Face Recognition

Jesika Haria

6.S064 Introduction to Machine Learning

Project 2

April 17, 2013

1 PCA and Image Reconstruction

I first plotted a few random images to familiarize myself with the project, and then plotted the mean image of all the faces, as is seen in Figure 1:

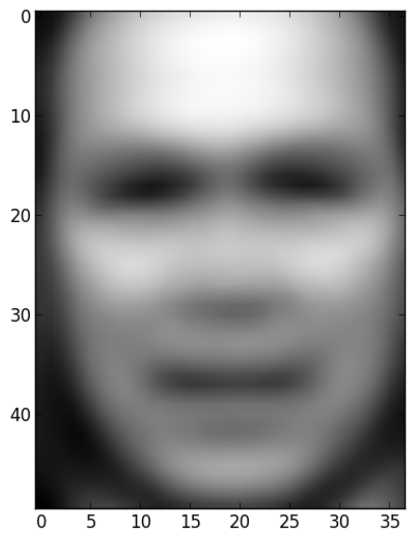


Figure 1: Mean of Faces

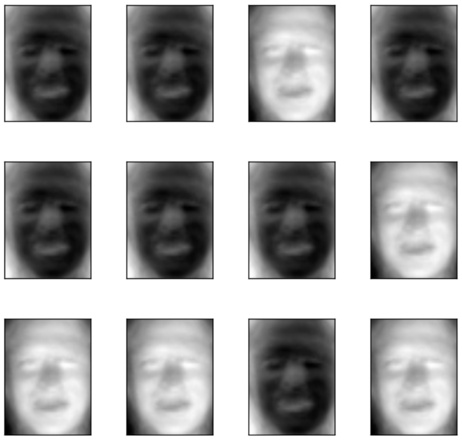
Then, I performed Principal Component Analysis on the data matrix for values of $l = 1, 10, 50, 100, 500$, and 1288, as can be seen in Figure 2.

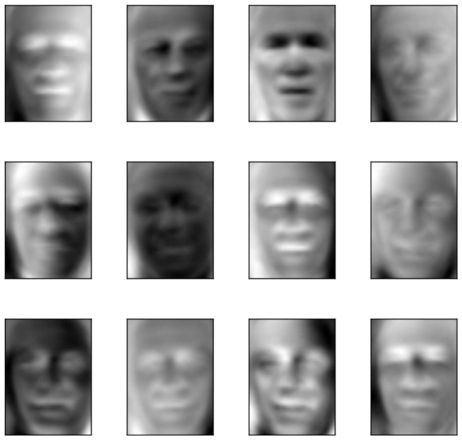
As expected, the sharpness and thus distinguishability of the images increases with increasing l, due to increasing amount of information. To that effect, there are three main bands of recognition:

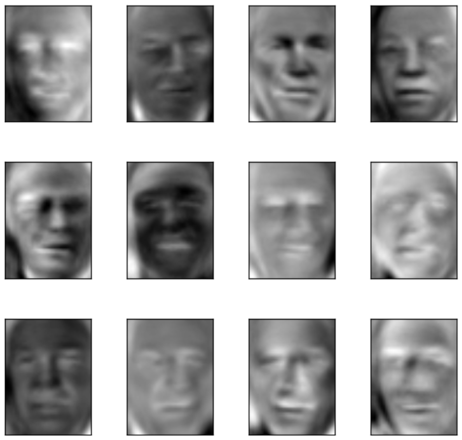
1. $1 \leq l \leq 10$: Face is discernible as a human, but individual faces indistinguishable.

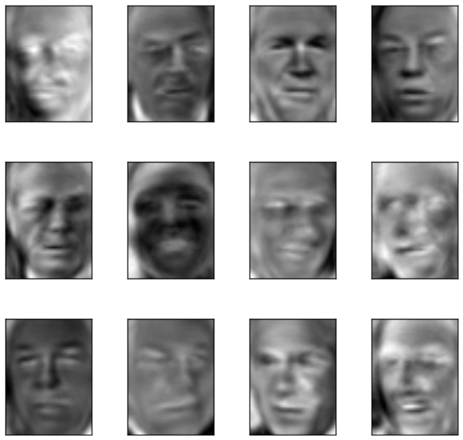
2. $10 \leq l \leq 100$: Differences between faces observed, but hard to differentiate. Classification is error prone.

