* Fall 2018: CSCI-B 551
* Report for Assignment 4
* Ankit Mathur, Nitesh Jaswal, Nishant Jain

**K-nearest Neighbors**

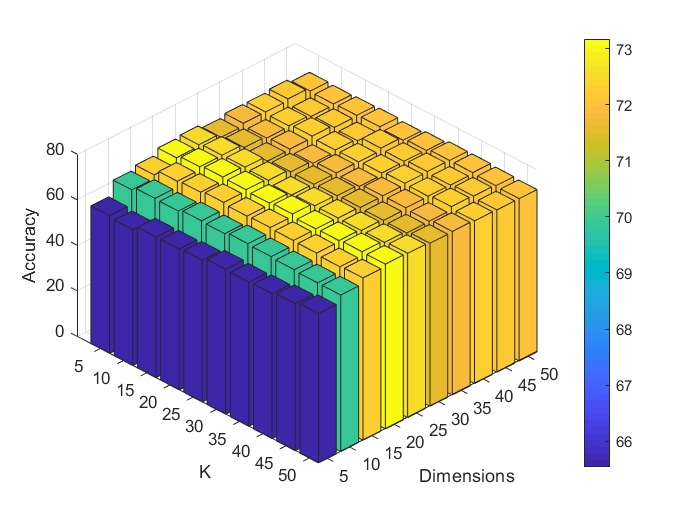
The accuracy of KNN, being an unsupervised algorithm, depends only on the number of neighbors being chosen to predict the label of the test image. However, computing 192 distances for each test image vector consumes a lot of time.

Due to this reason, our model implements Principal Component Analysis (PCA) on the training image vectors to reduce the 192 dimensions into a much smaller number (henceforth called ***dim***).

Below is the table and a 3D chart showing the variation of accuracy for different values of ***k*** and ***dim***:



\*All accuracies recorded on SICE Linux servers



As we can observe, the model performs best for ***k=15*** and ***dim=20***. Moreover, the accuracies for ***dim=20*** is almost always trumps all other values of ***dim*** for any value of ***k*** in terms of accuracy.

As for the time taken by the model to conduct the PCA during training and then predict the test label, it increases as the value of ***dim*** increases, which is what one would expect.

Below is a table that showcases the variation of times taken by the model for different values of ***k*** and ***dim***:



\*All times recorded on SICE Linux servers in seconds \*\*Variation of testing times

As we can see, our model takes a little over 2 minutes to implement the parameters that have the best accuracy (**73.2%** for ***k=15*** and ***dim=20***).

**Hence, we would recommend using these parameters for predicting the orientation of test images given the training data provided to us.**

We also tested various scenarios for different sizes of training data for the best-case parameters. The next page showcases a table to showcase the variation of accuracy and time taken for different number of training samples:



\*All times recorded on SICE Linux servers in seconds \*\*Variation of testing times

As we can see, the model performs better with more training data, which is quite intuitive given the way KNN works. More training samples mean more points in the feature space for KNN to predict the test label.

**Adaboost**

Since Adaboost is a binary classification technique, for this image classification problem our model uses the 1-vs-all classifier methodology during training. Trivially, the accuracy and training time for Adaboost depend only on the number of decision stumps per classifier. Each decision stump compares the pixel values in two randomly generated pairs of pixel values.

Below is a table that compares the accuracies and times taken for the Adaboost model for different number of random pairs (aka decision stumps):



\*All accuracies and times recorded on SICE Linux servers \*\*Time taken both include training and testing

The key thing to note here is that since our model generates random pairs, the accuracies mentioned above will almost always change for the same number of random pairs at every run. However, we can easily observe that the accuracy sort of converges to ~70% after 1,000 random pairs. As for the time taken, it, quite obviously, increases with the number of random pairs.

**Hence, we would recommend using 6,000 random pairs for the Adaboost technique to optimize both accuracy and training time.** The testing time for Adaboost is always less than 10 seconds, so it can be ignored while taking a decision on the parameters to be considered for Adaboost.

We also simulated Adaboost to see how the accuracy and the time taken varies with different size of training data. Below is a table that showcases the results:



\*All accuracies and times recorded on SICE Linux servers \*\*Time taken both include training and testing

As we can observe, the accuracy gradually increases with the size of training data performing best for the entire training data. However, it is to be noted that the increase in accuracy is not big enough (~2%) after 12,000 samples whereas the time taken almost doubles as we move from 12,000 samples to 40,000 samples. While we can argue that it makes sense to use 12,000 samples given the optimization in training time, we recommend that we use the entire training data since the model still trains in a little above 4 minutes.

**Random Forest**

xxx