Project 1

Statistical Patter Recognition

Girguis Sedky

1 ML/Bayes rule estimation

In this method, the image pixel values were assumed to have a gaussian distribution. In addition, the prior probability of all the classes were assumed to be equal during training and testing. Under this assumption, the Bayesian classification rule reduces to assigning a state to the class that maximizes

$$P(\boldsymbol{x}|\omega_i) = \frac{1}{((2\pi)^d|\Sigma|)^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right), \tag{1}$$

where $d = 784$ is the length of the state vectors. This is equivalent to maximizing

$$\log\left(P(\boldsymbol{x}|\omega_i)\right) = -\frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}). \tag{2}$$

The sample mean and covariances for each class are

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{x}_j, \tag{3}$$

$$\hat{\Sigma}_i = \frac{1}{n}\sum_{j=1}^{n}(\boldsymbol{x}_j-\hat{\boldsymbol{\mu}}_i)(\boldsymbol{x}_j-\hat{\boldsymbol{\mu}}_i)^{\mathrm{T}}. \tag{4}$$

A training script was used on the training set comprised of 60,000 images to determine $\hat{\boldsymbol{\mu}}_i$ and $\hat{\Sigma}_i$ of every class and a testing script in turn used these values and the Bayesian classification algorithm to classify the test images.

Since the covariance matrices are large and some are singular, special computational techniques were used to ensure good results. A pseudo inverse operation was used to inverse $|\Sigma_i|$ and small multiple of the identity was added to $|\Sigma_i|$ before obtaining its determinant. **The classification error on the testing set came to about** 35%

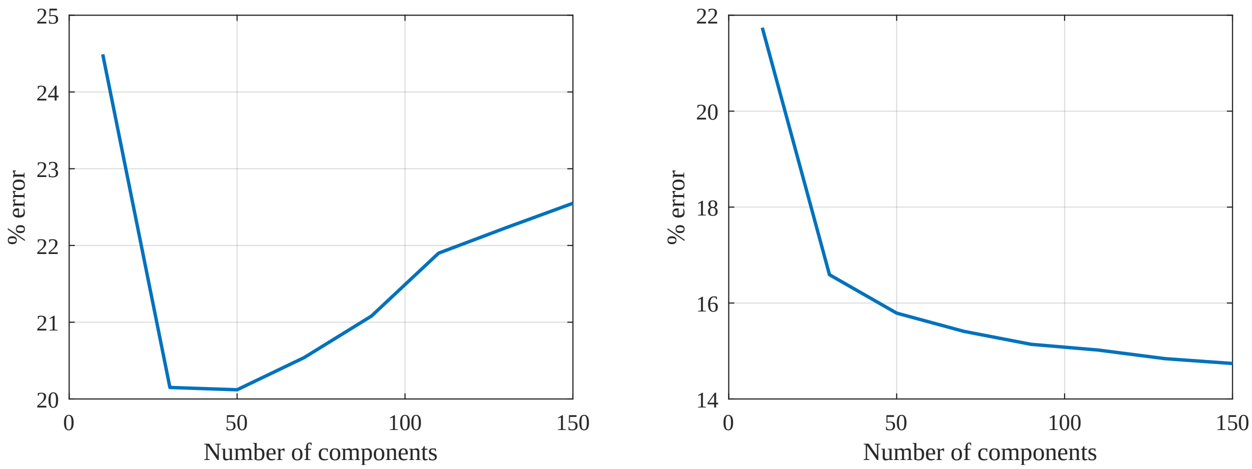
2 Nearest Neighborhood classifier

In this method, the image pixel values are arranged in a single column vector that lives in $\mathbb{R}^{784}$. In this space, each training image constitutes a prototype. The euclidiean distance between each testing image and all the training images is found, and the testing image is assigned to the class of the training image that is closest to it. **The classification error on the testing set came to about** 15%

3 PCA/LDA

3.1 PCA

Principal component analysis was used to reduce the dimensionality of the problem. The low-order state vectors were then fed into the Bayesian and Nearest Neighborhood classifier. Figure 1 demonstrates the percentage error in classification based on the number of dimensions kept. **The minimum classification error came to about 20.5% for 50 kept components in the case of the Bayesian classifier and 14.7% for 150 components kept in the case of the Nearest Neighborhood classier.**



(a) Bayesian classifier

(b) Nearest neighborhood classifier

Figure 1: Principal component analysis

3.2 LDA

Linear discriminant analysis was used to reduce the image state vectors from $\mathbb{R}^{784}$ to $\mathbb{R}^{M-1}$ where M is the number of classes. The built-in MATLAB function was used for this purpose. Again, the low-order state vectors were then fed into the Bayesian and Nearest Neighborhood classifier. **The minimum classification error came to about 25.4% in the case of the Bayesian classifier and 28% in the case of the Nearest Neighborhood classier.**

4 Summary

Table 1 shows the classification error of the different methods utilized in this study. The most accurate method of classification was to use principal component analysis to reduce the dimensionality of the state vectors to 50 and then to apply NN to the transformed data.

