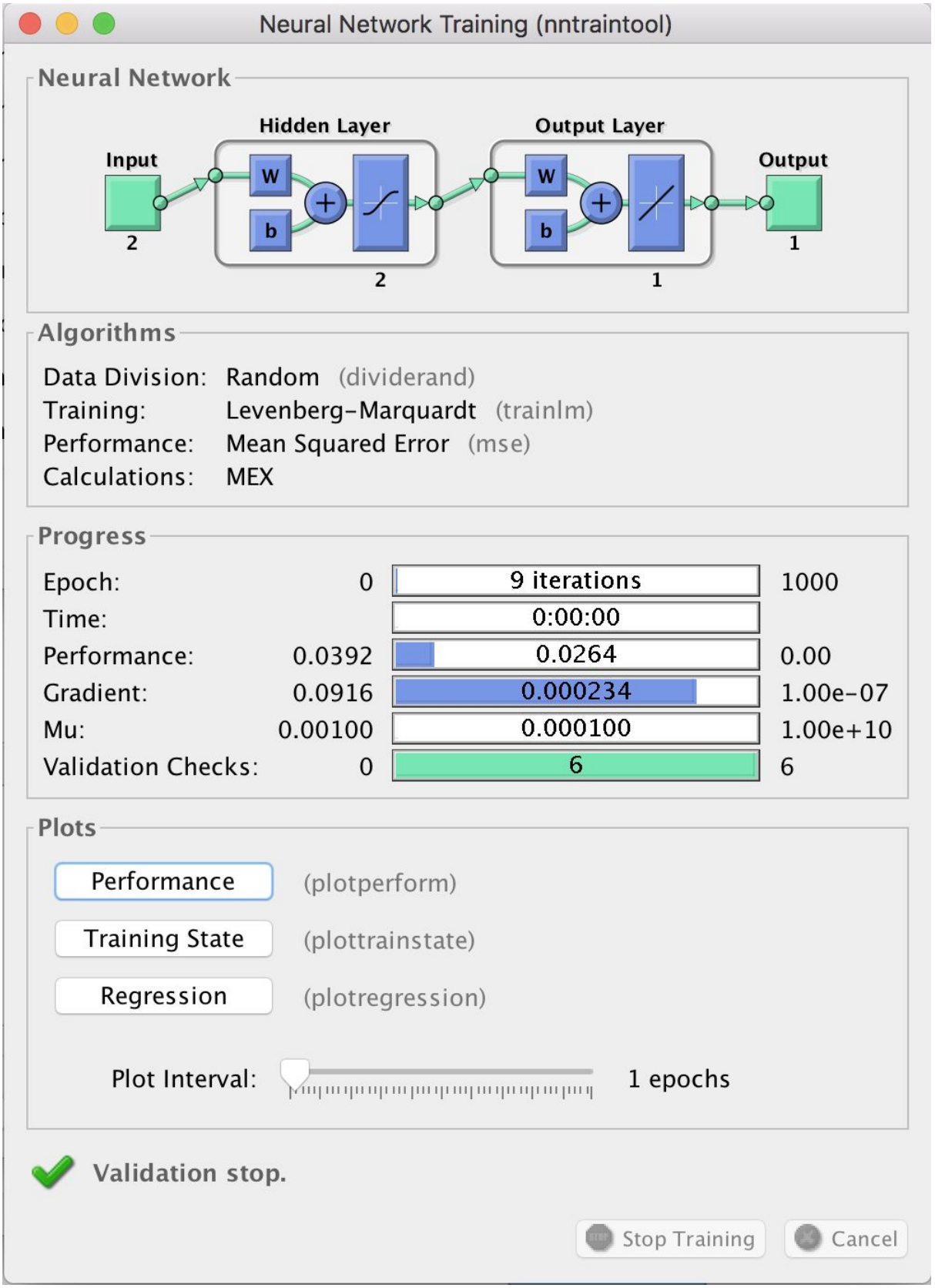
Обучение сети

Установим значения параметров во вкладке Train, как сказано в условии:





И нажмем *Train Network*. Качество обучения сети на выбранной обучающей последовательности показан на следующих графиках:


Neural Network Training (nntraintool)
Neural Network
Hidden Layer
Output Layer
Input
Output
W
b
2
1
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX
Progress
Epoch: 0 9 iterations 1000
Time: 0:00:00
Performance: 0.0392 0.0264 0.00
Gradient: 0.0916 0.000234 1.00e-07
Mu: 0.00100 0.000100 1.00e+10
Validation Checks: 0 6 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)
Plot Interval: 1 epochs
Validation stop.
Stop Training
Cancel

