Extended Yale Faces

Ron Wilson

May 27, 2020

Abstract

Methods of data clustering & classification are explored in this write-up. Specifically, Support Vector Machine (SVM) & Gaussian Mixture Models (GMM) are used to cluster & classify the Yale Faces B database.

1 Introduction and Overview

The Yale Faces B database contains two sets of files; one of cropped faces (38 different people, each with 68 different images) and one with uncropped faces (15 different people with 11 different images for each person). In the first part of this write-up, I will perform a Singular Value Decomposition (SVD) analysis in order to determine the number of modes necessary for good image reconstruction for both data sets, and then compare the two sets. For part two, I will use the results of the SVD analysis on the cropped faces to build classifiers that can identify individual people & identify genders, as well as develop an unsupervised learning algorithm to cluster the data and try to find a pattern in the clusters.

2 Theoretical Background

In this section, I will give some background on the methods used in the two parts of this write-up.

2.1 Part 1: SVD Analysis

The Singular Value Decomposition (SVD) of an m-by-n matrix A is defined as:

$$A = U\Sigma V^*. \tag{1}$$

In the context of image recognition, the matrix A contains n columns where each column represents the data from one image. By computing the SVD, we get the m-by-n matrix U, which is the face space where the columns of U are the new bases of the data from SVD. The matrix Σ is an n-by-n diagonal matrix, containing the n singular values in descending order. The first r singular values that are the most significant is the rank of the face space. These r modes are found by diagonalizing the matrix Σ and plotting the Singular Value Spectrum (SVS). Finally, the n-by-n matrix V represents the signature of the images. To reconstruct the images, we can use the rank r found from the SVS to perform the following operation:

$$B = U * \Sigma(:, 1:r) * V^{'}(:, 1:r). \tag{2}$$

Here, we are taking only the r necessary modes from Σ and the corresponding image signatures from the V matrix.

In MATLAB, we can compute the SVD of data matrix A using the function `[U,S,V] = svd(A,'econ')` (see Appendix A). Notice that this is the economy-size SVD. Doing this gives similarly accurate SVD results as equation 1, except that this reduced SVD removes the extra rows or columns of zeros from the matrix Σ, and the corresponding columns in U & V that they would be multiplied by in order to improve the execution time of the code.

2.2 Part 2: Clustering & Classification

For part 2, there are three different tests. Tests 1 and 2 are to develop classifiers for face identification and gender identification, respectively. These tests use the supervised learning method of Support Vector Machines (SVM). Test 3 is to develop an unsupervised learning algorithm to identify patterns in the clusters of the data set. For this, I used Gaussian Mixture Models (GMM).

2.2.1 Support Vector Machines

Support Vector Machines (SVM) are used to classify a data set. Through regression, the algorithm separates the two classes of a data set by finding the separation with the largest distance between the two classes. This separation is not necessarily linear. It uses the data points in the two classes that are nearest to each other to form the support vectors that determine the separation. In MATLAB, we can use the function `Mdl = fitcsvm(X,Y)` (See Appendix A) to find the SVM model for a two class model where X is the data set and Y is the class labels. SVM can be extended to multi-class models larger than two classes. This is done in MATLAB with the function `Mdl = fitcecoc(X,Y)` (See Appendix A). This is an error-correcting output codes (ECOC) model, which reduces the multi-class data set into a set of multiple two class models to use the SVM algorithm on.

2.2.2 Gaussian Mixture Models

In order to cluster data in an unsupervised way, we can use a Gaussian Mixture Model (GMM). This fits a data set with a mixture distribution, which is a probability distribution of the observations of the data set. Specifically, this method uses a Gaussian probability distribution on the observations of the data. In MATLAB, we can use the function `GMModel = fitgmdist(X,k)` (See Appendix A) to find this fitted probability distribution on data set X with k observations. Because this is an unsupervised method, we do not know how many observations to look for, so it is important to try multiple k values to find the ways in which the data naturally clusters.

3 Algorithm Implementation and Development

Here I will outline the algorithms used to analyze the Yale Faces B database. See Appendix B for the MATLAB code corresponding to these algorithms.

3.1 Part 1: SVD Analysis

The goal in part 1 is to perform an SVD analysis to find the number of modes necessary to generate a good image reconstruction of the Yale faces. Here are the steps that I used to accomplish this (note that the algorithm is the same for both the cropped faces and the uncropped faces):

1. To load the data, I used a `for` loop to change directories to access each of the sub-folders of images. Using the MATLAB function `imread`, I get the data from each image within the sub-folders. I then reshaped the data into a column vector and stored the data from each image as columns in a matrix (so each column in the matrix represents a different image).

