CS F320 Foundation of Data Science Assignment-1 Report

Introduction:

Description of the model: With the help of provided code, we can produce 6 different types of models varying in the algorithm they use -

1. Gradient Descent without Regularization.
2. Gradient Descent with Lasso(L1) regularization.
3. Gradient Descent with Ridge(L2) regularization.
4. Stochastic Gradient Descent without Regularization.
5. Stochastic Gradient Descent with Lasso(L1) regularization.
6. Stochastic Gradient Descent with Ridge(L2) regularization.

We can also modify the learning rate at which the model adapts as well as the number of iterations it runs for(if the system cannot handle many iterations).
We implemented the algorithms in the following way:

1. Initially, we read the data from the csv file provided using the pandas module present in python. Then we shuffle and normalize the data using appropriate methods.
2. Next we perform a 70-30 train-test split for training the model and testing it later.
3. Next we define two utility functions - generateTrainingDataPointsMatrix and generateTestingDataPointsMatrix. Their use will be evident when we explain the implementation of the actual algorithms itself.
4. In the actual algorithm named gradientDescent, we first develop a data_train matrix which has the terms associated with the coefficients of the degree of equation we use. That is-
   For 1st degree
   $$\text{data\_train} = \begin{bmatrix} 1 & s_1 & t_1 & -p_1 \\ 1 & s_2 & t_2 & -p_2 \\ . & . & . & . \\ . & . & . & . \end{bmatrix}$$
   For 2nd degree
   $$\text{data\_train} = \begin{bmatrix} 1 & s_1 & t_1 & s_1^2 & s_1t_1 & t_1^2 & -p_1 \\ 1 & s_2 & t_2 & s_2^2 & s_2t_2 & t_2^2 & -p_2 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix}$$
5. Next we create a column matrix with coefficients and one extra row with a 1

For first degree: $\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ 1 \end{bmatrix}$ For second degree: $\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ 1 \end{bmatrix}$

6. Now we matrix multiply coefficient matrix and data_train matrix to obtain a matrix which has error at every given training data point

$$\text{error matrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & s_1 & t_1 & -p_1 \\ 1 & s_2 & t_2 & -p_2 \\ . & . & . & . \\ . & . & . & . \end{bmatrix} = \begin{bmatrix} w_0 + w_1 s_1 + w_2 t_1 - p_1 \\ w_0 + w_1 s_2 + w_2 t_2 - p_2 \\ . \\ 0. \end{bmatrix}$$

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & s_1 & t_1 & s_1^2 & s_1 t_1 & t_1^2 & -p_1 \\ 1 & s_2 & t_2 & s_2^2 & s_2 t_2 & t_2^2 & -p_2 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix} = \begin{bmatrix} w_0 + w_1 s_1 + w_2 t_1 + w_3 s_1^2 + w_4 s_1 t_1 + w_5 t_1^2 - p_1 \\ w_0 + w_1 s_2 + w_2 t_2 + w_3 s_2^2 + w_4 s_2 t_2 + w_5 t_2^2 - p_2 \\ . \\ . \end{bmatrix}$$

7. Now this matrix can be used to calculate the errors required since it is just squaring of each term. This is directly available in numpy ndarray multiplication.
8. Now to create the slope which is the gradient matrix

$$\text{slope matrix} = \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \\ \frac{\partial E}{\partial w_2} \\ . \end{bmatrix}$$

9. If we expand this matrix, we get

Degree 1: $\begin{bmatrix} \sum_{n=1}^{N}(w_0 + w_1 s_n + w_2 t_n - p_n)(1) \\ \sum_{n=1}^{N}(w_0 + w_1 s_n + w_2 t_n - p_n)(s_n) \\ \sum_{n=1}^{N}(w_0 + w_1 s_n + w_2 t_n - p_n)(t_n) \end{bmatrix}$

