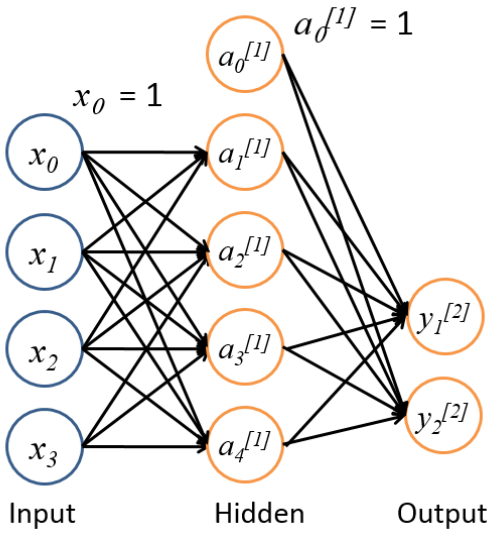
ROCHESTER INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING
CMPE 677 Machine Intelligence
HW #7

Name:____Kaiwen Zheng_____

Due 11:55pm, Fri, 11/10/2017, submit via Dropbox. Work alone. All questions pertain to Matlab/Octav.

1. (8 pts) Use the below figure to answer the following questions.

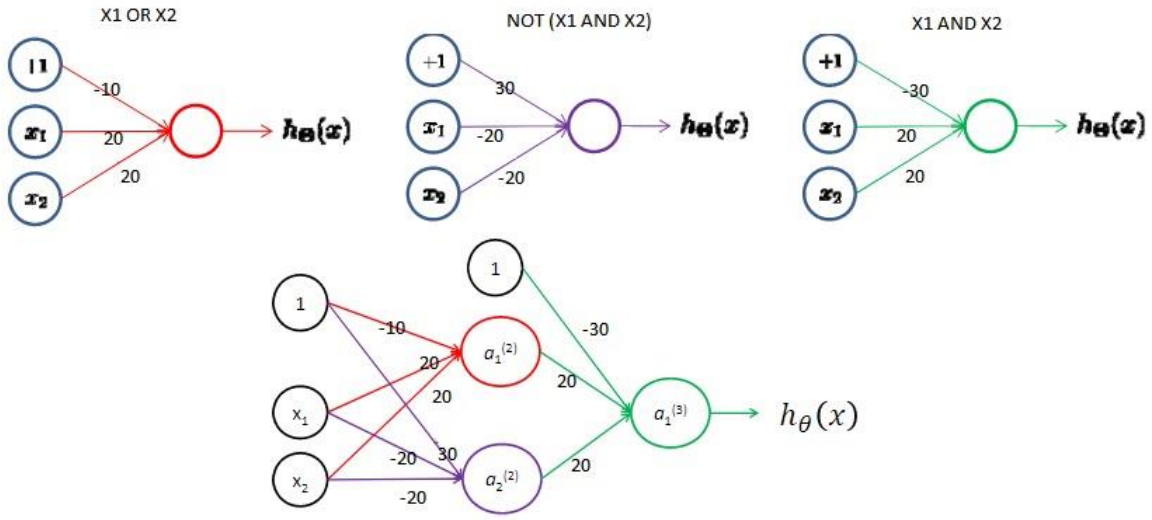


a) What is the dimension of the input features? _____3 or 3x1_____
b) Would this work better as a happy-neutral-sad or happy-sad classifier?____happy-sad_____
c) What is the dimension of the $\theta^{(2)}$?_____10 or 5x2_____
d) $x_0^{(1)}$ and $a_0^{(2)}$ are called ___bias____terms.
e) If the softmax function is used for the output layer, how many nodes use activation functions?____5____
f) If you had training samples, $\{x_i,y_i\}$, i=1…10,000, would you pre-train with an unsupervised technique? Yes or No___No_____

2. (8 pts) The XOR function between two binary variables x_1 and x_2 can be implemented using an OR gate, an AND gate, plus a two input to one output function of the form:
x_1 XOR x_2 = (x_1 OR x_2) AND (out = *func*(x_1 , x_2))
Determine and show what the *func* function is, then diagram each of the three terms as a two input (plus bias term) to one output neural network, then diagram how the three terms can be combined as a single neural network which implements the XOR function.

