Machine Learning : Assignment 1 report

Aaveg Jain : 2021CS10073

September 2023

1 Introduction

In this assignment, we aim to explore linear regression, logistic regression, and Gaussian Discrimant Analysis(GDA). We also visualise the models we train to get some intuition of their accuracy.

2 Problem 1 : Linear regression

2.1 a)

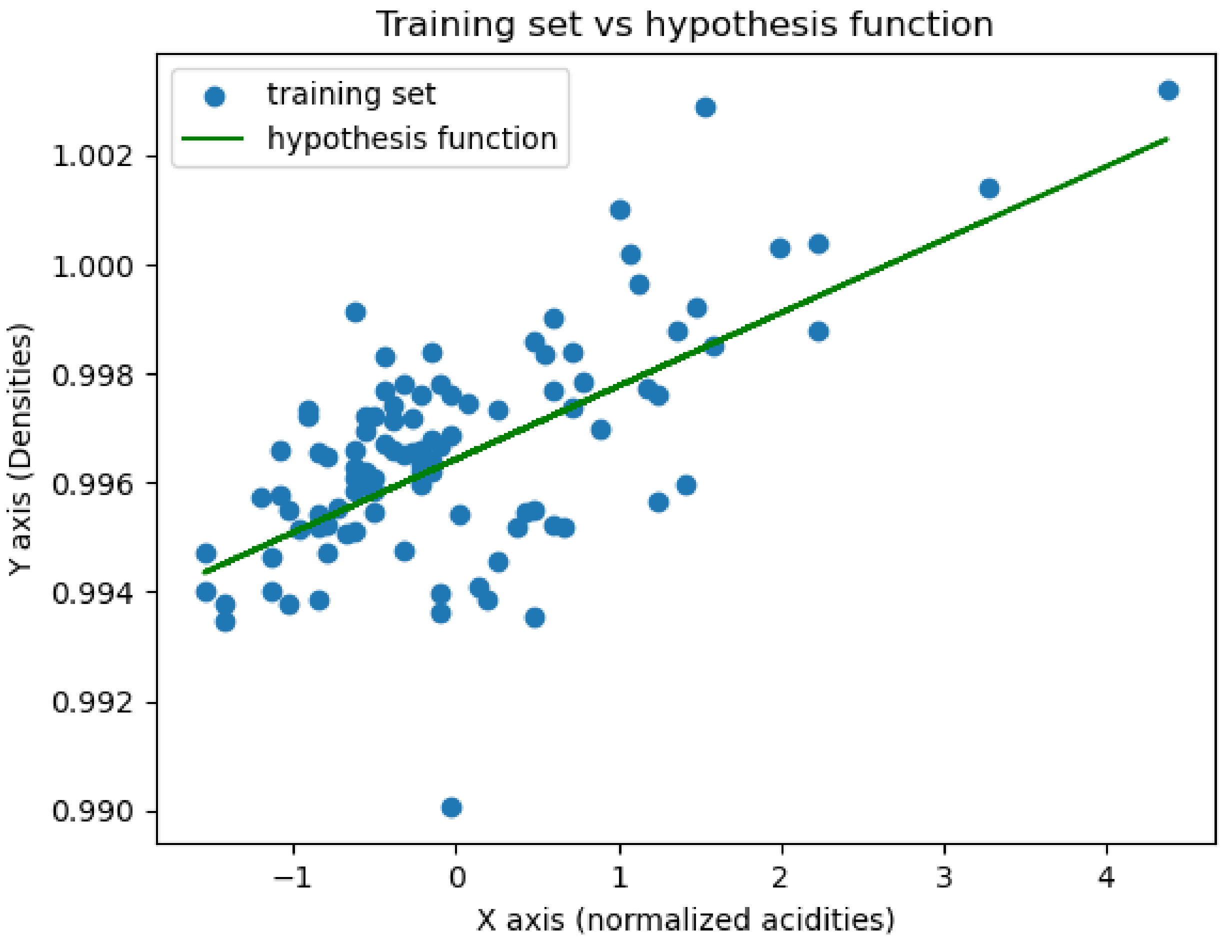
In this part, we implemented batch gradient descent to perform linear regression on the training set. **Learning parameter** : 0.1

Convergence criteria : $\left|J(\theta^{(t+1)}) - J(\theta^{(t)})\right| < \epsilon$

where ϵ is defined to be a small fraction (1e-7) of the original error in the data, and rest notations are as discussed in the class.

