

## **B551 Assignment 4: Machine learning Fall 2018**

### **Classifiers Implemented:**

#### **KNN Classifier:**

- It is nonparametric pattern recognition technique used for data classification. In KNN, the object is classified by majority votes of its neighbors i.e. object is assigned to the class with most common among its k neighbors. Thus, theoretically, KNN can have an arbitrarily shaped decision boundary.

In this classifier, we are storing image feature vectors and its labels. In the classification stage of testing data, an unlabeled image vector is classified by assigning the label which is most frequent among k training image vectors. Here, k is user-defined parameter, we are using K equal to 29. For classification of testing data, we are using cosine distance to find K-nearest image vectors.

The cosine distance between two vectors a and b is defined as  $1 - \text{cosine-similarity}(a, b)$  and cosine similarity is defined as

$$\text{cosine-similarity}(a, b) = \frac{a \cdot b}{(\|a\| \cdot \|b\|)}$$

where  $\|x\|$  is the L2 norm of the vector x.

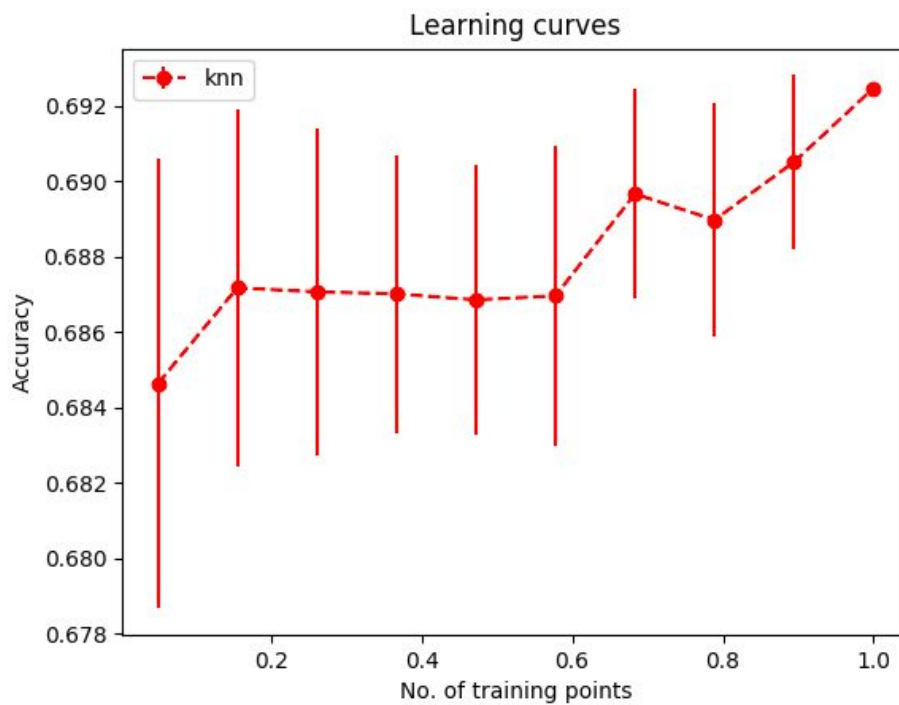
While Euclidean distance measures the magnitude of distance between the two vectors, Cosine distance measures the angle between the two vectors. Thus, cosine distance would be more robust to brightness and contrast changes than Euclidean distance.

To further reduce the amount of noise in the features, we applied PCA and reduced each image to 75 features.

