Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network

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
[X] Returning accuracy and costs for plotting
[X] Did NOT include the MNIST data with my submission

Not sure, Details: Second test case for the iris dataset didn't perform as expected.

INSTRUCTIONS

The project code is stored in the file – **BackPropProj2.m**

function [weight, bias, acc, cost] = BackPropProj2(inputs, targets, nodeLayers, numEpochs, batchSize, eta, split, momentum, lambda, transFunction, costFun)

Input parameters to the function:
inputs: a matrix of input values
targets: a matrix of target/output values
nodeLayers: a vector with number of nodes in each layer (total number of node layers)
numEpochs: desired number of epochs to run
batchSize: number of instances in a mini-batch
eta: learning rate for gradient descent
split: splits the data into training, testing and validation dataset (like [80 10 10])
momentum: momentum coefficient
lambda: lambda, the coefficient for L2 regularization
transFunction: activation or transfer function – 'sigmoid','tanh','relu' or 'softmax'

Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network

costFun: cost or error function – 'quadratic', 'cross-entropy' or 'log-likelihood'

Outputs generated:
weight: weights for each layer
bias: bias for each layer
acc: a cell array where first cell represents the training accuracy, and the second and the third represent validation and testing accuracies respectively
cost: a cell where first cell represents the training cost, and the second and the third represent validation and testing cost respectively

Invoking the function on the command window:

```
>> [weight bias acc cost] = BackPropProj2(irisinputs, iristargets, [4,20,3], 40, 10, 0.1, [80, 10, 10], 0.3, 5, 'sigmoid', 'cross');
```

Files included with submission:
BackPropProj2.m, dTransferPrime.m, transfer.m, SigmoidPrime.m

DESCRIPTION:

This project is implemented in Matlab and the code is well documented which helps to understand the functionality of the network.

function [weight, bias, acc, cost] = BackPropProj2(inputs, targets, nodeLayers, numEpochs, batchSize, eta, split, momentum, lambda, transFunction, costFun)

Overview:
The dataset is partitioned in train, test and validation sets using the ratio mentioned in the split parameter. 'dividerand' method is used to derive partitions.
Weights and biases are initialized by randn function with a mean of zero and std deviation of 1/sqrt(n-1).
Further, the input matrix is divided into mini batches as specified by the batchSize parameter and batch shuffling is done using the randperm function.
'transFunApply' function takes weighted inputs and applies transfer function as requested.
The network is learned epoch by epoch and batch after batch –

> Feedforward the network - For each layer z{layer} and a{layer} is calculated
> The output layer Error for each activation function is computed
> Hidden layer errors are then calculated and errors are back propagated through the network.
> The momentum parameter is added to the Gradient Descent to find the minimum coefficients.
> Final output layer errors, accuracies and correct values are computed and evaluated.

Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network

Early stopping strategy used- Validation cost is checked and halted when it increases after 65% of the total number of epochs.
Accuracy and cost plots are then generated.

CODE:

BackPropProj2.m

```
%CSC578 - Project 2
%Improving a Neural Network
%Oct 22, 2017

function [weight, bias, acc, cost] = BackPropProj2(inputs, targets, nodeLayers, numEpochs, batchSize,
eta, split, momentum, lambda, transFunction, costFun)
  %Partition dataset into train, test and validation sets using split attribute
  if (isempty(split))
      trainInputs = inputs;
      testInputs = [];
      valInputs = [];
      trainTargets = targets;
      testTargets = [];
      valTargets = [];
  else
      % using the dividerand function to split - [trainInd,valInd,testInd] =
dividerand(Q,trainRatio,valRatio,testRatio)
      [trainInd, valInd, testInd] = dividerand((size(inputs,2)), split(1)/100, split(2)/100,
split(3)/100);
      trainInputs = inputs(:, trainInd);
      valInputs = inputs(:, valInd);
      testInputs = inputs(:, testInd);
      trainTargets = targets(:, trainInd);
      valTargets = targets(:, valInd);
      testTargets = targets(:, testInd);
  end

  %Initializations
  L = size(nodeLayers,2); %total number of node layers
  weight = cell(1,L); %Initialize a cell to hold the weight
  bias = cell(1,L); %Initialize a cell to hold the biases
  bigDeltaWeight = {}; %Initialize a cell to hold deltas for the weights
  bigDeltaBias = {}; %Initialize a cell to hold deltas for the bias
  trainInputSize = size(trainInputs,2);%Hold size of the train inputs
  valInputSize = size(valInputs,2);%Hold size of the validation inputs
  testInputSize = size(testInputs,2);%Hold size of the test inputs
  batchValues = {}; %Initialize a cell to hold batch values
  targetValues = {}; %Initialize a cell to hold target values
  batchIndex = 1; %Initialize a counter variable
  %To store costs for training,testing and validation sets
  trainCost = zeros(1,numEpochs);
  testCost = zeros(1,numEpochs);
  valCost = zeros(1,numEpochs);
  %Initialize cells for accuracy and cost to hold the final train,validation and test values
  acc = {};
  cost = {};

  for layer = 2 : L
    weight{layer} = randn(nodeLayers(layer), nodeLayers(layer-1))/sqrt(nodeLayers(layer-1));
    bias{layer} = randn(nodeLayers(layer), 1);
  end

  %Dividing the input matrix into mini batches
  %Increment by the value batchSize(step) for each iteration
  for initPos = 1:batchSize:trainInputSize
    if trainInputSize - initPos >= batchSize
        miniBatch = trainInputs(:, initPos:initPos+batchSize-1);
        batchValues{batchIndex} = miniBatch;
        target = trainTargets(:, initPos:initPos+batchSize-1);
        targetValues{batchIndex} = target;
        batchIndex = batchIndex + 1;
    else
        miniBatch = trainInputs(:, initPos:end);
        batchValues{batchIndex} = miniBatch;
        target = trainTargets(:, initPos:end);
        targetValues{batchIndex} = target;
    end
  end

  %To diaplay output headers
  fprintf('|        TRAIN           ||      VALIDATION          ||          TEST\n');
  fprintf('------------------------------------------------------------------------------------------
\n');
  fprintf('Epoch  | Cost | Corr  |  Acc  ||  Cost |  Corr |  Acc ||  Cost | Corr |  Acc \n');
  fprintf('------------------------------------------------------------------------------------------
\n');

  %Displaying user input for 'softmax' and 'relu' transfer functions
  if (strcmp(transFunction, 'softmax'))
      userInSoftMax = input('User Input required: Softmax function can only be used in the last layer
\n');
  elseif (strcmp(transFunction,'relu'))
      userInReLu = input('User Input required: Relu can only be used in the hidden layers \n');
  end

  function a = transFunApply(input)
    if (strcmp(transFunction, 'sigmoid'))
        a = transfer(input, transFunction);
    elseif (strcmp(transFunction, 'tanh'))
        if layer == L
            a = transfer(input, transFunction);
        else
            a = tanh(input);
        end
    elseif (strcmp(transFunction, 'relu'))
        if layer == L
            a = transfer(input, userInReLu);%ReLu cannot be used in the last layer.
        else
            a = max(0, input);
        end
    elseif (strcmp(transFunction, 'softmax'))
        if layer == L
            a = transfer(input, transFunction);
        else
            a = transfer(input, userInSoftMax);%Cannot use softmax if it's the last layer
        end
    else
        error('Not a valid Transfer function. Input sigmoid,tanh,relu or softmax')
    end
  end

  %Loop through each epoch and batch
  for epoch = 1:numEpochs
      %Using the randperm function for minibatch shuffling
      random = randperm(size(batchValues,2));
      batchCounter = 1;
      for batch = 1:size(batchValues,2)
          z = {}; %Initialize a cell to hold values for the intermediate nodes
          a = {}; %Initialize a cell to hold the activation function
          a{1} = batchValues{random(batchCounter)}; %Input value of the batch is assigned to the first
element of the activation cell

          %Feedforward the network - For each layer calculate z{layer} and a{layer}
          for layer = 2 : L
             z{layer} = weight{layer} * a{layer - 1} + bias{layer};
             a{layer} = transFunApply(z{layer});
          end

          delta = {}; %Initialize a cell to hold the error values
          error = (a{L} - targetValues{random(batch)});

          %Calculating the output layer Error for each activation function
          if (strcmp(transFunction, 'tanh'))
            delta{L} = error .* (1-tanh(z{L}).^2);
          elseif (strcmp(transFunction, 'sigmoid'))
            delta{L} = error .* SigmoidPrime(z{L});
          elseif (strcmp(transFunction, 'softmax'))
            %If softmax is the trans function for the last output layer
            delta{L} = error .* ones(size(z{L}));
          elseif (strcmp(transFunction, 'relu'))
            if (strcmp(userInReLu, 'softmax'))
```

```
                delta{L} = error .* ones(size(z{L}));
            else
                delta{L} = error .* (dTransferPrime(z{L}, userInReLu));
            end
          end

          % Back propagate error through the network from L to 2nd layer
          % Calculating Hidden layer errors
          for layer = (L - 1) : -1 : 2
              if (strcmp(transFunction, 'softmax'))%If softmax is the transfer function - it cannot be
used in hidden layers
                delta{layer} = (weight{layer + 1}.' * delta{layer + 1}) .* dTransferPrime(z{layer},
userInSoftMax);
              else
                delta{layer} = (weight{layer + 1}.' * delta{layer + 1}) .* dTransferPrime(z{layer},
transFunction);
              end
          end

         %Gradient Descent and finding the minimum
         %For each layer from L to 2nd layer
          for layer = L : -1 : 2
              if epoch == 1 && batch == 1
                  weight{layer} = weight{layer} - eta/length(batchValues{random(batch)}) * delta{layer}
* a{layer - 1}.';
                  bias{layer} = bias{layer} - eta/length(batchValues{random(batch)}) *
sum(delta{layer}, 2);
                  bigDeltaWeight{layer} = eta/length(batchValues{random(batch)}) * delta{layer} *
a{layer - 1}.';
                  bigDeltaBias{layer} = eta/length(batchValues{random(batch)}) * sum(delta{layer}, 2);
              else %Using the momentum parameter to find the minimum coefficients
                  weight{layer} = weight{layer} + bigDeltaWeight{layer};
                  bias{layer} = bias{layer} + bigDeltaBias{layer};
                  bigDeltaWeight{layer} = momentum .* bigDeltaWeight{layer} -
eta/length(batchValues{random(batch)}) * delta{layer} * a{layer - 1}.';
                  bigDeltaBias{layer} = momentum .* bigDeltaBias{layer} -
eta/length(batchValues{random(batch)}) * sum(delta{layer}, 2);
              end
          end
          batchCounter = batchCounter + 1;
      end

      %Compute final output values after updating weights
      trainOut = {}; %Initialize a cell to hold the final training output values
      trainOut{1} = trainInputs; %Assign inputs to first element of the output cell
      valOut = {}; %Initialize a cell to hold the final validation output values
      valOut{1} = valInputs;
      testOut = {}; %Initialize a cell to hold the final testing output values
      testOut{1} = testInputs;

      ztrain = cell(1,L);
      zval = cell(1,L);
      ztest = cell(1,L);
      weightSum = 0;
      %For each layer from the 2nd layer
      for layer = 2 : L
          ztrain{layer} = (weight{layer} * trainOut{layer-1}) + (bias{layer});
          zval{layer} = (weight{layer} * valOut{layer-1}) + (bias{layer});
          ztest{layer} = (weight{layer} * testOut{layer-1}) + (bias{layer});

          trainOut{layer} = transFunApply(ztrain{layer});
          valOut{layer} = transFunApply(zval{layer});
          testOut{layer} = transFunApply(ztest{layer});

          weightSum = weightSum + sum(sum(weight{layer}.^2));
      end

      L2Train = lambda/(2*trainInputSize) * weightSum;
      L2Val = lambda/(2*valInputSize) * weightSum;
      L2Test = lambda/(2*testInputSize) * weightSum;

      %Compute the number of correct cases
      %correct = correct + sum(all(targets==round(output{L}),1),2);
      trainCorrect = sum(all(trainTargets == round(trainOut{L}),1),2);
      valCorrect = sum(all(valTargets == round(valOut{L}),1),2);
      testCorrect = sum(all(testTargets == round(testOut{L}),1),2);

      %Computing train, validation and test set accuracies
      trainAccuracy = trainCorrect/trainInputSize;
      valAccuracy = valCorrect/valInputSize;
      testAccuracy = testCorrect/testInputSize;

      % Computing cost
      if (strcmp(costFun,'quad')) %Quadratic
          trainCost = 1/(2*trainInputSize) * sum(sum((0.5*(trainTargets - trainOut{L}).^2)))+L2Train;
          valCost = 1/(2*valInputSize) * sum(sum((0.5*(valTargets - valOut{L}).^2)))+L2Val;
          testCost = 1/(2*testInputSize) * sum(sum((0.5*(testTargets - testOut{L}).^2)))+L2Test;
      elseif (strcmp(costFun,'cross')) %Cross-Entropy
          trainCost = -1/(trainInputSize) .* sum(sum(trainTargets .* log(trainOut{L}+eps) + (1-
trainTargets) .* log(1-trainOut{L}))+eps)+L2Train;
          valCost = -1/(valInputSize) .* sum(sum(valTargets .* log(valOut{L}+eps) + (1-valTargets) .*
log(1-valOut{L}))+eps)+L2Val;
          testCost = -1/(testInputSize) .* sum(sum(testTargets .* log(testOut{L}+eps) + (1-testTargets)
.* log(1-testOut{L}))+eps)+L2Test;
      elseif (strcmp(costFun,'log')) %log-likelihood
          trainCost = sum(-log(max(trainOut{L})+eps)/trainInputSize)+L2Train;
          valCost = sum(-log(max(valOut{L})+eps)/valInputSize)+L2Val;
          testCost = sum(-log(max(testOut{L})+eps)/testInputSize)+L2Test;
      end

      % store costs for early stopping
      trainCost(epoch) = trainCost;
      valCost(epoch) = valCost;
      testCost(epoch) = testCost;

      fprintf('%d\t| %.5f | %d/%d | %.5f || %.5f | %d/%d | %.5f || %.5f | %d/%d | %.5f\n', ...
      epoch,trainCost(epoch),trainCorrect,trainInputSize,trainAccuracy,...
      valCost(epoch),valCorrect,valInputSize,valAccuracy,...
      testCost(epoch),testCorrect,testInputSize,testAccuracy);

      %Early stopping conditions and plots
      %Storing accuracy and costs for each epoch
      acc{1}(epoch) = trainAccuracy;
      acc{2}(epoch) = valAccuracy;
      acc{3}(epoch) = testAccuracy;
      cost{1}(epoch) = trainCost(epoch);
      cost{2}(epoch) = valCost(epoch);
      cost{3}(epoch) = testCost(epoch);

      %Early Stopping based on Accuracy -- If all the input cases are correct - accuracy=1
      if trainCorrect == trainInputSize && valCorrect == valInputSize && testCorrect == testInputSize
          fprintf('Accuracy is 1 and all the cases are identified correct - Early stopping \n');
          subplot(2,1,1)
          plot(cost{1}); hold on; plot(cost{2});plot(cost{3});
          title('Cost plot'); xlabel('Number of epochs'); ylabel('cost');
          legend('Training cost','Validation cost', 'Testing cost');hold off;

          subplot(2,1,2);
          plot(acc{1});hold on;plot(acc{2});plot(acc{3});
          title('Accuracy plot'); xlabel('Number of epochs'); ylabel('Accuracy');
          legend('Training acc', 'Validation acc', 'Testing acc');hold off;
          break
      %Early stopping strategy - check if  Validation cost increases when the number of epochs is
greater than 65% of the epochs
      elseif epoch > round(numEpochs*0.65)
          if valCost(epoch) > valCost(epoch-1)
              subplot(2,1,1);
              fprintf('Validation cost increased after 65 percent of epochs were executed -- Early
stopping criteria \n');
              plot(cost{1}); hold on;plot(cost{2}); plot(cost{3});
              title('Cost plot'); xlabel('Number of epochs'); ylabel('cost');
              legend('Training cost','Validation cost', 'Testing cost');hold off;

              subplot(2,1,2);
              plot(acc{1}); hold on;plot(acc{2}); plot(acc{3});
              title('Accuracy plot'); xlabel('Number of epochs'); ylabel('Accuracy');
              legend('Training acc', 'Validation acc', 'Testing acc');hold off;
          break
          end
      else
```

```
          subplot(2,1,1);
          plot(cost{1});hold on;plot(cost{2}); plot(cost{3});
          title('Cost plot'); xlabel('Number of epochs'); ylabel('cost');
          legend('Training cost','Validation cost', 'Testing cost');hold off;

          subplot(2,1,2);
          plot(acc{1});hold on;plot(acc{2}); plot(acc{3});
          title('Accuracy plot'); xlabel('Number of epochs'); ylabel('Accuracy');
          legend('Training acc', 'Validation acc', 'Testing acc');hold off;
      end
  end
end
```

