DATA603/MSML603 Project 1: Facial Recognition

DATA/MSML 603

Heeya Trisal, James Chen and Ya'aqob Aisin

24 January 2020

Introduction

The project at hand was to implement different classifier techniques in order to achieve face recognition. Our class was provided a set of faces and their corresponding labels of “neutral”, “expressive (smiley)”, or “illumination”.

Tasks:

- Split the data into training and testing sets and use the training set to train the classifiers
- Implement preprocessing techniques which include Principal Component Analysis (PCA) or Multiple Discriminant Analysis (MDA)
- Use classifiers such as Bayes and k-Nearest Neighbors(k-NN) to classify the test data
- Experiment through varying the ratio between training and testing data, varying k in k-NN, varying nPCA (to study results on bayesian, varying the PCA dimension, and varying the number of subjects in pose

Results of KNN (varying K) (Face)

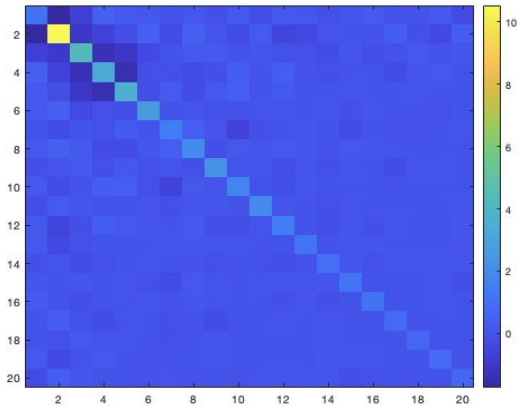
When testing K nearest neighbours in the first split I conducted with splitting the data into test and train using neutral and smiley; under k = 5 I yielded results of Neutral = 126 and smiley=74. When I varied the value of k to 10 I yielded results of Neutral = 118 and smiley = 81. The difference does not seem to be too significant, although there were more smiley classified by the model than before. An issue of this however is as we increase K, the complexity of KNN decreases and there is more “smoothing” of the data. On the other hand, when we choose the lower values in kNN, the model won’t consider the larger distribution of data, low bias and higher variance. With k = 5, the accuracy of the classiifer was 55.5% while with k = 10 the accuracy yielded was 54.5%. The minimal difference shows that varying k does not have a major effect on the accuracy of the model but more so on the variance, bias towards a certain class and complexity of the classified model.

Results of Train/Test Split (varying split ratio)

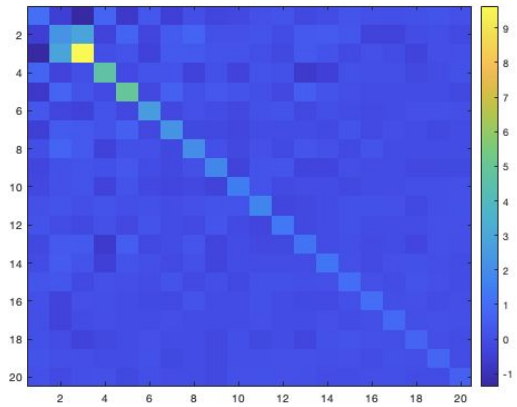
The first split conducted with the face data was splitting the data into testing and training for neutral and smiley. 100 faces were used for the training set and 100 smiley faces in the testing set. The second split conducted was splitting the data into testing and training for neutral and illumination. The following are the results:

- The number of faces in the training set were increased from 100 to 150 having a positive effect on the accuracy of the KNN classifier :Accuracy increased to 59% when k = 5. We can see that classification accuracy improves as we introduce more features to the training set for the model to learn off of.
- Increased the distance between means to about double what it was previously and reduced the distance between covariance matrices when training data was decreased.

```
norm(m1-m2) = 3.214500e+00
```



