COL774 Assignment 1 Report

Saem Habeeb
2022MT62004

September 5, 2025

Contents

1 Linear Regression

1.1 Overview

In this question, our objective was to implement linear regression using batch gradient descent to minimize the mean square error of the predicted values by the regression model. We experimented with various learning rates and plotted 3D plots and contour plots showing how at each step batch gradient descent minimized the mean squared error.

1.2 Parameters chosen and results

I plotted the data and found that y and x looked perfectly correlated, so i chose to use the higher learning rate of 0.1.

With learning rate = 0.1 and the stopping criterion of $l_\infty(\text{gradient}) < \epsilon$ where $\epsilon = 1e-6$ the model gave $\theta = [6.21867782, 29.06475422]$ as the final set of parameters.

see Figure 1 for the plot of the hypothesis function and the data points

See Figure 2 and Figure 3 to see the steps of gradient descent starting from $\theta = [0, 0]$.

As a part of the question, we also experimented with the learning rate (η). Specifically, we drew the contours of the cost function for $\eta = \{0.001, 0.025, 0.1\}$. See Figure 4, Figure 5, Figure 6

We observe that for small learning rates, model takes many small steps to converge resulting in a lower rate of convergence. For example, for $\eta = 0.001$, it took 17194 steps. Whereas for higher learning rates, we get better rate convergence. For $\eta = 0.1$, it took 165 steps. The training cost was not much different for the different η's.

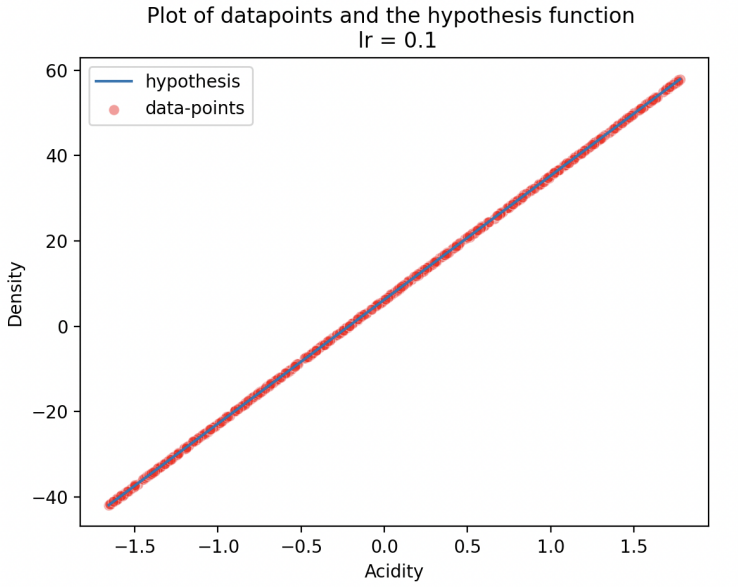


Figure 1: Visualization of the hypothesis function and the data points

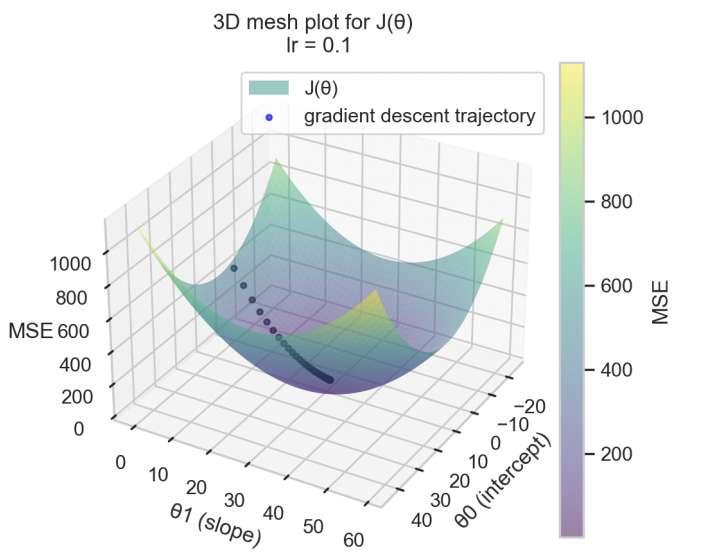


Figure 2: 3D Visualization of the cost function $J(\theta_0, \theta_1)$ and the steps of gradient descent