2. Next, I computed the reduced SVD of the data matrix. However, I converted the data matrix using the MATLAB function `double` to get it into the correct input type for SVD.

3. Now, I diagonalized the matrix Σ to produce the singular values. Dividing this diagonalization by its sum and multiplying by 100 gives the percentage of the singular values. Plotting these values yields the Singular Value Spectrum (SVS), which shows us which modes are the most important in representing the data of the images. See figure 1 for the SVS of the cropped images, and see figure 2 for the SVS of the uncropped images.

4. Finally, I reconstructed the images by computing the matrix B from equation 2 using the necessary number of modes r found from the SVS. I tested the reconstructions by plotting four random images versus their reconstructed image.

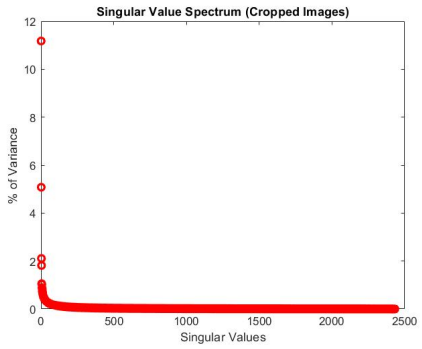

Figure 1: Cropped Images SVS

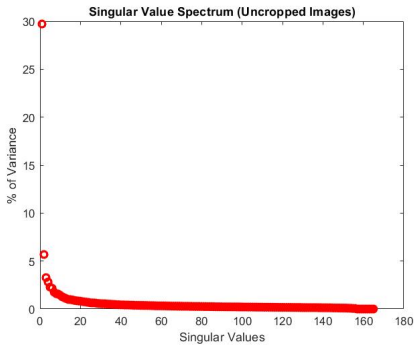

Figure 2: Uncropped Images SVS

3.2 Part 2: Clustering & Classification

The three clustering & classification tests on the cropped images are performed using supervised learning for tests 1 & 2 and unsupervised learning for test 3.

3.2.1 Supervised Learning

1. The first step for the supervised learning algorithms is to get the labels of the data set. In test 1, the labels are a matrix of values from 1 to 38 representing the individual face of the image. For test 2, the label matrix consists of only 1s & 2s where 1 is a female face & 2 is a male face.

2. Next, we need to separate the data into training and testing sets. There are 2432 different images, so I used a training set of 2000 images and saved the remaining 432 images for the testing set. These sets are chosen randomly such that each run of the code will produce different training and testing sets. These randomly generated indices are also used to get the labels corresponding to each set. Note that the data set that I am splitting into training and testing sets is the data from the signature matrix V found from SVD. Here, however, we are only using the r most important modes from V which were found in part 1.

3. Now that we have a training set and the labels corresponding to the training set, we can train an SVM classifier. Test 1 uses 38 class labels so we use the `Mdl = fitcecoc(X,Y)` MATLAB function where X is the training set and Y is the class labels. In test 2, we only have 2 class labels so we can use the `Mdl = fitcsvm(X,Y)` MATLAB function, instead. See Appendix A for these MATLAB functions.

4. Then, with the SVM trained models, we use the MATLAB function `label = predict(Mdl,X)` (See Appendix A) to generate predicted labels for the testing set X.

5. Finally, with these predicted labels, we can compare with the true labels that we had pre-defined to get the accuracy of the classification and plot this accuracy as well as the predicted labels vs the true labels.

6. One last important step is to cross-validate these classifier models. This is done with the MATLAB functions `CVMdl = crossval(Mdl)` & `loss = kfoldLoss(CVMdl)` (See Appendix A), which returns the 10-fold general cross-validation error of the classifiers. I also plotted these errors to compare the two supervised learning tests.

3.2.2 Unsupervised Learning

1. Just as we did for the supervised learning algorithm, we need to split the signature matrix V into training and testing sets. However, here we do not have class labels to identify as we are searching for the clusters.

2. Next, I used the MATLAB function `GMModel = fitgmdist(X,k)` (See Appendix A) to fit a GMM distribution to the training set X with k components. I did this twice using two different choices of k. From tests 1 & 2 we know that the data should have natural clusters of 2 (gender identification) & 38 (face identification). So I chose $k = 2$ & $k = 38$.

