Polynomial Curve Fitting and Handwritten Digit Recognition

Aditi Khandelwal (2018EE10434) , Kamna Meena (2018EE10470)

April 15, 2021

Polynomial Curve Fitting

For all the experiments data was normalized because the data was on very different scales. The order of y was high in comparison to the order of x and degrees of x. If we don't normalize the data, the contours are narrow and tall and it takes longer to converge. Z-Score Scaling was done on the data. The z- score transform makes the data zero mean with variance 1.Figure 1

$$X = \frac{X - \mu}{\sigma}$$

where μ =0 and σ =1.

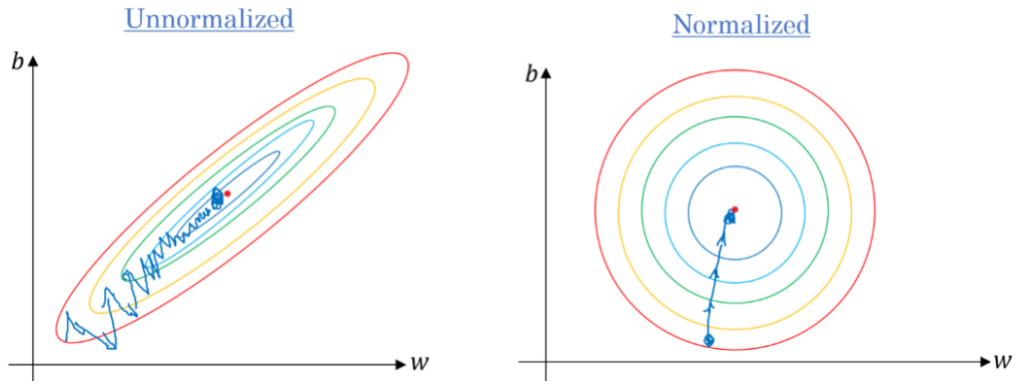


Figure 1: Gradient Descent on Normalized and Unnormalized Contour Plots[1]

Experiments on First 20 Data points

These data points were randomly split into 80:20 ratio. 16 points were used for training and 4 for validation
Loss Functions used[2]:

- Mean Absolute Error: It is the mean of absolute sum of the difference in predicted y and true target values.

$$MAE = \frac{\sum_i |y_i - y_i^p|}{N}$$

- Log Cosh Error: We have taken the mean of summation of logarithmic hyperbolic cosine of the difference in predicted y and true target values.

$$LogCoshError = \frac{1}{N}\sum_i log(cosh(y_i^p - y_i))$$

- Huber Loss*: It is smoother MAE formulation. It is acts like MSE when error is small and MAE when error is large, making itself differentiable at zero.

$$HuberLoss = \begin{cases} \frac{1}{2}(y_i^p - y_i)^2 & when |y_i - y_i^p| \leq \delta, \\ = \delta|y_i - y_i^p| - \frac{1}{2}\delta^2, & ow \end{cases}$$

The above listed error functions are all convex functions.

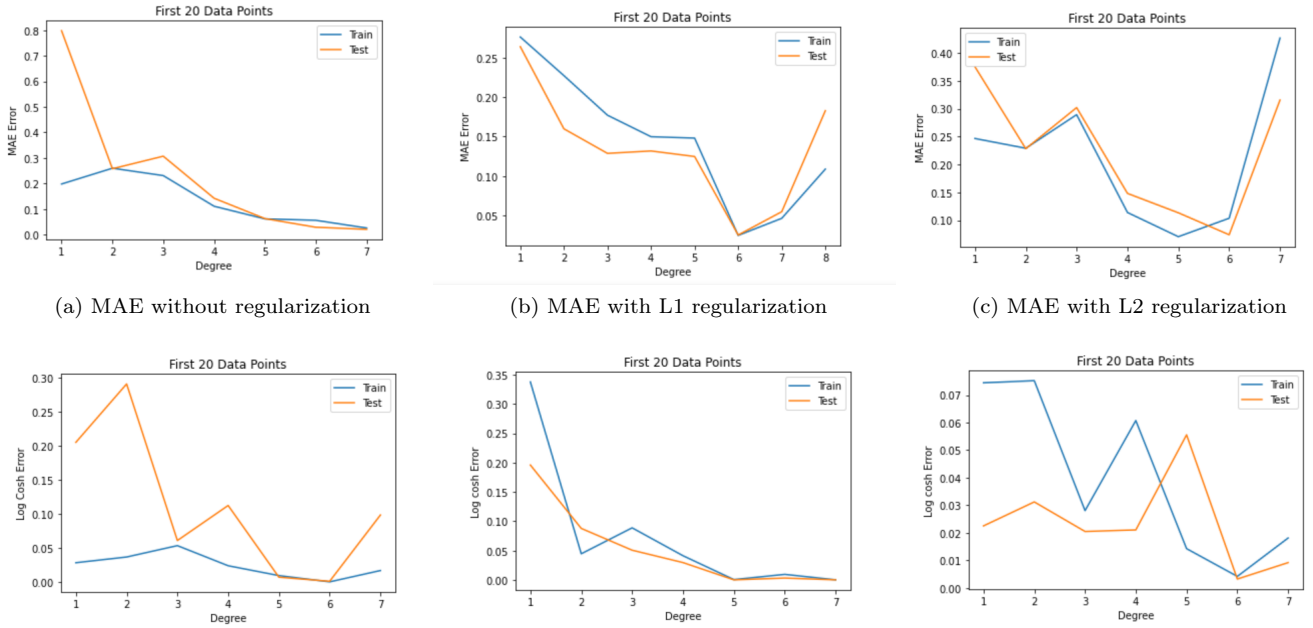
The estimate of the noise variance is equal to the MSE Loss that is:

$$MSE = \frac{\sum_i (y_i - y_i^p)^2}{N}$$

Accuracy of the regressors were measured using the r2 Score (Coefficient of Determination). It gives us the proportion of variance in the target that our regressor is able to explain by the regression variables. This score becomes negative when the predictions are worse than the mean[3].

$$Score = 1 - \frac{\sum_i (y_i^{True} - y_i^p)^2}{\sum_i (y_i^{True} - y_{mean}^p)^2}$$

Following plots were obtained:-



(a) MAE without regularization (b) MAE with L1 regularization (c) MAE with L2 regularization

(d) Log Cosh Error without regularization (e) Log Cosh Error with L1 regularization (f) Log Cosh with L2 regularization

Figure 2: Results

Looking at the above plots, we observe that polynomial of degree 5 or 6 are better than lower or higher degrees. It is unexpected that at degree 7, the error in training and testing increases which can be explained as the number of features increase, the degree increases exponentially which leads to numerical errors like underflow or overflow. These errors are being observed when we train the model at degrees 7 and above where the training error starts to rise up in contrary to what we expect. For the given data, we do not observe any overfitting as such, the regularization plays a role when the variance in the model is high. So when we regularize, we are only able to observe that L2 decreases the high variance problem for MAE but L1 makes the performance poorer since we are adding a penalty term in the objective which is not required and increases the error. Similar thing can be observed in log cosh error formulation as well.

So we can finally conclude that polynomials of degree less than 5 underfit. Polynomials of degree 5 and 6 are good fit for the data and polynomials of degree above 7 should have overfitted but because of numerical error, we are getting high error for degrees beyond 6.

The Noise Variance is reported as follows:

Error Formulation	Training Noise Variance Estimate	Testing Noise Variance Estimate	Error on Training	Error on Testing	r2 Score on Training	r2 Score on Testing
Degree 4						
MAE without Regularization	0.0339	0.0246	0.111	0.1424	0.9647	0.9665
MAE with L1 Regularization (LR = 0.01)	0.05223	0.0247	0.1495	0.1314	0.9461	0.9701
MAE with L2 Regularization (LR = 0.01)	0.0350	0.0229	0.0828	0.1171	0.9580	0.9780
Log Cosh without Regularization	0.0486	0.2600	0.0239	0.1125	0.9251	0.8524
Log Cosh with L1 Regularization	0.0870	0.0608	0.0412	0.0296	0.9147	-14.11
Log Cosh with L2 Regularization	0.115	0.0292	0.0541	0.01449	0.8870	0.9483
Degree 5						
MAE without Regularization	0.0149	0.0055	0.0617	0.0628	0.9867	0.9829
MAE with L1 Regularization (LR = 0.01)	0.0465	0.0184	0.1477	0.1243	0.9499	0.9793
MAE with L2 Regularization (LR = 0.01)	0.0032	0.0130	0.0361	0.0790	0.9949	0.9919
Log Cosh without Regularization	0.01918	0.0142	0.0094	0.0070	0.9810	0.9799
Log Cosh with L1 Regularization	0.0014	0.0004	0.0007	0.0002	0.9985	0.9994
Log Cosh with L2 Regularization	0.0213	0.1103	0.01060	0.05188	0.97040	0.9301
Degree 6						
MAE without Regularization	0.0115	0.0022	0.0562	0.0289	0.9892	0.9958
MAE with L1 Regularization (LR = 0.0001)	0.0026	0.0005	0.0247	0.0252	0.9983	0.9994
MAE with L2 Regularization (LR = 0.01)	0.0086	0.0009	0.1037	0.0741	0.9926	0.9983
Log Cosh without Regularization	0.0004	0.0022	0.0002	0.0011	0.9995	0.9975
Log Cosh with L1 Regularization	0.0193	0.0066	0.0096	0.0033	0.9800	0.9919
Log Cosh with L2 Regularization	0.0051	0.0032	0.0042	0.0032	0.9950	0.9941

To assess how the loss function converges with different learning rates, we plotted the following curve on different learning rate, 0.001, 0.01 and 0.1.