Ans:

```
% x1 XOR x2 = (x1 OR x2) AND (NOT(x1 AND x2))
%
% (NOT(x1 AND x2))
% a1 = -20, a2 = -20, w = 30
```



3. (10 pts) Please check which statements are true:

a) Today's deep networks have just as many neurons as the human brain.
b) Today's computers have faster clock speed than the human brain.
c) Artificial neurons faithfully replicate neurons in the human brain.
d) The cerebral cortex has similar structure throughout, enabling one section of the brain to learn tasks from other sections of the brain.
e) Neurons in the eye are tuned to specific visual sensory patterns.
f) Signals emerging from the eye have Gabor-like (dampened sinusoid) responses.
g) Signals emerging from VI have Gabor-like (dampened sinusoid) responses.
h) The brain is organized in a hierarchical fashion such that higher level layers, such as V3, encode more abstract representations of lower level layers, such as V1.
i) Scientists have recently discovered how the brain learns and are currently building an artificial brain similar to Hal, in the 2001: A Space Odyssey.
j) In all likelihood, computers will be smarter than humans some day in the future

Ans: b, d, e, g, h, j

4. (4 pts) The hyperbolic tangent and sigmoid activation functions have been very popular over the years. Recently, a new activation function is becoming popular. What is it's name?_____ReLu_____

5. (6 pts) In the fish, chips, and ketchup example, the difference between the actual price paid and our guess can be distributed over our weight estimates in proportion to the current weight estimates or input values. Which is preferred___proportional to input values_____ and why?

Ans: in proportion to input values, because all of the information we have is the input values.

6. (6 pts) In the fish, chips, and ketchup example, we solved for the change in error as a function of the weights. The logistic neuron had the formula:

$$\frac{\partial E}{\partial \theta_j} = \sum_{i=i}^{n} a(1-a)x(y_i - a_i)$$

Describe the meaning of each term.

Ans:

(y_i - a_i) is the difference between the prediction and the ground truth.

a * (1 - a) is the derivative for the activation function.

x_j is the individual neuron.

7. (12 pts) Backpropagation is used to train neural networks of many layers, using the same strategies as the fish, chips, and ketchup example, except it does so one layer at a time. Which statements concerning backpropagation are true:
   a) Backpropagation can solve for an arbitrary number of nodes and layers.
   b) In recent years, backpropagation has gone out of favor to better methods.
   c) Backpropagation works with supervised or unsupervised data.
   d) Given enough epochs, backpropagation will converge to the optimal solution if the learning rate is small enough.
   e) Initializing the weights to all zeros often helps convergence.
   f) Backpropagation can support weight regularization.
   g) Weight regularization is limited to norm-2.
   h) Backpropagation is a binary classifier, but can be extended to multi-class classification using methods similar to logistic regression and SVM.
   i) Backpropagation can use any activation function that is invertible.

Ans: a, f, i

8. (24 pts) Neural networks learn their weights iteratively. A sample's Xdata is forward propagated through the network, the output is compared to ground truth, and the error is then back propagated to update all the weights. This is repeated over and over for each training sample. In this exercise, we will work with a three layer network. The input layer will be 20×20 mnist digits '0' through '9', followed by 25 hidden units, and followed by ten output units for classification.