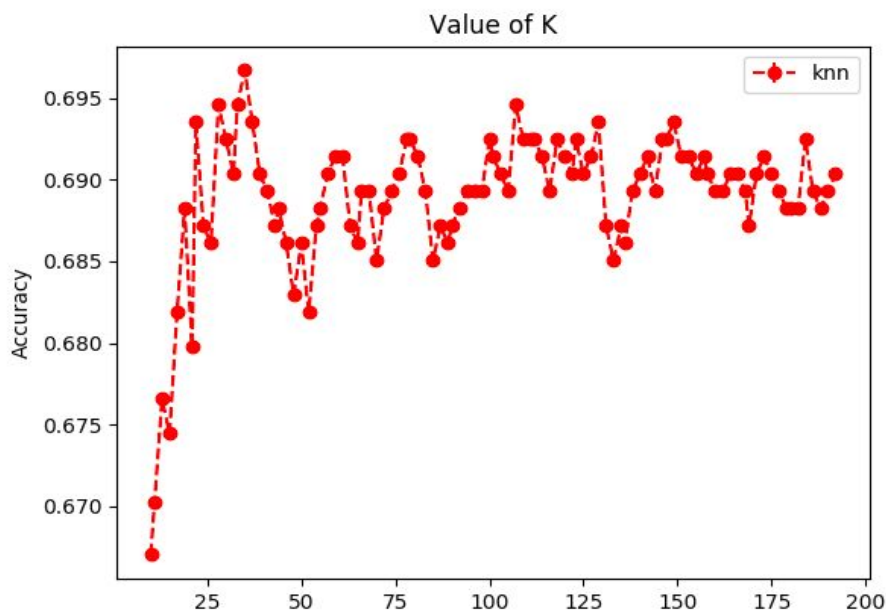
As it can be seen from the image, as the number of data points increases the accuracy of the model increases. The vertical lines in the graph show the standard deviation as we have taken 20 samples for each data size.

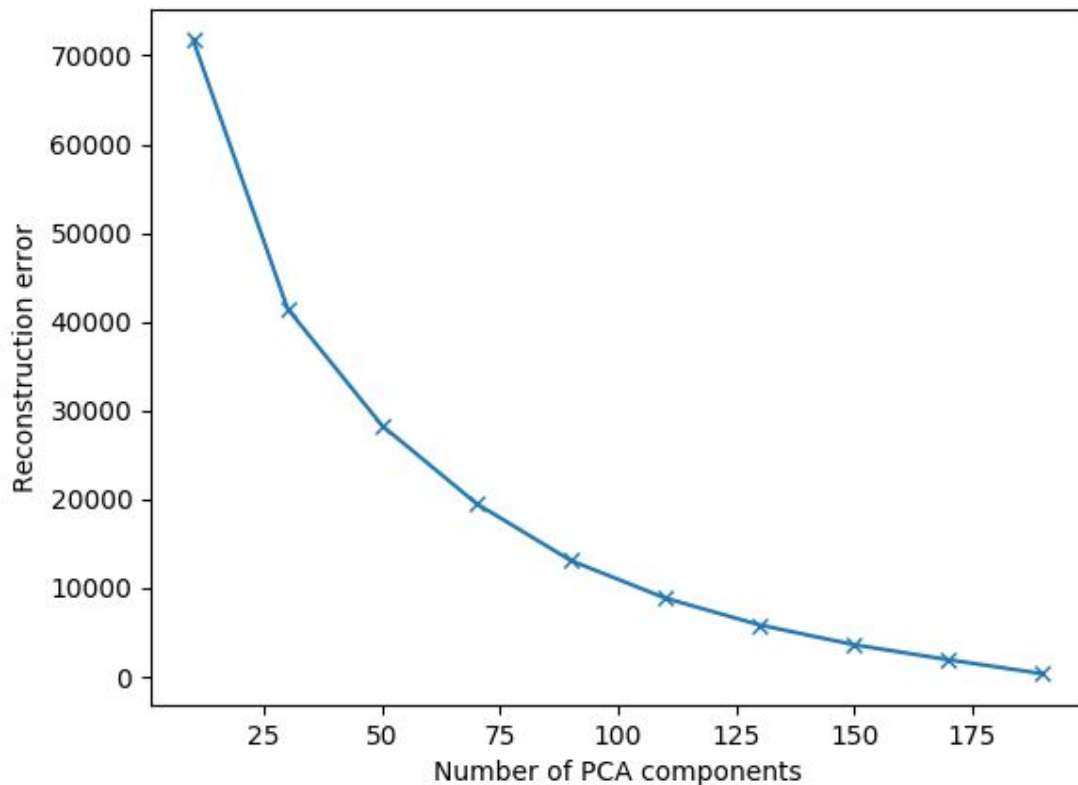
X-axis: the proportion of training data

Y-axis: accuracy on the test data



The below images shows the accuracy with the value of k. Since we are using pca we noticed that performance is best around 25 features.