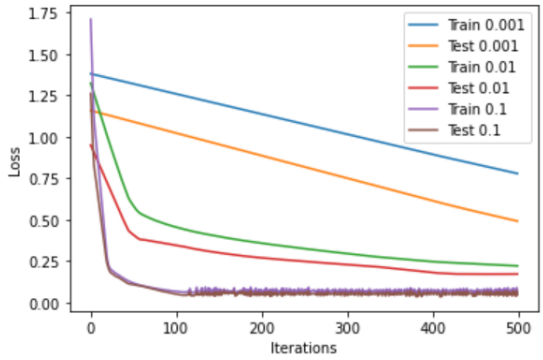
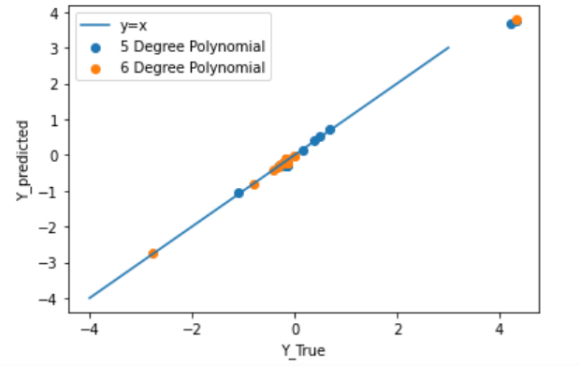


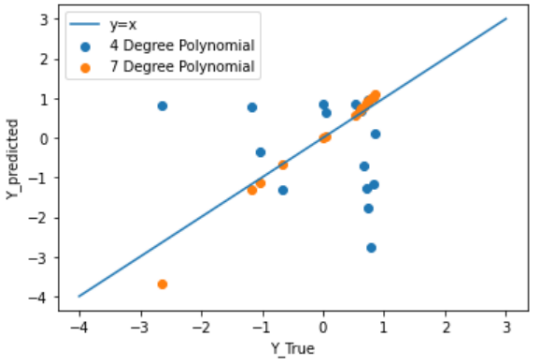
Figure 3: Effect of increasing the learning rate on convergence

Goodness of fit of 5 and 6 degree polynomial in comparison to 4 and 7 degree.

Training Data Points:

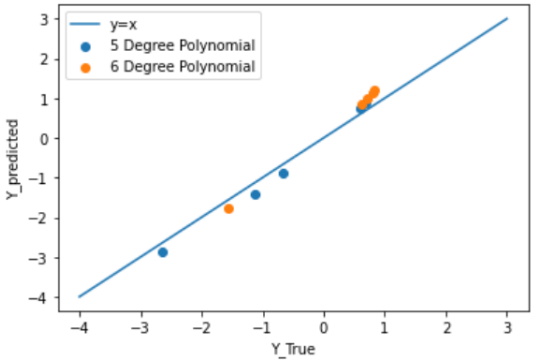


(a) 5 and 6 Degree Polynomial on Training Data Points

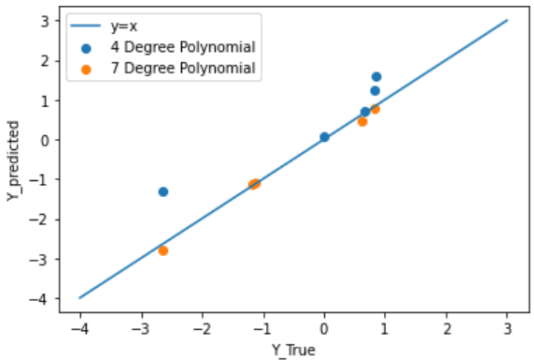


(b) 4 and 7 Degree Polynomial on Training Data Points

Testing Data Points:



(c) 5 and 6 Degree Polynomial on Testing Data Points

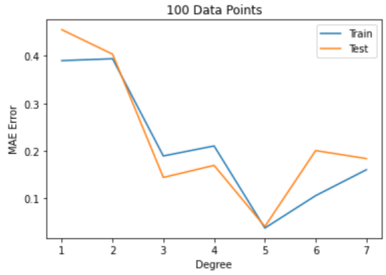


(d) 4 and 7 Degree Polynomial on Testing Data Points

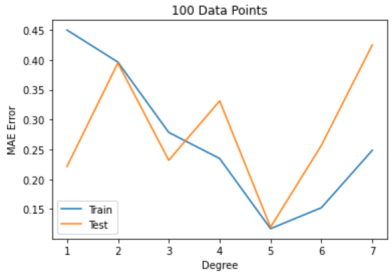
Figure 4: Goodness of fit by observing on Training and Testing Sets