Start with the below code:

```
%% question 8a
close all ; clear all;
```

```
%Download hwk7files_forMycourses.zip and place them into an appropriate
%hwk7 directory.
%Update the two paths below for your machine
cd C:\Users\rwpeec\Desktop\rwpeec\rit\CMPE-677_MachineIntelligence\hwk\hwk7
addpath C:\Users\rwpeec\Desktop\rwpeec\rit\CMPE-677_MachineIntelligence\hwk\hwk6\libsvm-3.18\windows

% We will use mnist hand written digits, '0' through '9'
load('ex4data1.mat');  %5000 Mnist digits from Andrew Ng class
n = size(X, 1);  %number of samples = 5000, 500 from each class
D = size(X,2);  %number of dimensions/sample.  20x20=400
C = length(unique(y));  %number of classes, class values are 1...10, where 10 is a digit '0'

% Randomly select 100 data points to display
sel = randperm(size(X, 1));
sel = sel(1:100);

displayData(X(sel, :));  %This function is from Andrew Ng's online class

%Convert X and y data into Matlab nnet format:
inputs = X';
%one-hot encoding ground truth values
targets = zeros(C,n);
for ii=1:n
      targets(y(ii),ii) =  1;
end
%If given one-hot encoding, can convert back to vector of ground truth
%class values:
% target1Dvector=zeros(n,1);
% for ii=1:n
%       target1Dvector(ii) = find(targets(:,ii) == 1);
% end
% max(target1Dvector-y) %this will be zero

fprintf('\nLoading Saved Neural Network Parameters ...\n')

% Load the weights into variables Theta1 and Theta2
load('ex4weights.mat');  %Pre-learned weights from Andrew Ng class

% Unroll parameters
nn_params = [Theta1(:) ; Theta2(:)];
```

Now update the file nnCostFunction.m. This file computes the cost of our objective function and updates the gradients using the backpropagation function. We will modify this function three times:

a. Return the computed cost using our pre-trained weights without regularization,
b. Return the computed cost using our pre-trained weights with regularization,
c. Implement the forward propagation portion of the backpropagation algorithm

a) The neural network cost function we will be using is:

$$J(\theta) = \left[-\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y^{(i,c)} log\left(h_\theta\left(x^{(i)}\right)\right)_c + \left(1-y^{(i,c)}\right) log\left(1-h_\theta\left(x^{(i)}\right)\right)_c\right]$$

$$+ \quad \frac{\lambda}{2n}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\theta_{ji}^{(l)}\right)^2$$

Update nnCostFunction.m using this formula without the regularization term (look for <<update next line>>). Continuing with the Matlab code in this problem, execute the following lines of code and report the cost:

```
% Weight regularization parameter (we set this to 0 here).
lambda = 0;
 J = nnCostFunction(nn_params, input_layer_size, hidden_layer_size, ...
                    num_labels, X, y, lambda);
fprintf(['Cost at parameters (no regularization): %f \n'], J);
```

Ans (Cost and line of code from nnCostFunction.m):

```
% cost function- first without regularization
J = -(1/n)*sum(sum(ynn.*log(out)+(1-ynn).*log(1-out)));
```

b) Update nnCostFunction.m using the formula including the regularization term. Execute the following lines of code and report the cost:

```
% Weight regularization parameter (we set this to 1 here).
lambda = 1;
J = nnCostFunction(nn_params, input_layer_size, hidden_layer_size, ...
                    num_labels, X, y, lambda);
fprintf(['Cost at parameters (with regularization): %f \n'], J);
```

Ans (Cost and line of code from nnCostFunction.m):

```
% Add regularization to cost function
sumLayer1 = sum(sum(Theta1(:,2:end).^2));
sumLayer2 = sum(sum(Theta2(:,2:end).^2));
reg = (lambda/(2*n))*(sumLayer1 + sumLayer2);
J = J+reg;
```

