Final Project - Comprehensive Classifier Creation

Sam Neyhart
Jon Clark Freeman
Kevin Sayarath
ECE 471 - Professor Qi

April 23, 2017

Abstract

The objective of this final project was to integrate various classification techniques to achieve the best performance on the chosen dataset. This project was a group effort where we explored both supervised and unsupervised classificaiton techniques including MPP (all 3 cases), kNN (with different k's), BPNN (developed), decision tree, and k-means, winner-take-all, and kohonen maps. We also explored two classifier fusion techniques including majority vote fusion and naive Bayes.

Introduction

This project performed for ECE 471 is an exercise using all the classificaiton methods we have used in past projects with the additional exploration of two new fusion methods which we previously had no experience with. The dataset we are using is titled "Poker Hand Dataset," publised by Robert Cattrel et. al., released in January 2007 *link to dataset*. By experimenting with various classification techniques, we learn the best specific methodology to classify this particular dataset.

Overall the project served its purpose and was an opportunity to review and become experts with the classification methods learned during this course. We are proud to announce we did not use any supporting libraries that do heavy-lifting of core computations, i.e. all classification methods were re-written by team members for this project.

Overall the project served its purpose and was an opportunity to learn much about the concepts of this course. For this lab the numpy library was very helpful.

Technical Approach

MPP The mpp.py Python script implements the MPP algorithm using basic Python, the numpy library, and the matplotlib library. It performs MPP parametric-based classification by first calculating the mean and covariance matricies from the dataset. *Bayes/Discriminant Func*

Case 1 The features are statistically independent, and have the same variance. Geometrically, the samples fall in equal-size hyperspherical clusters. Decision boundary: hyperplane of d-1 dimension. This classification technique employs the linear discriminant function and linear machine. Additionally, when prior probabilities are the same, the discriminant function is actually measuring the minimum distance from each feature to each of the c mean vectors.

Case 2 The covariance matrices for all the classes are identical but not a scalar of identity matrix. Geometrically, the samples fall in hyperellipsoidal clusters. Decision boundary: hyperplane of d-1 dimension.

Case 3 No assumption: the covariance matrices are different for each class. Quadratic classifier. Decision boundary: hyperquadratic for 2-D Gaussian.

kNN The knn.py Python script implements the kNN algorithm usign basic Python, the numpy library, and the scipy library. It performs kNN classification, or majority voting, using Euclidean distance to assign a random sample according to the majority representation of classes within the enclosing hypersphere of k nearest neighbors. We experiment with different k's and evaluate performance.

BPNN The backprop.py script implements the backpropogation algorithm using basic Python, the numpy library, and the random library. Backpropogation is a type of multilayer feedforward network that calculates the difference between unit output and expected output, "back-propogating" this delta from the output layers back toward the feeding layers. This technique causes the network to "learn" correct classifictions in a supervised architecture. *num layers*

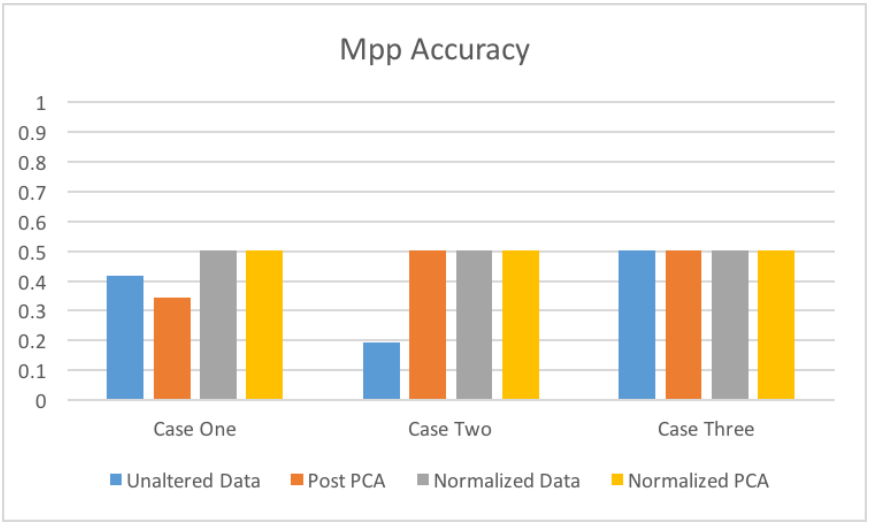
Decision Tree The dtree.py script implements the decision tree architecture using basic Python and the numpy library. Lec20 - slides 7, 8, 9, 11, 12, 13

K-means The *kmeans.py* file impliments the k-means algorithm using basic python and the numpy library. It will perform k-means clustering on the data and then subsequently output the mean values for the requested number of clusters. This implementation is incredibly slow however so in order to speed the process of obtaining the various versions of the *flowers.ppm* image required, the Project4.py file was created.