Degree 1: $\begin{bmatrix} \sum_{n=1}^{N}\left(w_0 + w_1 s_n + w_2 t_n + w_3 s_n^2 + w_4 s_n t_n + w_5 t_n^2 - p_n\right)(1) \\ \sum_{n=1}^{N}\left(w_0 + w_1 s_n + w_2 t_n + w_3 s_n^2 + w_4 s_n t_n + w_5 t_n^2 - p_n\right)(s_n) \\ \sum_{n=1}^{N}\left(w_0 + w_1 s_n + w_2 t_n + w_3 s_n^2 + w_4 s_n t_n + w_5 t_n^2 - p_n\right)(t_n) \\ \sum_{n=1}^{N}\left(w_0 + w_1 s_n + w_2 t_n + w_3 s_n^2 + w_4 s_n t_n + w_5 t_n^2 - p_n\right)\left(s_n^2\right) \\ \sum_{n=1}^{N}\left(w_0 + w_1 s_n + w_2 t_n + w_3 s_n^2 + w_4 s_n t_n + w_5 t_n^2 - p_n\right)(s_n t_n) \\ \sum_{n=1}^{N}\left(w_0 + w_1 s_n + w_2 t_n + w_3 s_n^2 + w_4 s_n t_n + w_5 t_n^2 - p_n\right)\left(t_n^2\right) \end{bmatrix}$

There is a pattern here. It is basically summation of error at each point multiplied by values corresponding to one of our columns in the data_train matrix(1, s, t, s^2,etc). We can do this for all the data points and add them up to the gradient. Now for each coefficient we update this manually to create the slope matrix.

$$\frac{\partial E}{\partial w_1} = \text{Sum of all terms in } \begin{bmatrix} (w_0 + w_1 s_1 + w_2 t_1 - p_1)(s_1) \\ (w_0 + w_1 s_2 + w_2 t_2 - p_2)(s_2) \\ \vdots \\ \vdots \end{bmatrix}$$

First Degree

10. Now we can multiply the slope matrix with learning rate and subtract it from the old coefficients to get new coefficients. In this way we implement the algorithm.
11. In Ridge and lasso regularization, we just modify the slope matrix before applying the algorithm. This is done by adding a column matrix containing the lambda value(with appropriate sign) for lasso and lambda multiplied by corresponding coefficient value to slope matrix for ridge

$$\begin{bmatrix} \sum_{n=1}^{N}(w_0 + w_1 s_n + w_2 t_n - p_n)(1) \ + \ \lambda \\ \sum_{n=1}^{N}(w_0 + w_1 s_n + w_2 t_n - p_n)(s_n) \ + \ \lambda \\ \sum_{n=1}^{N}(w_0 + w_1 s_n + w_2 t_n - p_n)(t_n) \ + \ \lambda \end{bmatrix} \text{ for lasso}$$

$$\begin{bmatrix} \sum_{n=1}^{N}(w_0 + w_1 s_n + w_2 t_n - p_n)(1) \ + \ \lambda w_0 \\ \sum_{n=1}^{N}(w_0 + w_1 s_n + w_2 t_n - p_n)(s_n) \ + \ \lambda w_1 \\ \sum_{n=1}^{N}(w_0 + w_1 s_n + w_2 t_n - p_n)(t_n) \ + \ \lambda w_2 \end{bmatrix} \text{ for ridge}$$

Description of algorithms:

The regression model equation for 1 degree equation with 2 features is

$\hat{y} = w_0 + w_1 x_1 + w_2 x_2$

The cost function is(without regularization):

E(w) = $\frac{1}{2} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$

Gradient Descent algorithm is an optimization algorithm that is used to estimate the coefficients (w_j) that minimizes the given cost function. It is an iterative algorithm since it applies the same formula on the coefficients repeatedly until we reach a desired precision. The principle involved in Gradient Descent algorithm is

$W^{k+1} = W^k - \eta \sum_{n=1}^{N} \nabla(E(W))|_{W=W^k}$

Where k is the iteration value.