c) You will now compute the gradients used in the backpropagation algorithm. In the nnCostFunction.m file, locate the section of code that does the calculation of gradients. You will find a loop over all *n* samples. Uncomment the matlab code in that loop (can do this by selecting all lines in the loop and then hit Ctrl-T). Update the 'z2 =' and 'z3 =' lines as well as the 'deltaPart3 =' line. The 'z2 =' and 'z3 =' lines compute the summation of all inputs times all weights for layers 2 and 3 respectively. The 'deltaPart3 =' line computes the error between a3 and the ground truth of the sample. Continuing with the Matlab code in this problem, execute the following lines of code to test your implementation of back propagation:

```
fprintf('\nInitializing Neural Network Parameters ...\n')
initial_Theta1 = randInitializeWeights(input_layer_size, hidden_layer_size);
initial_Theta2 = randInitializeWeights(hidden_layer_size, num_labels);
 % Unroll parameters
initial_nn_params = [initial_Theta1(:) ; initial_Theta2(:)];

fprintf('\nTraining Neural Network... \n')

%  After you have completed the assignment, change the MaxIter to a larger
%  value to see how more training helps.
options = optimset('MaxIter', 5);
```

```
%  You can try different values of lambda, but keep lambda=1 for this exercise
lambda = 1;

% Create "short hand" for the cost function to be minimized
costFunction = @(p) nnCostFunction(p, ...
                                   input_layer_size, ...
                                  hidden_layer_size, ...
                                  num_labels, X, y, lambda);

% Now, costFunction is a function that takes in only one argument (the
% neural network parameters)
[nn_params, cost] = fmincg(costFunction, initial_nn_params, options);

% Obtain Theta1 and Theta2 back from nn_params
Theta1 = reshape(nn_params(1:hidden_layer_size * (input_layer_size + 1)), ...
                 hidden_layer_size, (input_layer_size + 1));

Theta2 = reshape(nn_params((1 + (hidden_layer_size * (input_layer_size + 1))):end), ...
                 num_labels, (hidden_layer_size + 1));

% Visual weights- can uncomment out next two lines to see weights
% fprintf('\nVisualizing Neural Network... \n')
% displayData(Theta1(:, 2:end));

pred = predict(Theta1, Theta2, X);
fprintf('\nTraining Set Accuracy: %f\n', mean(double(pred == y)) * 100);
```

With five iterations, you will get a training set accuracy close to 50%. Modify the code for 50 iterations and report the training set accuracy. Show all code.

Ans:

```
%backpropogation, calculation of gradients
for t = 1:n
  % Step 1)  Forward propagate
  a1 = X1(t,:); % we did the bias above
  z2 = Theta1*a1';
  a2 = sigmoid(z2);
  a2 = [1;a2];  % need to add the bias back to this one
  z3 = Theta2*a2;
  a3 = sigmoid(z3);
  z2 = [1;z2];  % still need to worry about the bias effect

  % Step 2)  Compute error
  deltaPart3 = a3 - ynn(:,t);

  % Step 3) Back propagate error through activation function
  deltaPart2 = (Theta2' * deltaPart3) .* sigmoidGradient( z2 );

  % Step 4)  Update weights
  Theta2_grad = Theta2_grad + deltaPart3 * a2';
  Theta1_grad = Theta1_grad + deltaPart2(2:end) * a1;
end;
```

```
% Training Set Accuracy after 50 iterations: 94.960000
```

9. (22 pts) Matlab has built-in support for neural networks. These networks can be used for linear regression, classification, unsupervised clustering, or recurring frameworks. We will update our classify677.m file to include classification using neural networks. First, use the below code to learn how to use the patternnet.m function, which is the built-in neural network function in Matlab used for generic classification problems.

```
%% question 8a
close all ; clear all;
%Download hwk7files_forMycourses.zip and place them into an appropriate
%hwk7 directory.  Update the two paths below for your machine
cd C:\Users\rwpeec\Desktop\rwpeec\rit\CMPE-677_MachineIntelligence\hwk\hwk7
addpath C:\Users\rwpeec\Desktop\rwpeec\rit\CMPE-677_MachineIntelligence\hwk\hwk6\libsvm-3.18\windows

% We will use mnist hand written digits, '0' through '9'
load('ex4data1.mat');  %5000 Mnist digits from Andrew Ng class
n = size(X, 1);  %number of samples = 5000, 500 from each class
D = size(X,2);  %number of dimensions/sample.  20x20=400
C = length(unique(y));  %number of classes, class values are 1...10, where 10 is a digit '0'

%Convert X and y data into Matlab nnet format:
inputs = X';
%one-hot encoding ground truth values
targets = zeros(C,n);
for ii=1:n
      targets(y(ii),ii) =  1;
end

% Create a Pattern Recognition Network, with one hidden layer containing 25
% nodes
hiddenLayerSize = 25;
setdemorandstream(2014784333);   %seed for random number generator
net = patternnet(hiddenLayerSize);

% Set up Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 0.7;  %note- splits are done in a random fashion
net.divideParam.valRatio = 0.15;
net.divideParam.testRatio = 0.15;

% Train the Network
[net,tr] = train(net,inputs,targets);  %return neural net and a training record
plotperform(tr); %shows train, validation, and test per epoch

% Test the returned network on the testing split
testX = inputs(:,tr.testInd);
testT = targets(:,tr.testInd);
testY = net(testX);   %pass input test values into network
testIndices = vec2ind(testY);  %converts nnet float for each class into most likely class per
sample
figure; plotconfusion(testT,testY)
```