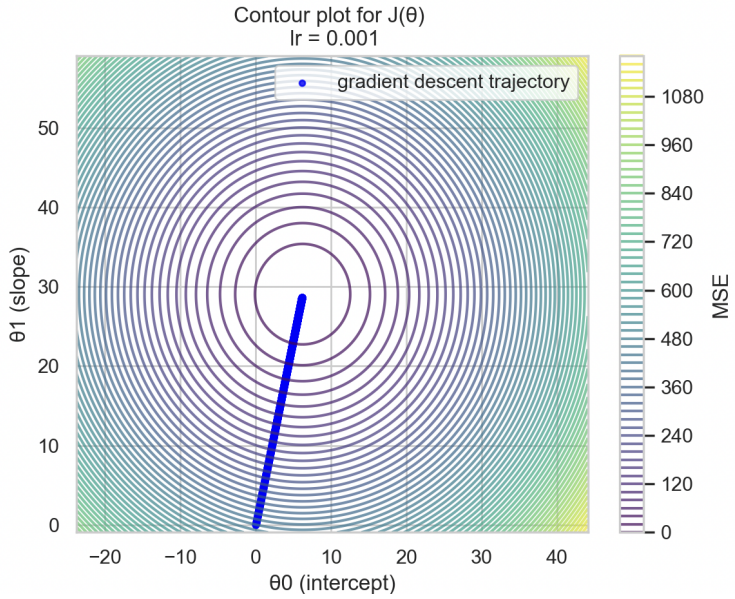


Figure 4: Contour plot of the cost function and the steps of gradient descent lr = 0.001