A variation of the above algorithm is devised since gradient descent requires computation of all the data points in every iteration of the algorithm. This is called Stochastic Gradient descent where in each iteration of the algorithm, we take a random data point instead of the whole dataset and calculate the gradient for it and apply the same algorithm.

Due to unboundedness of the coefficients, the model sometimes tries to overfit the data. This can be evident from the fact that as the degree of the polynomial increases, the training error might decrease but the testing error will increase. This means that the data is unable to generalize for future predictions.

This can be prevented if we can penalise the coefficients too while minimizing the cost function. This can be done by adding a regularizer term to the cost function and minimizing the new function so that coefficients do not wander as they want.

The order of penalizing the coefficients will give rise to two different types of regression. First order penalizing leads to Lasso regression while Second order peanilizing will give rise to Ridge regression.

E(w) = $\frac{1}{2} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 + \lambda \sum_{j=1}^{N} |w|_j$ => Cost function for Lasso regularization.

E(w) = $\frac{1}{2} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 + \lambda \sum_{j=1}^{N} w_j^2$ => Cost function for Ridge regularization.

Part - A:

Polynomial Regression using Gradient Descent and Stochastic Gradient Descent algorithms, without regularisation.

Gradient Descent			
Degree	Minimum Training Error	Minimum Testing Error	Learning rate
0	1145.3744	503.6494	0
1	228.9704	100.9422	0.9×10^{-5}
2	226.3398	100.4860	0.9×10^{-5}
3	211.0086	91.3444	0.9×10^{-5}
4	211.0037	91.3887	0.9×10^{-5}
5	210.9385	91.7772	0.4×10^{-5}
6	212.1706	92.6807	0.8×10^{-6}
7	213.9254	93.8459	0.33×10^{-6}
8	219.7798	97.4178	0.7×10^{-7}
9	275.2536	126.8930	0.13×10^{-7}

Stochastic Gradient Descent			
Degree	Minimum Training Error	Minimum Testing Error	Learning rate
0	1157.1651	490.9964	0.1×10^{-2}
1	228.9716	100.3682	0.1×10^{-1}
2	231.4013	103.5476	0.1×10^{-1}
3	211.41	90.8662	0.1×10^{-1}
4	213.7681	93.0982	0.5×10^{-3}
5	226.3973	99.0446	0.5×10^{-4}
6	226.1247	100.1483	0.5×10^{-4}
7	710.0590	319.3334	0.2×10^{-5}
8	8706.7987	4450.7042	0.2×10^{-6}
9	11762.5367	5821.6145	0.2×10^{-6}

Part - B:

Polynomial Regression of 9th degree with Ridge and Lasso regularisations.

Algorithm	ln(Lambda)	Minimum Training Error	Minimum Testing Error
Ridge Regression (GD)	-10	260.2847	120.5289
	-1000	275.2527	126.8954
	-40	253.1920	115.2936
	10	20918	6765
	-20	257.2384	118.3833
Lasso Regression (GD)	-10	264.8957	121.9348
	-1000	279.2537	127.6754
	-40	257.4502	116.3462
	10	22048	7036
	-20	260.5639	121.4568
Ridge Regression (SGD)	-10	12447.355	5738.27
	-1000	13953.7302	6123.6378
	-40	14489.0605	7007.2081
	10	20135.8934	8444.3952
	-20	11895.342	5208.3298
Lasso Regression (SGD)	-10	14367.835	5632.87
	-1000	14054.7697	6150.6362
	-40	14356.0985	7087.5621
	10	21363.8924	8364.3732
	-20	12035.972	5318.1739

Conclusion:

Part A:

1. Out of all the polynomials, we have fixed the number of iterations and changed the learning parameter so that we can compare the models for each degree.
2. Upon viewing the results, we can see that third degree has the minimum testing error therefore **3rd degree** is the best model.
3. We can also observe that as the degree is increasing, the testing error is increasing(training error is increasing here since we have fixed the number of iterations. If we increase the number of iterations, we can achieve a more minimum training error), indicating the overfitting nature of the model to the training data. This can be seen by the visualization through the plots provided too.

