Ryan Frohman

Project 1 Report

Classification Task 1: Identify Subject Label from Test Image

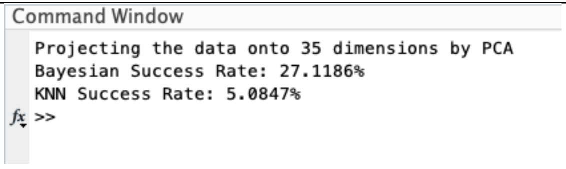
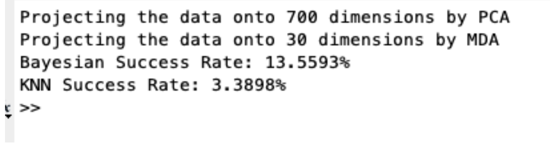
For this task, I sought to take in training data (in the form of images) from various subjects to train a classifier that could read test data and determine which subject each test image is. For the task, the data I chose was *pose.mat,* as it seemed like an optimal choice for this task: *data.mat* only contains 3 images per subject, which would provide poor classification due to the low amount of training images, 2. *Illumination.mat* and *pose.mat* both seemed like good choices since they had more than 3 images per class, so I chose the latter.

To start off, the data was split into training and test data. Roughly 80% of the data from pose.mat was used as training, and the other 20% was used as test data after the classifier had been trained. The first objective post-split was to apply a PCA on the dataset.

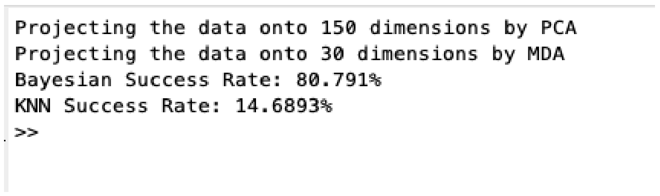
To do so, I calculated the means of the data overall. After this was done, I was able to center the data, allowing me to compute the centralized covariance matrix $\hat{C}$. Once $\hat{C}$ was found, I found the m optimal dimensions to project the data onto by finding the m eigenvectors associated with the m highest eigenvalues. I then projected the centralized training and test data onto the m eigenvectors corresponding to the m highest eigenvalues.

When considering an optimal choice for m, I used various values to maximize the accuracy of the classifier. When using a value too low such as m = 35, the classifier was too inaccurate to successfully classify the test data. When using a value too high such as m = 800, the classifier was also too inaccurate to successfully classify the data, as the matrix was extremely small, appearing almost singular. The results of these two projection choices are shown below:

M = 35	M = 700

 Command Window
`Projecting the data onto 35 dimensions by PCA`
`Bayesian Success Rate: 27.1186%`
`KNN Success Rate: 5.0847%`
`>>`
	 `Projecting the data onto 700 dimensions by PCA`
	`Projecting the data onto 30 dimensions by MDA`
	`Bayesian Success Rate: 13.5593%`
	`KNN Success Rate: 3.3898%`
	`>>`

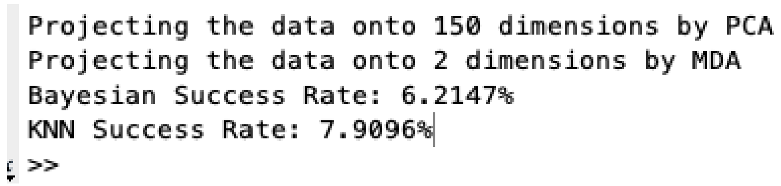
It is with this in consideration that I chose a value of m = 150. The number of dimensions is not too small such that it discards key information about the data (shown in the eigenvalues), but it also does not risk the data being thrown off by too high of a dimensionality. The results of this are shown below. With m being 150, this gave me the highest accuracy results for both Bayesian classification and k-nearest neighbor classification.



```
Projecting the data onto 150 dimensions by PCA
Projecting the data onto 30 dimensions by MDA
Bayesian Success Rate: 80.791%
KNN Success Rate: 14.6893%
>>
```

