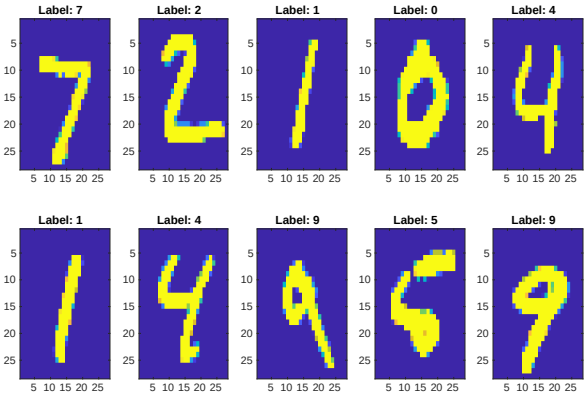
Numerical Linear Algebra TMA265/MMA600
Computer Labs

Larisa Beilina, larisa@chalmers.se

Instructions

- To pass this course you should do **any 2** computer assignments described below.
- You can work in groups by 2 persons.
- Sent final report for every computer assignment with description of your work together with Matlab or C++/PETSc programs to my e-mail before deadline. Report should have description of used techniques, tables and figures confirming your investigations. Analysis of obtained results is necessary to present in section "Numerical examples" and summarized results - in section "Conclusions". You can download latex or pdf-template for report from the course homepage.
- Matlab and C++ programs for examples in the book [1] are available for download from the course homepage: go to the link of the book [1] and click to "GitHub Page with MATLAB® Source Codes" on the bottom of this page, or copy the link below:

  `https://github.com/springer-math/Numerical_Linear_Algebra_Theory_and_Applications`



Computer exercise 1 (1 b.p.)

Solution of least squares problem

Consider the nonlinear model equation

$$y(T) = A \cdot \exp^{\frac{E}{T-T_0}}$$

presenting one of the models of the viscosity of glasses (see paper G. S. Fulcher, "ANALYSIS OF RECENT MEASUREMENTS OF THE VISCOSITY OF GLASSES" on the course homepage). Here, T is the known temperature, $y(T)$ is the known output data. Determine parameters A, E, T_0 which are positive constants by knowing T and output data $y(T)$.

Hints:

1. Transform first the nonlinear function $y(T)$ to the linear one and formulate then the linear least squares problem. Discretize T by N points and compute discrete values of $y(T)$ as $y_i = y(T_i)$ for the known values of parameters A, E, T_0. Then forget about these parameters (we will call them exact parameters A^*, E^*, T_0^*) and solve the linear least squares problem to recover these exact parameters.

2. You can choose exact parameters A^*, E^*, T_0^* as well as the interval for the temperature T as some positive constants accordingly to the Table II of the paper G. S. Fulcher, "ANALYSIS OF RECENT MEASUREMENTS OF THE VISCOSITY OF GLASSES".

   For example, take $E^* = 6 \cdot 10^3, A^* = exp^{-2.64}, T_0^* = 400, T = 750 + 10 * i, i = 1, ..., N$, where N is the number of discretization points. Interval for T can be, for example, $T = [750, 2000]$.

3. Investigate effect of random initialization noise δ in data $Y(T)$ obtained after transformation procedure, on the reconstruction of parameters A, E, T_0.

   Random noise δ to data $Y(T)$ can be added using the formula

   $$Y_\delta(T) = Y(T)(1 + \delta\alpha), \tag{0.1}$$

   where $\alpha \in (-1, 1)$ is randomly distributed number and $\delta \in [0, 1]$ is the noise level. For example, if noise in data is 5%, then $\delta = 0.05$. You can use several Matlab's functions to test adding of the noise. Below is an example of the Matlab code which shows how to add noise for solution of Poisson's equation (example of section 8.4.4 of the course book [1]) (see Figure 0.1):

```
r = randi([-1 1],size(u),1)
for j=1:n
  for i=1:n
    udelta(n*(i-1)+j) = u(n*(i-1)+j)*(1 + 0.1*r(n*(i-1)+j));
  end
end
```

Try also add normally distributed Gaussian noise

$$N(y|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y-\mu)^2}{2\sigma^2}}.$$

Here, μ is mean, σ^2 is variance, σ is standard deviation.

