**CS657 Assignment 4 Report**

**Procedure**

**Step1: Initialization**

getInputValueWithNoise(); //get input value with noise

getRandomWeight(); // initialize weights for all neurons

getRandomThreshould(); //initialize threshold for all neurons

**Step2: Activation**

getHiddenLayerOutput(); //calculate hidden layer output

getOutputLayerOutput(); // calculate output

**Step3: Weight training**

getErrorAndErrorGradient(); //calculate error and gradient for output

getHiddenWeigthCorrection(); //calculate hidden layer weight correction

getErrorGradientForHiddenLayer(); //calculate input layer gradient

getInputWeightCorrection(); //calculate input layer weight correction

updateWeightsAndThresholds(); //update all weights and thresholds for next iteration

**Step4: Iteration**

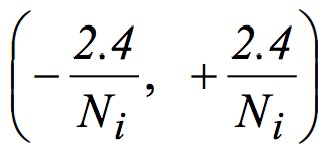
Iterate until squared error < 0.001

**Result of Training:**

**Experimental run parameters:**  
a=0.1

Number of hidden neurons: 10

Original weights and thresholds:



where Ni is the input number of the neuron.

**Experimental run result:**

1. **Epoch to convergence is** 46505
2. **The final weights for input layer are:**

the W00 is 3.8008053509209994

the W01 is 1.4991926319785205

the W02 is -1.5226873354331967

the W03 is 0.5260904331528309

the W04 is 4.376393682985628

the W05 is 0.10215751291757183

the W06 is -3.22838595026314

the W07 is -4.4168039228893665

the W08 is -0.10688158966474111

the W09 is 2.077787455655991

the W10 is -4.1984207996284315

the W11 is 1.8677427732595764

the W12 is -1.6414137159236792

the W13 is -0.33692696125668975

the W14 is 5.518916281571226

the W15 is 0.18069209948630308

the W16 is 3.8429703686821113

the W17 is -5.0579831344435595

the W18 is 1.6778171522925

the W19 is -3.0359874767779207

1. **The final weights for output layer are:**

the W00 is -5.6219621226074095

the W01 is 0.8490535812793275

the W10 is 0.2535485283359482

the W11 is 2.926287476164985

the W20 is -1.6289849183105425

the W21 is -2.003562636869403

the W30 is -0.7848449959567777

the W31 is -0.3089032755743213

the W40 is 6.83670687085285

the W41 is 2.698431605926706

the W50 is -0.7427592167978425

the W51 is -0.10949320045006129

the W60 is -4.656726316675144

the W61 is -0.12139772635813256

the W70 is 7.003020741911057

the W71 is -7.334436476977564

the W80 is -0.1778608885676996

the W81 is 1.2843307533825072

the W90 is 2.081079632139378

the W91 is -1.4893420527245051

1. **The final thresholds for hidden layer are:**

threshould-0 is -2.2284818963411652

threshould-1 is 1.7814816931611417

threshould-2 is -0.6780765402212773

threshould-3 is -0.8342864163675311

threshould-4 is 1.7263549205103739

threshould-5 is -0.9597140506857942

threshould-6 is -2.094826172516369

threshould-7 is -6.84730684536872

threshould-8 is 0.04804433439276019

threshould-9 is 1.4714511971339164

1. **The final thresholds for output layer are:**

threshould-0 is 1.3315073900119643

threshould-1 is 1.329129323716811

**How you chose the number of neurons in each layer**

**Number of input layer:**Each time the input values are X0 and X1, so the number of input layer is 2

**Number of hidden layer:**There are some empirically-derived rules-of-thumb, of these, the most commonly relied on is 'the optimal size of the hidden layer is usually between the size of the input and size of the output layers.

In this case I choose the number of hidden layer by experimental run, I tried some of the neuron number and the result is as follow:

Table-1 Number of neuron in hidden layers vs. epoch to convergence

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Neuron Num | 5 | 10 | 15 | 20 | 25 |
| Epoch to convergence | 50047 | 49225 | 36398 | 35252 | 40858 |

So I choose 20 as the number of hidden layer neuron for quick convergence.