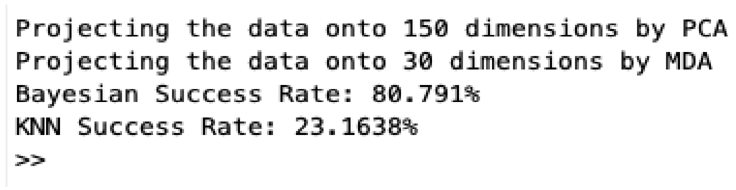
After performing a PCA, I performed an MDA on the data to project it further down while aiming for maximal discriminability. After calculating the now m-dimensional mean vectors $\underline{m}_i$ for each class i = 1,…,68, I calculated $\underline{m}$, the weighted average of all of these class averages. I then used this to calculate the between-class scatter matrix S_b and scatter matrix of each class S_i for i = 1,…,68, and then calculating S_w , the sum of all S_i 's. I then found the optimal n directions to project the data down onto by finding the n eigenvectors corresponding to the n highest eigenvalues of $S_w^{-1}S_b$. I again projected both the training and test data down to n-dimensions.

When considering a value for n, it is important to remember that the number of dimensions for linear discriminant projection must be less than the number of classes. Keeping this in mind, it is also important to keep crucial information about the data (highlighted in the eigenvalues) by not projecting the data too low. Choosing values for n such as 2 result in the classification not being as accurate as it could be with other values. Shown below are the results of n = 2.



```
Projecting the data onto 150 dimensions by PCA
Projecting the data onto 2 dimensions by MDA
Bayesian Success Rate: 6.2147%
KNN Success Rate: 7.9096%
>>
```

After experimenting with various values, it appears that choosing n = 30 results in the greatest classification accuracy, shown below.

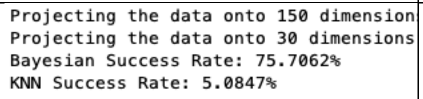
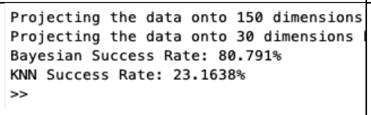
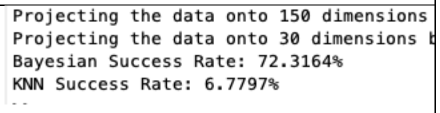


```
Projecting the data onto 150 dimensions by PCA
Projecting the data onto 30 dimensions by MDA
Bayesian Success Rate: 80.791%
KNN Success Rate: 23.1638%
>>
```

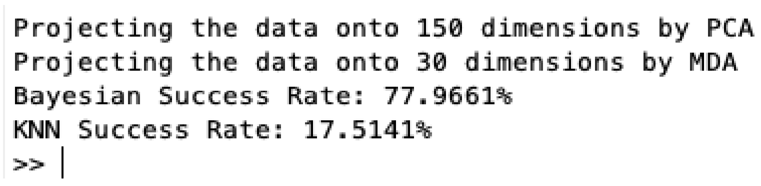
Lastly, the now n-dimensional data, projected using PCA and MDA, can be used to train classifiers. The first classifier used is the Bayesian classifier. After assuming a Gaussian distribution of the data, I found the n-dimensional mean vectors and nxn covariance matrices of each class using the MLE formulas. I then tested each n-dimensional test point and tested it into the discriminant function of each class. I then assigned the test point to the class with the highest discriminant function result. If the assignment matched the test point's true class, I added one point to the Bayesian successes, initially set to zero. At the end, I divided the number of successes by the total amount of test points and then multiplied by 100 to get an accuracy percentage. As shown above in the figures, with an optimal choice of m and n for PCA and LDA, my accuracy with Bayesian classification is between 74% and 80% on various runs. Given that there are only 8-12 images to train with for each class, this is a quite good accuracy rate. With more training data, the classification accuracy can be even higher.

