ASSIGNMENT – I

Report

Name : Ankit Kumar Sinha

SR. No : 21289

Task 1: Principal Component Analysis

1. Fill in the normalize function in 'utils.py' to normalize the pixels of the images between -1 and 1.
2. Implement PCA to reduce the dimensionality of the images from 784 to a lower number of principal components ($k = 5, 10, 20, 50, 100, 200, 500$). For this, fill in the class methods in 'model.py' for the class 'PCA'.

Methodology:

(1): Normalize function implementation

1. The MNSIT train data sets contain 60000 images of 28X28 = 784 total pixels
2. The values of pixels are from 0 to 255.
3. To normalize the values of images I first divided the images pixel by 255
   x = x/255, the values of pixels is now between 0 to 1
4. Then to keep the values of pixels from -1 to 1 instead, I used
   x = (x-0.5) *2
5. The above steps produce the pixels from -1 to 1

(2): PCA implementation

1. The PCA helps in reducing the dimensions of datasets without losing maximum information
2. It does so by projecting the datasets in another space where the maximum of variance in conserved.
3. For the implementation of the PCA , covariance matrix of the datasets is required. It is calculated from (X-mean(X)) = A and covariance matrix is (A.T)A .
4. The shape of the covariance matrix is 784 X 784
5. Now I have calculated the top k eigenvectors according to their maximum eigenvalues. For our case, it is around K = [5, 10, 20, 50, 100, 200]
6. Now, the X can be projected to lower dimension based on the number of eigenvectors considered. i.e Projected value for K= 5 is X . [eigenvector 1 ….. eigenvector5] = X'
   The shape of the projected value X' is 60000 X 5

Task 2: Multi-Class SVM

Since there are 10 classes in MNIST, you have to perform 1-vs-rest classification for each class separately (thereby training 10 SVM models, one for each class). For multi-class prediction, choose the class corresponding to the model that has the highest value of $w^T x + b$.

1. Implement soft SVM from scratch (linear kernel) using stochiastic gradient descent (use the subgradients when the gradient is undefined). Essentially, fill in the class methods in 'model.py' for the class 'SupportVectorModel'.
2. Now, use the above trained models to create a multi-class classifier. Fill in the class methods in 'model.py' for the class 'MultiClassSVM'.
3. Evaluate the performance of the model on the test set and report the accuracy, precision, recall, and F1 score.
4. Plot the performance metrics (accuracy, precision, recall, and F1 score) with respect to the number of principal components used. Fill in the codes for plotting in 'utils.py'.

Methodology:

(1): Soft SVM using stochastic gradient descent.

1. SVM is used for classification. This model creates the hyperplane with the objective of maximum margin (difference in the distance between sample classes from the edges). This is a powerful algorithm and it has been delivering reliable outputs as a substitute to neural nets. For non-separable data it is using kernel (to get the dot products of the support vectors in higher space) where it can be linearly separable through the hyper plane.
2. First I assigned the weights w and bias to be zero, and we have defined the initial hyper plane. This hyper plane is tuned with the help of gradient descent. Gradient descent is the optimization algorithm which try to find minima w.r.t objective function.
3. The objective function is defined here is :

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

4. Then I defined margin term :

$$\max\left\{0, 1 - y_n\left(w^T x_n + b\right)\right\}$$

5. Then I took the gradient of the above margin term w.r.t to w and b.
6. Since the margin function is discontinuous , I got two condition after taking gradient , one condition for margin above 1 , where the gradient of margin is zero for both w and b, another condition for margin below 1, where the gradient of margin w.r.t w is $-c\ y_n x_n$ and gradient of margin w.r.t b is $-c\ y_n$
7. The gradient w.r.t to objective function for w is $w - c\ y_n x_n$ if margin is less than 1 or else w
8. The gradient w.r.t to objective function for b is $-c\ y_n$ if margin is less than 1 or else 0
9. The update eqn will be :
   For weights is :
   $w_{t+1} = w_t - \eta(w_t - c\ y_n x_n)$ for margin <1

$w_{t+1} = w_t - \eta(w_t)$ otherwise

For bias is :
$b_{t+1} = b_t - \eta(-c\, y_n)$ for margin <1
$b_{t+1} = b_t - \eta(0)$ otherwise

10. We are using stochastic gradient descent where only one instance is used for updating of parameters per iteration.
11. In this way the weights tuned such that the loss is minimized and the hyper plane margin is maximized.
12. For prediction we can use $sign\ (w^t x + b)$
13. Note that the above implementation is for only binary classifications, and the encoding of the two classes is 1 and -1.

(2): Multi class SVM (1 vs rest approach)

1. Since the single SVM is binary classifier , for multiple class we are using 10 SVM trained each one for one specific class. Then the final output of 10 SVM is compared to determine the most probable class the datapoints belongs to.
2. Here I have list of 10 instances of SVM classes.
3. Then every model is trained for one particular class. The data points are divided into target and non target variables by encoding the target class as 1 and non target class as -1.
4. I suspect there would be bias in training since the target class is less and non target class include all other classes, so data points would be high for -1.
5. After training the model is tested on the test data sets. The accuracy of the model is predicted by determining the labels of test data points. The model which have high values of predicted scores $(w^t x + b)$ is the class the data points belongs to.
6. For comparison I appended the scores predicted by the models according to the index. I.e the index 0 will contains the scores of model 0 in matrix, index 1 will contains the scores of model 1 in the matrix and so on.
7. Finally based on the scores the index is returned as class label.

(3): Performance evaluation metrics

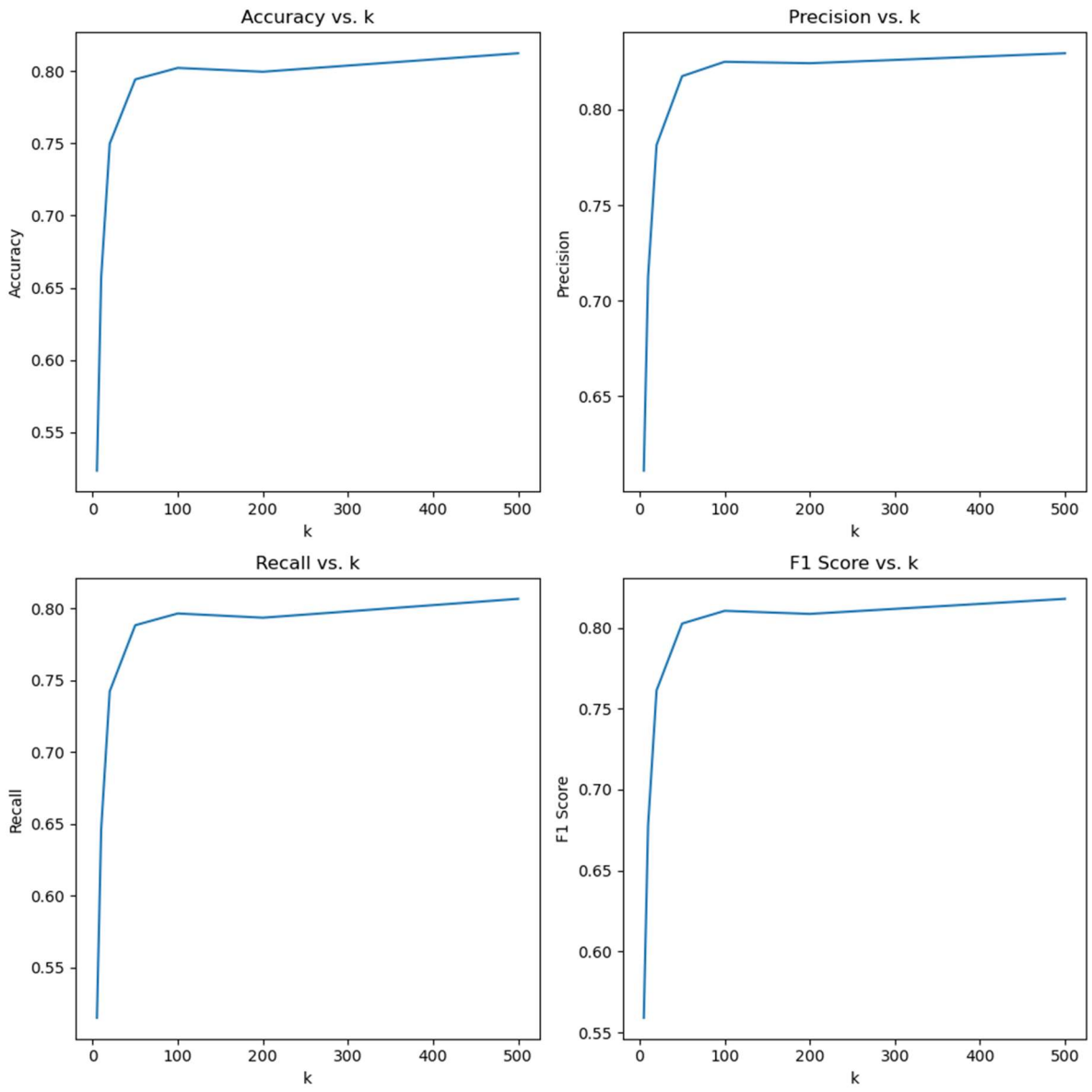
1. Accuracy : The number of correct labels predicted by the model in the test data
2. Precision : Precision is the fraction of true positive predictions among all positive predictions. It measures how accurate the model's positive predictions are. It is calculated as:
   Precision = true positives / (true positives + false positives)

3. Recall : It is the fraction of true positive predictions among all actual positive instances in the dataset. It measures how well the model can identify all positive instances. It is calculated as:
   Recall = true positives / (true positives + false negatives)
4. F1 Score : It is the weighted harmonic mean of precision and recall. It provides a single score that balances both precision and recall, making it a good overall measure of a model's performance. F1 score is calculated as: F1 = 2 * (precision * recall) / (precision + recall)

Hyper-parameters : Learning rate : 0.0001 , C : 1 , Number of iterations: 100000

K (number of dimensions)	Accuracy	Precision	Recall	F1
5	52.3%	0.61	0.51	0.55
10	65.7%	0.71	0.64	0.67
20	74.9%	0.78	0.74	0.76
50	79.4%	0.81	0.78	0.80
100	80.2%	0.82	0.79	0.81
200	79.9%	0.82	0.79	0.80
500	81.2%	0.82	0.80	0.81

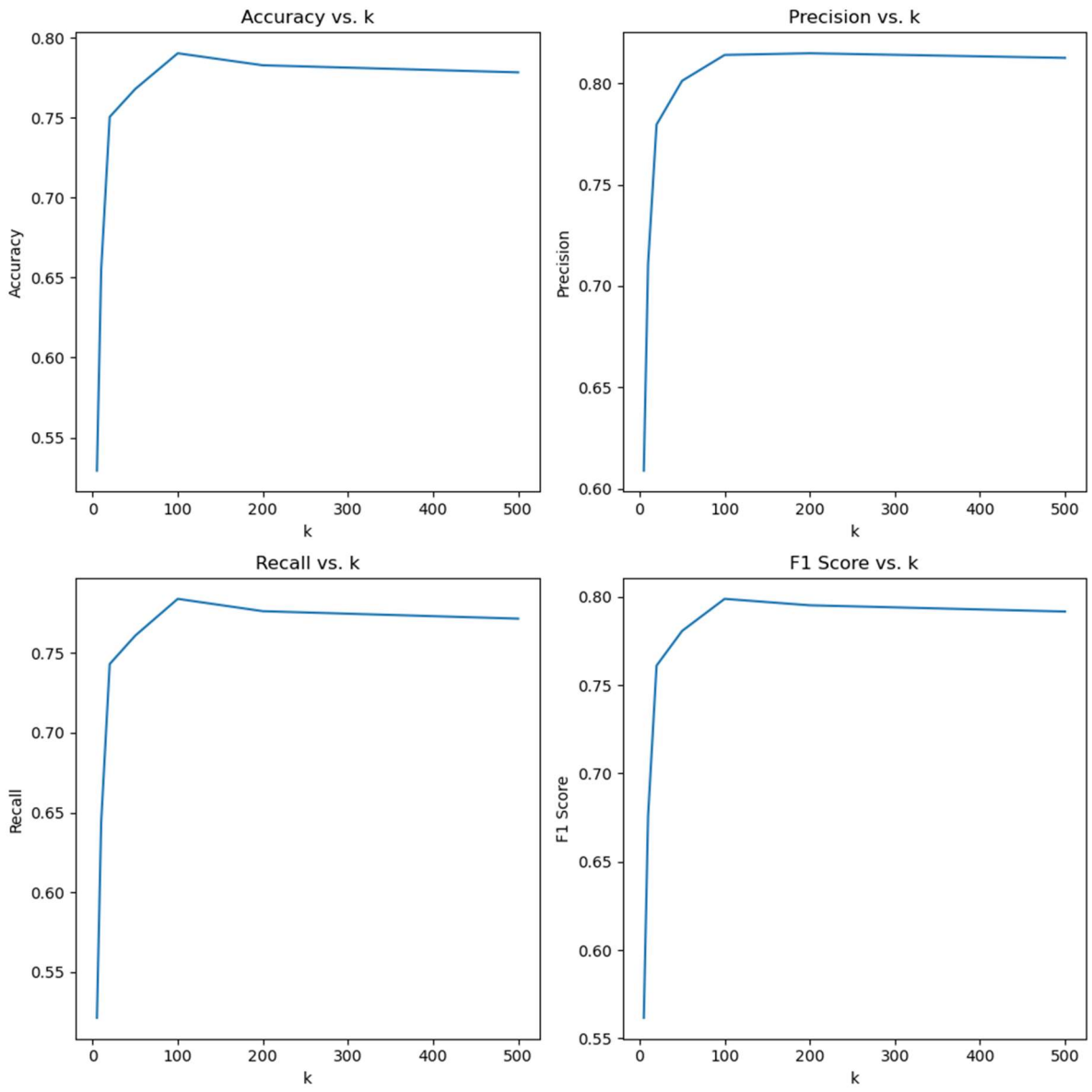
Plots :



Hyper-parameters : Learning rate : 0.00001 , C : 0.5 , Number of iterations: 100000

K (number of dimensions)	Accuracy	Precision	Recall	F1
5	52.9%	0.60	0.52	0.56
10	65.5%	0.71	0.64	0.67
20	75.0%	0.77	0.74	0.76
50	76.7%	0.80	0.76	0.78
100	79.2%	0.81	0.78	0.79
200	78.2%	0.81	0.77	0.79
500	77.8%	0.81	0.77	0.79

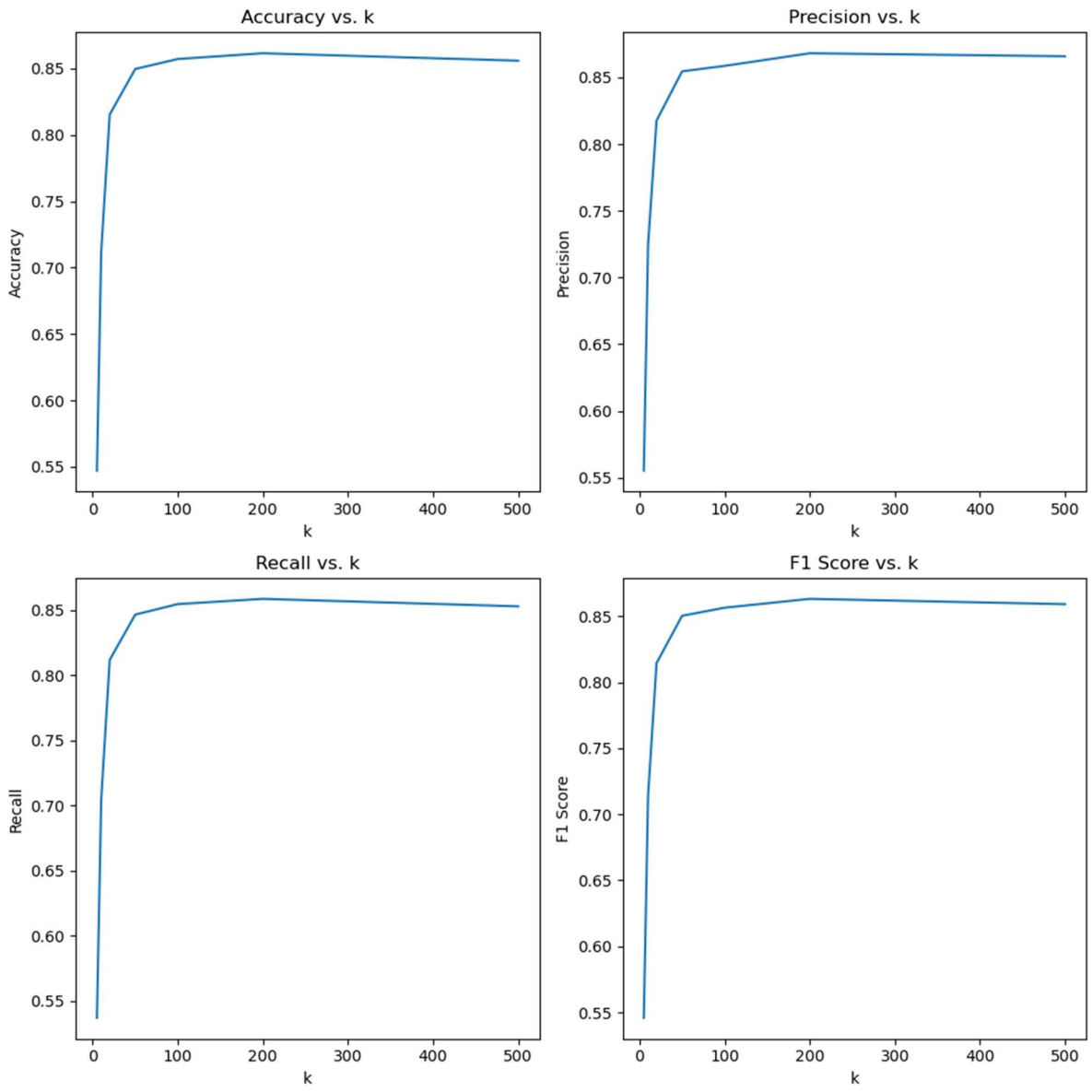
Plots :



Hyper-parameters : Learning rate : 0.0001 , C : 10 , Number of iterations: 100000

K (number of dimensions)	Accuracy	Precision	Recall	F1
5	54.7%	0.55	0.53	0.54
10	71.1%	0.72	0.70	0.71
20	81.5%	0.81	0.81	0.81
50	84.5%	0.85	0.84	0.85
100	85.7%	0.85	0.85	0.85
200	86.1%	0.86	0.85	0.86
500	85.6%	0.86	0.85	0.85

Plots :



- **Observation :**

1. As the number of dimension increases for data points the accuracy of the model increases. This is intuitive because with the increase of dimension increases the more information is captured.
2. The training of data is biased since number of target class data points is less than non target class which include other remaining classes have large data points. To avoid this bias we can use techniques like oversampling of the target class.
3. Through SGD numbers of iterations are required high to train the model and also optimization of the objective function is not smooth, we can use instead batch gradient descent where few numbers of instances are taken into consideration for calculating the gradient. The benefits of this would be model can generalize well and optimization of objective function can be smooth.