Table 1: Summary of classification results

Method	Percentage Error
Bayes	35%
NN	15%
PCA, Bayes (50)	20.5%
PCA, NN (150)	14.7%
LDA, Bayes	25.4%
LDA, NN	28%

Appendix

ML/Bayes rule estimation

Training script

```
%% training script
% find the estimated averages and covariances under gaussian assumptions of
% the training data and maximum likelihood
clear; clc; close all;
% Change the filenames if you've saved the files under different names
% On some platforms, the files might be saved as
% train-images.idx3-ubyte / train-labels.idx1-ubyte
images = loadMNISTImages('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data/train-images-idx3-ubyte');
labels = loadMNISTLabels('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data/train-labels-idx1-ubyte');
mu_hat_mat = [];
Epsilon_hat_cell= {};
% number of labels
labels_num = 10;
for jj=1:labels_num
images_label = images(:,labels==jj-1);
% n = number of images
n = size(images_label,2);
% L = number of pixels in an image
L = size(images_label,1);
% Find the maximum likelihoods estimates for the mean and covariance
mu_hat = mean(images_label,2);
Epsilon_hat = cov(images_label');
mu_hat_mat(:,jj) = mu_hat;
Epsilon_hat_cell{jj} = Epsilon_hat;

end
save('Estimates.mat','mu_hat_mat','Epsilon_hat_cell');
```

Testing script

```
%% Test script
% Test the ML algorthim based on gaussian assumptions of
% the training data and maximum likelihood
clear; clc; close all;
% Change the filenames if you've saved the files under different names
% On some platforms, the files might be saved as
% train-images.idx3-ubyte / train-labels.idx1-ubyte
```

```
images = loadMNISTImages('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data/t10k-images-idx3-ubyte');
labels = loadMNISTLabels('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data/t10k-labels-idx1-ubyte');
load('Estimates.mat');
% number of labels
labels_num = 10;
for ii=1:length(images)
% pick an image to classify
image = images(:,ii);
% n = number of images
n = size(image,2);
% L = number of pixels in an image
L = size(image,1);
for jj=1:labels_num
% Find the posterior probablity estimate of each one
cov = Epsilon_hat_cell{jj};
cov_det = cov+(10^-10)*eye(size(cov));  % Make covariance matrix full rank so that the logdet function can find a determinant
mu =  mu_hat_mat(:,jj);                 % average vector
% Find the likelihood of this class
logP(jj) = -0.5*logdet(cov_det)-0.5*(image-mu)'*pinv(cov)*(image-mu);

end
% Choose the macimym likelihood and assign the image to this class
[~,I] = max(logP);
labels_est(ii) = I-1;
clear P;
end
labels_est = labels_est';
% Test accuracy of classifying algorithm
a = find(labels_est==labels);
Performance = length(a)*100/length(labels);
save('Perfomance.mat','Perfomance');
```

NN

```
%% Nearest Neighborhood
clear; clc; close all;
% Girguis Sedky
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Load data
% Prototypes
```

```matlab
images_train = loadMNISTImages('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data\train-images-idx3-ubyte');
labels_train = loadMNISTLabels('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data\train-labels-idx1-ubyte');
% Test data
images_test = loadMNISTImages('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data\t10k-images-idx3-ubyte');
labels_test = loadMNISTLabels('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data\t10k-labels-idx1-ubyte');

%% Loop throuogh test data
for j = 1:length(images_test)
    % Pick image
    image = images_test(:,j);
    % Find eucliden distance to all the training set
    Dist = vecnorm(images_train-image,2,1);
    [~,I] = min(Dist);
    labels_est(j) = labels_train(I);
end
%% Test accuracy of classifying algorithm
a = find(labels_est==labels_test');
Performance = length(a)*100/length(labels_test);
save('Performance.mat','Performance')
```