transfer.m

```
%Brunda Chouthoy
%CSC578 - Project 2
%Improving a Neural Network
%Oct 22, 2017

%Transfer functions
function f = transfer(z, fun)
    if (strcmp(fun, 'sigmoid'))
        f=logsig(z);
    elseif (strcmp(fun, 'tanh'))
        f=tanh(z);
    elseif (strcmp(fun, 'relu'))
        f=max(0,z);
    elseif (strcmp(fun, 'softmax'))
        f=softmax(z);
    end
end
```

dTransferPrime.m

```
%Brunda Chouthoy
%CSC578 - Project 2
%Improving a Neural Network
%Oct 22, 2017

%dTransfer function finds derivative of the activation function
function df = dTransferPrime(z, fun)
    if (strcmp(fun, 'sigmoid'))
        df=transfer(z,fun).*(1-transfer(z,fun));
    elseif (strcmp(fun, 'tanh'))
        df=1-transfer(z,fun).^2;
    elseif (strcmp(fun, 'relu'))
        df=double(z>0);
    elseif (strcmp(fun, 'softmax'))
        df=transfer(z,fun).*(1-transfer(z,fun));
    end
end
```

ANALYSIS:

The implemented network in this project works properly for all the datasets with changes in the number of hidden layers, number of epochs, learning rates, lambda regularization and momentum coefficients, different transfer functions (softmax, relu, tanh and sigmoid) and cost functions (quad, cross and log). The code was tested using 3 different datasets – iris.csv, MNIST and xor.csv for different configuration settings. The outputs and the graphs are displayed in the output section below. Second test case provided for the iris dataset with ‘relu’ transfer function didn’t seem to perform as expected – the accuracy values are very low.

Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network

IDEAS FOR ENHANCEMENT:

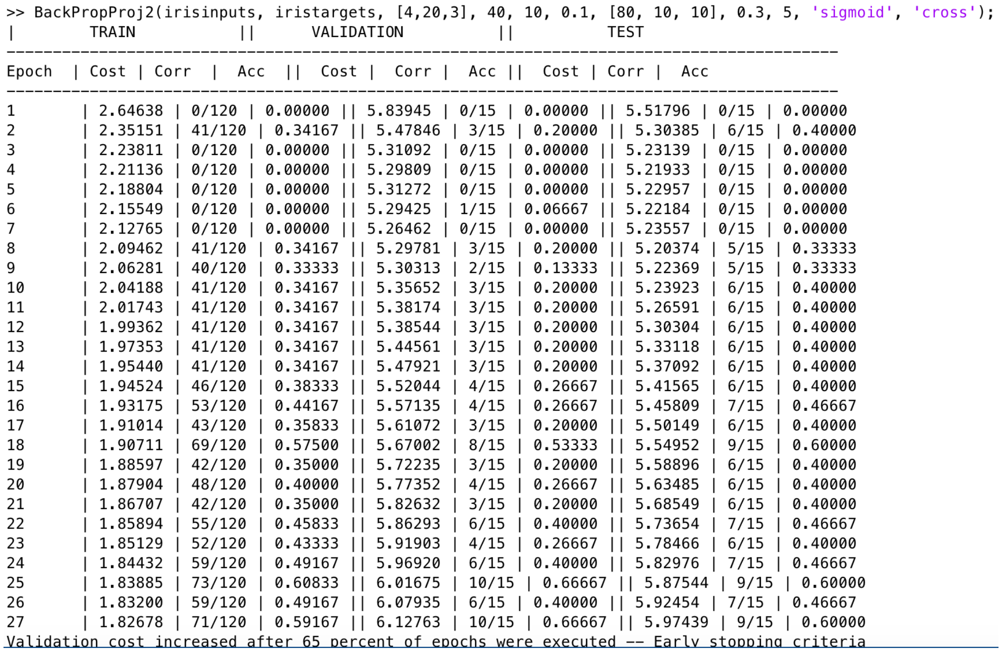
- Make this network more scalable by increasing the dataset size so that the network is exposed to different samples of data and will result is a better performance.
- Use a grid search method to choose the best configuration parameters for the network that results in better and higher performance.

OUTPUTS:

1)

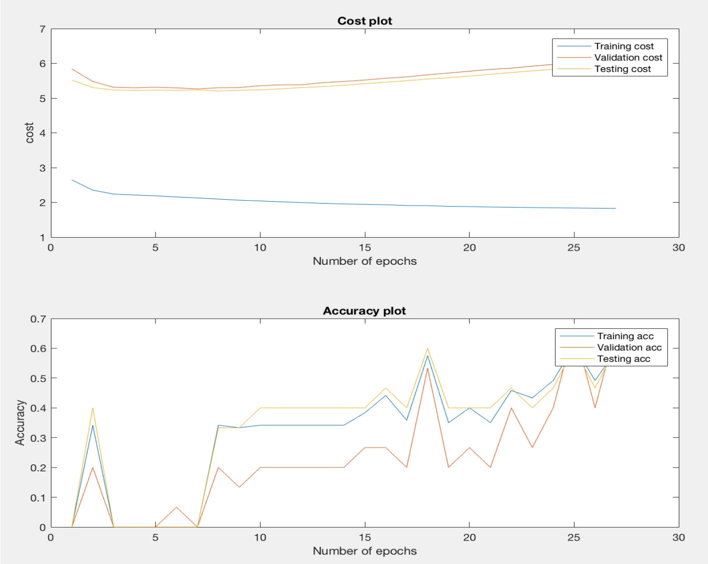
data set	epochs	hids	batch	eta	trans.	cost	mom.	reg.
iris.csv	40	20	10	0.1	sigmoid	cross^A	.3	5

