Homework 1 Report

Ji Won Min, Victoria Barenne

March 29, 2021

1 Linear Regression

We were given a dataset describing the effect of five features, $x^{(i)} = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5})^T$, each corresponding to weight, height, age, exercise time per day and calories consumed per day on weight changes, $y^{(i)}$ of people. The sample size of the training dataset is $m = 200$. The aim of this section is to implement the Gradient Descent algorithm and the Stochastic Gradient Descent algorithm in order to minimize the cost function, given by:

$$J(\Theta) = \frac{1}{400}\sum_{i=1}^{200}(h_\Theta(x^{(i)}) - y^{(i)})^2, \tag{1}$$

where $h_\Theta(x^{(i)}) = \Theta_0 + \Theta_1 x_1^{(i)} + \cdots + \Theta_5 x_5^{(i)} = \Theta^T x^{(i)}$ is the linear regression function of the data.

1.1 Gradient Descent (GD)

1.1.1 Gradient of the cost function

The gradient of the cost function is found by taking its partial derivatives:

$$\frac{\partial}{\partial \Theta_j} J(\Theta) = \frac{1}{200}\sum_{i=1}^{200}(h_\Theta(x^{(i)}) - y^{(i)}) \cdot \frac{\partial}{\partial \Theta_j} h_\Theta(x^{(i)}) = \frac{1}{200}\sum_{i=1}^{200}(\Theta^T x^{(i)} - y^{(i)}) \cdot x_j^{(i)}. \tag{2}$$

Therefore, in matrix-vector notation, we can write the following:

$$\nabla J(\Theta) = \frac{1}{200}(X^T X\Theta - X^T Y), \tag{3}$$

where $(X^T X\Theta)_j = \sum_{i=1}^{200} \Theta^T x^{(i)} x_j^{(i)}$ and $(X^T Y)_j = \sum_{i=1}^{200} y^{(i)} x_j^{(i)}$.

1.1.2 Implementing the gradient descent method

The following is the equation for the gradient descent algorithm:

$$\Theta_j := \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta), \quad \forall j = 0, 1, ..., n, \tag{4}$$

where α is the learning rate of the algorithm. The algorithm was used in MATLAB to minimize the cost function. The learning rate used was $\alpha = 0.1$. This is important because if the set learning rate too small, convergence will be slow and if it is set too high, is may not converge at all. However, we did not deal with this during the course, so it was set at random here. Also, 200 iterations of the algorithm were used.
After the implementation of the gradient descent method, and the fitting of the model with the training set (weighttrain.mat), the theta parameters were given by:

Θ_0	Θ_1	Θ_2	Θ_3	Θ_4	Θ_5
22.298	4.688	2.312	-1.423	5.995	-8.250

Table 1: Theta parameter values of linear regression of the train data (rounded to 2 decimal points)

Each theta parameter describes the effect of each feature on the weight change.

1.1.3 Plot of cost function against number of iterations

See figure 1 for the plot of cost function in number of iterations. We see that the cost function converges to zero as the number of iterations increases.

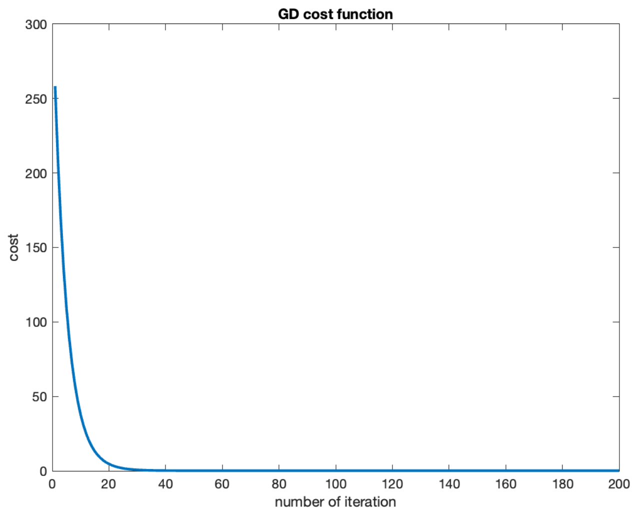

Figure 1: Plot of cost function against number of iterations in the Gradient Descent algorithm.

