Numerical Linear Algebra
TMA265/MMA600
Computer exercise 4:
Principal Component Analysis for image recognition

Larisa Beilina, larisa@chalmers.se

Computer exercise 4 (1 b.p. - 3 b.p.)

Principal Component Analysis for image recognition

This exercise can be viewed as background for the Master's project "Applications of Principal Component Analysis in computer vision"

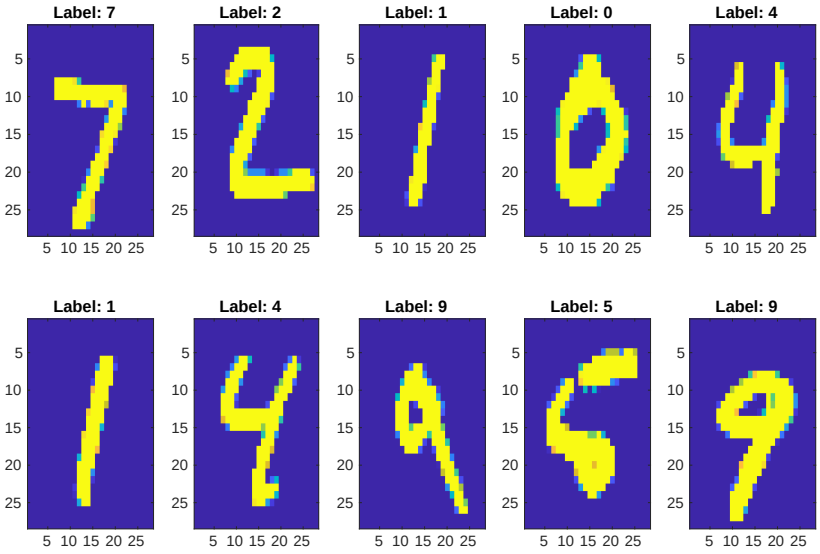


Figure 0.1: Images from MNIST dataset mnist_test_10.csv visualised via the program loadmnist_matlab.m

- Principal component analysis (PCA) is a machine learning technique which is widely used for data compression in image processing (data visualization) or in the determination of object orientation.
- PCA problem is closely related to the numerical linear algebra (NLA) problem of finding eigenvalues and eigenvectors for the covariance matrix.
- Further reading for AI algorithms: [2, 4, 5].

Computer exercise 4

- Use PCA to find patterns (recognize handwritten numbers) in MNIST Dataset of Handwitten Digits which can be downloaded from the link

  `http://makeyourownneuralnetwork.blogspot.com/2015/03/the-mnist-dataset-of-handwitten-digits.html`

  or from the course homepage. Test the code for different test and train sets.

- Optional: use PCA to classify skin images from the ISIC project, see link

  `https://www.isic-archive.com/#!/topWithHeader/wideContentTop/main`

  Link for download of data:

  `https://challenge.isic-archive.com/data#2020`

Hints:

- Study the lecture about PCA and the Matlab program which performs PCA for 2 datasets

  `ExamplePCA.m`

  on the course homepage.

- Use Matlab programs

  ```
  loadmnist_matlab.m
  import_mnist.m
  ```

  on the course homepage to download MNIST Datasets

  ```
  mnist_test_10.csv
  mnist_train.csv
  ```

- Study and modify the Matlab programs of section 1 which recognize 10 handwritten numbers via PCA analysis.

- Create your own training and test datasets from MNIST dataset and apply the Matlab code on them.