Below is example how to add Gaussian noise $N(y|\mu,\sigma^2)$ with mean $\mu = 0$ and variance $\sigma^2 = 0.01$ to matrix A in MATLAB:

```
Anoise = A + 0.01*randn(size(A)) + 0;
```

4. Solve the linear least squares problem using the method of normal equations, QR and then SVD decompositions. Analyze obtained results by computing the relative errors e_A, e_E, e_{T_0} in the computed parameters depending on the different noise level $\delta \in [0, 1]$ in data $Y_\sigma(T)$ for every method.

   The relative errors e_A, e_E, e_{T_0} in the computed parameters A, E, T_0 compute as:

$$\begin{aligned} e_A &= \frac{|A - A^*|}{|A^*|}, \\ e_E &= \frac{|E - E^*|}{|E^*|}, \\ e_{T_0} &= \frac{|T_0 - T_0^*|}{|T_0^*|}. \end{aligned} \tag{0.2}$$

   Here, A^*, E^*, T_0^* are exact values and A, E, T_0 are computed one. Present results how relative errors (0.2) depend on the random noise $\delta \in [0, 1]$ in graphical form and in the corresponding table.

5. Choose different number of discretization points N in the interval for temperature T and present results of computations in graphical form and in the corresponding table. More precisely, present how relative errors (0.2) depend on the number of measurements N if you solve the linear least squares problem using 3 methods: the method of normal equations, QR and then SVD decomposition.

6. Using results obtained in items 4 and 5, analyze, what is the minimal number of observations N to get reasonable reconstruction of parameters A, E, T_0 within the noise level σ ?

