Project 3 Final Report
AI 301
Tali Peters & Armora Rama

Machine Learning Algorithm: Principal Component Analysis

Problem Description:

We decided to explore more the PCA (Principal Component Analysis) algorithm because it is a concept that has been using a lot lately and is one of the most import algorithms in machine learning. PCA is a mathematical analysis that is used to transform large set of variables in a smaller one but that still keeps the important features and the information from the large set. In this way it is easier to graph and visualize.

This algorithm is used in face recognition or word recognition. It is actually one of the algorithms that was used in Google Lenses . In this project we decided to use this with KNN sklearn classifier with the iris dataset.

Key Concepts:

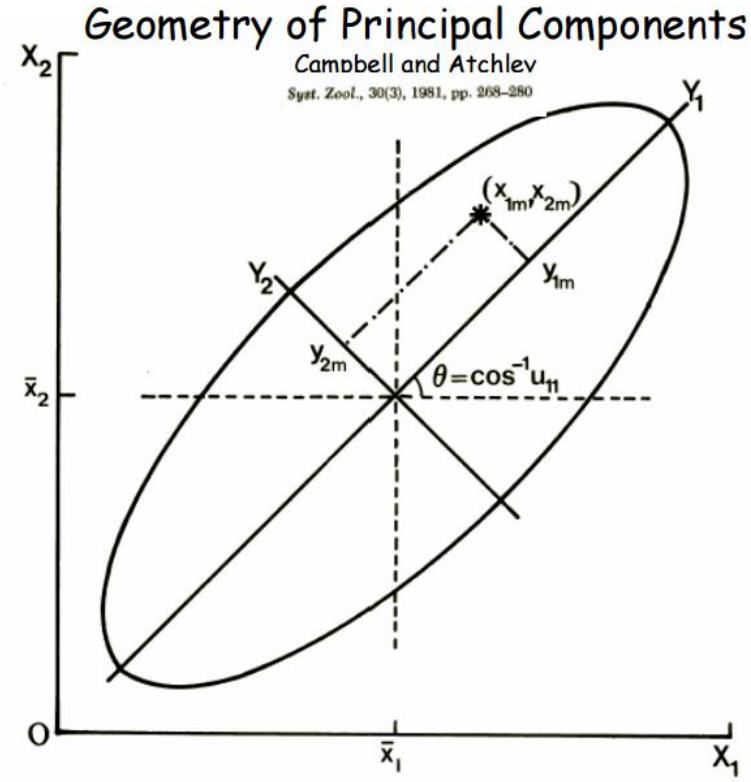
- **Goal** : Data reduction
- **Variables chosen:** The ones that have the highest correlation in the large dataset
- **KNN**: The algorithm with which we run comparisons for PCA
- **Dimension reduction:** Reducing the variables from a larger set to a smaller set that still hold the features
- **Mathematically Speaking:** Square Symmetric Matrix

Related Class Concepts:

- **Classifiers**
    - *K- Nearest Neighbor*
        - KNN uses all the training samples. This means that in large datasets we will have a lot of features to keep in consideration.
        - PCA takes care of that by splitting the data in smaller set of variables
        - The result will have less noise.

- **Other concepts**
  - *LDA (which is another form of pca)*
  - *Feature Selection*
  - *Relevant Features*
  - *Redundant Features*

- **Geometry graph of PCA**



This is a graph taken from a PCA analysis from NCSU(North Carolina State University)

Experimentation:

The objective of these experiments is to test the PCA algorithm with various parameter configurations and dataset. The classification algorithm that we used was the k nearest neighbors sklearn classifier. We look at the effectiveness of PCA with the iris data set and the digits dataset. We will first look graphically at how KNN behaves with the iris data set with and without PCA using both uniform and distance distribution of weights. Later we look at the accuracy of KNN and PCA on the digits data set and the iris data set.

We want to demonstrate when PCA is most effective with the sklearn implementation of knn on different datasets and the best parameter configuration for that to happen.

