DA2401:Machine Learning LAB

END SEM PROJECT

Racharla Dhanush(DA24B019)

1 Introduction

The MNIST dataset consists of 70,000 images of handwritten digits (0–9), each represented as a 28×28 grayscale image. Every image is flattened into a vector of 784 pixel values ranging from 0 to 255.In this dataset, the column named "label" represents the target class. There is also a feature called "even", which indicates whether the digit is even or not. I removed this feature from the dataset because it introduces bias.

The goal of this project is to design and implement a complete multi-class digit classification system from scratch using only Python, NumPy, and SciPy.

In this project, I implemented Softmax Regression and a KNN + Bagging Classifier. I experimented with both models to evaluate their performance on MNIST data.At the end,Bagging+KNN classifier running better than Softmax.

2 Implementing Softmax Regression

Initially,I implemented softmax regressor from scratch.Softmax Regression, also known as Multinomial Logistic Regression, is a generalized form of binary logistic regression used for multi-class classification problems.I provided my Softmax algorithm in algorithms.py file.

There are two parameters need to do tuning for getting better accuracies.They are learning rate and epochs.I caluculated accuracies based on different parameters.

Table 1: Top 5 f1 scores of Softmax Regression

f1 score	**lr**	**epochs**	**Runtime(in sec)**
0.9015	0.3	1000	53.43
0.9016	0.3	2000	105.78
0.8966	0.1	1000	54.45
0.9019	0.1	2500	132.38
0.9028	0.5	3000	164.71

These results show that the model is quite sensitive to both the learning rate and the number of training epochs.Increasing the number of epochs generally improves the F1 score, the gains become marginal compared to the increase in runtime. For example, raising the epoch count from 1000 to 2000 (with learning rate 0.3) nearly doubles the runtime while improving the F1 score by only 0.0001. By decreasin lr with same number of epochs shows a large variation in accuracy and not a big change in runtime.After certain no.of epochs the accuracy remains constant.So,lr=0.5 and no.of epochs=3000 is the best one I think.

But after doing KNN+Bagging Classifier I got better accuracies than these, with considerable amount of runtime.

3 KNN+Bagging Classifier

K-Nearest Neighbors (KNN) classifies a new sample by comparing it to all training samples and finding the k closest points based on a distance metric (usually Euclidean distance). It then assigns the class that appears most frequently among those k neighbors.

KNN alone has limitations,it is highly sensitive to noise and outliers.Our dataset has 784 features and 10,000 datapoints,so for each datapoint it calucualtes eucledian distance with 784 features and there are 10000 such points.So it is computationally more expensive.

3.1 Impact of Bagging

By decreasing variance and noise sensitivity, agging enhances KNN. Bagging trains multiple KNN models, each on a different bootstrap sample (randomly selected subset with replacement), whereas standard KNN bases its predictions on the entire training dataset. Every KNN model learns a slightly different neighborhood structure because every subset is slightly different.

All KNN models vote by majority to determine the final output during prediction. The shortcomings of individual KNN classifiers, particularly their sensitivity to outliers or incorrectly labeled points, are mitigated by this ensemble method, which yields a prediction that is far more reliable and accurate.

- I normalized the values by dividing it with 255 so that distance can be caluculated easily.

Main Parameters involved in this model are k value from KNN, no.of estimators and sample size from bagging classifier.

Table 2: Top f1 scores of Softmax Regression

f1 score	**k**	**n_estimators**	**Sample size**
0.9300	10	5	0.8
0.9266	15	5	0.8
0.9366	10	7	0.8
0.9347	5	10	0.5
0.9474	5	10	0.8
0.9410	5	5	0.8

- From this table,we observed that if k value is low the model is less biased and gives more accuracy.But accuracy becomes constant after certain k.So,k=5 is better.
- n-estimators refer to the number of models that are trained independently on different bootstrap samples of the training data.If this n-estimators increases the accuracy value also increases.Low estimators gives high variance and low accuracy.estimators from 5-10 gives best accuracy and reduce variance.So,it is betteer to take n-estimators=5.
- sample size ratio refers to fraction of the original training dataset is used to create each bootstrap sample in the Bagging ensemble.When ratio is <0.7 there is more diversity between estimators and has high bias.So,acc is small.For ratio between 0.7-0.9 has good accuracy and best balance of diversity.The best choice is 0.8.

My final parameters for KNN+Bagging Classifier are k=5,n-estimators=8 and sample size=0.8.

Fo this model I got bias=0.0584,variance=0.0147.

Even though this model gives good accuracy, it takes more time to run. For $n_{\text{estimators}}$ (time complexity $\mathbf{O(n)}$), when $n = 5$ it takes about 3 minutes, and when $n = 10$ it takes approximately 6 minutes. Increasing the number of estimators means training more KNN models, which results in more computation.

For the sample–size ratio, the time complexity is $O(\mathbf{sample\_ratio} \times N)$ where N is the size of the training dataset.

3.2 Including PCA

By including PCA it solved my time problem.PCA (Principal Component Analysis) is used to reduce the 784-dimensional MNIST images into a smaller set of meaningful features while preserving most of the variance.