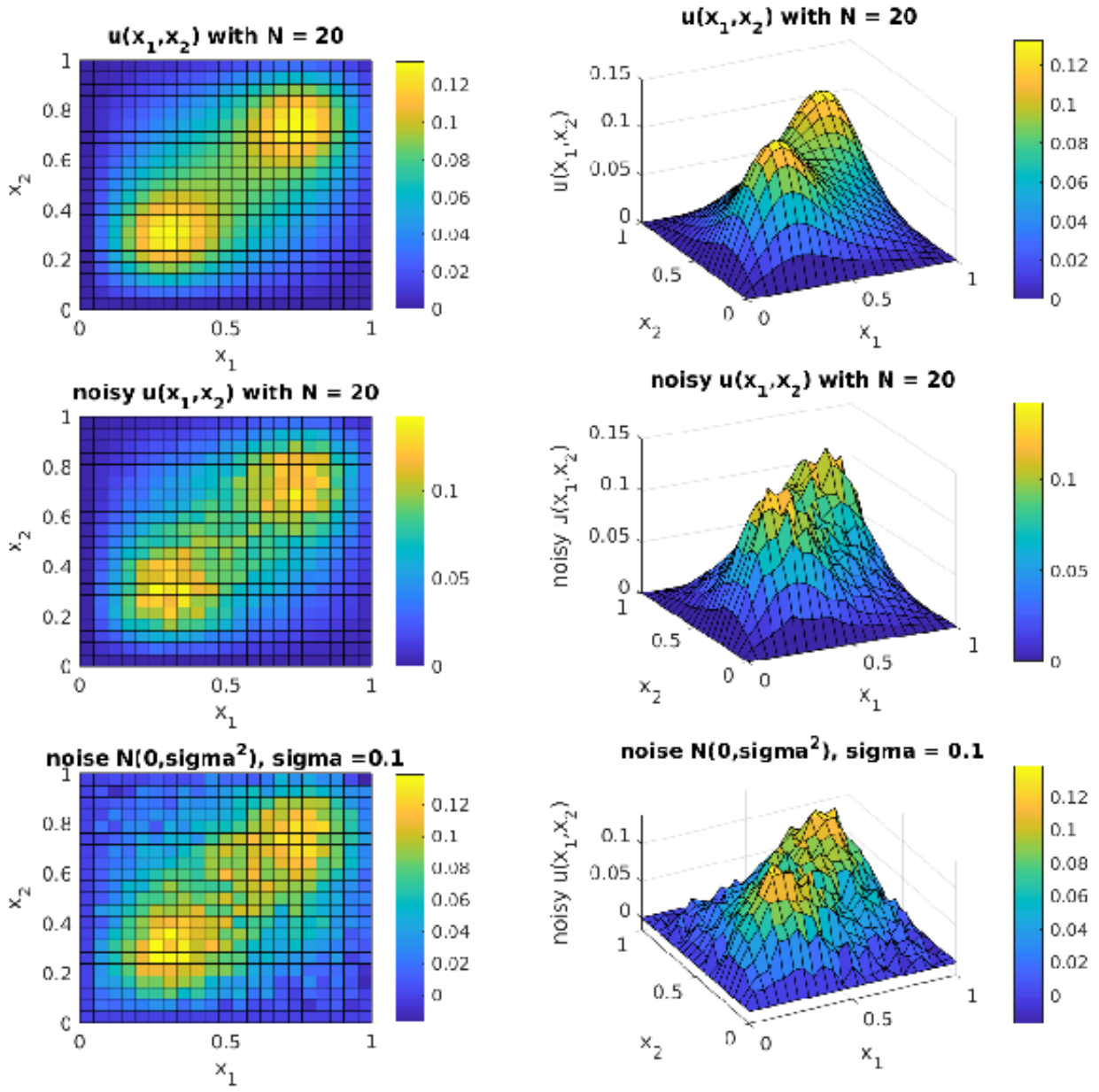


Figure 0.1: Top figures: Solution of Poisson's equation (example of section 8.4.4 of the course book [1]). Middle figures: Noisy solution obtained via (0.1). Bottom figures: noisy solution obtained via adding normally distributed Gaussian noise $N(y|0, 0.01)$.

Computer exercise 2 (1 b.p.)

Least squares and machine learning algorithms for classification problem

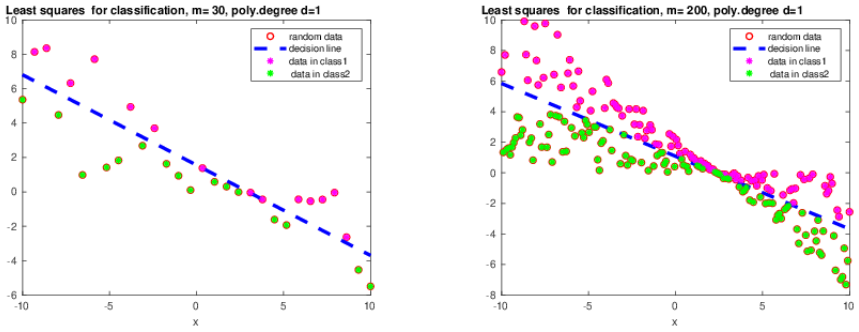


Figure 0.2: Examples of linear regression for classification for different number of input points.

In this exercise we will study different linear and quadratic classifiers: least squares classifier and perceptron learning algorithm using training sets described below.

Computer exercise 2

Implement in MATLAB least squares classifier and perceptron learning algorithm and present decision lines for following training sets:

- For randomly distributed data $y_i, i = 1, ..., m$ generated by the line

$$-1.2 + 0.5x + y = 0 \tag{0.3}$$

  on the interval $x = [-10, 10]$. Generate random data $(x, y_\sigma(x))$ using the formula (similar with comp.ex. 1)

$$y_\sigma(x) = y(x)(1 + \delta\alpha),$$

  where $\alpha \in (-1, 1)$ is randomly distributed number and $\delta \in [0, 1]$ is the noise level. For example, if the noise level in data is 5%, then $\delta = 0.05$. Then separate your points into 2 classes as follows: all points $(x_i, y_{\sigma i})$ which will be on the left side of the line (0.3) mark with code 1, and all points $(x_i, y_{\sigma i})$ which will be on the right side of the line (0.3) mark with code 0. In this way you will construct also your target vector t. Your points separated into 2 classes should look similarly as in Figure 0.3.

- Perform different experiments with different number of generated data points $m > 0$ which you choose as you want (for example, $m = 10, 100, 1000$).

- Add again noise as in (0.3) to already separated points into 2 classes and obtain new noisy data $y_\sigma(x)$. When you will add large noise σ you will observe that two classes

are not more linearly separated. Check if classification algorithms are still working. Explain why some of algorithms are not working properly.

