**EX-4**

1. **nncostfunction**

%%%%%%%%%%% Part 1: Calculating J w/o Regularization %%%%%%%%%%%%%%%

X = [ones(m,1), X]; % Adding 1 as first column in X

a1 = X; % 5000 x 401

z2 = a1 \* Theta1'; % m x hidden\_layer\_size == 5000 x 25

a2 = sigmoid(z2); % m x hidden\_layer\_size == 5000 x 25

a2 = [ones(size(a2,1),1), a2]; % Adding 1 as first column in z = (Adding bias unit) % m x (hidden\_layer\_size + 1) == 5000 x 26

z3 = a2 \* Theta2'; % m x num\_labels == 5000 x 10

a3 = sigmoid(z3); % m x num\_labels == 5000 x 10

h\_x = a3; % m x num\_labels == 5000 x 10

%Converting y into vector of 0's and 1's for multi-class classification

%%%%% WORKING %%%%%

% y\_Vec = zeros(m,num\_labels);

% for i = 1:m

% y\_Vec(i,y(i)) = 1;

% end

%%%%%%%%%%%%%%%%%%%

y\_Vec = (1:num\_labels)==y; % m x num\_labels == 5000 x 10

%Costfunction Without regularization

J = (1/m) \* sum(sum((-y\_Vec.\*log(h\_x))-((1-y\_Vec).\*log(1-h\_x)))); %scalar

%%%%%%%%%%% Part 2: Implementing Backpropogation for Theta\_gra w/o Regularization %%%%%%%%%%%%%

%%%%%%% WORKING: Backpropogation using for loop %%%%%%%

% for t=1:m

% % Here X is including 1 column at begining

%

% % for layer-1

% a1 = X(t,:)'; % (n+1) x 1 == 401 x 1

%

% % for layer-2

% z2 = Theta1 \* a1; % hidden\_layer\_size x 1 == 25 x 1

% a2 = [1; sigmoid(z2)]; % (hidden\_layer\_size+1) x 1 == 26 x 1

%

% % for layer-3

% z3 = Theta2 \* a2; % num\_labels x 1 == 10 x 1

% a3 = sigmoid(z3); % num\_labels x 1 == 10 x 1

%

% yVector = (1:num\_labels)'==y(t); % num\_labels x 1 == 10 x 1

%

% %calculating delta values

% delta3 = a3 - yVector; % num\_labels x 1 == 10 x 1

%

% delta2 = (Theta2' \* delta3) .\* [1; sigmoidGradient(z2)]; % (hidden\_layer\_size+1) x 1 == 26 x 1

%

% delta2 = delta2(2:end); % hidden\_layer\_size x 1 == 25 x 1 %Removing delta2 for bias node

%

% % delta\_1 is not calculated because we do not associate error with the input

%

% % CAPITAL delta update

% Theta1\_grad = Theta1\_grad + (delta2 \* a1'); % 25 x 401

% Theta2\_grad = Theta2\_grad + (delta3 \* a2'); % 10 x 26

%

% end

%

% Theta1\_grad = (1/m) \* Theta1\_grad; % 25 x 401

% Theta2\_grad = (1/m) \* Theta2\_grad; % 10 x 26

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%% WORKING: Backpropogation (Vectorized Implementation) %%%%%%%

% Here X is including 1 column at begining

A1 = X; % 5000 x 401

Z2 = A1 \* Theta1'; % m x hidden\_layer\_size == 5000 x 25

A2 = sigmoid(Z2); % m x hidden\_layer\_size == 5000 x 25

A2 = [ones(size(A2,1),1), A2]; % Adding 1 as first column in z = (Adding bias unit) % m x (hidden\_layer\_size + 1) == 5000 x 26

Z3 = A2 \* Theta2'; % m x num\_labels == 5000 x 10

A3 = sigmoid(Z3); % m x num\_labels == 5000 x 10

% h\_x = a3; % m x num\_labels == 5000 x 10

y\_Vec = (1:num\_labels)==y; % m x num\_labels == 5000 x 10

DELTA3 = A3 - y\_Vec; % 5000 x 10

DELTA2 = (DELTA3 \* Theta2) .\* [ones(size(Z2,1),1) sigmoidGradient(Z2)]; % 5000 x 26

DELTA2 = DELTA2(:,2:end); % 5000 x 25 %Removing delta2 for bias node

Theta1\_grad = (1/m) \* (DELTA2' \* A1); % 25 x 401

Theta2\_grad = (1/m) \* (DELTA3' \* A2); % 10 x 26

%%%%%%%%%%%% WORKING: DIRECT CALCULATION OF THETA GRADIENT WITH REGULARISATION %%%%%%%%%%%

% %Regularization term is later added in Part 3

% Theta1\_grad = (1/m) \* Theta1\_grad + (lambda/m) \* [zeros(size(Theta1, 1), 1) Theta1(:,2:end)]; % 25 x 401

% Theta2\_grad = (1/m) \* Theta2\_grad + (lambda/m) \* [zeros(size(Theta2, 1), 1) Theta2(:,2:end)]; % 10 x 26

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%% Part 3: Adding Regularisation term in J and Theta\_grad %%%%%%%%%%%%%

reg\_term = (lambda/(2\*m)) \* (sum(sum(Theta1(:,2:end).^2)) + sum(sum(Theta2(:,2:end).^2))); %scalar

%Costfunction With regularization

J = J + reg\_term; %scalar

%Calculating gradients for the regularization

Theta1\_grad\_reg\_term = (lambda/m) \* [zeros(size(Theta1, 1), 1) Theta1(:,2:end)]; % 25 x 401

Theta2\_grad\_reg\_term = (lambda/m) \* [zeros(size(Theta2, 1), 1) Theta2(:,2:end)]; % 10 x 26

%Adding regularization term to earlier calculated Theta\_grad

Theta1\_grad = Theta1\_grad + Theta1\_grad\_reg\_term;

Theta2\_grad = Theta2\_grad + Theta2\_grad\_reg\_term;

1. **randintializeweights**

epsilon\_init = 0.12;

W = rand(L\_out, 1 + L\_in) \* 2 \* epsilon\_init - epsilon\_init;

1. **sigmoidGradient**

g = sigmoid(z) .\* (1 - sigmoid(z)) ;