E0-270: Assignment 1

Task 1: Principal Component Analysis

Methodology :

- Normalized the given data matrix across each feature such that each entry in the data matrix lie in the range of -1 to 1.
- Some of the features in the original data matrix are in such a way all entries are the same. In such a case, all of those entries are converted into -1.
- Calculated the covariance matrix S of the normalised matrix and took the eigen vectors corresponding to top K eigenvalues of S as principal components.
- Normalized matrix(N*D) was converted appropriately to the size of N*K by projecting each datapoint onto K principal components.

Task 2: Multi-Class SVM

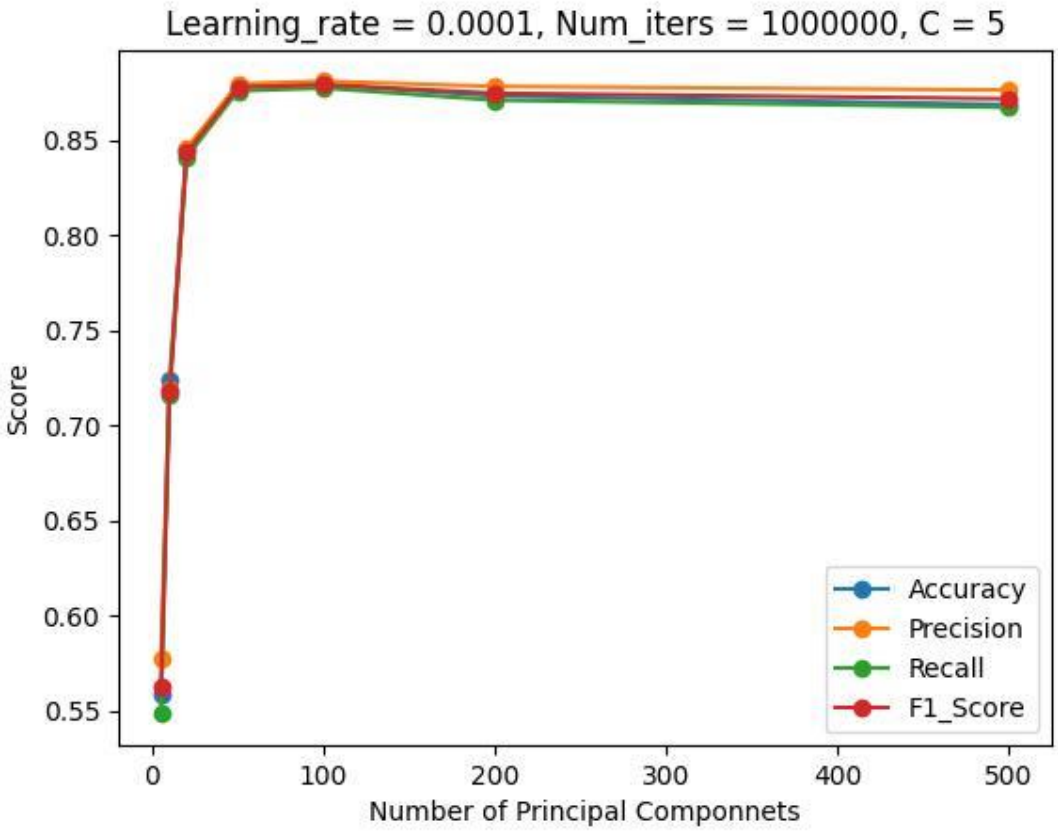
Methodology :

- Using the strategy of the 1-vs-rest approach, For each class J, constructed a dataset D_J such that the labels corresponding to class J are set to 1 and labels corresponding to class other than J are set to -1.
- For each dataset D_J, trained an SVM model M_J with linear kernel for a specific number of iterations using SGD optimiser by randomly taking one example in each iteration and updating the weights w.r.t that specific example.
- The weights corresponding to the trained model M_J are w_J,b_J.
- When the gradient of loss function w.r.t the weights is not defined for hinge loss function, then the gradient is taken as zero.
- The model M_J is responsible for classifying each data point as class J or not of class J.
- Now having trained models M_0, M_1,… M_9, we can now classify a new datapoint x as beloging to class i if $w_i^T x + b_i >= w_j^T x + b_j$ for all j = 0,1..9.
- Precision and Recall can be used as evaluation metrics of the trained model by testing on the test dataset.
- Macro average precision is calculated as the average of precision of each class in the test dataset. Sameway Macro average recall is calculated as the average of recall of each class in the test dataset.