Optional (not necessary): compute missclassification rate E using the formula (see [4], p. 211-214):

$$\mathrm{E} = \frac{\sum_{i=1}^{K} N_{F,i}}{\sum_{i=1}^{K}(N_{T,i} + N_{F,i})}, \tag{0.4}$$

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

$$\mathrm{P}(i) = \frac{N_{T,i}}{N_{T,i} + N_{F,j}}. \tag{0.5}$$

Try answer to the following questions:

- Analyze what happens with performance of perceptron learning algorithm if we take different learning rates $\eta \in (0, 1]$? For what values of η perceptron learning algorithm is more sensitive and when the iterative process is too slow?
- Analyze which one of the studied classification algorithms perform best and why?
- Try to explain in which case the perceptron algorithm will fail to separate data.

Hints:

1. In this exercise we will assume that we will work in domains with two classes: positive class and negative class. We will assume that each training example $\mathbf{x}$ can have values 0 or 1 and we will label positive examples with $c(\mathbf{x}) = 1$ and negative with $c(\mathbf{x}) = 0$.
2. We will also assume that these classes are linearly separable. These two classes can be separated by a linear function of the form

$$\omega_0 + \omega_1 x + \omega_2 y = 0, \tag{0.6}$$

where x, y are Cartesian coordinates. Note that the equation (0.6) can be rewritten for the case of a linear least squares problem as

$$\omega_0 + \omega_1 x = -\omega_2 y \tag{0.7}$$

or as

$$-\frac{\omega_0}{\omega_2} - \frac{\omega_1}{\omega_2} x = y. \tag{0.8}$$

3. We can determine coefficients $\omega_0, \omega_1, \omega_2$ in (0.6) by solving 2 different least squares problem:
   - By solving the following least squares problem for fitting the data y:

$$\min_{\omega} \|A\omega - y\|_2^2 \tag{0.9}$$

with $\omega = [\omega_1, \omega_2]^T = [-\frac{\omega_0}{\omega_2}, -\frac{\omega_1}{\omega_2}]^T$, where rows in the matrix A are given by

$$[1, x_k], \quad k = 1, ..., m,$$

and the known data vector is $y = [y_1, ..., y_m]^T$.

– Or by solving the following least squares problem:

$$\min_{\omega} \|f(x, y, \omega) - t\|_2^2 \tag{0.10}$$

with the linear model equation

$$f(x, y, \omega) = \omega_0 + \omega_1 x + \omega_2 y \tag{0.11}$$

and the target values of the vector $t = \{t_i\}, i = 1, ..., m$ which are defined as

$$t_i = \begin{cases} 1 & \text{point of 1 class,} \\ 0 & \text{point of 2 class.} \end{cases} \tag{0.12}$$

Thus, in the least squares problem

$$\min_{\omega} \|A\omega - y\|_2^2 \tag{0.13}$$

rows in matrix A will be:

$$[1, x_k, y_k], \quad k = 1, ..., m.$$

4. The Perceptron learning algorithm is taken from [4] and is presented below. Decision line then can be presented in Matlab for already computed weights by Perceptron learning algorithm using the formula (0.8).

5. Useful links for the literature in AI: [2, 3, 4]. Details about linear and quadratic perceptron learning algorithm and their implementation can be found in the paper *Numerical analysis of least squares and perceptron learning for classification problems* which can be downloaded from the link

   `https://arxiv.org/pdf/2004.01138.pdf`

6. Example of the Matlab code for classification of 2 classes using least squares, linear and quadratic perceptron on IRIS flower data set

   `http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/2021/Matlab/iris.csv`

   is available on the course homepage:

   `http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/2021/Matlab/testiris2.m`

   The complete dataset IRIS can be downloaded from the link:

   `https://en.wikipedia.org/wiki/Iris_flower_data_set`

Perceptron learning algorithm [4]

Assume that two classes c($\mathbf{x}$)=1 and c($\mathbf{x}$)=0 are linearly separable.

Step 0. Initialize all weights ω_i in

$$\sum_{i=0}^{n} \omega_i x_i = 0$$

to small random numbers (note $x_0 = 1$). Choose an appropriate learning rate $\eta \in (0, 1]$.

Step 1. For each training example $\mathbf{x} = (x_1, ..., x_n)$ whose class is $c(\mathbf{x})$ do:

- (i) Assign $h(\mathbf{x}) = 1$ if
$$\sum_{i=0}^{n} \omega_i x_i > 0$$
and assign $h(\mathbf{x}) = 0$ otherwise.
- (ii) Update each weight using the formula
$$\omega_i = \omega_i + \eta \cdot [c(\mathbf{x}) - h(\mathbf{x})] \cdot x_i.$$

Step 2. If $c(\mathbf{x}) = h(\mathbf{x})$ for **all training examples** stop, otherwise go to Step 1.