1.1.4 Test data

Using the learned model on the test data, i.e. using the parameter Θ found in Section 1.1.2, we get the following value for the cost function: $J(\Theta) = 0.3124$

1.2 Stochastic Gradient Descent (SGD)

1.2.1 Implementing the Stochastic Gradient Descent method

The following is the algorithm for the stochastic gradient descent method:

Algorithm 1: Stochastic Gradient Descent

```
Initialize $\Theta \in \mathbb{R}^d$ and the learning rate $\alpha$
Repeat until convergence:
for every $i = 1, \cdots, m$ at random do
  for every $j = 0, \cdots, d$ do
    $\Theta_j = \Theta_j + \alpha(h_\Theta(x^{(i)}) - y^{(i)} \cdot x_j^{(i)}$
  end for
end for
```

1.2.2 Plot of the cost function against number of iterations

With implementation of the SGD method in MATLAB with the train data, the same theta parameters as the ones from the GD method were found.
Moreover, Figure 2 shows the cost function plot against the number of iterations using the Stochastic Gradient Descent. The same $\alpha = 0.1$ and number of iterations (200) were used from the previous gradient method to compare the two algorithms.

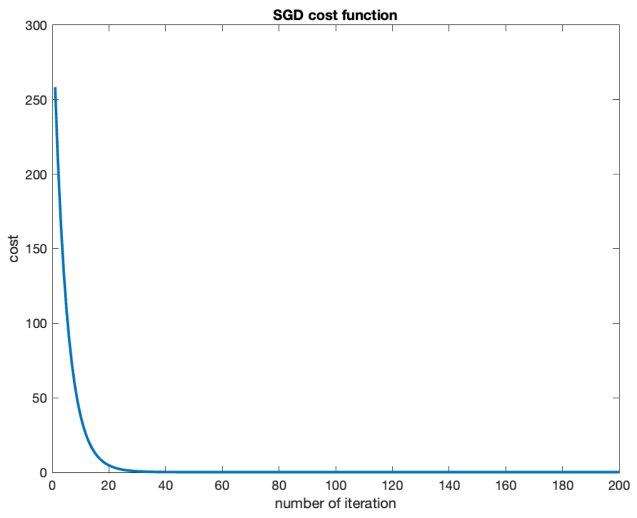

Figure 2: Plot of cost function against number of iterations in the Stochastic Gradient Descent algorithm.

1.2.3 Coefficients and test data

The Stochastic Gradient Descent algorithm converges to the same coefficients as those found with the Gradient Descent algorithm. Thus, the value of the cost function is the same: $J(\Theta) = 0.3124$

1.3 Comparison of the cost function plots for GD and SGD methods

The cost function plots for the GD and SGD methods seem almost the same. We saw previously that the theta values as well as cost values for both methods were the same, which explains the similarity in the graphs. A slight difference in the plots might be because of the rate of the convergence of the two methods, which is further explained in the next section.

1.4 Convergence of Gradient Descent and Stochastic Gradient Descent

To check which method converged faster, we used the tic toc function on MATLAB, which we averaged for 100 values and found that for 200 iterations:

- GD converged in $t = 0.1174$ seconds
- SGD converged in $t = 0.1016$ seconds

Therefore, we can conclude SGD algorithm converges faster than the GD algorithm, which follows with the course of MGT-302: Data Driven Business Analytics.

2 K-means clustering

Our goal of this section is to use the k-means clustering method to predict k centroids and a label $c^{(i)}$ for each given data point.

Algorithm 2: k-means clustering algorithm (from MGT-302 course)

```
Initializing cluster centroids randomly, $\mu_1, \mu_2, ...\mu_k \in \mathbb{R}^d$
for every $i$ do
    $C^{(i)} = argmin_j \|x^{(i)} - \mu_j\|^2$
end for
for every $j$ do
    $\mu_j = \frac{\sum_{i=1}^{m} 1_{C^{(i)}=j} x^{(i)}}{\sum_{i=1}^{m} 1_{C^{(i)}=j}}$
end for
```

The k-means clustering algorithm was written in MATLAB code. See Figure 3 for the k-means clustering of the given data (k-means.mat) for $k = 2$.

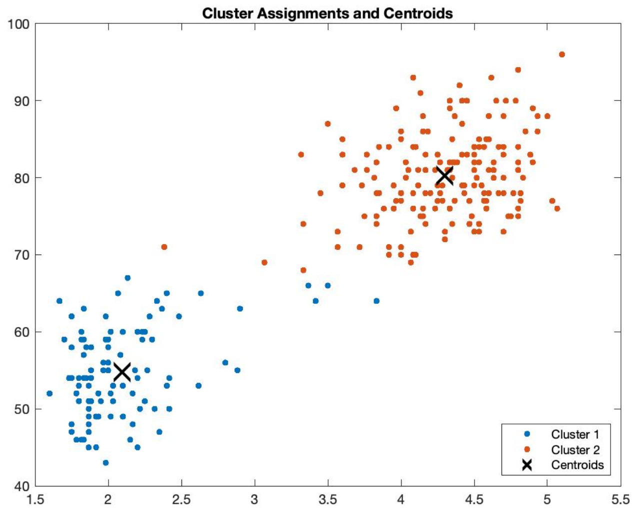

Figure 3: K-means clustering of the data for k = 2.

Figure 4 shows the k-means clustering algorithm implemented to the data for $k = 3, 4, 5$ and 6.

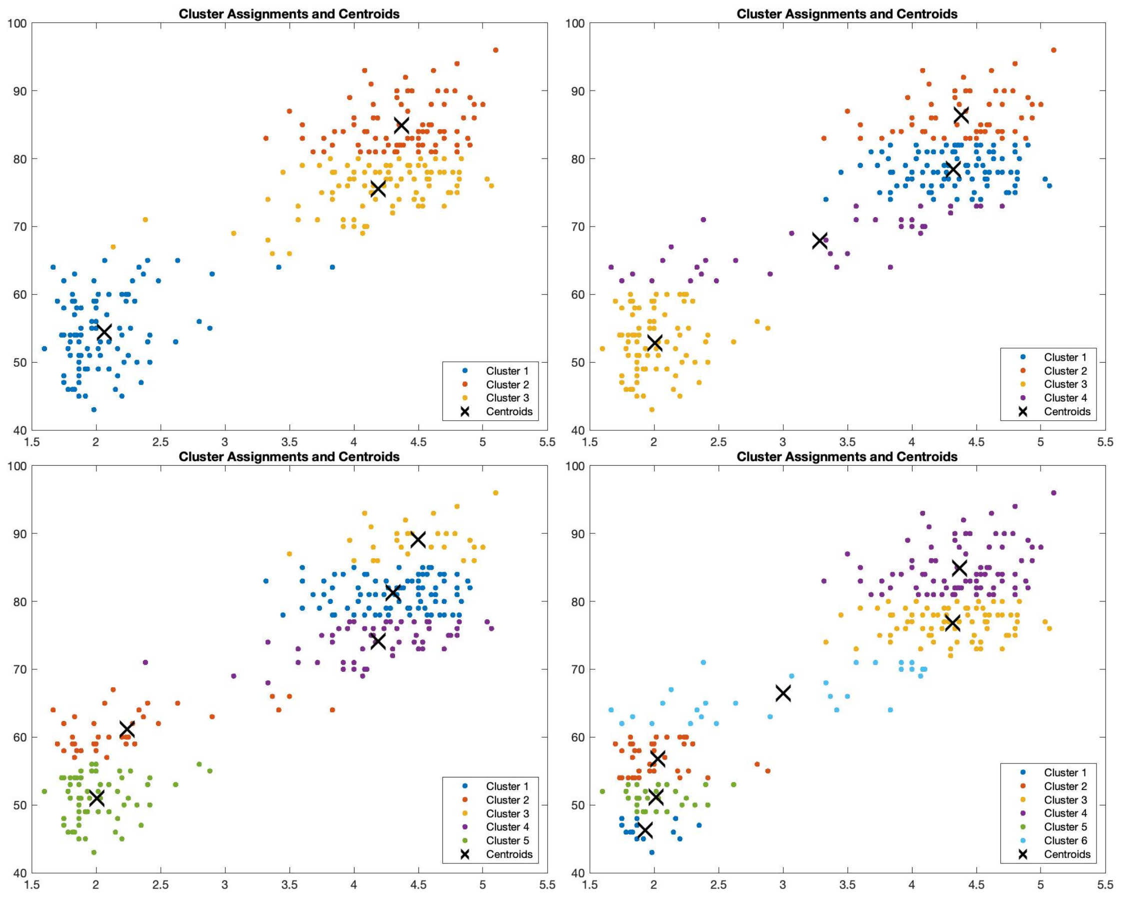

Figure 4: From left to right, top to bottom: (a) $k =$ (b) $k = 4$ (c) $k = 5$ (d) $k = 6$

The cost function, computed by calculating square root the sum of the squared of the 2-norm between each of the data points with its assigned centroid :

$$cost = \sqrt{\sum_{i=1}^{n} \|x^{(i)} - \mu_i\|^2} \tag{5}$$

where n is the number of points, and μ_i is the assigned centroid for each point $x^{(i)}$.

k	Cost
2	94.34
3	76.41
4	64.82
5	45.56
6	41.09

Table 2: Cost value for each k value for k = 2,3,4,5 and 6 (rounded to 2 decimals)

We can see that as k value increases, the cost value decreases. This is because when the number of clusters increases by one, some of the distances between the points and the centroids decreases (some of the points will form a new cluster that has smaller distance to their closest centroid).

3 Support Vector Machine

3.1 Plot of the decision boundary for different choices of C

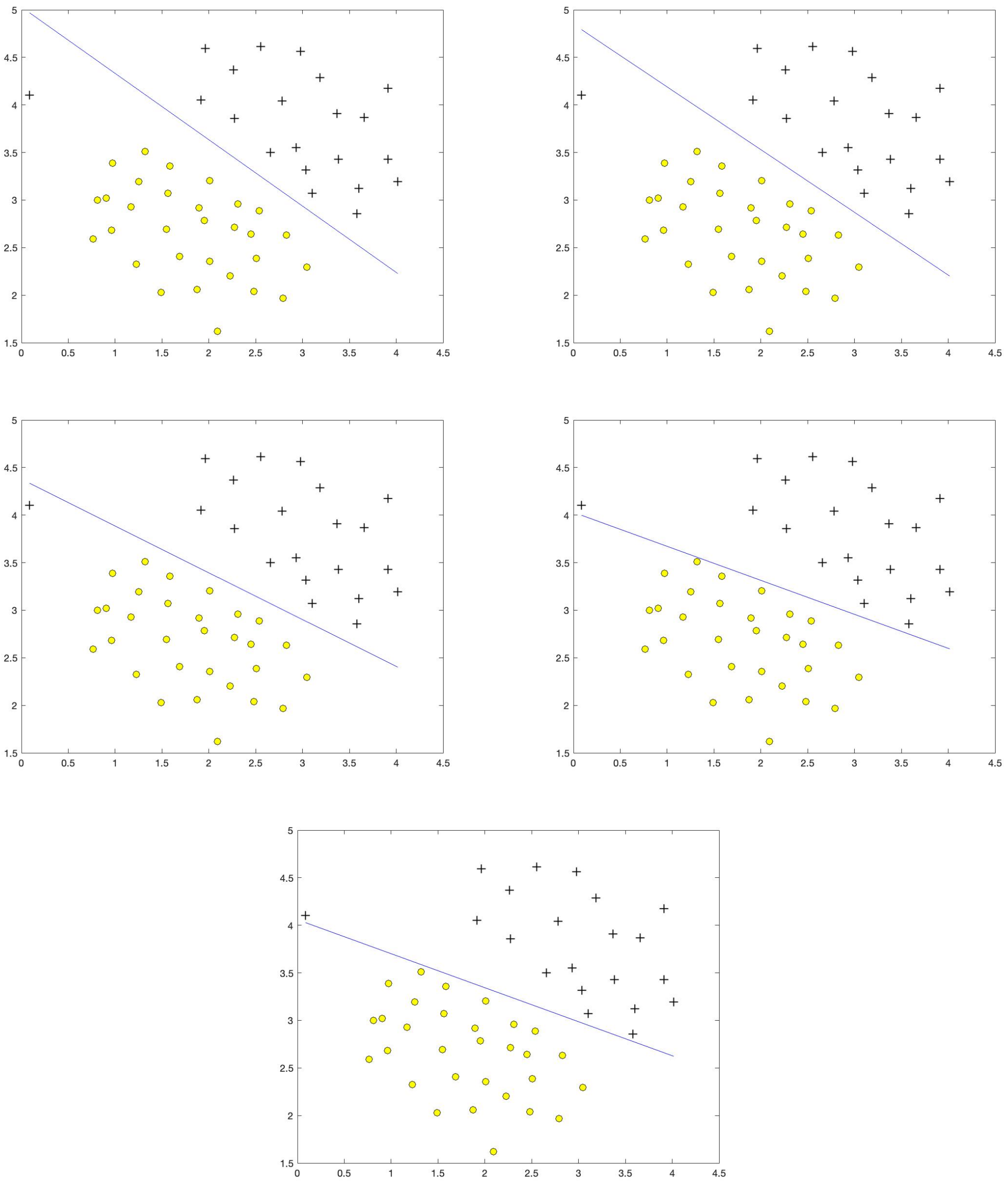

Figure 5: From left to right, top to bottom: (a) C=0.1 (b) C=1 (c) C=10 (d) C=50 (e) C=100

3.2 Discussion on the effects of C

The effect of C seems to be the significance of the misclassification of the data. For simplification, we will call the data points plotted with black '$\times$' points to be group 1, and the ones plotted with yellow 'o' points to be group 2. Now if we look at the data point at the far left classified as group 1 (call this point a), we can easily guess that this point might have been a misclassified (or is simply an anomaly).
Figure 4 shows a series of plots with a range of C values. From these, it can be postulated that the larger the C values, the less the misclassified points are allowed. In other words, larger the C value, higher the importance of classifying all the points correctly.
For example, the margin line for $C = 0.1$ goes between the two groups, almost without taking into account point a (and point a is in group 2 in this case). However, as C increases, the margin line moves to classify point a as group 1, and the importance of classifying point a correctly becomes more significant.
From the SVM hypothesis equation that was given to us for the linear kernel, we see that C is part of of an optimization problem, where it is a multiplicator of the sum of the 'classification' cost functions. This also suggests that the C value is an indicator for the significance of the correct classification of the data points.

3.3 Discussion on the best choice of C

Here, our aim is to minimize the 'misclassification' while maximizing the margin between the two classes (groups 1 and 2). While a larger C value such as $C = 100$ might seem to be a tempting choice at first because it classifies all the training data correctly, if point a were to be an anomaly or a wrongly classified point, the margin line drawn would be less accurate and therefore lead the future testing data to be more probable for misclassification. Considering that it is only one point of the training set that is misclassified, and also from the SVM hypothesis equation, the choice of C value would be at the lower end of the possible values, such as C = 1.