Dr. J. Hossain

Assignment 1

Due Date: Aug 8, 2025 – 11:59pm

Instructions

1. Read problems carefully
2. You can NOT get help from other or online sources
3. You can use python for your calculation if needed. Add your python codes, output of your codes, and figures (if you are asked to plot) with your solution
4. Submit your PDF solution to TA

Problems

P-1: [Marks: 5]: Consider hyperplane $h(w) = a + b^T w$, and a starting point w^0. Find $w^1 = w^0 + d$ where $\|d\|_2 = 1$, such that $h(w^1)$ is minimum.

P-2: [Marks: 5]: Consider the function $f(x) = x_1^2 + x_2$.

a) Express the Hessian of the function.

b) Is the function convex? Prove your answer.

P-3 [Marks: 9]: Distance travelled in free fall is given by $d = \frac{g}{2}t^2$ where g is the acceleration of gravity, d denotes the distance, and t denotes the time. You are planning to estimate g using regression. Someone has designed an experiment to measure the distance travelled in free fall and to record corresponding time. He/she has tried with four different values of distances and recorded t^2 for each trial (three trials for each distance) from his/her experiment as listed below:

d (cm)	100	100	100	127	127	127	152	152	152	178	178	178
t^2 (s^2)	0.36	0.38	0.46	0.46	0.49	0.51	0.50	0.53	0.56	0.55	0.58	0.61

1) You are going to design regression model to fit a line through the given data. Write python code to make a scatter plot of the given data. You need to plot t^2 in y-axis and distance d in x-axis. (**Marks: 2**)
2) Calculate the parameters (slope and intercept) that minimize the mean square error (MSE). Print these parameters' values with four digits after decimal point. **Marks: 2**
3) From the calculated optimal slope in 2), estimate g [hint: $d = \frac{g}{2}t^2$]. Compare with the typical value of g the acceleration of gravity. **Marks: 5**

P-4 [Marks: 8]: Say you were given with *P*=20 data points. You used polynomial regression of the form $w_0 + w_1x_1 + w_2x_1^2 + \ldots + w_Dx_1^D = y$ to model the relationship between input and output. At first, you used polynomial of order *D=2*. The parameters that minimize mean square error (MSE) for this polynomial regression are w_0=5, w_1=1.5, and w_2=0.03. Then you used polynomial order of *D=3*. The parameters for this polynomial regression are w_0=2, w_1=0.5, w_2=0.01, and w_3=-0.001 that minimize MSE. You are also given 4 data points to validate and compare your models' results in the following table.

Input feature x_1	Output label y_1
4	10
8	20
9	25
14	35

1. Calculate MSE for the given data set for both of your regression models (models with *D=2* and *D=3*) (**Marks: 4**)
2. Compare your models' validation MSEs and make a conclusion which one of these models (model with *D=2* or model with *D=3*) is more appropriate for your regression problem. (**Marks: 4**)

P-5 (Marks: 9)

Python code is posted below. In this problem, you will design hyper parameter which is a learning rate (α). Consider a function $f(w_0) = {w_0}^4 - 5{w_0}^2 - 3w_0$. You plan to find the minimum for this function using gradient descent algorithm. In this problem, you will investigate the effect of starting point for w_0 for gradient descent iteration. Assume your initial guess/starting point $w_0 = -2.0$.

1. Calculate w_0 for next two iterations assuming step size $\alpha = 0.1$. Show all steps. (**Marks: 2**)
2. Calculate the optimal w_0 using gradient descent for a given learning rate. Try different values of α e.g., 0.2, 0.1, 0.01, 0.001 (**assume initial value of** $w_0 = -2.0$). (**Marks: 2**)
    a. Report the optimal value of w_0 for each α (if there is no optimal value is obtained for a given α, you can mention "no optimal value is obtained") and the minimum value of the function for each α using the gradient descent algorithm. (**Marks: 2**)
    b. For your answer in a (previous question), did you obtain the **global optimal** value of w_0? If not, mention and show how would you get the global optimal value of w_0? (**Marks: 3**)

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


# Function and its derivative
f = lambda v: v**4 - 5*v**2 - 3*v
f_derivative = lambda w: 4*w**3 - 10*w - 3

max_no_iteration = 10000
alpha = 0.2
epsilon = 0.001
w = np.zeros(max_no_iteration)
w[0] = 0
```

```
# Gradient Descent
for i in range(1, max_no_iteration):
    w[i] = w[i-1] - alpha * f_derivative(w[i-1])
    if abs(f_derivative(w[i-1])) < epsilon:
        print(f'Convergence after {i} iterations.')
        print(f'The value of v at convergence is approximately: {w[i]:.10f}')
        break
```