Computer exercise 3 (2 b.p.)

Regularized Least squares and machine learning algorithms for classification problem

This exercise can be viewed as beginning for the Master's project "Classification of skin cancer using regularized neural networks" for the skin images from the ISIC project, see link

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

In this exercise we will study regularized versions of least squares and perceptron learning algorithms for solution of classification problem. Tikhonov's regularization techniques are presented in the paper *Numerical analysis of least squares and perceptron learning for classification problems* which can be downloaded from the link

`https://arxiv.org/pdf/2004.01138.pdf`

Details about AI algorithms for classification together with machine learning techniques for choosing the reg.parameter can be found in [2, 3, 4].

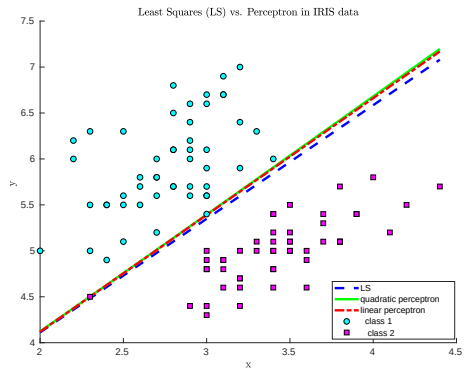


Figure 0.3: Decision lines computed by least squares and the perceptron learning algorithm for separation of two classes using Iris dataset. The dataset iris.csv is available for download from the course page.

Computer exercise 3

Implement in MATLAB all these classification algorithms and present decision lines for following training sets:

- I) Classify IRIS flower data set into several classes using regularized versions of least squares and perceptron learning algorithms. The dataset can be downloaded from the link:

  `https://en.wikipedia.org/wiki/Iris_flower_data_set`

- II) Try to test different regularization techniques (1-2 techniques is enough to test, choose any Tikhonovs technique or machine learning technique) for choosing the regularization parameter. More precisely, test some of Tikhonov's techniques (a priori rule, Morozov's discrepancy, balancing principle) described in Section 5 in the paper *Numerical analysis of least squares and perceptron learning for classification problems* which can be downloaded from the link

  `https://arxiv.org/pdf/2004.01138.pdf`

  Machine learning techniques for choosing the regularization parameter are presented in Section 7 of [3], see `http://www.deeplearningbook.org`.

- III) Optional (not necessary): compute missclassification rate E using the formula (see [4], p. 211-214):

$$\mathrm{E} = \frac{\sum_{i=1}^{K} N_{F,i}}{\sum_{i=1}^{K}(N_{T,i} + N_{F,i})}, \tag{0.14}$$

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

$$\mathrm{P}(i) = \frac{N_{T,i}}{N_{T,i} + N_{F,j}}. \tag{0.15}$$

- IV) Optional (not necessary, but if you want to obtain 2 b.p. for this comp.lab.): take some experimental data for classification from the link

  `https://archive.ics.uci.edu/ml/datasets.html`

  or the link for skin images:

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

  and classify them using regularized versions of least squares and perceptron learning algorithms.

Try answer to the following questions:

- Analyze different proposed techniques for choosing the regularization parameter. Which one of techniques works best ?
- Analyze what happens with performance of perceptron learning algorithm if we take different learning rates $\eta \in (0, 1]$? For what values of η perceptron learning algorithm is more sensitive and when the iterative process is too slow?
- Analyze which one of the studied classification algorithms perform best and why?
- Try to explain in which case the perceptron algorithm will fail to separate data.

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"

