FACE RECOGNITION WITH PCA, LDA, KNN AND BAYES

Kevin Chen
University of Maryland, College Park

ABSTRACT

We apply face-recognition techniques on the Illumination dataset, which consists of cropped 40x48 pixel images of 68 subjects under 21 different illuminations. We first try dimensionality reduction techniques on the dataset using principal component analysis (PCA) and linear discriminant analysis (LDA). We visualize and explore the spectrum of uncovered eigenfaces. We then implement k-nearest neighbors (KNN) and bayes classifiers on the Illumination dataset and explore the effect of varying parameters and compositions of different techniques on overall facial recognition performance, including: varying the number of eigenvectors (limiting eigenvectors, removing top eigenvectors) found in PCA, comparing PCA, LDA and no dimensionality reduction in KNN and bayes classification (in terms of time and classification performance).

METHODS

Data Preprocessing

Images to Disk

The illumination dataset consists of cropped 40x48 pixel images of 68 subjects under 21 different illuminations. Using scipy, a python library, we import the illumination.mat file, encoded in a MATLAB data format, into memory. We then recover PNG images from these 1920x1 vectors (40x48) and export these to disk, with the naming scheme illumination/person00/person00_illum00.png. This is done to ensure ease of immediate visualization, checkpointing of important data, as well as consistency with modern computer vision dataset organization practices. We continue to persist images to disk in this manner after downstream train-test dataset partitioning and image processing.

Train/Test Split

We split our image dataset into two partitions: train and test, housed in different folders to ensure that our test set is untainted by downstream processing. We ensure that the training and test sets are sampled from the same distribution by taking an equal proportion of samples from each class, or person. The samples from each class are randomized, with a fixed random seed (999) assigned at the beginning of the process (Figure 1). With an 80-20 train-test partition, we have 1088 training samples and 340 test samples. With this partioning, the two datasets show similar distributions of average pixel value (Figure 2).

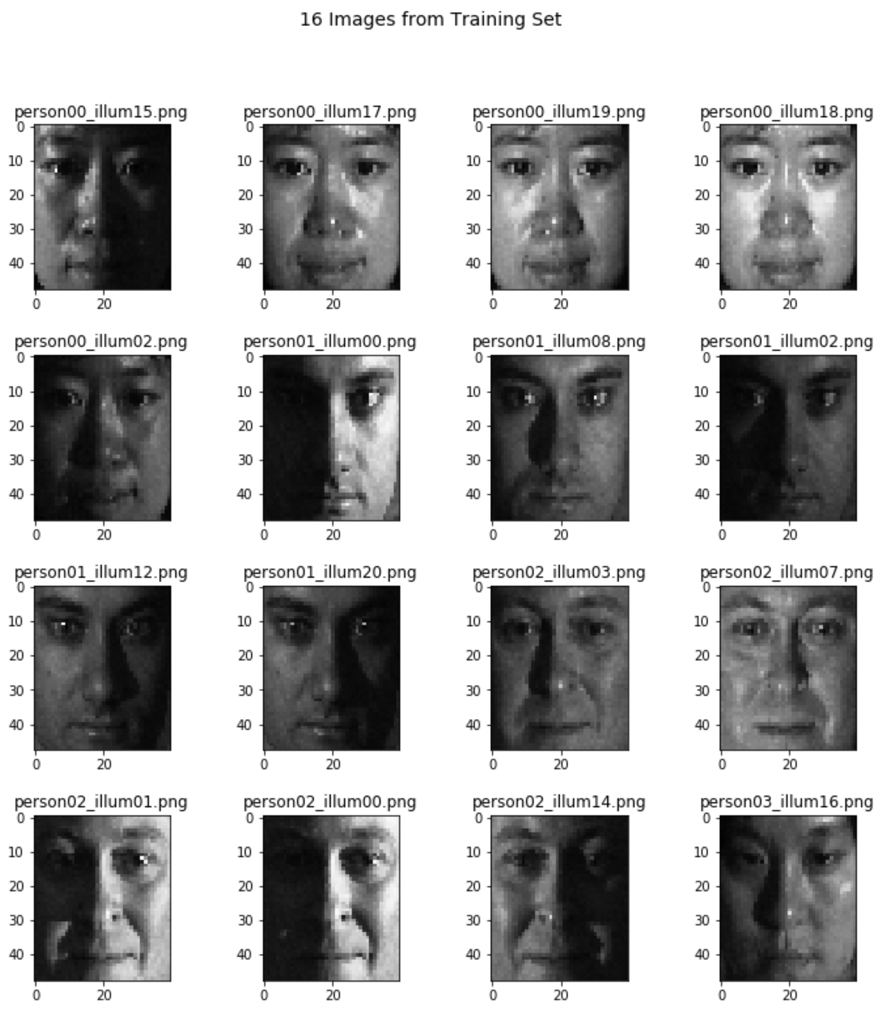

Figure 1. The first 16 images from our randomized training set on Illumination.