Experiments on 100 Data Points

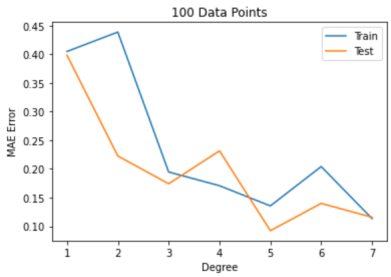
Following plots were obtained:



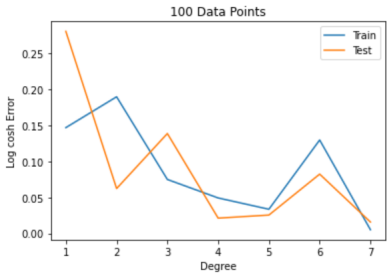
(a) MAE without regularization



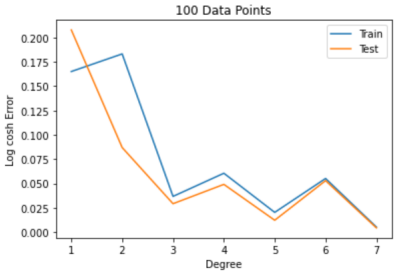
(b) MAE with L1 regularization



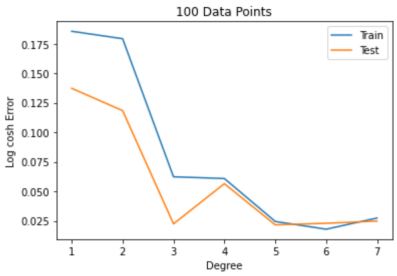
(c) MAE with L2 regularization



(d) Log Cosh without regularization



(e) Log Cosh with L1 regularization



(f) Log Cosh with L2 regularization

Figure 5: Results

As explained for the 20 data points also, if we see the plots above and data in the table we find that polynomial of degree 5 best fits the data. Estimate of Noise Variance can be found in the following table:

Error Formulation	Training Noise Variance Estimate	Testing Noise Variance Estimate	Error on Training	Error on Testing	r2 Score on Training	r2 Score on Testing
Degree 4						
MAE without Regularization	0.2378	0.0846	0.2105	0.1696	0.7802	0.8646
MAE with L1 Regularization (LR = 0.01)	0.3111	0.4802	0.2347	0.3311	0.6480	0.6563
MAE with L2 Regularization (LR = 0.01)	0.0884	0.1574	0.1708	0.2316	0.9054	0.8709
Log Cosh without Regularization	0.1095	0.0433	0.0492	0.0212	0.9080	0.6488
Log Cosh with L1 Regularization	0.1453	0.1121	0.0604	0.0489	0.8463	0.9003
Log Cosh with L2 Regularization	0.0711	0.0593	0.0608	0.0564	0.9395	0.7364
Degree 5						
MAE without Regularization	0.0071	0.0049	0.0377	0.0409	0.9935	0.9909
MAE with L1 Regularization (LR = 0.01)	0.0622	0.0319	0.1172	0.1197	0.9389	0.9598
MAE with L2 Regularization (LR = 0.01)	0.0485	0.0058	0.1356	0.0924	0.9600	0.9066
Log Cosh without Regularization	0.0706	0.0522	0.0335	0.0255	0.9346	0.9162
Log Cosh with L1 Regularization	0.0411	0.0243	0.0202	0.0121	0.9657	0.7819
Log Cosh with L2 Regularization	0.0165	0.0105	0.0244	0.0216	0.9841	0.9866
Degree 6						
MAE without Regularization	0.0390	0.1491	0.1059	0.2008	0.9514	0.9053
MAE with L1 Regularization (LR = 0.01)	0.0373	0.1549	0.1523	0.2568	0.9650	0.7717
MAE with L2 Regularization (LR = 0.01)	0.1063	0.0319	0.2042	0.1400	0.9053	0.9207
Log Cosh without Regularization	0.3329	0.1811	0.1294	0.0822	0.7093	0.5089
Log Cosh with L1 Regularization	0.1171	0.1088	0.0550	0.0528	0.8768	0.8965
Log Cosh with L2 Regularization	0.0168	0.0281	0.0179	0.0229	0.9849	0.9256

Convergence with different Learning Rate on 100 Data Points:

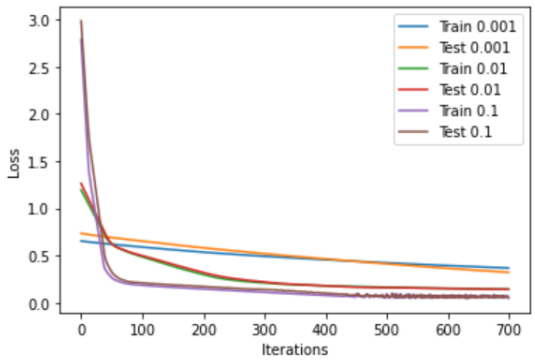
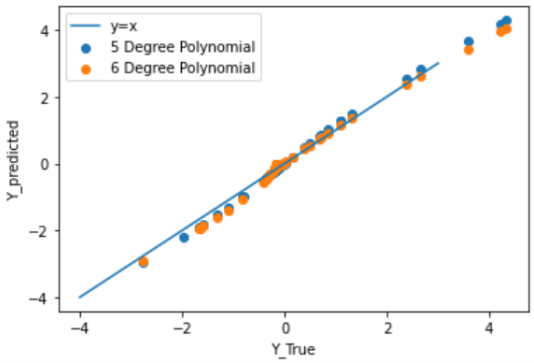