3. Then, we can partition the testing set into k clusters using the MATLAB function `idx = cluster(gm,X)` (See Appendix A) where `gm` is the GMM distribution and X is the testing set.

4. Knowing that a 2 cluster model could correspond to gender identification and a 38 cluster model could correspond to face identification, I took the labels corresponding to these possible classifications to compare the accuracy of the clustering models.

4 Computational Results

In this section, I will discuss the results of the SVD analysis and clustering & classification algorithms.

4.1 Part 1: SVD Analysis

Recall the Singular Value Spectrum's of the cropped images and the uncropped images in figures 1 and 2, respectively. The first piece to the SVD analysis is to determine the rank r of the face space. For the cropped images, there are four important modes separated from the rest of the singular values. Similarly, the first two modes of the uncropped SVS are separated from the rest. Thus, $r = 4$ for the cropped images and $r = 2$ for the uncropped images are the number of necessary modes to get a "good" image reconstruction. In figure 3, I plotted four original cropped images versus their rank 4 reconstructions. Figure 4 shows the rank 2 reconstructions of four uncropped images and the original image.

We can see that the facial structure for the rank 4 cropped image reconstructions is similar to the original faces, however the rank 2 reconstruction of the uncropped images is slightly similar, but very noisy. This is because the images in the uncropped data set have the faces in different spatial positions, making it difficult for the signature matrix V to represent every image at low rank reconstructions. To better reconstruct the faces, I chose to get 95% of the total variance of the data for reconstruction. By taking the sum of the modes of the SVS until we get to 95%, we found that $r = 1586$ and $r = 118$ for the cropped images and the uncropped images, respectively.

Using these 95% of the variance modes, we reconstructed the same four images for each data set versus their original image in figure 5 for the cropped images and figure 6 for the uncropped images. Here, we have much clearer reconstruction of the cropped images and we are even able to get very good reconstructions of the uncropped images. This shows that the SVD analysis is still able to capture the data from images when they are positioned differently as long as you use a large enough rank reconstruction.

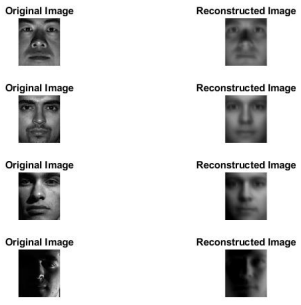

Figure 3: r = 4 Reconstruction (Cropped)

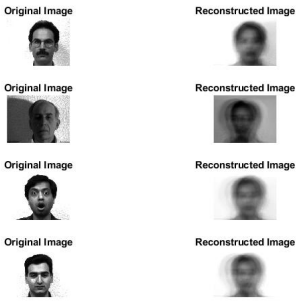

Figure 4: r = 2 Reconstruction (Uncropped)

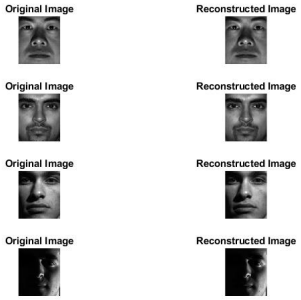

Figure 5: r = 1586 Reconstruction (Cropped)

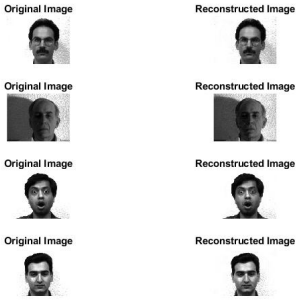

Figure 6: r = 118 Reconstruction (Uncropped)

4.2 Part 2: Clustering & Classification

In part 2, we use the signature matrix V only for the cropped images on each of the three tests. Test 1 used Support Vector Machines to develop a classifier to identify the individual 38 faces. To get the most accurate classifier, I used the first 95% of the modes from V to get the training and testing sets. In figure 7, I plotted the predictors for the testing set from SVM versus the true labels, as well as the accuracy of each predicted label in the testing set and the overall accuracy of the classifications. We can see that the classifier got most of the predictions correct. The accuracy of the classifier was 63.89%. However, it is important to note that this used a random sample of 2000 images for training and 432 images for testing. Because of the randomness of the test, this accuracy could be higher or lower each time it is run. I would also expect the test to have better accuracy if we had a much larger training set. Although, for this test it makes sense that the accuracy isn't all that high, since being able to identify individual faces is more challenging than identifying something such as the gender of the faces.