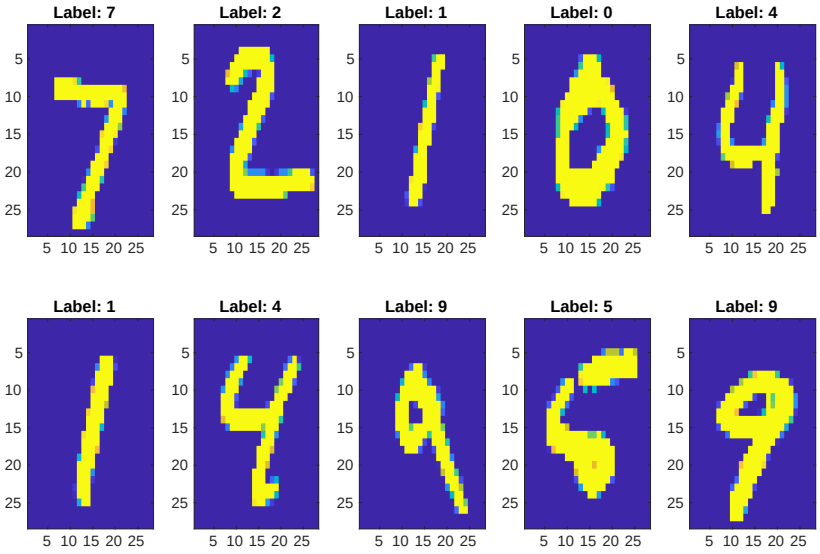


Figure 0.4: Images from MNIST dataset mnist_test_10.csv visualised via the program loadmnist_matlab.m

- 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, 3, 4].

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`

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.

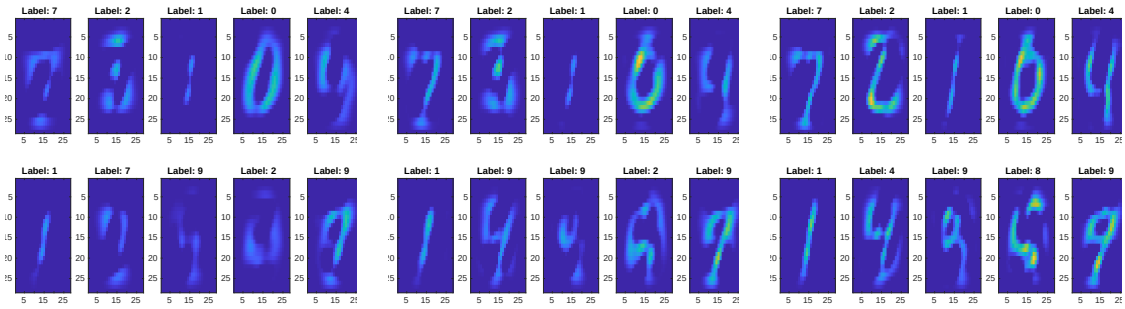
- Compute missclassification rate E using the formula (see [4], p. 211-214):

$$\mathrm{E} = \frac{\sum_{i=1}^{K} N_{F,i}}{\sum_{i=1}^{K}(N_{T,i} + N_{F,i})}, \tag{0.16}$$

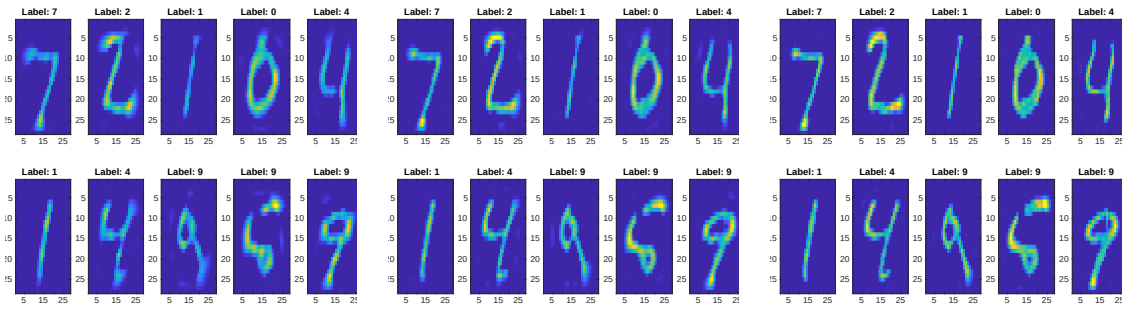
  where K is the number of classes, $N_{T,i}$ is the number of images of the class i which are classified correctly, $N_{F,i}$ is the number of images of the class i which are classified wrong. Precision for class i can be computed as

$$\mathrm{P}(i) = \frac{N_{T,i}}{N_{T,i} + N_{F,j}}. \tag{0.17}$$

- Optional: use PCA to classify skin images from the ISIC project, see link
  `https://www.isic-archive.com/#!/topWithHeader/wideContentTop/main`
  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.