Results :

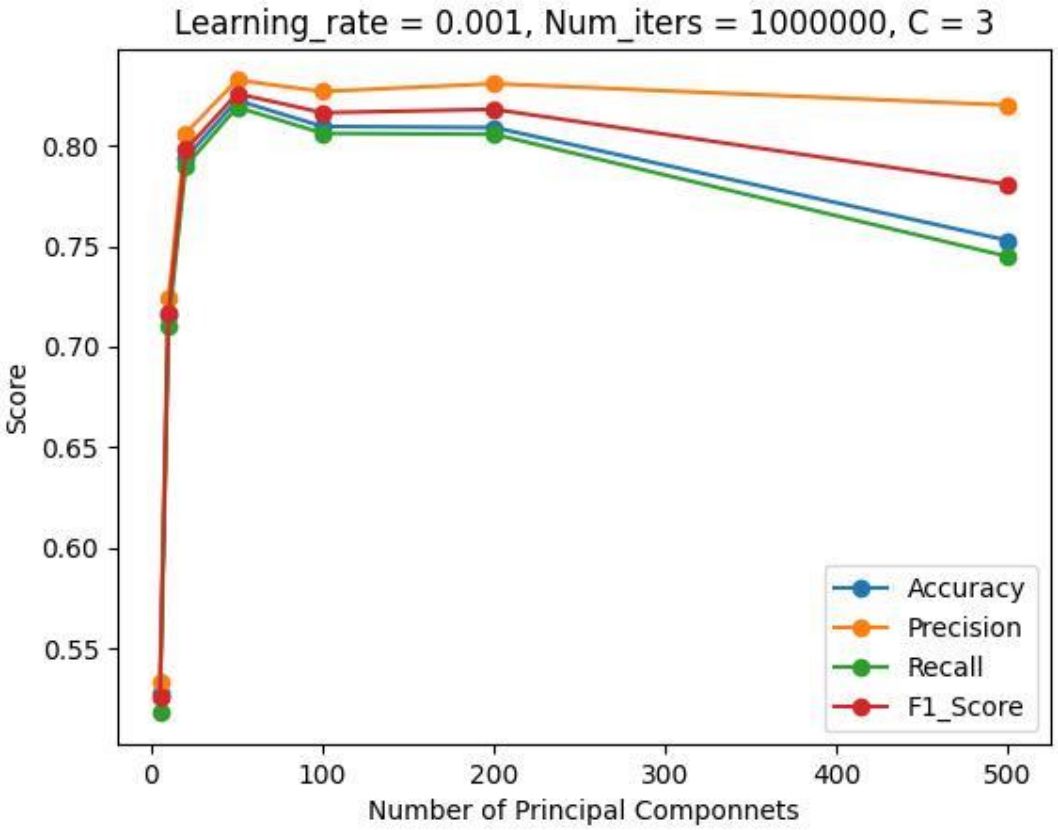
- *Learning_rate=0.0001, Num_iterations=1000000, C=5*
    - *k=5, accuracy=0.5589, precision=0.577, recall=0.548, f1_score=0.5624*
    - *k=10, accuracy=0.7235, precision=0.719, recall=0.716, f1_score=0.7175*
    - *k=20, accuracy=0.8443, precision=0.845, recall=0.840, f1_score=0.8432*
    - *k=100, accuracy=0.879, precision=0.880, recall=0.877, f1_score=0.8791*
    - *k=200, accuracy=0.873, precision=0.878, recall=0.870, f1_score=0.8745*
    - *k=500, accuracy=0.868, precision=0.876, recall=0.867, f1_score=0.8717*

- **Plot of above results K vs Accuracy, Precision, Recall, F1_Score**



- *Learning_rate=0.001, Num_iterations=1000000, C=3*
    - *k=5, accuracy=0.5275, precision=0.533, recall=0.5177, f1_score=0.5253*
    - *k=10, accuracy=0.7158, precision=0.723, recall=0.7099, f1_score=0.7168*
    - *k=20, accuracy=0.7938, precision=0.806, recall=0.7898, f1_score=0.7978*
    - *k=50, accuracy=0.8225, precision=0.832, recall=0.8191, f1_score=0.8257*
    - *k=100, accuracy=0.8095, precision=0.826, recall=0.8061, f1_score=0.8163*
    - *k=200, accuracy=0.809, precision=0.8308, recall=0.8055, f1_score=0.8180*
    - *k=500, accuracy=0.7528, precision=0.820, recall=0.7447, f1_score=0.7806*