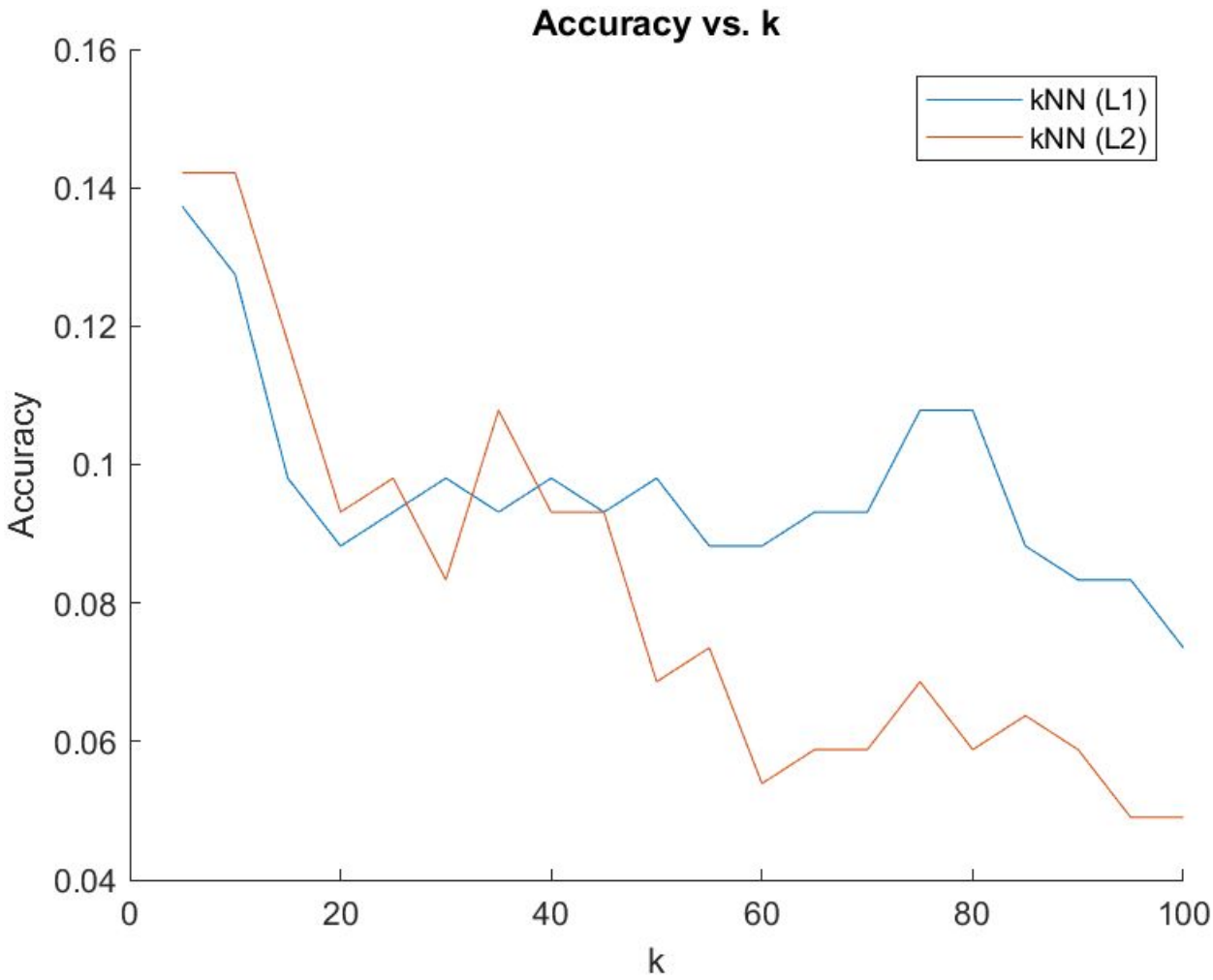
```
norm(mu1-mu2) = 7.495824e+00
```



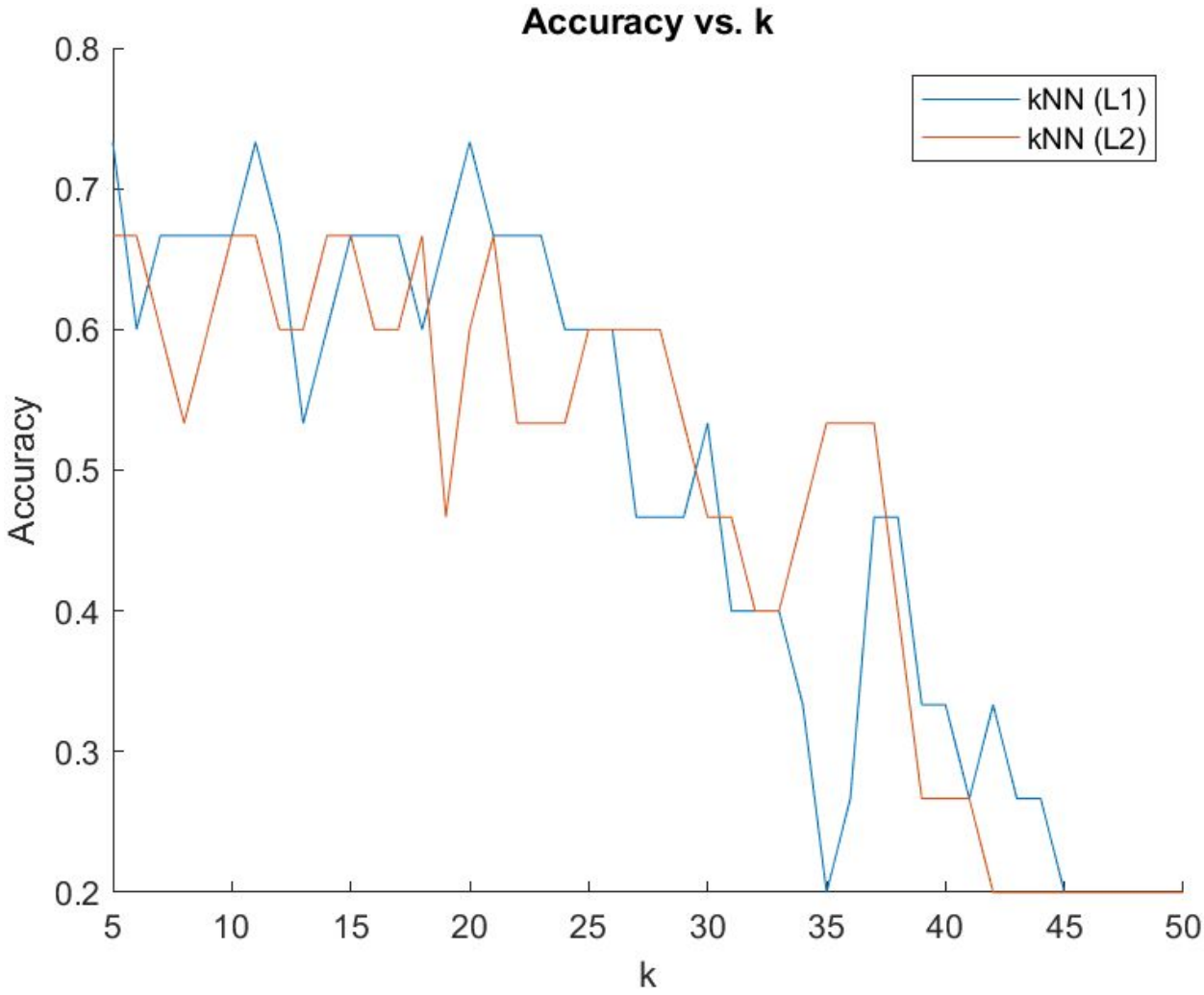
As an alternative method, a training set containing 150 neutral faces and 150 smiley faces and a testing set containing 50 neutral faces and 50 faces. Principal Component Analysis was conducted before the split. The entire procedure was done using Python instead of using MATLAB. This training-testing split has increased the accuracy to above 80%. Initially, the number of neighbors was set to be 3, and the experiment reach an accuracy of 85%; however, after increasing the number of neighbors to 5, the accuracy was decreased to 84% -- although the accuracy has a very slim decrease, it is still above a 80% threshold, which can be deem as highly accurate.

Varying K (Pose)

Fixing the number of subjects at 68, PCA dimensions at 5, and using the first 10 poses from each subject for training and the last 3 poses for testing, we find that increasing k, the number of nearest neighbors, didn't seem to improve the performance of the kNN algorithm. Probably this is due to the fact that there only were 10 training examples from each class, so any choice of k > 10 would be guaranteed to contain at least one observation of an incorrect class. Furthermore, any choice of k > 20 would be guaranteed to contain a majority of observations from incorrect classes, and for even larger k, it would be increasingly likely that there could be 10 of both the correct and incorrect classes in the nearest neighbor set. In that case, it would be pure chance whether the tie break (choosing the smaller class number) resolved in favor of the correct class.



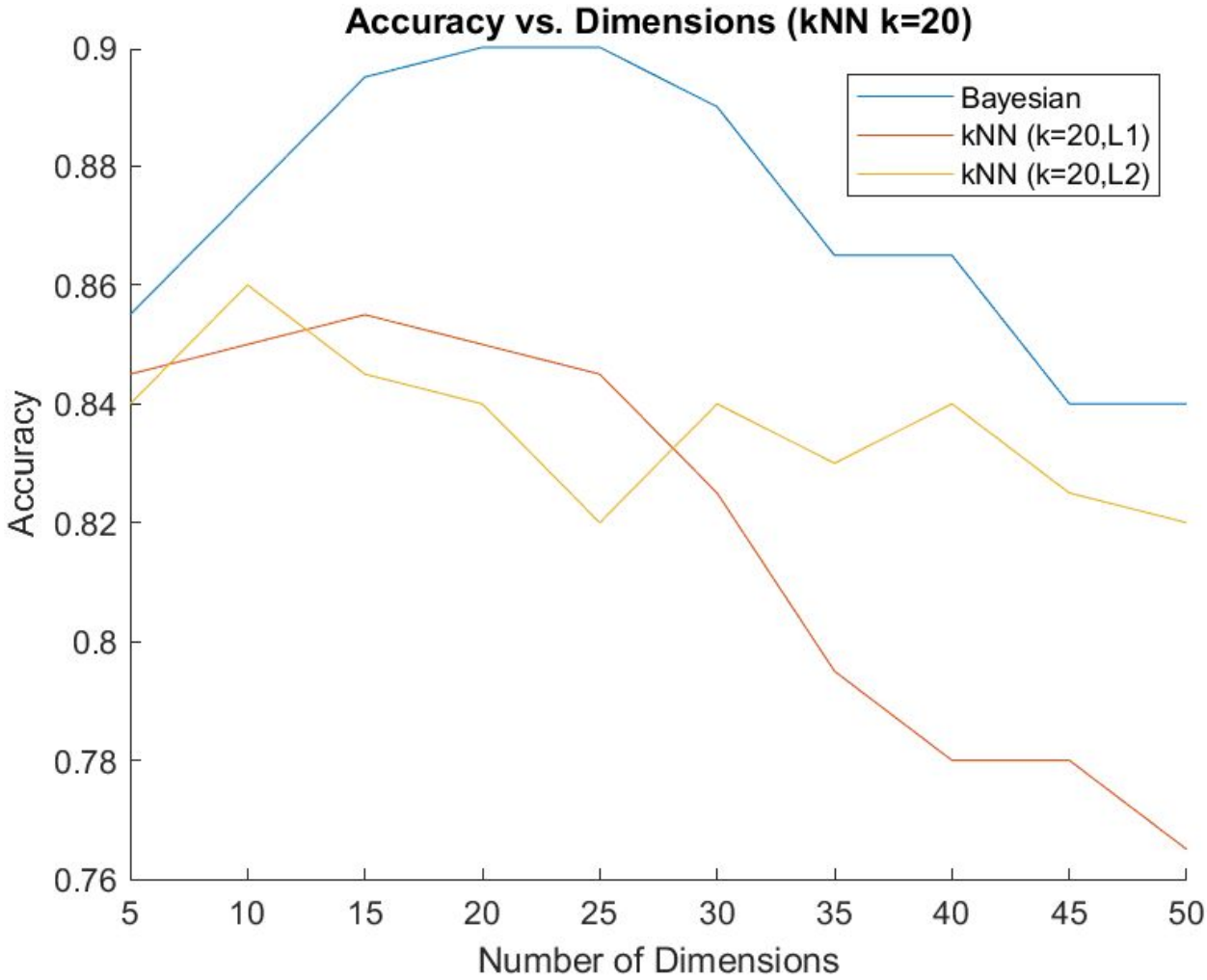
We also tried varying k with 10 subjects, but we got similar results. In both of these cases, we could have randomly resampled the test and training set and averaged out the accuracy to try to get a less noisy sense of the impact of varying k. However, since the test and training set was already so small for each class, it shouldn't have made much of a difference from the general trend we observe here.