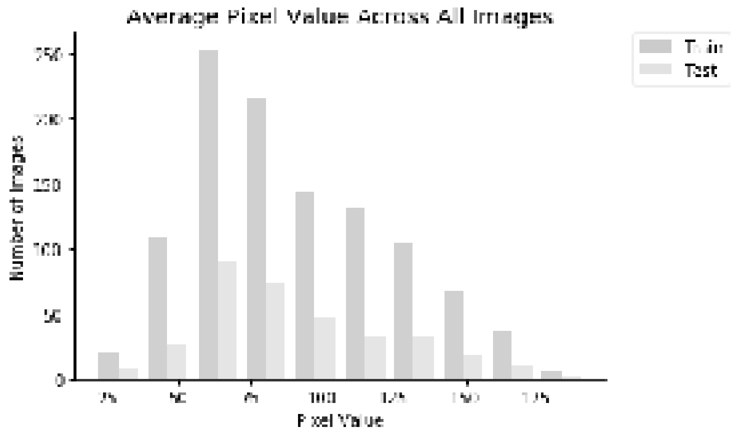

Figure 2. Histogram of average pixel values of each image in the training and test sets of Illumination.

Dimensionality Reduction

All code has been provided in the .zip file in the notebooks/ and src/ directories.

Principal Component Analysis (PCA)
The primary use of PCA in this project is for the purpose of dimensionality reduction. We treat each image as a flattened 1 dimensional vector and find the covariance matrix on the matrix of images, where each image is a column vector. We then find the top k eigenvectors of this covariance matrix, in order of the magnitude of their eigenvalues. These eigenvectors represent the eigenfaces of our training dataset. We can then project images onto these eigenfaces, approximately treating all faces as a linear combination of these eigenvectors. This gives us a dimensionality reduction, provided we limit the number of eigenvectors we choose to use to represent each image.

Preprocessing (Mean Centering)
In order to properly apply PCA, we must first ensure our image vectors are mean centered. We sum up all image vectors in the training dataset and divide by the magnitude of the training dataset to get the average image. We subtract each image in the training set by this average. This ensures that our training set is centered at mean 0. We also subtract each image in the test dataset by this same mean image, to ensure we are applying the same transformations on the test dataset that we are on the train dataset. As a sanity check, we visualize the average images from both the training and test sets and find that they are nearly identical (Figure 3).

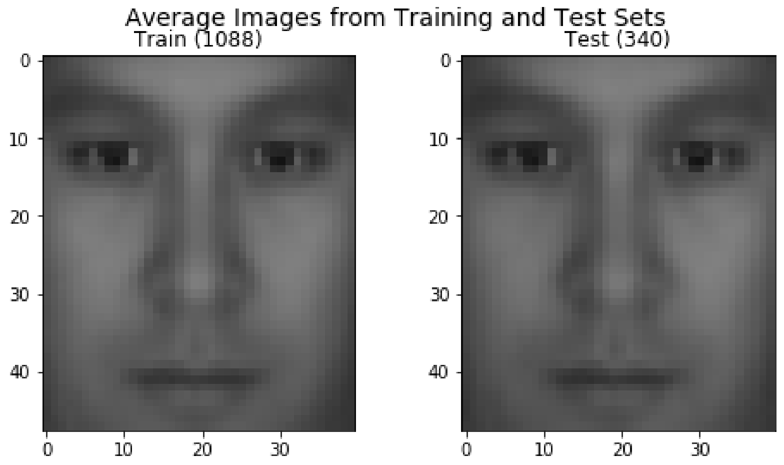

Figure 3. Recovered images from the averaged vectors of the training and test sets of Illumination.

Linear Discriminant Analysis (LDA)
In the same vein as PCA, we use LDA as an alternative means for dimensionality reduction. We compute the within-class and between-class scatter matrices and then solve the generalized eigenvalue problem for inv(Sw)*Sb. These new eigenvectors give us the axes of our new feature space. These eigenvectors represent the fisherfaces of our training dataset, and images can be similarly projected onto these fisherfaces, as they can with eigenfaces in PCA.

Dimensionality Reduction Through Facial Projection
For any given image vector, we can reduce its dimension by projecting it onto a subset of the eigenface/fisherface vectors. We do this by taking a MxN matrix of N eigenfaces/fisherfaces, where each

eigenface/fisherface is a column, and taking the product of the transpose of that matrix (NxM) with the aforementioned image vector (Mx1) to form a dimensionally reduced Nx1 vector. Each coordinate of this transformed vector represents the distance from the image to the eigenface/fisherface.

