**CHECKLIST:**

[**X**] Early stopping

[**X**] Cost functions

[**X**] Quadratic

[**X**] Cross-entropy

[**X**] log-likelihood

[**X**] allow choice of cost function with a parameter

[**X**] Momentum

[**X**] L2 Regularization

[**X**] Better initial weights

[**X**] Transfer functions

[**X**] tanh

[**X**] softmax

[**X**] ReLU

[**X**] Minibatch shuffling

[**X**] Learning rate schedule description

[**X**] Returning learned network

Comment: I have built up my code from the previous project and there I have used weights and biases as global variables. Both of them area accessible after execution. This is as good as returning both weights and biases.

[**X**] Returning accuracy and costs for plotting

[**X**] Did NOT include the MNIST data with my submission

Incomplete, Details \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Not sure, Details \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Instructions:**

Code is written in octave 4.0.3 version. Load the all the files in the same folder including the data files.

In the submission folder there are three files:

1. xorScript.m – For xor.csv
2. irisScript.m - For iris.csv
3. mnistScript.m – For mnist data

Open any of these file in the octave editor and change the value of any of the below parameters:

1. layerVector – row vector showing number of neurons in each layer including input and output layer
2. newEpoch – number of epochs to run
3. batchSize – mini batch size
4. eta – value learning rate
5. transferFuncName – name of the transfer function. It can have following values:
   1. “sigmoid”
   2. “relu”
   3. “softmax”
   4. “tanh”
6. costFuncName – name of the cost function. It can have following values
   1. “cross”
   2. “quad”
   3. “log”
7. mom –
8. lambdaReg – lambda value for L2 regularization

Apart from these there are other optional parameters

1. splitinfo – row vector containing the percentage to split the input data into training, test and validation data
2. oldWeights – To pass old weights calculated previously
3. oldBiases – To pass old biases calculated previously

After making changes save the file and open the octave command window and run the following command:

**>> source <scriptname>**

Script can also be executed from the editor window using the run button.

If you are running for a different dataset, then call the following function in the command window:

**start(input,targer,layerVector,numEpoch,batchSize,eta,transferFuncName,costFuncName,lambdaReg,mom);**

where layerVector is row vector.

**Description:**

Following files are included in the attachment:

1. xorScript.m -> Script for executing XOR.csv data
2. irisScript.m -> Script for executing iris.csv data
3. mnistScript.m -> Script for executing mnist data
4. Globalvariables.m -> Lists all the global variables used in the application
5. start.m -> Starting point for the application
6. backpropagation.m -> Implements back propagation algorithm
7. shuffle.m -> Shuffles input data and creates training, validation and test data
8. cross\_entropy.m -> Implements cross entropy cost function
9. log\_cost.m -> Implements log cost function
10. quadratic.m -> Implements quadratic cost function
11. sigmoid.m -> Implements sigmoid transfer function
12. sigmoid\_prime.m -> First order of sigmoid function
13. softmax.m -> Implements softmax transfer function
14. softmax\_prime.m -> First order derivative of softmax function
15. tanhTransfer.m -> Implements tanh transfer function
16. tanhTransfer\_prime.m -> First order derivative of tanh function
17. relu.m -> Implements relu transfer function
18. relu\_prime.m -> First order derivative of relu function
19. cost\_derivative.m -> Implementing cost derivative
20. deltaValForLogAndCross.m -> Calculating last delta value for log and cross entropy function
21. deltaValForQuadratic.m -> Calculating last delta value for quadratic function
22. round\_correctCal.m -> Rounding for sigmoid, tanh and relu
23. softmax\_correctCal.m -> Rounding for softmax function
24. sumOfWeightsSquared.m -> Calculating second parameter of cost function (Eq. 85)
25. plotAccuraciesAndCosts.m -> For plotting accuracies

All the codes are well-documented and written for understanding rather than efficiency. The first 3 files mentioned above are the script files which makes executing different scenarios easy. All those 3 files are just loading the respective data and calling the start function.

**Code:**

**Analysis:**

**Output:**

Dataset: iris.csv

Run1: Epochs: 40, Hidden: 20, Batch: 10, eta: 0.1, Transfer Function: sigmoid, Cost Function: Cross Entropy, Momentum: 0.3, lambda: 5

Run2: Epochs: 40, Hidden: 20, Batch: 10, eta: 0.1, Transfer Function: relu, Cost Function: Cross Entropy, Momentum: 0.3, lambda: 5

Run3: Epochs: 40, Hidden: 20, Batch: 10, eta: 0.1, Transfer Function: relu, Cost Function: Cross Entropy, Momentum: 0, lambda: 5

Dataset: MNIST

Run1: Epochs: 30, Hidden: 30, Batch: 10, eta: 3.0, Transfer Function: sigmoid, Cost Function: Quadratic, Momentum: 0.3, lambda: 5

Run2: Epochs: 30, Hidden: 30, Batch: 10, eta: 3.0, Transfer Function: softmax, Cost Function: log, Momentum: 0.3, lambda: 0

Run3: Epochs: 30, Hidden: 30, Batch: 10, eta: 3.0, Transfer Function: softmax, Cost Function: log, Momentum: 0.3, lambda: 5

Dataset: xor.csv

Run1: Epochs: 20, Hidden: [3,2], Batch: 1, eta: 0.1, Transfer Function: sigmoid, Cost Function: Cross Entropy, Momentum: 0.3, lambda: 5

Run2: Epochs: 20, Hidden: [3,2], Batch: 1, eta: 0.1, Transfer Function: tanh, Cost Function: Cross Entropy, Momentum: 0.3, lambda: 5

Run3: Epochs: 20, Hidden: [3,2], Batch: 1, eta: 0.1, Transfer Function: relu, Cost Function: Cross Entropy, Momentum: 0.3, lambda: 5