- Compute missclassification rate E using the formula (see [5], p. 211-214):

$$\mathrm{E} = \frac{\sum_{i=1}^{K} N_{F,i}}{\sum_{i=1}^{K}(N_{T,i} + N_{F,i})}, \tag{0.1}$$

  where K is the number of classes, $N_{T,i}$ is the number of images of the class i which are classified correctly, $N_{F,i}$ is the number of images of the class i which are classified wrong. Precision for class i can be computed as

$$\mathrm{P}(i) = \frac{N_{T,i}}{N_{T,i} + N_{F,j}}. \tag{0.2}$$

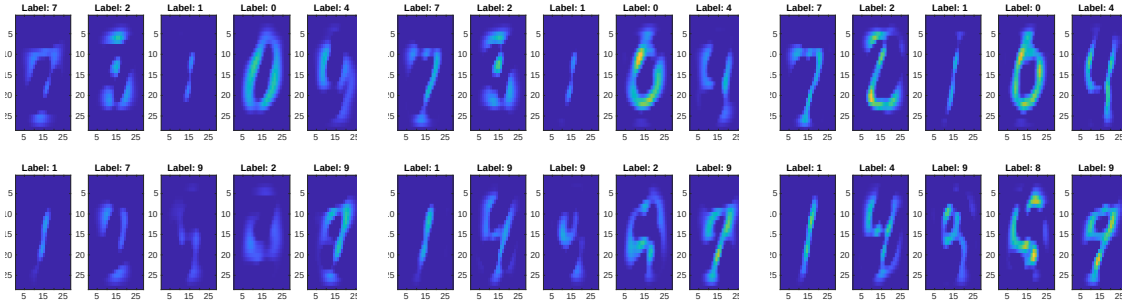
- Optional: use PCA to classify skin images from the ISIC project.

  Link for download of data:

  `https://challenge.isic-archive.com/data#2020`

  Choose the test dataset from ISIC database and compare with other images from the training dataset in order to determine to which one class belongs image from the train dataset. During classification assume that you don't know true class of images from train dataset. Compare then classified images with true ones.

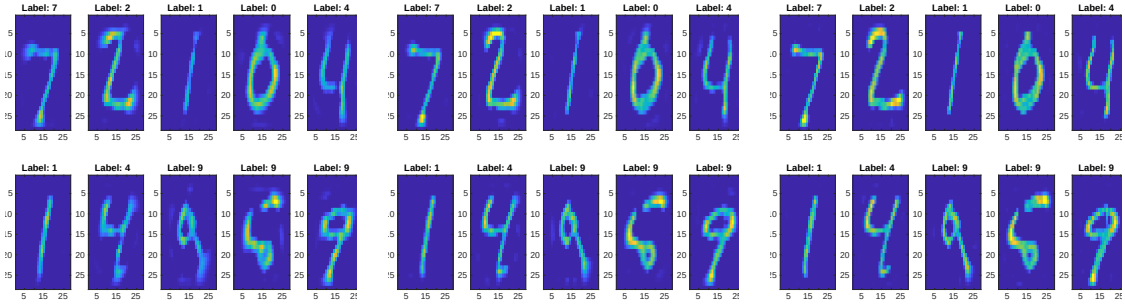
Programs



k=6

k=12

k=25



k=50

k=100

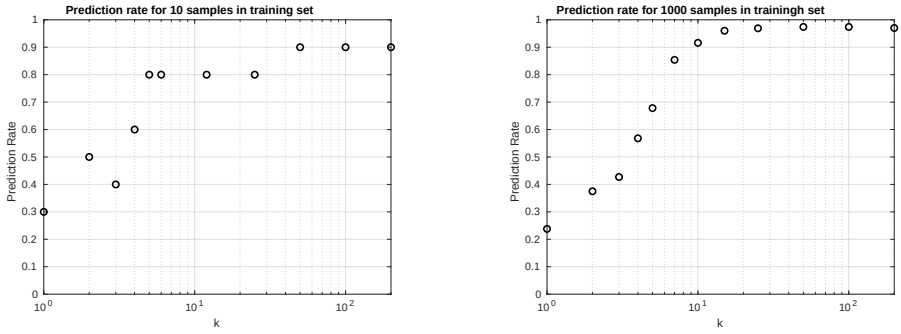
k=200

Figure 1.1: Recognition of handwritten numbers with Matlab code of this section using PCA for different number of principal components k.