Classifiers

K-Nearest Neighbors

We establish euclidean distance as a similarity measure between any two image vectors. Then to classify any test image, we take the euclidean distance of that image with every test image in the training dataset. The k images in the training dataset with the least distance are called the k-nearest neighbors of the test image. The labels of the k-nearest neighbors determines the label of the test image, by majority vote (other tie-breaking methods can be used). This method is computationally intensive at prediction time (linear in terms of the number of samples in the training set), but constant in the learning phase.

Bayes

Bayes theorem tells us that the probability of event A conditioned on event B is equivalent to the probability of B conditioned on A multiplied by the probability of A, divided by the probability of B. In other words, the posterior probability = (likelihood * prior) / evidence. That means we can calculate the posterior probability of an event conditioned on multiple events with this formula if we naively assume that the events are all independent. In the context of face recognition, we can treat each pixel in our image vector as a random variable, and interpret our problem as finding the class with the highest posterior given the pixels in the test image.

First, in the our case of our dataset, the priors of each class (class frequency / the total number of samples) will be the same, so the priors will just end up being some constant multiplier for all classes. (Note, this is not the case for datasets without uniform class distributions). Thus, we can skip calculation of the priors. The evidence will also remain constant across all classes, so we can skip its calculation as well.

So all that remains is to compute the likelihood of each class. Using the naive assumption stated above, we can compute the likelihood, or the probability of an image given a person, as the product of the probabilities of individual pixels given that person. If we use the gaussian assumption i.e assume the variables (pixel values) are normally distributed, we can use maximum likelihood estimation to estimate the likelihoods for each pixel, and thus the likelihood for the entire image.

RESULTS

Principal Component Analysis

Eigenfaces

We generate the eigenvectors of the covariance matrix of all images in the training set and sort them by descending magnitude. The top 10 rendered eigenvectors clearly resemble faces. The next few rows of 10 also vaguely have facial characteristics, but by row 6 and beyond (i.e the 60th eigenvector), the facial characteristics have severely degraded (Figure 4).

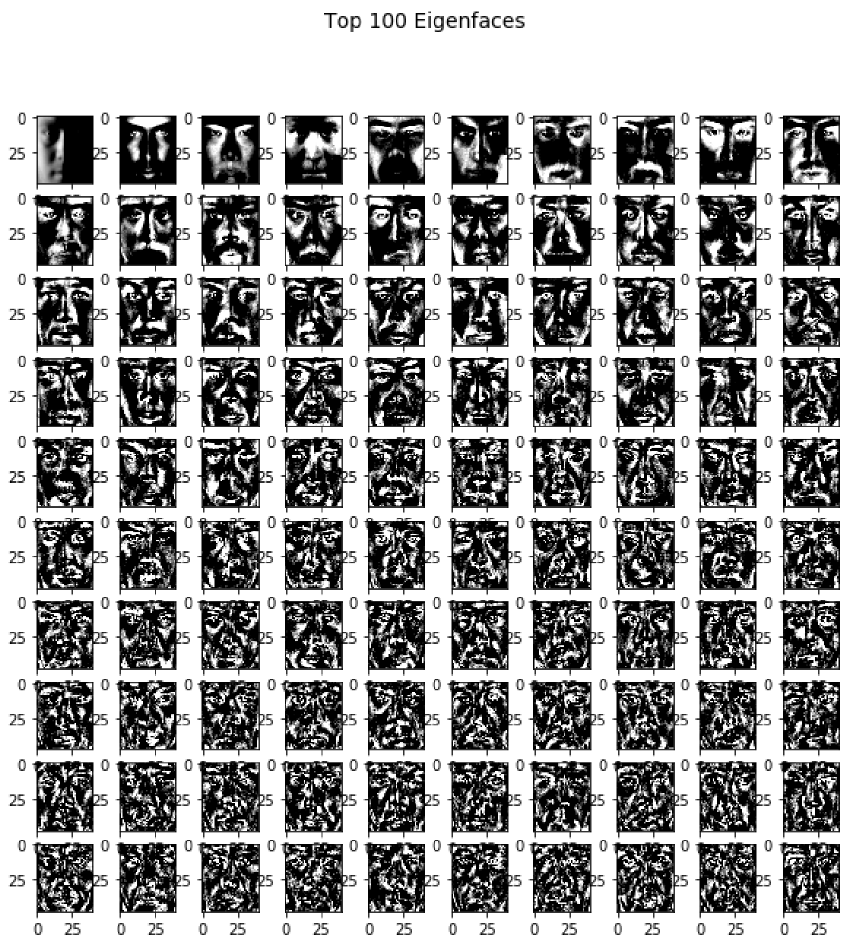

Figure 4. 100 eigenfaces recovered from the top eigenvectors of the covariance matrix of the training set.