The other type of classification performed was k-nearest neighbor classification. For each test point, I found the k-nearest training points to that point and assigned the test point to the class with the highest representation in the k-nearest points. If there happened to be a tie, I would sum up the individual distances for each of the points in the classes that tied and then pick the class with the smaller sum (overall closer points). This did not occur often due to doing projection onto optimal directions using PCA and LDA. But when it did, I resolved the issue this way.

Unfortunately, even with projection of the data onto the optimal dimensions retaining as much of the data as possible, the k-nearest neighbor classifier could not classify the data well enough to be greater than 25% accuracy. The projection down works great under the Bayes classifier, but quite poorly under this one. With that said, I still experimented with the values of k to get the greatest possible accuracy. The accuracy seemed to peak at values of k in the range 20 to 25, and dropping on either side of that range, shown below in the figure.

K = 2	K = 20	K = 300
 Projecting the data onto 150 dimension Projecting the data onto 30 dimensions Bayesian Success Rate: 75.7062% KNN Success Rate: 5.0847%	 Projecting the data onto 150 dimensions Projecting the data onto 30 dimensions Bayesian Success Rate: 80.791% KNN Success Rate: 23.1638% >>	 Projecting the data onto 150 dimensions Projecting the data onto 30 dimensions b Bayesian Success Rate: 72.3164% KNN Success Rate: 6.7797%

Overall, the average results of Classification Task 1 are shown below under what I found to be the best possible conditions to work with.



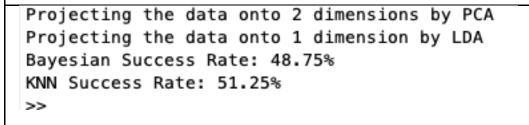
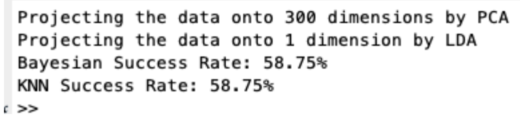
```
Projecting the data onto 150 dimensions by PCA
Projecting the data onto 30 dimensions by MDA
Bayesian Success Rate: 77.9661%
KNN Success Rate: 17.5141%
>>
```

Classification Task 2: Neutral vs. Facial Expression

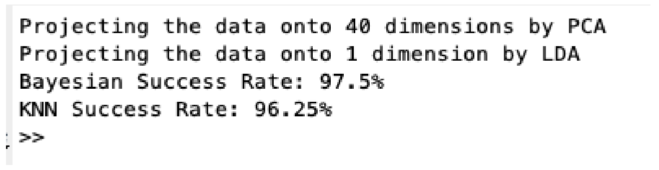
For the second task, I sought to take in training data from *data.mat*, containing neutral and facial expression images, and train a classifier to look at test images and determine whether they are neutral faces or contain a facial expression. The data for this task was split in the same manner that it was for task 1, referenced above.

Next, I applied PCA on the dataset. I first computed the centralized covariance matrix $\hat{C}$, then found the m eigenvectors corresponding to its highest m eigenvalues of $\hat{C}$ and projected the training and test data down.

When considering an optimal choice for m, I used various values to maximize the accuracy of the classifier. When using a value too low such as m = 2, the classifier was too inaccurate to successfully classify the test data. When using a value too high such as m = 350, the classifier was also too inaccurate to successfully classify the data as well. The results of these two projection choices are shown below:

M = 2	M = 300
 Projecting the data onto 2 dimensions by PCA Projecting the data onto 1 dimension by LDA Bayesian Success Rate: 48.75% KNN Success Rate: 51.25% >>	 Projecting the data onto 300 dimensions by PCA Projecting the data onto 1 dimension by LDA Bayesian Success Rate: 58.75% KNN Success Rate: 58.75% >>

