For question 1-3, please submit a PDF file on Canvas.
For question 4 (programming question), please submit an .ipynb file via Canvas.

1. [*6 points*] Prove Bayes' Theorem. Briefly explain why it is useful for machine learning problems.

2. [10 points] In Module 2, we gave the normal equation (i.e., closed-form solution) for linear regression using MSE as the cost function. **Prove that the closed-form solution for Ridge Regression** is $\boldsymbol{w} = (\lambda I + X^T \cdot X)^{-1} \cdot X^T \cdot \boldsymbol{y}$, where I is the identity matrix, $X = (x^{(1)}, x^{(2)}, \dots, x^{(m)})^T$ is the input data matrix, $x^{(i)} = (1, x_1, x_2, \dots, x_n)$ is the i-th data sample, and $\boldsymbol{y} = (y^{(1)}, y^{(2)}, \dots, y^{m})$. Assume the hypothesis function $h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$, and $y^{(j)}$ is the measurement of $h_w(x)$ for the j-th training sample. The cost function of the Ridge Regression is $E(\boldsymbol{w}) = \sum_{i=1}^{m}(\boldsymbol{w}^T \cdot \boldsymbol{x}^{(i)} - y^{(i)})^2 + \lambda \sum_{i=1}^{m} w_i^2$.

3. [10 points] Assume we have K different classes in a multi-class Softmax Regression model. The posterior probability is $\hat{p}_k = \delta(s_k(x))_k = \frac{\exp(s_k(x))}{\sum_{j=1}^{K} \exp(s_j(x))}$ for $k = 1, 2, \dots, K$, where $s_k(x) = {\theta_k}^T \cdot x$, input x is an n-dimension vector, and K the total number of classes.
   1) To learn this Softmax Regression model, how many parameters we need to estimate? What are these parameters?
   2) Consider the cross-entropy cost function $J(\Theta)$ of m training samples $\{(x_i, y_i)\}_{i=1,2,\dots,m}$ as below. Derive the gradient of $J(\Theta)$ regarding to θ_k.

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} {y_k}^{(i)} \log({\hat{p}_k}^{(i)})$$

   where ${y_k}^{(i)}$ = 1 if the ith instance belongs to class k; 0 otherwise.

4. [44 *points*] Write a program to find the coefficients for a linear regression model for the dataset provided (data2.txt). Assume a linear model: $y = w_0 + w_1 * x$. You need to

   1) Plot the data (i.e., x-axis for the 1st column, y-axis for the 2nd column),

   and use Python to implement the following methods to find the coefficients:

   2) Normal equation, and

   3) Gradient Descent using **batch** AND **stochastic** modes respectively:
      a) Split dataset into 80% for training and 20% for testing.
      b) Plot MSE vs. iteration of each mode for both training set and testing set; compare batch and stochastic modes in terms of accuracy (of testing set) and speed of convergence. (You need to determine an appropriate termination condition, e.g., when cost function is less than a threshold, and/or after a given number of iterations.)
      c) Plot MSE vs. learning rate (using 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1) and determine the best learning rate.

Please implement the algorithms by yourself and **do NOT use the fit() function** of the library.