EE 240: Pattern Recognition and Machine Learning
Homework 1

Due date: April 19, 2023

Description: These questions will explore different aspects of Bayes theorem, maximum likelihood estimation, and linear regression.

Reading assignment: AML: Ch. 1 & 5, ESL Ch. 1–2

Homework and lab assignment submission policy:
All homework must be submitted online via https://eLearn.ucr.edu.
Submit your homeworks as a single Python notebook that is free of any typos or errors. Talk to your TA to make sure that the Python version match.
Homework solutions should be written and submitted individually, but discussions among students are encouraged.
All assignments should be submitted by the due date. You will incur 25% penalty for every late day.

H1.1 **Exercise 1.10 in AMLbook**: Here is an experiment that illustrates the difference between a single bin and multiple bins. Run a computer simulation for flipping 1,000 fair coins. Flip each coin independently 10 times. Let's focus on 3 coins as follows. c_1 is the first coin flipped; c_{rand} is a coin you choose at random; c_{min} is the coin that had the minimum frequency of heads (pick the earlier one in case of a tie). Let ν_1, ν_{rand}, and ν_{min} be the fraction of heads you obtain for the respective three coins. For a coin, let μ be its probability of heads. **(10 pts)**

(a) What is μ for the three coins selected?

(b) Repeat this entire experiment a large number of times (e.g., 100,000 runs of the entire experiment) to get several instances of ν_1, ν_{rand}, and ν_{min} and plot the histograms of the distributions of ν_1, ν_{rand}, and ν_{min}. Notice that which coins end up being c_{rand} and c_{min} may differ from one run to another.

(c) Using part 1b plot estimates for $\mathbb{P}[|\nu - \mu| > \epsilon]$ as a function of ϵ, together with the Hoeffding bound $2e^{-2\epsilon^2 N}$ on the same graph.

(d) Which coins obey the Hoeffding bound, and which do not? Explain why.

H1.2 Posterior probability estimation for bin selection problem. (Feel free to write a script for this)

(a) Suppose we have ten bins (four labeled A, six labeled B). Each bin has balls with two colors (red and blue). The distribution of red and blue balls in bin A is (0.3, 0.7). The distribution of red and blue balls in bin B is (0.7, 0.3). We randomly select a bin and draw two balls with *replacement*. That is, we select a bin, pick one ball, put it back, and pick another ball from the *same* bin. Estimate the probability that we selected bin A given the selected balls are red and blue. **(10 pts)**

(b) Suppose we have ten bins (four labeled A, six labeled B). Each bin has balls with four colors (red, blue, white, black). The distribution of balls in bin A is (0.1, 0.3, 0.2, 0.4). The distribution of balls in bin B is (0.4, 0.2, 0.3, 0.1). We randomly select a bin and draw two balls with replacement. Estimate the probability that we selected bin A given the selected balls are red and blue. **(10 pts)**

H1.3 Let us consider the problem of nearest-mean classifier. Suppose we are given N training samples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ from two classes with $y_n \in \{+1, -1\}$. We saw in lecture 2 that we can decide a label for a test vector $\mathbf{x}$ as $g(\mathbf{x}) = \text{sign}\Big(\mathbf{w}^T\mathbf{x} + b\Big)$, where $\mathbf{w} = 2(\mu_+ - \mu_-)$ and $b = \|\mu_-\|_2^2 - \|\mu_+\|_2^2$. μ_+ is a mean vector for samples in the $+ve$ class and μ_- is a mean vector for samples the $-ve$ class. Show that $\mathbf{w}^T\mathbf{x} + b \equiv \sum_{n=1}^{N} \alpha_n \langle \mathbf{x}_n, \mathbf{x} \rangle + b$ and calculate the values of the α_n. **(10 pts)**

H1.4 K-nearest neighbor classification for MNIST data: In this problem set, we consider the problem of handwritten digit recognition. We will use a subset of the MNIST database, which has become a benchmark for testing a wide range of classification algorithms. See http://yann.lecun.com/exdb/mnist/ if you'd like to read more about it. You may want to import MNIST dataset using sklearn:

The sklearn.datasets package is able to directly download data sets from the repository using the function `fetch_openml`.