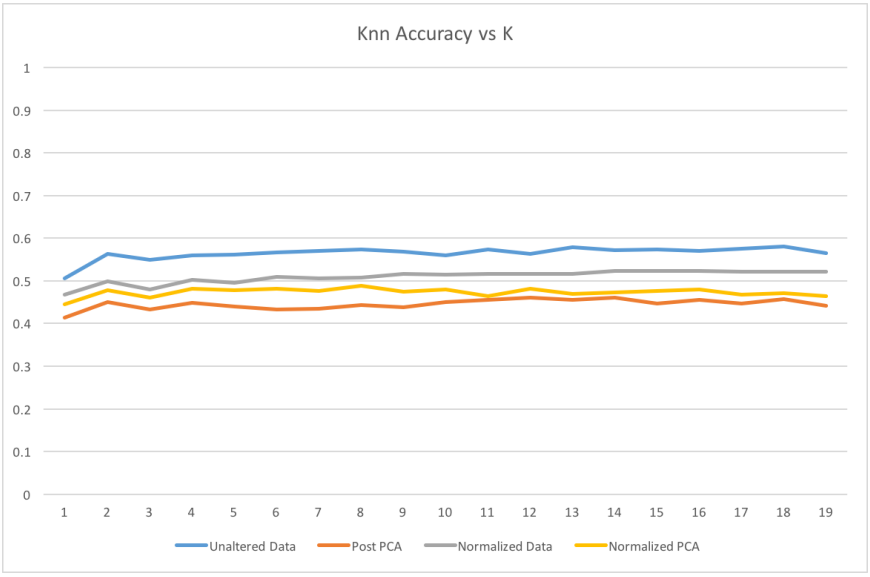
Winner-Take-All This project makes use of the file *winner.py* which contains instructions which implement winner-takes-all clustering. This algorithm is much faster than the k-means algorithm. The winner-take-all clustering algorithm is initialized with random cluster centers which can result in differing final values.

Kohonen Maps SOMETHING

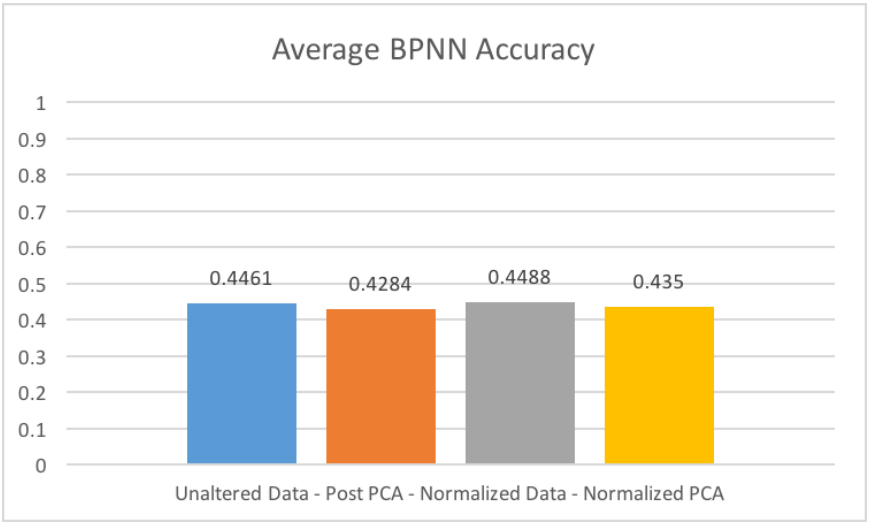
Experiments and Results



MPP



kNN



BPNN

Decision Tree The Decision tree algorithm was actually able to classify with an accuracy of one hundred percent when run on the testing set. The algorithm simply ran through each of the hands and had a node that decided for each of the groups since each group is defined in such a way that is known before hand and has to be a certain combination of the cards. This lead the algorithm to be very fast as well since it only goes through the data set one time to classify each hand.

K-means

Winner-Take-All

Kohonen Maps The original image *flowers.ppm* is displayed above. All experiments are performed on this image. The first expiriment is the kmeans clustering of colors in *flowers.png*. The resulting images can be seen on the following page. To generate these images k-means clustering was performed with [1, 2, 4, 8, 16 , 32, 64, 128, 256] clusters respectively.

The k-means images are seen above. The visual differences on these images are easy to see for the first couple images but by k=16 it becomes harder to find difference between the original image and the processed image. A list of the colors included in the each of these images is found in the apendix.

Discussion

This lab was very interesting to perform. Using python was a good choice because it greatly simplified the k-means part of the lab. The lab certainly increased my understanding of clustering algorithms and gave me greater insight to how and why they are used. The whole concept of unsupervised learning is very interesting and it was fascinating to see it in action here.

Discussion

This lab was very interesting to perform. Using python was a good choice because it greatly simplified the k-means part of the lab. The lab certainly increased my understanding of clustering algorithms and gave me greater insight to how and why they are used. The whole concept of unsupervised learning is very interesting and it was fascinating to see it in action here.

Appendix

Project4.py

kmeans.py

winner.py

kmeans-colors.txt

wta-colors.txt