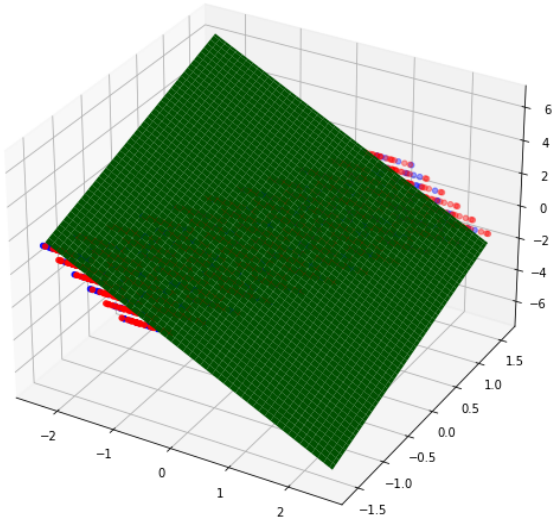
Part B:

1. In both Ridge and lasso regression, a high log lambda value(positive) resulted in high training and testing error since it makes the graph too flat. This is due to the high influence of lambda in the cost function. On the other hand, with a low log lambda value (log lambda -> negative infinity), the model predicts almost the same values as the model without any regularization. This is due to the fact that lambda is so small that regularization has become insignificant in the cost function.
2. The coefficients are already very very small in gradient descent itself. Since regularization is all about constraining the values and since they are already small, both regularized and un-regularized algorithms yield very similar values.

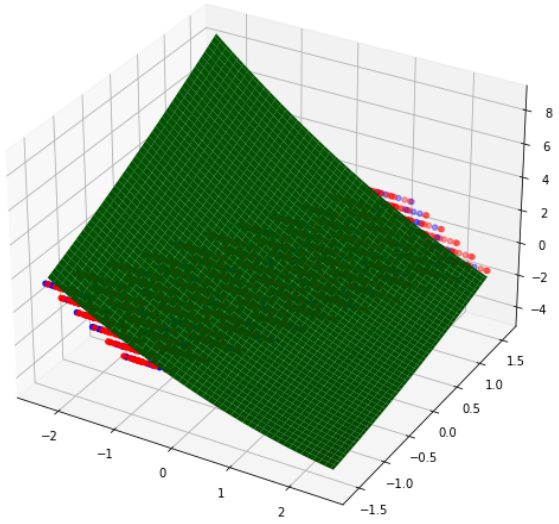
Overall, the model generated in part A is a better choice than part B(although the difference is very marginal). The very nature of the data set provided is the main reason that both regularized and un-regularized models yield results with very small differences.

Plots:

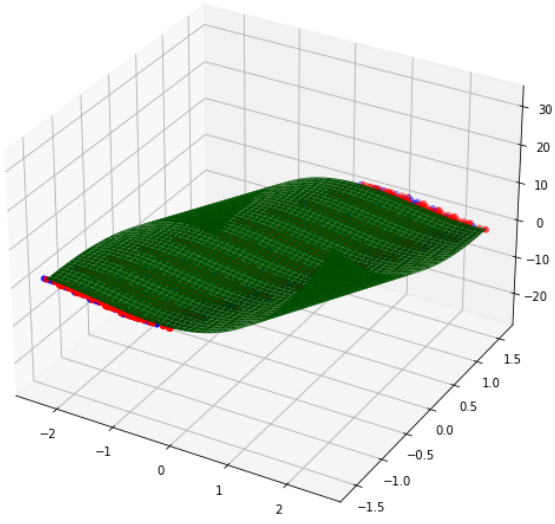
GD degree-1:



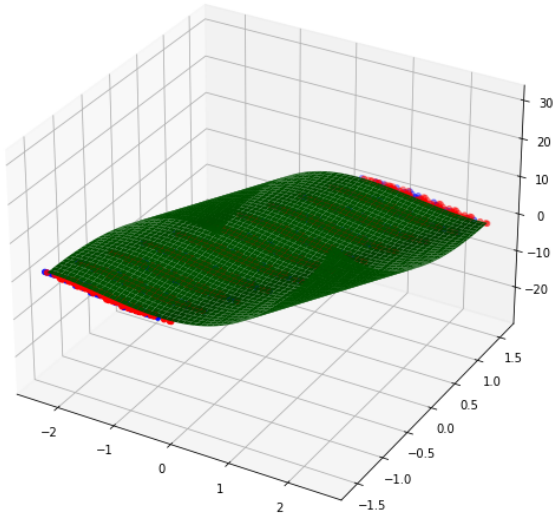
GD degree-2:



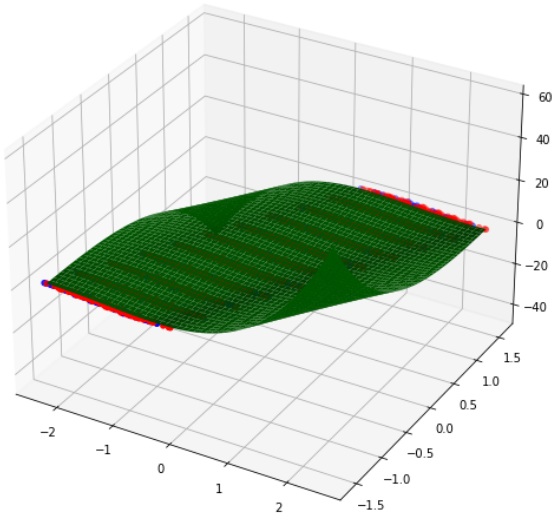
GD Degree-3:



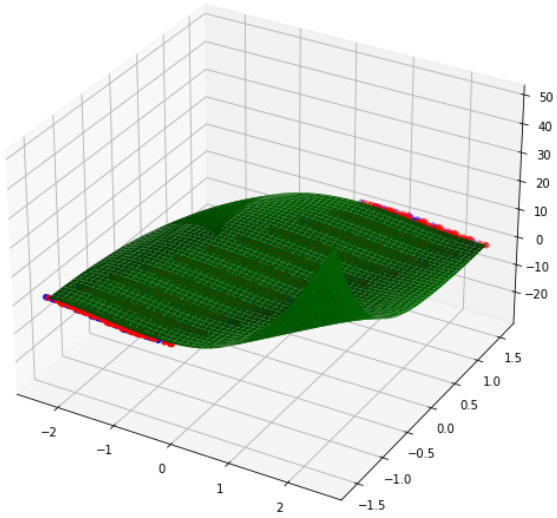
GD Degree-4:



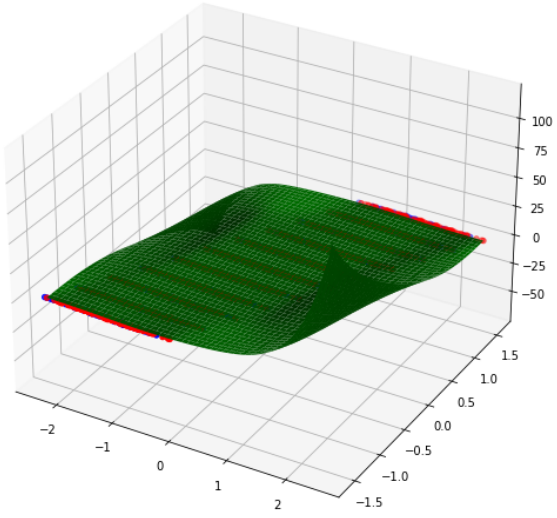
GD Degree-5:



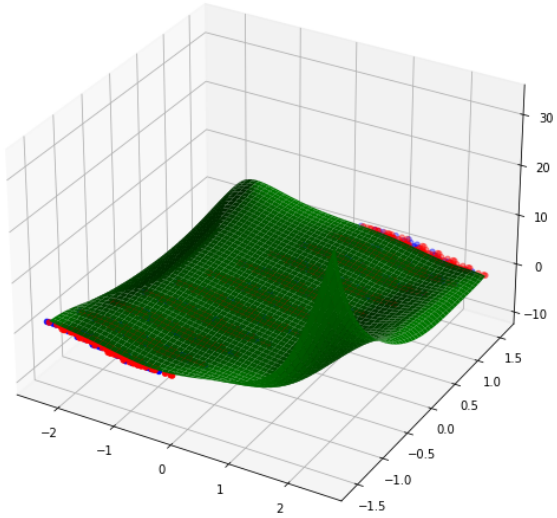
GD Degree-6:



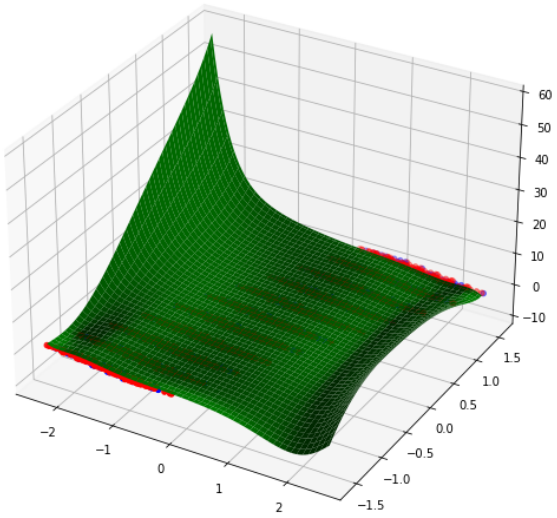
GD Degree-7:



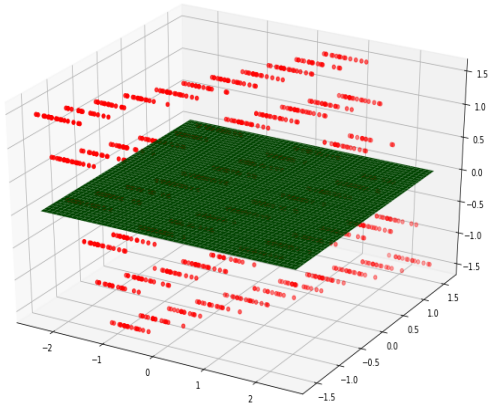
GD Degree-8:



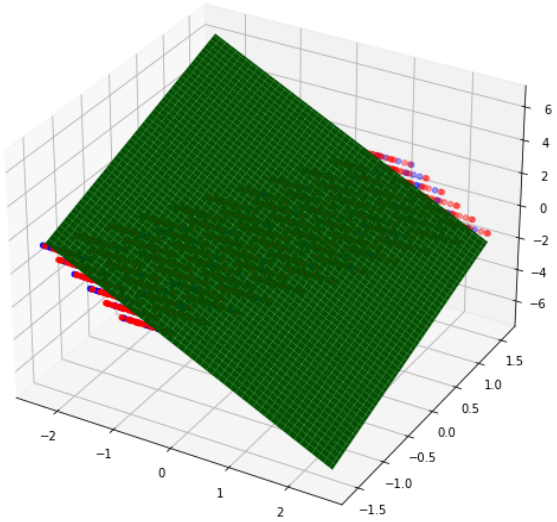
GD Degree-9:



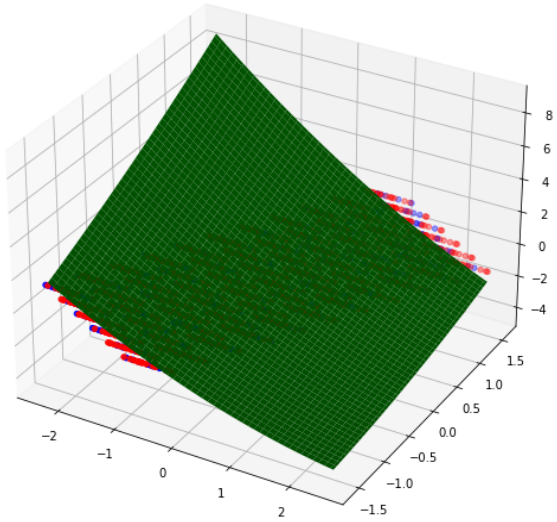
GD Degree-0:



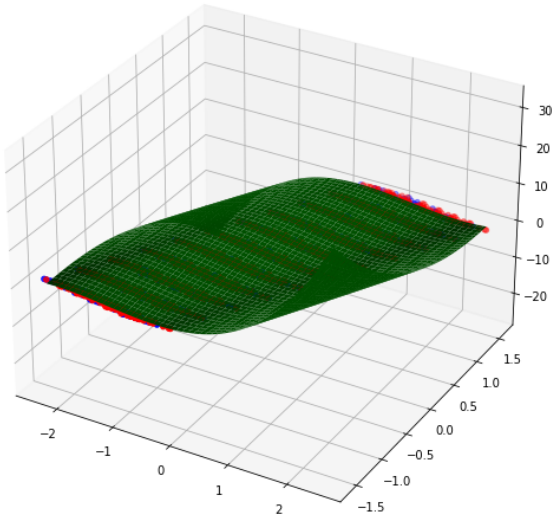
SGD degree-1:



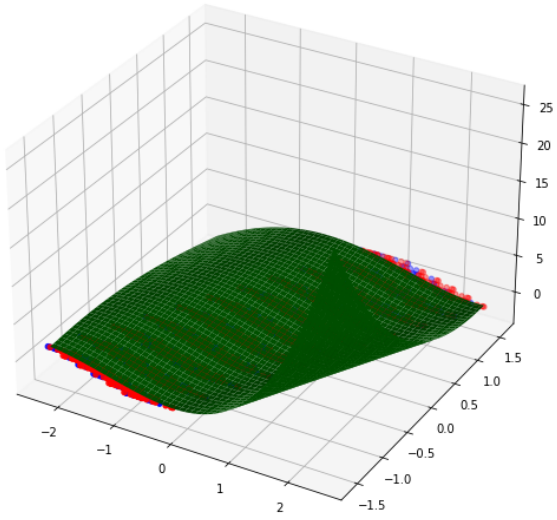
SGD degree-2:



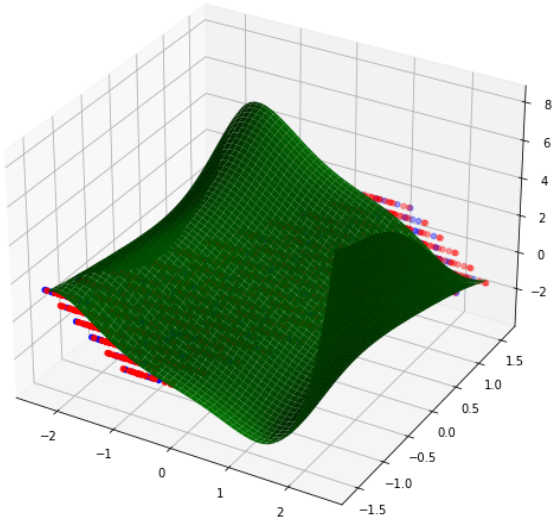
SGD degree-3:



