DA2401 Endsem Report

Athreya Ganesh

DA24B002

1 Introduction

This report details my observations and evaluations on a multi-class classification designed on MNIST data to accurately predict handwritten digits from 0 to 9.

I have tried Softmax Regression, Perceptron Classifier, and KNN classifier. I observed that KNN Classifier works the best when given the raw data, producing a F1 score of just above 0.95 when the value of k is set to be 5, outperforming all other models by a huge margin.

2 System Architecture

I used the superior performance of the KNN model, combined it with preprocessed data using PCA, and reduced the dimensions to make it run smoother and faster.

2.1 Data Preprocessing

- Raw pixel values (0-255) were converted to the range (0-1)
- PCA is applied to the 784-dimensional data to reduce it to only the 100 principle components, to retain as much variance as I can.

2.2 Model Training and Prediction

- KNN is selected due to its high raw output on unaltered MNIST data, compared to softmax and perceptron.
- KNN model is trained on the PCA dataset, which takes very little time.
- The model for KNN Proceeds with the tuned hyperparameter $k = 7$ to predict labels for every datapoint in the validation set.

3 Hyperparameter Tuning for Each Model

3.1 Softmax Regression

The hyperparameter runing I've done for softmax regression have been tabulated below (with no data preprocessing):

learning_rate	n_iterations	f1_score	accuracy	time
0.01	10000	0.8982	0.8995	54 sec
0.001	10000	0.8592	0.8615	54 sec
0.001	50000	0.8896	0.8911	285 sec

We see that the f1 score isn't going past 0.9, hence we change the regression.

3.2 Bagged Softmax

I use Bagged Softmax with subsample_ratio and subsample_features both being 0.8, in hopes that it would outperform the regular Softmax. My observations have been tabulated below:

learning_rate	n_iterations	n_estimators	f1_score	accuracy	time
0.01	10000	10	0.8915	0.8931	450 sec
0.01	10000	4	0.8923	272	0.8939

I realised it wouldn't outperform KNN at all, and didn't proceed further.

3.3 Perceptron

I used a one-for-all multi-class perceptron to help seperate the data if it was linearly seperable.

learning_rate	n_iterations	f1_score	accuracy	time
0.001	1000	0.8499	0.8519	75 sec
0.001	5000	0.8520	0.8539	200 sec
0.001	10000	0.8520	0.8539	641 sec

The perceptron classifier stagnates after 5000 iterations.

3.4 KNN Classifier

Out of all the classifiers, KNN performed the best.

k	f1_score	accuracy	time
4	0.9459	0.9463	234 sec
5	0.9502	0.9503	490 sec
6	0.9479	0.9579	466 sec

Though $k = 5$ went way past the stated time limit, I will assume my system underperformed considering the time taken by $k = 6$. Hence, I will be proceeding with KNN.

4 Observations and Thoughts

Since the original KNN observations were made on the raw data, I had to redo the entire process to find the optimal value of K, which turned out to be $k = 7$.

k	f1_score	accuracy	time
6	0.9527	0.9528	80 sec
7	0.9539	0.9540	80 sec
8	0.9502	0.9504	79 sec

I tried bagging the model but it turned out to be less accurate than the original model, when comparing with the old optimum value $k = 5$

n_estimators	f1_score	accuracy	time
10	0.9463	0.9468	1142 sec
2	0.9274	0.9280	229 sec
4	0.9384	0.9388	460 sec

Hence, I will be only using the regular KNN with PCA.

5 Conclusion

To conclude:

- Simple models like softmax and perceptron exhibited high bias and required extensive training, while KNN displayed low bias but high variance and computational cost in its original high-dimensional form.
- Applying PCA lead to a small loss in data variance but lead to a much more efficient and fast computation.
- KNN classification trained on MNIST data preprocessed with PCA, with the optimum value of $k = 7$ produces a f1 score of 0.9539 on the validation data, in under 5 minutes.