40.319 STATISTICAL AND MACHINE LEARNING
SPRING 2020 HOMEWORK 1

DUE 12 FEB. TOTAL 40 POINTS.

Submit your written/typed solutions as a PDF using Gradescope (entry code `9KRKRW`). Upload your Python code as a file `[student-id].py` using this link.

`https://www.dropbox.com/request/L4Fo929hHLkxKvvqveLP`

0. Slack [1 Participation Point]

We will be using Slack (`sutd-sml-2020.slack.com`) for course announcements and Q&A. Please sign up using this link

`https://join.slack.com/t/sutd-sml-2020/signup`

Write down your Display Name in the homework submission.

1. Linear Algebra Review [10 Points]

Let $\theta := (\theta_1, \ldots, \theta_d) \in \mathbb{R}^d$ be a vector, and $\theta_0 \in \mathbb{R}$ be a scalar. Let the hyperplane $\mathcal{H}$ be the set of all points $x := (x_1, \ldots, x_d) \in \mathbb{R}^d$ such that $0 = \theta^\top x + \theta_0$, where

$$\theta^\top x = \theta_1 x_1 + \cdots + \theta_d x_d$$

is the dot product. The goal is to find the shortest distance between $\mathcal{H}$ and a point $y \in \mathbb{R}^d$. There are many ways to solve this problem, but we will be using Lagrange multipliers to familiarize ourselves with this powerful method.

Let $\tilde{x}$ be the point on $\mathcal{H}$ that is closest to y. Then $\tilde{x}$ solves the optimization problem

$$\begin{aligned} \underset{x \in \mathbb{R}^d}{\text{minimize}} \quad & (x-y)^\top (x-y) \\ \text{subject to} \quad & \theta^\top x + \theta_0 = 0. \end{aligned}$$

The Lagrangian for this optimization problem is

$$L(x, \lambda) = (x-y)^\top (x-y) + \lambda(\theta^\top x + \theta_0)$$

where λ is the Lagrange multiplier.

Date: January 26, 2020.

1.1. Write down the derivatives of $L(x, \lambda)$ with respect to $x_1, \ldots, x_d$ and λ.

1.2. Equate the derivatives to zero, and solve the equations to find $\tilde{x}$.

1.3. Use $\tilde{x}$ to find the distance of y to the hyperplane $\mathcal{H}$.

2. Probability Review [10 Points]

Let X and Y be independent Poisson random variables, i.e.

$$\mathbb{P}(X = x) = \frac{\alpha^x e^{-\alpha}}{x!}, \quad \mathbb{P}(Y = y) = \frac{\beta^y e^{-\beta}}{y!}, \quad \text{for all } x, y \geq 0.$$

for some rates $\alpha, \beta > 0$. Let the random variable $Z = X + Y$ be their sum.

2.1. Write $\mathbb{P}(Z = z)$ as a sum of products of $\mathbb{P}(X = x)$ and $\mathbb{P}(Y = y)$.

2.2. Show that Z is also Poisson, and find its rate γ.

3. Linear Regression [20 Points]

We will use PyTorch to perform linear regression using gradient descent. Import the Boston housing data from the following link.

`https://www.dropbox.com/s/uy0upqyisinak2b/boston.csv?dl=0`

We will train a linear model that predicts the prices of houses `MEDV` using three inputs:

(i) average number of rooms per dwelling `RM`;
(ii) index of accessibility to radial highways `RAD`;
(iii) per capita crime rate by town `CRIM`.

You can access the selected inputs and target variables using the following code:

```
import matplotlib.pyplot as plt
import numpy
csv = 'boston.csv'
data = numpy.genfromtxt(csv,delimiter=',')
```

The data contains 506 observations on housing prices in suburban Boston. The first three columns are the inputs `RM`, `RAD` and `CRIM`. The last column is the target `MEDV`.

Convert the data to PyTorch tensors using the following code.

```
import torch
inputs = data[:, [0,1,2]]
inputs = inputs.astype(numpy.float32)
inputs = torch.from_numpy(inputs)
target = data[:,3]
target = target.astype(numpy.float32)
target = torch.from_numpy(target)
```

3.1. Write the code to generate (random) weights $w_{\texttt{RM}}, w_{\texttt{RAD}}, w_{\texttt{CRIM}}$ and bias b. After that, write a function to compute the linear model.

3.2. Write a function that computes the mean squared error (MSE).

3.3. Complete the loop below to update the weights and bias using a fixed learning rate (try different values from 0.01 to 0.0001) over 200 iterations/epochs.

```
for i in range(200):
  print("Epoch", i, ":")

  # compute the model predictions
  # compute the loss and its gradient

  print("Loss=", loss)

  with torch.no_grad():

    # update the weights
    # update the bias

    w.grad.zero_()
    b.grad.zero_()
```

(We use `w.grad.zero_()` and `b.grad.zero_()` to reset the gradients to zero because PyTorch accumulates gradients.)

3.4. Use the `matplotlib` library to plot the MSE against the number of iterations. Print the output to the PDF file that you are submitting on Gradescope.

For this problem, DO NOT use the in-built functions for the loss or the linear model in the torch library. Upload the final script as a file named `[student-id].py` using the Dropbox link at the start of this assignment.