Problems for Week 12

Abhishek Bhattacharya

CSC 645

1)The data files (from assignment three) facetrain.txt nofacetrain.txt facetest.txt nofacetest.txt are made from images of faces and non-faces as follows. The images were converted to black and white and divided into a 7 by 7 grid, and each block was averaged to produce 49 numbers for each image, which are recorded in the rows of the above files.

This is clearly not a very intelligent way to extract features for face detection, but suffices for experimentation.

Build a two layer neural network for this data that is trained using the backprop method, and report on the performance on the test data.

ans. A two layer network looks like:

$$y = \sum_{j=1}^{M} w_j^{(2)} h\Big(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \Big) + w_0^{(2)} \tag{1}$$

where y is the posterior probability of Class 1. So the observation $\mathbf{x}$ is classified into Class 1 if $y > 0.5$ else into Class 2. To motivate the choice of h and initial weights, we look at the Fischer Linear Discriminant (FLD) classifier. That can be expressed as:

$$y = h\Big(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \Big) \tag{2}$$

where h is the Heaviside function. We replace h by the smooth logistic sigmoid:

$$h(x) = \frac{1}{1 + \exp(-x)}$$

and then (1) can be seen as a generalization of (2).

In the Network training phase, we start with the FLD weights obtained in Home-work 11 and forward propagate to get z,y. Then we compute $\nabla E(\mathbf{w})$ where

$$E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)^2$$

Note that

$$h'(a) = \frac{\exp(-a)}{\big(1+\exp(-a)\big)^2}$$

To solve $\nabla E(\mathbf{w}) = 0$ numerically, we use the Online gradient descent approach. That is we feed $\mathbf{x}$, t randomly from the training set and update $\mathbf{w}$ sequentially:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_n(\mathbf{w}^{(\tau)}), \quad \eta = 0.001$$

We update the weights if there is a decrease in error. We find the test and training sets errors for different M values to get the optimal$M$. Here is the Matlab code for the algorithm.

```
N = 200; D=49;
x = [face_tr
     noface_tr]; t = [zeros(100,1)
                      ones(100,1)];

x_test = [face_test
         noface_test]; t_test = [zeros(13,1)
                                 ones(13,1)];
x_test = [ones(26,1) x_test];

% randomly permute the training data
ind = randperm(N); x = x(ind,:); t = t(ind,:);
x = [ones(200,1) x];      % adding bias
Err = zeros(10,2); % traing error for different M
E_test = zeros(10,1); % test data error
miss_tr = zeros(10,1); % training missclassification
miss_test = zeros(10,1); % test missclassification

for M =1:10
 %M=1;
%intial forward propogation
w1 = zeros(M+1,D+1); w1(2,2:D+1) = w(1:D);
w2 = zeros(1,M+1); w2(2) = 1;
```

```
a = x*w1';
z = 1./(1 + exp(-a));
y = z*w2';

% initial error value
Err(M,1) = 1/2*(norm(y-t))^2; Err(M,2)=1/2*(norm(y-t))^2;

% ONLINE ERROR GRADIENT
for n =1:200
    Eold = 1/2*(norm(y-t))^2;
    % ERROR GRADIENT USING (x_n,t_n)
    delta2 = y(n)-t(n);
    delta1 = zeros(M+1,1);
    for j=0:M
        delta1(j+1) = exp(-a(n,j+1))/(1+ exp(-a(n,j+1)))^2*w2(j+1)*delta2;
    end
    delE1 = delta1*x(n,:); delE2 = delta2*z(n,:);

    % updating the weights & y: learning rate, eta
    eta = 10^(-3); w1n = w1 - eta*delE1; w2n = w2 - eta*delE2;

    % new forward propogation
    an = x*w1'; zn = 1./(1 + exp(-an)); yn = zn*w2n';
    % new error valuue
    En = 1/2*(norm(yn-t))^2;

    % update the weights if the error decreases
    if (En < Eold)
        w1 = w1n; w2 = w2n; a = an; z = zn; y = yn; Err(M,2) = En;
    end


end

% calculate error for test data
a_test = x_test*w1'; z_test = 1./(1 + exp(-a_test)); y_test = z_test*w2';
E_test(M) =  1/2*(norm(y_test-t_test))^2;

% percent miss-classifications
y2 = (y>0.5); miss_tr(M) = (200-sum(t==y2))/2;
y2 = (y_test>0.5); miss_test(M) = (26-sum(t_test==y2))/26*100;
```

```
end

Err                         % training set error for different M
E_test                      % TEST SET ERROR FOR DIFF. M
miss_tr                     % percent missclassification in training data for diffe
miss_test             % percent missclassification in test data for different M
```

I try $M = 1, 2, \ldots, 10$ and here are the errors values.
Intial FLD error on training data = 8.9369
Final error on training and test data:

M	Training-Error	Test-Error
1	1.4528	0.0000
2	2.0873	4.4882
3	2.7598	4.6895
4	3.1371	0.4936
5	1.6310	5.4834
6	1.5096	2.8033
7	1.7635	0.0007
8	1.2725	0.0018
9	3.1604	0.4206
10	3.0702	0.8241

M=8 seems to be the best choice. Percent missclassification for training set data is 2% while that for the test data is 0.