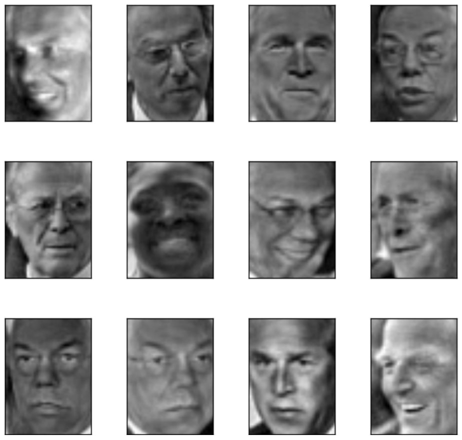
3. $100 \leq l \leq 1288$: Faces can be easily told apart, and increasing l only contributes to a slight increase in resolution.


(a) l = 1


(b) l = 10


(c) l = 50


(d) l = 100


(e) l = 500

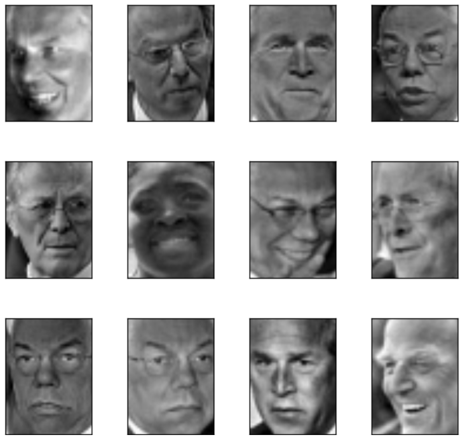

(f) l = 1288

Figure 2: PCA with different l values

2 Classification

In this section, I explored classification through Support Vector Machines, first by analyzing the impact of the parameter C on the classification score.

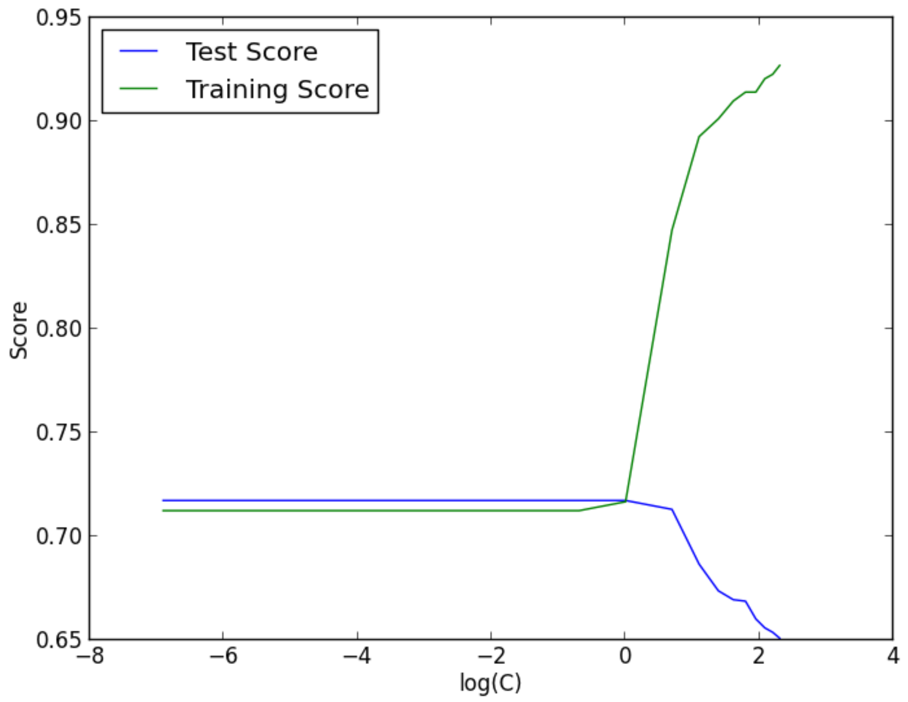


Figure 3: Training and Test Error with varying values of $log(C)$

From the negative side of the $log(C)$ axis in Figure 3, it seems that both the Test Score and Training Score are at par with each other between 0.70 and 0.75, with the Test Score being higher by 0.01. This changes at $log(C) = 0$. The value of the Training Score then suddenly shoots up to 0.925, while the Test Score drops to 0.65, for values of around $log(C) = 2$.
To understand this phenomenon, we can think of adjusting the parameter C as controlling the bias-variance tradeoff. As we increase the value of C, we reduce the value of λ - this results in lower bias and higher variance classifiers. Thinking of C as a penalty for margin less than 1, a large C means that there is a large penalty for non separable points, so we favor satisfying the