Results:

Iris data set with KNN graphically:

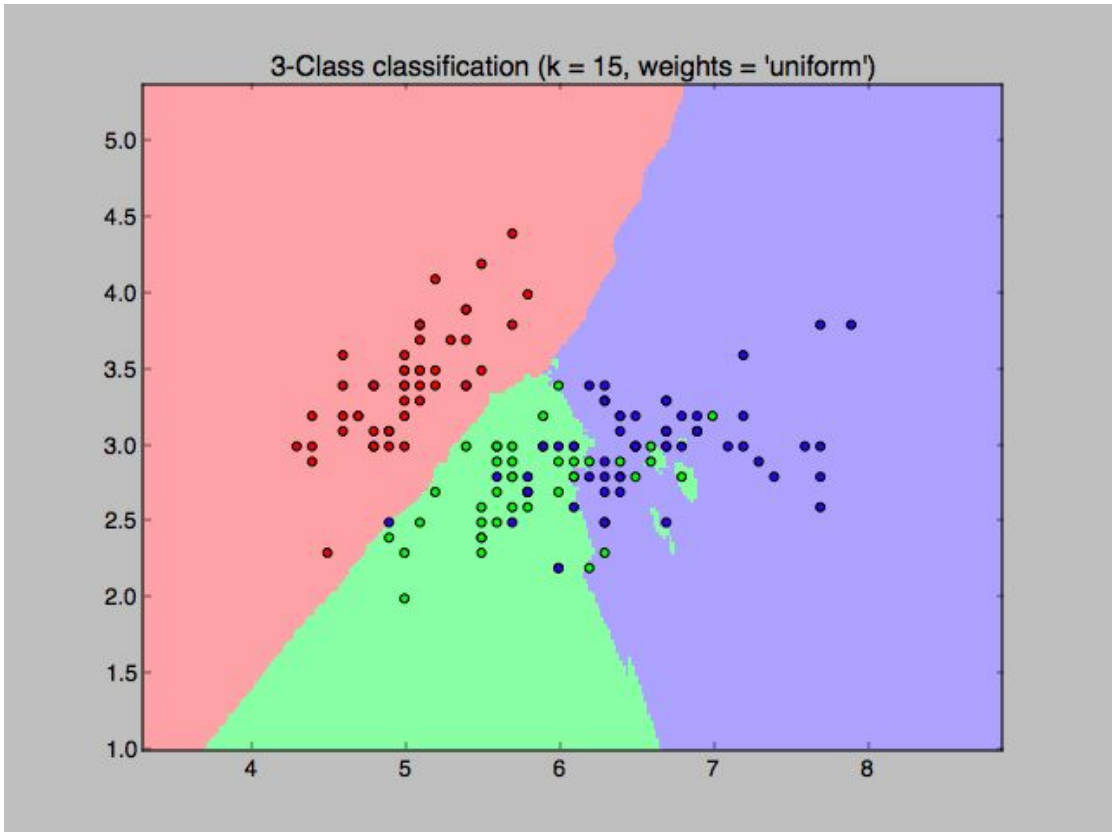


Figure 1.a

