CS F320: Foundations of Data Science

Assignment-1

Shashank Pandey, 2020A7PS0238H
Kishan Koundinya, 2020A7PS0213H
Prithvi Hegde, 2020A7PS1718H

Assignment 1-A:

We know that the beta distribution is a conjugate prior, i.e, the overall distribution follows the same beta shape even after updating it with new data.

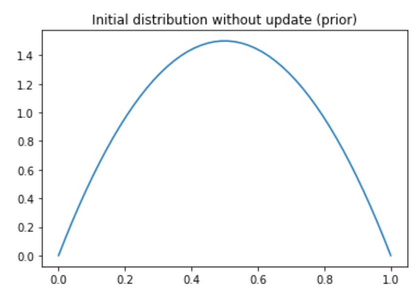
The beta distribution is mathematically described by:

$$\mathrm{Beta}(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1}$$

On receiving new information (i.e, on updating it) we find the new distribution that emerges also follows a beta distribution. This posterior distribution is described as:

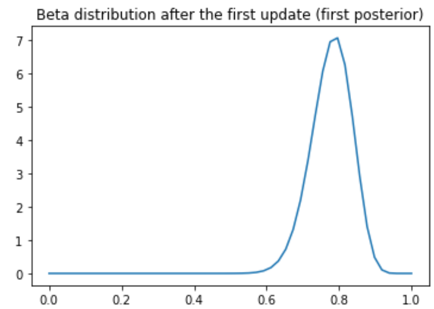
$$p(\mu|m,l,a,b) = \frac{\Gamma(m+a+l+b)}{\Gamma(m+a)\Gamma(l+b)}\mu^{m+a-1}(1-\mu)^{l+b-1}.$$

Therefore, our initial distribution (which is just a simple beta distribution, with a = 2 and b = 2) is plotted and found to be the following.



Updating the beta distribution:

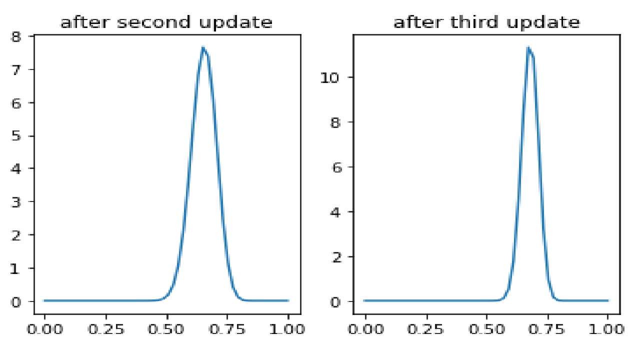
We find that out of 50 people surveyed, 40 people liked and 10 people disliked this update. Therefore, we update our values of (a,b) to now be (42,12). The posterior distribution reflects this accordingly.



In the next survey, we found that 13 people liked and 17 disliked the update. Therefore, our old priors (the beta distribution with a = 42, b = 12, which was the state after the first update) now get updated to become the posteriors, as **a = 55, and b = 29.**

In the final update, we find that 70 people liked and 30 people disliked the update. We therefore, once again, update our beta distribution to calculate a new posterior distribution,
with a = 125, and b = 59.

We can see both distributions below.



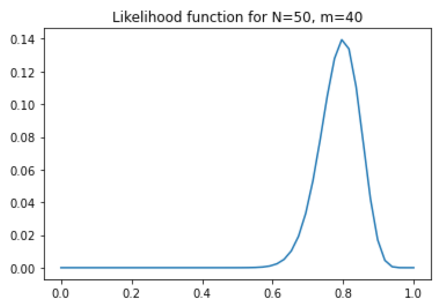
Calculating the likelihoods:

The likelihood function for a beta distribution is given by:

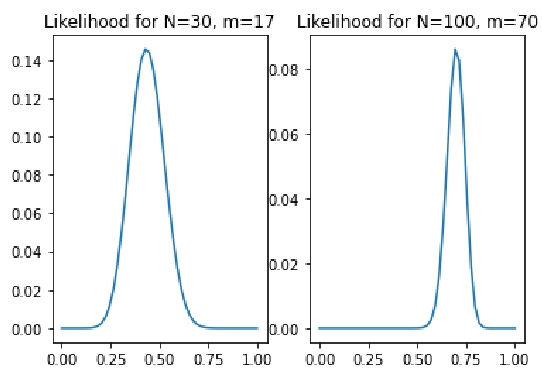
$$\mathrm{Bin}(m|N,\mu) = \binom{N}{m}\mu^m(1-\mu)^{N-m}$$

$$\binom{N}{m} \equiv \frac{N!}{(N-m)!m!}$$

Using this, we can calculate the likelihood of the first update (with m (number of successes (+ve responses)) = 40, and N (no.of responses) = 50).
When we plot this likelihood, we get the following distribution,



Similarly, we can plot the likelihoods of the next two updates (with m = 13, N = 30 for the first update, and with m = 70, N = 100 in the second update) as follows:



Assignment 1-B:

- Firstly, after reading the dataset from the CSV file, we randomly sampled the dataset into training and test sets, in an 80:20 ratio.
- Gradient Descent was implemented with **Learning Rate** (Alpha) = 0.01, **Maximum limit on iterations** = 1000, and a maximum threshold of 10^{-6}.
- The Stochastic Gradient Descent algorithm was implemented with a **Learning Rate** of 0.001, **Maximum limit on iterations** = 1000, and a maximum threshold of 10^{-6}. after randomly shuffling the dataset, batches were created, with **Batch Size** = 20, which was decided by taking a look at the number of points in the dataset.
- L1, L2, L0.5, L4 regularization were implemented with α = 0.01 and 1000 iterations.
- The trisurf function of matplotlib was used to plot the predicted polynomials of degree 1-9, against a scatter plot of both the training and test datasets. The training points are represented in red, and the test data points in blue.
- We have also plotted the training errors of the n-th degree polynomial found by gradient descent, vs the number of iterations, to show how the error generated by the loss function decreases iteratively.

Please note that all the values shown below vary on each run of the program depending on the random sample chosen for training and testing.

Tabulation of Training and Testing Errors:

Training Error for degree 1 is: 0.02848183139026568
Training Error for degree 2 is: 0.02923224895921321
Training Error for degree 3 is: 0.030073240889001047
Training Error for degree 4 is: 0.030843730185552937
Training Error for degree 5 is: 0.030836478870300904
Training Error for degree 6 is: 0.030825276877712626
Training Error for degree 7 is: 0.030820917356512406
Training Error for degree 8 is: 0.030810788755709306
Training Error for degree 9 is: 0.030797531591156174

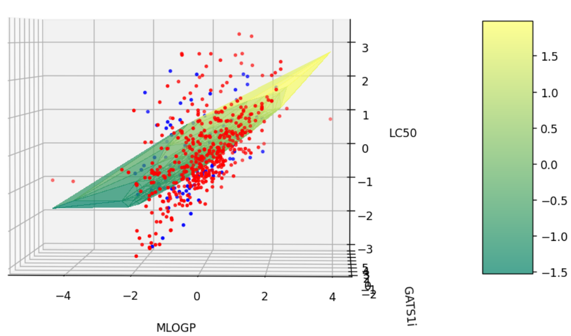
Testing Error for degree 1 is: 0.27178196374909687
Testing Error for degree 2 is: 0.2720648764636124
Testing Error for degree 3 is: 0.3030296984584212
Testing Error for degree 4 is: 2.5553482474506053
Testing Error for degree 5 is: 13.01709965751872
Testing Error for degree 6 is: 174.47024211414342
Testing Error for degree 7 is: 1144.6829022439858
Testing Error for degree 8 is: 3391.970204545584
Testing Error for degree 9 is: 15681.25103467914

Best Degree is: 1, with error of: 0.27178196374909687

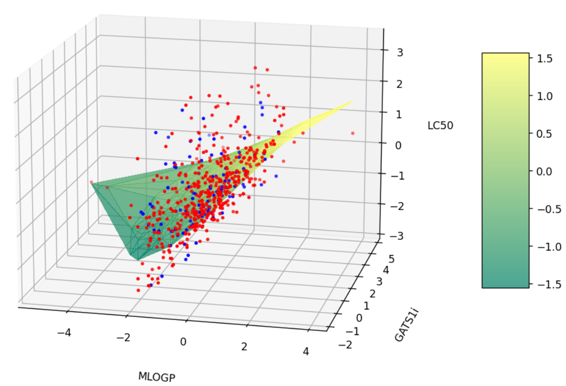
SGD testing Error for degree 1 is: 0.28346501002782254
SGD testing Error for degree 2 is: 0.31368237707039663
SGD testing Error for degree 3 is: 0.368760347911074
SGD testing Error for degree 4 is: 4.025392922614705
SGD testing Error for degree 5 is: 34.588148979874916
SGD testing Error for degree 6 is: 475.57104235967637
SGD testing Error for degree 7 is: 4271.1750597172295
SGD testing Error for degree 8 is: 34512.528790756
SGD testing Error for degree 9 is: 155609.93070392724

Best Degree SGD is: 1, with error of: 0.28346501002782254

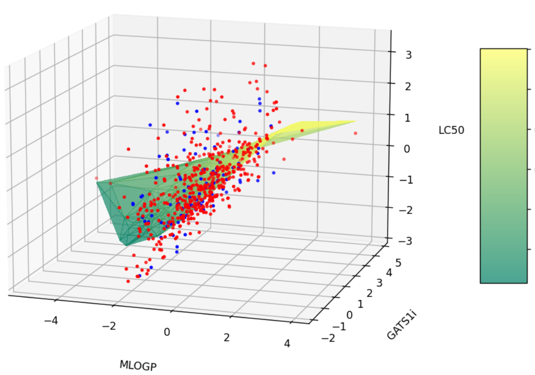
Surface Plots of the predicted polynomials:



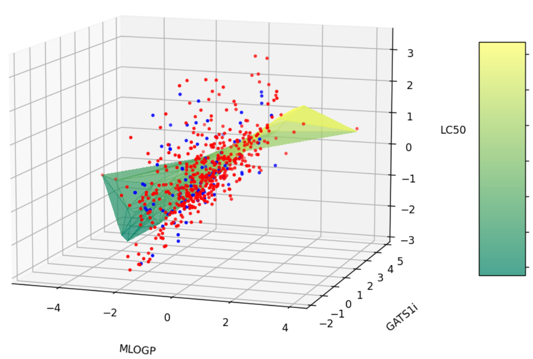
Predicted Polynomial for regression model of degree 1



Predicted Polynomial for regression model of degree 2

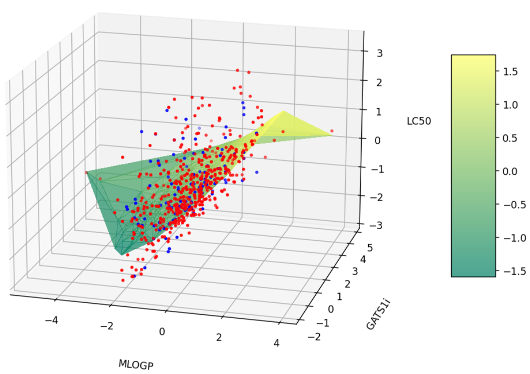


Predicted Polynomial for regression model of degree 3

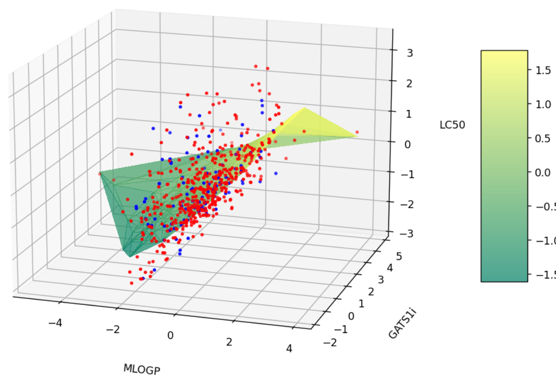


Predicted Polynomial for regression model of degree 4

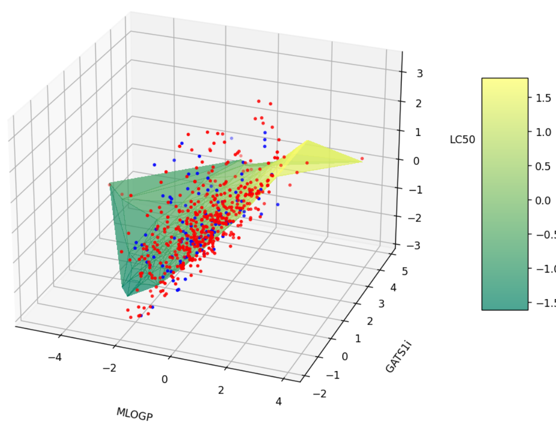
Predicted Polynomial for regression model of degree 5



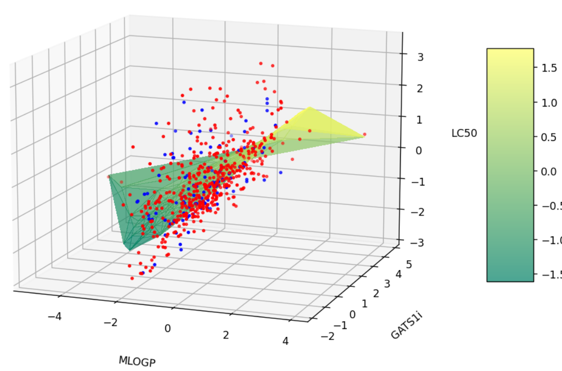
Predicted Polynomial for regression model of degree 6



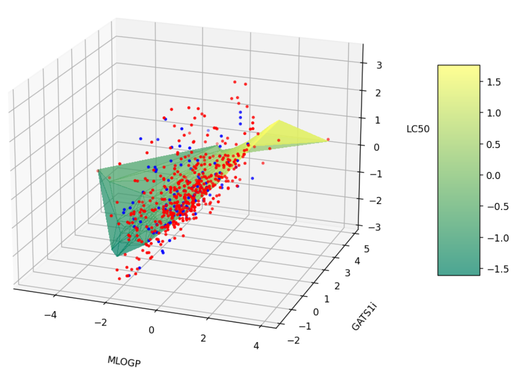
Predicted Polynomial for regression model of degree 7



Predicted Polynomial for regression model of degree 8

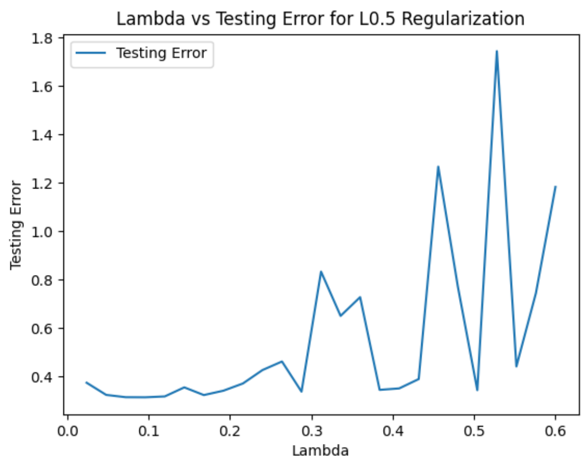


Predicted Polynomial for regression model of degree 9

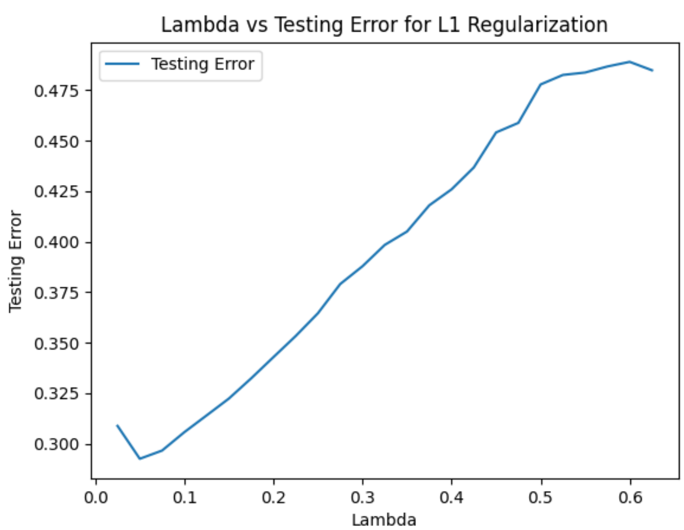


Regularisation:

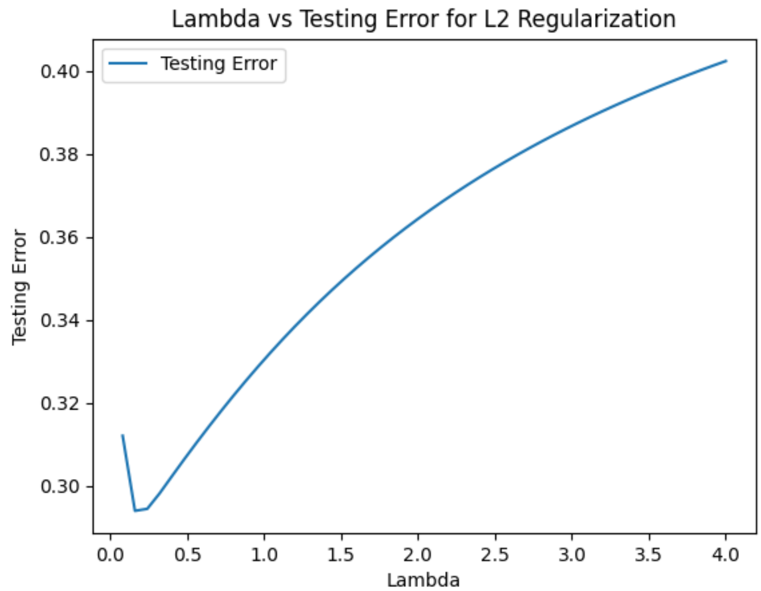
Regularisation was done on polynomial of degree 9 with four models obtained by taking q = 0.5, 1, 2, and 4. The regression output for the different regularisation models showed significantly lesser testing errors than normal gradient descent with degree 9 polynomial, showing the impact of regularisation. The optimal value of regularisation parameter λ was chosen by experimenting on multiple values of lambda and choosing the one which gives minimal training error. The graphs of λ versus Testing error is shown here for each model.



λ versus Testing Error for L0.5 regularisation
Optimal value of λ = 0.096, with error of 0.31422

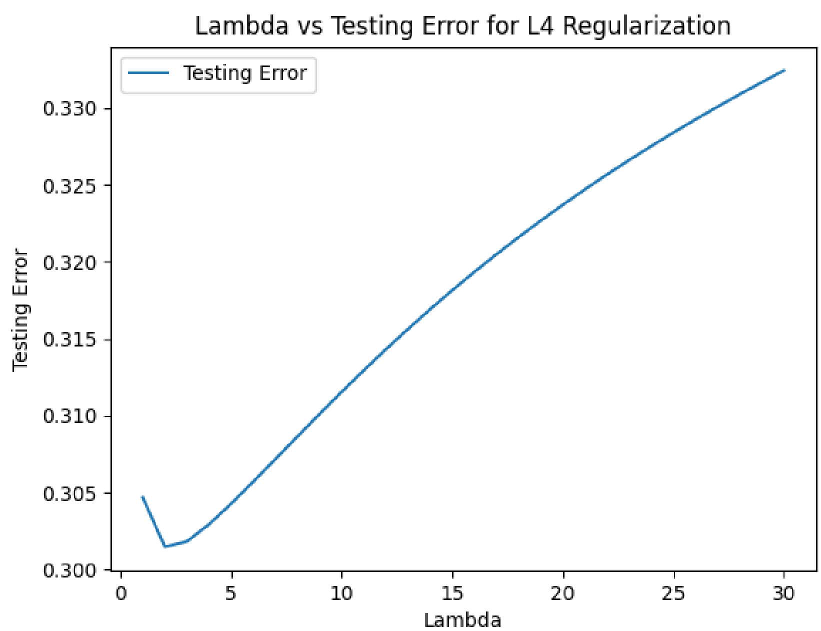


λ versus Testing Error for L1 regularisation
Optimal value of λ = 0.05, with error of 0.2945



λ versus Testing Error for L2 regularisation

Optimal value of λ = 0.16, with error of 0.2939



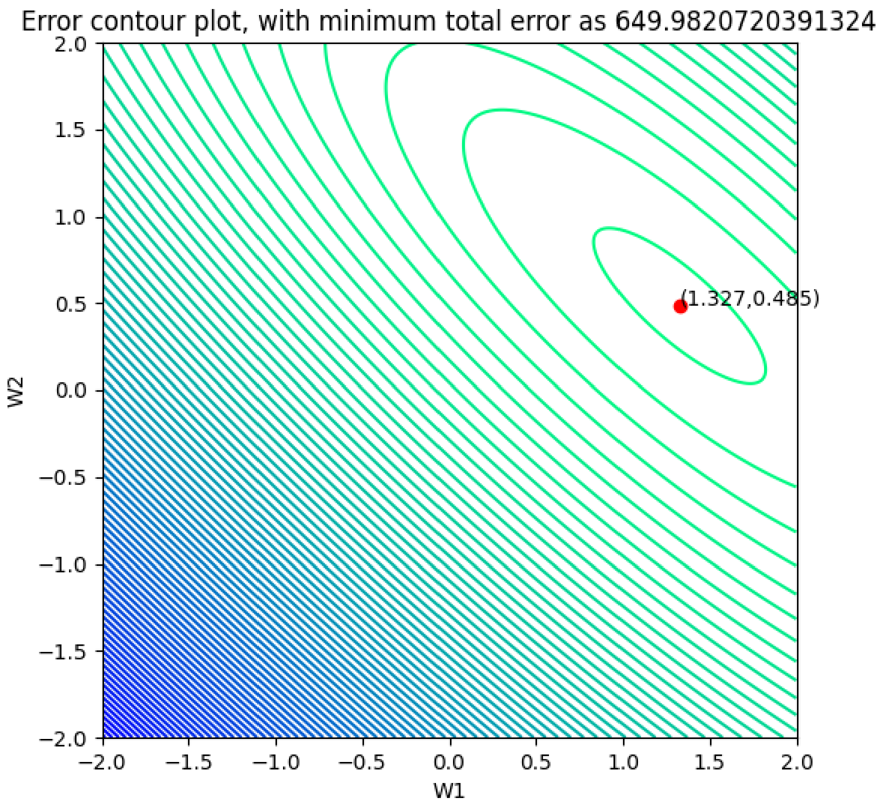
λ versus Testing Error for L4 regularisation

Optimal value of λ = 2, with error of 0.3014

By comparing the different models, L2 regularisation consistently gave the best errors and hence performed the regularisation the best, followed by L1, followed by L0.5 and L4. But despite this, the classic degree 1 regression model always had the lowest testing error, and this could be because of the bias added due to regularisation.

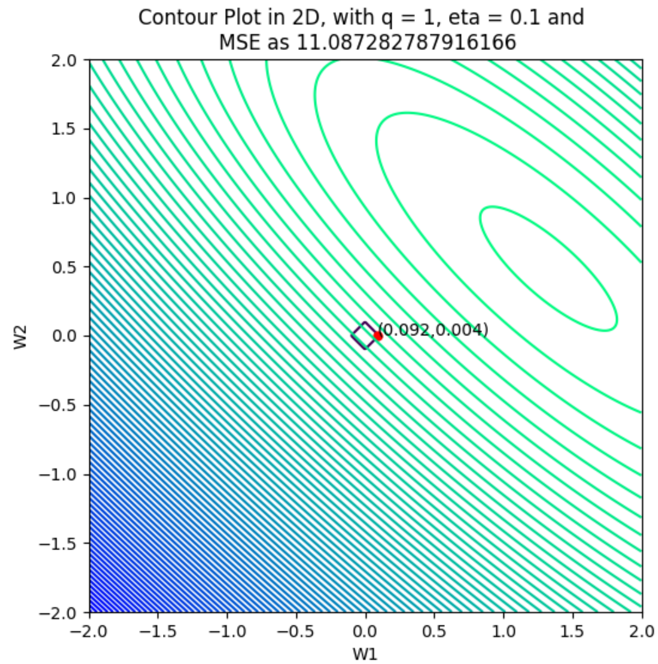
Assignment 1-C:

Unregularised error contour plot:



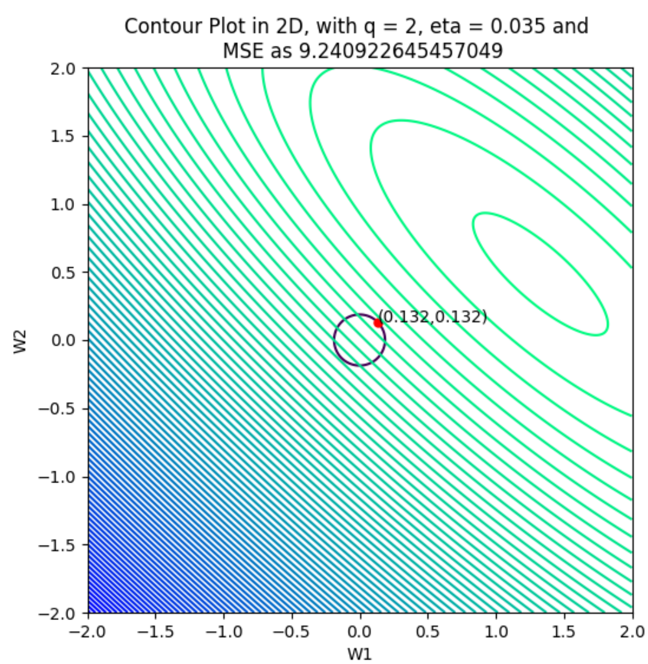
Minima of error function occurs at: $w_1 = 1.327$, $w_2 = 0.485$

Contour plot for error function with q = 1, η = 0.1



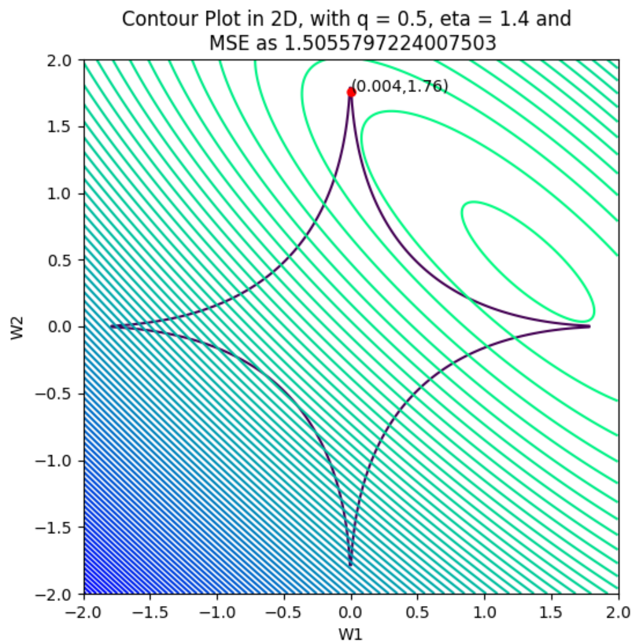
Minima of error function occurs at: $w_1 = 0.092$, $w_2 = 0.004$
Mean Squared Error = 11.0873

Contour plot for error function with q = 2, η = 0.035



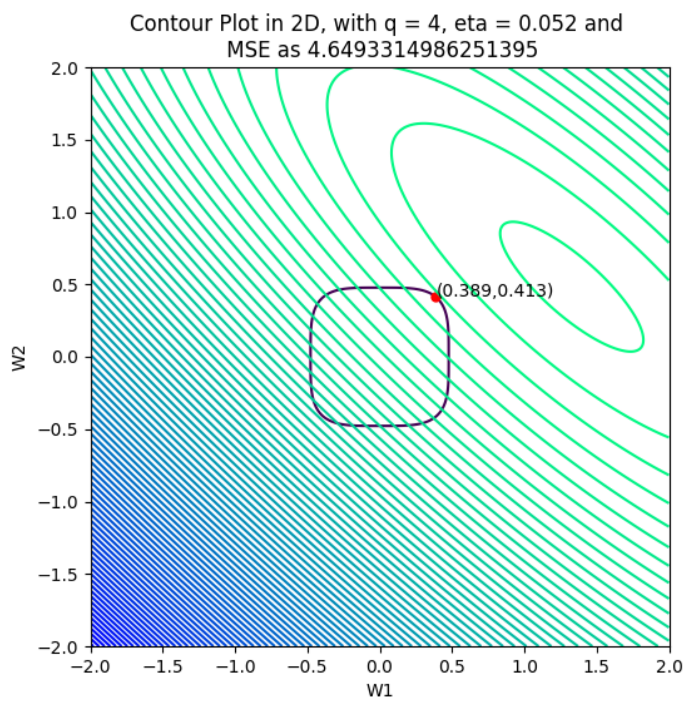
Minima of error function occurs at: $w_1 = 0.132$, $w_2 = 0.132$
Mean Squared Error = 9.2409

Contour plot for error function with q = 0.5, η = 1.4



Minima of error function occurs at: $w_1 = 0.004$, $w_2 = 1.176$
Mean Squared Error = 1.5055

Contour plot for error function with q = 4, η = 0.052



Minima of error function occurs at: $w_1 = 0.389$, $w_2 = 0.413$
Mean Squared Error = 4.6493