- **Plot of above results K vs Accuracy, Precision, Recall, F1_Score**



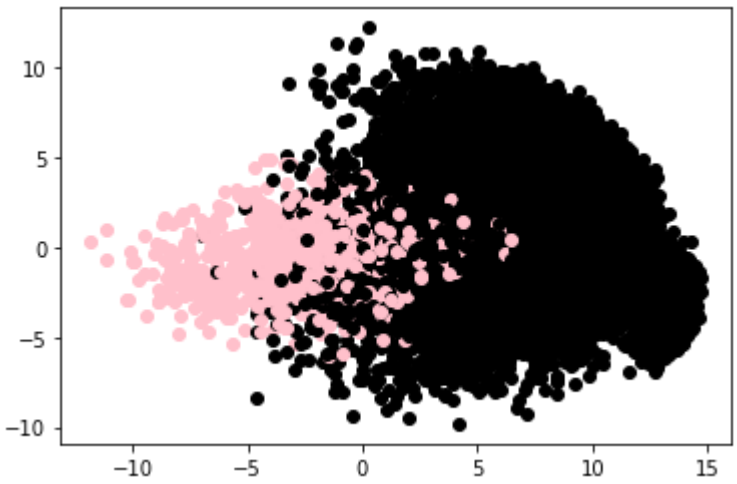
- I was able to achieve an accuracy of 86% on the test dataset with K = 500 components.

Analysis :

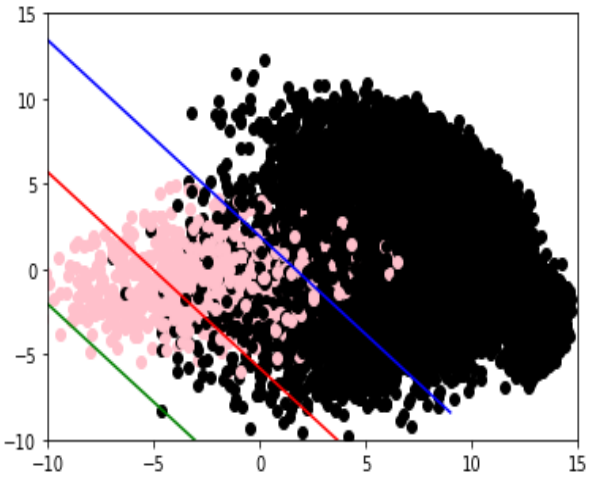
- It was found that better hyperparameters are *learning_rate = 1e-3, num_iters = 1e6, C = 5.*
- Though we are not applying Multi-Class SVM on the original dataset, the higher accuracy we are attaining is attributed to the fact that by using PCA, we are trying to reduce the dimensionality of datapoint such that the maximum amount of information is retained in the transformed space.
- The performance of the model gets improved as we increase the K(number of principal components), this is because the larger the K, the larger the retained variance of the original data.
- We are also able to get an accuracy of ~80% with just K = 50 principal components, and with K>50 we cannot see a significant improvement in the accuracy of the model. Because with K = 50 components, we are able to retain almost 85% of the variance of original data.
- The effect of C can be seen in such a way that a smaller C value will result in a larger margin and more errors on the training set, while a larger C value will result in a smaller margin and fewer errors on the training set.

  Consider the below plot as PCA with K = 2,

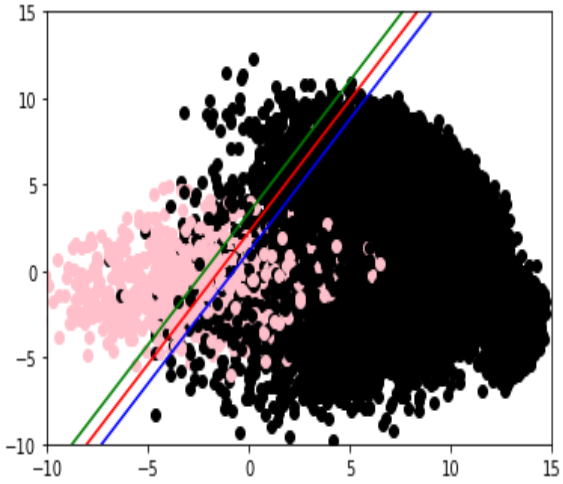
  - **Pink represents class 0**
  - **Black represents remaining classes.**



SVM with C = 0.5



SVM with C = 10



- For the given dataset, C = 5 was a better value that maximises the margin and minimizes the misclassification.
- It was also observed that even with num_iters = 1e3 and K = 500, we are able to attain 80% accuracy. Though the model did not look at all the 60K examples, the performance is attributed to the fact that SGD randomly picks an example in each iteration and updates weights based on that example. i.e. model was able to look at almost all the representatives of the data.
- We can increase the model's accuracy using the polynomial kernels. As we can see from the above data of class 0 and other classes with K = 2, the data was not linearly separable.
- When building model M_J, the corresponding dataset was very imbalanced. Possibly by balancing the dataset the accuracy of model can be improved.