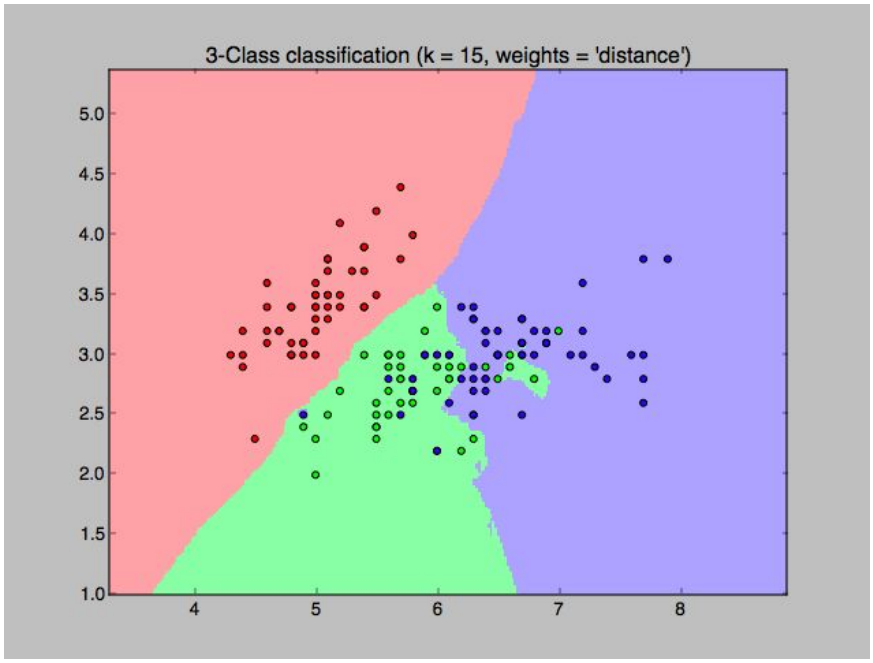


Figure 1.b

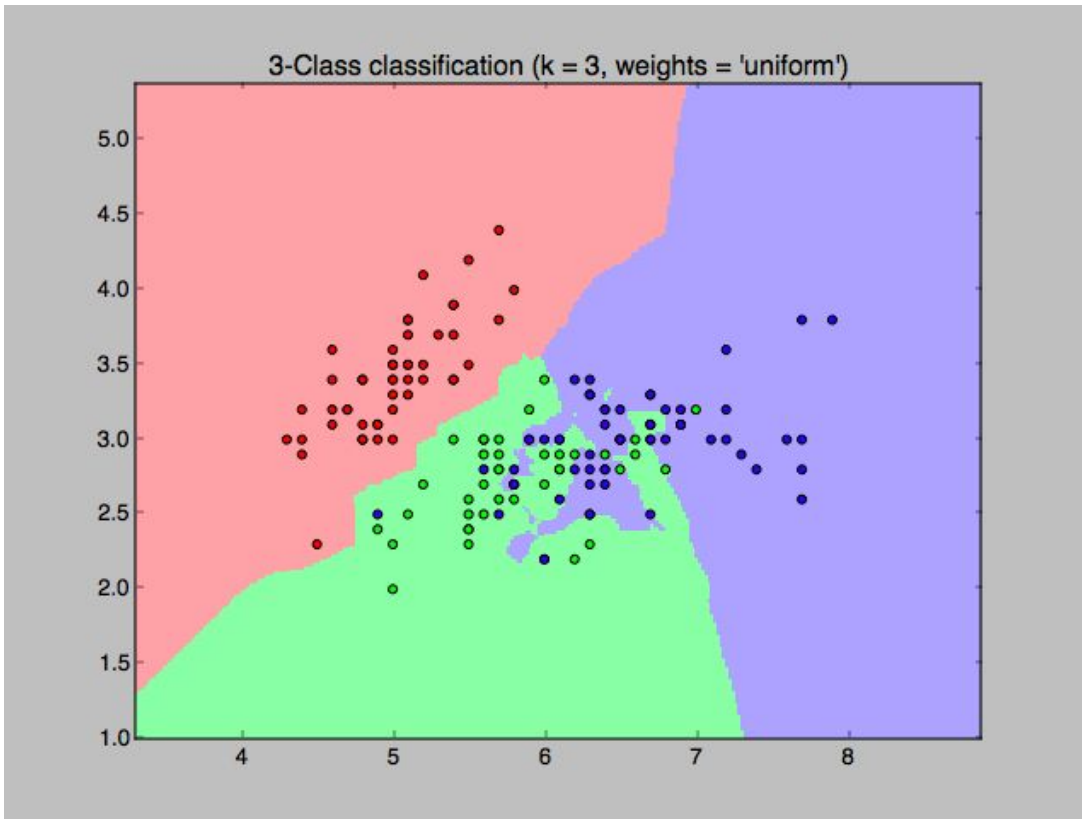


Figure 1.c

