E0-270 Assignment-1

Smit Radadiya
Department of Computer Science and Automation, IISc, Bangalore.
smitradadiya@iisc.ac.in

March 2023

1 Task-1(PCA)

1.1 Normalization

The task is to normalize data between -1 to 1. So I have implemented a function that uses min-max normalization($0 \leq data \leq 1$) and after multiplying it by 2 and finally subtracting it by 1 to convert it into $-1 \leq data \leq 1$. Here min is the minimum of a dataset, and max is the maximum.

$$data = 2 * \left(\frac{data - min}{max - min}\right) - 1 \tag{1}$$

1.2 PCA

In the fit function of PCA, first of all, I found the Covariance matrix of the given data and then took the first k(5, 10, 20, 50, 100, 200, 500) components of that covariance matrix as a transformation. For the transformation, I just performed matrix multiplication.

2 Task-2(SVM)

2.1 Binary SVM

In this task, I implemented the SVM model for Binary classification with SGD(stochastic gradient descent) with subgradients when the gradient is undefined. Here we have given an equation for optimization.

$$\min_{w,b} \quad \frac{||w||^2}{2} + c\sum_{n=1}^{N} max(0, 1 - y_n(w^T x_n + b)) \tag{2}$$

As we have to use SGD for updating, we want to define when and how to update. So whenever there is a misclassification, we update w and b. Here the eq(2) is not continuous, so we can't directly find the derivative of eq(2). But we update only when it is misclassification$[y_n(w^T x_n + b) < 1]$.

$$w_{new} = w_{old} - \lambda(w_{old} - Cy_n x_n) \tag{3}$$

$$b_{new} = b_{old} + \lambda C y_n \tag{4}$$

Here I took 300K iterations for training SVM. Datapoint is picked uniformly at random from 60K training samples. I choose $\lambda = 0.001 \ \ and \ \ C = 10$ as a hyperparameter.

2.2 Multi-Class SVM

MNIST has 10 classes, so we have to train 10 such SVM to classify these 10 classes. Since SVM is a binary classifier, we must build a 1-vs-rest classifier. For that first process, the data to make suitable for our model. We have 10 SVM in a list, So I train each to correspond to its index of the class. I replace all the y's with -1 except the one for which I train and fit the model.

In the prediction, the function returns a vector of distances from the decision boundary and stores it in a matrix, And takes a maximum from each row for the class label. All functions are defined and plotted accordingly as given to use a **"macro average"** for Precision, Recall and F1 score.

3 Results

Components	Accuracy	Precision	Recall	F1 Score
5	0.5294	0.5485	0.5190	0.5334
10	0.7471	0.7378	0.74	0.7389
20	0.843	0.8470	0.8395	0.8433
50	0.8826	0.8824	0.8804	0.8814
100	0.9001	0.8996	0.8985	0.8991
200	0.9042	0.9040	0.9023	0.9032
500	0.899	0.8942	0.8879	0.8910

Table 1: Results for 120K iterations

This table shows the matrix for the 120K iterations. We get approximately 90%. For the 100 to 500 components, we get roughly the same accuracy, precision, recall and F1 score. I also tried the 240K iterations, and the results are the same for the higher components, But for the lower(e.g. 5, 10, 20) components, there is some difference in the matrix.
Here is the matrix for the 270K iterations.

Components	Accuracy	Precision	Recall	F1 Score
5	0.5612	0.5583	0.5513	0.5547
10	0.722	0.7109	0.7137	0.7123
20	0.8441	0.85	0.84	0.8450
50	0.8872	0.8888	0.8848	0.8868
100	0.899	0.90	0.8966	0.8987
200	0.9082	0.9073	0.9067	0.9070
500	0.9068	0.9057	0.9058	0.9058

Table 2: Results for 270K iterations

Here, the results are interesting; we got the same matrix with 120K and 270K iterations.

3.1 Plots

Some plots for the results of the SVM model(1-vs-Rest) when trained on the MNIST dataset with 270K iterations.

