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:

\[ y = \sum_{j=1}^{M} w_j^{(2)} h\left( \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_0^{(2)} \]  

where \( y \) is the posterior probability of Class 1. So the observation \( x \) 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:

\[ y = h\left( \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) \]  

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 $\nabla E(w)$ where

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

Note that

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

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

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

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.

```matlab
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
  %initial forward proopagation
  w1 = zeros(M+1,D+1); w1(2,2:D+1) = w(1:D);
  w2 = zeros(1,M+1); w2(2) = 1;
  %...
\[ a = xw_1'; \]
\[ z = 1./(1 + \exp(-a)); \]
\[ y = zw_2'; \]

% initial error value
\[ \text{Err}(M,1) = 1/2*(\text{norm}(y-t))^2; \]
\[ \text{Err}(M,2)=1/2*(\text{norm}(y-t))^2; \]

% ONLINE ERROR GRADIENT
for \( n =1:200 \)
\[ \text{Eold} = 1/2*(\text{norm}(y-t))^2; \]
% ERROR GRADIENT USING \((x_n,t_n)\)
\[ \delta_2 = y(n)-t(n); \]
\[ \delta_1 = \text{zeros}(M+1,1); \]
for \( j=0:M \)
\[ \delta_1(j+1) = \exp(-a(n,j+1))/(1+ \exp(-a(n,j+1)))^2w_2(j+1)\delta_2; \]
end
\[ \delta_1 = \delta_1*x(n,:); \]
\[ \delta_2 = \delta_2*z(n,:); \]

% updating the weights & \( y \): learning rate, \( \eta \)
\[ \eta = 10^(-3); \]
\[ w_1n = w_1 - \eta*\delta_1; \]
\[ w_2n = w_2 - \eta*\delta_2; \]

% new forward propogation
\[ \text{a}_n = xw_1'; \]
\[ \text{z}_n = 1./(1 + \exp(-\text{a}_n)); \]
\[ \text{y}_n = \text{z}_n*w_2n'; \]
% new error value
\[ \text{E}_n = 1/2*(\text{norm}(\text{y}_n-t))^2; \]

% update the weights if the error decreases
if \( \text{E}_n < \text{Eold} \)
\[ w_1 = w_1n; \]
\[ w_2 = w_2n; \]
\[ a = \text{a}_n; \]
\[ z = \text{z}_n; \]
\[ y = \text{y}_n; \]
\[ \text{Err}(M,2) = \text{E}_n; \]
end

% calculate error for test data
\[ \text{a}_\text{test} = x\text{test}w_1'; \]
\[ \text{z}_\text{test} = 1./(1 + \exp(-\text{a}_\text{test})); \]
\[ \text{y}_\text{test} = \text{z}_\text{test}w_2'; \]
\[ \text{E}_\text{test}(M) = 1/2*(\text{norm}(\text{y}_\text{test}-t\text{test}))^2; \]

% percent miss-classifications
\[ \text{y}_2 = (\text{y}>0.5); \]
\[ \text{miss\_tr}(M) = (200-\text{sum}(t==\text{y}_2))/2; \]
\[ \text{y}_2 = (\text{y}_\text{test}>0.5); \]
\[ \text{miss\_test}(M) = (26-\text{sum}(t\text{test}==\text{y}_2))/26*100; \]
Err % training set error for different M
E_test % TEST SET ERROR FOR DIFF. M
miss_tr % percent misclassification in training data for different M
miss_test % percent misclassification 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:

<table>
<thead>
<tr>
<th>M</th>
<th>Training-Error</th>
<th>Test-Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.4528</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>2.0873</td>
<td>4.4882</td>
</tr>
<tr>
<td>3</td>
<td>2.7598</td>
<td>4.6895</td>
</tr>
<tr>
<td>4</td>
<td>3.1371</td>
<td>0.4936</td>
</tr>
<tr>
<td>5</td>
<td>1.6310</td>
<td>5.4834</td>
</tr>
<tr>
<td>6</td>
<td>1.5096</td>
<td>2.8033</td>
</tr>
<tr>
<td>7</td>
<td>1.7635</td>
<td>0.0007</td>
</tr>
<tr>
<td>8</td>
<td>1.2725</td>
<td>0.0018</td>
</tr>
<tr>
<td>9</td>
<td>3.1604</td>
<td>0.4206</td>
</tr>
<tr>
<td>10</td>
<td>3.0702</td>
<td>0.8241</td>
</tr>
</tbody>
</table>

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