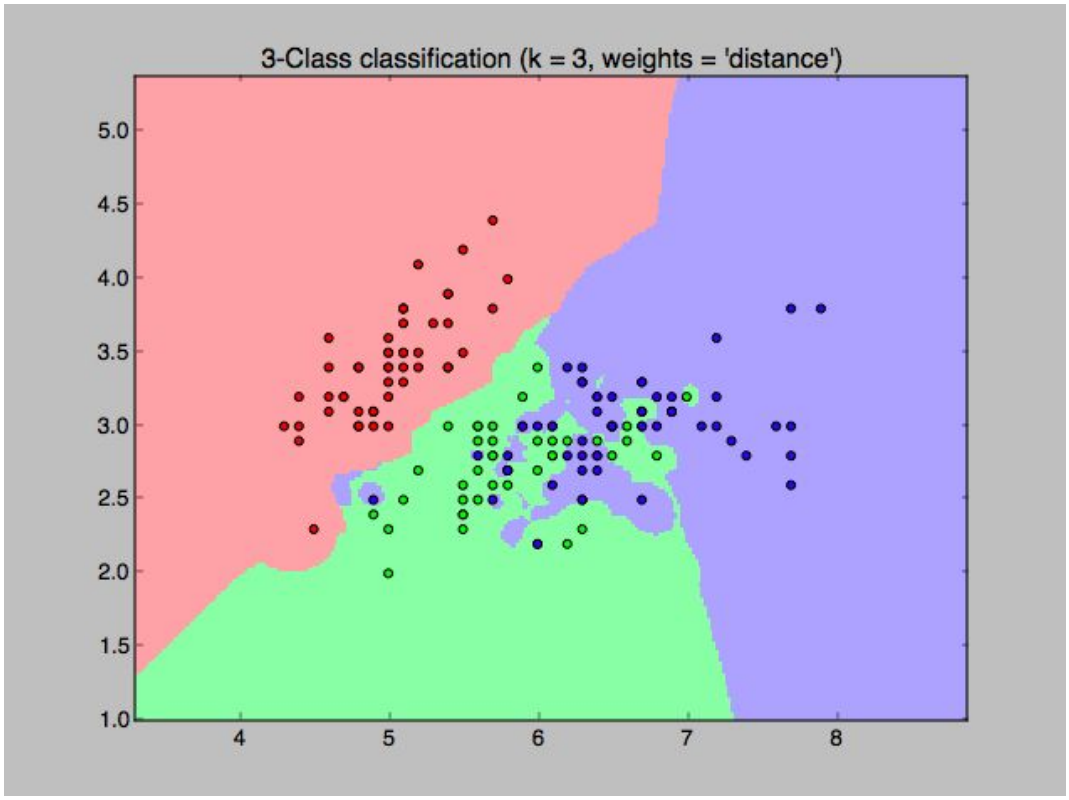


Figure 1.d

Iris data set with KNN and PCA graphically:

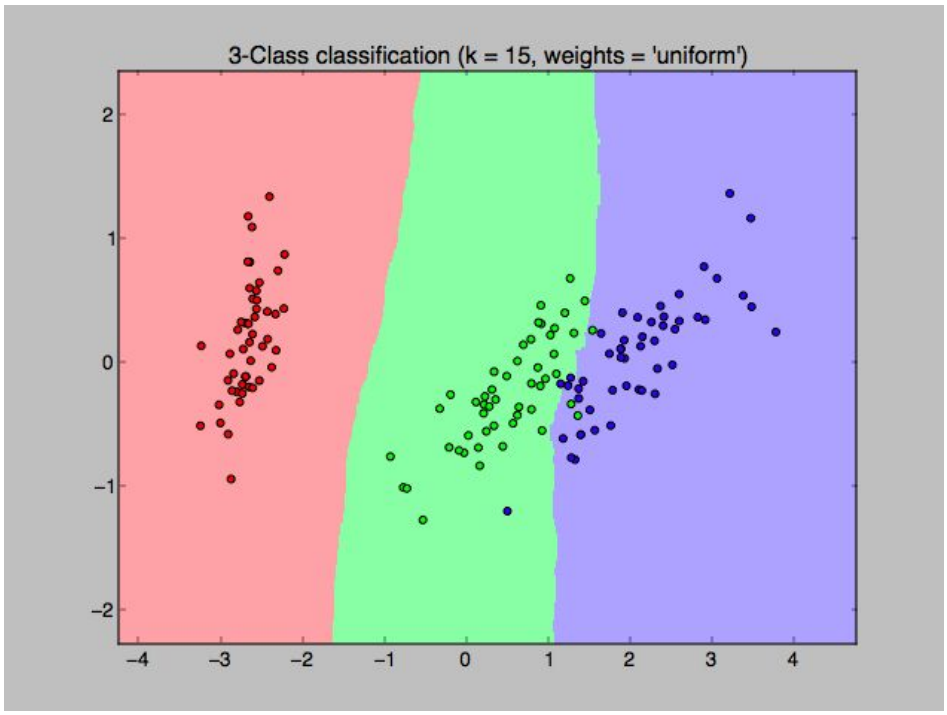


Figure 2.a

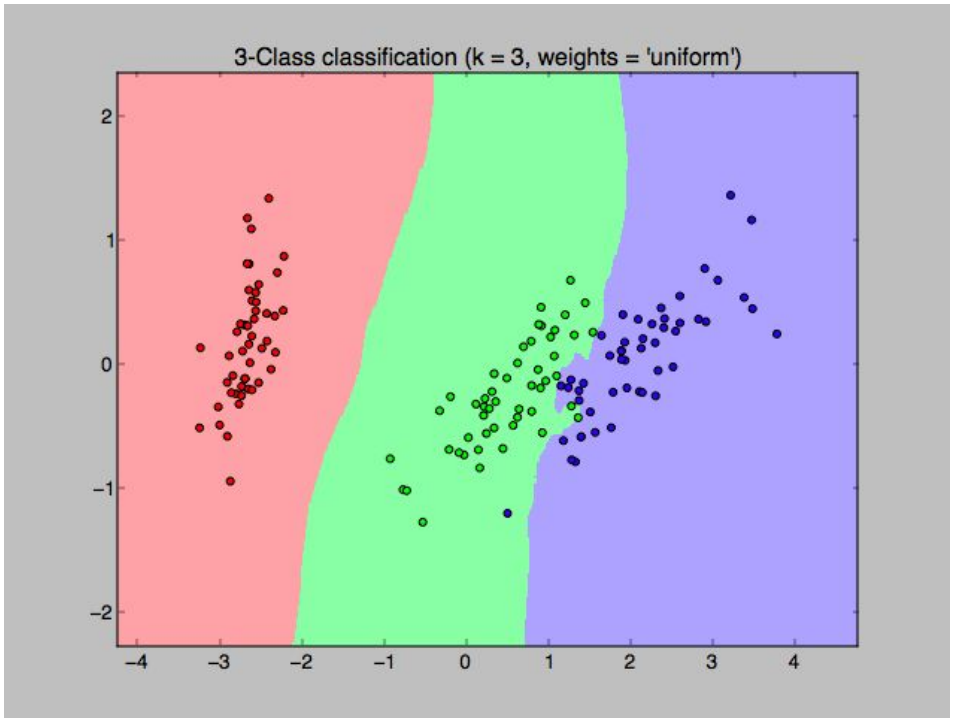


Figure 2.b

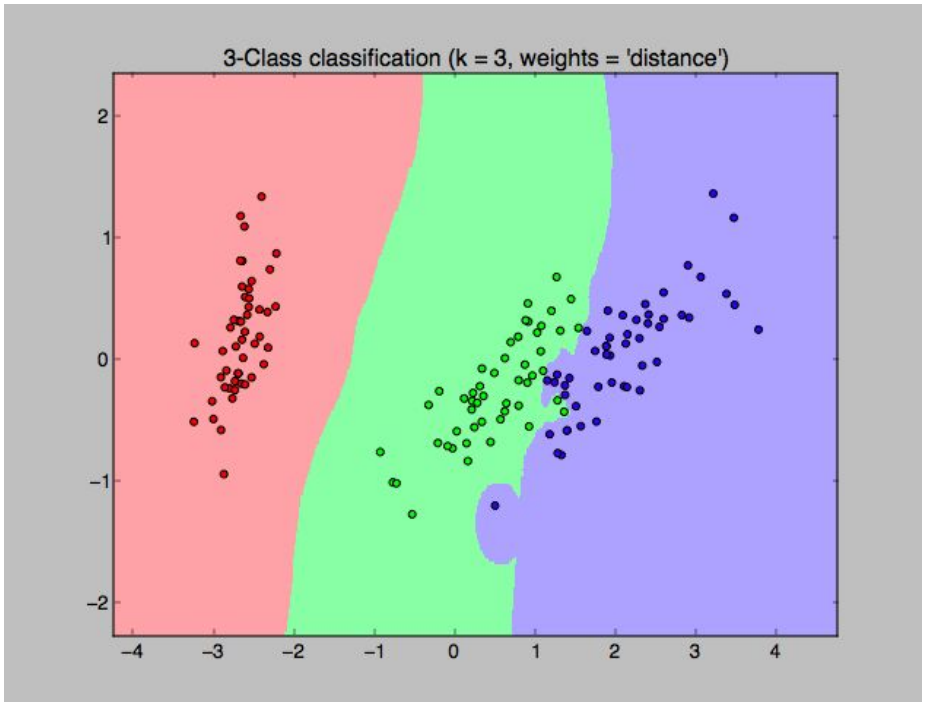


Figure 2.c

Accuracy Testing:

```
Digits data set with KNN

weights 1
Training...
(1438, 64)
Testing...
(359, 64)
accTr =  0.986787204451
Press enter to continue...
Done
Digits data set with KNN after PCA

weights 1
Training...
(1438, 2)
Testing...
(359, 2)
accTr =  0.705146036161
Press enter to continue...
Done
```

Figure 3.a: Digits accuracy k=15, n_components=3

```
Iris data set with KNN

weights 1
Training...
(120, 4)
Testing...
(30, 4)
accTr =  0.958333333333
Press enter to continue...
Done
Iris data set with KNN after PCA

weights 1
Training...
(120, 2)
Testing...
(30, 2)
accTr =  0.958333333333
Press enter to continue...
```

Figure 3.b: Iris accuracy k=15

```
Iris data set with KNN

weights 1
Training...
(120, 4)
Testing...
(30, 4)
accTr =  0.975
Press enter to continue...
Done
Iris data set with KNN after PCA

weights 1
Training...
(120, 2)
Testing...
(30, 2)
accTr =  0.966666666667
Press enter to continue...
Done
```

Figure 3.c: Iris accuracy k=10

```
Iris data set with KNN

weights 1
Training...
(120, 4)
Testing...
(30, 4)
accTr =  0.983333333333
Press enter to continue...
Done
Iris data set with KNN after PCA

weights 1
Training...
(120, 2)
Testing...
(30, 2)
accTr =  0.975
Press enter to continue...
```

Figure 3.d: Iris accuracy k=5

```
Iris data set with KNN

weights 3
Training...
(120, 4)
Testing...
(30, 4)
accTr =  0.983333333333
Press enter to continue...
Done
Iris data set with KNN after PCA

weights 3
Training...
(120, 2)
Testing...
(30, 2)
accTr =  0.991666666667
Press enter to continue...
Done
```

Figure 3.e: Iris accuracy k=3

```
Iris data set with KNN

weights 3
Training...
(120, 4)
Testing...
(30, 4)
accTr =  1.0
Press enter to continue...
Done
Iris data set with KNN after PCA

weights 3
Training...
(120, 2)
Testing...
(30, 2)
accTr =  1.0
Press enter to continue...
Done
```

