Chinmayee Makaraju(1002091569)

EE5353 : Neural network and Deep Learning

Function:

```
function [Pe_test] = mlp_calc_PE(ic,y,Nv)

a = 0;
[max_val,ic_pred] = max(y,[],2);
for i = 1:Nv

            if ic(i) == ic_pred(i)
                a = a + 0;
else

a = a + 1;

end

end

        PE = (a*100)/Nv;
end
```

Main:

```
        clear all
        close all
        clc
        file = 2;
        % Get user input
        training_file = 'Gongtrn.dat' ;
        N = 16  ;
M = 10 ;

        OR_Method = 1;
        %% Read training classification file
[x, ic, Nv] = readClassificationFiles(training_file, N, M);

        %% One hot encoding method
t = generate_t(ic, M, Nv);
Nh = input('Enter the number of Hidden units (Nh): ');

fprintf('Appendix II: Training an MLP Using One-Step CG for
Nh = %f and data = %s\n',Nh,training_file)
Nw = (N+1)*(Nh + M) + M * Nh;
```

```
       Nv = size(x,1);
% % Make inputs zero mean, and use the same means to fix
validation data

% % (Notes II-E-4 Lemma 3: During BP training, input weights
are % % sensitive to input means.)
input_means = mean(x);
%input_means = zeros(1,N);

       x = x - repmat(input_means, [Nv 1]);
       Xa = [ones(Nv,1) x];
cascade = 0;

% Randomly initialize the input weights
% (Notes II-E-4 Lemma 2: Random initial weights insure that
no hidden units % are identical.)
mlp_randn(0,0,1);
Wi = mlp_randn(Nh, N+1);

       Wi = mlp_net_control(Xa, Wi);
       [R C] = mlp_calc_RC(Xa, Wi, t, cascade);
Et = sum(bsxfun(@times,t,t))/Nv;
n_const = 1;
[E,Wo] = ols3_sn(R, C, Et, n_const); % the new ols3 is fast
enough and may produce better results.

       fprintf('Initial Error is -> %f\n' ,E);
       if(cascade)
           Woh = Wo(:,2:end);
else

end

Nit = input('Enter the number of iterations (Nit): '); P =
zeros(Nh,N+1);
Poi = zeros(M,N+1);
Poh = zeros(M,Nh);

       XD = 1;
       Error = zeros(1,Nit);
       data= zeros(1,Nit);
       for it = 1:Nit
net = mlp_calc_net(Xa, Wi); % net function
O = mlp_calc_activation(net); %activation function

           dO = O .* (1-O);    % f'(np(k))
           [y ~] = mlp_calc_outputs(Xa, Wi, Wo);
           if(OR_Method)
               t = gore_or_fast(y, ic);
end
```

```
            %% Gradient calculations
Woh = Wo(:, N+2:end);
Woi = Wo(:, 1:N+1);
[G, Goi,Goh] =
One_step_gradient_calculations(Xa,t,O,dO,Woh,y);

XN = (sum(sum(G.^2)) + sum(sum(Goi.^2)) + sum(sum(Goh.^2)));
IP = (sum(sum(G.P)) + sum(sum(Goi . Poi)) + sum(sum(Goh .*
Poh)))/XN;

            B1 = XN/XD;
            XD = XN;
            P = G + B1*P;
            Poi = Goi + B1*Poi;
            Poh = Goh + B1*Poh;
            %% Learning factor calculations
z = One_Step_OLF_calc(Xa,dO,O,Woh, P,Poi,Poh,t,y);

            Wi = Wi + (z * P);
            Woi = Woi + (z*Poi);
            Woh = Woh + (z*Poh);
            Wo = [Woi ,Woh];
            [y ~] = mlp_calc_outputs(Xa, Wi, Wo);
            %% calculate Probability of error below
            PE= mlp_calc_PE(ic,y,Nv);
            %%%%%%%%%%%%%%%%%%%%%%%%%
            MSE = mlp_calc_mse(t, y);
fprintf('(%d)Error is -> %f--- Xn -> %f -----IP -> %f -----
PE -> %f\n' ,it,MSE,XN,IP,PE);

            Error(1,it) = MSE;
            data(1,it)= PE;
end

        figure
        plot(data,1:it);
        xlabel('Probability of Error');
        ylabel('Iterations');
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Get user input
        testing_file = 'Gongtst.tst' ;
N = 16 ; M = 10 ;

%% Read training file
% Write the code for testing the Classification data
Gongtst.tst [x, ic, Nv] =
readClassificationFiles(testing_file, N, M);
%% One hot encoding method
t = generate_t(ic, M, Nv);
```

```
Nh = input('Enter the number of Hidden units (Nh): ');
%fprintf('Appendix II: Training an MLP Using One-Step CG for
Nh = %f and data = %s\n',Nh,training_file)
Nw = (N+1)*(Nh + M) + M * Nh;
Nv = size(x,1);
% % Make inputs zero mean, and use the same means to fix
validation data % % (Notes II-E-4 Lemma 3: During BP
training, input weights are
% % sensitive to input means.)
x = x - repmat(input_means, [Nv 1]);
Xa = [ones(Nv,1) x];
[y ~] = mlp_calc_outputs(Xa, Wi, Wo);
PE_testing= mlp_calc_PE(ic,y,Nv);
fprintf('Error is -> %f\n' ,MSE);

fprintf('PE is -> %f\n' ,PE_testing);
```

Plot: