Exercise 1 (Direction of principal components, 1 point) Below are a number of 2D-data sets. Plot the two principal components.

![Plot of principal components](image)

Exercise 2 (Interpreting principal components, 2 points) A carsharing service runs a survey among 1000 students, who provide information concerning their 1- income, 2- distance they cover by car per month, 3- distance they cover by bike per month, 4- distance they cover by public transport per month, 5- distance they cover by foot per month. Then they run a PCA on the data. Provide answers to the following questions:

- What would it mean if a single eigenvector covered 95% of the total data variance?
- How would you interpret the result if the eigenvector $v_1 = [0, 0, 1, -1, 0]$ covers 90% of the total data variance?
- Why might it be necessary to rescale the data before running PCA in order to obtain a sensible result?

Exercise 3 (Generating samples from a Gaussian distribution, 0.5+0.5+0.5+1+0.5 points) You are given the mean $\mu$ and the covariance matrix $\Sigma$ of a $d$-dimensional normal density $\mathcal{N}(\mu, \Sigma)$ and you want to sample $n$ points from this density. Assuming that $\Sigma$ is positive definite, the following MATLAB code will do this for you:

```matlab
S1 = chol(Sigma); X = repmat(mu,n,1) + randn(n,d)*S1;
```

The command $S1 = chol(Sigma)$ generates an upper triangular matrix $S1$ which satisfies $\Sigma = S1 \cdot S1'$. This decomposition is called the Cholesky decomposition. An alternative method, which also works when $\Sigma$ is only positive semi-definite, is to decompose $\Sigma$ to eigenvectors and eigenvalues by $[V,D] = eig(Sigma)$ and then form $S2$ by $S2 = V \cdot sqrt(D)$. However, the Cholesky decomposition is numerically more stable and computationally faster than eigen decomposition method.

(a) Show that in eigen decomposition, $\Sigma = S2 \cdot S2'$.

(b) Generate $n = 2000$ points in 3 dimensional space from a Gaussian distribution with mean $\mu = [0, 0, 0]$ and Covariance $\Sigma = [2 0 0; 0 1 0; 0 0 4]$. Plot it with `plot3`.

(c) What are the eigenvalues and eigenvectors of the covariance matrix $\Sigma$?
(d) Assume you know eigenvalues and eigenvectors of your covariance matrix:

\[
\Lambda = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}, V = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}.
\]

Generate \( n = 400 \) points in 2 dimensional space from a Gaussian distribution with mean zero and covariance matrix corresponding to these eigenvalues and eigenvectors (\( \Sigma = V\Lambda V' \)). Plot the points and guess the approximate direction of principal components in the figure.

(e) Add the eigenvectors in \( V \) to your plot. Compare your guessed directions with these eigenvectors.

Exercise 4 (PCA, 2+1 points)

(a) Implement PCA in MATLAB. Do it in a three line MATLAB code: Subtract the mean of your data, calculate the covariance matrix \( C \), and find its eigenvalues and eigenvectors using the MATLAB command \([V,D] = \text{eig}(C)\).

(b) To test your code (if you could not solve part (a), you can use the MATLAB command \text{pca} ) generate 500 samples from a Gaussian distribution with mean \( \mu = [1,1] \) and covariance \( \Sigma = [2, -1; -1, 2] \). For generating the points you can either use your code from Exercise 3, or use the MATLAB command \text{normrnd} . Apply your PCA code on this data and compare the result with the eigenvectors of the covariance matrix \( \Sigma \).

Exercise 5 (PCA on USPS data, 1+3 points)

(a) Apply the PCA method on images of digits 5 from USPS dataset (use the training data of the complete dataset — available on the course webpage from Assignment 4). Plot the first and the second principal components as 16x16 grayscale images. You can either use your PCA implementation from Exercise 4 or the MATLAB command \text{pca}.

(b) Choose three images of digits 5 from USPS dataset at random and project them onto 1- the first principal component, 2- the first and the second principal component in \( \mathbb{R}^{256} \) (i.e. as a result you should obtain vectors in the original space — this is View 1 in the notation of the lecture notes). Create a 3 \times 3-subplot (use \text{help subplot} in case you do not know how this works) showing the original images in the first row, the results from 1 in the second row, and the results from 2 in the third row (using \text{imagesc} ).

Exercise 6 (Isomap on USPS data, 1+1+1 points) In this exercise you will implement the Isomap method to embed digits 1,2,3,4 from USPS dataset into \( \mathbb{R}^2 \). The code for building kNN graph and the Isomap algorithm itself is provided on the course web page.