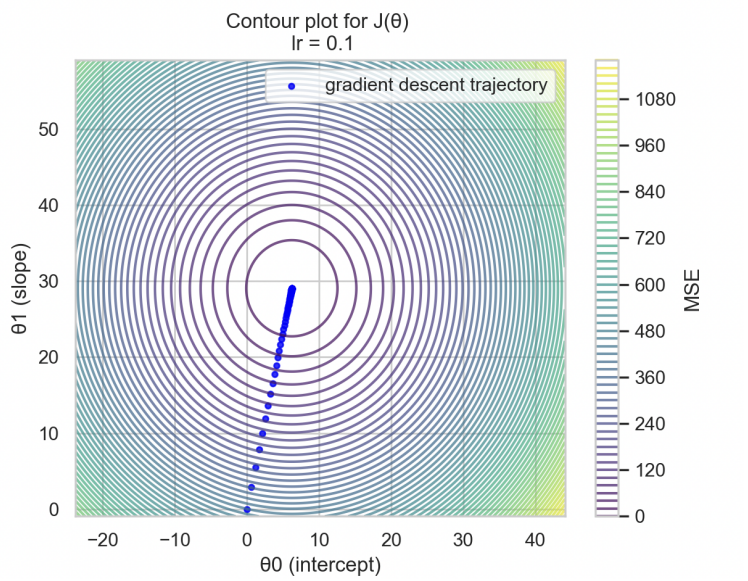


Figure 3: Contour plot of the cost function and the steps of gradient descent

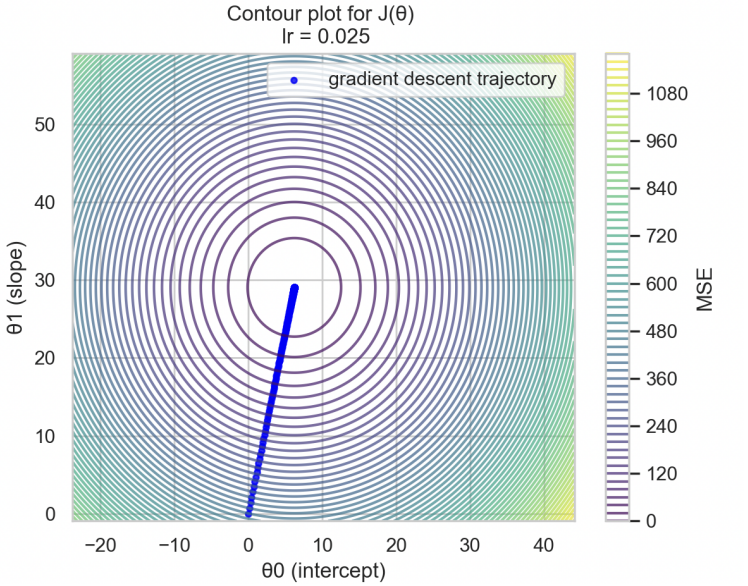


Figure 5: Contour plot of the cost function and the steps of gradient descent lr = 0.025

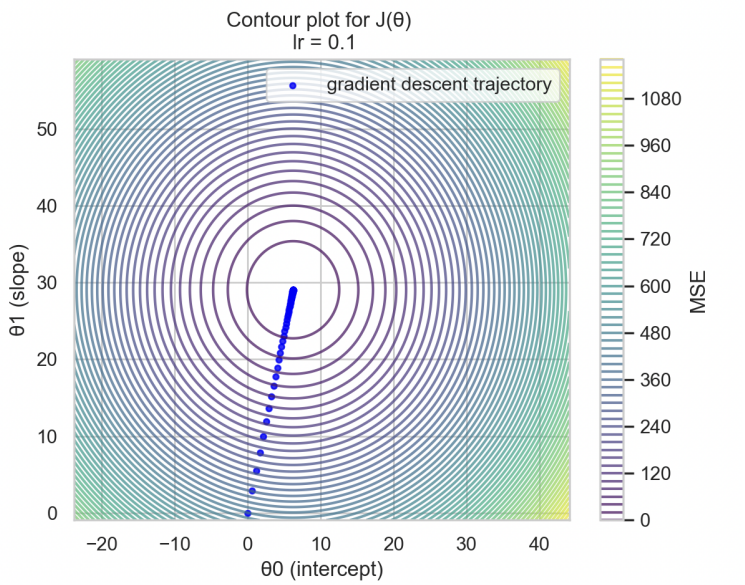


Figure 6: Contour plot of the cost function and the steps of gradient descent lr = 0.1

2 Sampling, Closed Form and Stochastic Gradient Descent

2.1 Overview

In this question, we generated normally distributed synthetic data (X). We also fixed a particular θ and generated the prediction of the hypothesis Y as $y^{(i)} = \theta^T x^{(i)} + \epsilon$ where ϵ is the added gaussian noise and $\epsilon \sim N(0, \sigma^2)$.

We split the above data into training data(80%) and test data(20%). Using the above generated data, we implemented the stochastic gradient descent(SGD) and tried to predict θ. We experimented with varying batch sizes and observed the hence predicted θ and their rates of convergence.

We also implemented the closed form solution for linear regression and compared it's result with the gradient descent approach.

We made prediction on the test data using the SGD linear regression model for varying batch sizes and compared the Mean Squared Error (MSE) of the training and the test data.

We also plotted the trajectory of θ as the parameters are updated (until convergence) for varying batch sizes and found some interesting difference in the shape of movement for each batch size.

2.2 The Data and the hyper-parameters

$$X = [x_0 \ x_1 \ x_2]^T$$

where $x_0 = 1$, $x_1 \sim N(3, 4)$, $x_2 \sim N(-1, 4)$ and we take a sample of size $m = 10^6$ using the numpy library's random module.

$$\theta = [\theta_0 \ \theta_1 \ \theta_2]^T = [3 \ 1 \ 2]^T$$
$$y^{(i)} = \theta^T x^{(i)} + \epsilon$$

where $\epsilon \sim N(0, 2)$

So, after splitting into test data and training data, we get 800,000 data points for training and 200,000 for testing.

