Analysis of the Titanic Dataset
Peter Magnuson
Mathematical Aspects of Data Analysis

Abstract

I have decided to test my skills in data analysis by analyzing a dataset using techniques learned throughout the semester. I have chosen the commonly known Titanic dataset, in part because I have experimented on it previously using regressions,[1] and would like to verify that I have grown more advanced in this field. Additionally, it is a binary classification problem, so that can be helpful when observing the data. I will be experimenting with PCA, KPCA, and comparing the classification accuracy of the resulting lower-dimensional data and original high dimensional data.

Methodology

The Titanic dataset is constructed from real world data, namely the passenger survival data from the infamous disaster. Other than a binary feature indicated whether a passenger survived, the data contains 7 other features for each passenger: class, came, sex, age, siblings/spouses aboard, parents/children aboard, and ticket fare. In addition to working on the data without the survival indicator, I am also discarding the passenger name data, as I see it both difficult to discretize the name, and little use in doing so.

I first tried the traditional PCA in order to reduce the dimensionality. I found two marginally larger eigenvalues, and transformed the data set into the two dimensional space spanned by the corresponding eigenvectors. The below results display that data, where blue indicated a passenger surviving, and red indicates their death:

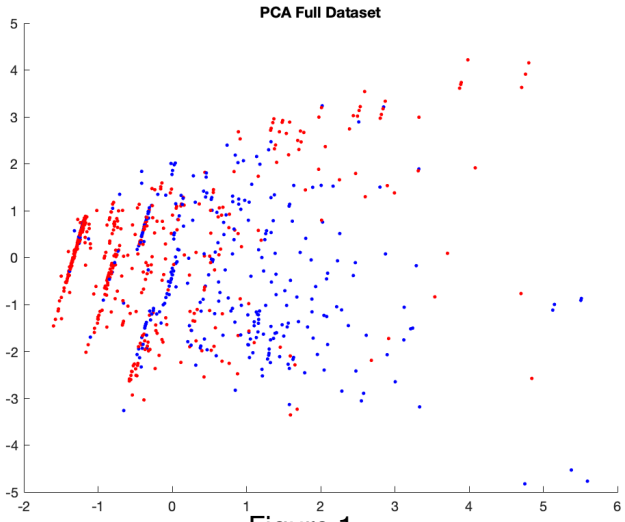

Figure 1

[1] This was in an intro to ML class. We ran only high-dimension linear regressions, my computer has since crashed, so I do not expect to be duplicating really any work.

As you can see, there might be some separation in the data, but it is hard to determine the nature of it.

Next, I experimented with preforming PCA on subset of the data. One such subset was that in which all passenger class, and fare data had been removed:

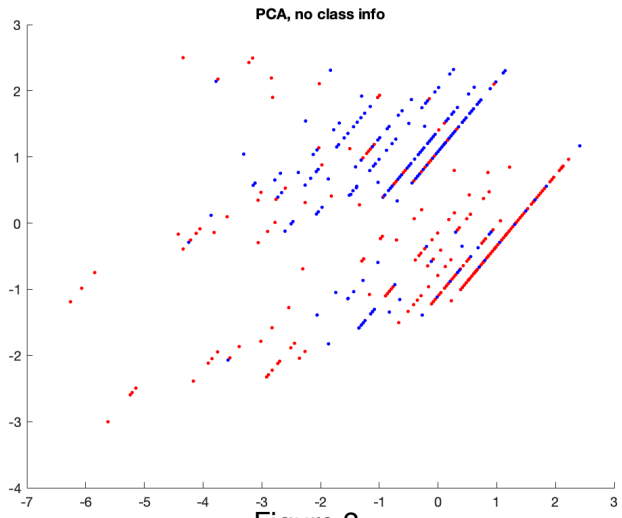

Figure 2

There appears to be a more distinct separation in the data, but whatever the separating hyperplane, it doesn't appear linear in two-two-dimensions. Finally, the most interesting result achieved through this method came from removing spousal count from the the data used to generate Figure 2. The resulting picture:

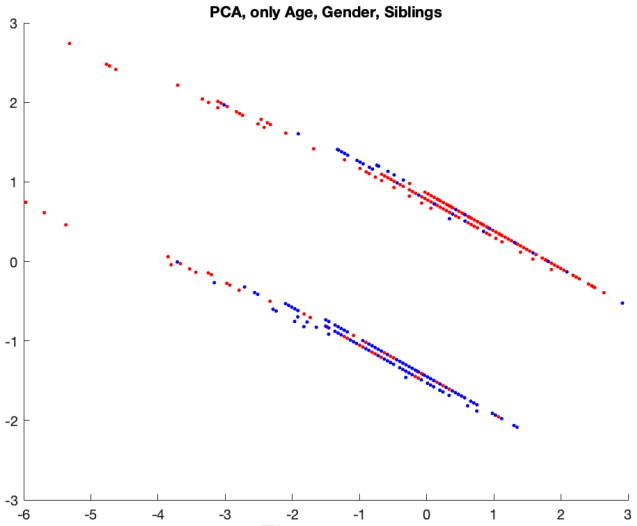

Figure 3

While not perfect, Figure 3 appears to have the data almost separated by a line, and provided clues as to what to do next.

Next, I decided to run similar experiments, but replace PCA with KPCA. First, I tried KPCA using the gaussian kernel and the full dataset. Below is the picture for sigma = .8, transformed into the vector spaced described by two principal components:

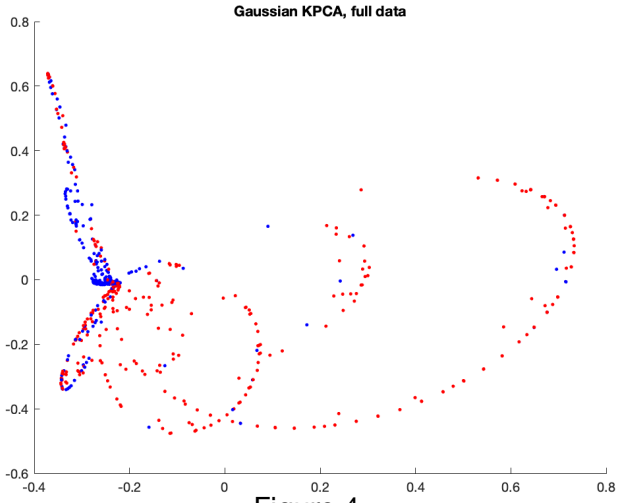

Figure 4

As seen in the picture above, the improvement over normal PCA, in terms of separation of data, is significant. This result became even more evident when the Gaussian kernel was applied to the restricted dataset from Figure 3, sigma = .9:

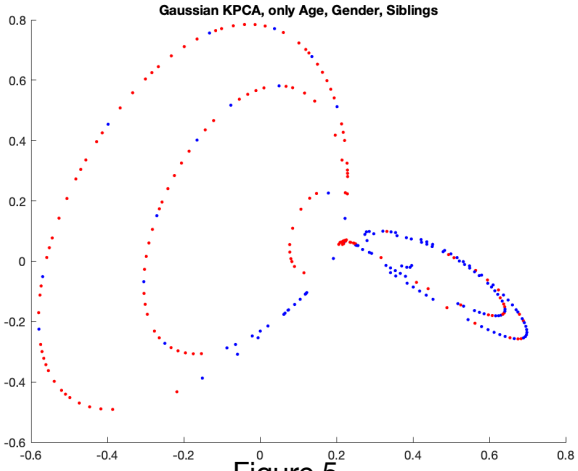

Figure 5

As shown, the Gaussian kernel appears to have relatively good job separating the restricted data. Additionally, I experimented with applying the polynomial kernel, with some success. The data transformed into the two primary components for p = 3:

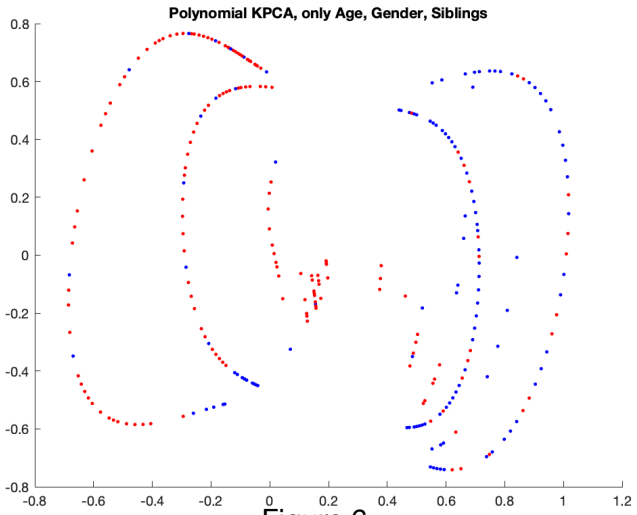

Figure 6

As can be observed, the polynomial kernel seems to have done a better a better job creating a large margin between the general classes, but also appears to have more mixing of the classes in those groups. This inspired the idea to attempt to both well-separate the data, and keep the classes relatively homogenous, by transforming the data into a vector space of the first principal component Gaussian KPCA and the first of polynomial KPCA. The result, using sigma = .9 and p = 3 on the restricted dataset:

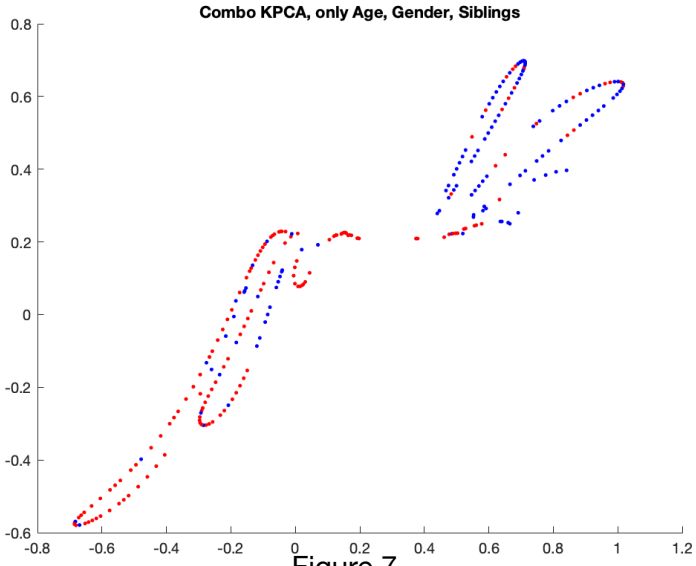

Figure 7

Finally, I wanted to test the classification improvements when using an SVM on the high dimensional data in comparison to our new kernelized datasets. Using Matlab's built in SVM with no additional options. I trained an SVM on the high-dimensional and transformed data, and tested the accuracy though the k-folds cross-validation method, where k = 10:

Data Form	Train Time (Seconds)	Accuracy
Original	1.904593	78.58%
PCA All Features	4.174031	66.07%
PCA, Only Age, Gender, Siblings	1.978597	78.13%
Gaussian KPCA, AGS	0.015401	75.42%
Polynomial KPCA, AGS	0.018774	77.43%
Combo KPCA, AGS	0.013911	72.83%

Conclusion

Overall, KPCA appears to be a great first step when attempting data analysis and data classification. As shown in the table above, transforming a dataset with KPCA allowed SVMs to train 100x faster than training on the higher dimensional dataset, while losing only a marginal amount of accuracy. In addition to building my skills and confidence in manipulating datasets, this project raises questions of how to find good kernels quickly, and methodologies to confidentially thrown out useless data, as I did every step of the project with the full dataset and the reduced dataset, and found the reduced dataset preforming better almost always.

References

Titanic dataset found at: http://web.stanford.edu/class/cs109/titanic.html

All code used was adapted from the code examples in the dropbox folder.

Lecture Slides.