Project 1 Report

By: Harun Gopal

Table of Contents

Introduction

For this project, I implemented a total of four classifiers: PCA, LDA/MDA , Bayes' Classifier, and k-Nearest Neighbor Classifier. In this report I will discuss some of the experiments I conducted using these classifiers and what these experiments taught me about these classifiers. I discuss each task and their respective experiments separately despite using the same classifiers for each. Although I did implement all four of these classifiers in MATLAB, I ran into some bugs that I was yet unable to fix despite my best efforts and help from others. Most notably, my Bayes' Classifier does not work with data containing large number of classes and my k-NN classifier only achieves as well as baseline on testing data. Despite these bugs, I attempt to conduct experiments to learn something new about these classifiers by playing around with them. My PCA and LDA/MDA implementations work wonderfully as I was able to test them thoroughly. Additionally, I won't be discussing any code here. I have commented all the code provided so it should be easy to follow exactly what I do.

Task One

Introduction

In Task One, the objective is for a classifier to determine subject number given an image. For this task, I chose to use the illumination data and classify the subjects in that dataset. As discussed in the previous section, there are a few unsolved issues with my implementation that make it difficult to collect useful data for task one. As such, I will put a much bigger emphasis on my PCA and MDA functions with respect to data with large number of classes. In the Bayes' Classifier and k-NN Classifier sections I will discuss the progress of my current implementation and what I would have to do going forward to improve upon them.

Bayes' and k-Nearest Neighbor Classifier

My Bayes' Classifier and k-Nearest Neighbor Classifiers unfortunately do not provide valid testing accuracy data. My Bayes' Classifier runs into a problem where the covariance matrix becomes singular and because of that, the classifier can't compute the discriminants of higher number functions. I found this odd because the same exact code worked for datasets with less classes without the singularity issues.

My k-NN classifier technically does output testing accuracies, however they are so close to the baseline 7% that I think they are negligible. There is some bug that essentially makes my k-NN guess during testing instead of using the learned knowledge. Unfortunately, I was not able to discover and fix this bug in time.

These bugs are present even though I made sure to follow the algorithm for each corresponding classifier. I am certain that if I had put even more time into this, I would eventually fix this bug. Not having either of these classifiers functioning for Task One makes it very difficult to showcase my working PCA and MDA functions.

PCA + MDA

I was able to implement PCA and MDA and use it to reduce the dimensionality of the illumination data. Given that my Bayes' Classifier and k-NN Classifiers do not provide valid testing accuracies, it is difficult to perform an experiment to showcase the effect of this kind of dimensionality reduction. Given that PCA does not change with the number of classes present, the experiment I do with PCA in Task Two should suffice to prove that it works. However, MDA is dependent on the number of classes present. As to refrain from including code in this report, I have created another MATLAB live script called "MDAandPCA.mlx" where I display that my code for these two dimensionality reduction tools do indeed work.

Task Two

Introduction

In Task Two, the objective is for a classifier to distinguish between a neutral and facial expression. Given this scenario, the classification algorithm would only have to distinguish between two classes, which is considerably simpler than the large number of classes present in Task One. Regardless, of this simplicity, fine tuning the parameters is still necessary to achieve the optimal results. The following sections detail the parameters I played with and the effects to the classifier I observed due to the changes I made

Bayes' Classifier

When testing the accuracy of the Bayes' Classifier on the expression data of Task 2, I often found that the accuracy numbers would always hover around 50%. Given that this classification task only had two classes to classify to, the baseline accuracy for if the classifier just guessed would be 50%. This signified to me that gaussian classification in this situation was doing quite poorly to the point where guessing would produce nearly identical results. I suspect this occurs because the Bayes' Classifier assumes a gaussian distribution which I am inclined to conclude is incorrect for our data.

However, despite the low testing performance, the Bayes' Classifier performed exceptionally with respect to fitting the training data. This would make sense because the Bayes' Classifier is known to create a fit as to minimize error within the training set. Even despite the required assumptions for the Bayes Classifier being wrong, it still manages to achiever over 95% accuracy on the training set.