Accuracy vs. k
kNN (L1)
kNN (L2)
Accuracy
k
0.8
0.7
0.6
0.5
0.4
0.3
0.2
5
10
15
20
25
30
35
40
45
50

Varying PCA dimension (Face)

With the face data (and randomly splitting 100/100 for training/test from each class), it seems that increasing the number of dimensions remaining after dimension reduction with PCA doesn't improve the performance of the kNN algorithm, while it peaks at around 20 or 25 dimensions for the Gaussian Bayesian classifier.

Note that for large enough d (100 dimensions, for example), the sample covariance matrices for each class become singular, which makes the Bayesian classifier unstable. Fortunately, this doesn't seem to matter here since performance for the Bayesian classifier drops off significantly after 30 dimensions anyway.

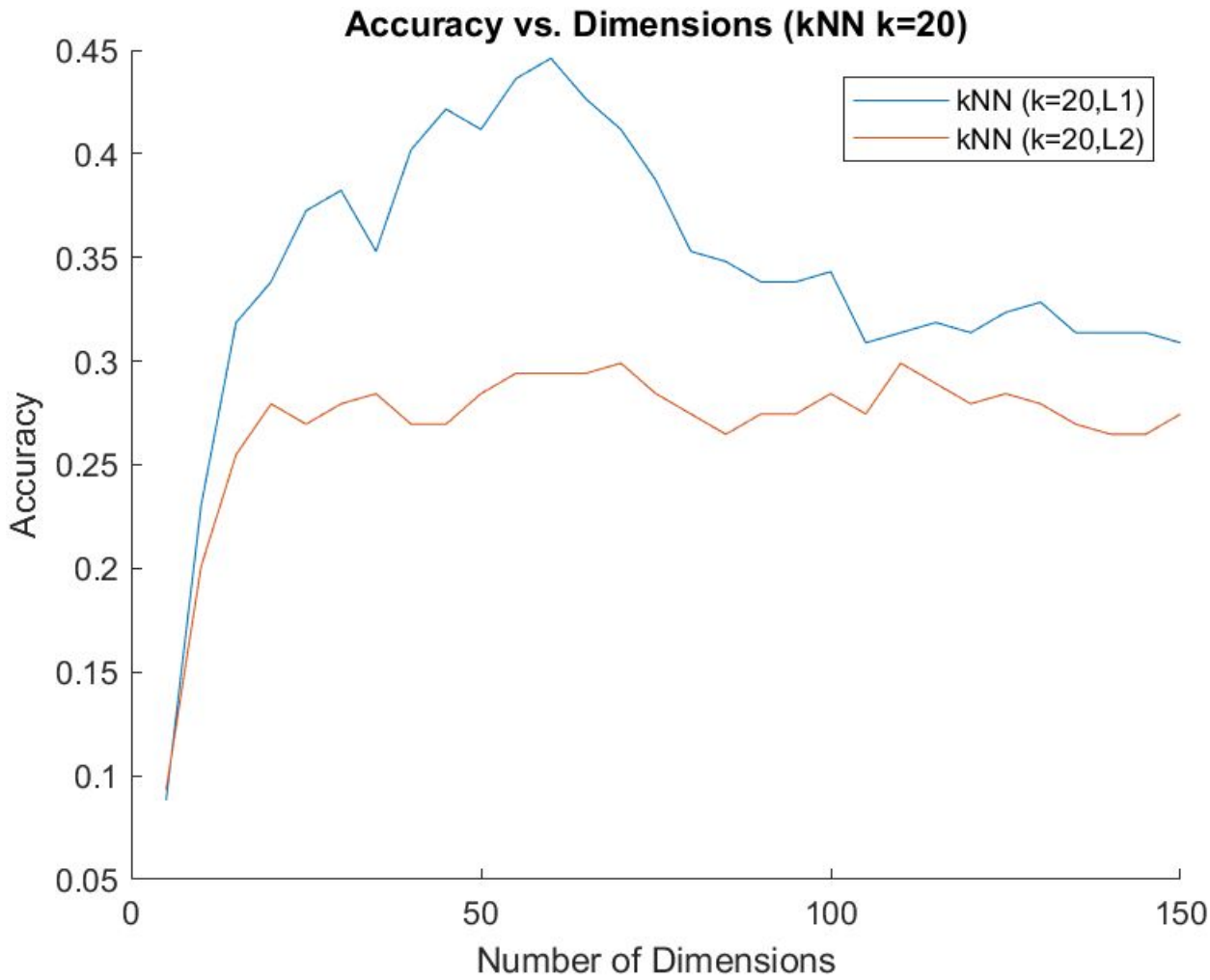


Varying PCA dimension (Pose)

With the pose data, it was possible to attempt to predict all 68 subjects as long as we used PCA to sufficiently reduce the dimensionality of the data. At 6 or more dimensions, the covariance matrices for each class would blow up to become singular, due to the fact that we only had 10 training examples for each class.

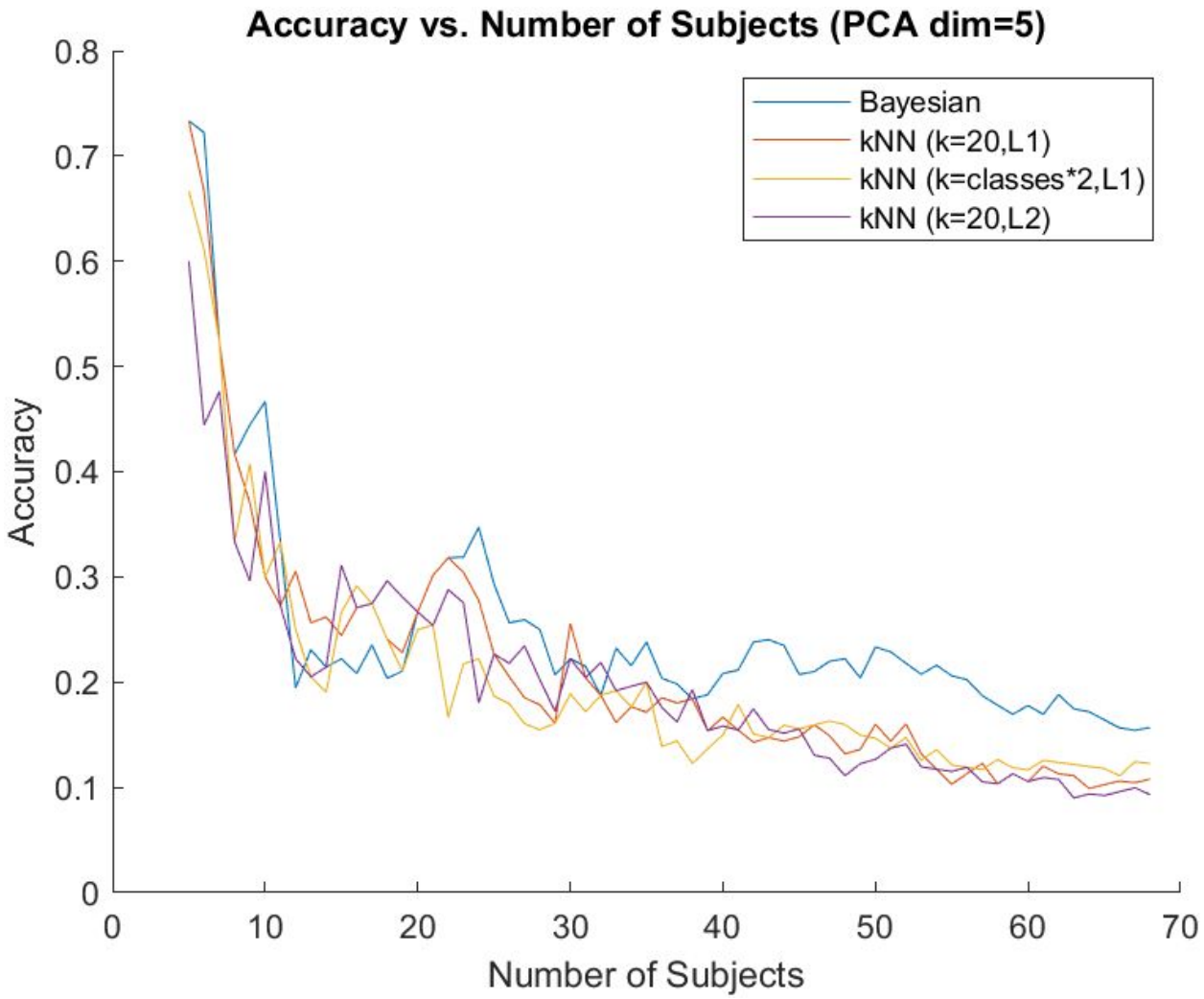
As a result, all of the work in this section on varying the PCA dimensionality on the pose dataset was applied solely to the kNN algorithm.

Using all 68 subjects, and fixing k=20, we plotted the prediction accuracy for kNN with PCA dimensionality reduction at intervals of 5 dimensions. We can see that the accuracy of the kNN algorithm rises steeply up to 20 dimensions before leveling off using the L2 norm, and peaking at 60 dimensions before falling off when using the L1 norm.



While a peak of 45% accuracy using the L1 norm and 60 dimensions might not seem great, we should keep in mind that we are trying to predict 68 different classes with only 10 training examples for each class.

Varying Number of Subjects (Pose)



As mentioned above, it wasn't possible to apply a Gaussian Bayesian classifier if our data had more than 5 dimensions, so we applied PCA to reduce our data to 5 dimensions for the results in this section.

Unsurprisingly, the performance of our algorithms fell off as the number of subjects increased. This is because we still were only including 10 training examples for each class, while dramatically increasing the difficulty of the problem by increasing the number of classes for our classifiers to choose between.

Still, the accuracy of our Gaussian Bayesian classifier seemed to scale comparatively better, ending at just above 15% accuracy (15.69% at 68 classes), which is not great, but also not terrible for picking between 68 different classes with such limited training data. Perhaps if we had been able to implement additional training examples from the illumination dataset, we would've been able to do even better.

The kNN algorithm, using k=20 and the L1 norm, didn't fare quite as well, dropping to 10.78% accuracy at 68 subjects. We tried to tweak it by scaling k with the number of classes and by switching to the L2 norm, but it didn't seem to fundamentally change the performance of the algorithm.

This is probably because kNN is much more adversely impacted by reducing the dimensionality of the data to 5 dimensions than the Gaussian Bayesian classifier. As we observed in the previous section, the ideal dimensionality for kNN in this dataset seems to be around 60 dimensions for the L1 norm.

Code:
James Chen: https://github.com/jchen-data/data603-project1
Yaaqob Aisin https://github.com/BYakovAisin/MSML-603--Project-1
Heeya Trisal https://github.com/htrisal/DATA603Project1