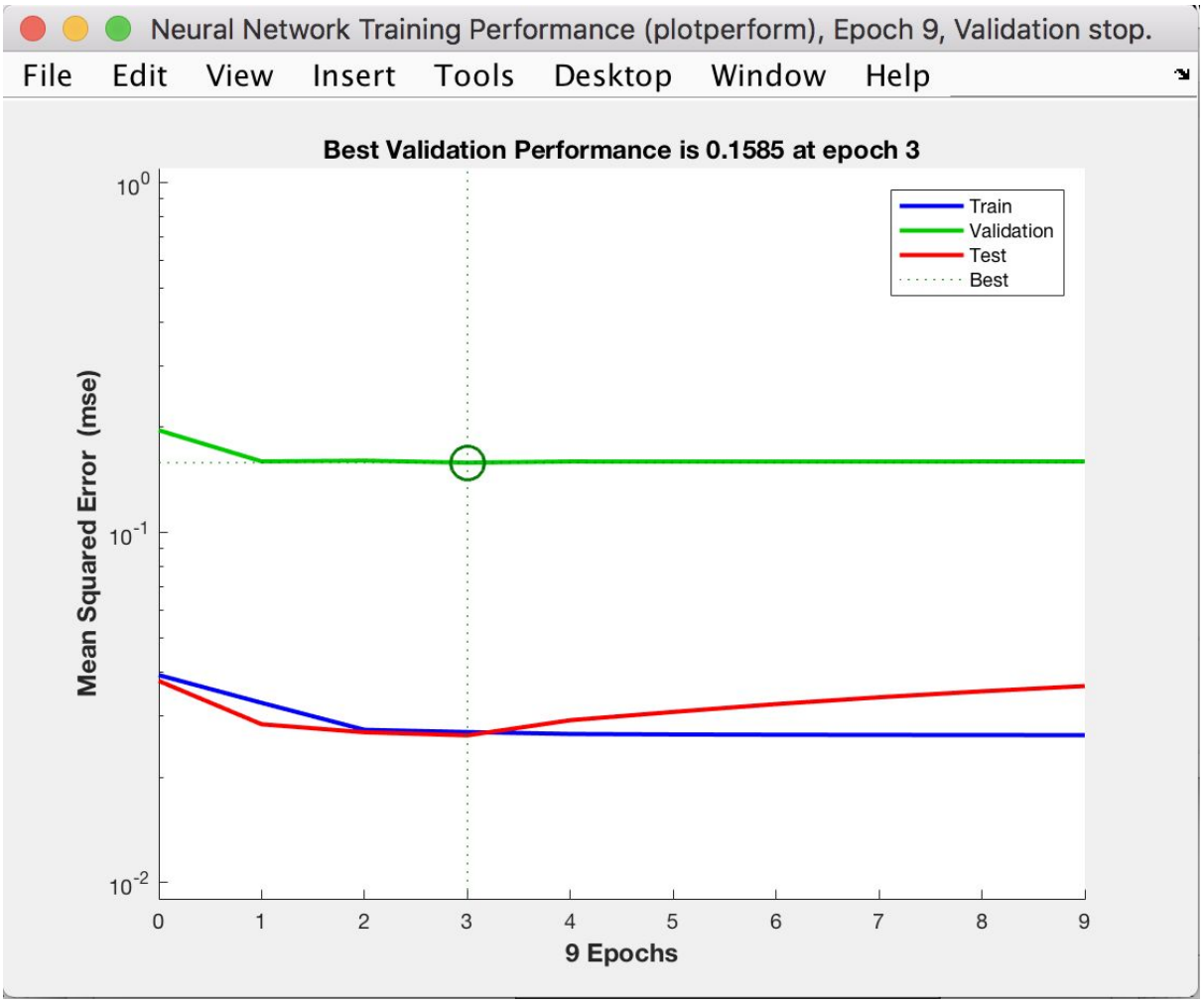


График Performance

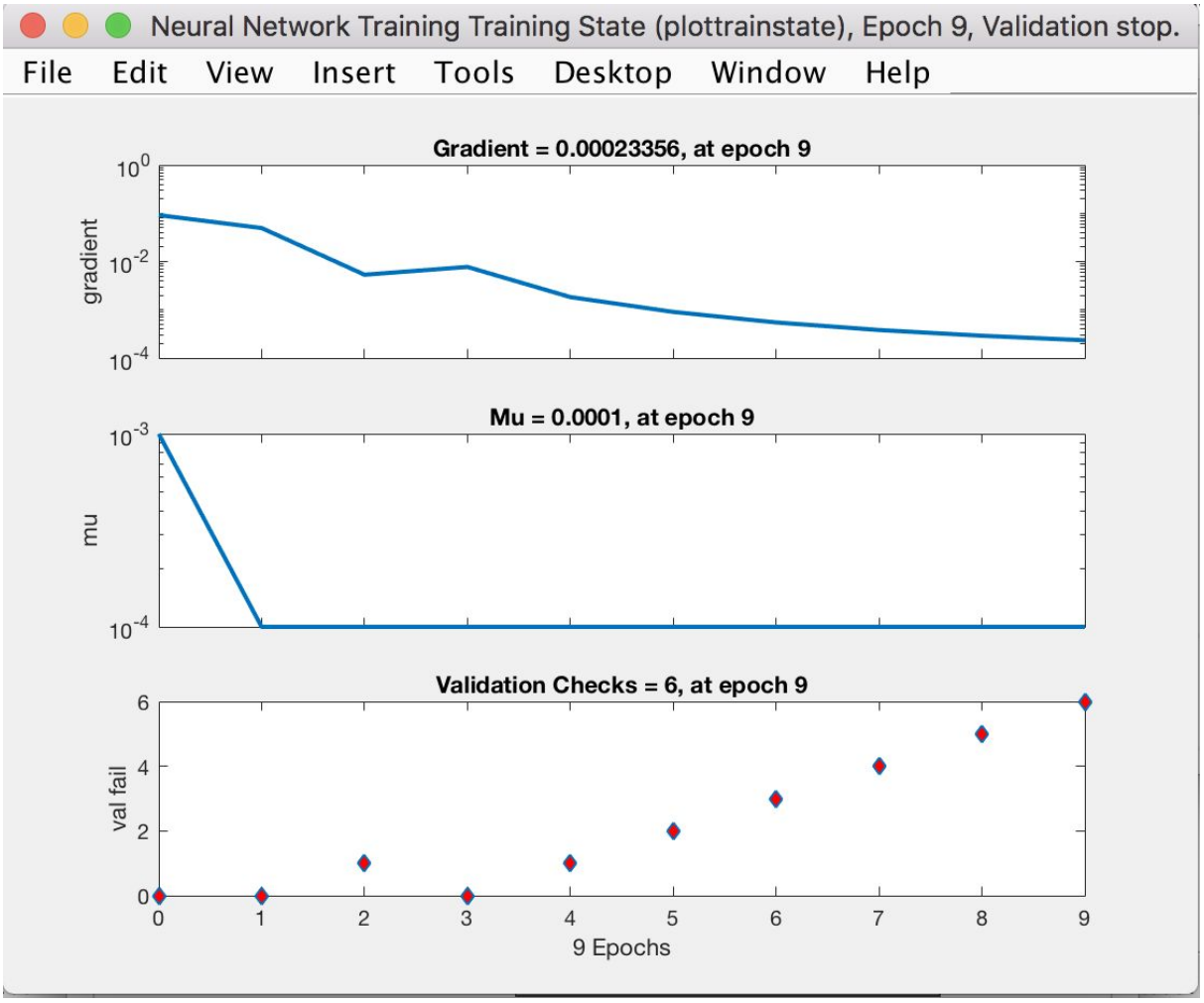


График Training State

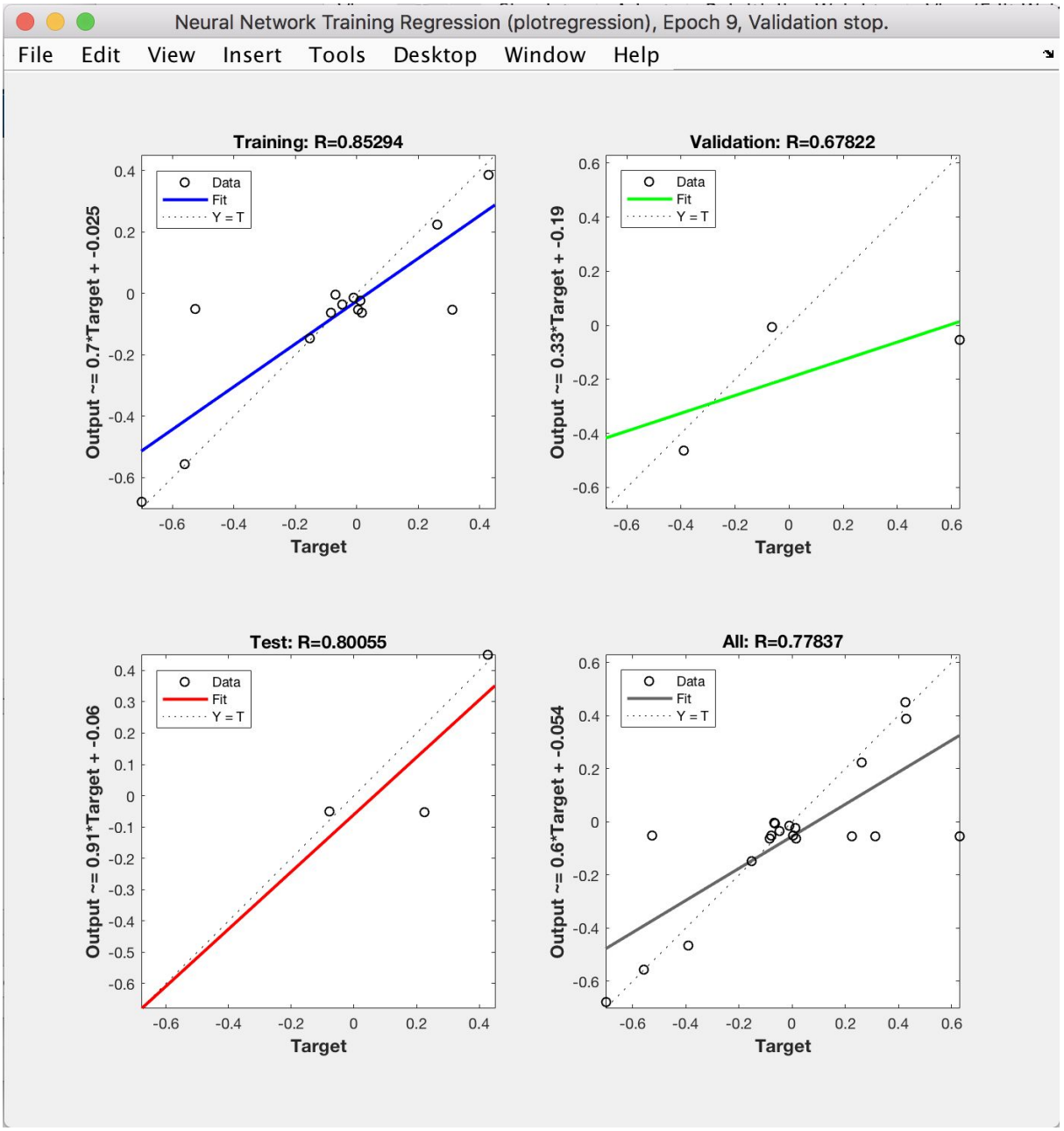


График Regression

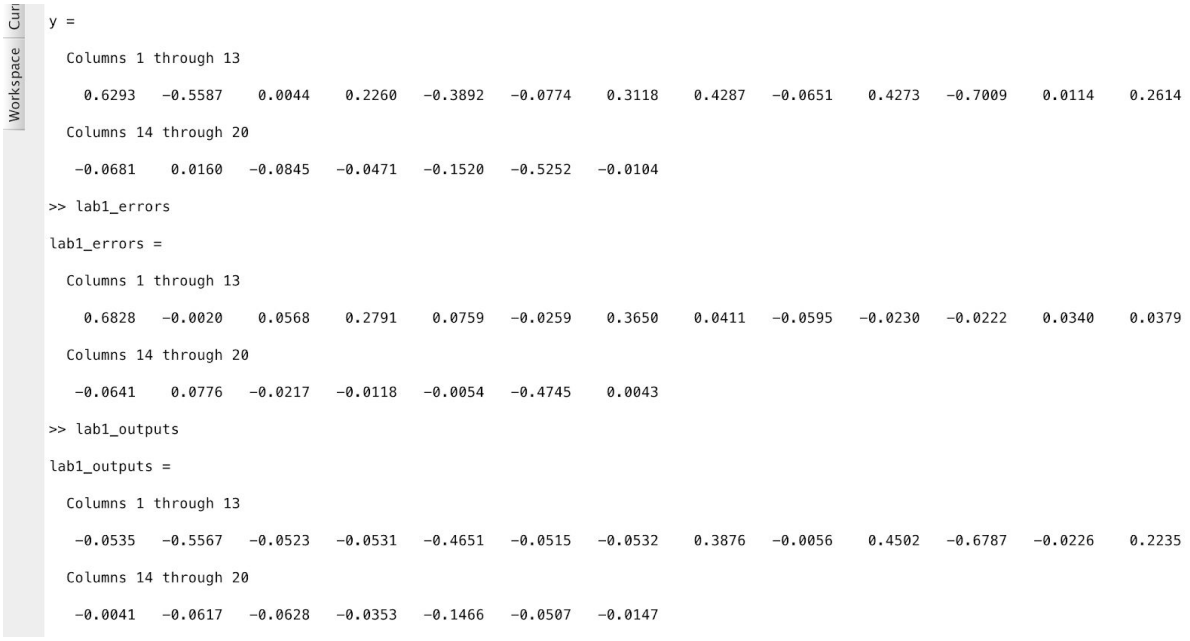
Результаты обучения

Результаты обучения сети можно посмотреть в окне *Network/Data Manager*, а именно выбирая параметры *lab1_outputs* и *lab1_errors*:

$lab_1 outputs$ = [-0.053503 -0.55667 -0.052302 -0.053126 -0.4651 -0.051539 -0.053213 0.3876 -0.005608 0.45023 -0.67868 -0.022636 0.22348 -0.0040626 -0.061654 -0.062779 -0.035276 -0.1466 -0.050654 -0.014699]

$lab_1 errors$ = [0.68278 -0.002041 0.05675 0.27909 0.075925 -0.025869 0.36503 0.041095 -0.059484 -0.022976 -0.022227 0.034042 0.037935 -0.064062 0.077612 -0.021746 -0.011804 -0.0053586 -0.47454 0.0043324]