Test Data percentage split	Testing Data Accuracy	Training Data Accuracy
30%	52.5%	99.64%
20%	47.5%	99.06%
10%	55.5%	98.20%

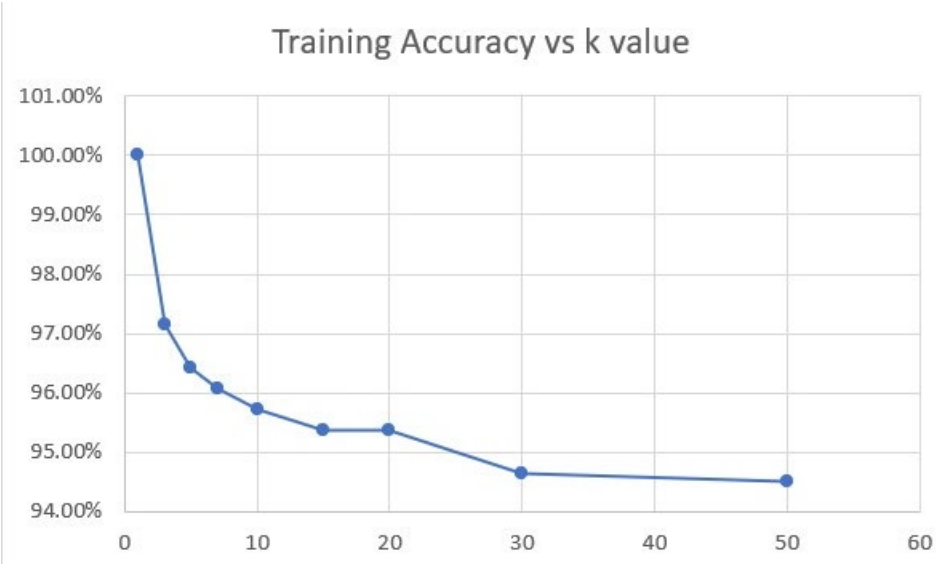
Overall, there seems to be no correlation between the Test Data Percentage and the accuracy of the classifier. The only noticeable difference I found was increased variability of the testing data accuracy as the percentage decreased. To obtain this data, I preprocessed the data by decreasing the dimensions

down to 200 via PCA before using LDA to compress the data down to one dimension. Overall, this table shows Bayes' Classifier is great at fitting to the training data but does not generalize well in this example.

k-Nearest Neighbor Classifier

Unlike Bayes' Classifier, there is a very clear parameter that has an immediate effect on testing performance when it comes to the K-Nearest Neighbors algorithm. Of course, that is the value of k itself. I tested various values of k to see its impact on testing performance. When running my K-NN algorithm, I achieve an average of about 55% on testing data, but with very high variability. This suggests to me that my algorithm has some kind of bug because I don't expect the accuracy of this classifier to be that low. This bug persisted despite my best attempts to fix it and talking to Chenghao(TA) about it. Due to this, I decided to check to see how the k parameter affects the accuracy on the training data rather than the testing data. To obtain the best results, I reduce the dimensions to 100 via PCA, and then reduced the dimensions again to 1 via LDA.

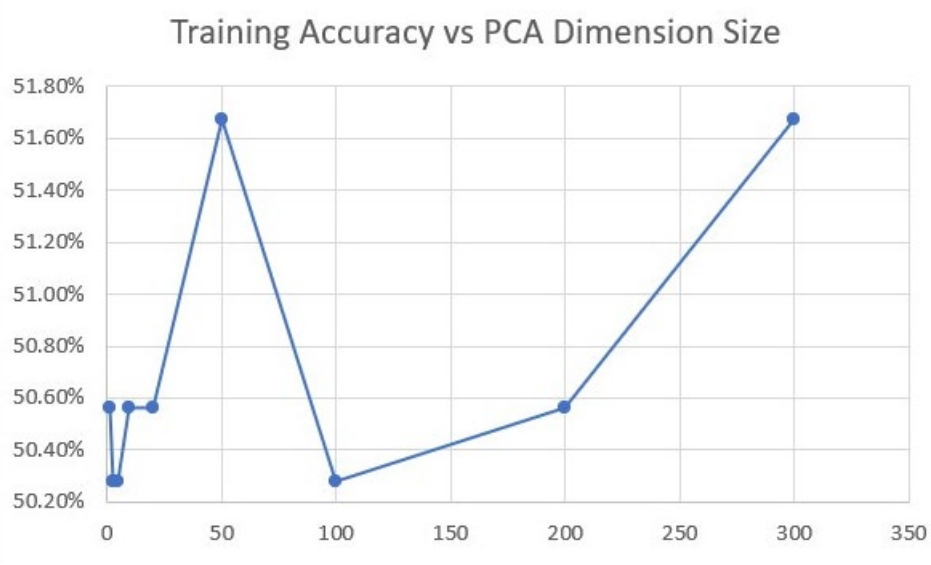
To obtain the best possible results, I reduce the dimensions of the data to 100 via PCA, and then reduced it further to 1 dimension via LDA. Due to a bug in my code that eluded me, my accuracies had very high variability so to compensate I took an average over 10 runs each to get the best representation of the data.