**Number of output layer:**

Output values are always sum and carry so number of output layer is 2

**How you chose the activation function?**

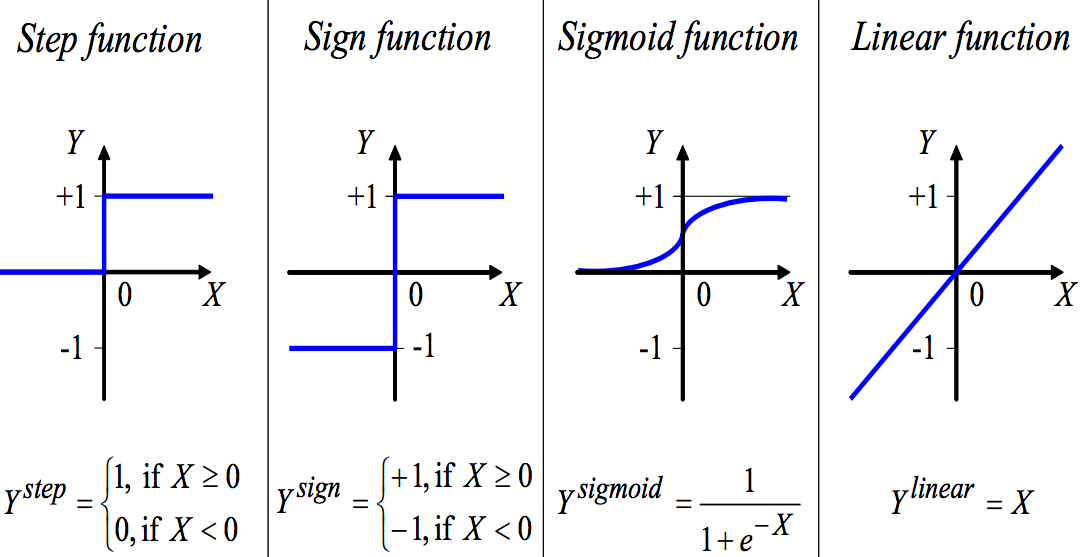
****

Fig.1 activation function

In all these activation functions, the y value of sigmoid function is between 0 and 1 which is approximate to the input and output value (0 and 1) in this problem. In addition, the curve of sigmoid function is smooth and continuous, which is similar to the input with noise and output with error. Therefore I choose Sigmoid function in this problem.