k=6 k=12 k=25



k=50 k=100 k=200

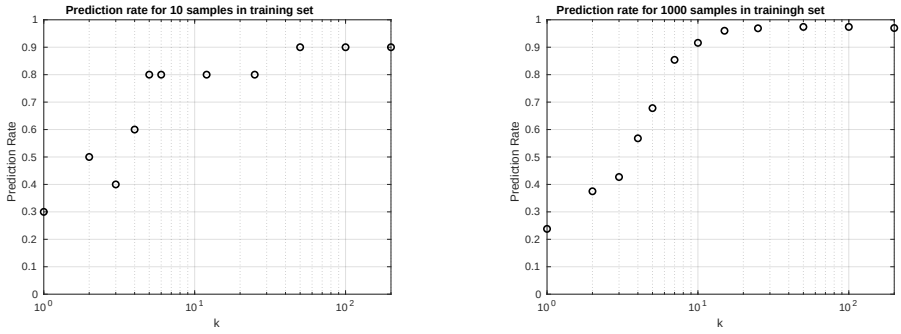
Figure 1.1: Recognition of handwritten numbers with Matlab code of this section using PCA for different number of principal components k.

Programs

1.1 Main Matlab program for recognition of handwritten digits

```
clear
close all
clc

all_examples = 1;
number_examples_train = 31857;
```



$N = 10$ $N = 1000$

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

```
% number 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
```

Computer exercise 5 (3 b.p.).

1.3 Solution of Helmholtz equation

This exercise can be viewed as part of the Master's project "Efficient implementation of Helmholtz equation with applications in medical imaging", see Master's projects homepage for description of this project or go to the link

`http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/1617/BOOK/MasterProject_Helmholtz.pdf`

Solve the Helmholtz equation

$$\begin{aligned} \Delta u(x,\omega) + \omega^2 \varepsilon'(x) u(x,\omega) &= i\omega J, \\ \lim_{|x|\to\infty} u\,(x,\omega) &= 0. \end{aligned} \tag{1.1}$$

in two dimensions using C++/PETSC. Here, $\varepsilon'(x)$ is the spatially distributed complex dielectric function which can be expressed as

$$\varepsilon'(x) = \varepsilon_r(x)\frac{1}{c^2} - i\mu_0\frac{\sigma(x)}{\omega}, \tag{1.2}$$

where $\varepsilon_r(x) = \varepsilon(x)/\varepsilon_0$ and $\sigma(x)$ are the dimensionless relative dielectric permittivity and electric conductivity functions, respectively, ε_0, μ_0 are the permittivity and permeability of the free space, respectively, and $c = 1/\sqrt{\varepsilon_0\mu_0}$ is the speed of light in free space., and ω is the angular frequency.

Take appropriate values for ω, ε', J. For example, take

$$\omega = \{40, 60, 80, 100\}, \varepsilon_r = \{2, 4, 6\}; \sigma = \{5, 0.5, 0.05\}, J = 1.$$

Analyze obtained results for different $\omega, \varepsilon_r, \sigma, J$.
Information about PETSc can be found on the link:
`https://www.mcs.anl.gov/petsc/`

Hints:

1. Study Example 12.5 of the course book [1] where is presented solution of the Dirichlet problem for the Poisson's equation on a unit square using different iterative methods implemented in C++/PETSc. C++/PETSc programs for solution of this problem are available for download from the course homepage: go to the link of the book [1] and click to "GitHub Page with MATLAB® Source Codes" on the bottom of this page or go to the link

   `https://github.com/springer-math/Numerical_Linear_Algebra_Theory_and_Applications`

   Choose then

   `PETSC_code`

   The different iterative methods are encoded by numbers 1-7 in the main program

`Main.cpp`

in the following order:

- 1 - Jacobi's method,
- 2 - Gauss-Seidel method,
- 3 - Successive Overrelaxation method (SOR),
- 4 - Conjugate Gradient method,
- 5 - Conjugate Gradient method (Algorithm 12.13),
- 6 - Preconditioned Conjugate Gradient method,
- 7 - Preconditioned Conjugate Gradient method (Algorithm 12.14).