Parameters Learned : $\theta = [0.996,\ 0.00134]$ where $\theta = [\theta_0,\ \theta_1]$

2.2 b)



2.3 c), d), e)

The animations for these parts have been submitted seperately. In part e), we consider different learning parameters 0.001, 0.025, 0.1 . The number of iterations taken for these are 6101, 306, 81 respectively.

The learned θ for each parameter are [0.994, 0.00134], [0.996, 0.00134], [0.996, 0.00134] respectively.

We observe that there is a negligible difference between the learned values of the parameters, however, the number of iterations required increases almost linearly with the learning parameter. The same can be seen from the plot of the contours plotted at each iteration for the above cases. We observe that as the learning parameter reduces, the speed of convergence to the convergent theta slows down, although in each case we converge essentially to the same θ

3 Problem 2: Stochastic Gradient Descent

3.1 b)

While implementing mini batch gradient descent, the convergence criteria used is as follows:

We initially shuffle the training set. This is performed only once in the beginning.

We update the parameters for each batch, and we compute the average batch error for k consecutive batches. Then we consider the next k consecutive batches and compute their average batch error. The difference between these average values should be less than some ϵ for convergence to hold. Also, we keep a cap on the maximum number of iterations (batches) our algorithm runs through, as for smaller batch sizes, the noise in the path of θ may cause it to start oscillating around the convergent value.

k and ϵ were found experimentally, whereas the learning parameter is fixed at 0.001.

Batch sizes are -

$$r_1 = 1, r_2 = 100, r_3 = 10000, r_4 = 1000000$$

and corresponding learned θ are

$$\theta_{r_1} = \begin{bmatrix} 3.0034 \\ 1.0190 \\ 2.0742 \end{bmatrix} \theta_{r_2} = \begin{bmatrix} 2.9882 \\ 1.0053 \\ 1.9947 \end{bmatrix} \theta_{r_3} = \begin{bmatrix} 2.8185 \\ 1.0396 \\ 1.9863 \end{bmatrix} \theta_{r_4} = \begin{bmatrix} 2.8180 \\ 1.0396 \\ 1.9862 \end{bmatrix}$$

$$\theta_0 = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$$

3.2 c)

Observations related to this part-

- We observe that the different algorithms converge to observably different values of theta, depending on the convergence criteria, and the batch size.
- The relative difference between the learned parameters and the original hypothesis, are (
- We observe that the no of iterations taken(23480, 20120, 10010, 10001) decreases as we increase the batch size, although the total time taken increases with increasing batch size. This can be rationalised by the following argument-
  The noise in the trajectory of theta is more for less batch sizes, thus they require more iterations to converge, however as the time needed for one iteration is also less for small batch sizes (one iteration means updating parameters using 1 batch), the overall time decreases with increasing batch size. Thus the speed of convergence is more for smaller batch sizes, as time taken is less to converge and the parameters update more frequently, but the no of iterations is also more for the smaller batch sizes.
- Defining error as follows-
  $$J(\theta) = \frac{\sum_{i=1}^{m}(y^{(i)} - h_\theta(x^{(i)}))^2}{2m}$$
  where m is the training data size. We observe that errors (rms) for increasing batch sizes are
  $$J(\theta_{r_1}) = 1.27 \;\; J(\theta_{r_2}) = 0.99 \;\; J(\theta_{r_3}) = 1.08 \;\; J(\theta_{r_4}) = 1.08$$
  whereas the error wrt the original hypothesis is
  $$J(\theta_0) = 0.98$$
  Thus we observe that the percentage errors are approximately(wrt original error) 30%, 1%, 10%, 10%. Thus we observe that from a perspective of both performance and accuracy, batch size = 100 performs the most optimally. The maximum error is for batch size = 1 (SGD), this can be attributed to the noisy path taken in case of SGD. Also we observe that for cases of batch size = 1e4 and 1e6, the convergence is due to early stopping by capping the maximum number of iterations. This was done as a tradeoff between performance and accuracy.

