ML Assignment Report

Harsh Vishwakarma
MTech, CSA
21532

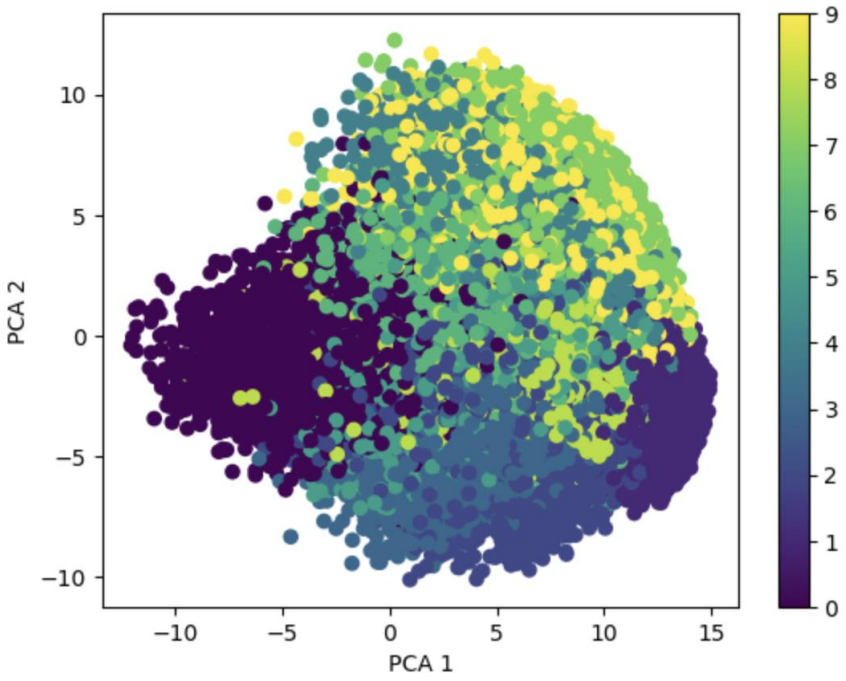
PCA

The Principal Component Analysis reduces the dimensions of the data giving those features in decreasing order which gives maximum information, or we can say those principal components that capture the maximum variance.

Methodology:

- The fit_transform() is calling the fit() which calculates components and then transform() that applies transformation of dimensionality reduction on X
- Fit(): The function computes the principal components of the input data using the eigen decomposition of the covariance matrix, a standard method for PCA (Principal Component Analysis). The **self.n_components** parameter specifies the number of principal components to compute, which is typically much smaller than the number of original features.
- Transform() : it calculates the dot product of X and components, resulting in an X that is reduced in dimensions.
- The number of components that X will contain goes as a parameter to the model.

Results:



The above graph shows the result of PCA model showing first Principal Component on X axis and second Principal Component on Y axis across all 10 classes

This shape of **X_train will become 60000x(num_components)**

MultiClassSVM :

In contrast to binary SVM, which separates the data into two classes using a linear or nonlinear hyperplane, multi-class SVM aims to separate the data into more than two classes using multiple hyperplanes.

Methodology:

The basic idea of multi-class SVM is to train multiple binary SVMs, each of which separates one class from the rest of the data.

In the one-vs-all approach, each SVM is trained to separate one class from all the others, and the final classification is based on the SVM with the highest confidence score.

- The SVM Model updates the weights and bias as per Stochastic Gradient Descent Method by shuffling the X_train in each iteration.
- We are implementing the Soft Margin SVM, so the value of C taken should be small, the best accuracy is coming for C=10.0

- The updating of weights is based on the gradient of the loss function w.r.t w and same with bias taking gradient w.r.t b

```
if (y[i]*(np.dot(X[i], self.w)+self.b)) < 1:
    #misclassified update for ours weights
    self.w = self.w + learning_rate * ( C*(X[i] * y[i])+ (-1*(self.w)) )
    self.b = self.b + learning_rate*(C*y[i])
else:
    #correct classification, update our weights
    self.w = self.w - learning_rate * (self.w)
    self.b = self.b
```

- The above code shoes the implementation of weight update for each data point
- Since we are using SGD, the gradients are getting updated for a single data point in a shuffling manner in each iteration

We are implementing Multiclass SVM using the models of SVM by calling fit function across each class and converting the y_train to another vector containing binary values {1,-1}, 1 for the ground truth for that class and –1 for all other classes.

