Homework 1: Linear Algebra and Probability Review

Spring 2020
(Due: Friday, Jan 24, 2020)

Homework is due at 11:59am. Please put your homework in the dropbox located at MSEE 330. No late homework will be accepted.

Objective

As the first homework assignment, we would like you to refresh some of the concepts in the Background chapter, and have some hands-on experience with Python. Here are the specific objectives.

(a) Familiarize yourself with tools in Python that will be helpful to you in later part of the course. In addition to basic Python functions and objects, you will gain experience working with functions that simulate random data sampling from probability distributions, and visualize the data;

(b) Review of some important concepts in linear algebra and probability. Warm up with some proof techniques that will be used later in the course.

Exercise 1: Installing Python and Getting Started (0 point)

To get started with the homework, please download and install Python on your local machine. Here are a few steps to guide you through. For additional information, e.g., video demonstrations, please visit our course website.

(a) If you are a beginner to Python, we suggest you download Anaconda at `https://www.anaconda.com/download/`. Follow the instruction and install on your local machine.

(b) Once you have installed Anaconda, open an environment and install Spyder.

(c) Make sure you have standard packages installed: `scipy`, `numpy`, `matplotlib`, `cvxpy`, `cvxopt`, and `imageio`.

(d) After you have installed all these packages, open Spyder and type your hello world program.

```
import numpy as np
import scipy
import matplotlib.pyplot as plt
import cvxpy as cp
import csv
import imageio

print("Hello World!")
```

If you are already familiar with Python, you may skip this exercise. Please contact our teaching assistants if you need any help.

Exercise 2: Generating 1D Random Variables

In this exercise, we will use Python to draw random samples from a 1D Gaussian and visualize the data using a the histogram.

(a) Let X be a random variable with $X \sim \mathcal{N}(\mu, \sigma^2)$. The PDF of X is written explicitly as

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{1}$$

Prove that $\mathbb{E}[X] = \mu$ and $\text{Var}[X] = \sigma^2$.

(b) Let $\mu = 0$ and $\sigma = 1$ so that $X \sim \mathcal{N}(0, 1)$. Plot $f_X(x)$ using `matplotlib.pyplot.plot` for the range $x \in [-3, 3]$. Use `matplotlib.pyplot.savefig` to save your figure.

(c) Let us investigate the use of histograms in data visualization.

(i) Use `numpy.random.normal` to draw 1000 random samples from $\mathcal{N}(0, 1)$.

(ii) Make two histogram plots using `matplotlib.pyplot.hist`, with the number of bins m set to 4 and 1000.

(iii) Use `scipy.stats.norm.fit` to estimate the mean and standard deviation of your data. Report the estimated values.

(iv) Plot the fitted gaussian curve on top of the two histogram plots using `scipy.stats.norm.pdf`.

(v) Are the two histograms representative of your data's distribution? How are they different in terms of data representation?

(d) A practical way to estimate the optimal bin width is to make use of what is called the **cross validation estimator of risk** (CVER) of the dataset. Denoting $h = (\text{max data value} - \text{min data value})/m$ as the bin width, with $m =$ the number of bins (assuming you applied no rescaling to your raw data), we seek h^* that minimizes the CVER $\widehat{J}(h)$, expressed as follows:

$$\widehat{J}(h) = \frac{2}{h(n-1)} - \frac{n+1}{h(n-1)} \sum_{j=1}^{m} \widehat{p_j}^{\,2}, \tag{2}$$

where $\{\widehat{p_j}\}_{j=1}^m$ is the empirical probability of a sample falling into each bin, and n is the total number of samples.

Plot $\widehat{J}(h)$ with respect to m the number of bins, for $m = 1, 2, ..., 200$. Find the m^* that minimizes $\widehat{J}(h)$, plot the histogram of your data with that m^*, and plot the Gaussian curve fitted to your data on top of your histogram. How is your current histogram different from those you obtained in part (c)?

Note: If you are interested in why $\widehat{J}(h)$ plays an important role in estimating the optimal bin width of the histogram, see the additional note of this homework.

Exercise 3: Generating 2D Random Variables

In this exercise, we consider the following question: suppose that we are given a random number generator that can only generate zero-mean unit variance Gaussians, i.e., $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, how do we transform the distribution of $\boldsymbol{X}$ to an arbitrary Gaussian distribution? We will first derive a few equations, and then verify them with an empirical example, by drawing samples from the 2D Gaussian, applying the transform to the dataset, and checking if the transformed dataset really takes the form of the desired Gaussian.

(a) Let $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be a 2D Gaussian. The PDF of $\boldsymbol{X}$ is given by

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^2|\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}, \tag{3}$$

where in this exercise we assume

$$\boldsymbol{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}, \quad \text{and} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{4}$$

(i) Simplify the expression $f_{\boldsymbol{X}}(\boldsymbol{x})$ for the particular choices of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ here. Show your derivation.