Methods 1-5 use inbuilt PETSc functions, and methods 6,7 implement algorithms 12.13, 12.14 of the book [1], respectively. For example, we can run the program Main.cpp using SOR method as follows:

```
> nohup Main 3 > result.m
```

After running the results will be printed in the file result.m and can be viewed in Matlab using the command

```
surf(result).
```

2. Modify PETSc code of the Example 12.5 of [1] such that the equation (1.1) can be solved. Note that solution of the equation (1.1) is complex. You should include

```
#include <complex>
```

to be able work with complex numbers in C++. For example, below is example of definition of the complex array in C++ and assigning values to the real and imaginary parts:

```
complex<double> *complex2d = new complex<double>[nno];
 double a = 5.4;
  double b = 3.1;

 for (int i=0; i < nno; i++)
 {
   complex2d[i].real() = a;
   complex2d[i].imag() = b;
 }
   delete[] complex2d;
```

Example of the definition of the complex right hand side in PETSc is presented below:

```
PetscScalar right_hand_side(const PetscReal x, const PetscReal y)
{
  PetscReal realpart, imagpart;
  PetscReal   pi = 3.14159265359;
  realpart = pi*sin(2*pi*x)*cos(2*pi*y);
  imagpart = x*x + y*y;
  PetscScalar f(rpart, ipart);
  return f;
}
```

3. Example of Makefile for running C++/PETSc code at Chalmers is presented in Example 12.5 of [1] and can be as follows:

```
PETSC_ARCH=/chalmers/sw/sup64/petsc-3.7.4

include ${PETSC_ARCH}/lib/petsc/conf/variables
include ${PETSC_ARCH}/lib/petsc/conf/rules

CXX=g++
CXXFLAGS=-Wall -Wextra -g -O0 -c -Iinclude -I${PETSC_ARCH}/include
LD=g++
LFLAGS=

OBJECTS=Main.o CG.o Create.o DiscretePoisson2D.o GaussSeidel.o
 Jacobi.o  PCG.o Solver.o SOR.o
Run=Main

all: $(Run)

$(CXX) $(CXXFLAGS) -o $@ $<

$(Run): $(OBJECTS)
$(LD) $(LFLAGS) $(OBJECTS) $(PETSC_LIB) -o $@
```

To compile PETSc with complex numbers you need to write in Makefile:

```
PETSC_ARCH=/chalmers/sw/sup64/petsc-3.7.4c
```

4. Choose the two-dimensional convex computational domain $\Omega\,\Omega = [0,1]\times[0,1]$. Choose boundary condition at the boundary of $\partial\Omega$ such that the condition $\lim_{|x|\to\infty} u\,(x,\omega) = 0$ is satisfied, for example, take some functions in the form of Gaussian exp^{-x^2}.

5. Choose the following boundary condition $u(x,\omega) = -\omega g(x,\omega)$, where $g(x,\omega)$ is given by (1.4). More precisely, solve the Helmholtz equation

$$\begin{aligned} \Delta u(x,\omega) + \omega^2 \varepsilon(x) u(x,\omega) &= f(x,\omega), \\ u(x,\omega) &= -\omega g(x,\omega). \end{aligned} \tag{1.3}$$

Take as $g(x,\omega), x=(x_1,x_2)$, the function

$$u(x_1,x_2)=\sin(2\pi x_1)\sin(2\pi x_2)+ix_1(1-x_1)x_2(1-x_2) \tag{1.4}$$

which is the exact solution of the equation (1.3) with the right hand side

$$\begin{aligned} f(x_1,x_2)=&-(8\pi^2)\sin(2\pi x_1)\sin(2\pi x_2)-2ix_1(1-x_1)-2ix_2(1-x_2)\\ &+\omega^2\varepsilon(x)(\sin(2\pi x_1)\sin(2\pi x_2)+ix_1(1-x_1)x_2(1-x_2)) \end{aligned} \tag{1.5}$$

6. Try also the following boundary condition $\partial_n u(x,\omega)=-\omega g(x,\omega)$.
7. Values of c,μ_0,ε_0 in (1.2) are known constants.
   - Vacuum permittivity, sometimes called the electric constant ε_0 and measured in F/m (farad per meter):
     $$\varepsilon_0\approx 8.85\cdot 10^{-12}$$
   - The permeability of free space,or the magnetic constant μ_0 measured in H/m (henries per meter):
     $$\mu_0\approx 12.57\cdot 10^{-7}$$
   - The speed of light in a free space is given by formula $c=1/\sqrt{\varepsilon_0\mu_0}$ and is measured in m/c (metres per second):
     $$c\approx 300\ 000\ 000$$

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

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