>> iris = csvread('iris.csv');
>> irisinputs = iris(:, 1:4).';
>> iristargets = iris(:, 5:7).';
>> BackPropProj2(irisinputs, iristargets, [4,20,3], 40, 10, 0.1, [80, 10, 10], 0.3, 5, 'sigmoid', 'cross');



```
>> BackPropProj2(irisinputs, iristargets, [4,20,3], 40, 10, 0.1, [80, 10, 10], 0.3, 5, 'sigmoid', 'cross');
|        TRAIN           ||      VALIDATION          ||          TEST
-------------------------------------------------------------------------------------------
Epoch  | Cost | Corr  |  Acc  ||  Cost |  Corr |  Acc ||  Cost | Corr |  Acc
-------------------------------------------------------------------------------------------
1       | 2.64638 | 0/120 | 0.00000 || 5.83945 | 0/15 | 0.00000 || 5.51796 | 0/15 | 0.00000
2       | 2.35151 | 41/120 | 0.34167 || 5.47846 | 3/15 | 0.20000 || 5.30385 | 6/15 | 0.40000
3       | 2.23811 | 0/120 | 0.00000 || 5.31092 | 0/15 | 0.00000 || 5.23139 | 0/15 | 0.00000
4       | 2.21136 | 0/120 | 0.00000 || 5.29809 | 0/15 | 0.00000 || 5.21933 | 0/15 | 0.00000
5       | 2.18804 | 0/120 | 0.00000 || 5.31272 | 0/15 | 0.00000 || 5.22957 | 0/15 | 0.00000
6       | 2.15549 | 0/120 | 0.00000 || 5.29425 | 1/15 | 0.06667 || 5.22184 | 0/15 | 0.00000
7       | 2.12765 | 0/120 | 0.00000 || 5.26462 | 0/15 | 0.00000 || 5.23557 | 0/15 | 0.00000
8       | 2.09462 | 41/120 | 0.34167 || 5.29781 | 3/15 | 0.20000 || 5.20374 | 5/15 | 0.33333
9       | 2.06281 | 40/120 | 0.33333 || 5.30313 | 2/15 | 0.13333 || 5.22369 | 5/15 | 0.33333
10      | 2.04188 | 41/120 | 0.34167 || 5.35652 | 3/15 | 0.20000 || 5.23923 | 6/15 | 0.40000
11      | 2.01743 | 41/120 | 0.34167 || 5.38174 | 3/15 | 0.20000 || 5.26591 | 6/15 | 0.40000
12      | 1.99362 | 41/120 | 0.34167 || 5.38544 | 3/15 | 0.20000 || 5.30304 | 6/15 | 0.40000
13      | 1.97353 | 41/120 | 0.34167 || 5.44561 | 3/15 | 0.20000 || 5.33118 | 6/15 | 0.40000
14      | 1.95440 | 41/120 | 0.34167 || 5.47921 | 3/15 | 0.20000 || 5.37092 | 6/15 | 0.40000
15      | 1.94524 | 46/120 | 0.38333 || 5.52044 | 4/15 | 0.26667 || 5.41565 | 6/15 | 0.40000
16      | 1.93175 | 53/120 | 0.44167 || 5.57135 | 4/15 | 0.26667 || 5.45809 | 7/15 | 0.46667
17      | 1.91014 | 43/120 | 0.35833 || 5.61072 | 3/15 | 0.20000 || 5.50149 | 6/15 | 0.40000
18      | 1.90711 | 69/120 | 0.57500 || 5.67002 | 8/15 | 0.53333 || 5.54952 | 9/15 | 0.60000
19      | 1.88597 | 42/120 | 0.35000 || 5.72235 | 3/15 | 0.20000 || 5.58896 | 6/15 | 0.40000
20      | 1.87904 | 48/120 | 0.40000 || 5.77352 | 4/15 | 0.26667 || 5.63485 | 6/15 | 0.40000
21      | 1.86707 | 42/120 | 0.35000 || 5.82632 | 3/15 | 0.20000 || 5.68549 | 6/15 | 0.40000
22      | 1.85894 | 55/120 | 0.45833 || 5.86293 | 6/15 | 0.40000 || 5.73654 | 7/15 | 0.46667
23      | 1.85129 | 52/120 | 0.43333 || 5.91903 | 4/15 | 0.26667 || 5.78466 | 6/15 | 0.40000
24      | 1.84432 | 59/120 | 0.49167 || 5.96920 | 6/15 | 0.40000 || 5.82976 | 7/15 | 0.46667
25      | 1.83885 | 73/120 | 0.60833 || 6.01675 | 10/15 | 0.66667 || 5.87544 | 9/15 | 0.60000
26      | 1.83200 | 59/120 | 0.49167 || 6.07935 | 6/15 | 0.40000 || 5.92454 | 7/15 | 0.46667
27      | 1.82678 | 71/120 | 0.59167 || 6.12763 | 10/15 | 0.66667 || 5.97439 | 9/15 | 0.60000
Validation cost increased after 65 percent of epochs were executed -- Early stopping criteria
```

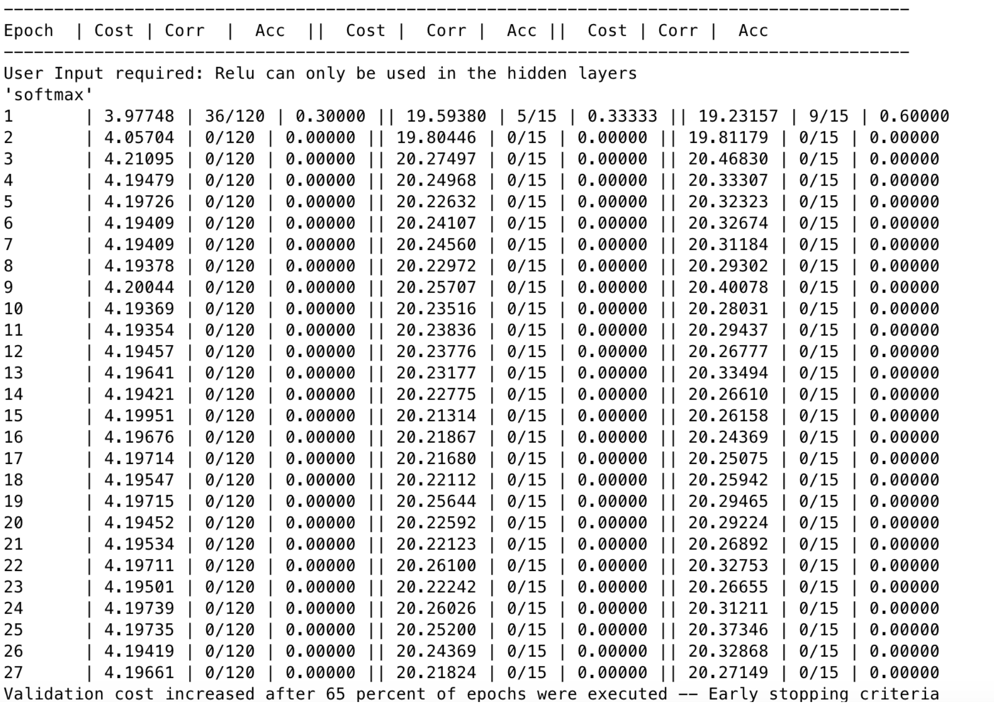
Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network



2)

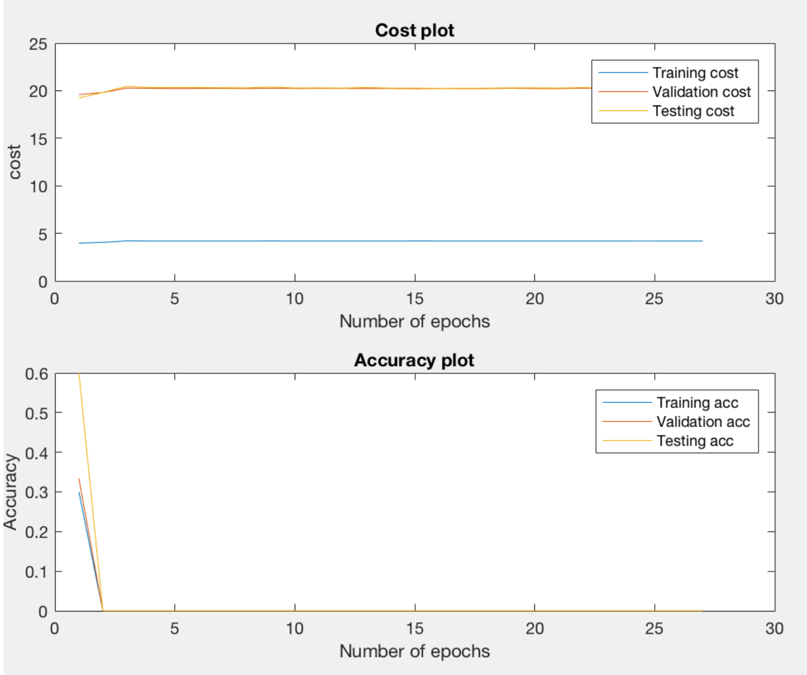
iris.csv	40	20	10	0.1	relu	cross	.3	5

>> BackPropProj2(irisinputs, iristargets, [4,20,3], 40, 10, 0.1, [80, 10, 10], 0.3, 5, 'relu', 'cross');