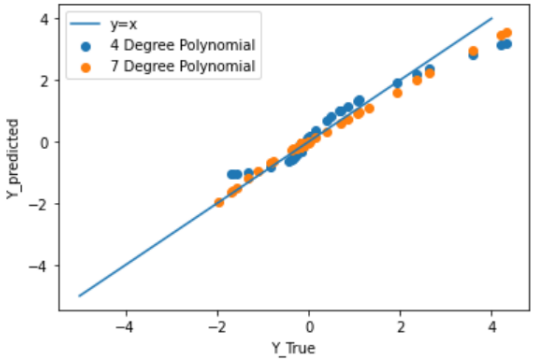
Figure 6: Effect of increasing the learning rate on convergence

Goodness of the fit for 100 Data Points:

Training Data Points:

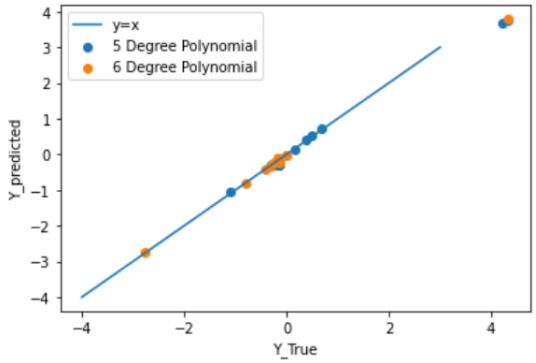


(a) 5 and 6 Degree Polynomial on Training Data Points

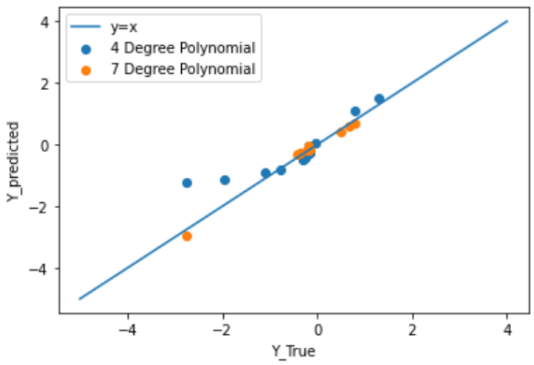


(b) 4 and 7 Degree Polynomial on Training Data Points

Testing Data Points:



(c) 5 and 6 Degree Polynomial on Testing Data Points



(d) 4 and 7 Degree Polynomial on Testing Data Points

Figure 7: Goodness of fit by observing on Training and Testing Sets

Understanding L1 and L2 Regularization:

This plots were generated using the Log Cosh Function to see the effect of regularization.

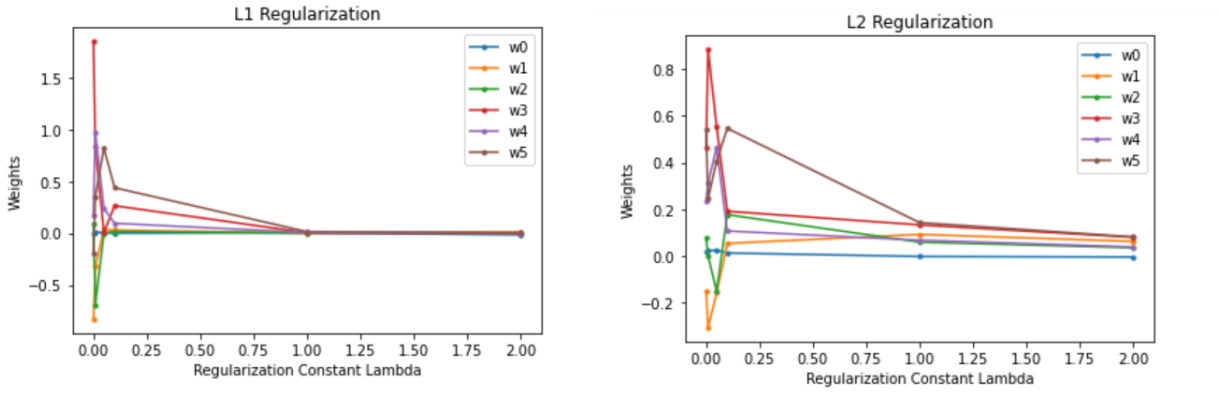


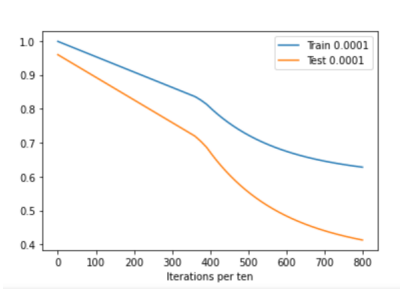
Figure 8: Comparison b/w L1 and L2 Regularization

From the above plots we can infer:

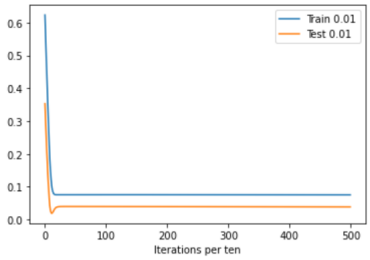
- In the L1 Regularization plot, we see that most the weights are zero, so L1 regularization behaves like feature selector making some weights zero so those features are ignored in $w^T x$ term. Noise was observed in the gradient descent with L1, which can be explained by the non differentiable nature of the convex function.
- L2 regularization, we see that it reduces the weights but keeps most of the features. This regularization helps in reducing the variance of the model making the overfitting models to fit best.

Results of Huber Loss:

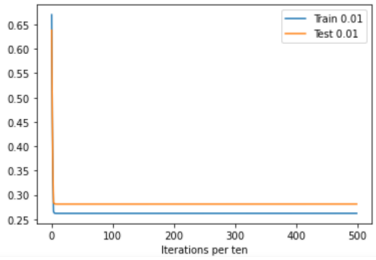
We also tried to experiment with huber loss, but despite tuning the hyperparameters for the regressor, we witnessed a problem of exploding gradients. Following are the learning curves obtaining for polynomial of degree 1.



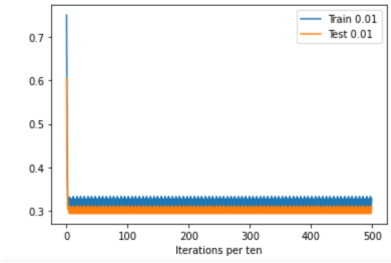
(a) Huber Loss without regularization



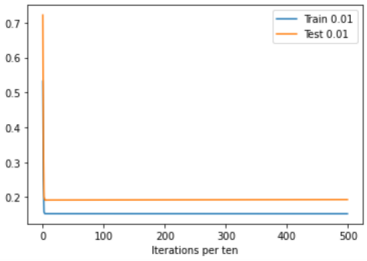
(b) Huber Loss with L1 regularization



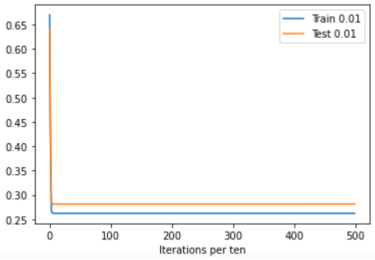
(c) Huber Loss with L2 regularization



(d) Huber Loss without regularization



(e) Huber Loss with L1 regularization



(f) Huber Loss with L2 regularization

Figure 9: Results

Error Formulation	Training Noise Variance Estimate	Testing Noise Variance Estimate	Error on Training	Error on Testing	r2 Score on Training	r2 Score on Testing
20 Data Points						
Huber Loss without Regularization	2.181	1.1966	0.6279	0.4131	-3.2539	-3.2014
Huber Loss with L1 Regularization (LR = 0.01)	0.1574	0.0770	0.075	0.0385	0.7047	0.7371
Huber Loss with L2 Regularization (LR = 0.01)	1.84	1.439	0.613	0.566	-2.237	-12.47
100 Data Points						
Huber Loss without Regularization	0.9536	0.6905	0.3281	0.3084	-0.474	-2.0072
Huber Loss with L1 Regularization (LR = 0.01)	0.6599	0.5064	0.1525	0.1930	-0.3670	0.066
Huber Loss with L2 Regularization (LR = 0.01)	0.8868	1.2600	0.2623	0.28142	-0.954	-0.909

Handwritten Digit Recognition

We are given a data set containing 5000 training examples of handwritten digits(0 to 9) of size (28, 28) each where each pixel is represented by floating point number indicating the grayscale intensity at that location. We read the data into a matrix X of size 5000×784 where every row is a training example for a handwritten digit image.
The second part of the training set contains corresponding labels for the training set which we stored in y of size 5000×1.
Number of examples(m) = 5000
Number of features(n) = 784 + 1 (including the bias term)
There are 10 labels/classes(0 to 9). We will make use of one-vs-all classification technique by training 10 different logistic regression classifiers. We implemented vectorized logistic regression as it is more efficient than using for-loops.
We randomly splitted the data set into training and testing set in the ratio of 8:2, i.e, there will be 4000 training examples and 1000 testing examples is.