Figure 3.f: Iris accuracy k=3 weights=distance

```
Digits data set with KNN

weights 3
Training...
(1438, 64)
Testing...
(359, 64)
accTr =  0.993741307371
Press enter to continue...
Done
Digits data set with KNN after PCA with 10 components

weights 3
Training...
(1438, 10)
Testing...
(359, 10)
accTr =  0.990264255911
Press enter to continue...
Done
```

Figure 3.g digits accuracy k=3, n_components=10

```
Digits data set with KNN

weights 3
Training...
(1438, 64)
Testing...
(359, 64)
accTr =  0.993741307371
Press enter to continue...
Done
Digits data set with KNN after PCA with 20 components

weights 3
Training...
(1438, 20)
Testing...
(359, 20)
accTr =  0.994436717663
Press enter to continue...
Done
```

Figure 3.h digits accuracy k=3, n_components=20

```
Digits data set with KNN

weights 3
Training...
(1438, 64)
Testing...
(359, 64)
accTr =  0.993741307371
Press enter to continue...
Done
Digits data set with KNN after PCA with 40 components

weights 3
Training...
(1438, 40)
Testing...
(359, 40)
accTr =  0.993741307371
Press enter to continue...
Done
```

Figure 3.i digits accuracy k=3, n_components=40

Result Analysis:

The best parameters for the iris data set when using KNN and PCA are when k=3 and weight is uniform as we can see in figure 3.e. The digits data set, even when run through k = [1, 3, 5, 15, 20], never does better with PCA while having weight be dependant on distance gives 1.0 accuracy every time, seen in figure 3.f. When running KNN and PCA on the iris data set most choices of k do worse when PCA is added as we demonstrate in figures 3.b, 3.c, and 3.d.

Although the KNN example graphs, figures 1.a and 1.b, start off with a k of 15, it is interesting to note how the classification changes when k is 3. A k of 15 has a cleaner cut between the data in between the green and blue areas of the graph. When we decrease k the lines on the graph become more tangled. We found 3 to be the best accuracy for KNN with PCA and output the graphs of figure 2.b and 2.c to show how the data is laid out at those times. When k was 15 the weights being uniform or distance had no effect on the graph, but when k was 3 the difference between the graphs was visible.

Overall the classification of the Iris data set performed well with PCA when k was smaller. One the biggest advantage to using PCA is to deal with excessively large data

sets. This made it disappointing when the digits data set, which is significantly larger than the iris data set, didn't perform as well with the PCA.

These results then prompted more testing of different parameters in PCA and KNN. We have, up to this point, looked mainly at the parameters in KNN, thus we turned to the PCA parameters. One of the most important parts of PCA is the number of components. The n_components parameter has to be less than the n_features parameter. For the iris set this means the most n_components can be is 3, while the number for the digits set is much larger.

We see this in figures 3.g, h, and i. The best value for n_components for the digits data set then is n_components=20. Now when we compare the iris set and the digits set we can see that the best performance of accuracy, 0.992 and 0.994 respectively, PCA can perform just as well if not better for the larger digits set.

Conclusion:

One of the biggest takeaways of this project was the use of PCA for making data more manageable and reducing noise. When working on projects without access to large computing power PCA and related algorithms can be invaluable. A downside to PCA is that we have to know ahead of time that our data is highly correlated for it to work well. Finally the relationship between the number of components and the number of features proved to be highly relevant to the practical application of PCA.

If there was more time to continue this project we would look into more datasets with various number of classes to them. The Iris data set as a 3 class data set and the digits data set with 10 classes doesn't provide a deep enough look into how the amount of classes affect the use of PCA. We would also look into other classifiers such as Decision Trees and the Perceptron algorithm to compare with k nearest neighbors.

Citations :

1. **Ravi, K. (2014, August). PCA and KNN: Reduce model complexity and improve training efficiency. Retrieved December, 2016, from http://keshavanravi.com/wp-content/uploads/2014/10/pca-knn-reduce.pdf**
2. **N. (2012, October 19). Introduction to PCA and FactorAnalysis. Retrieved December 08, 2016, from ftp://statgen.ncsu.edu/pub/thorne/molevoclass/AtchleyOct19.pdf**