3.3 d) (plots in the end)

We observe that as we vary batch size, the noisiness of the path of movement of theta decreases with increasing batch size. It is the most noisy for batch size = 1 (SGD) and least for batch gradient descent. This makes intuitive sense also, as we know that in case of SGD we update the parameters on basis of gradient of error in only 1 test input. As the test inputs can be individually scattered due to their variance, the path taken by theta can be very different from that towards the convergent theta ([3,1,2]) in this case. However as we increase the batch size, the variance in the gradient reduces as more test inputs are used, and thus theta starts changing more towards the convergent theta, and the noise in the path reduces.

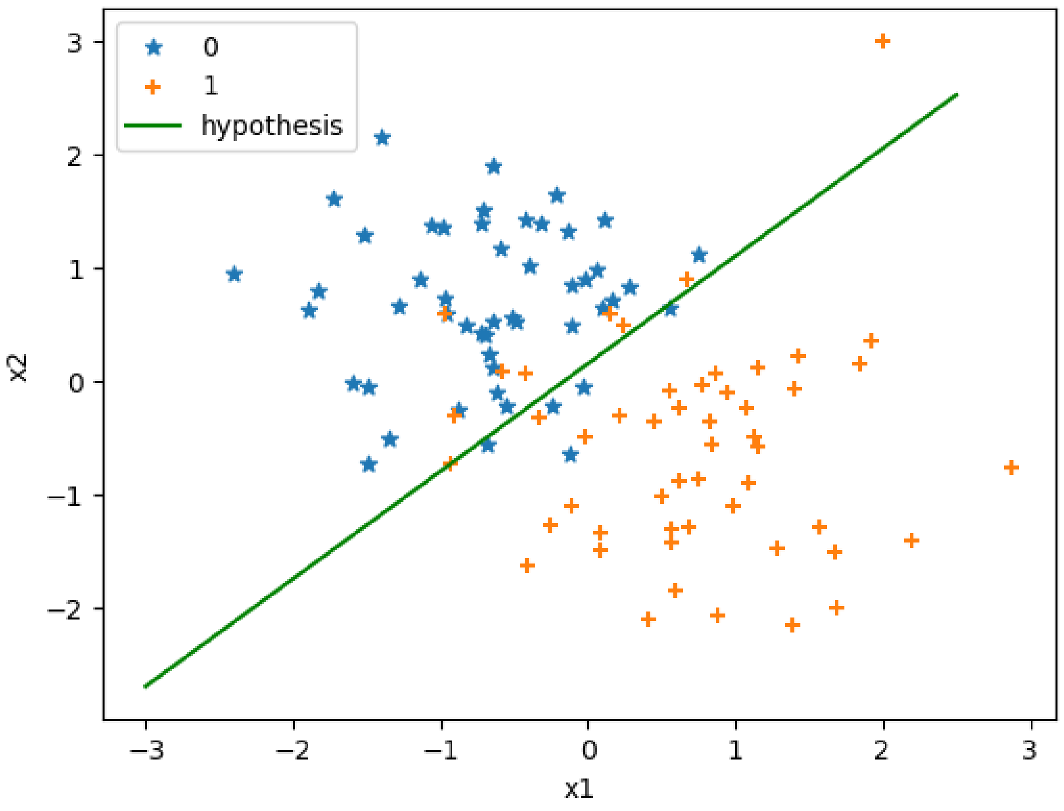
4 Problem 3: Logistic Regression

4.1 a)

Learned theta is

$$\theta_{logistic} = \begin{bmatrix} 0.40123187 \\ 2.58849794 \\ -2.7255376 \end{bmatrix}$$

4.2 b)