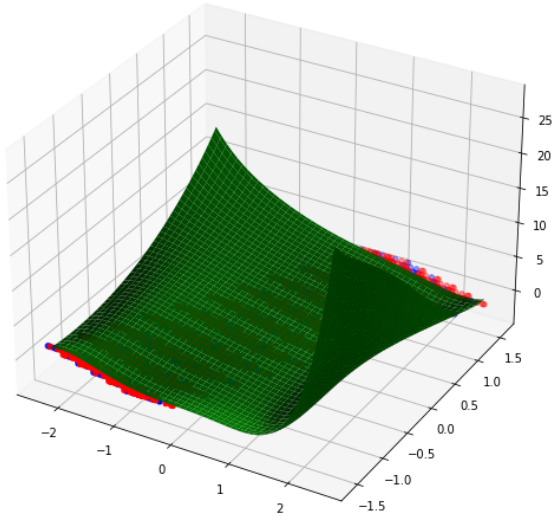
SGD degree-4:



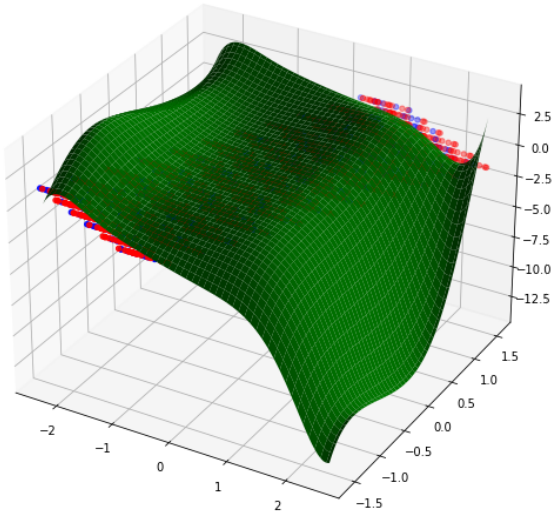
SGD degree-5:



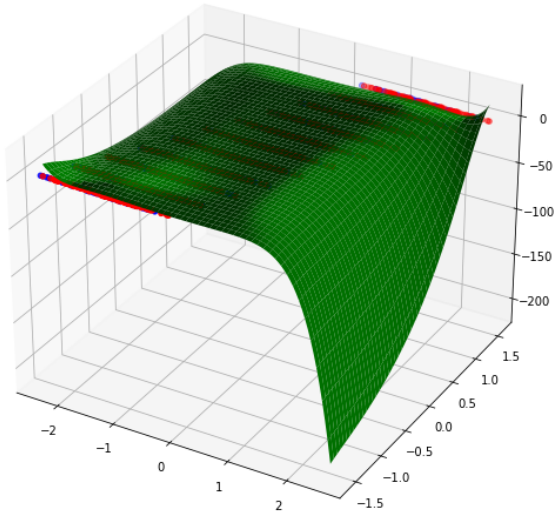
SGD degree-6:



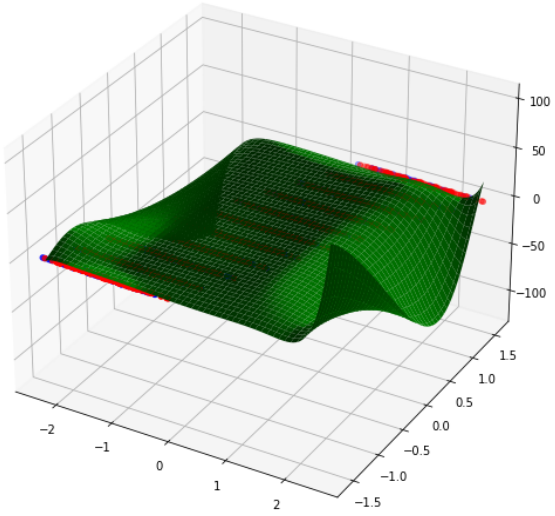
SGD degree-7:



SGD degree-8:



SGD degree-9:



SGD degree-0: