**Multi-class logistic re-gression to recognize handwritten digits. However, logistic regression cannotform morecomplex hypotheses as it is only a linear classier**

**&**

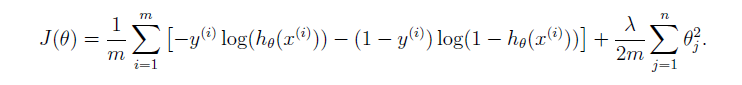
Implemented feedforward propagation for neural networks and used it to predict handwritten digits with the weights we

provided

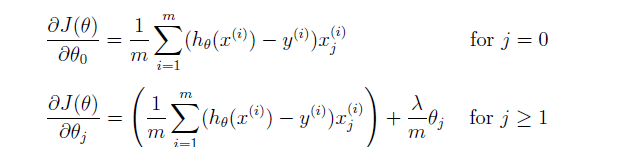
1. Always print the input and the output befire writing the logic
2. **[J grad] = lrCostFunction(theta\_t, X\_t, y\_t, lambda\_t);**

J

grad



**Code : J = (1/m) \* sum(CostFormulaPart1-CostFormulaPart2) + reg\_term;**



Code : **grad(1) = (1/m)\* (X(:,1)'\*(h\_x-y)); % 1 x 1**

**reg\_term\_gard = (lambda/m)\* theta(2:end);**

**grad(2:end) = (1/m)\* (X(:,2:end)'\*(h\_x-y))+ reg\_term\_gard**

Explanation :

One special keyword you can use in indexing is the end keyword

in indexing.

For example, A(:, 2:end) will only return elements from the 2nd

to last column of A.

**2nd one vs all**

In particular, your code should return all the classi\_er parameters

in a matrix \_ 2 RK\_(N+1) , where each row of \_ corresponds to the learned

logistic regression parameters for one class. You can do this with a \for"-loop

from 1 to K, training each classi\_er independently.

Matlab numbers 1 to 10(0)

1 = [0 2 = [ 0 ] 3 ka bhi something aisa kuch hoga

1

0

0 ]

Optinize Theta

Training One-vs-All Logistic Regression...

Iteration 50 | Cost: 1.400667e-02

Iteration 50 | Cost: 5.725250e-02

Iteration 50 | Cost: 6.419702e-02

Iteration 50 | Cost: 3.588632e-02

Iteration 50 | Cost: 6.186553e-02

Iteration 50 | Cost: 2.187608e-02

Iteration 50 | Cost: 3.380615e-02

Iteration 50 | Cost: 8.563031e-02

Iteration 50 | Cost: 7.982705e-02

Iteration 50 | Cost: 9.908205e-03

predictOneVsAll

% Making my examples in row

prob\_mat = X\* all\_theta'; % 5000 x 10 == no\_of\_input\_image x num\_labels

[prob, p] = max(prob\_mat,[],2); % m x 1

%returns maximum element in each row == max. probability and its index for each input image

% p " for each row kya output hoga

% prob accuracy : 95.02000 probability of predicted output

**Neural Network**

% Randomly select 100 data points to display // Just we are only displaying 100 images not 5000 images

Since the images are of size 20\_20, this gives us 400 input layer

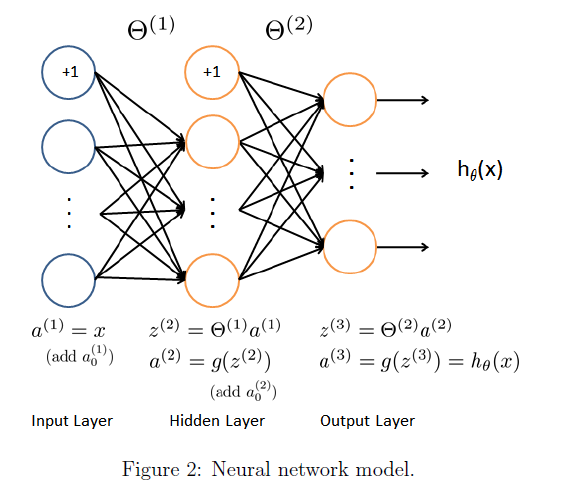
401(units in input layer) 25(units in second layer  ) 10(Output)

% The matrices Theta1 and Theta2 will now be in your Octave

% environment

% Theta1 has size 25 x 401

% Theta2 has size 10 x 26



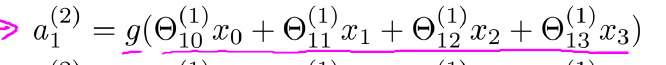
% theta dimensions = S\_(j+1) x ((S\_j)+1) // Refer the Formula in Slide

% theta1 = 25 x 401 (hidden unit x input unit(+1))

% theta2 = 10 x 26 (output unit x hidden unit(+1))

% \*\*\*\*\*\*\*\*\*\*\*\*\*\*Imagine the diagram why theta1 and theta2 has this

dimension??????\*\*\*\*\*\*



*% theta1:*

*% 1st row indicates: theta corresponding to all nodes from layer1 connecting to for 1st node of layer2*

*(Imagine a Diagram )*

*% 2nd row indicates: theta corresponding to all nodes from layer1 connecting to for 2nd node of layer2*

*% and*

*% 1st Column indicates: theta corresponding to node1 from layer1 to all nodes in layer2*

*% 2nd Column indicates: theta corresponding to node2 from layer1 to all nodes in layer2*

*%*

Is this the value of Theta 1 = [0.00000000280464 ] and 1st row and 401 column

prob\_mat = X \* all\_theta'; *% 5000 x 10 == no\_of\_input\_image x num\_labels*

[prob, p] = max(prob\_mat,[],2); *% m x 1*

*%returns maximum element in each row == max. probability and its index for each input image*

*%p: predicted output (index)*

*%prob: probability of predicted output*

Vs

*%%%%%%%% WORKING: Computation per input image %%%%%%%%%*

*% for i = 1:m % To iterate through each input sample*

*% one\_image = X(i,:); % 1 x 401 == 1 x no\_of\_features*

*% prob\_mat = one\_image \* all\_theta'; % 1 x 10 == 1 x num\_labels*

*% [prob, out] = max(prob\_mat);*

*% %out: predicted output*

*% %prob: probability of predicted output*

*% p(i) = out;*

*% end*

[prob, p] = max(a3,[],2);

*%returns maximum element in each row == max. probability and its index for each input image*

*%p: predicted output (index)*

*%prob: probability of predicted output*