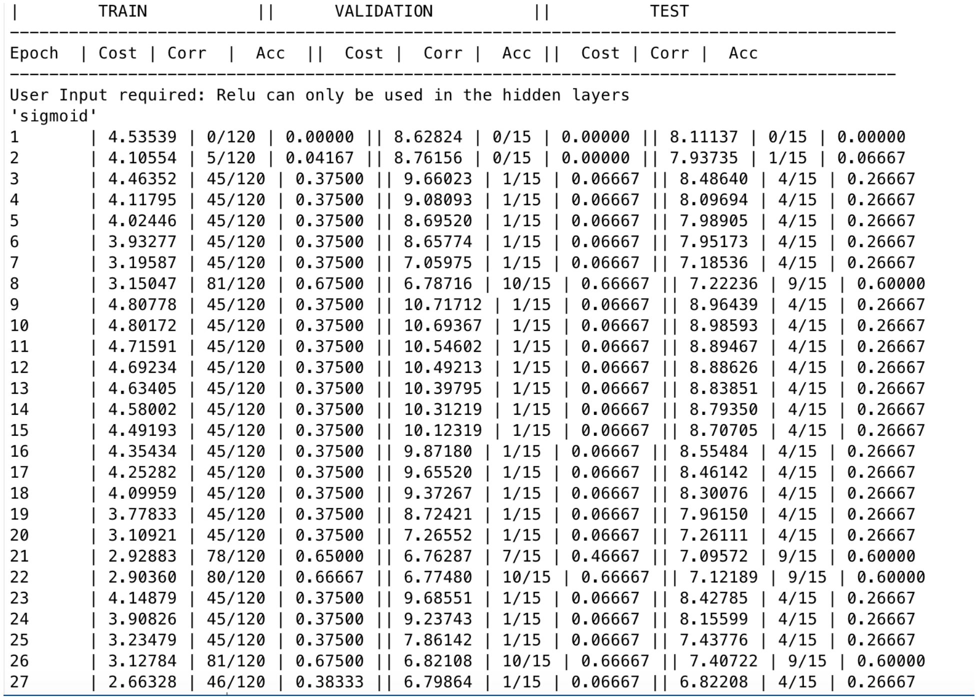
```
-----------------------------------------------------------------------------------------
Epoch  | Cost | Corr  |  Acc  ||  Cost |  Corr |  Acc ||  Cost | Corr |  Acc
-----------------------------------------------------------------------------------------
User Input required: Relu can only be used in the hidden layers
'softmax'
1       | 3.97748 | 36/120 | 0.30000 || 19.59380 | 5/15 | 0.33333 || 19.23157 | 9/15 | 0.60000
2       | 4.05704 | 0/120 | 0.00000 || 19.80446 | 0/15 | 0.00000 || 19.81179 | 0/15 | 0.00000
3       | 4.21095 | 0/120 | 0.00000 || 20.27497 | 0/15 | 0.00000 || 20.46830 | 0/15 | 0.00000
4       | 4.19479 | 0/120 | 0.00000 || 20.24968 | 0/15 | 0.00000 || 20.33307 | 0/15 | 0.00000
5       | 4.19726 | 0/120 | 0.00000 || 20.22632 | 0/15 | 0.00000 || 20.32323 | 0/15 | 0.00000
6       | 4.19409 | 0/120 | 0.00000 || 20.24107 | 0/15 | 0.00000 || 20.32674 | 0/15 | 0.00000
7       | 4.19409 | 0/120 | 0.00000 || 20.24560 | 0/15 | 0.00000 || 20.31184 | 0/15 | 0.00000
8       | 4.19378 | 0/120 | 0.00000 || 20.22972 | 0/15 | 0.00000 || 20.29302 | 0/15 | 0.00000
9       | 4.20044 | 0/120 | 0.00000 || 20.25707 | 0/15 | 0.00000 || 20.40078 | 0/15 | 0.00000
10      | 4.19369 | 0/120 | 0.00000 || 20.23516 | 0/15 | 0.00000 || 20.28031 | 0/15 | 0.00000
11      | 4.19354 | 0/120 | 0.00000 || 20.23836 | 0/15 | 0.00000 || 20.29437 | 0/15 | 0.00000
12      | 4.19457 | 0/120 | 0.00000 || 20.23776 | 0/15 | 0.00000 || 20.26777 | 0/15 | 0.00000
13      | 4.19641 | 0/120 | 0.00000 || 20.23177 | 0/15 | 0.00000 || 20.33494 | 0/15 | 0.00000
14      | 4.19421 | 0/120 | 0.00000 || 20.22775 | 0/15 | 0.00000 || 20.26610 | 0/15 | 0.00000
15      | 4.19951 | 0/120 | 0.00000 || 20.21314 | 0/15 | 0.00000 || 20.26158 | 0/15 | 0.00000
16      | 4.19676 | 0/120 | 0.00000 || 20.21867 | 0/15 | 0.00000 || 20.24369 | 0/15 | 0.00000
17      | 4.19714 | 0/120 | 0.00000 || 20.21680 | 0/15 | 0.00000 || 20.25075 | 0/15 | 0.00000
18      | 4.19547 | 0/120 | 0.00000 || 20.22112 | 0/15 | 0.00000 || 20.25942 | 0/15 | 0.00000
19      | 4.19715 | 0/120 | 0.00000 || 20.25644 | 0/15 | 0.00000 || 20.29465 | 0/15 | 0.00000
20      | 4.19452 | 0/120 | 0.00000 || 20.22592 | 0/15 | 0.00000 || 20.29224 | 0/15 | 0.00000
21      | 4.19534 | 0/120 | 0.00000 || 20.22123 | 0/15 | 0.00000 || 20.26892 | 0/15 | 0.00000
22      | 4.19711 | 0/120 | 0.00000 || 20.26100 | 0/15 | 0.00000 || 20.32753 | 0/15 | 0.00000
23      | 4.19501 | 0/120 | 0.00000 || 20.22242 | 0/15 | 0.00000 || 20.26655 | 0/15 | 0.00000
24      | 4.19739 | 0/120 | 0.00000 || 20.26026 | 0/15 | 0.00000 || 20.31211 | 0/15 | 0.00000
25      | 4.19735 | 0/120 | 0.00000 || 20.25200 | 0/15 | 0.00000 || 20.37346 | 0/15 | 0.00000
26      | 4.19419 | 0/120 | 0.00000 || 20.24369 | 0/15 | 0.00000 || 20.32868 | 0/15 | 0.00000
27      | 4.19661 | 0/120 | 0.00000 || 20.21824 | 0/15 | 0.00000 || 20.27149 | 0/15 | 0.00000
Validation cost increased after 65 percent of epochs were executed -- Early stopping criteria
```

Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network



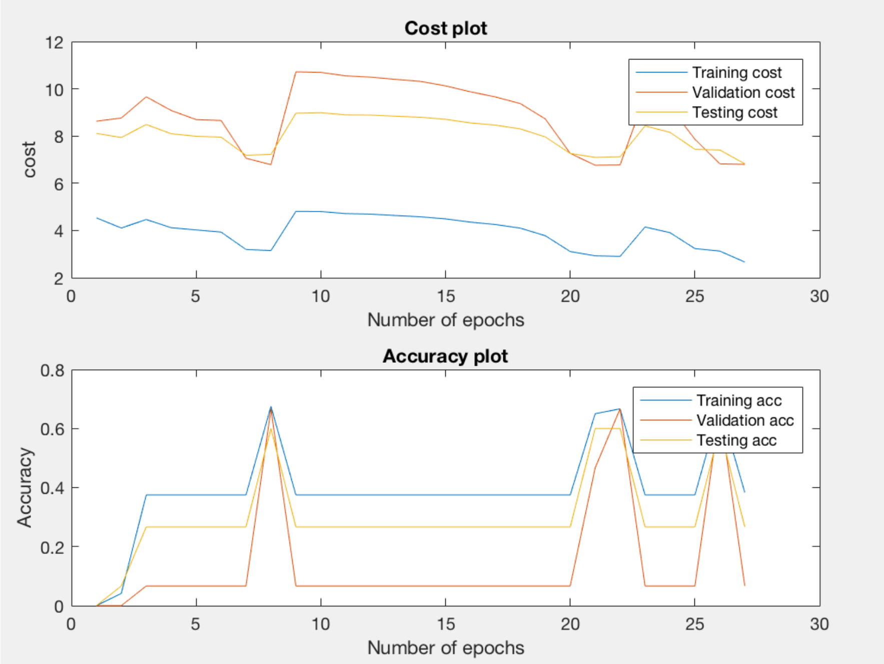
3)

iris.csv	40	20	10	0.1	relu	cross	0	5



```
|        TRAIN          ||      VALIDATION         ||          TEST
------------------------------------------------------------------------------------------
Epoch  | Cost | Corr  |  Acc  ||  Cost |  Corr |  Acc ||  Cost | Corr |  Acc
------------------------------------------------------------------------------------------
User Input required: Relu can only be used in the hidden layers
'sigmoid'
1       | 4.53539 | 0/120 | 0.00000 || 8.62824 | 0/15 | 0.00000 || 8.11137 | 0/15 | 0.00000
2       | 4.10554 | 5/120 | 0.04167 || 8.76156 | 0/15 | 0.00000 || 7.93735 | 1/15 | 0.06667
3       | 4.46352 | 45/120 | 0.37500 || 9.66023 | 1/15 | 0.06667 || 8.48640 | 4/15 | 0.26667
4       | 4.11795 | 45/120 | 0.37500 || 9.08093 | 1/15 | 0.06667 || 8.09694 | 4/15 | 0.26667
5       | 4.02446 | 45/120 | 0.37500 || 8.69520 | 1/15 | 0.06667 || 7.98905 | 4/15 | 0.26667
6       | 3.93277 | 45/120 | 0.37500 || 8.65774 | 1/15 | 0.06667 || 7.95173 | 4/15 | 0.26667
7       | 3.19587 | 45/120 | 0.37500 || 7.05975 | 1/15 | 0.06667 || 7.18536 | 4/15 | 0.26667
8       | 3.15047 | 81/120 | 0.67500 || 6.78716 | 10/15 | 0.66667 || 7.22236 | 9/15 | 0.60000
9       | 4.80778 | 45/120 | 0.37500 || 10.71712 | 1/15 | 0.06667 || 8.96439 | 4/15 | 0.26667
10      | 4.80172 | 45/120 | 0.37500 || 10.69367 | 1/15 | 0.06667 || 8.98593 | 4/15 | 0.26667
11      | 4.71591 | 45/120 | 0.37500 || 10.54602 | 1/15 | 0.06667 || 8.89467 | 4/15 | 0.26667
12      | 4.69234 | 45/120 | 0.37500 || 10.49213 | 1/15 | 0.06667 || 8.88626 | 4/15 | 0.26667
13      | 4.63405 | 45/120 | 0.37500 || 10.39795 | 1/15 | 0.06667 || 8.83851 | 4/15 | 0.26667
14      | 4.58002 | 45/120 | 0.37500 || 10.31219 | 1/15 | 0.06667 || 8.79350 | 4/15 | 0.26667
15      | 4.49193 | 45/120 | 0.37500 || 10.12319 | 1/15 | 0.06667 || 8.70705 | 4/15 | 0.26667
16      | 4.35434 | 45/120 | 0.37500 || 9.87180 | 1/15 | 0.06667 || 8.55484 | 4/15 | 0.26667
17      | 4.25282 | 45/120 | 0.37500 || 9.65520 | 1/15 | 0.06667 || 8.46142 | 4/15 | 0.26667
18      | 4.09959 | 45/120 | 0.37500 || 9.37267 | 1/15 | 0.06667 || 8.30076 | 4/15 | 0.26667
19      | 3.77833 | 45/120 | 0.37500 || 8.72421 | 1/15 | 0.06667 || 7.96150 | 4/15 | 0.26667
20      | 3.10921 | 45/120 | 0.37500 || 7.26552 | 1/15 | 0.06667 || 7.26111 | 4/15 | 0.26667
21      | 2.92883 | 78/120 | 0.65000 || 6.76287 | 7/15 | 0.46667 || 7.09572 | 9/15 | 0.60000
22      | 2.90360 | 80/120 | 0.66667 || 6.77480 | 10/15 | 0.66667 || 7.12189 | 9/15 | 0.60000
23      | 4.14879 | 45/120 | 0.37500 || 9.68551 | 1/15 | 0.06667 || 8.42785 | 4/15 | 0.26667
24      | 3.90826 | 45/120 | 0.37500 || 9.23743 | 1/15 | 0.06667 || 8.15599 | 4/15 | 0.26667
25      | 3.23479 | 45/120 | 0.37500 || 7.86142 | 1/15 | 0.06667 || 7.43776 | 4/15 | 0.26667
26      | 3.12784 | 81/120 | 0.67500 || 6.82108 | 10/15 | 0.66667 || 7.40722 | 9/15 | 0.60000
27      | 2.66328 | 46/120 | 0.38333 || 6.79864 | 1/15 | 0.06667 || 6.82208 | 4/15 | 0.26667
```

Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network

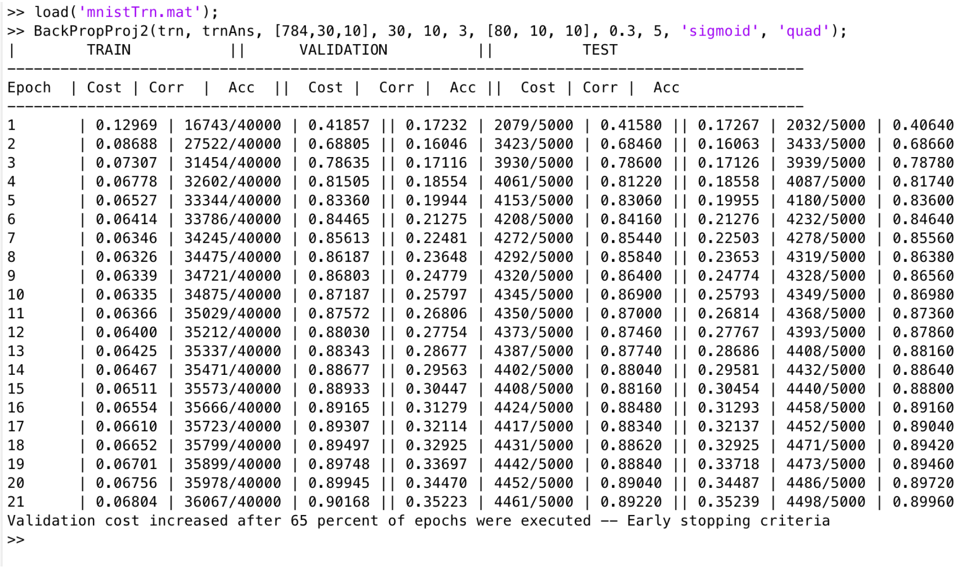


4)