Кроме того, мы можем выгрузить полученные результаты в рабочую область, что позволяет печатать/обрабатывать значения в дальнейшем:



Результаты работы сети

А также получить сам объект сети:

```
>> lab1

lab1 =

    Neural Network

                name: 'Custom Neural Network'
            userdata: (your custom info)

    dimensions:

             numInputs: 1
             numLayers: 2
            numOutputs: 1
        numInputDelays: 0
        numLayerDelays: 0
     numFeedbackDelays: 0
     numWeightElements: 9
            sampleTime: 1

    connections:

           biasConnect: [1; 1]
          inputConnect: [1; 0]
          layerConnect: [0 0; 1 0]
         outputConnect: [0 1]

    subobjects:

                 input: Equivalent to inputs{1}
                output: Equivalent to outputs{2}

                inputs: {1x1 cell array of 1 input}
                layers: {2x1 cell array of 2 layers}
               outputs: {1x2 cell array of 1 output}
                biases: {2x1 cell array of 2 biases}
```

Свойства объекта сети

Выводы

При выполнении первой лабораторной работы были получены необходимые навыки в построении базовой модели нейронной сети, умение устанавливать параметры сети, а также возможности ее обработки.

Часть вторая

Цель работы

Изучить свойства линейного нейрона

Содержание

Постановка задачи

Построить и натренировать простую линейную нейронную сеть (персептрон) для выполнения логических функций двух переменных OR, AND, NOR, NAND, XOR, NXOR.

Исходные данные

Для задач:

- AND:
  X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
  Y_train = {0, 0, 0, 1};
- OR:
  X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
  Y_train = {0, 1, 1, 1};
- NAND:
  X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
  Y_train = {1, 1, 1, 0};
- NOR:
  X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
  Y_train = {1, 0, 0, 0};
- XOR:
  X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
  Y_train = {0, 1, 1, 0};
- NXOR:
  X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
  Y_train = {1, 0, 0, 1};

Алгоритм решения

Обучение OR, NOR, AND, NAND

Построим персептрон с помощью функции *newp*:

```
net = newp(inputs, neurons);
```

где `inputs` - [0 1; 0 1], поскольку все логические значения вкладываются в этот интервал, `neurons` - 1, поскольку достаточно одного.