We have used the following performance metrics:

- Confusion Matrix: It tells us about the classes our classifier is getting confused with.
- Precision: Suppose for class A, it tells us about what percentage of predicted variable is actually 'A'.
- Recall: We get some true values, but how much the classifier is able to recall the true values.
- F1 Score: It is the harmonic mean of the precision and recall values.

1 Without Regularization

Since we have 10 different models, we needed to find the optimal parameters for each model using gradient descent algorithm. Our fit() function returns all the classifier learned parameters in a matrix W of size (10, 785), where each row of W corresponds to the learned logistic regression parameters for one class. Using parameter matrix W, we make predictions on the training set and testing set, and determine the corresponding accuracy.

1.1 Results

We observed that cost is decreasing every iteration which shows that parameters are being learned.

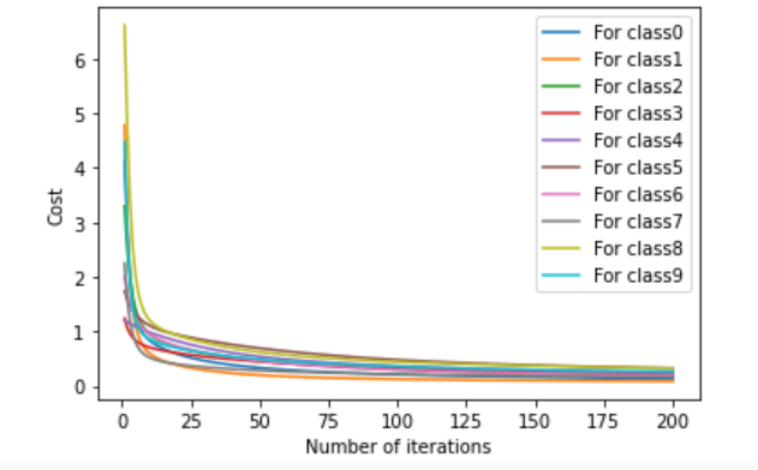


Figure 10: Learning Curves for 10 Classes

Test Data Stats :

	class	Precision	Recall	F1 Score
0	0.0	0.099	0.095	0.09696
1	1.0	0.097	0.111	0.10353
2	2.0	0.103	0.101	0.10199
3	3.0	0.098	0.105	0.10138
4	4.0	0.108	0.104	0.10596
5	5.0	0.096	0.077	0.08546
6	6.0	0.085	0.093	0.08882
7	7.0	0.112	0.103	0.10731
8	8.0	0.100	0.105	0.10244
9	9.0	0.102	0.106	0.10396

Figure 11: Performance metric

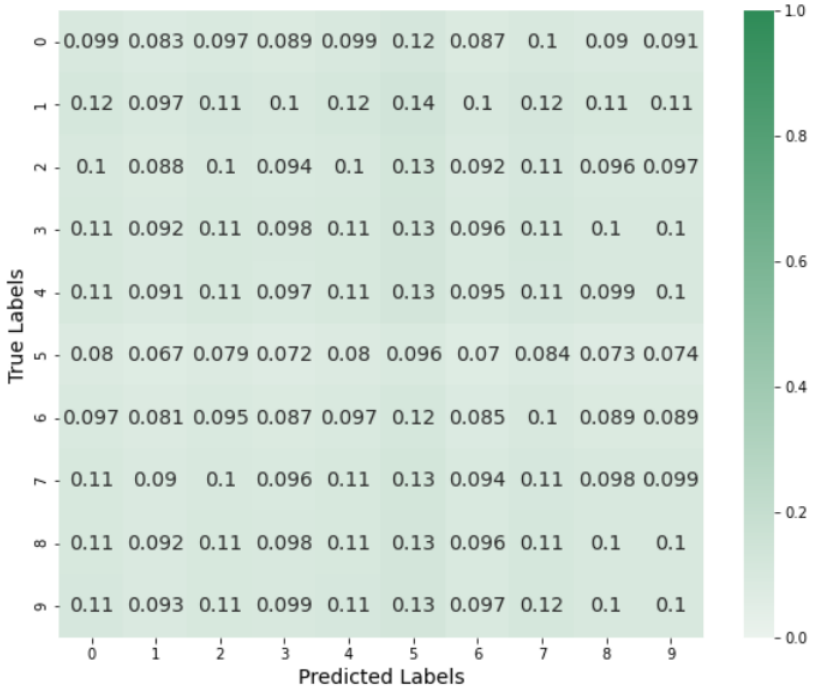


Figure 12: Confusion matrix

1.2 Effect of number of iterations on the model

Learning rate = 0.1

Number of iterations	Training set accuracy	Testing set accuracy
10	15.45	15.4
50	43.475	42.5
100	61.5	60.4
500	82.625	77.6
1000	87.75	83.8

Training and testing set accuracy increases with increase in number of iterations.