1.1 Main Matlab program for recognition of handwritten digits

```
clear
close all
clc

all_examples = 1;
```



$N = 10$ $N = 1000$

Figure 1.2: Prediction rate in the recognition of handwritten numbers for different number N of samples in the test set.

```
number_examples_train = 31857;

% number of principle components
N_dim_to_keep = 100;
plotting = 1;


% import data from test and training sets
% here, Xdata is set of images with handwritten numbers
% and ydata is label for  every image in this set

[Xdata, ydata] = import_mnist(all_examples, number_examples_train);


d = size(Xdata,2);
N = size(Xdata,1);



% Choose 10 sample images as  a test set
[Xdata_n, ydata_n] = import_mnist(0, 10);

testSet = 11;
% The rest of the training set will be  training set  used for computations
Xdata = Xdata(testSet:end,:);
ydata = ydata(testSet:end);

xmean=mean(Xdata,1);
```

```
% compute adjusted data for training set
Xdata_adj = (Xdata-xmean);
%compute adjusted data for test set
Xdata_n_adj = (Xdata_n-xmean);

C = cov(Xdata_adj);
[U,S,V] = svd(C);


%projection in principal directions

% for training set
PCA = Xdata_adj*V(:,1:N_dim_to_keep);

% for test set
PCA_n = Xdata_n_adj*V(:,1:N_dim_to_keep);


n_correct = 0;
y_pred_data = [];
%y_pred_datatest = [];

for i=1:length(ydata_n)
    min_d=100000;
    for j=1:length(ydata)
       d=norm(PCA_n(i,:)-PCA(j,:));
        if d<min_d
            min_d=d;
            number=j;
           % y_pred_datatest(j) =ydata(number);
        end
    end

    if ydata_n(i)==ydata(number)
        n_correct = n_correct+1;
    end
    y_pred_data(i) = ydata(number);

end


sprintf('Prediction rate %d with k = %d',n_correct/length(ydata_n),N_dim_to_keep)
%%  plot images from tne test set and their   labels (recognition)
if plotting==true
    XdataPlot = (PCA_n*V(:,1:N_dim_to_keep)');

    sch=0;
```

```
    Nx = 5;
    Ny = 2;
    figure
    for i=1:Nx
        for j=1:Ny
        sch = sch+1;
        Ximage1 = reshape(XdataPlot(sch,:),[28 28]);
        subplot(2, 5, sch);
        %plot transposed data
        image(Ximage1');
        title(['Label: ',num2str(y_pred_data(sch))]);
        end
    end

end

  % Plot prediction rate for the training set


figure
k = [1,2,3,4,5,6,12,25,50,100,200];
Vec = [0.3,0.5,0.4,0.6,0.8,0.8,0.8,0.8,0.9,0.9,0.9];
semilogx(k,Vec,'ok','linewidth',1.5)
 title(['Prediction rate for 10 samples in training set ']);
 axis([0 200 0 1])
 xlabel('k')
 ylabel('Prediction Rate')
 grid on
```

1.2 The function **import_mnist.m**

```
function [X_train, y_train] = import_mnist(all_examples,number_examples)

% load data mnist
if all_examples == true
  data = csvread('mnist_train.csv');
  %  data = csvread('mnist_test_10.csv');
    X_train = data(1:end, 2:end);
    y_train = data(1:end, 1)';
    disp("Loaded data")
else
    data = csvread('mnist_test_10.csv');
    X_train = data(1:number_examples, 2:end);
    y_train = data(1:number_examples, 1)';
    disp("Loaded data")
end
```

```
end
```

References

[1] L. Beilina, E. Karchevskii, M. Karchevskii, Numerical Linear Algebra: Theory and Applications, Springer, 2017.

[2] Christopher M. Bishop, *Pattern recognition and machine learning*, Springer, 2009.

[3] G. S. Fulcher, Analysis of recent measurements of the viscosity of glasses, *Journal of American Ceramic Society*, `https://doi.org/10.1111/j.1151-2916.1925.tb16731.x`, 1925

[4] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`

[5] Miroslav Kurbat, *An Introduction to Machine Learning*, Springer, 2017.