(ii) Using `matplotlib.pyplot.contour`, plot the contour of $f_{\boldsymbol{X}}(\boldsymbol{x})$ for the range $\boldsymbol{x} \in [-1,5] \times [0,10]$.

(b) Suppose $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. We would like to derive a transformation that can map $\boldsymbol{X}$ to an arbitrary Gaussian.

(i) Let $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ be a d-dimensional random vector. Let $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b} \in \mathbb{R}^d$. Let $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{b}$ be an affine transformation of $\boldsymbol{X}$. Let $\boldsymbol{\mu_Y} \overset{\text{def}}{=} \mathbb{E}[\boldsymbol{Y}]$ be the mean vector and $\boldsymbol{\Sigma_Y} \overset{\text{def}}{=} \mathbb{E}[(\boldsymbol{Y} - \boldsymbol{\mu_Y})(\boldsymbol{Y} - \boldsymbol{\mu_Y})^T]$ be the covariance matrix. Show that

$$\boldsymbol{\mu_Y} = \boldsymbol{b}, \quad \text{and} \quad \boldsymbol{\Sigma_Y} = \boldsymbol{A}\boldsymbol{A}^T. \tag{5}$$

(ii) Show that $\boldsymbol{\Sigma_Y}$ is symmetric positive semi-definite.

(iii) Under what condition on $\boldsymbol{A}$ would $\boldsymbol{\Sigma_Y}$ become a symmetric positive definite matrix?

(iv) Consider a random variable $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu_Y}, \boldsymbol{\Sigma_Y})$ such that

$$\boldsymbol{\mu_Y} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}, \quad \text{and} \quad \boldsymbol{\Sigma_Y} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

Determine $\boldsymbol{A}$ and $\boldsymbol{b}$ which could satisfy Equation (5).

Hint: Consider eigen-decomposition of $\boldsymbol{\Sigma_Y}$. You may compute the eigen-decomposition numerically.

(c) Now let us verify our results from part (b) with an empirical example.

(i) Use `numpy.random.multivariate_normal` to draw 5000 random samples from the 2D standard normal distribution, and make a scatter plot of the data point using `matplotlib.pyplot.scatter`.

(ii) Apply the affine transformation you derived in part (b)(iv) to the data points, and make a scatter plot of the transformed data points. Now check your answer by using the Python function `numpy.linalg.eig` to obtain the trasformation and making a new scatter plot of the transformed data points.

(iii) Do your results from parts (c)(i) and (ii) support your theoretical findings from part (b)? You are welcome to utilize Python functions you find useful and include plots in your answer.

Exercise 4: Norm and Positive Semi-Definiteness

The aim of this exercise is to reinforce your understanding of the vital concepts of norms, the two famous inequalities, eigen-decomposition, and the notion of positive (semi-)definiteness, which will be ubiquitous throughout the semester.

(a) Schur's lemma (one of the several named after Issai Schur) is one of the most commonly used inequalities in estimating quadratic forms. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, vectors $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^n$, the inequality takes the form

$$|\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{y}| \le \sqrt{RC}\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2, \text{ where } R = \max_j \sum_{k=1}^{n} |[\boldsymbol{A}]_{j,k}|, \; C = \max_k \sum_{j=1}^{m} |[\boldsymbol{A}]_{j,k}| \tag{6}$$

Prove this inequality.

Hint: Use the Cauchy-Schwarz inequality.

(b) Recall from the lectures the concepts related to positive (semi-)definite matrices.

   (i) Prove that any positive definite matrix $\boldsymbol{A}$ is invertible.

   (ii) Find a function $f : \mathbb{R}^2 \to \mathbb{R}$ whose Hessian is invertible but not positive definite anywhere in $\mathbb{R}^2$.

   (iii) Under what extra condition is any positive semi-definite matrix positive definite? Justify your answer.

(c) Recall the concept of eigen-decomposition: for **any** symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, there exist a diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ with eigenvalues of $\boldsymbol{A}$ on its diagonal, and orthonormal matrix $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ with eigenvectors of $\boldsymbol{A}$ as its columns, such that $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$. Prove that there exists $\boldsymbol{A}^\dagger \in \mathbb{R}^{n\times n}$ such that the following holds:

$$\boldsymbol{A}\boldsymbol{A}^\dagger\boldsymbol{A} = \boldsymbol{A} \tag{7}$$

Hint: You can use the fact that, for symmetric $\boldsymbol{A}$ with rank $k \leq n$, it is possible to eigen-decompose $\boldsymbol{A}$ such that the first k diagonal entries of $\boldsymbol{\Lambda}$ are nonzero, and the rest are all zeros. Then define $\boldsymbol{A}^\dagger = \boldsymbol{U}\boldsymbol{\Lambda}^-\boldsymbol{U}^T$ where $[\boldsymbol{\Lambda}^-]_{j,j} = [\boldsymbol{\Lambda}]_{j,j}^{-1}$ for $1 \leq j \leq k$, and 0 everywhere else. $\boldsymbol{A}^\dagger$ is what is called the **pseudoinverse** of $\boldsymbol{A}$.