Statistical and Machine Learning (01.113)
Homework 1

DUE ON 12 Feb (in class)

1 Problem 1

Let $\theta \in \mathbb{R}^d$ be a fixed vector and θ_0 be a constant. Let $x \in \mathbb{R}^d$ be variable. Consider the hyperplane in $\mathbb{R}^d$ whose equation is given by $\langle \theta, x \rangle + \theta_0 = 0$. Given a point $y \in \mathbb{R}^d$, find the shortest distance from y to the hyperplane.
Hint: Normalize θ, i.e. let $n = \frac{\theta}{\|\theta\|}$ and rewrite the equation of the hyperplane in terms of n.

2 Problem 2

A continuous random variable X is said to have the *standard normal distribution*, with mean $\mu = 0$ and variance $\sigma^2 = 1$, that is $X \sim N(0, 1)$, if it has a probability density function (pdf) defined by

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}, \qquad x \in \mathbb{R}.$$

Prove that

$$\int_{\mathbb{R}} f_X(x)\, \mathrm{d}x = 1.$$

Hint: Let $I = \int_{\mathbb{R}} e^{\frac{-x^2}{2}}\, \mathrm{d}x$. Express I^2 as a double integral over $\mathbb{R}^2$ and convert to polar coordinates.

3 Problem 3

Let X and Y be random variables with a joint normal distribution such that $\mathbb{E}[X] = 0 = \mathbb{E}[Y]$, $\mathbb{E}\left[X^2\right] = 1 = \mathbb{E}\left[Y^2\right]$, and the covariance $\mathbb{E}[XY] = \rho$ where $0 < |\rho| < 1$.

(a) Write down the joint probability distribution $p(x, y)$ of X and Y.

(b) Let B denote the inverse of the covariance matrix of $[X, Y]^T$. Perform the decomposition $B = PDP^{-1}$, where D is a diagonal matrix and P is an orthogonal matrix.

(c) Use the result above to transform $(x, y) \to (u, v)$ such that under the new coordinates, the joint distribution can be factorized; i.e. $q(u, v) = q_1(u)q_2(v)$.

Hint: For (b), compute the eigenvalues and eigenvectors of B. For (c), recall the definition of the adjoint of a matrix.

4 Problem 4

We will now use PyTorch to perform linear regression using gradient descent. Import the Boston data from *sklearn* datasets to generate 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    #To generate the plots of question (d)
import numpy
csv = 'https://www.dropbox.com/s/0rjqoaygjbk3sp8/boston_house_prices_3features.txt?dl=1'
data = numpy.genfromtxt(csv,delimiter=',', skip_header=1)
```

The data contains 506 observations on housing prices for Boston suburbs. The first three columns corresponds to the inputs RM, RAD and $CRIM$, respectively. The last column is the target $MEDV$.

Import PyTorch and format the data as follows:

```
import torch
# Convert inputs and target to tensors
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)
```

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

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

(c) 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_()
```

(Note that we reset the gradients to zero by using *w.grad.zero_()* and *b.grad.zero_()* because PyTorch accumulates gradients.)

(d) Use the matplotlib library to plot the evolution of MSE at every iteration.

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 in your Dropbox folder and name it as "**HW1.py**".