This data shows what we would expect. When k=1, the accuracy is 100 because the algorithm would have the training data stored and remember what their labels are. However, when k = 2 or higher, we see a huge drop in training performance. This doesn't necessarily correlate to testing performance though as a higher k value may cause a higher testing accuracy.

PCA Dimensions

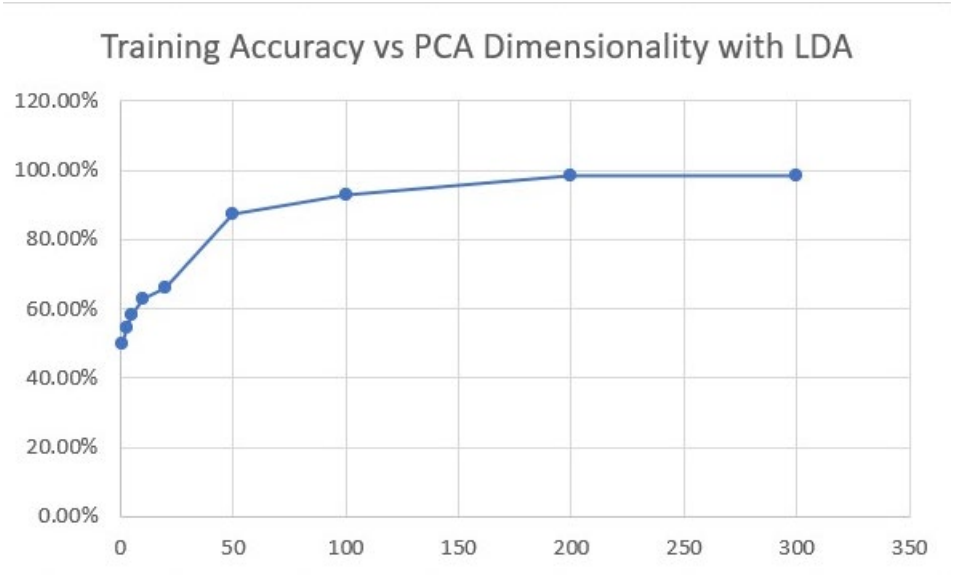
PCA is a great tool that allows us to reduce data with a large dimensionality to a more manageable size. My implementation of PCA works well, however it is difficult to quantify when both implementation of my Bayes' Classifier and K-NN Classifier offer testing accuracy so close to baseline. So instead, I will be viewing the effect of different dimensions of input data on the training accuracy (as opposed to the testing accuracy) of the Bayes' Classifier.



As this data shows there really was no direct correlation between the number of PCA dimensions and the training accuracy of the Bayes Classifier. I suspect that this is because I didn't implement LDA which would create much greater separability between the classes. I will explore this theory in the next section.

Linear Discriminant Analysis

Since Task Two only contains two classes, the highest dimension I can LDA to is only one dimension. To see the effects of LDA in this scenario, I will implement a preprocessing pipeline where I reduce the dimensionality of my original data via PCA, and then feed that into my LDA to return 1 dimensional data. I will then calculate the training accuracy of a Bayes' Classifier on it.



This data shows a clear correlation between the number of PCA dimensions and the training accuracy of Bayes' Classifier. However, this correlation only presented itself after implementing LDA dimensionality reduction on the PCA data. This indicates a few things. First, PCA doesn't by itself help with classifying the data points into classes. The LDA function does a much better job of helping the Bayes' Classifier classify data points. Secondly, this graph shows how PCA dimensionality reduction does indeed reduce the overall information present within the system. As the number of PCA dimensions increased, the LDA was able to do a better job separating the data by class which resulted in a better Bayes' Classifier training accuracy.