In preparation for the following, load the data from \text{usps_train_complete.mat} (available on the course webpage from Assignment 4). Select 300 examples from each of digits \{1,2,3,4\} and put them in variable \( X \). Put the corresponding labels in \( Y \).

(a) Set the connectivity parameter in the kNN graph to \( k = 10 \) and use the following code to plot the embedding in 2 dimensional space using Isomap. Read the manual of the command \text{scatter} to understand how it works.

\[
A = \text{buildKnnGraph}(X,k);
D = \text{graphallshortestpaths}(A,'\text{Directed}', \text{false});
xy = \text{Isomap}(D,2);
figure;
\text{scatter}(xy(:,1),xy(:,2),10,Y,'\text{filled}');
\]

(b) Play with the parameter \( k \). Describe the effect of the parameter on the embedding.

(c) Project the data onto the first two principal components of PCA in \( \mathbb{R}^2 \) (i.e. as a result you should obtain vectors in \( \mathbb{R}^2 \) — this is View 2 in the notation of the lecture notes). Plot the embedding, again using the command \text{scatter}. You can either use your PCA implementation from Exercise 4 or the MATLAB command \text{pca} to perform PCA.
% Machine Learning Assignment 3

clear all; close all; clc;

Part (a)

% [V,D] = eig(Sigma)

Part (b)

% the Cholesky decomposition is a decomposition of a Hermitian, % positive-definite matrix into the product of a lower triangular matrix % and its conjugate transpose,
% R = chol(A) produces an upper triangular matrix R from % the diagonal and upper triangle of matrix A, satisfying % the equation R'*R=A.
% B = repmat(A,m,n) creates a large matrix B % consisting of an m-by-n tiling of copies of A. % each d dimensional point is a row
d = 3; n = 2000; mu = [0, 0 ,0]; Sigma=[2 0 0;0 1 0;0 0 4]; S1 = chol(Sigma); X = repmat(mu,n,1) + randn(n,d)*S1;

figure(1); plot3(X(:,1), X(:,2), X(:,3), '.'); grid on
Part (c)

\[ [V \ D] = \text{eig}(\Sigma) \]

\[ V \cdot D \cdot V' \] is equal to \( \Sigma \)

\[ V = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

\[ D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} \]

\[ \text{ans} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{bmatrix} \]
Part (d)

\[ A = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}; \% \text{eigenvalues of cov matrix} \]
\[ V = (1/\sqrt{2}) \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}; \% \text{eigen vectors cov matrix} \]
\[ d = 2; \ n = 400; \ mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \]
\[ \text{Sigma} = V^T A V; \]
\[ S1 = \text{chol(Sigma)}; \ X = \text{repmat(mu,n,1) + randn(n,d)*S1}; \]

figure(2); hold all;
plot(V(:,1)); \% 1rst eigenvector
plot(X(:,1),X(:,2), '.' );

Exercise 4

Each row represents a d dimensional point

\[ d = 2; \ n = 500; \ mu = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \text{Sigma} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}; \]
\[ S1 = \text{chol(Sigma)}; \ X = \text{repmat(mu,n,1) + randn(n,d)*S1}; \]
\[ d= \text{size(X,2)}; \]
\[ \text{x\_mean} = \text{mean(X,2)}; \% \text{mean of columns of matrix X (horizontal)} \]
\[ \text{Xc} = X - \text{x\_mean} \times \text{ones(1,d)}; \% \text{centered data matrix} \]
\[ \text{C} = (1/n) \times \text{Xc}' \times \text{Xc}; \% \text{covariance matrix} \]
C = cov(Xc); % built-in command produces the same thing
[V,D] = eig(Sigma)

figure(3); hold all;
plot(V(:,1)); % 1rst eigenvector
plot(V(:,2)); % 2nd eigenvector
plot(X(:,1),X(:,2),'.');
legend('1rst eigvect', '2nd eigenvect')

V =
-0.7071   -0.7071
-0.7071    0.7071

D =
1   0
0   3
EXERCISE 1

Figure 0.1. First 2 components

EXERCISE 2

- Data is spread along this direction mostly.
- Negative correlated.
- Same scale (same units) allow us to compare and interpret properly.

EXERCISE 3

In linear algebra, the Cholesky decomposition or Cholesky factorization is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose.

Solving a linear system. The Cholesky decomposition is mainly used for the numerical solution of linear equations $Ax = b$. If $A$ is symmetric and positive definite, then we can solve $Ax = b$ by first computing the Cholesky decomposition $A = LL^T$, then solving $Ly = b$ for $y$ by forward substitution, and finally solving $L^T x = y$ for $x$ by back substitution. A closely related variant of the classical Cholesky decomposition is the $LDL$ decomposition.