**Learning rate, Epoch and squares error:**

Produce graphs to show the mean squares error (learning rate, etc.) versus Epoch

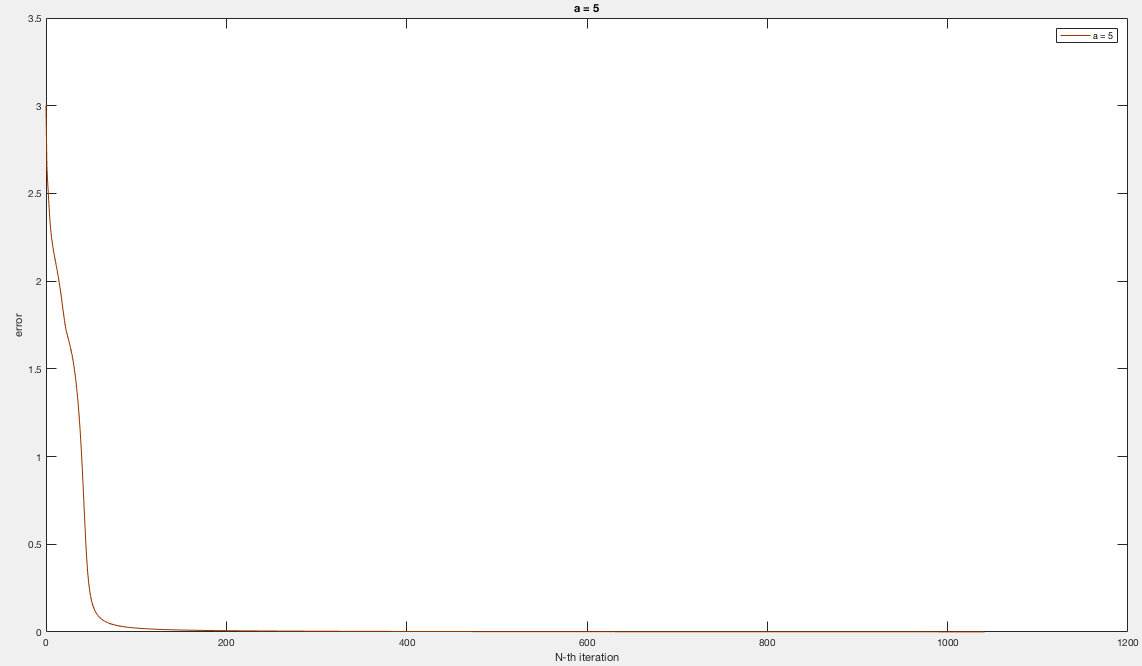


Fig.1 Learning rate = 5

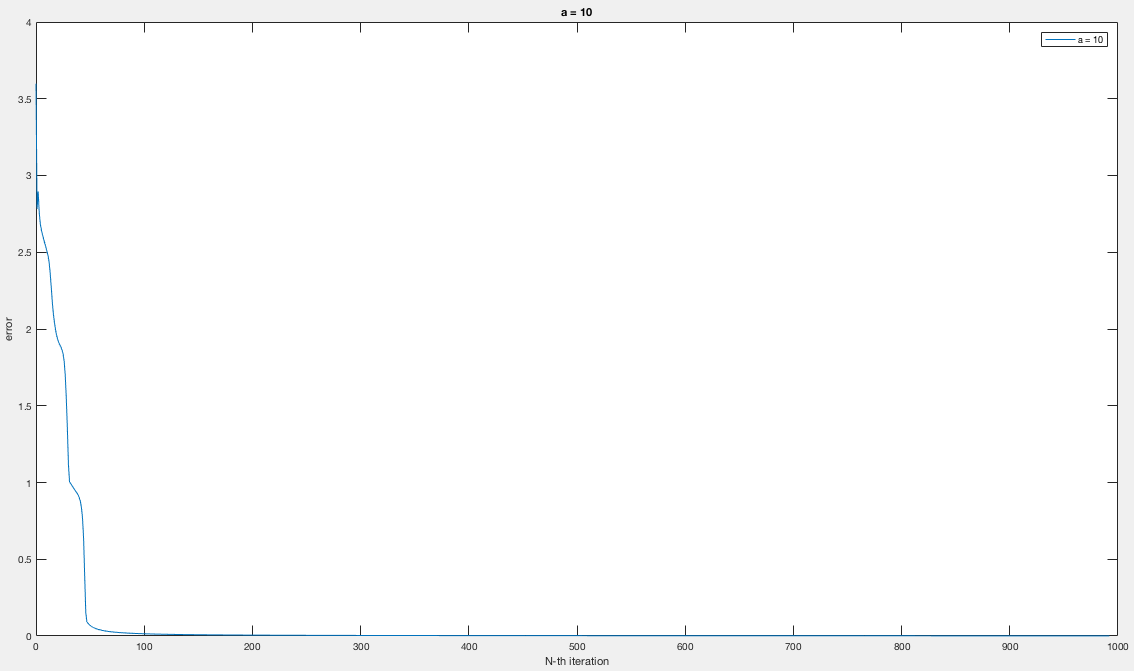


Fig.2 Learning rate = 10

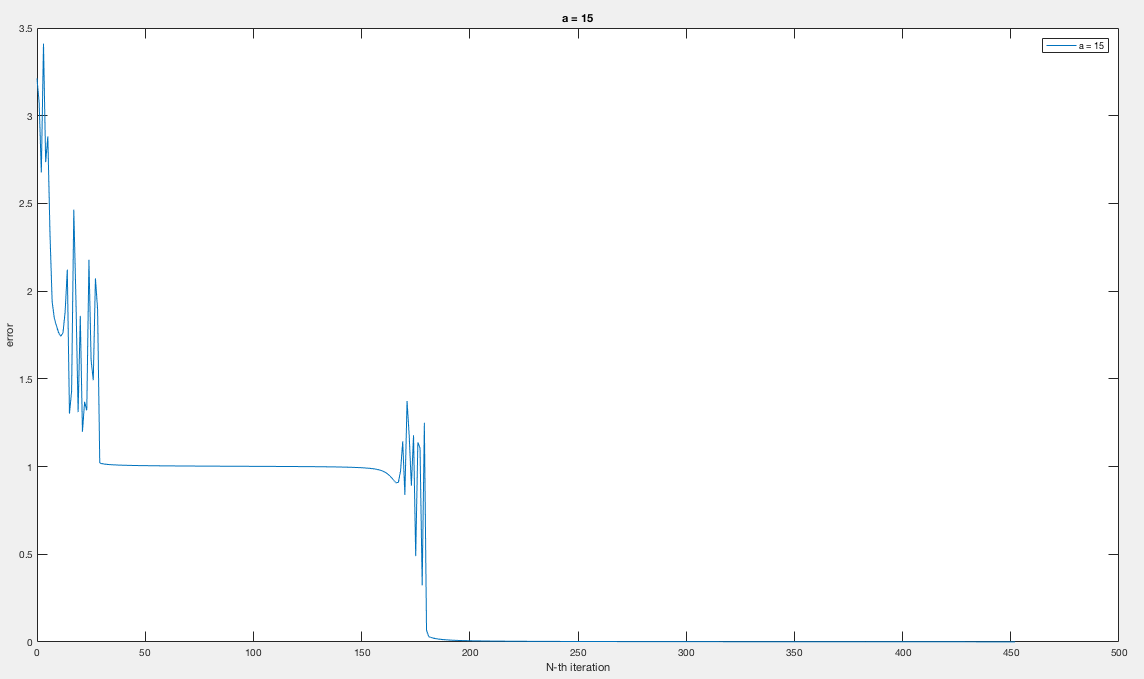


Fig.3 Learning rate = 15

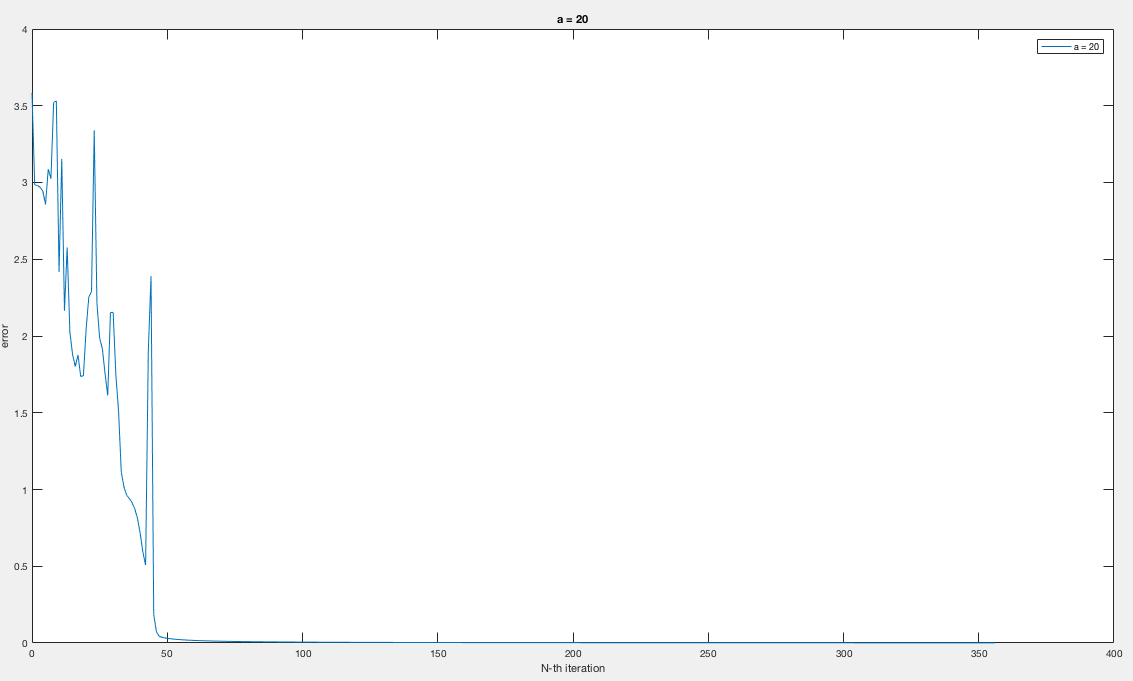


Fig.4 Learning rate = 20

**Effect of learning rate**

The greater the learning rate, the faster to reach convergence. However, error fluctuates if the learning rate is not appropriate for the system

**Result of Test Run**

Generate 100 test values with noise to test the trained system:

I define a tolerance range (-0.2,0.2) to test the trained system, so if

|output value – desire value|<0.2

which means the value is in the range of tolerance, so it’s a correct output. Here below is the result:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Num of Neuron in hidden layer | 5 | 10 | 15 | 20 | 25 |
| error | 9.999939432537026E-4 | 9.999963045513307E-4 | 9.999981018760987E-4 | 9.999922803655549E-4 | 9.999733923994408E-4 |
| Accuracy | 100% | 100% | 100% | 100% | 100% |

**Findings**

1. The greater the learning rate, the faster to reach convergence
2. The error fluctuates if the learning rate is too great or not appropriate for the system.
3. After training the neural network, the accuracy is 100%
4. The number of neurons in hidden layer can be determined by experimental run.

**Conclusions**

In sum, after training the neural network, the program can get the error smaller than a threshold (0.001 in this case), the epoch to reach the convergence varies based on how many hidden neurons used, how large the dataset to train the system and the value of learning rate. The accuracy of a trained neural network is 100% according to the experimental run.