Assignment (2)

Artificial Intelligence and Machine Learning (24-787)

Due date: 09/27/2022 @ 11:59 pm EST

Note: In case a problem requires programming, it should be programmed in Python. While programming, you should use plain Python, unless otherwise stated. For example, if the intention of a problem is to gain familiarity with the numpy library, it will be clearly noted in that problem to use numpy.

File Submission Structure

1. **HW2 Writeup**: Submit a combined pdf file to Gradescope. This file should contain the answers to the theoretical questions as well as the pdf form of the FILE.ipynb notebooks.

- Convert the jupyter notebook to a pdf file. Ensure that the submitted notebooks have been run and the cell outputs are visible - Hint: ***Restart and Run All*** option in the Kernel menu. Make sure all plots are visible in the pdf. For more details on how to convert a jupyter notebook to a PDF file, please reference this link.
- If an assignment has theoretical and mathematical derivation, you may scan your handwritten solution, or type your solution using Latex or Word.
- Then concatenate them all together in your favorite PDF viewer/editor. The file name should be formatted as **HW-assignmentnumber-andrew-ID.pdf**. For example for assignment 1, your filename should be HW-1-andrewid.pdf
- Submit this final PDF on Gradescope, and **make sure to tag the questions correctly!**

2. **HW2 Code**: Submit a ZIP folder to Gradescope, containing the FILE.ipynb notebooks for each of the programming questions.

- The ZIP folder containing your iPython notebook solutions should be named as **HW-assignmentnumber-andrew-ID.zip**
- You can refer to the numpy documentation while working on this assignment. Any deviations from the submission structure shown below would attract penalty to the assignment score. Please use Piazza for any questions on the assignment.

PROBLEM 1

Probability and Bayes Rule (Theoretical problem) **[15 points]**

We are given a box which contains ten coins in total where three coins have a head on each side, three coins have a tail on each side, and the remaining four coins are fair (a head on a side and a tail on the other side).

a) One of these ten coins is selected randomly and tossed once, what is the probability of getting a head?

b) If we get a tail, what is the probability that the selected coin has a tail on both sides? If we get a tail, what is the probability that it is a fair coin?

c) If the first toss is tail, and another coin is selected at random from the remaining nine coins and tossed once, what is the probability of getting a tail again?

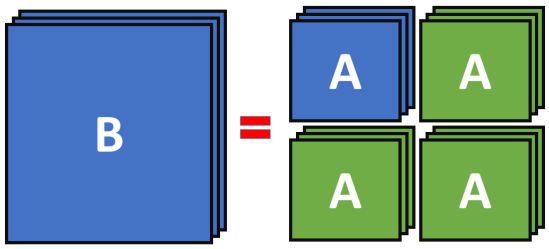

Fig. 1: Construct b matrix by repeating matrix a

PROBLEM 2

Introduction to Numpy **[15 points]**

This question introduces numpy and some of the most powerful features it offers. Remember that array indexing in python starts from 0 unlike MATLAB which starts from 1.

Set the random seed to 24787 using numpy random seed and create a random integer array **a** of size (3,4,4) with values between 0 and 7, both included. To make visualization simple, you can think of the dimensions of a 3d array as (depth, rows, columns). Print **a** and its shape. Locate the locations of all the fours in the array **a** and print out the row and column indices (0 being the first row).

Create an array **b** of size (3,8,8) by repeating matrix **a** using the tile function. Example shown in the fig 1.

Calculate and store the sum of matrix **b** along the depth in **c**. Use standard Numpy functions only, and observe the shape of **c**. Print **c** and its shape.

Set the seed back to 24787 and create two new arrays **a** and **b** of size (1000,1000). Create a function matmul which takes in two matrices as input and performs a matrix multiplication each time taking a dot product of a row from matrix 1 and the corresponding column from matrix 2 for each element in the product matrix. Run your function on the matrices created earlier and print the time taken to calculate the product. Repeat the same experiment, this time using the matrix product operator "@" and time the execution. Show the correctness of your implementation of the product function by comparing with the matrix obtained by the product operator. Also explain in two lines why the inbuilt implementation is faster than the function you wrote.

PROBLEM 3

Topic: Maximum Likelihood Estimate (MLE) **[30 points]**

a) (Theoretical problem)- 10 points: For a two parameter distribution, the probability density function is:

$$f(x) = \theta\sigma^{\theta}x^{-\theta-1} \quad \text{for} \quad x \geq \sigma \tag{1}$$

What is the MLE for σ and θ ? *Express MLE of σ in terms of x_i. It should be rather straightforward.

b) (Theoretical problem)- 10 points: At a tech-giant, the number of devices that failed the durability test among the ten batches are x_i = 13, 2, 16, 5, 11, 16, 18, 5, 8, 15. In this case, the distribution parameters are λ and ξ, and follows log-normal distribution: ln(x) $\sim$ N(μ,σ^2).
Estimate the distribution parameter using MLE. Think of λ as mean and ξ as standard deviation.

c) (Theoretical problem)- 10 points: Let $x_1, ..., x_n$ be a random sample from the distribution with probability density function:

$$f(x;\theta) = \frac{x}{\theta^2}e^{\frac{-x}{\theta}} \quad \text{for} \quad x > 0 \tag{2}$$

where $\theta > 0$ is an unknown parameter.

Find the maximum likelihood estimate (MLE) for θ .

PROBLEM 4

Logistic regression **[40 points]**

The goal of this exercise is to find the weights and biases associated with the logistic regression algorithm that classifies data into two classes (categories). Please use the jupyter notebook code template provided in conjunction with the following text to solve this question. The following convention should be used while answering the questions in the notebook: Input array X, Label array Y, Predicted array $\widehat{Y}$, Parameters θ, Number of Datapoints N, i^{th} element of A being a_i.
You can use **numpy**, **matplotlib** and **pandas** libraries in this problem. **Note:** Make sure to consider and learn the bias parameter in this problem.

- Load up the csv files **class0-input.csv**, **class1-input.csv** and **labels.csv** from the directory **q4-data/** and store them in arrays **X** and **labels** respectively. Check the size of **labels**.
- Visualize the data and implement the functions as directed in the jupyter notebook.
- Calculate the weights by calling the functions you previously defined in the '**main()**' function.
- Then calculate the accuracy of the predicted labels from weights calculated.
- Visualize the misclassification as directed in the jupyter notebook.
- Now use **sklearn**'s logistic regression function to fit a model and write down the accuracy.
- Compare the Accuracy between your Model and the sklearn Model.