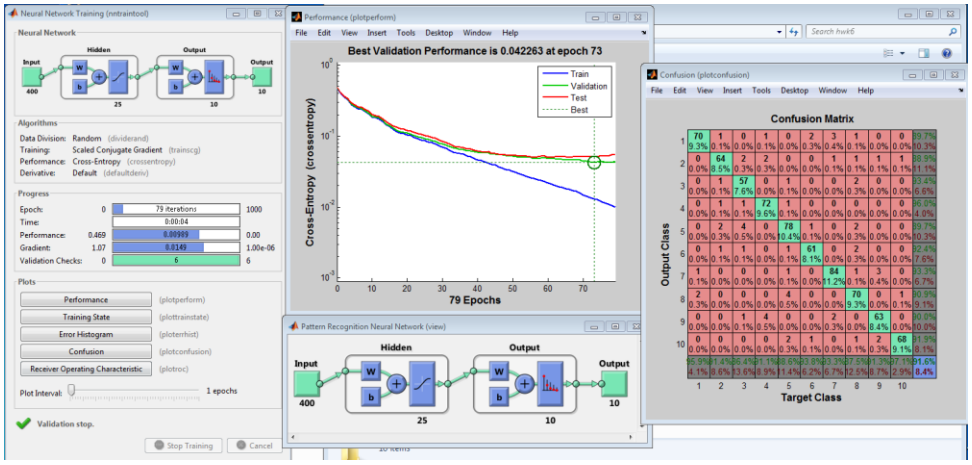
```
[c,cm] = confusion(testT,testY);

fprintf('Percentage Correct Classification   : %f%%\n', 100*(1-c));  %Should be approx 91.6%
fprintf('Percentage Incorrect Classification : %f%%\n', 100*c);  %Should be approx 8.4%
%Should be approx 91.6%

% If your nnet had 2 class output, try this set:
%[inputs,targets] = cancer_dataset;  %breast cancer dataset built into Matlab
% you can use the receiver
% operating characteristic (ROC) plot to measure of how well
% the neural network has fit data is the data.  This shows how the
% false positive and true positive rates relate as the thresholding of
% outputs is varied from 0 to 1.
% The farther left and up the line is, the fewer false positives need to
% be accepted in order to get a high true positive rate. The best classifiers
% will have a line going from the bottom left corner, to the top left corner,
% to the top right corner, or close to that.
% Note: for this to work, testY needs to be a softMax, or similar
% continuous ouput
%plotroc(testT,testY)

% View the Network
view(net); %shows each layer with number of inputs and outputs to each
```

Do not continue until you understand each line of the above code. Your percentage correct classification should be about 91.6%, and you should see four plots similar to:



Modify classify677_hwk7.m to be able to handle nnet functionality. You only need to modify classify677_hwk7.m where it states <insert code here>. When done, test with the following code:

```
%% question 9b
close all ; clear all;
%Download hwk7files_forMycourses.zip and place them into an appropriate
%hwk7 directory.  Update the two paths below for your machine
cd C:\Users\rwpeec\Desktop\rwpeec\rit\CMPE-677_MachineIntelligence\hwk\hwk7
addpath C:\Users\rwpeec\Desktop\rwpeec\rit\CMPE-677_MachineIntelligence\hwk\hwk6\libsvm-3.18\windows
```

```
% We will use mnist hand written digits, '0' through '9'
load('ex4data1.mat');  %5000 Mnist digits from Andrew Ng class
n = size(X, 1);  %number of samples = 5000, 500 from each class
D = size(X,2);  %number of dimensions/sample.  20x20=400
C = length(unique(y));  %number of classes, class values are 1...10, where 10 is a digit '0'

options.numberOfFolds = 5;
options.method = 'SVM';
[confusionMatrix_svm,accuracy_svm] =  classify677_hwk7(X,y,options);

options.method = 'nnet';
options.nnet_hiddenLayerSize = 25;
[confusionMatrix_nnet1,accuracy_nnet1] =  classify677_hwk7(X,y,options);

fprintf('Linear SVM accuracy is: %0.2f%%\n',accuracy_svm*100);
fprintf('Nnet accuracy with %d hidden layers, num nodes per layer = %d is:
%0.2f%%\n',length(options.nnet_hiddenLayerSize),options.nnet_hiddenLayerSize,accuracy_nnet
1*100);

options.method = 'nnet';
options.nnet_hiddenLayerSize = [25 10];
[confusionMatrix_nnet2,accuracy_nnet2] =  classify677_hwk7(X,y,options);
fprintf('Nnet accuracy with %d hidden layers, num nodes per layer = [%d %d] is:
%0.2f%%\n',length(options.nnet_hiddenLayerSize),options.nnet_hiddenLayerSize,accuracy_nnet
2*100);
```

After you get your code working, what are the three printouts from the fprintf statements?

Ans (only show fprintf output, do not show code or plots):

```
% Total nSV = 1344
% SVM: Accuracy =  90.68%
% Confusion Matrix:
%    488      3      0      4      2      0      1      0      2      0
%      6    453      3     11      4      3      6      7      0      7
%      6     12    437      0     25      0      5      8      6      1
%      3      4      2    462      1      3      1      2     18      4
%      3      7     29      5    427      6      0     12      6      5
%      3      5      0      7      5    474      0      2      0      4
%      6      6      4     19      1      0    441      0     23      0
%     13     18     17      3     14      5      1    423      2      4
%      2      4      5     18      3      1     21      3    440      3
%      0      4      1      3      1      2      0      0      0    489
% nnet: Accuracy =  90.98%
% Confusion Matrix:
%    479      1      3      1      2      0      2     10      2      0
%      5    433     13     10      8      7      8     12      1      3
%      2     13    448      0     17      2      9      5      4      0
%      3      3      1    460      1      9      2      3     17      1
%      2      4     22      4    432      8      2     14      6      6
%      3     13      0      3      8    464      0      4      0      5
%      5      4      0     10      0      0    463      4     12      2
%      5     12     10      4      9      7      3    442      6      2
%      3      2      5     11      5      0     19      3    448      4
```

```
%      0      0      4      2      3      2      1      6      2    480
% Linear SVM accuracy is: 90.68%
% Nnet accuracy with 1 hidden layers, num nodes per layer = 25 is:
90.98%
% nnet: Accuracy =  89.44%
% Confusion Matrix:
%    473      3      1      2      9      1      6      5      0      0
%      5    429     16      8      3      7     11     16      1      4
%      4     12    431      2     25      2      7      8      6      3
%      4      3      1    452      3     10      2      4     21      0
%      1      3     24      5    426     12      0     15      9      5
%      1     10      0      7     11    462      2      3      1      3
%      8      3      4      8      0      0    455      1     20      1
%     12      9     16      6     20      6      3    419      2      7
%      2      0      5     18      2      1     14      6    448      4
%      0      3      3      1      4      6      3      2      1    477
% Nnet accuracy with 2 hidden layers, num nodes per layer = [25 10] is:
89.44
```