Generating correlated Gaussian numbers. A general way to generate correlated (normal distributed) random numbers with a given correlation matrix $C$ is to generate uncorrelated numbers $R$ and then multiplying them with $L$ the Cholesky matrix of $C$. Suppose $X$ is build of uncorrelated normal random vector with mean zero

$$cov(X, X) = E[XX^T] = \begin{pmatrix} 1 & 0 \\ : & : \\ 0 & 1 \end{pmatrix}$$

and that we want to generated correlated random vector with correlation matrix $C$, that using Cholesky decomposition is written as

$$C = LL^T$$

then $LX$ has the desired covariance given by

**Covariance Matrix.** The covariance matrix (also called the variance-covariance matrix) of an \( n \times 1 \) random vector is an \( n \times n \) matrix whose \( i, j^{th} \) element is the covariance between the \( i^{th} \) and the \( j^{th} \) random variables. Let’s define a matrix of observations where each row is an experiment (or individual) consisting of \( d \) features (dimensions).

\[
X = \begin{bmatrix}
    a_1 & \ldots & a_d \\
    b_1 & \vdots & b_d \\
    n_1 & \ldots & n_d
\end{bmatrix}
\]

so the covariance

\[
\Rightarrow \text{cov}(X, X) = (XX^T)_{n \times n} \quad \text{interindividual}
\]

\[
\Rightarrow \text{cov}(X, X) = (X^TX)_{d \times d} \quad \text{interdimensional}
\]

As we are interested in the information of each of the \( d \) dimensions (features) we proceed with the second.

**Fact.** When the matrix \( X \) is a Hermitian matrix (resp. symmetric matrix), eigenvectors of \( X \) can be chosen to form an orthonormal basis of \( C^n \) (resp. \( R^n \)). Under such circumstance \( P \) will be a unitary matrix (resp. orthogonal matrix) and \( P^{-1} \) equals the conjugate transpose (resp. transpose) of \( P \).

<table>
<thead>
<tr>
<th>Orthogonal</th>
<th>Unitary</th>
</tr>
</thead>
<tbody>
<tr>
<td>( OO^* = I = O^*O )</td>
<td>(UU^* = I = U^*U )</td>
</tr>
<tr>
<td>( O ) is a real square matrix</td>
<td>( U ) is a complex square matrix</td>
</tr>
<tr>
<td>( O ) is the transpose</td>
<td>( U^* ) is the conjugate transpose</td>
</tr>
</tbody>
</table>

**Figure 0.2.** Unitary vs Orthogonal Matrix

Let’s consider the transformation change of basis of a matrix \( X \)

\[
X = \begin{bmatrix}
    u & v \\
    \vdots & \vdots
\end{bmatrix} \begin{pmatrix}
    \lambda & 0 \\
    0 & \mu
\end{pmatrix} \begin{bmatrix}
    u & v \\
    \vdots & \vdots
\end{bmatrix}^{-1}
\]

where \( u, v \) are eigenvectors. For every \( n \times n \) real symmetric matrix (e.g. covariance matrix), the eigenvalues are real and the eigenvectors can be chosen such that they are orthogonal to each other. An Orthogonal matrix is a square matrix with real entries whose columns and rows are orthogonal unit vectors. So the matrix of eigenvectors \( P \) is orthogonal then its inverse \( P^{-1} \) is equal to its transpose \( P^T \)

\[
V^TV = \begin{pmatrix}
    v_1 & \ldots & v_1 \\
    u_1 & \ldots & u_1
\end{pmatrix} \begin{pmatrix}
    v_1 & u_1 \\
    \vdots & \vdots
\end{pmatrix}
\]

\[
= \begin{pmatrix}
    v_1^Tv_1 & v_1^Tu_1 \\
    u_1^Tv_1 & u_1^Tu_1
\end{pmatrix} = 1
\]

taking the transpose

\[
(V^TV)^T = VV^T = 1^T
\]

\[
\Rightarrow V^T = V^{-1}
\]

Thus a real symmetric matrix \( X \) can be decomposed as

\[
X = \begin{bmatrix}
    u & v \\
    \vdots & \vdots
\end{bmatrix} \begin{pmatrix}
    \lambda & 0 \\
    0 & \mu
\end{pmatrix} \begin{bmatrix}
    u & v \\
    \vdots & \vdots
\end{bmatrix}^T
\]

which resembles the Cholesky \( LDL \) decomposition.