Since the top eigenvectors have the most powerful facial representation (and are responsible for the most variance in the new feature space), we can use a subset of the top eigenvectors to model the majority of the facial characteristics of a given person (Figure 4). However, we also note that the very top few eigenvectors (i.e 1-3) may overfit to the training set, and which can hurt generalization performance on the test set.

Computing Covariance and Eigenvalues/Eigenvectors

For a training set of size 1088 composed of 1920x1 vectors, computing the covariance matrix took a wall time of 1.08 s ± 19.6 ms per loop (mean ± std. dev. of 7 runs, 1 loop each), and computing all eigenvectors of the covariance matrix took a wall time of 1.1s.

Linear Discriminant Analysis

Fisherfaces

We generate the eigenvectors from the LDA problem. Some of the top eigenvectors appear to have facial characteristics, but not to the same qualitative degree as the eigenfaces in Figure 4 (Figure 5).

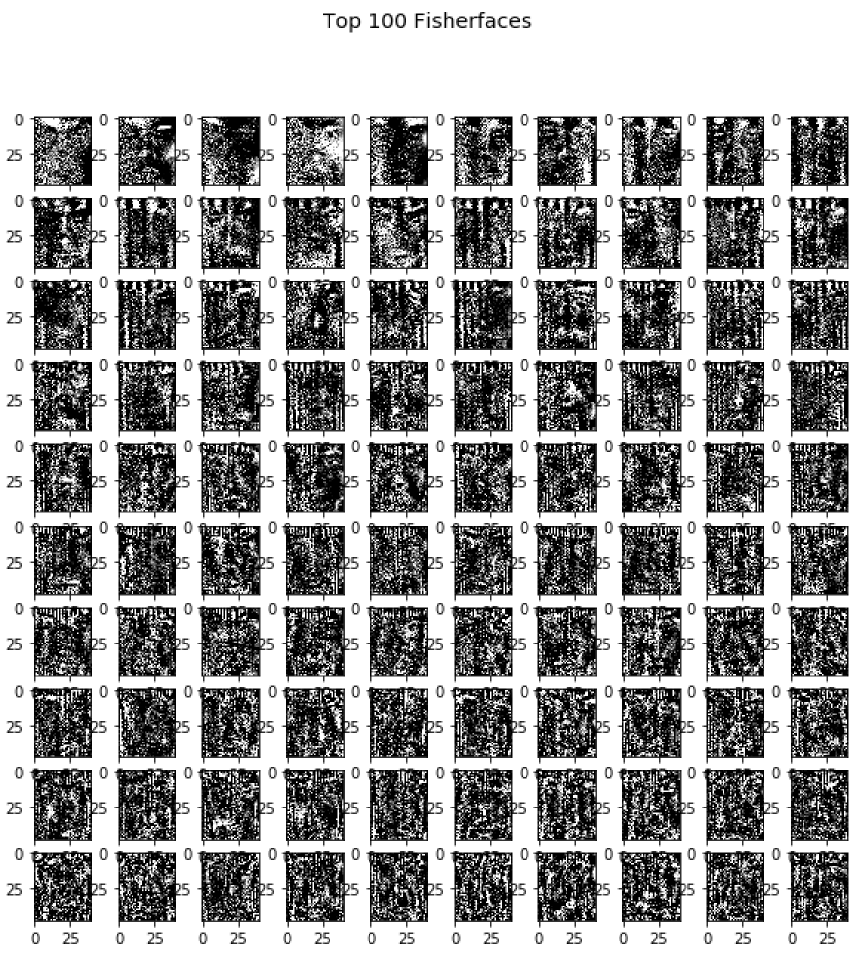

Figure 5. 100 fisherfaces recovered from the top eigenvectors of the LDA problem

Computing Eigenvalues/Eigenvectors

For a training set of size 1088 composed of 1920x1 vectors, solving the LDA eigenvalue problem took a wall time of 2.35 s ± 106 ms per loop (mean ± std. dev. of 7 runs, 1 loop each).

K-Nearest Neighbors Classifier

Prediction is Significantly Faster After Dimensionality Reduction

For a training set of size 1088 composed of 1920x1 vectors, computing the class of one test image using 1-NN classification took a wall time of 1.11 s ± 6.36 ms per loop (mean ± std. dev. of 7 runs, 1 loop each). Computing the accuracy of 1-NN classification using the same training set on a test set of 340 images took a wall time of 6min 11s.