We also vary the batch size(r) for SGD and

$$r = \{1, 80, 8000, 800000\}$$

also, the learning rate(η) = 0.001 for SGD.

2.3 The Stopping Criterion

As the cost doesn't necessarily decrease monotonically in SGD, simply using the absolute value of consecutive costs will not work. Instead we took the absolute value of the difference of the average of cost over the last 5 steps and the average of cost over the previous 5 steps i.e. if we are at the k^{th} step right now, we check for

$$|\frac{\sum_{i=k-5}^{k-1} J_i}{5} - \frac{\sum_{i=k-6}^{k-10} J_i}{5}| < \epsilon, (\epsilon = 10^{-4})$$

2.4 Observations ans Results

2.4.1 The SGD

The number of iterations needed for convergence increased as the value of r increased (Figure 7). This was because in each iteration of SGD, we updated the θ m/r times i.e. for small r we took more number of steps per iteration.

The overall time taken for convergence was least for $r = 80$ and the highest for $r = 800,000$. This is because for $r = 800,000$, we take only 1 step per iteration so it needed many iterations to converge. Whereas for $r = 80$, we take $10,000$ steps per iteration and each step is quite accurate as well as 80 sample points contribute to calculation of gradient for each step. It is kind of a balance between the number of steps taken per iteration and the accuracy of each step.

For $r = \{1, 80, 8000\}$ the $\hat{\theta}$ and θ are very close but for $r = 800000$ they are far. The training MSE was also the highest for $r = 800,000$

```
r = 1, error = 2.0059277386607173, num_iter = 12, theta = [2.9809382  0.98786295 2.01588326], error in theta = 0.027621334619884523
r = 80, error = 1.9997678245634039, num_iter = 13, theta = [2.99820391 0.99631088 2.00297848], error in theta = 0.005070194538135334
r = 8000, error = 1.99993281735886, num_iter = 157, theta = [2.96139679 1.0082402  1.99852818], error in theta = 0.03950031620338162
r = 800000, error = 2.0606181638584644, num_iter = 6604, theta = [2.53931792 1.10072854 1.96752781], error in theta = 0.4726824104522829
```

Figure 7: Results obtained from SGD for various r

2.4.2 The Closed form Solution

The closed form solution as mentioned in the class and the assignment can be found using the following formula

$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

where X is the matrix of size $m_{\text{train}} \times n$ containing the training data points. Here m_{train} is 800,000 and $n = 3$.

Using the above formula, the $\hat{\theta}$ obtained and the Euclidean distance from actual θ are in Figure 8

```
theta from closed form solution = [3.00156384 0.99974438 2.00123432], error in theta = 0.002008601625490376
```

Figure 8: Results obtained from closed form solution for linear regression

As we can see the Euclidean distance of $\hat{\theta}$ and θ is the least among the $\hat{\theta}$'s obtained yet(from SGD).

2.4.3 Evaluation on the test set

```
r = 1, test error = 2.0042483152829584, train error = 2.0059277386607173,
 relative error% (|test_error-train_error|/train_error * 100) = 0.08372302478254404
r = 80, test error = 1.9978350440895913, train error = 1.9997678245634039,
 relative error% (|test_error-train_error|/train_error * 100) = 0.09665024359687771
r = 8000, test error = 1.9981659352251762, train error = 1.99993281735886,
 relative error% (|test_error-train_error|/train_error * 100) = 0.08834707437908747
r = 800000, test error = 2.0613075598182196, train error = 2.0606181638584644,
 relative error% (|test_error-train_error|/train_error * 100) = 0.03345578389274538
```

Figure 9: Results from evaluation on test data

The relative error in the test and train error is clearly negligible still, we can see wherever the train error was low, the relative error is a bit higher (e.g. $r = 80$). For $r = 800,000$ the relative error is the least.