5 Problem 4: Gaussian Discriminant Analysis

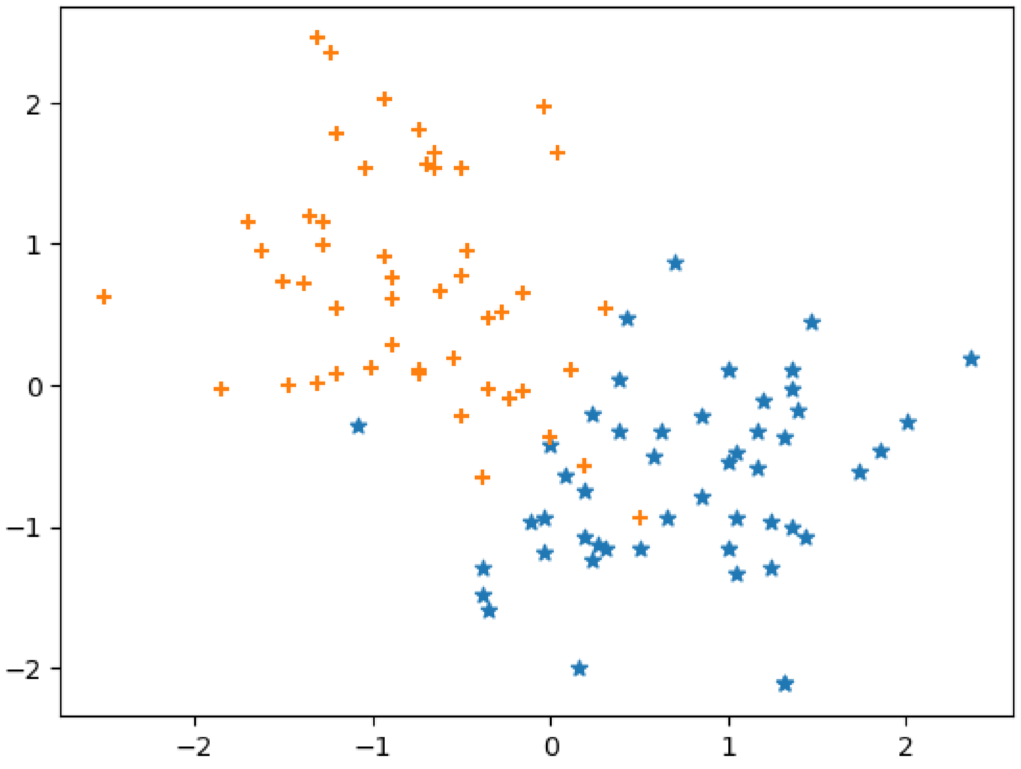
5.1 a)

$$\mu_0 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

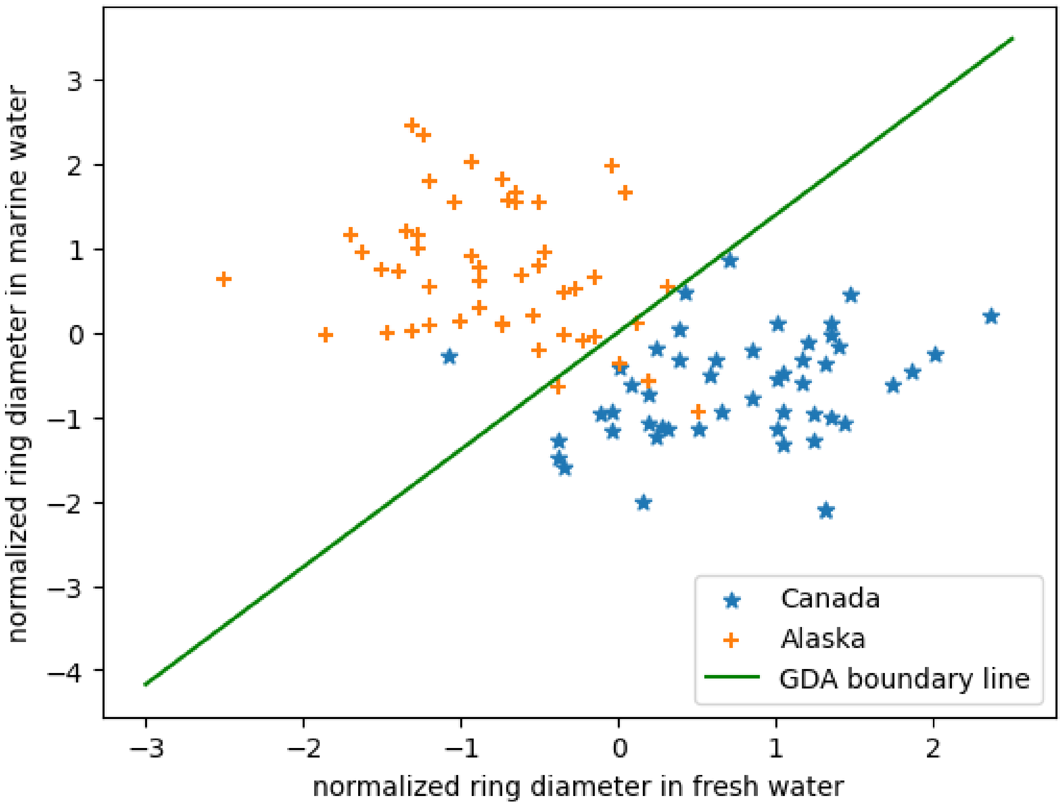
$$\mu_1 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix} = -\mu_0$$

$$\Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$$

5.2 b) Plot of training data



5.3 c)



Equation of the boundary:

$$\theta^T x = 0 \ \ where \ \ \theta = [\theta_0, \theta_1, \theta_2]$$

$$\theta_0 = \mu_1^T \Sigma^{-1} \mu_1 - \mu_0^T \Sigma^{-1} \mu_0$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = 2 * \Sigma^{-1} (\mu_0 - \mu_1)^T$$

5.4 d)

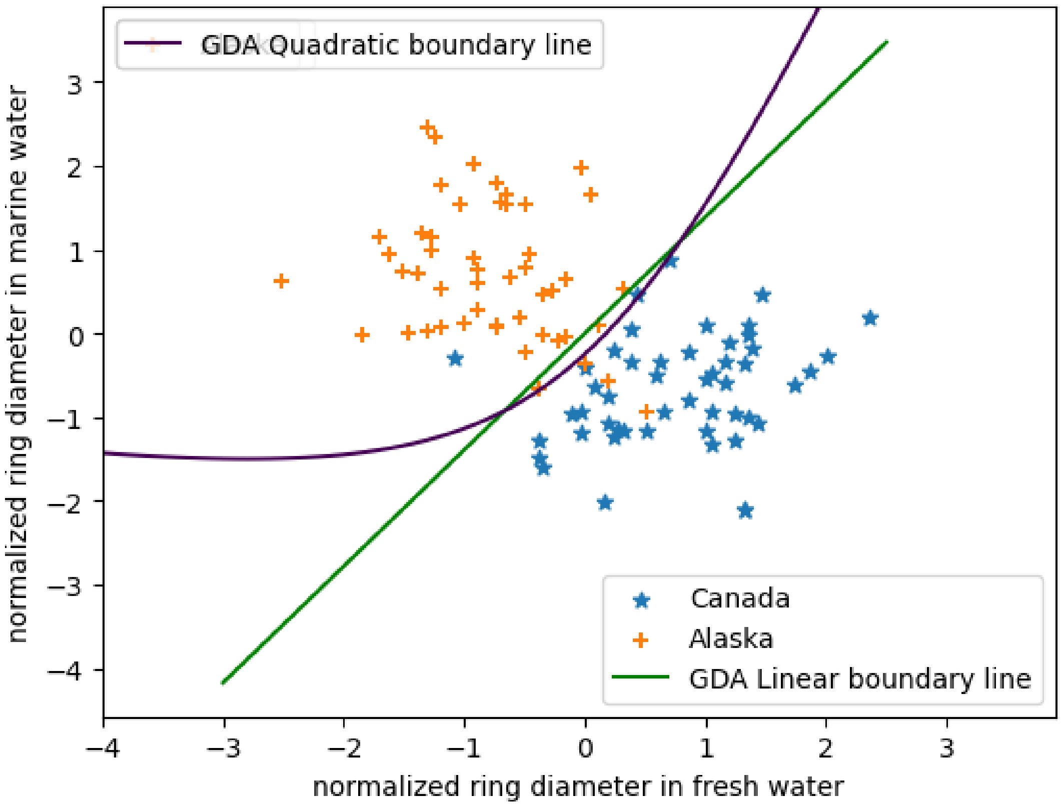
$$\mu_0 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix} = -\mu_0$$

$$\Sigma_0 = \begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{bmatrix}$$

5.5 e)



equation of quadratic boundary:

$$x^T(\Sigma_1^{-1} - \Sigma_0^{-1})x + 2 * x^T(\Sigma_0^{-1}\mu_0 - \Sigma_1^{-1}\mu_1) + (\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0) = 0$$

5.6 f)

Analysis of the boundaries- We see that the linear and the quadratic boundary lines divide the points with very high accuracy, and are also very closely related to each other. Their accuracies are approximately 94% and 96% respectively. Thus the quadratic boundary is a slightly better fit than the linear boundary. This is because of the weaker set of assumptions taken in the GDA model for the quadratic case, where we allow the two classes to have different covariance matrix.