In test 2, we attempt to identify the genders to see if SVM is indeed more accurate for lower class identifications. Figure 8 displays the results of the gender classifier. As expected, the accuracy here is better than the accuracy of the face identification classifier with an accuracy of 78.70%. However, it is still important to note the randomness of the classifier. To be sure of the comparison between the two classifiers, I calculated

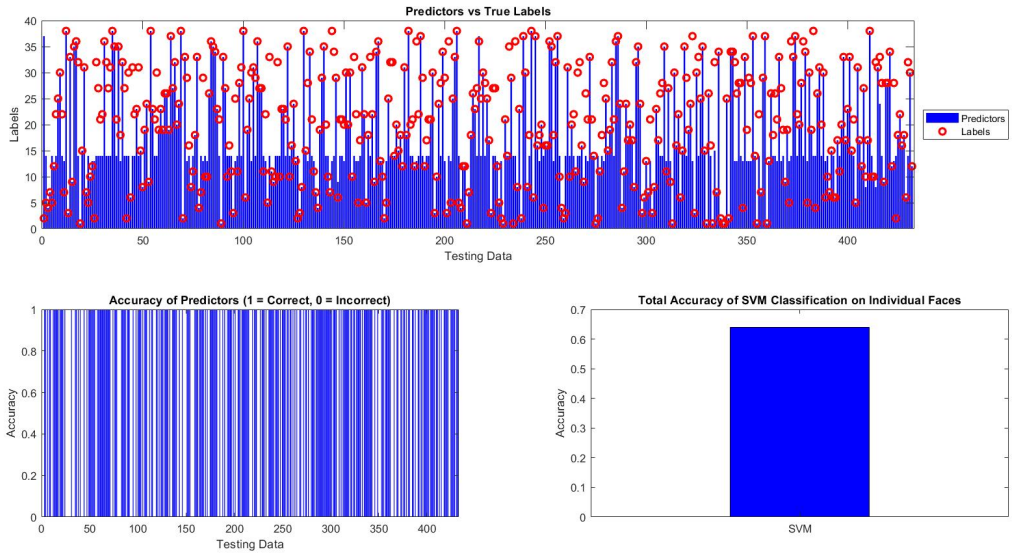

Figure 7: Support Vector Machine model for individual face classification

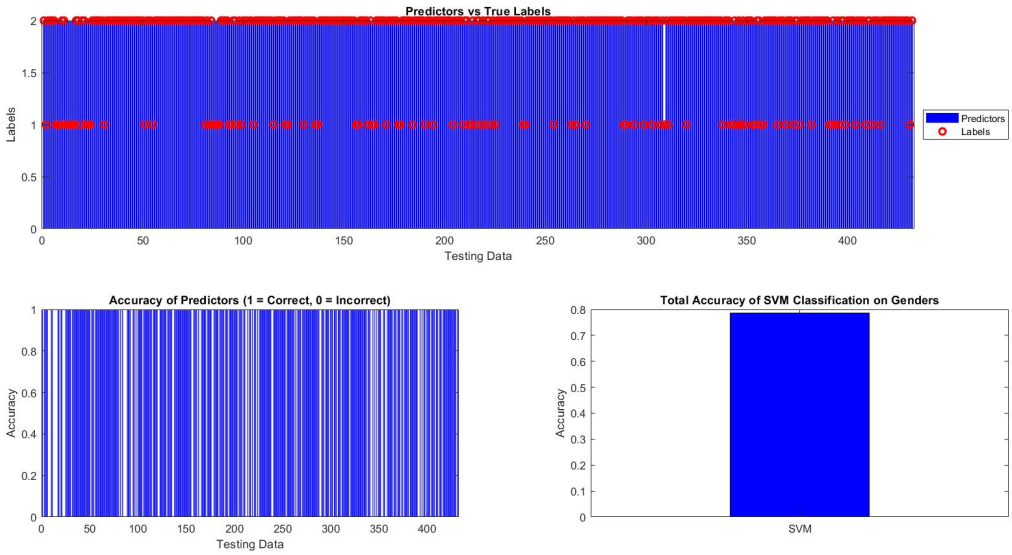

Figure 8: Support Vector Machine model for gender classification

the 10-fold general cross-validated error of each classifier to find the potential deviation in their accuracy's. These error measurements can be seen in figure 9. The CV error for test 1 was 40.20% and the CV error for test 2 was 20.95%. So, the gender classification is more reliable, which also supports the idea that it is easier for the classifier to identify 2 classes rather than 38 classes. Still, these error measurements are higher than I would like to be sure of the results of the classifications, especially the error for the face classifier. However, these errors would likely be smaller if we had a larger training set.