2.4.4 Trajectories of $\hat{\theta}$'s

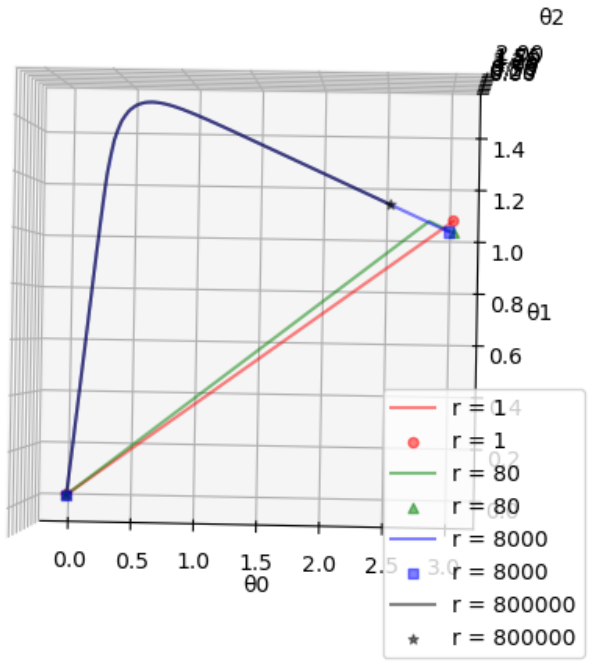


Figure 10: Trajectory of $\hat{\theta}$ for various r

Figure 10 shows the required trajectories.

As after each iteration, the number of updates to $\hat{\theta}$ ($m/r = \{800000, 10000, 100, 1\}$ for the values of r in order) are more for $r1 = \{1, 80\}$ as compared to $r2 = \{8000, 800000\}$, it looks almost as if the convergence for the set r1 is much more direct than the set r2.

If the value of $\hat{\theta}$ was plotted after each update to $\hat{\theta}$, the trajectories of the r's in r1 would have been closer (after some smoothening) to the r's in r2.

3 Logistic Regression

3.1 Overview

In this question, we were asked to implement and train a logistic regression model for the given data. ($x^{(i)} \in R^2$) and $y^{(i)} \in \{0, 1\}$. We have to maximize the log-likelihood function

$$L(\theta) = \sum_{i=1}^{m} \left[y^{(i)} \log h_\theta\big(x^{(i)}\big) + \big(1 - y^{(i)}\big) \log \big(1 - h_\theta\big(x^{(i)}\big)\big) \right]$$

. where

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

This can be done using Newton's method using the following update step.

$$\theta := \theta - H^{-1} \nabla_\theta L(\theta)$$

where H is the hessian matrix of $L(\theta)$

3.2 Results

The

$$\hat{\theta} = [0.40125316, 2.5885477, -2.72558849]^T$$

and the final value of log-likelihood is:

$$\min_\theta L(\theta) = -22.834144984$$

Look at the graph in Figure 11 to visualize the decision boundary and the original data points.

Figure 11: Trajectory of $\hat{\theta}$ for various r

4 Gaussian Discriminant Analysis

4.1 Overview

In this question we explored the Gaussian Discriminant Analysis (GDA) and classified the given data using it. We implemented GDA for both the cases, a) the co-variance matrices are same for both the classes and b) the co-variance matrices could be different for the two classes. We also found the decision boundaries in both the cases.

4.1.1 Formula for finding the decision boundary

For the case where we assume $\Sigma = \sigma_0 = \sigma_1$, the decision boundary is:

$$(\mu_1 - \mu_0)^\top \Sigma^{-1} x + \tfrac{1}{2}\left(\mu_0^\top \Sigma^{-1} \mu_0 - \mu_1^\top \Sigma^{-1} \mu_1\right) + \log \frac{\phi}{1-\phi} = 0$$

where ϕ is the probability that an observation is in class 1.

For the general case, the decision boundary is:

$$x^\top A x + b^\top x + c = 0,$$

with

$$A = \tfrac{1}{2}\left(\Sigma_0^{-1} - \Sigma_1^{-1}\right),$$

$$b = \Sigma_1^{-1}\mu_1 - \Sigma_0^{-1}\mu_0,$$

$$c = -\tfrac{1}{2}\mu_1^\top \Sigma_1^{-1}\mu_1 + \tfrac{1}{2}\mu_0^\top \Sigma_0^{-1}\mu_0 - \tfrac{1}{2}\log|\Sigma_1| + \tfrac{1}{2}\log|\Sigma_0| + \log\frac{\phi}{1-\phi}$$

4.1.2 Observations and Results

The data is Normalized before training the model.