By projecting the data onto the top principal components, PCA removes noise, eliminates redundant information, and makes distance calculations more effective.

By keeping k=5,n-estimators=10 and sample-size ratio=0.8 I changed PCA components values and found accuracy and time.

PCA Components	Runtime (sec)	Macro F1 Score
100	24.50	0.9535
80	21.12	0.9534
150	31.69	0.9494
110	24.52	0.9476

Table 3: Effect of PCA components on runtime and F1 score.

The highest F1 score (0.9535) occurs at 100 components, while 80 components offers nearly identical accuracy with the fastest runtime (21.12 seconds).

Increasing PCA components beyond 120–150 leads to longer runtime and slightly lower accuracy, as more noisy or less informative features are included. Overall, the results demonstrate that 80–100 PCA components capture most of the important variance in MNIST, resulting in strong accuracy and efficient computation.

My final model parameters are **k=5,n-estimators=10,sample-size ratio=0.8 and principal components=100**.For this I got **0.9535** accuracy.

After adding PCA I got bias error=0.0488 and variance error=0.0118.

Here is the confusion matrix for final model.

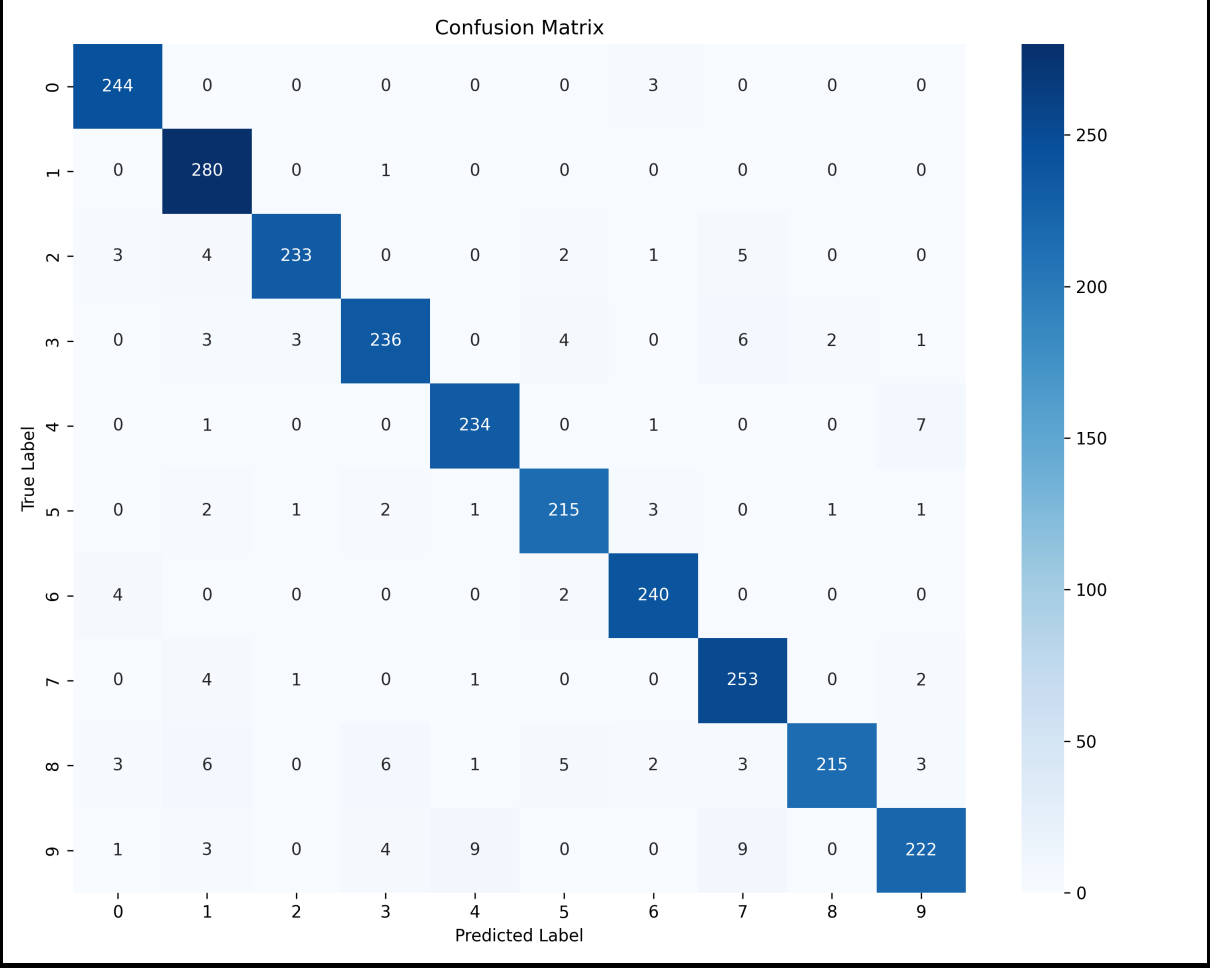

Figure 1: Confusion Matrix of PCA + Bagging + KNN