If we reduce the dimension of each vector from 1920 to 10 using PCA, computing the class of a single test image takes a wall time of 10.7 ms ± 232 µs per loop (mean ± std. dev. of 7 runs, 100 loops each). This is approximately a 100x increase in efficiency in comparison to the raw image prediction used previously.

Baseline Classification Accuracy of 1-Neighbor Raw Image Prediction
We find an KNN classification accuracy of 92.6% on the 80-20 train/test split with 1-Neighbor prediction. Because of the significant wall time, we proceed with dimensionally reduced KNN classification using PCA or LDA for all further analyses.

LDA Performs Worse Than PCA and Increasing Eigenvectors Helps Generalization
As we increase the number of eigenvectors that we project our training and test images onto, our classification accuracy increases (Figure 6). This is consistent with our intuition- the more information we preserve during dimensionality reduction, the more informative our analyses.

Under PCA, when we project to 30 eigenfaces, our KNN classifier get a classification accuracy of 85.9%. Increases in accuracy for PCA/LDA taper off around this point. Nonrigorously, it appears that projecting onto the top 30 eigenvectors is a good middle-ground between the preservation of accuracy and the achievement of a reasonable runtime.

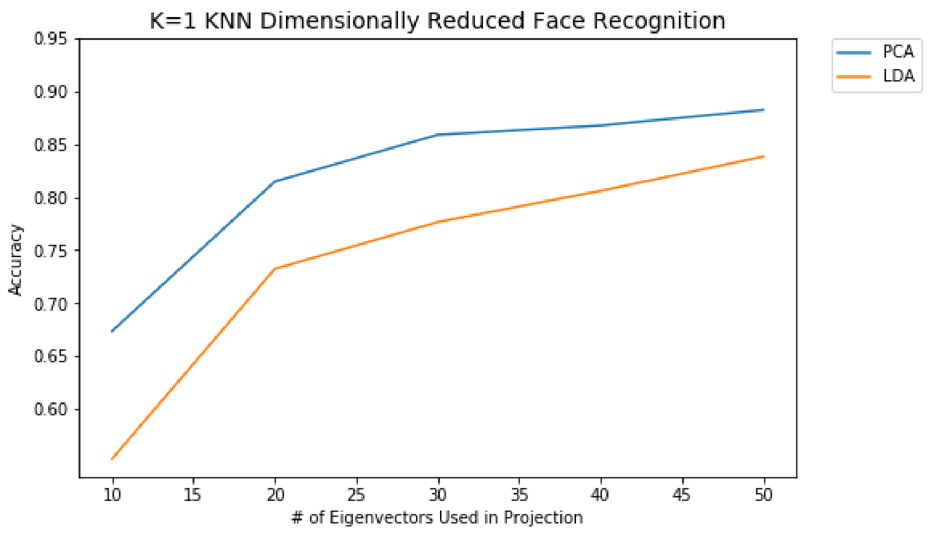

Figure 6. Plot of test accuracies with 1-NN classification using top K eigenvector projections for PCA and LDA. The number of eigenfaces/fisherfaces used are incremented by intervals of 10 (10..50).

We also see that PCA outperforms LDA in terms of test accuracy performance at every point (Figure 6).

Deleting Top Eigenvectors Increases Generalization Performance
Interestingly, once we fix the top 30 eigenvectors as our basis to project onto, if we begin systematically deleting the top 5 eigenvectors from this basis, we gain a remarkable increase in test accuracy with the PCA projection, from 85.9% with 0 vectors deleted to 99.4% with 5 vectors deleted (Figure 7). If we continue to delete these eigenvectors, our accuracy hovers at a high plateau at around 90% all the way up to the top 20 eigenvectors being deleted, before quickly dropping off into below 50% accuracy and below as 25 eigenvectors are deleted and only 5 remain (Figure 8). The LDA projection follows a similar pattern but with significantly worse accuracy than PCA at every point (Figure 7, 8).

This suggests that the top few eigenfaces (and perhaps fisherfaces) overfit to the training set and cause a drastic decrease in generalization performance.

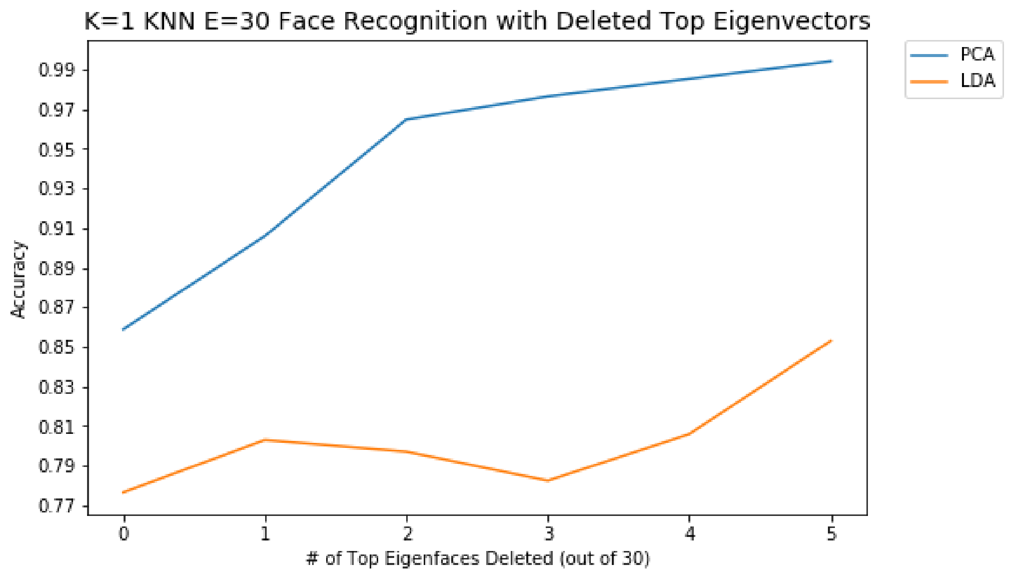

Figure 7. Plot of test accuracies with 1-NN classification using top 30 eigenvector projections. The top 5 eigenvectors are incrementally deleted and the datasets are reprojected and classified.

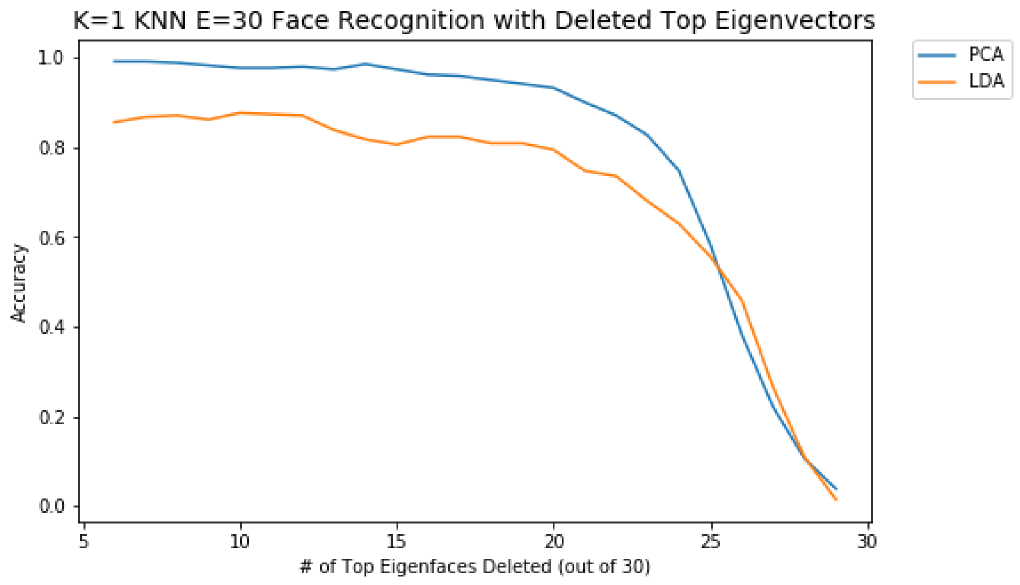

Figure 8. Plot of test accuracies with 1-NN classification using top 30 eigenvector projections. The top 6-29 eigenvectors are incrementally deleted and the datasets are reprojected and classified.

Increasing Number of Neighbors Lowers Accuracy

Once we fix the eigenvectors to the top 30 (minus the top 5), which gave us good performance above with PCA KNN, we find that increasing the number of neighbors consistently decreases our test accuracy, from 99.4% with 1 neighbor classification down to 92.6% with 5 neighbor classification (Figure 9).

The result is even more drastic with LDA, where the test accuracy drops from above 80% to below 40% with 5 neighbor classification (Figure 9).

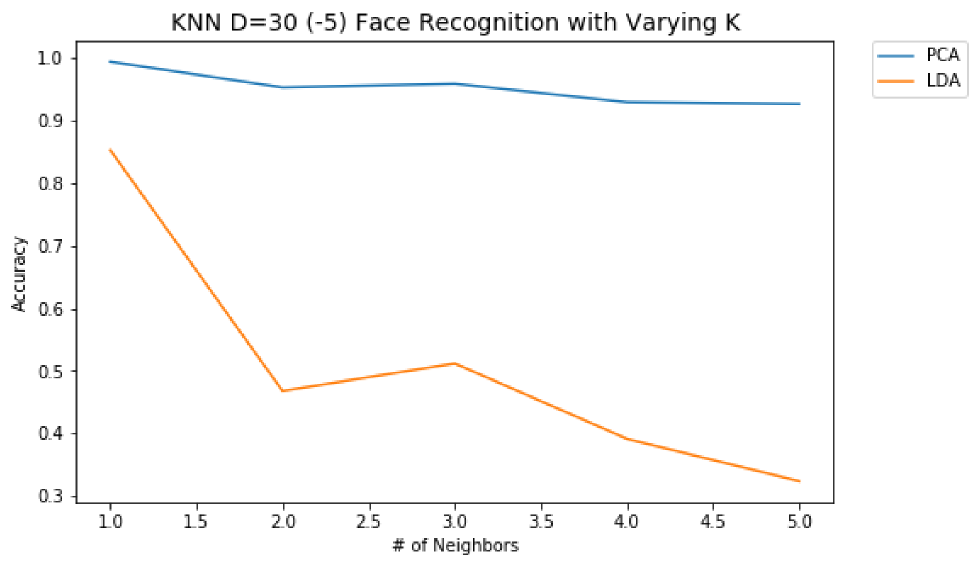

Figure 9. Plot of test accuracies with K-NN classification using top 30 (-5) eigenvector projections. The number of neighbors, K, is increased from 1 to 5, in intervals of 1.

Bayes Classifier

Prediction is Significantly Faster After Dimensionality Reduction, Moreso Than KNN Prediction
For a training set of size 1088 composed of 1920x1 vectors, computing the class of one test image using Bayes classification took a wall time of 8.61 s ± 130 ms per loop (mean ± std. dev. of 7 runs, 1 loop each). Computing the accuracy of Bayes classification using the same training set on our full test set of 340 images was not precisely computed because of the lengthy estimated duration: ~8.6s * 340 images = ~48 minutes.

If we reduce the dimension of each vector from 1920 to 10 using PCA, computing the class of a single test image takes a wall time of 4.83 ms ± 139 µs per loop (mean ± std. dev. of 7 runs, 100 loops each). This is approximately a 160x increase in efficiency in comparison to the raw image prediction used previously.

Interestingly, the inference time for Bayes classification is significantly slower than KNN classification (means: 8.6s vs 1.1s) when operating on the raw 1920x1 image vectors, but is signficantly faster when operating on dimensionally reduced image vectors (means: 4.8ms vs 10.7ms) (section: KNN). It appears that in terms of runtime, dimensionality reduction may be more significant to Bayes classifiers than KNN classifiers at inference time.

LDA Performs Significantly Worse Than PCA and Increasing Eigenvectors Helps Generalization
As we increase the number of eigenvectors that we project our training and test images onto, our Bayes classification accuracy increases (Figure 11). This is consistent with our analysis from KNN- the more information we preserve during dimensionality reduction, the more informative our analyses.

Increases in accuracy for PCA begin tapering after 10 eigenvectors, whereas increases for LDA seem to be relatively constant for the top 50 eigenvectors (Figure 11). With PCA, using the top 50 eigenvectors, we can achieve a classification accuracy of 93.2%.

To compare with KNN, our PCA-KNN classification accuracy was 85.9% at 30 eigenfaces, but our PCA-Bayes classification accuracy was 89.7% at 30 eigenfaces (Section: PCA, Figure 11).

And similar to KNN, we see that with Bayes classification, we achieve a far better generalization performance with PCA dimensionality reduction than we do with LDA (Figure 11).

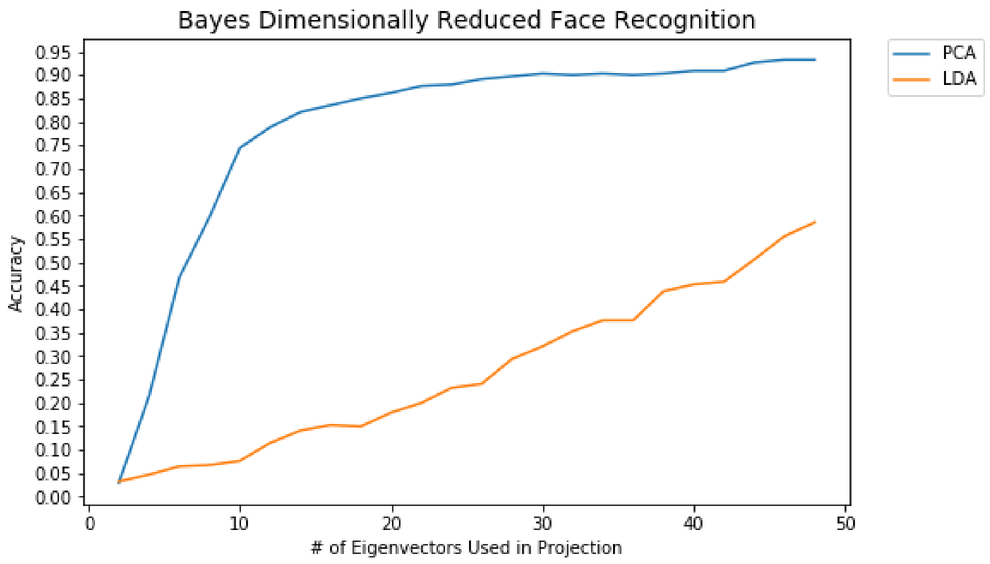

Figure 11. Plot of test accuracies with Bayes classification using top 50 eigenvector projections. The number of eigenvectors are increased iteratively by 2 (2..50).

Deleting Top Eigenvectors Decreases Generalization Performance
When we fix the eigenvectors received from PCA/LDA at 50 and iteratively delete the top eigenvectors from this set, we see no marked increase in Bayes classification accuracy (Figure 12). Instead, we see the PCA Bayes classification accuracy slowly decrease from 93.2% accuracy at 0 eigenvectors deleted down to 87.9% accuracy at 30 top eigenvectors deleted, and then sharply decrease to 75.6% accuracy at 40 top eigenvectors deleted (Figure 12). A similar trend occurs with LDA Bayes classification accuracy, albeit with a seeminly slight but still unremarkable rise in accuracy when the top 10-20 eigenvectors are deleted.

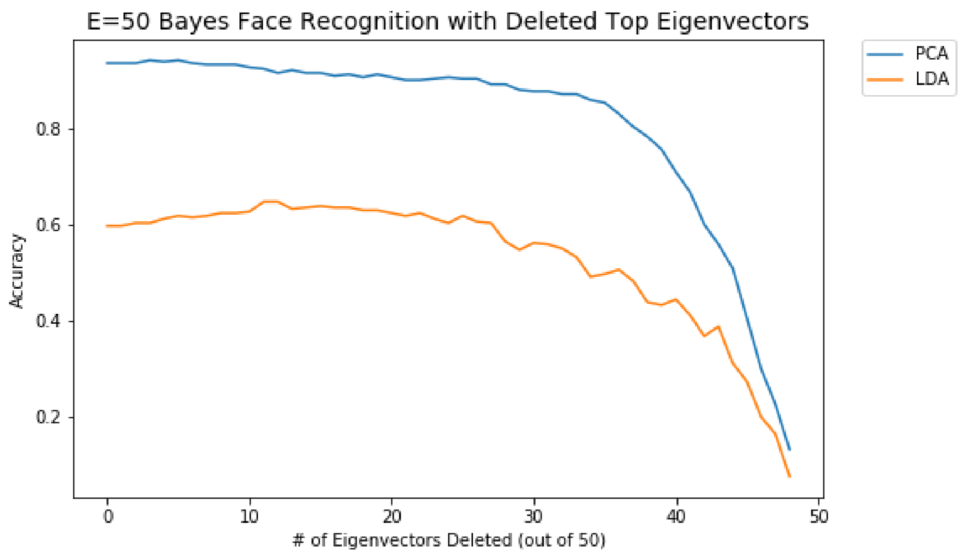

Figure 12. Plot of test accuracies with Bayes classification using top 50 eigenvector projections. The number of top eigenvectors that are deleted increase iteratively by 1.

CONCLUSION

These concluding remarks have been kept brief, in accordance with the project guidelines.

On the Illumination dataset with an 80-20 train-test split, our experiments seem to indicate that classifiers perform better (attaining greater test classification accuracy) when they use PCA dimensionally reduced representations of the input space than when they use LDA representations.

Dimensionality reduction seems to be a necessity to perform inference in a reasonable amount of time, especially with the Bayes classifier. We also find that after PCA dimensionality reduction on the input space, we can preserve most of the accuracy of using raw images with KNN, and even exceed its accuracy through manipulation of the subset of eigenvectors chosen.

We also find that with KNN, removing the top few eigenvectors increases accuracy (possible by reducing overfitting on the training set). This does not happen with Bayes. With KNN we also find that increasing the number of neighbors only decreases classification accuracy.

We achieve the best performance in our experiments using a K=1 KNN classifier with the top 30 PCA eigenvectors - top 5 eigenvectors = next top 25 eigenvectors. This gives us a 99.4% classification accuracy. This could be improved upon in a series of experiments whose purpose would be to extract the greatest classification accuracy possible.