PCA

4.1 ML/Bayes rule estimation with PCA

```matlab
%% ML with PCA
% find the estimated averages and covariances under gaussian assumptions of
% the training data and maximum likelihood, utilize PCA
clear; clc; close all;
% Girguis Sedky
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Load data
images_train = loadMNISTImages('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data\train-images-idx3-ubyte');
labels_train = loadMNISTLabels('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data\train-labels-idx1-ubyte'
```

```
);
images_test = loadMNISTImages('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data\t10k-images-idx3-ubyte');
labels_test = loadMNISTLabels('G:\My Drive\Classes&Books\StatisticalPatternRecognition\Project\Data\t10k-labels-idx1-ubyte');
mu_hat_mat = [];
Epsilon_hat_cell= {};
% number of labels
labels_num = 10;
% PCA
% Find the prinicipal components
coeff = pca(images_train');
% Number of dimensions kept
m = [10:20:150];
%% Loop over different values of m, number of dimensions kept
for ii=1:length(m)
%% Training Stage
% reduce the dimensionality of the system to 20
images_Transf = coeff(:,1:m(ii))'*images_train;
%% Loop over all the pictures in one class inside the training set
for jj=1:labels_num
images_label = images_Transf(:,labels_train==jj-1);
% n = number of images
n = size(images_label,2);
% L = number of pixels in an image
L = size(images_label,1);
% Find the maximum likelihoods estimates for the mean and covariance
mu_hat = mean(images_label,2);
Epsilon_hat = cov(images_label');
mu_hat_mat(:,jj) = mu_hat;
Epsilon_hat_cell{jj} = Epsilon_hat;
end
%% Test Stage
for kk=1:length(images_test)
% pick an image to classify
image = images_test(:,kk);
% reduce the dimensionality of the system to 20
image_Transf = coeff(:,1:m(ii))'*image;
for jj=1:labels_num
% Find the posterior probablity estimate of each one
COV = Epsilon_hat_cell{jj};
mu =  mu_hat_mat(:,jj);
P(jj) = (1/(sqrt(det(COV)*(2*pi)^m(ii))))*exp(-0.5*(image_Transf-mu)'*inv(COV)*(image_Transf-mu));
end
```

```
[~,I] = max(P);
labels_est(kk) = I-1;
clear P;
end
clear mu_hat_mat Epsilon_hat_cell;
%% Test accuracy of classifying algorithm
a = find(labels_est==labels_test');
Performance(ii) = length(a)*100/length(labels_test);

end
%% save and plot
save('Performance.mat','m','Performance');
plot(m,100-Performance,'LineWidth',1.5);
xlabel('Number of components');
ylabel('% error');
grid on;
PrintPlot(gcf,'plot.png','-dpng');
PrintPlot(gcf,'plot.pdf','-dpdf');
```

4.2 NN with PCA

```
%% Nearest Neighborhood, with PCA, investigate the effect of different
    components
clear; clc; close all;
% Girguis Sedky
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Load data
% Prototypes
images_train = loadMNISTImages('G:\My Drive\Classes&Books\
    StatisticalPatternRecognition\Project\Data\train-images-idx3-ubyte'
    );
labels_train = loadMNISTLabels('G:\My Drive\Classes&Books\
    StatisticalPatternRecognition\Project\Data\train-labels-idx1-ubyte'
    );
% Test data
images_test = loadMNISTImages('G:\My Drive\Classes&Books\
    StatisticalPatternRecognition\Project\Data\t10k-images-idx3-ubyte')
    ;
labels_test = loadMNISTLabels('G:\My Drive\Classes&Books\
    StatisticalPatternRecognition\Project\Data\t10k-labels-idx1-ubyte')
    ;
% PCA
% Find the prinicipal components
coeff = pca(images_train');
% Number of dimensions kept
```

```
m = [10:20:150];
%% Loop over different values of m, number of dimensions kept
for ii=1:length(m)
% reduce the dimensionality of the system to 20
images_train_Transf = coeff(:,1:m(ii))'*images_train;
images_test_Transf = coeff(:,1:m(ii))'*images_test;
    %% Loop throuogh test data
for j = 1:length(images_test_Transf)
    % Pick image
    image = images_test_Transf(:,j);
    % Find eucliden distance to all the training set
    Dist = vecnorm(images_train_Transf-image,2,1);
    [~,I] = min(Dist);
    labels_est(j) = labels_train(I);
end
%% Test accuracy of classifying algorithm
a = find(labels_est==labels_test');
Performance(ii) = length(a)*100/length(labels_test);
end
%% save and plot
save('Performance.mat','m','Performance');
plot(m,100-Performance,'LineWidth',1.5);
xlabel('Number of components');
ylabel('% error');
grid on;
PrintPlot(gcf,'plot.png','-dpng');
PrintPlot(gcf,'plot.pdf','-dpdf');
```

LDA

```
%% Reduce the data diemsnionality using LDA
clear; clc; close all;
% Girguis Sedky
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Load data
images_train = loadMNISTImages('G:\My Drive\Classes&Books\
    StatisticalPatternRecognition\Project\Data\train-images-idx3-ubyte'
    );
labels_train = loadMNISTLabels('G:\My Drive\Classes&Books\
    StatisticalPatternRecognition\Project\Data\train-labels-idx1-ubyte'
    );
images_test = loadMNISTImages('G:\My Drive\Classes&Books\
    StatisticalPatternRecognition\Project\Data\t10k-images-idx3-ubyte')
    ;
labels_test = loadMNISTLabels('G:\My Drive\Classes&Books\
```

```
StatisticalPatternRecognition\Project\Data\t10k-labels-idx1-ubyte');
% number of labels
A = LDA(images_train',labels_train);
A = A(1:end-1,1:end-1);
images_train = A*images_train;
images_test = A*images_test;
save('LDA_Data.mat','images_train','images_test','labels_train','labels_test');
```