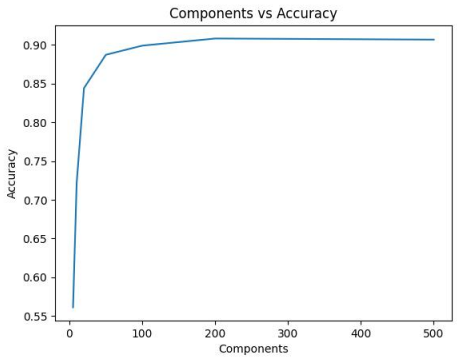

Figure 1: Components vs Accuracy

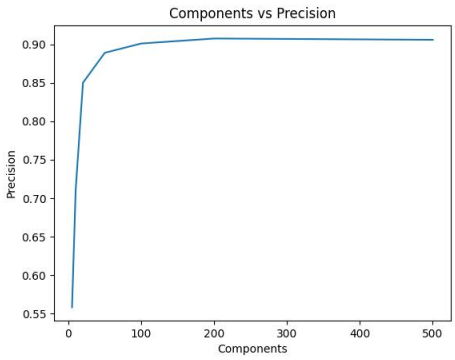

Figure 2: Components vs Precision

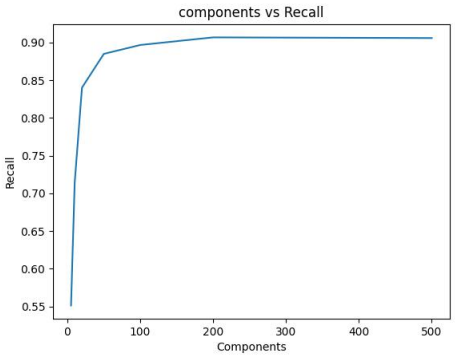

Figure 3: Components vs Recall

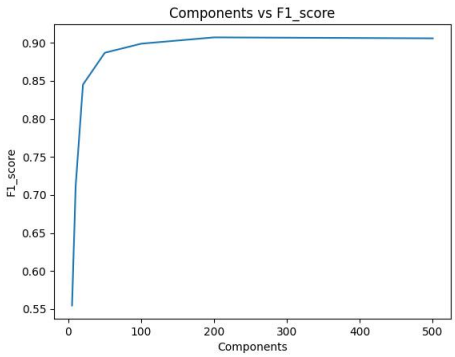

Figure 4: Components vs F1 Score

4 Analysis

We are trying to capture the maximum variance with minimum dimensions(Components) using PCA. Here is the graph showing the variance captured vs components.

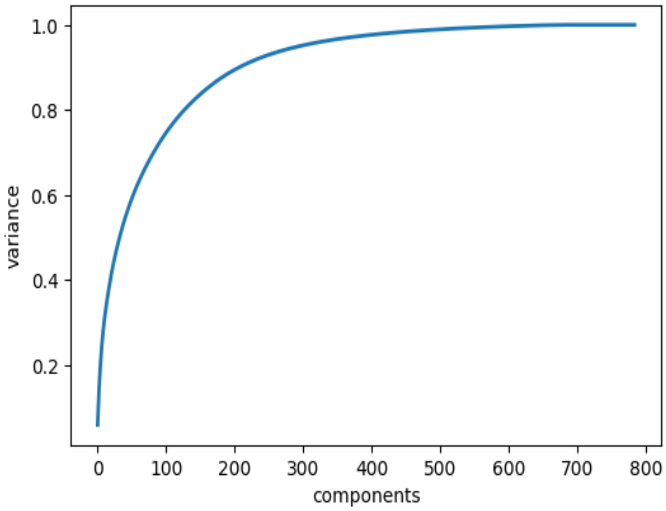

Figure 5: Variance vs Components

We can see that as we reach 400 components, we can capture almost 100% of the variance. We can get decent accuracy at 50 and 100 components by practical results. It is because of normalized data. When we normalized the data with a small range, its variance became small, and for a model like PCA, it is easy to capture high variance with fewer components.
We get 90% accuracy with k = 200 components which is same as without PCA(i.e. 784 dimensions). So we can say that the best choice for k is around 200. So using PCA, we reduced our working dimensions by approximately 75%($784 \rightarrow 200$), overall reducing computational complexity.
For the SVM formulation, we have two hypermeters, learning rate and C. As C increases, the margin reduces for the soft SVM, and we converge very quickly, So I took C=10, which is not very high or low. For the large value of C, our hyperplane oscillates back and forth, which causes misclassification, and the rate of convergence is very low. We might require more iteration to get good accuracy.
The choice of learning rate(λ) is also crucial. It also has similar relationships with the convergence of hyperplanes. The large value of λ leads to a low convergence rate, and a low value of λ leads to smaller steps towards convergence. I took $\lambda = 0.0001$, which gives me the best accuracy.
The choice of a number of iterations is related to computational complexity. I tried for 60K, 120K, 150K, 240K and 270K. All the results were around 87% to 91%.

5 Conclusion

On the MNIST dataset of 10 class classification SVM gives around 90% to 91% accuracy, But we know from the literature that the best accuracy we can get is around 99.96%, which is far from our results. So the conclusion is to look for suitable parameters for the model. It might be possible that the misclassified data points are not linearly separable, So look for some **kernel method that makes it linearly separable**, Or use **Deep Learning(Neural Network)** which is capable enough to classify non-linear data.