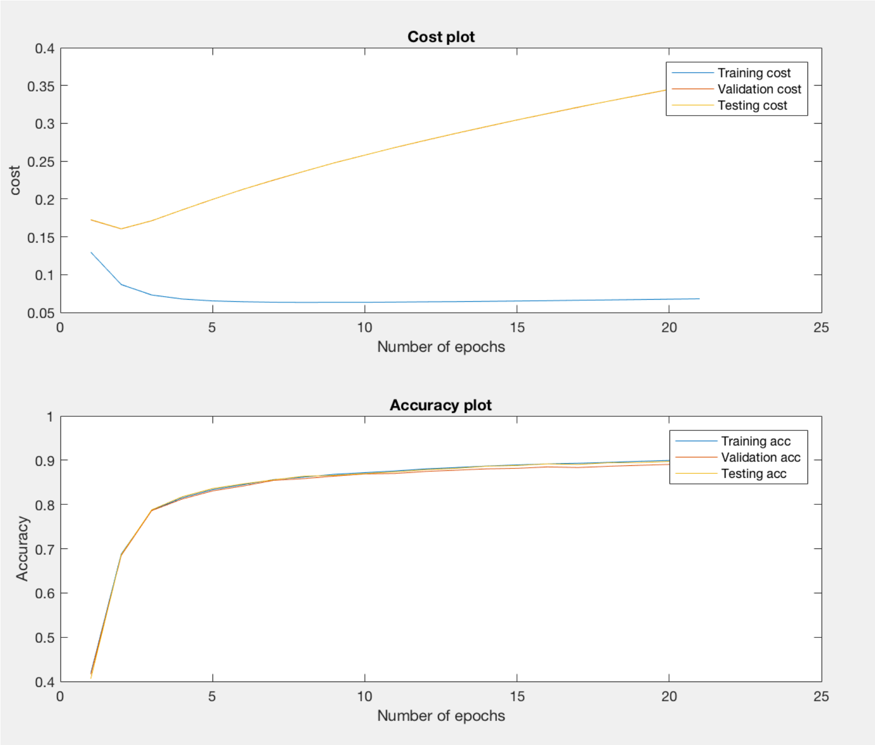
MNIST	30	30	10	3.0	sigmoid	quad	.3	5

```
>> load('mnistTrn.mat');
>> BackPropProj2(trn, trnAns, [784,30,10], 30, 10, 3, [80, 10, 10], 0.3, 5, 'sigmoid',
'quad');
```

Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network



```
>> load('mnistTrn.mat');
>> BackPropProj2(trn, trnAns, [784,30,10], 30, 10, 3, [80, 10, 10], 0.3, 5, 'sigmoid', 'quad');
|        TRAIN           ||      VALIDATION          ||          TEST
------------------------------------------------------------------------------------------
Epoch   | Cost | Corr   |  Acc   ||  Cost |  Corr |  Acc ||  Cost | Corr |  Acc
------------------------------------------------------------------------------------------
1        | 0.12969 | 16743/40000 | 0.41857 || 0.17232 | 2079/5000 | 0.41580 || 0.17267 | 2032/5000 | 0.40640
2        | 0.08688 | 27522/40000 | 0.68805 || 0.16046 | 3423/5000 | 0.68460 || 0.16063 | 3433/5000 | 0.68660
3        | 0.07307 | 31454/40000 | 0.78635 || 0.17116 | 3930/5000 | 0.78600 || 0.17126 | 3939/5000 | 0.78780
4        | 0.06778 | 32602/40000 | 0.81505 || 0.18554 | 4061/5000 | 0.81220 || 0.18558 | 4087/5000 | 0.81740
5        | 0.06527 | 33344/40000 | 0.83360 || 0.19944 | 4153/5000 | 0.83060 || 0.19955 | 4180/5000 | 0.83600
6        | 0.06414 | 33786/40000 | 0.84465 || 0.21275 | 4208/5000 | 0.84160 || 0.21276 | 4232/5000 | 0.84640
7        | 0.06346 | 34245/40000 | 0.85613 || 0.22481 | 4272/5000 | 0.85440 || 0.22503 | 4278/5000 | 0.85560
8        | 0.06326 | 34475/40000 | 0.86187 || 0.23648 | 4292/5000 | 0.85840 || 0.23653 | 4319/5000 | 0.86380
9        | 0.06339 | 34721/40000 | 0.86803 || 0.24779 | 4320/5000 | 0.86400 || 0.24774 | 4328/5000 | 0.86560
10       | 0.06335 | 34875/40000 | 0.87187 || 0.25797 | 4345/5000 | 0.86900 || 0.25793 | 4349/5000 | 0.86980
11       | 0.06366 | 35029/40000 | 0.87572 || 0.26806 | 4350/5000 | 0.87000 || 0.26814 | 4368/5000 | 0.87360
12       | 0.06400 | 35212/40000 | 0.88030 || 0.27754 | 4373/5000 | 0.87460 || 0.27767 | 4393/5000 | 0.87860
13       | 0.06425 | 35337/40000 | 0.88343 || 0.28677 | 4387/5000 | 0.87740 || 0.28686 | 4408/5000 | 0.88160
14       | 0.06467 | 35471/40000 | 0.88677 || 0.29563 | 4402/5000 | 0.88040 || 0.29581 | 4432/5000 | 0.88640
15       | 0.06511 | 35573/40000 | 0.88933 || 0.30447 | 4408/5000 | 0.88160 || 0.30454 | 4440/5000 | 0.88800
16       | 0.06554 | 35666/40000 | 0.89165 || 0.31279 | 4424/5000 | 0.88480 || 0.31293 | 4458/5000 | 0.89160
17       | 0.06610 | 35723/40000 | 0.89307 || 0.32114 | 4417/5000 | 0.88340 || 0.32137 | 4452/5000 | 0.89040
18       | 0.06652 | 35799/40000 | 0.89497 || 0.32925 | 4431/5000 | 0.88620 || 0.32925 | 4471/5000 | 0.89420
19       | 0.06701 | 35899/40000 | 0.89748 || 0.33697 | 4442/5000 | 0.88840 || 0.33718 | 4473/5000 | 0.89460
20       | 0.06756 | 35978/40000 | 0.89945 || 0.34470 | 4452/5000 | 0.89040 || 0.34487 | 4486/5000 | 0.89720
21       | 0.06804 | 36067/40000 | 0.90168 || 0.35223 | 4461/5000 | 0.89220 || 0.35239 | 4498/5000 | 0.89960
Validation cost increased after 65 percent of epochs were executed -- Early stopping criteria
>>
```

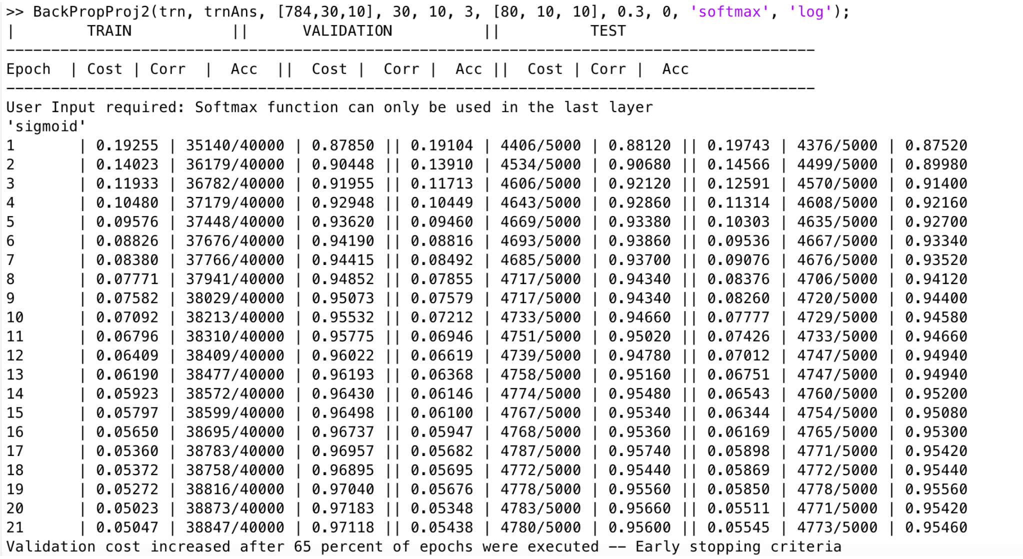


5)

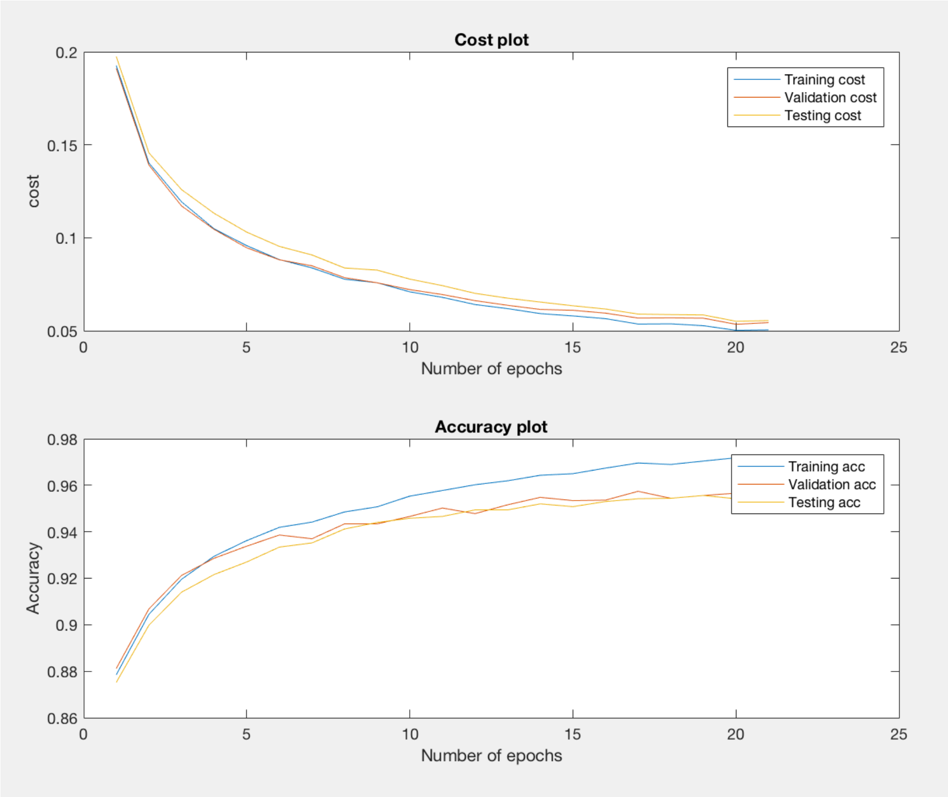
MNIST	30	30	10	3.0	softmax[B]	log	.3	0

BackPropProj2(trn, trnAns, [784,30,10], 30, 10, 3, [80, 10, 10], 0.3, 0, 'softmax', 'log');

Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network



```
>> BackPropProj2(trn, trnAns, [784,30,10], 30, 10, 3, [80, 10, 10], 0.3, 0, 'softmax', 'log');
|        TRAIN           ||      VALIDATION          ||          TEST
---------------------------------------------------------------------------------------
Epoch  | Cost | Corr  |  Acc  ||  Cost |  Corr |  Acc ||  Cost | Corr |  Acc
---------------------------------------------------------------------------------------
User Input required: Softmax function can only be used in the last layer
'sigmoid'
1       | 0.19255 | 35140/40000 | 0.87850 || 0.19104 | 4406/5000 | 0.88120 || 0.19743 | 4376/5000 | 0.87520
2       | 0.14023 | 36179/40000 | 0.90448 || 0.13910 | 4534/5000 | 0.90680 || 0.14566 | 4499/5000 | 0.89980
3       | 0.11933 | 36782/40000 | 0.91955 || 0.11713 | 4606/5000 | 0.92120 || 0.12591 | 4570/5000 | 0.91400
4       | 0.10480 | 37179/40000 | 0.92948 || 0.10449 | 4643/5000 | 0.92860 || 0.11314 | 4608/5000 | 0.92160
5       | 0.09576 | 37448/40000 | 0.93620 || 0.09460 | 4669/5000 | 0.93380 || 0.10303 | 4635/5000 | 0.92700
6       | 0.08826 | 37676/40000 | 0.94190 || 0.08816 | 4693/5000 | 0.93860 || 0.09536 | 4667/5000 | 0.93340
7       | 0.08380 | 37766/40000 | 0.94415 || 0.08492 | 4685/5000 | 0.93700 || 0.09076 | 4676/5000 | 0.93520
8       | 0.07771 | 37941/40000 | 0.94852 || 0.07855 | 4717/5000 | 0.94340 || 0.08376 | 4706/5000 | 0.94120
9       | 0.07582 | 38029/40000 | 0.95073 || 0.07579 | 4717/5000 | 0.94340 || 0.08260 | 4720/5000 | 0.94400
10      | 0.07092 | 38213/40000 | 0.95532 || 0.07212 | 4733/5000 | 0.94660 || 0.07777 | 4729/5000 | 0.94580
11      | 0.06796 | 38310/40000 | 0.95775 || 0.06946 | 4751/5000 | 0.95020 || 0.07426 | 4733/5000 | 0.94660
12      | 0.06409 | 38409/40000 | 0.96022 || 0.06619 | 4739/5000 | 0.94780 || 0.07012 | 4747/5000 | 0.94940
13      | 0.06190 | 38477/40000 | 0.96193 || 0.06368 | 4758/5000 | 0.95160 || 0.06751 | 4747/5000 | 0.94940
14      | 0.05923 | 38572/40000 | 0.96430 || 0.06146 | 4774/5000 | 0.95480 || 0.06543 | 4760/5000 | 0.95200
15      | 0.05797 | 38599/40000 | 0.96498 || 0.06100 | 4767/5000 | 0.95340 || 0.06344 | 4754/5000 | 0.95080
16      | 0.05650 | 38695/40000 | 0.96737 || 0.05947 | 4768/5000 | 0.95360 || 0.06169 | 4765/5000 | 0.95300
17      | 0.05360 | 38783/40000 | 0.96957 || 0.05682 | 4787/5000 | 0.95740 || 0.05898 | 4771/5000 | 0.95420
18      | 0.05372 | 38758/40000 | 0.96895 || 0.05695 | 4772/5000 | 0.95440 || 0.05869 | 4772/5000 | 0.95440
19      | 0.05272 | 38816/40000 | 0.97040 || 0.05676 | 4778/5000 | 0.95560 || 0.05850 | 4778/5000 | 0.95560
20      | 0.05023 | 38873/40000 | 0.97183 || 0.05348 | 4783/5000 | 0.95660 || 0.05511 | 4771/5000 | 0.95420
21      | 0.05047 | 38847/40000 | 0.97118 || 0.05438 | 4780/5000 | 0.95600 || 0.05545 | 4773/5000 | 0.95460
Validation cost increased after 65 percent of epochs were executed -- Early stopping criteria
```

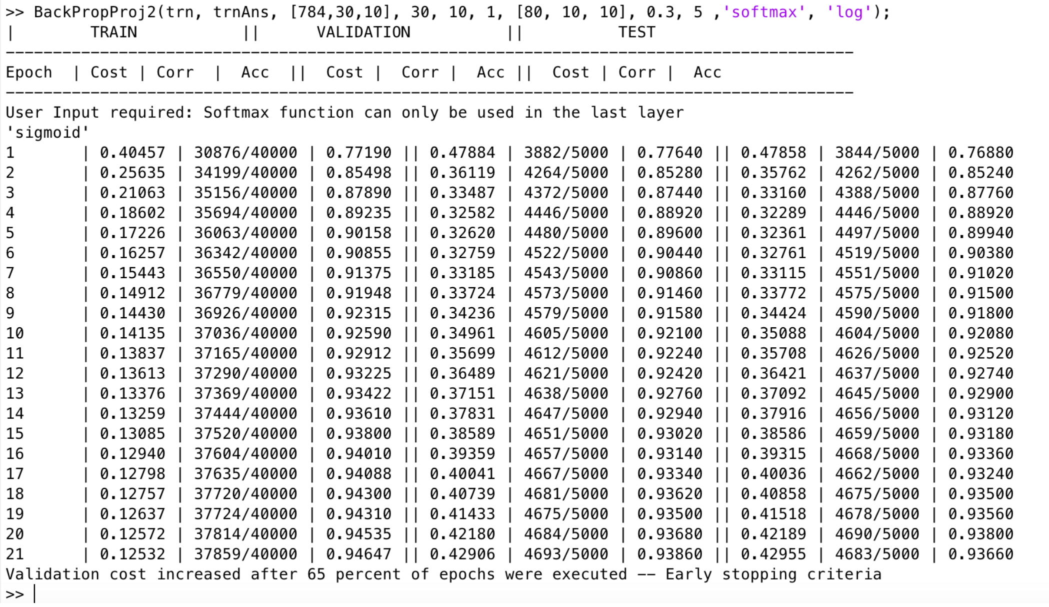


6)

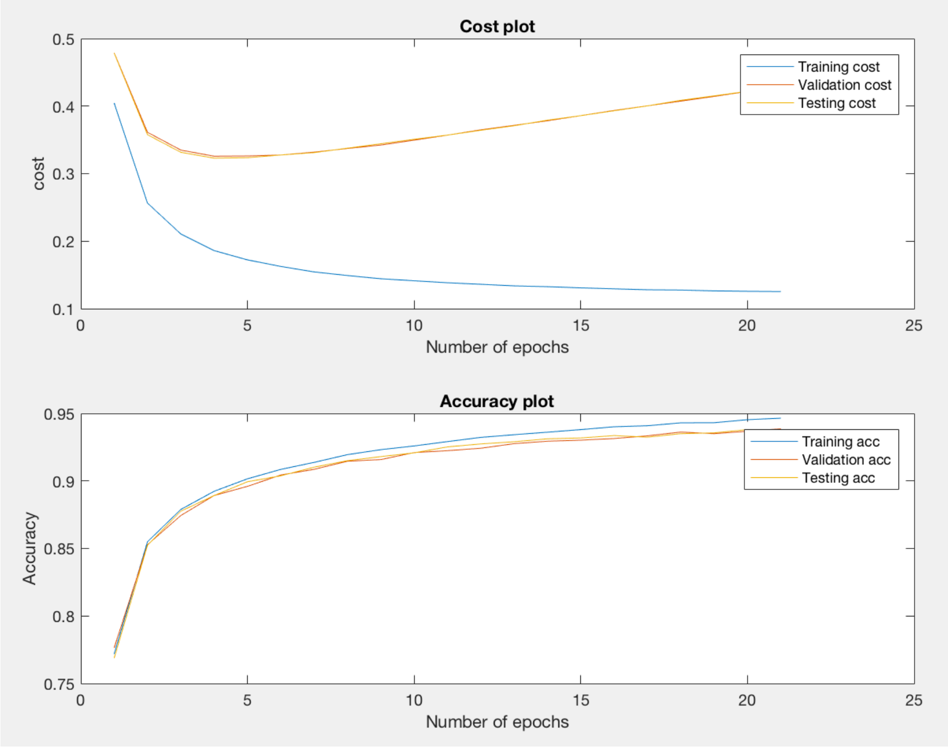
MNIST	30	30	10	1.0	softmax^B	log	.3	5

Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network

>> BackPropProj2(trn, trnAns, [784,30,10], 30, 10, 1, [80, 10, 10], 0.3, 5, 'softmax', 'log');



```
>> BackPropProj2(trn, trnAns, [784,30,10], 30, 10, 1, [80, 10, 10], 0.3, 5 ,'softmax', 'log');
|        TRAIN            ||      VALIDATION          ||           TEST
---------------------------------------------------------------------------------------------
Epoch  | Cost | Corr  |  Acc  ||  Cost |  Corr |  Acc ||  Cost | Corr |  Acc
---------------------------------------------------------------------------------------------
User Input required: Softmax function can only be used in the last layer
'sigmoid'
1       | 0.40457 | 30876/40000 | 0.77190 || 0.47884 | 3882/5000 | 0.77640 || 0.47858 | 3844/5000 | 0.76880
2       | 0.25635 | 34199/40000 | 0.85498 || 0.36119 | 4264/5000 | 0.85280 || 0.35762 | 4262/5000 | 0.85240
3       | 0.21063 | 35156/40000 | 0.87890 || 0.33487 | 4372/5000 | 0.87440 || 0.33160 | 4388/5000 | 0.87760
4       | 0.18602 | 35694/40000 | 0.89235 || 0.32582 | 4446/5000 | 0.88920 || 0.32289 | 4446/5000 | 0.88920
5       | 0.17226 | 36063/40000 | 0.90158 || 0.32620 | 4480/5000 | 0.89600 || 0.32361 | 4497/5000 | 0.89940
6       | 0.16257 | 36342/40000 | 0.90855 || 0.32759 | 4522/5000 | 0.90440 || 0.32761 | 4519/5000 | 0.90380
7       | 0.15443 | 36550/40000 | 0.91375 || 0.33185 | 4543/5000 | 0.90860 || 0.33115 | 4551/5000 | 0.91020
8       | 0.14912 | 36779/40000 | 0.91948 || 0.33724 | 4573/5000 | 0.91460 || 0.33772 | 4575/5000 | 0.91500
9       | 0.14430 | 36926/40000 | 0.92315 || 0.34236 | 4579/5000 | 0.91580 || 0.34424 | 4590/5000 | 0.91800
10      | 0.14135 | 37036/40000 | 0.92590 || 0.34961 | 4605/5000 | 0.92100 || 0.35088 | 4604/5000 | 0.92080
11      | 0.13837 | 37165/40000 | 0.92912 || 0.35699 | 4612/5000 | 0.92240 || 0.35708 | 4626/5000 | 0.92520
12      | 0.13613 | 37290/40000 | 0.93225 || 0.36489 | 4621/5000 | 0.92420 || 0.36421 | 4637/5000 | 0.92740
13      | 0.13376 | 37369/40000 | 0.93422 || 0.37151 | 4638/5000 | 0.92760 || 0.37092 | 4645/5000 | 0.92900
14      | 0.13259 | 37444/40000 | 0.93610 || 0.37831 | 4647/5000 | 0.92940 || 0.37916 | 4656/5000 | 0.93120
15      | 0.13085 | 37520/40000 | 0.93800 || 0.38589 | 4651/5000 | 0.93020 || 0.38586 | 4659/5000 | 0.93180
16      | 0.12940 | 37604/40000 | 0.94010 || 0.39359 | 4657/5000 | 0.93140 || 0.39315 | 4668/5000 | 0.93360
17      | 0.12798 | 37635/40000 | 0.94088 || 0.40041 | 4667/5000 | 0.93340 || 0.40036 | 4662/5000 | 0.93240
18      | 0.12757 | 37720/40000 | 0.94300 || 0.40739 | 4681/5000 | 0.93620 || 0.40858 | 4675/5000 | 0.93500
19      | 0.12637 | 37724/40000 | 0.94310 || 0.41433 | 4675/5000 | 0.93500 || 0.41518 | 4678/5000 | 0.93560
20      | 0.12572 | 37814/40000 | 0.94535 || 0.42180 | 4684/5000 | 0.93680 || 0.42189 | 4690/5000 | 0.93800
21      | 0.12532 | 37859/40000 | 0.94647 || 0.42906 | 4693/5000 | 0.93860 || 0.42955 | 4683/5000 | 0.93660
Validation cost increased after 65 percent of epochs were executed -- Early stopping criteria
>> |
```