Количество циклов адаптации:

```
net.adaptParam.passes = 20;
```

Тренируем сеть:

```
net = train(net, X_train, Y_train);
```

Симуляция работы нейрона на тестовых входах:

```
Y_test = sim(net, X_test);
```

Все это помещается в один скрипт:

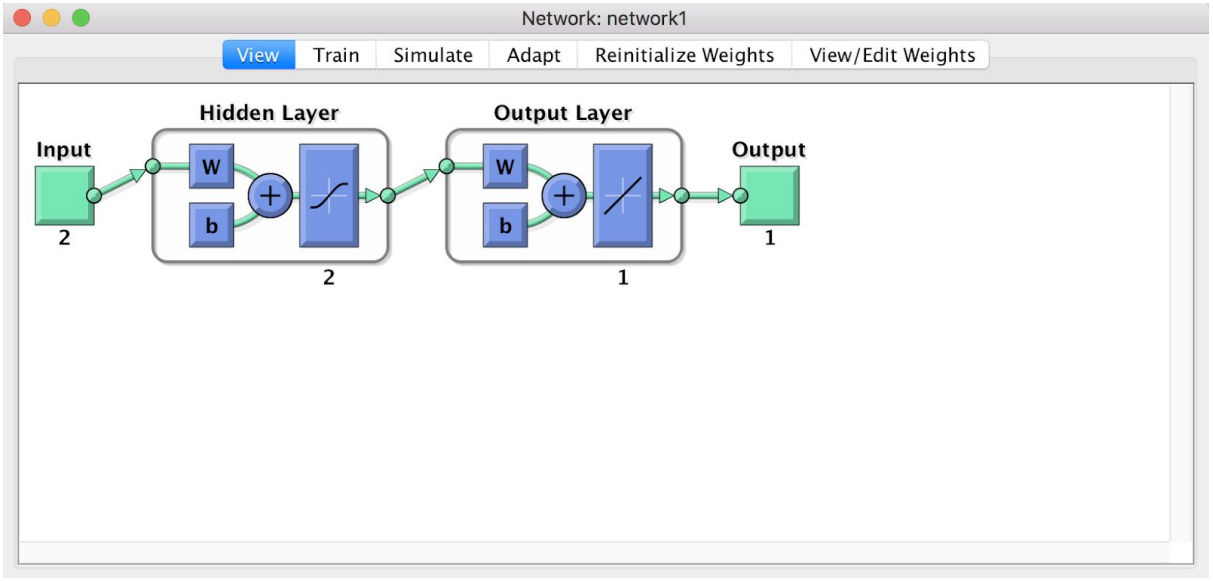
```
function Main()
    disp('Solving for loqical AND');
    X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
    Y_train = {0, 0, 0, 1};
    X_test = X_train;
    Y = build_logiql(X_train, Y_train, X_test, [0 1; 0
1], 1);
    % Output:
    Y

    disp('Solving for loqical OR');
    X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
    Y_train = {0, 1, 1, 1};
    X_test = X_train;
```

```
        Y = build_logiql(X_train, Y_train, X_test, [0 1; 0
1], 1);
        % Output:
        Y

        disp('Solving for loqical NAND');
        X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
        Y_train = {1, 1, 1, 0};
        X_test = X_train;
        Y = build_logiql(X_train, Y_train, X_test, [0 1; 0
1], 1);
        % Output:
        Y

        disp('Solving for loqical NOR');
        X_train = {[0; 0] [0; 1] [1; 0] [1; 1]};
        Y_train = {1, 0, 0, 0};
        X_test = X_train;
        Y = build_logiql(X_train, Y_train, X_test, [0 1; 0
1], 1);
        % Output:
        Y
    end

    function [Y_test] = build_logiql(X_train, Y_train,
X_test, inputs, neurons)
    %     declaring a network with two inputs [0, 1] interval
each
        net = newp(inputs, neurons);
        net.adaptParam.passes = 20;
        net = train(net, X_train, Y_train);
        Y_test = sim(net, X_test);
    end
```