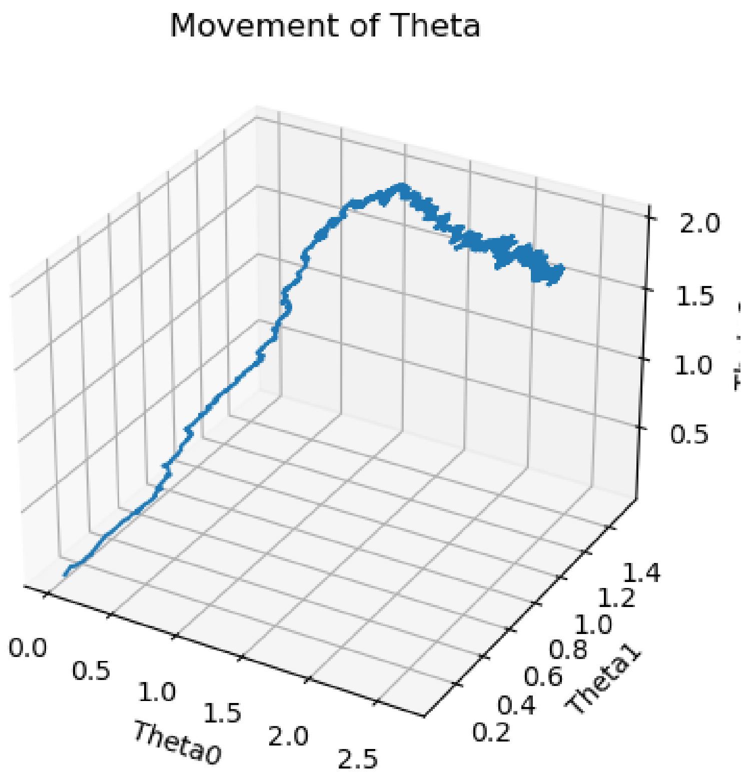


Figure 1: Batch size 1

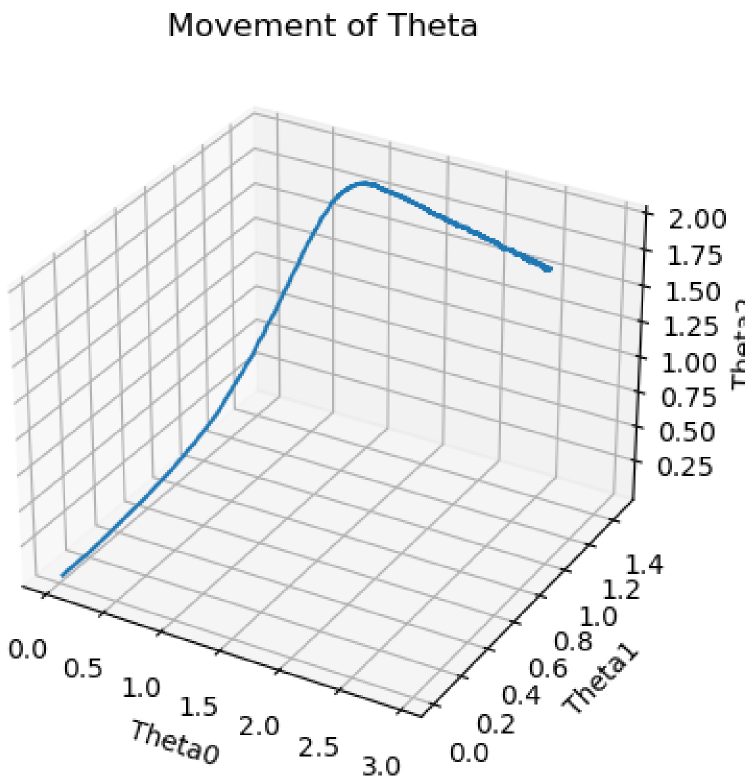


Figure 2: Batch size 100