classification constraints more. As a result, we store many support vectors, and hence overfit. Naturally, we perform very well on the training data, but we have poor generalization on the test data. This explains the Training Score tending to 1.00 while the Test Score drops to 0.65.

We now experiment with varying values of the number of components, l. As seen in Figure 4, the Test Score and the Training Score in the case of

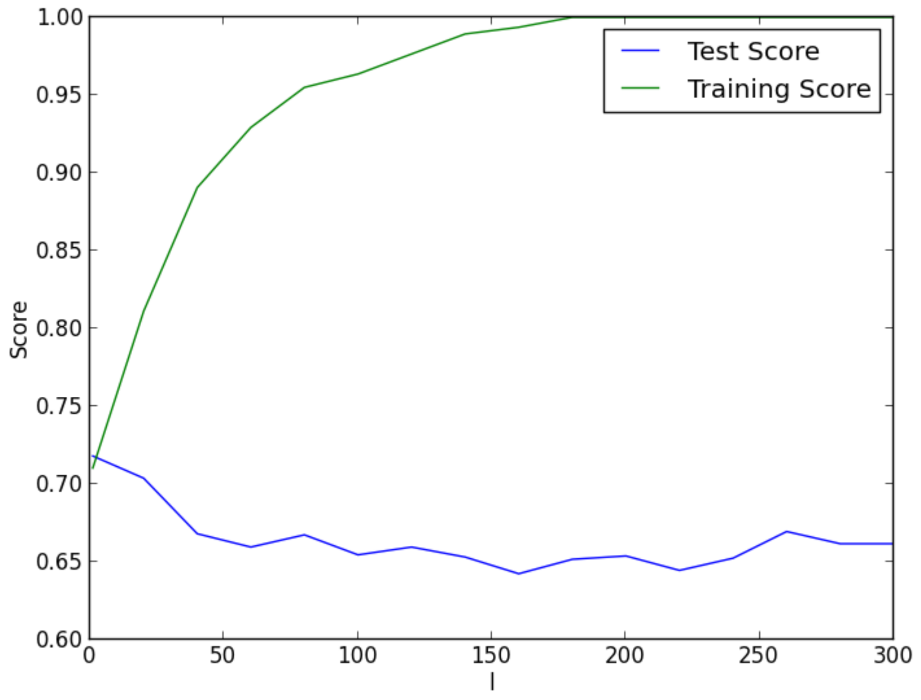


Figure 4: Training and Test Error with varying values of l

increasing l start off with similar values at around 0.72. However, within $l < 10$, the Test Score plummeted to around 0.65, while the training score increased to 1.00. The additional components derived from larger l over fit the data. In the extreme case, if we classify each picture by its pixel, then we will always have a perfect clustering in Training Data, but the classification will not generalize well. This is similar to the effect of adding components beyond the optimal.

3 Clustering

In this section, we apply a clustering algorithm to the data. In order to find the optimal clustering algorithm, we need to first initialize the clusters appropriately. We compare the different initialization methods, `kMeansInit` and `cheatInit`, for the `kMedoids` clustering algorithm.

Trial #	Score
1	0.5
2	0.8125
3	0.5
4	0.5
5	0.5
6	0.5
7	0.5
8	0.8125
9	0.5
10	0.8125

(a) `kMeansInit` Score

Trial #	Score
1	0.8125

(b) `cheatInit` Score

Table 1: Trial Score Comparison for `kMeansInit` and `cheatInit`

For `cheatInit`, the clusters are initialized at distances that are already at the well-separated optimum according to the distance metric, and the clustering algorithm is less likely to get confused, or get stuck in local extrema. For `kMeansInit`, because the initial points are chosen randomly, and the other points are chosen such that they are furthest from the points already chosen. If a bad first point is picked it is possible to get stuck in local maxima, which can be suboptimal.