1.3 Effect of Learning rate on the model

Number of iterations = 100

Learning rate	Training set accuracy	Testing set accuracy
0.0001	10.4	8.8
0.001	5.6	5.9
0.01	16.4	16.5
0.1	63.35	62.2
1	87.35	82.1
10	87.725	82.9
100	78.7	73.4

If learning rate is too large, the cost function converges but it overshoots the minima and increases in a way that error increases.
So, Learning rate should be sufficiently small and sufficiently large such that cost function approaches its

global minima.
But if learning rate is too small, then the learning is slow as it requires very large number of epochs to converge.

2 With Regularization

To reduce the problem of overfitting, we add a regularization term to the cost function. Regularization parameter (λ) controls the trade between the two goals, one being, fitting the training set well, and other, keeping the parameters small and therefore, keeping hypothesis relatively simple to avoid overfitting.
w_0 is not considered for regularization.

$$Cost(w) = \frac{1}{m}\sum_{i=1}[y_i log(h(x_i)) + (1-y_i)log(1-h(x_i))] + \frac{\lambda}{2m}\sum_{j=1}^{n}(w_j)^2$$

Gradient of the cost function:

$$w_0 = \frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)x(0)_i$$

And, for j = 1, 2, 3, ..., n

$$w_j = \frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)x(j)_i + \frac{\lambda}{m}w_j$$

2.1 Results

We added the quadratic penalty term for L2 regularization with regularization constant = 10. We obtain the following results.

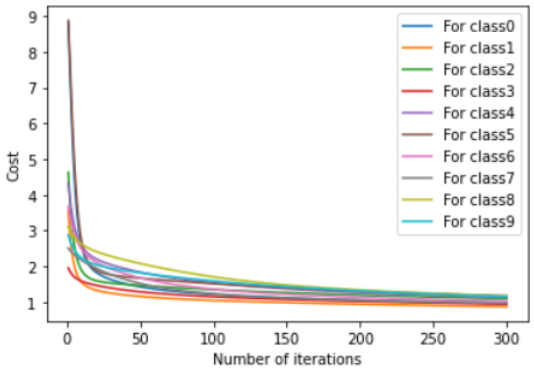


Figure 13: Learning Curves for 10 Classes

Test Data Stats :

	class	Precision	Recall	F1 Score
0	0.0	0.099	0.091	0.09483
1	1.0	0.097	0.121	0.10768
2	2.0	0.103	0.091	0.09663
3	3.0	0.098	0.117	0.10666
4	4.0	0.108	0.103	0.10544
5	5.0	0.096	0.060	0.07385
6	6.0	0.085	0.092	0.08836
7	7.0	0.112	0.123	0.11724
8	8.0	0.100	0.103	0.10148
9	9.0	0.102	0.099	0.10048

Figure 14: Performance metric

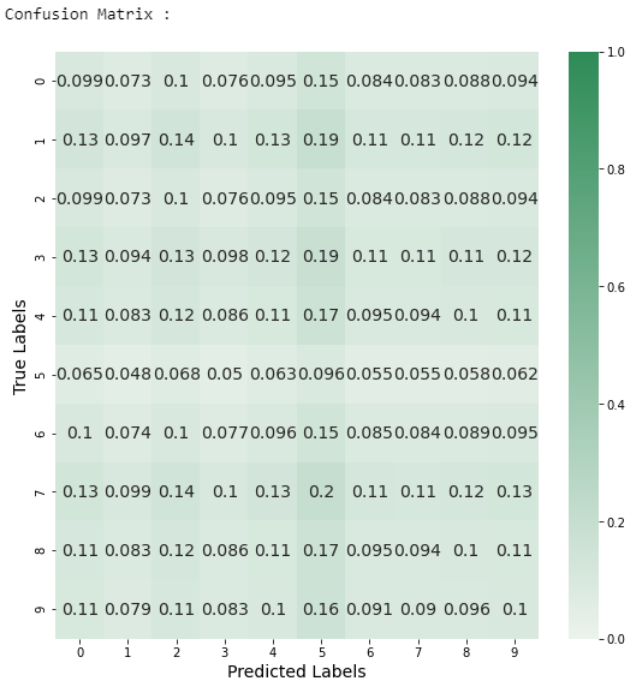


Figure 15: Confusion Matrix

2.2 Effect of regularization parameter on the model

Number of iterations = 100, and Learning rate = 0.1

Regularization Parameter	Training set accuracy	Testing set accuracy
0.001	44.925	41.2
0.01	48.925	46.2
0.1	44.3	43.4
1	44.675	41.0
10	44.375	43.6
100	51.5	52.2

Regularization plays no effect in improving the accuracy of our classifier since an extra penalty term is being added to the objective function which is not desired.

References

[1] https://medium.com/@pnkshir/concept-of-normalization-427108d1ccfa

[2] https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4

[3] https://towardsdatascience.com/tagged/r-squared#:~:text=What%20is%20R%2DSquared%3F, predicts%20none%20of%20the%20variance.