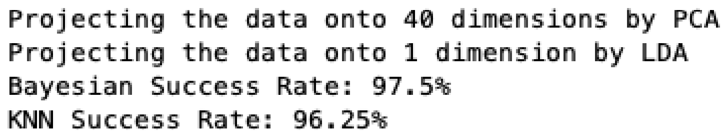
Taking this into account, I found that m = 40 had the best combination of dimensionality reduction while maintaining key aspects of the data evident in the eigenvalues. The results of m = 40 are shown below.



```
Projecting the data onto 40 dimensions by PCA
Projecting the data onto 1 dimension by LDA
Bayesian Success Rate: 97.5%
KNN Success Rate: 96.25%
>>
```

Next, I performed an LDA on the dataset. Since there are only two classes to distinguish, the only choice of dimensionality reduction is projecting the data onto one dimension. I calculated the means in post-PCA dimensions, calculated $S_w = S_1 + S_2$, and then found the optimal direction w by calculating $S_w^{-1}(m_1 - m_2)$, with m_1 being the mean of the one-dimensional neutral data and m_2 being the mean of the one-dimensional facial expression data. I then projected the data onto w.

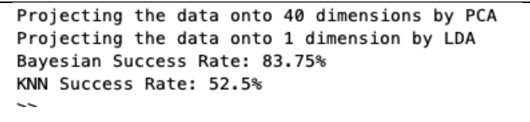
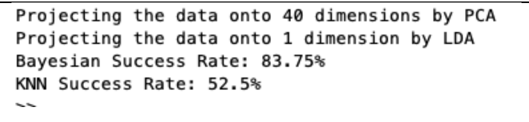
After projecting the data down to 40 dimensions with PCA and 1 dimension with LDA, I performed two types of classification. The first performed was Bayesian classification, in which I assumed a Gaussian distribution of each class and calculated sample means and variances using MLE formulas. I then put each one-dimensional point through the two discriminant functions for the two classes and picked the class with the higher value. Then, I check my classification and see if my assigned class is equal to the test point's true class and add one to the successes for each correctly assigned test point. I then check the Bayesian accuracy by dividing the number of successes by the size of the training data. Overall, the Bayesian classification is very accurate for the dataset, ranging from 80% to 97% accuracy (shown below).



```
Projecting the data onto 40 dimensions by PCA
Projecting the data onto 1 dimension by LDA
Bayesian Success Rate: 97.5%
KNN Success Rate: 96.25%
```

The second classification type is k-nearest neighbor classification. For each test point, I find the k nearest training points and assign the test point to the majority class representation. To get the majority class representation, I just sum up all of the values in the list of the k nearest points and assign class 0 (the neutral expression) if it's less than k/2, and 1 otherwise.

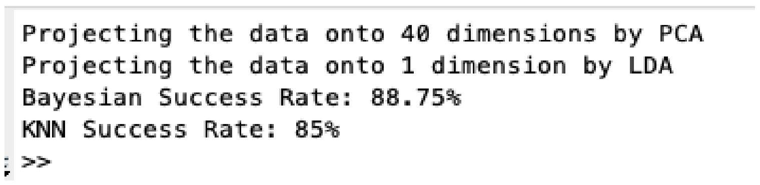
When designating what value of k to use, I found that using very small values such as 1 provide minimal accuracy, and very large values such as 315 provide lessened accuracy as well. The results are shown below.

K = 1	K = 315
	

```
Projecting the data onto 40 dimensions by PCA
Projecting the data onto 1 dimension by LDA
Bayesian Success Rate: 83.75%
KNN Success Rate: 52.5%
>>
```

```
Projecting the data onto 40 dimensions by PCA
Projecting the data onto 1 dimension by LDA
Bayesian Success Rate: 83.75%
KNN Success Rate: 52.5%
>>
```

With this in light, optimal values for k tend to be in the range of 10-20, where the k-nearest neighbor accuracy appears to frequently be between 70% and 95%. Overall, the average results for classification task 2 are shown below under what I found to be the optimal conditions.



```
Projecting the data onto 40 dimensions by PCA
Projecting the data onto 1 dimension by LDA
Bayesian Success Rate: 88.75%
KNN Success Rate: 85%
>>
```