For example, to download the MNIST digit recognition database:

```
>>> from sklearn.datasets import fetch_openml
>>> mnist = fetch_openml('mnist_784')
```

The MNIST database contains a total of 70000 examples of handwritten digits of size 28x28 pixels, labeled from 0 to 9:

```
>>> mnist.data.shape
(70000, 784)
>>> mnist.target.shape
(70000,)
>>> np.unique(mnist.target)
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.])
```

In the MNIST database, each training or test example is a 28×28 grayscale image. To ease programming of learning algorithms, these images have been converted to vectors of length $28^2 = 784$ by storing the pixels in raster scan (row-by-row) order.

In this question, we explore the performance of K nearest neighbor (K-NN) classifiers at distinguishing handwritten digits. Pick images corresponding to three digits in your student ID, say "1", "2", and "7". To determine neighborhoods, let's use the Euclidean distance between pairs of vector-encoded digits $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{l=1}^{784} [\mathbf{x}_i(l) - \mathbf{x}_j(l)]^2}.$$

(30 pts)

(a) Implement a function that finds the K nearest neighbors of any given test digit, and classifies it according to a majority vote of their class labels. Construct a training set with 200 examples of each class ($N = 600$ total examples). What is the empirical accuracy (fraction of data classified correctly) of 1-NN and 3-NN classifiers on the test examples from these classes?

(b) Plot 5 test digits that are correctly classified by the 1-NN classifier, and 5 which are incorrectly classified. Do you see any patterns?

H1.5 Linear regression: Implement a solution for house price prediction using Python. Data and starter codes in Python can be found at Python linear regression.

You should have a few modules installed:

```
sudo pip install sklearn
sudo pip install scipy
sudo pip install scikit-learn
```

The commands above work in Linux.

We start by loading the modules, and the dataset. Without data we cannot make good predictions.

The first step is to load the dataset. The data will be loaded using Python Pandas, a data analysis module. It will be loaded into a structure known as a Panda Data Frame, which allows for each manipulation of the rows and columns.

(a) Let us first fit the price based on size because we expect to find some correlation between the two. We create two arrays: X (size) and Y (price).The data will be split into a training and test set. We will use the training data to find a best fit line and make predictions on the test data.

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
import pandas as pd

# Load CSV and columns
df = pd.read_csv("Housing.csv")

Y = df['price']
X = df['lotsize']
X = X.values

# Split the data into training/testing sets
X_train = X[:-250]
X_test = X[-250:]

# Split the targets into training/testing sets
Y_train = Y[:-250]
Y_test = Y[-250:]

# Plot outputs
plt.scatter(X_test, Y_test,  color='black')
plt.title('Test Data')
plt.xlabel('Size')
plt.ylabel('Price')
plt.xticks(())
plt.yticks(())

plt.show()
```

Write a code to find the best linear fit $\mathbf{y} = \mathbf{X}\mathbf{w} + b$ by minimizing least-squares cost: $\|\mathbf{X}\mathbf{w} + b - \mathbf{y}\|_2^2$. You can either write the explicit solution or use a gradient descent method, but you must write the code yourself. Recall that the solution of least-squares problem is

$$\begin{bmatrix} \hat{\mathbf{w}} \\ \hat{b} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{X}^T \\ \mathbf{1}^T \end{bmatrix} [\mathbf{X} \;\; \mathbf{1}] \right)^{-1} \begin{bmatrix} \mathbf{X}^T \\ \mathbf{1}^T \end{bmatrix} \mathbf{y},$$

where $\mathbf{1}$ is a vector of all ones.

Compute the predicted price (Y) using the regression coefficients, $\mathbf{w}, b$ and plot the best fit.

```
# Write your code for linear regression (solving least-squares problem)
...

# Prediction on the test data
Y_predict = np.dot(X_test,w) + b

# Plot outputs
plt.plot(X_test, Y_predict, color='red',linewidth=3)
```

(b) Next you will fit the price based on size, number of bedrooms and bathrooms. We create two arrays: X (size, bedrooms, baths) and Y (price).The data will be split into a training and test set as before. We will use the training data to find a best fit "plane" and make predictions on the test data.

```
# Select regression variables
X = df['lotsize', 'bedrooms', 'bathrms']
X = X.values

# fill the remaining lines same as before and write your code for least-squares
...
```

(30 pts)

Maximum points: 100