Now, we study the effect of number of dimensions on the quality of the clustering, i.e. the clustering score.

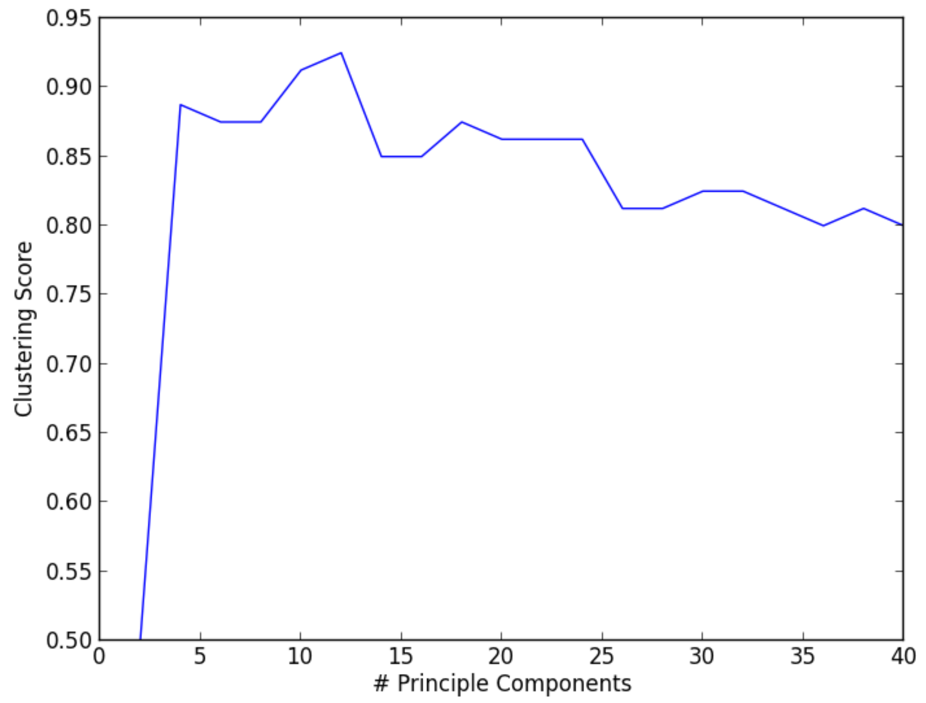


Figure 5: Effect of varying dimensions on clustering score

In Figure 5, increasing the number of dimensions helps to rapidly improve the quality of the clustering until around $l = 12$, where the highest clustering score is around 0.925. For number of dimensions that are higher than this value, the clustering score gradually reduces to around 0.825 for $l = 40$. The sharp increase in the accuracy of the clustering score while adding the first few principal components comes from the fact that we are using the most prominent distinguishable features first, i.e. grabbing the low-hanging fruit. After a certain point though, it seems as if assign components not only does not increase the accuracy, but even decreases it. The clustering algorithm performs worse under larger components (sharper and higher frequency parts of the image) because they might introduce noise, and other irrelevant information that detracts from the intrinsic feature set of the image.

For the experiment to find out which pairs of people were more similar to one another than others, I ran the `kMedoids` clustering algorithm on both the full data, and on data with $l = 12$ components (which was found to have the highest clustering score in Figure 5). Both were run on `l2Sq` as the distance metric, and on a mean of 40 images from each class. Each pair of classes was considered sequentially, and here I report the best and the worst over all the pairs of classes. Figure 6 illustrates a random image drawn from each class for reference.

It was observed that for the complete data (non PCA) case, Classes 9 and 16 were most distinguishable, at a score of 0.925; while Classes 1 and 8, 7 and 10 proved to be the hardest to distinguish, at a score of 0.5875.
For the case with $l = 12$, Classes 4 and 16 were the easiest to distinguish at a score of 0.9875, while Classes 1 and 8 were the hardest to distinguish, with a score of 0.5.
This gives us a very interesting observation that when the the signals are very strong and the classes are easily differentiable, lower dimensions optimize the clustering score, because they are more robust. However, when it is harder to distinguish the features between classes, the lower dimensions have a lower score. This might be because in these situations when the first few principal components don't provide a very clear differentiation, higher dimensions actually prove useful in clustering. The vectors with PCA using $l = 12$ lack this higher dimension information, and so they perform worse.
Visually, the results of the algorithms also make sense. The classes where the algorithms have a lower score typically correspond to those classes whose subjects have a similar face structure in terms of nose length, eye distance, eye size, mouth shape, size and position. It is important to note that the angle of the photo, lighting, expression might not play as important of a role in clustering since in the experiment, the clustering algorithm was run over the mean face over 40 images of each class.

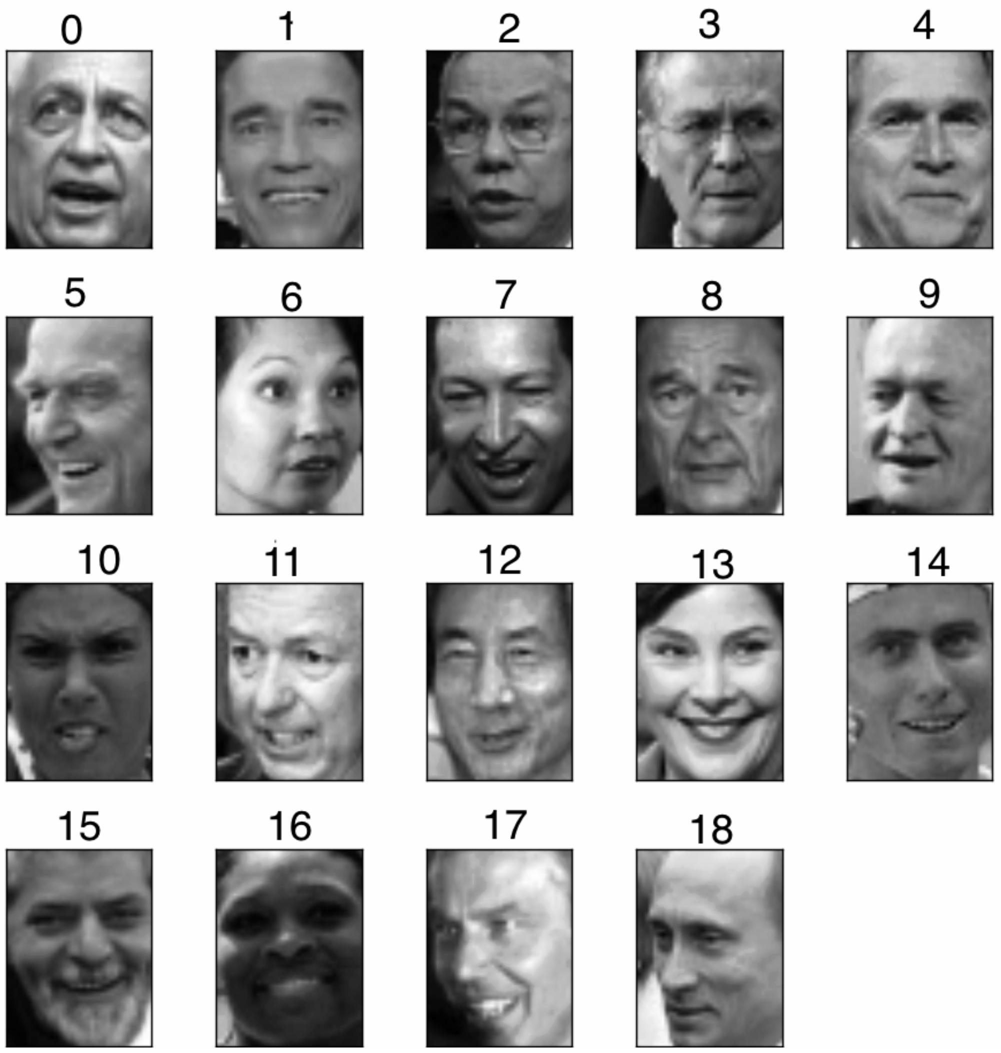


Figure 6: Sample image from person of each class