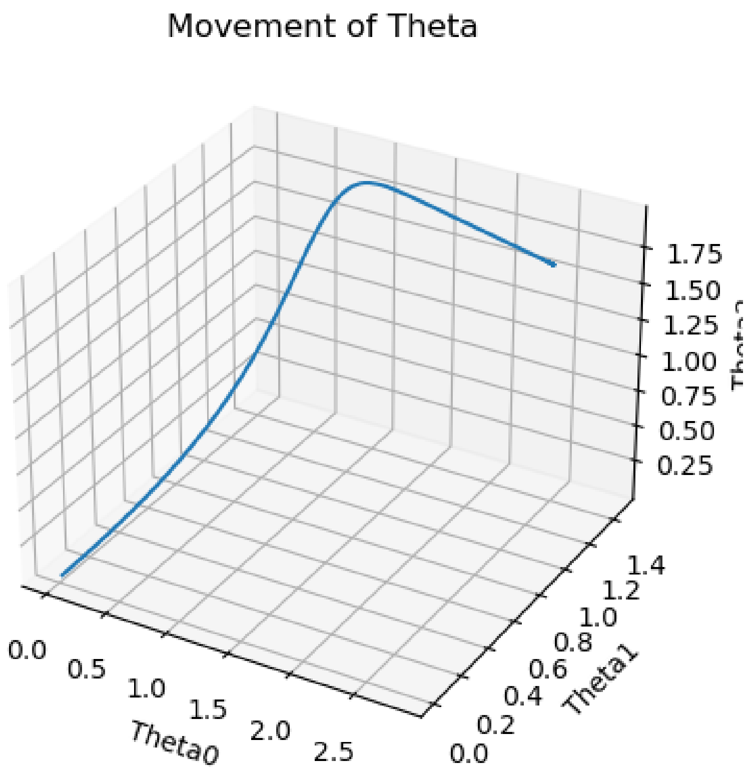


Figure 3: Batch size 10000

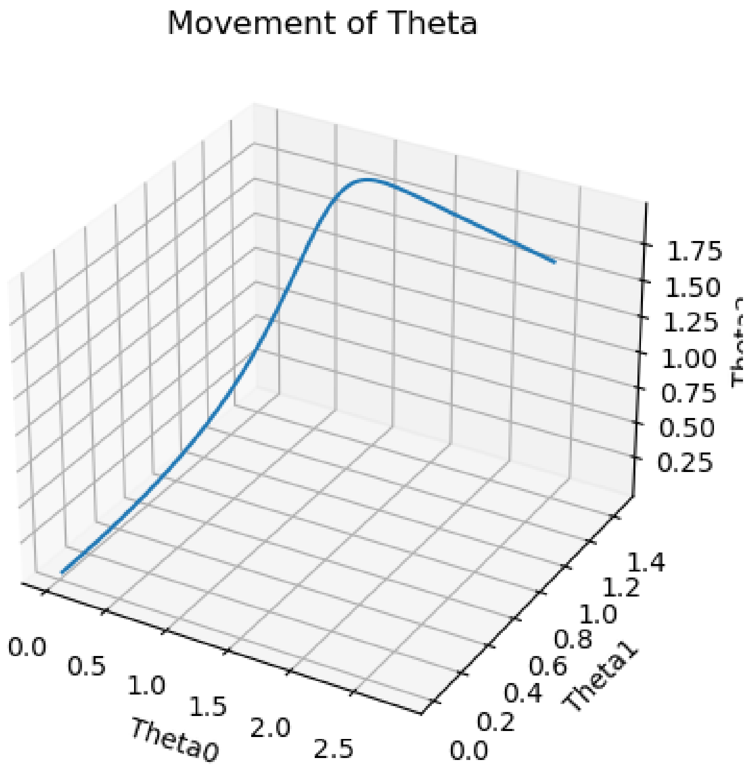


Figure 4: Batch size 1000000