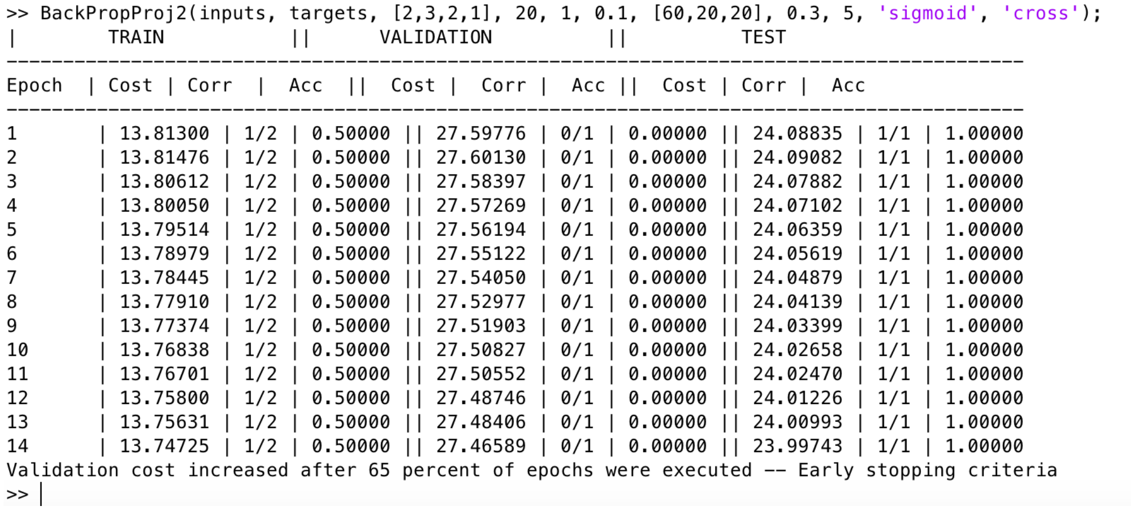


7)

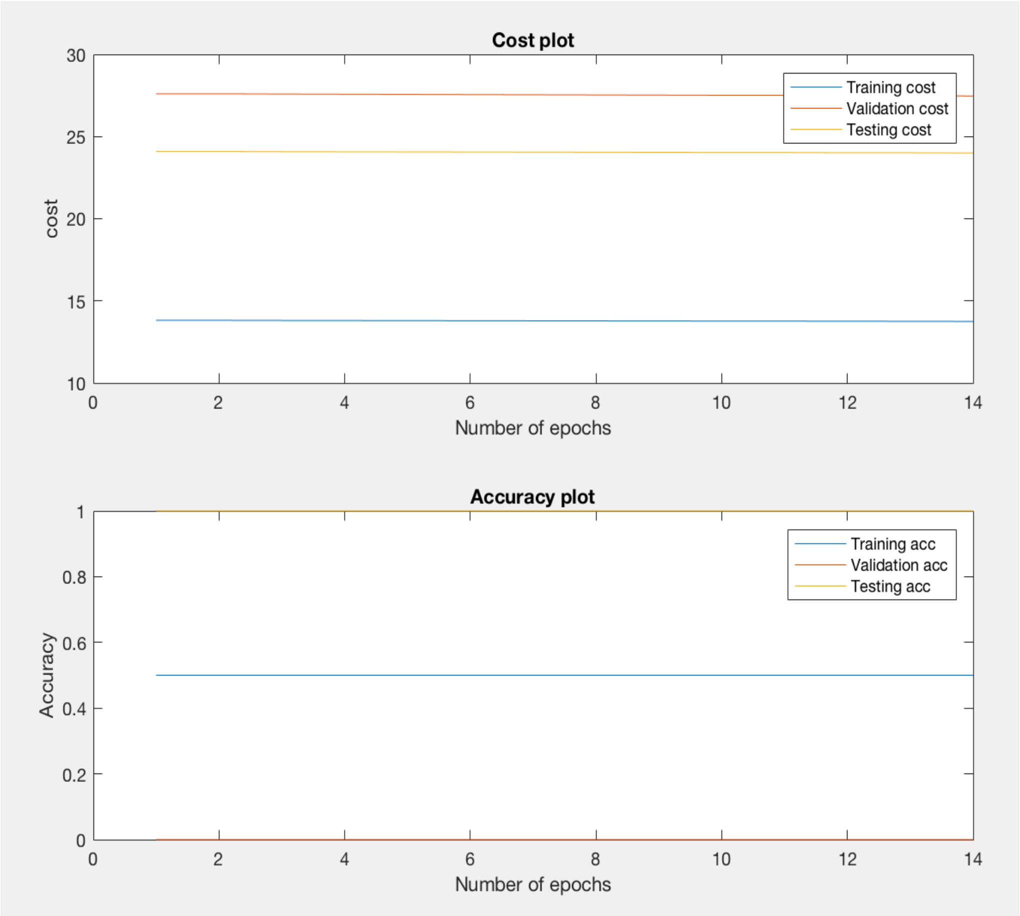
xor.csv	20	[3 2]	1	0.1	sigmoid	cross	.3	5

Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network

BackPropProj2(inputs, targets, [2,3,2,1], 20, 1, 0.1, [60,20,20], 0.3, 5, 'sigmoid', 'cross');



```
>> BackPropProj2(inputs, targets, [2,3,2,1], 20, 1, 0.1, [60,20,20], 0.3, 5, 'sigmoid', 'cross');
|        TRAIN           ||      VALIDATION         ||          TEST
-------------------------------------------------------------------------------------------
Epoch   | Cost | Corr   |   Acc   ||  Cost |   Corr |   Acc ||  Cost | Corr |   Acc
-------------------------------------------------------------------------------------------
1        | 13.81300 | 1/2 | 0.50000 || 27.59776 | 0/1 | 0.00000 || 24.08835 | 1/1 | 1.00000
2        | 13.81476 | 1/2 | 0.50000 || 27.60130 | 0/1 | 0.00000 || 24.09082 | 1/1 | 1.00000
3        | 13.80612 | 1/2 | 0.50000 || 27.58397 | 0/1 | 0.00000 || 24.07882 | 1/1 | 1.00000
4        | 13.80050 | 1/2 | 0.50000 || 27.57269 | 0/1 | 0.00000 || 24.07102 | 1/1 | 1.00000
5        | 13.79514 | 1/2 | 0.50000 || 27.56194 | 0/1 | 0.00000 || 24.06359 | 1/1 | 1.00000
6        | 13.78979 | 1/2 | 0.50000 || 27.55122 | 0/1 | 0.00000 || 24.05619 | 1/1 | 1.00000
7        | 13.78445 | 1/2 | 0.50000 || 27.54050 | 0/1 | 0.00000 || 24.04879 | 1/1 | 1.00000
8        | 13.77910 | 1/2 | 0.50000 || 27.52977 | 0/1 | 0.00000 || 24.04139 | 1/1 | 1.00000
9        | 13.77374 | 1/2 | 0.50000 || 27.51903 | 0/1 | 0.00000 || 24.03399 | 1/1 | 1.00000
10       | 13.76838 | 1/2 | 0.50000 || 27.50827 | 0/1 | 0.00000 || 24.02658 | 1/1 | 1.00000
11       | 13.76701 | 1/2 | 0.50000 || 27.50552 | 0/1 | 0.00000 || 24.02470 | 1/1 | 1.00000
12       | 13.75800 | 1/2 | 0.50000 || 27.48746 | 0/1 | 0.00000 || 24.01226 | 1/1 | 1.00000
13       | 13.75631 | 1/2 | 0.50000 || 27.48406 | 0/1 | 0.00000 || 24.00993 | 1/1 | 1.00000
14       | 13.74725 | 1/2 | 0.50000 || 27.46589 | 0/1 | 0.00000 || 23.99743 | 1/1 | 1.00000
Validation cost increased after 65 percent of epochs were executed -- Early stopping criteria
>> |
```

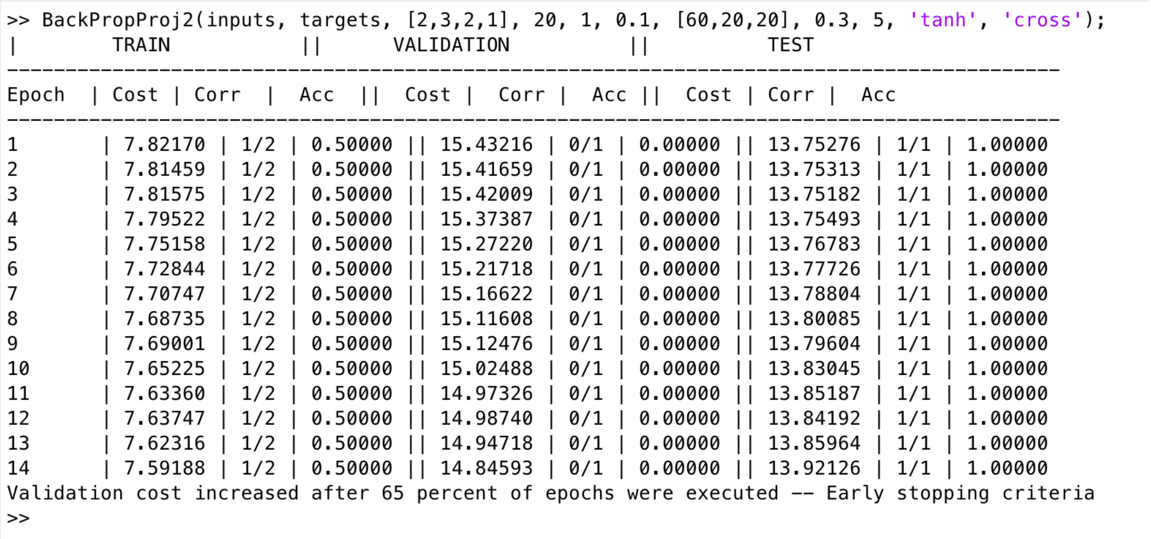


8)

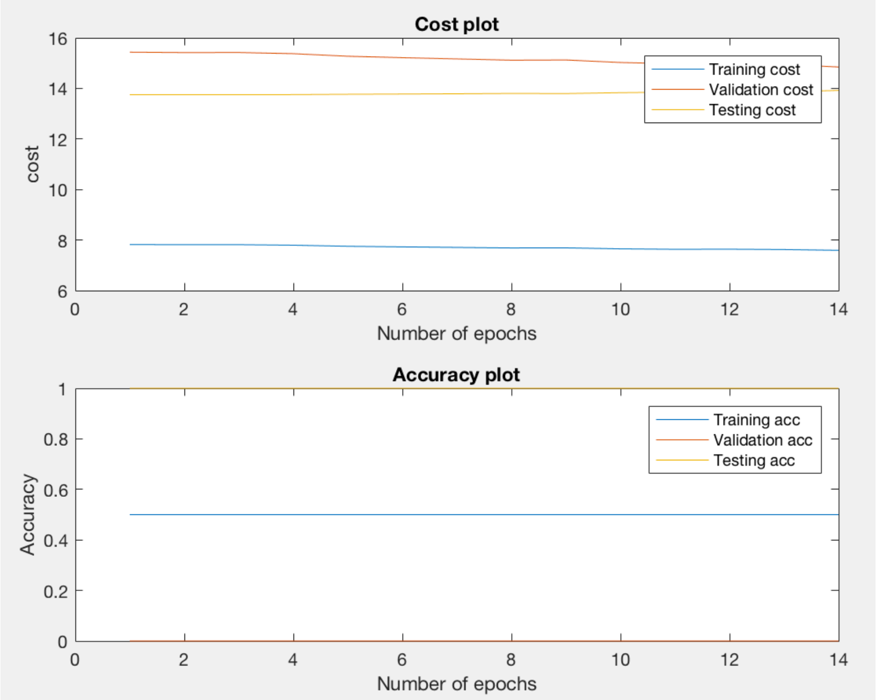
Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network

xor.csv	20	[3 2]	1	0.1	tanh^{C}	cross	.3	5

>> BackPropProj2(inputs, targets, [2,3,2,1], 20, 1, 0.1, [60,20,20], 0.3, 5, 'tanh', 'cross');



```
>> BackPropProj2(inputs, targets, [2,3,2,1], 20, 1, 0.1, [60,20,20], 0.3, 5, 'tanh', 'cross');
|        TRAIN            ||      VALIDATION          ||          TEST
------------------------------------------------------------------------------------------
Epoch   | Cost | Corr  |  Acc  ||  Cost |  Corr |  Acc ||  Cost | Corr |  Acc
------------------------------------------------------------------------------------------
1       | 7.82170 | 1/2 | 0.50000 || 15.43216 | 0/1 | 0.00000 || 13.75276 | 1/1 | 1.00000
2       | 7.81459 | 1/2 | 0.50000 || 15.41659 | 0/1 | 0.00000 || 13.75313 | 1/1 | 1.00000
3       | 7.81575 | 1/2 | 0.50000 || 15.42009 | 0/1 | 0.00000 || 13.75182 | 1/1 | 1.00000
4       | 7.79522 | 1/2 | 0.50000 || 15.37387 | 0/1 | 0.00000 || 13.75493 | 1/1 | 1.00000
5       | 7.75158 | 1/2 | 0.50000 || 15.27220 | 0/1 | 0.00000 || 13.76783 | 1/1 | 1.00000
6       | 7.72844 | 1/2 | 0.50000 || 15.21718 | 0/1 | 0.00000 || 13.77726 | 1/1 | 1.00000
7       | 7.70747 | 1/2 | 0.50000 || 15.16622 | 0/1 | 0.00000 || 13.78804 | 1/1 | 1.00000
8       | 7.68735 | 1/2 | 0.50000 || 15.11608 | 0/1 | 0.00000 || 13.80085 | 1/1 | 1.00000
9       | 7.69001 | 1/2 | 0.50000 || 15.12476 | 0/1 | 0.00000 || 13.79604 | 1/1 | 1.00000
10      | 7.65225 | 1/2 | 0.50000 || 15.02488 | 0/1 | 0.00000 || 13.83045 | 1/1 | 1.00000
11      | 7.63360 | 1/2 | 0.50000 || 14.97326 | 0/1 | 0.00000 || 13.85187 | 1/1 | 1.00000
12      | 7.63747 | 1/2 | 0.50000 || 14.98740 | 0/1 | 0.00000 || 13.84192 | 1/1 | 1.00000
13      | 7.62316 | 1/2 | 0.50000 || 14.94718 | 0/1 | 0.00000 || 13.85964 | 1/1 | 1.00000
14      | 7.59188 | 1/2 | 0.50000 || 14.84593 | 0/1 | 0.00000 || 13.92126 | 1/1 | 1.00000
Validation cost increased after 65 percent of epochs were executed -- Early stopping criteria
>>
```

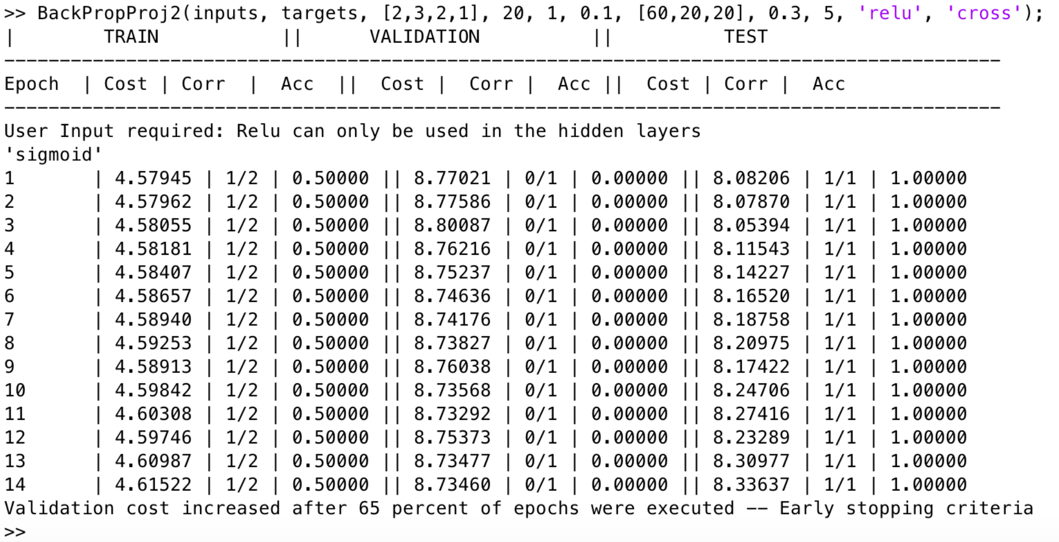


Brunda Chouthoy
CSC 578: Project 2
Improving a Neural Network

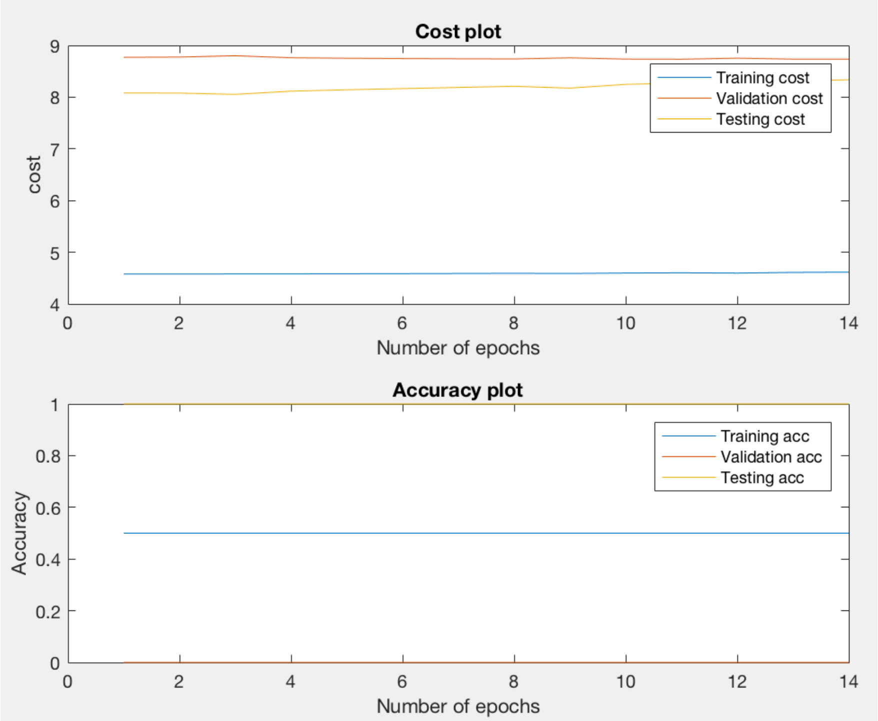
9)

xor.csv	20	[3 2]	1	0.1	relu	cross	.3	5

>> BackPropProj2(inputs, targets, [2,3,2,1], 20, 1, 0.1, [60,20,20], 0.3, 5, 'relu', 'cross');



```
>> BackPropProj2(inputs, targets, [2,3,2,1], 20, 1, 0.1, [60,20,20], 0.3, 5, 'relu', 'cross');
|        TRAIN            ||      VALIDATION         ||          TEST
-----------------------------------------------------------------------------------------
Epoch   | Cost | Corr   |  Acc   ||  Cost |  Corr |  Acc ||  Cost | Corr |  Acc
-----------------------------------------------------------------------------------------
User Input required: Relu can only be used in the hidden layers
'sigmoid'
1       | 4.57945 | 1/2 | 0.50000 || 8.77021 | 0/1 | 0.00000 || 8.08206 | 1/1 | 1.00000
2       | 4.57962 | 1/2 | 0.50000 || 8.77586 | 0/1 | 0.00000 || 8.07870 | 1/1 | 1.00000
3       | 4.58055 | 1/2 | 0.50000 || 8.80087 | 0/1 | 0.00000 || 8.05394 | 1/1 | 1.00000
4       | 4.58181 | 1/2 | 0.50000 || 8.76216 | 0/1 | 0.00000 || 8.11543 | 1/1 | 1.00000
5       | 4.58407 | 1/2 | 0.50000 || 8.75237 | 0/1 | 0.00000 || 8.14227 | 1/1 | 1.00000
6       | 4.58657 | 1/2 | 0.50000 || 8.74636 | 0/1 | 0.00000 || 8.16520 | 1/1 | 1.00000
7       | 4.58940 | 1/2 | 0.50000 || 8.74176 | 0/1 | 0.00000 || 8.18758 | 1/1 | 1.00000
8       | 4.59253 | 1/2 | 0.50000 || 8.73827 | 0/1 | 0.00000 || 8.20975 | 1/1 | 1.00000
9       | 4.58913 | 1/2 | 0.50000 || 8.76038 | 0/1 | 0.00000 || 8.17422 | 1/1 | 1.00000
10      | 4.59842 | 1/2 | 0.50000 || 8.73568 | 0/1 | 0.00000 || 8.24706 | 1/1 | 1.00000
11      | 4.60308 | 1/2 | 0.50000 || 8.73292 | 0/1 | 0.00000 || 8.27416 | 1/1 | 1.00000
12      | 4.59746 | 1/2 | 0.50000 || 8.75373 | 0/1 | 0.00000 || 8.23289 | 1/1 | 1.00000
13      | 4.60987 | 1/2 | 0.50000 || 8.73477 | 0/1 | 0.00000 || 8.30977 | 1/1 | 1.00000
14      | 4.61522 | 1/2 | 0.50000 || 8.73460 | 0/1 | 0.00000 || 8.33637 | 1/1 | 1.00000
Validation cost increased after 65 percent of epochs were executed -- Early stopping criteria
>>
```


Cost plot
cost
Number of epochs
Training cost
Validation cost
Testing cost
Accuracy plot
Accuracy
Number of epochs
Training acc
Validation acc
Testing acc