If we assume $\Sigma = \sigma_0 = \sigma_1$, the following values of μ_0, μ_1 and Σ are obtained

```
1. mu_0 = [-0.75529433  0.68509431], mu_1 = [ 0.75529433 -0.68509431]
 var-covar matrix
[[ 0.42953048 -0.02247228]
 [-0.02247228  0.53064579]]
```

Figure 12: The values of μ_0, μ_1 and Σ

The equation of the linear boundary is:

$$3.389254x_0 - 2.438583x_1 = 0$$

For the general case, the following values of μ_0, μ_1 and Σ_0, Σ_1 are obtained

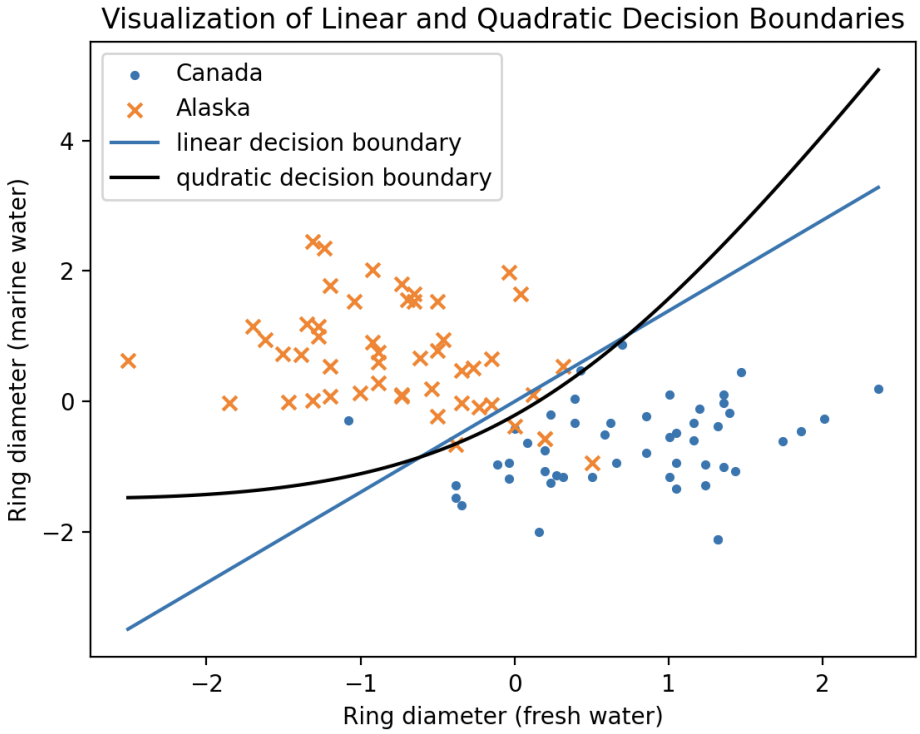
```
4. mu_0 = [-0.75529433  0.68509431], mu_1 = [ 0.75529433 -0.68509431]
 var-covar matrix for label 0
[[ 0.38158978 -0.15486516]
 [-0.15486516  0.64773717]]
var-covar matrix for label 1
[[0.47747117 0.1099206 ]
 [0.1099206  0.41355441]]
```

Figure 13: The values of μ_0, μ_1 and Σ_0, Σ_1

The equation of the quadratic boundary is:

$$0.33567x_0^2 + 1.2868x_0x_1 - 0.4330x_1^2 + 3.8079x_0 - 2.8597x_1 - 0.5848 = 0$$

The following graph shows the data points, the linear and the quadratic decision boundaries.



The quadratic boundary gives a better classification criterion that the linear boundary.