### **Adaboost classifier:**

Adaptive boosting is a data classification algorithm whose main focus is to classify data by converting a number of weak classifier into a strong one. Decision stumps are used to classify data. In this multiclass adaboost implementation, we are using one-vs-all paradigm to classify the data into 4 different classes(Orientation: 0, 90, 180, 270).

In this algorithm, we are classifying the data on a feature value such that the error should be minimum. For this, we are calculating entropy to decide which feature value classify data in the best possible way. After data classification, we are calculating the error for misclassified data and adjusting the weights such that weights of misclassified data should be more. In the next iteration, adaboost try to classify data with bigger weights. This process repeats a number of times and we create a number of weak classifiers. So to classify test data, we check majority votes of weak classifiers to decide label of unlabeled test data.

Number of Decision stumps	Accuracy
200	28.2078
400	34.4367
800	44.0084

**Image Classification Table**

Correctly Classified Images	Incorrectly Classified Images
('test/9646375952.jpg', 270) 	('test/9831599156.jpg', 270) 
('test/9658475535/.jpg', 90) 	('test/9926949024.jpg', 90) 
('test/9694305901.jpg', 90) 	('test/9944682215.jpg', 90) 
('test/9716486235.jpg', 270) 	('test/9995085083.jpg', 180) 

### **RandomForest classifier:**

Random forest classifier relies on the technique of creating many decision trees to avoid over-fitting. The idea is that each of these trees will capture some information about the dataset and when combined together they will generalize better than a single decision tree.

In our implementation of RandomForest classifier, we have made the following design decisions:

1. For selecting the best split criteria, we have maximized information gain over a randomly selected list of features and thresholds.
2. No further splitting is done if 3 or less than 3 examples are left at the node.
3. A subset of data is used for creating each of the decision trees.

Amount of data	Accuracy
1000	0.6786850477200425
5000	0.6886850477200425
10000	0.689289501590668
20000	0.6808059384941676
Full data	0.69

### Parameters tested for random forest:

#### 1. Tree depth (3,4,5,6,7)


The results got better from depth 3 to depth 6 but stopped improving after depth 7. There was only a slight improvement when we improved the depth from 6 to 7.

2. Stopping criteria for leaf node: We stopped growing the tree when the number of example at any node reaches the limit 3. We also tested values 2 and 1 where both of them performed worse than 3.

#### 3. Number of trees: (10-30-100-300)

We tried using a different number of trees but the accuracy on the training set stopped improving considerably after 30 trees for depth level 6. It only increased by 0.3 for 100 trees of depth 6 which also increased the performance on the training data which be due to overfitting.

**Image Classification Table:**

Correctly Classified Images	Incorrectly Classified Images
<p>(test/97443510.jpg)</p> 	<p>(test/9995085083.jpg)</p> 
<p>(test/9831599156.jpg)</p> 	<p>(test/9760490031.jpg)</p> 
<p>(test/9694305901.jpg', 90)</p> 	<p>(test/9623930399.jpg)</p> 
<p>(test/9658475535.jpg)</p> 	<p>(test/7600713030.jpg)</p> 

## **Best Classifier:**

In the current implementation, out of 3 implemented classifier, a random forest is giving the best accuracy. So for the best classifier, we are using the random forest as a image classifier. A detailed explanation is provided in random forest classifier.

The accuracy can be maximized by allowing the trees to grow to their maximum possible depth (ie equal to number of features) and then applying reduced error pruning.

In the absence of large computational resources to use the pruning strategy, the depth value can be selected on the basis of the following intuition:

1. Large values of depth increase the model capacity and accuracy but also increase the risk of overfitting.
2. Small values of depth reduce the risk of overfitting but also reduce the model capacity and thus accuracy of the model.

The use of an ensemble of a large number of trees mitigates the risk of overfitting while increasing model capacity. The test error reduces with increase in the size of ensemble and reaches a plateau. The ideal values can be determined by various model selection strategies. We used a sample of the training data and computed the test accuracy for various values of tree depth and ensemble size. The precision of these values can be increased by using a larger sample.