Notebook Exercise 2
Due date: 02/28/22

Collaboration Policy. Each student must hand in their own answers. Use of partial or entire solutions obtained from others or online is strictly prohibited. However, discussing the problems with other fellow students or forming study groups is encouraged.

General Instruction: All parts of this exercise must be done within a Notebook, with text answers (and other discussion) provided as markdown / LaTeX cells. Please make sure that the version of the notebook submitted by you has fully executed cells (i.e., submit it after a complete run of all the cells in the notebooks). In addition, submit any files that you are being asked to save as part of the exercise.

Restrictions: You can only use `numpy` and `pandas` packages within your code. Also use `matplotlib.pyplot` for plotting purposes.

Notebook Preamble: You can import the required libraries as follows (but you are allowed to use any other names of your liking):

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

1. *Logistic regression.*

   Let $m = 1000$ and $n = 4$.

   (a) Generate the data matrix $X \in \mathbb{R}^{m \times n}$ with entries drawn independently such that each is distributed as $\mathcal{N}(0, 1)$.

   (b) Let $\underline{\theta} = [1, -1, 2, -5]^T$. Generate labels $\underline{y} \in \mathbb{R}^m$, such that

   $$y_i = \text{sign}(\underline{\theta}^T \underline{x}^{(i)} + 0.5),$$

   where $\text{sign}(x)$ is equal to 1 if $x \geq 0$, and -1 otherwise.

   (c) Use gradient descent (as derived in class) to estimate the logistic regression coefficients for estimating $\text{P}(y = 1|\underline{x})$. Use step size $\alpha = 0.1/m$ and report the coefficients after 1000 iterations. Compare your result with the parameters used for generating the data in part (b).

   (d) Redo the previous part, this time with some noisy data, where now

   $$y_i = \text{sign}(\underline{\theta}^T \underline{x}^{(i)} + Z_i + 0.5),$$

   with $Z_i \sim \mathcal{N}(0, 2)$. The noise Z_i for different samples are drawn independently.

(e) Recall that in logistic regression our function $h(\underline{x})$ is supposed to estimate $\mathrm{P}(Y = 1|\underline{x})$. Therefore, the generated output is a soft decision. Use the following rule to map the output of your logistic regression to $+1$ and -1 values: if $h(\underline{x}) \geq 0.5$, let $\hat{y} = 1$, otherwise $\hat{y} = -1$. Compute the error probability of your classifier on training dataset of part (c). Next compute the error probability corresponding to the dataset you generated for part (d). (Note that for each dataset you need to use a different set of learned coefficients.) Compare the two error probabilities and report your observations.

Remark: Given labels $y^{(i)}$ and predicted labels $\hat{y}^{(i)}$, the error probability on training data is computed as

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{y^{(i)} \neq \hat{y}^{(i)}}.$$

In other words, the error probability shows the fraction of training samples that are wrongly labeled by our algorithm.

2. *Logistic regression using Python.* In this problem we are going to use the same synthetic datasets we used in the previous problem. Let $X \in \mathbb{R}^{m \times n}$ denote the input matrix and $\underline{y}$ and $\underline{y}_{\text{noisy}}$ denote the noise-free and noisy outputs labels, respectively. You need to import LogisticRegression as

```
from sklearn.linear_model import LogisticRegression
```

(a) Define two logistic regression models corresponding to the two datasets. For X and labels $\underline{y}$, the model can be defined as

```
model = LogisticRegression().fit(X.T,Y)
```

(b) Apply `model.predict(X.T,Y)` and `model.score(X.T,Y)` on both models and compare the scores for noisy and noise-free labels.

3. *Non-centered Data and Principal Component Analysis (PCA)*

(a) Create a matrix $A \in \mathbb{R}^{3\times 2}$ whose individual entries are drawn from a Gaussian distribution with mean 0 and variance 1 in an independent and identically distributed (iid) fashion. Also, create a vector $\underline{c} \in \mathbb{R}^3$ whose individual entries are iid and drawn from a Gaussian distribution with mean 0 and variance 3. Once generated, both A and $\underline{c}$ should not be changed for the rest of the problems in this section.

(b) Generate a synthetic dataset with 250 data samples $\{\underline{x}^{(i)}\}_{i=1}^{250}$ as follows. Each data sample $\underline{x}^{(i)} \in \mathbb{R}^3$ in the dataset is generated as $\underline{x}^{(i)} = A\underline{b}_i + \underline{c}$, where $\underline{b}_i \in \mathbb{R}^2$ is a random vector whose entries are iid Gaussian with mean 0 and variance 1. Note that we will have a different $\underline{b}_i$ for each data sample $\underline{x}^{(i)}$ (i.e., unlike A and $\underline{c}$, it is **not** fixed for each data sample). Store the data samples into a *data matrix* $X \in \mathbb{R}^{3\times 250}$, such that each data sample is a column in this data matrix.

(c) What is the rank of data matrix X? Verify this by printing the rank of X.

(d) Verify the importance of centering the data as an essential preprocessing step for PCA by carrying out the following steps:

i. Compute the top two principal component directions $U = [\underline{u}_1, \underline{u}_2]$ of the dataset *without* centering the data. Compute the corresponding reconstruction error as

$$\sum_{i=1}^{m} \|\underline{x}^{(i)} - \hat{\underline{x}}^{(i)}\|_2^2.$$

ii. Center the data and then repeat the previous part and compare the results.