Finally, test 3 was to develop an unsupervised algorithm to observe patterns in the clusters. This was done using Gaussian Mixture Models. However, the MATLAB function that fits the Gaussian mixture distribution reports an ill-conditioned error when using the 95% variance modes of V. To get around this issue, I used the $r = 4$ modes of V. This will lower the accuracy of the clustering algorithm, but ensures that the solution of the Gaussian distribution converges. I ran this algorithm on two different sets of clusters; one with 2 clusters and one with 38 clusters. So the algorithm should find behavior in the cropped images that corresponds to 2 unique clusters and 38 unique clusters. In order to visualize the results of the clustering al-

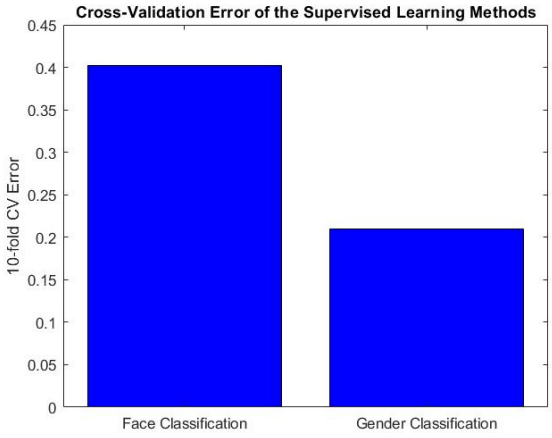

Figure 9: 10-fold Cross-Validation Error for the two SVM classification models

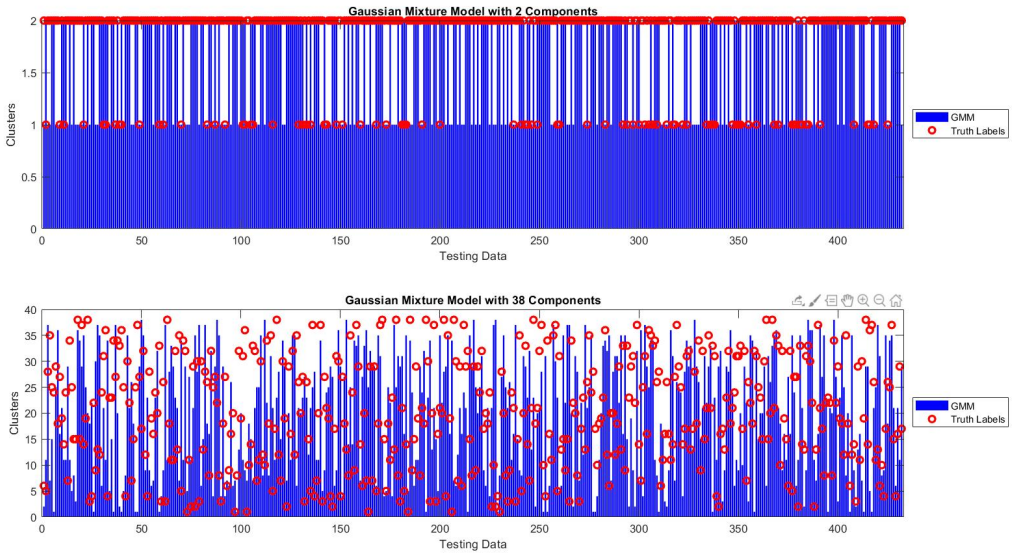

Figure 10: Gaussian Mixture Model classifications with 2 & 38 clusters

gorithm, I took labels corresponding to gender identification (2 clusters) and individual face identification (38 clusters) to compare with the results. This comparison can be seen in figure 10. However, we need to note that the image data may have other 2 & 38 cluster patterns, so I don't expect the accuracy with gender & individual face identifications to be as high as they were for the first two tests.

The accuracy's of the GMM clustering algorithm compared to gender & face identification is shown in figure 11. The accuracy for the 2 cluster algorithm compared to gender identification labels is 53.70%, while the accuracy for the 38 cluster algorithm compared to individual face identification labels is 1.389%. For a 2 cluster algorithm, getting above a coin flip is a decent predictor especially considering that the patterns in the data could be split into 2 clusters in ways other than gender identification. The accuracy for the 38 cluster algorithm is very bad, even lower than a 1 in 38 probability. This tells us that the cropped images likely do not naturally group into 38 clusters, which also explains why it had a lower accuracy when we attempted to identify 38 different individuals with the supervised learning algorithm.