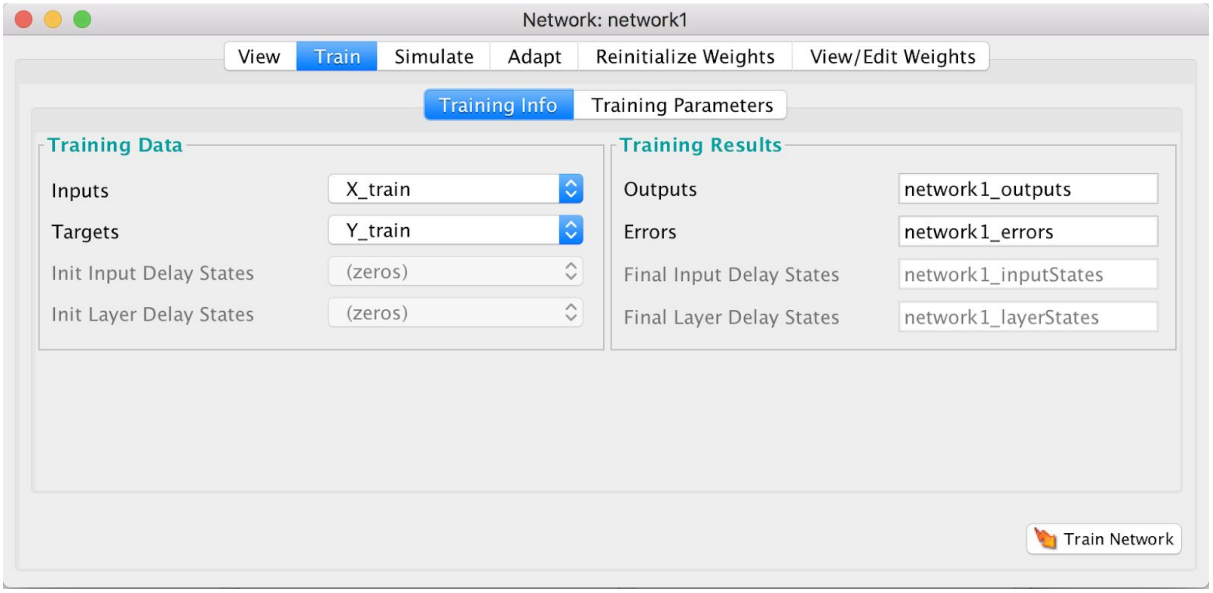
Обучение XOR, NXOR

Как известно, точки выполнения операций XOR, NXOR не могут быть линейно разделяемыми, поэтому использование линейного нейрона (персептрона) не представляется возможным. Для обучения нейросети выполнению операций необходимо использовать как минимум 2 слоя. Для

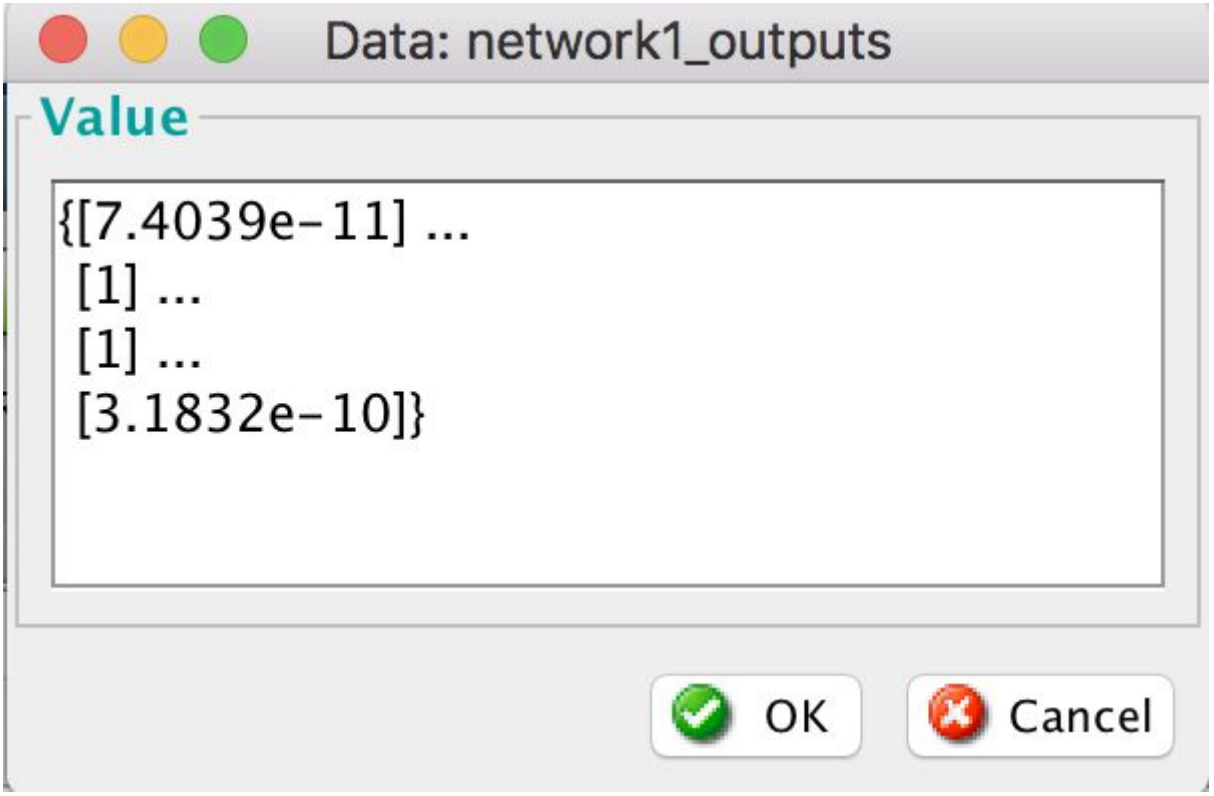
этого используем полученные знания и навыки из предыдущей л.р. Шаги будут выполнены похожие, поэтому приведу лишь необходимые скриншоты:



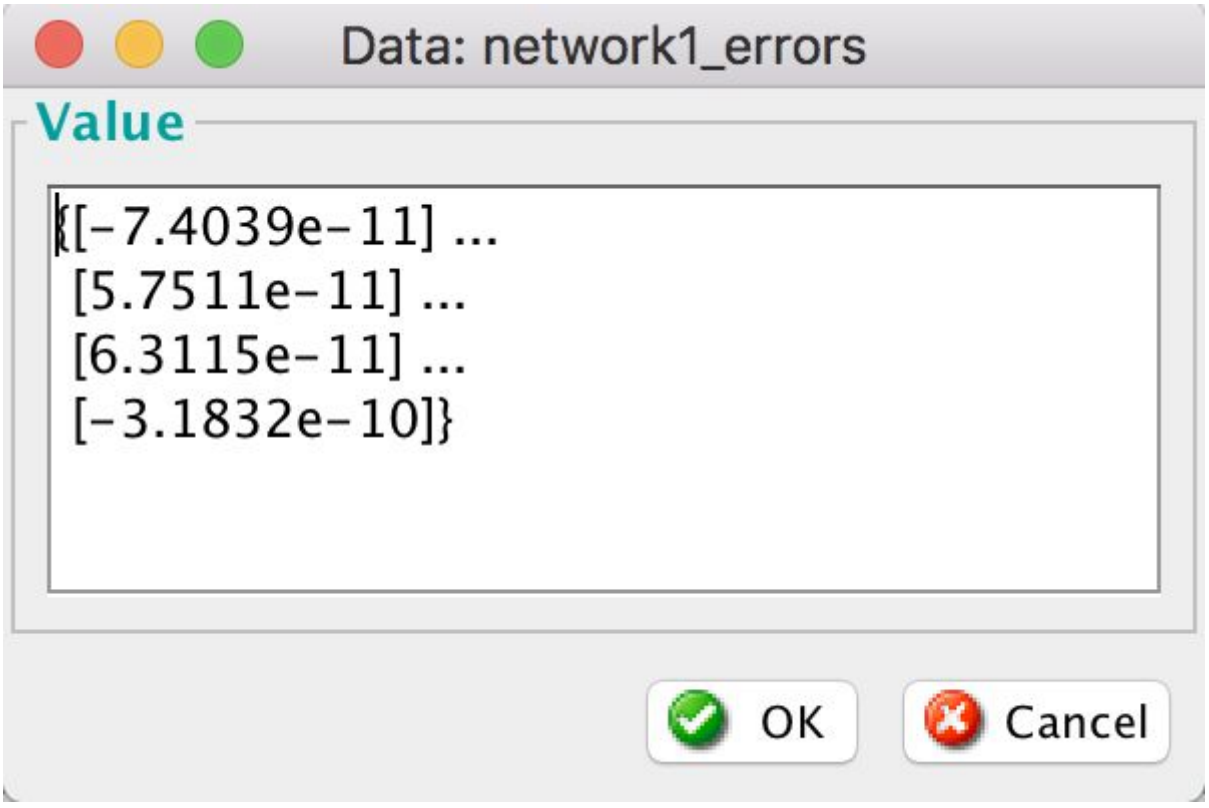
Создали сеть с двумя скрытыми слоями



Указали Inputs/Targets



Как видим, для XOR выход тестовых данных сходится с заложенными тестовыми данными



Ошибки нулевые

Аналогичная процедура проводится для операции NXOR, изменения касаются лишь входных параметров.

Выводы

При выполнении данной лабораторной работы была изучена теория персептрона, были получены навыки в его построении и применении для простых линейно разделяемых функций, а также с использованием графического интерфейса была построена нейронная сеть с двумя скрытыми слоями для обучения с использованием данных, линейно не разделяемых.