Homework 1

Spring 2024
(Due: Thursday, Jan 25, 2023, on Gradescope)

Please submit your homework through **gradescope**. You can write, scan, type, etc. But for the convenience of grading, please merge everything into a **single PDF**.

Objective

After this homework, you will be familiar with:

(a) Basic plotting tools in Python, for both 1D and 2D plots. Drawing random samples in Python.
(b) Cross-validation. See *Introduction to Probability for Data Science*, Chapter 3.2.5.
(c) Gaussian whitening. See *Introduction to Probability for Data Science* Chapter 5.7.4.
(d) Basic ideas of regression. See *Introduction to Probability for Data Science* Chapter 7.1.4.

You will be asked some of these questions in Quiz 1.

Exercise 1: Histogram and Cross-Validation

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

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

(a) 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.

(b) 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) (Optional) Ask yourself the following questions: Are the two histograms representative of your data's distribution? How are they different in terms of data representation?

(c) A practical way to estimate the optimal bin width is to make use of what is called the **cross validation estimator of risk** of the dataset. Denoting $h =$ (max data value−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 risk $\widehat{J}(h)$, 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.

(i) Plot $\widehat{J}(h)$ with respect to m the number of bins, for $m = 1, 2, ..., 200$.

(ii) Find the m^* that minimizes $\widehat{J}(h)$, plot the histogram of your data with that m^*.

(iii) Plot the Gaussian curve fitted to your data on top of your histogram.

Note: For additional discussions about this cross-validation technique, visit *Introduction to Probability for Data Science*, Chapter 3.2.5. More advanced materials can be found in the supplementary note of this homework.

Exercise 2: Gaussian Whitening

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) Write a Python program using `numpy.linalg.eig` to obtain $\boldsymbol{A}$ given $\boldsymbol{\Sigma_Y}$ in part (b)(iv). Apply the transformation to the data point, and make a scatter plot of the transformed data points to check whether the transformation is correct.

(iii) (Optional) Do your results from parts (c)(i) and (ii) support your theoretical findings from part (b)?

You can find more information about Gaussian whitening in *Introduction to Probability for Data Science*, Chapter 5.7.4.

Exercise 3: Linear Regression

Let us consider a polynomial fitting problem. We assume the following model:

$$y = \beta_0 + \beta_1 L_1(x) + \beta_2 L_2(x) + \ldots + \beta_p L_p(x) + \epsilon, \tag{6}$$

where $L_p(x)$ is the p-th Legendre polynomial, β_j are the coefficients, and ϵ is the error term. In Python, if you have specified a list of values of x, evaluating the Legendre polynomial is quite straight forward:

```
import numpy as np
from scipy.special import eval_legendre
x  = np.linspace(-1,1,50) # 50 points in the interval [-1,1]
L4 = eval_legendre(4,x)   # evaluate the 4th order Legendre polynomial for x
```

(a) Let $\beta_0 = -0.001$, $\beta_1 = 0.01$, $\beta_2 = 0.55$, $\beta_3 = 1.5$, $\beta_4 = 1.2$, and let $\epsilon \sim \text{Gaussian}(0, 0.2^2)$. Generate 50 points of y over the interval `x  = np.linspace(-1,1,50)`. That is,

```
x = np.linspace(-1,1,50) # 50 points in the interval [-1,1]
y = ... # fill this line
```

Scatter plot the data.

(b) Given the $N = 50$ data points, formulate the linear regression problem. Specifically, write down the expression

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{argmin}} \; \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2. \tag{7}$$

What are $\boldsymbol{y}$, $\boldsymbol{X}$, and $\boldsymbol{\beta}$? Derive the optimal solution for this simple regression problem. Express your answer in terms of $\boldsymbol{X}$ and $\boldsymbol{y}$.

(c) Write a Python code to compute the solution. Overlay your predicted curve with the scattered plot. For solving the regression problem, you can call `numpy.linalg.lstsq`.

(d) For the $\boldsymbol{y}$ you have generated, make 5 outlier points using the code below:

```
# ...
idx = [10,16,23,37,45]    # these are the locations of the outliers
y[idx] = 5                # set the outliers to have a value 5
# ...
```

Run your code in (c) again. Comment on the difference.

(e) Consider the optimization

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{argmin}} \; \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_1. \tag{8}$$

Convert the problem into a linear programming problem. Express your solution in the linear programming form:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}. \end{aligned} \tag{9}$$

What are $\boldsymbol{c}$, $\boldsymbol{x}$, $\boldsymbol{A}$, and $\boldsymbol{b}$?

(f) Solve the linear programming problem in Python using `scipy.optimize.linprog`, for the corrupted data in (d). Scatter plot the data, and overlay with your predicted curve. Hint: Remember to set `bounds=(None,None)` when you call `scipy.optimize.linprog`, because the variables in linprog are non-negative by default.

For this problem, you may want to check *Introduction to Probability for Data Science*, Chapter 7.1.4.