Results:

```
k=5, accuracy=0.5216, precision=0.5028912970924999, recall=0.5089513239106471, f1_score=0.5059031634477361
 14%|
k=10, accuracy=0.6979, precision=0.6946452694403231, recall=0.6895527168814144, f1_score=0.6920896252482501
 29%|
k=20, accuracy=0.8189, precision=0.8247875611895485, recall=0.8152706185069833, f1_score=0.8200014773528238
 43%|
k=50, accuracy=0.8758, precision=0.8767951481241733, recall=0.873326513957414, f1_score=0.8750573937303359
 57%|
k=100, accuracy=0.8854, precision=0.8862244628564593, recall=0.8830956770130186, f1_score=0.8846573035329348
 71%|
k=200, accuracy=0.8904, precision=0.8916300956091222, recall=0.8882774427367586, f1_score=0.8899506116265397
 86%|
k=500, accuracy=0.8931, precision=0.8930294223957972, recall=0.8911289512088174, f1_score=0.892078174618976
100%|
```

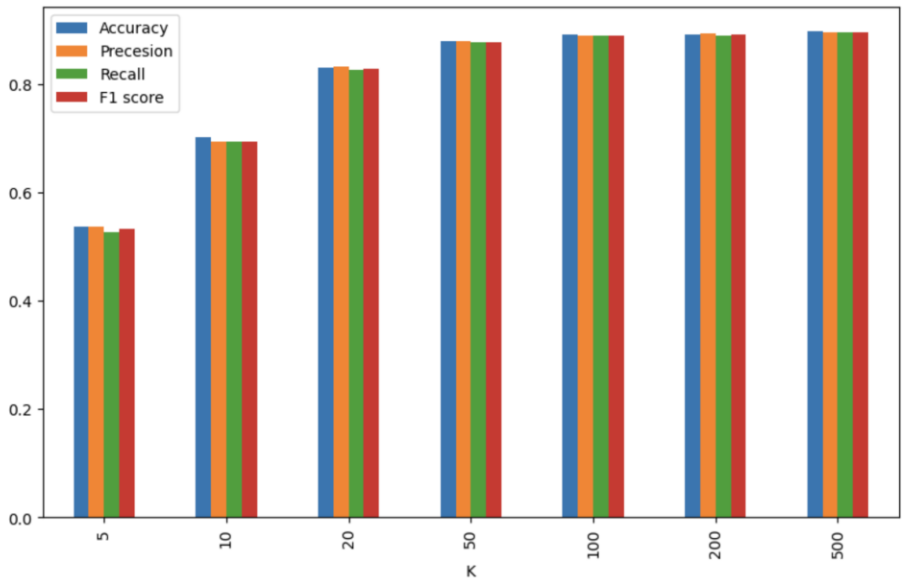
I am getting following results for below hyper parameters:

C = 10

Learning_rate = 1e-05

Num_iterations = 10

Plot for all these experiments:



Analysis of Results:

- There is significant improvement in accuray when we take num_components from 5 -> 10 -> 20 -> 50.
- After **num_components = 50**, there is not much change in the accuray of the prediction
- This is because the **first 50 Principal Components capture maximum variance** in the data, and hence the remaining features does not contain that much information.
- The **rank of the matrix X_train is 712**, so out of 784 only 712 vectors are linearly independent, and out of those only 50 vectors contain the maximum variance