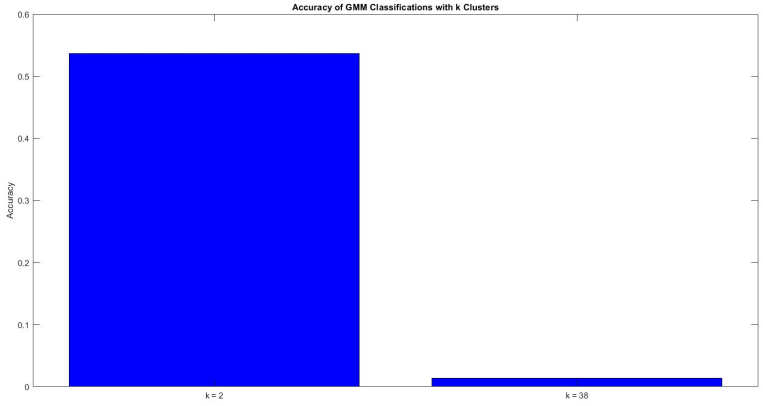

Figure 11: Accuracy's of the GMM classifications

5 Summary and Conclusions

SVD analysis is clearly very powerful in being able to accurately reconstruct images. Even when the images display different facial structures with faces in different positions, and with different backgrounds & lighting, we are able to get very solid reconstructions by taking higher ranks of the face space. In order to get a good low rank reconstruction, however, it would be better to crop the images.

For supervised learning classification algorithms, we can get a classifier with good accuracy on the cropped images, although we would do much better if we had a larger training set. For unsupervised learning, it is difficult to identify the correct choice of k clusters to search for, but we can find that if the data yields accuracy's higher than a 1 in k probability, then there is likely a natural pattern in those k clusters.

Appendix A MATLAB Functions

The following is a list of the important MATLAB functions that were used:

- `listing = dir(name)` returns attributes about `name`.
- `cd(newFolder)` changes the current folder to `newFolder`.
- `A = imread(filename)` reads the image from the file specified by `filename`.
- `Y = double(X)` converts the values in `X` to double precision.
- `[U,S,V] = svd(A,'econ')` produces an economy-size singular value decomposition of matrix `A`, such that `A = U*S*V'`.
- `imshow(I)` displays the image `I` in a figure.
- `Y = uint8(X)` converts the values in `X` to type `uint8`.
- `p = randperm(n)` returns a row vector containing a random permutation of the integers from 1 to `n` without repeating elements.
- `Mdl = fitcecoc(X,Y)` returns a trained ECOC model using the predictors `X` and the class labels `Y`.
- `label = predict(Mdl,X)` returns a vector of predicted class labels for the predictor data in the table or matrix `X`, based on the trained discriminant analysis classification model `Mdl`.
- `Mdl = fitcsvm(X,Y)` returns an SVM classifier trained using the predictors in the matrix `X` and the class labels in vector `Y` for one-class or two-class classification.
- `CVMdl = crossval(Mdl)` returns a cross-validated (partitioned) multiclass error-correcting output codes (ECOC) model (`CVMdl`) from a trained ECOC model (`Mdl`). By default, `crossval` uses 10-fold cross-validation on the training data to create `CVMdl`, a `ClassificationPartitionedECOC` model.
- `loss = kfoldLoss(CVMdl)` returns the classification loss obtained by the cross-validated ECOC model (`ClassificationPartitionedECOC`) `CVMdl`.
- `GMModel = fitgmdist(X,k,Name,Value)` returns a Gaussian mixture distribution model (`GMModel`) with `k` components fitted to data (`X`) with additional options specified by one or more `Name,Value` pair arguments.
- `idx = cluster(gm,X)` partitions the data in `X` into `k` clusters determined by the `k` Gaussian mixture components in `gm`.

Appendix B MATLAB Code

The MATLAB code used to generate the results of this write-up (`Extended_Yale_Faces.m`) can be found at the following GitHub repository: `https://github.com/ronwilson016/Data-Science-Projects/tree/master/Machine%20Learning/Extended%20Yale%20Faces/MATLAB%20Code`