Data driven business analytics
MGT-302
Homework 1

- This assignment can be solved individually or in groups of two students. You should mention the name of your partner in your report file. Note that both of the students in a group will get the same grade.
- Deadline: Monday 29 March 2022, 23:59 (No late submissions will be accepted)
- Upload a single zip file on Moodle containing your report and code. You can use any programming language.

1 Linear Regression [40 pts]

A scientist studying weight fluctuation has collected data from some people. She is specifically interested in studying the influence of some features on the weight fluctuations of people. She collects data $\{x^{(i)}, y^{(i)}\}_{i=1}^{m}$, where $y^{(i)}$ is a continuous variable indicating the weight change, and $x^{(i)} = (x_{i1}, x_{i2}, \ldots, x_{id})$ are the variables which record different properties of people. These features are weight, height, age, amount of exercise time per day, and incoming calories per day. She also proceeds to collect new data to predict the weight change by using sound statistical methods. The dataset contains both the training and testing data. **Note that you need to normalize the features.**

Assume that sample size is $m = 200$ and number of features is $d = 5$. We model the problem as linear regression with cost function

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2, \tag{1}$$

where $h_\Theta(x^{(i)}) = \Theta_0 + \Theta_1 x_1^{(i)} + \cdots + \Theta_d x_d^{(i)}$.

1. **Gradient Descent (GD)**
   (a) State the gradient of your cost function.
   (b) Implement the gradient descent method using your derived gradient to minimize the cost function. Fit the model (θ) using training data (weighttrain.mat), and report the learned model and the cost function for the learned model. Try to tune step-size α of GD to reach the fastest converging rate of decreasing cost function in time.
   Important notes: For this question, at each iteration, do not update all the Θ_is simultaneously, instead update them one by one. Also, set the initial value to zero vector. Moreover, make sure you do enough iterations so that your model converges.
   (c) Plot cost function in number of iterations.
   (d) Use your learned model for test data (weighttest.mat) and report the cost function for it.
2. **Stochastic Gradient Descent (SGD)**

(a) Briefly state SGD. Then repeat parts (b),(c),(d) for it. Then compare the two plots of part (b) for GD and SGD. Which algorithm converges faster?

(b) In this part, the step-size of SGD is choosen as a decreasing sequence $\alpha_t = \frac{b}{1+t}$. Compare the convergence rate of SGD with constant step-size and adaptive step-size. (You should also tune b to get the fastest converging rate of SGD with adaptive step-size.)

2 K-means Clustering [35 pts]

Recall that in the K-means clustering problem, we are given a training set $\{x^{(1)}, ..., x^{(m)}\}$, and want to group the data into a few cohesive clusters. Here, we are given feature vectors for each data point $x^{(i)} \in \mathbb{R}^n$ as usual; but no labels $y^{(i)}$ (making this an unsupervised learning problem). Our goal is to predict k centroids and a label $c^{(i)}$ for each data point.

1. Implement K-means algorithm using "kmeans.mat" dataset and visualize the output for $K = 4$. Your code should work for any number of clusters (not just $K = 4$).

2. Try different values for K. Also, try different values of initial condition. The algorithm must converge for all possible initial conditions, otherwise, there is a problem in your implementation. Observe the cost function value as K increases and explain it briefly.

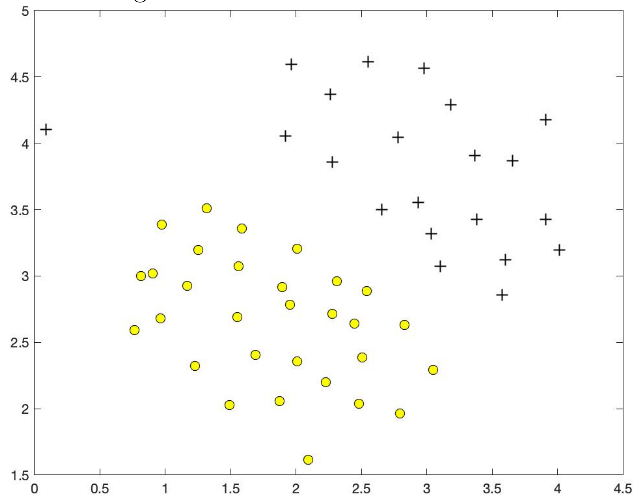
3 Support Vector Machine (SVM) [25 pts]

In this part you will see how SVM works (you are given the required implementations in MATLAB). For SVM hypothesis we use the following notation (linear kernel):

$$\min_{\theta} C \sum_{i=1}^{m} [y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)})] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 \tag{2}$$

You are given a 2D example dataset (data1.mat) which can be separated by a linear boundary (Fig. 1).

Figure 1: Visualization of data1.mat



In this part, you will try to use different values of the C parameter with SVMs. For a wide range of C (e.g. 1,10,50,100), use linear kernel for SVM to classify this data.

1. Plot the decision boundary for each choice of C.
2. Discuss what is the effect of C.
3. What value of C seems to be good for this dataset?

You do not need to implement the SVM classifier yourself as it requires some technical details that were not covered in the class. A MATLAB function (svmTrain.m) is given to you. You can also use other functions if you want (for example functions available in the MATLAB Statistics and Machine Learning Toolbox for efficiently training SVM's). If you are going to use "svmTrain.m", you can call it as follows:

model = svmTrain(X, y, C, @linearKernel, 1e-3, 20);

model is an object with this information:
model.X, model.y, model.kernelFunction, model.b, model.alphas, model.w

To make it even easier for you, you are given two additional functions:

plotData.m: It plots the data.

plotData(X, y);

visualizeBoundaryLinear.m: It plots the linear boundary of a learned model.

visualizeBoundaryLinear(X, y, model);

Implementation Note: Most SVM software packages (including svmTrain.m) automatically add the extra feature $x_0 = 1$ for you and automatically take care of learning the intercept term θ_0 . So when passing your training data to the SVM software, there is no need to add this extra feature $x_0 = 1$ yourself.