RecognizeMyFace
An Exploration of Classifiers for Facial Recognition

Durmus U. Karatay
Department of Physics
University of Washington
ukaratay@uw.edu

Matthias Smith
Department of Physics
University of Washington
mwsmith2@uw.edu

Abstract

In this study we explore the effects of using different classifers for recognition of different features after images are processed using Principal Component Analysis (PCA) to reduce dimensionality. We examine five different classifiers, namely, nearest-neighbor (NN), kNN, multivariate Gaussian Mixture Models (GMM), Linear Discriminant Analysis (LDA), and Support Vector Machines (SVM). After training our model, we selected the best performing classifiers on the validation set for each feature of the images. We report that different features require different classifiers.

1 Introduction

We use a well known approach for facial recognition which employs PCA for reducing the dimensionality of the problem. PCA treats the system as 2-dimensional and projects face images into a face-space that has the information of the variation among the training images. The basis of this face-space is referred to as the eigenfaces. We try to identify different sets of image features by calculating the projections of eigenfaces onto the images. We employ five different classifiers after doing a dimensionality reduction with PCA: nearest-neighbor, kNN, multivariate Gaussian Mixture Models, Linear Discriminant Analysis, and Support Vector Machines.

2 PCA

Given N images, the K ($\leq N$) most significant eigenfaces can approximate a face. All images are $W \times H$ matrices which are rearranged into $WH \times 1$ column vectors referred to as Γ_i. We calculate the average face as

$$\Psi = \frac{1}{N}\sum_{i=1}^{N}\Gamma_i.$$

We center each face by subtracting off Ψ, then we construct the image matrix, $\mathbf{A}$.

$$\Phi_i = \Gamma_i - \Psi.$$
$$\mathbf{A} = [\Phi_1, \Phi_2, ..., \Phi_N].$$

Since it is computationally expensive to calculate the eigenvectors of such a large matrix, we employ the method described in the original work of Turk et al [1].

$$\mathbf{C} = \mathbf{A}\mathbf{A}^T.$$

We obtain the eigenvectors of $\mathbf{C}$, u_i, by first calculating the eigenvectors of $\mathbf{A^TA}$, namely v_i.

$$u_i = \mathbf{A}v_i.$$

We need to select the most significant K eigenvectors by looking at the eigenvalues. The variance, σ can be used to determine K by selecting a threshold on the fractional variance, $F(\sigma, K)$.

$$\sigma = \sum_{i=1}^{N} \lambda_i,$$

$$F(\sigma, K) = \frac{\sum_{i=1}^{K} \lambda_i}{\sigma}.$$

The images in the training set are characterized by calculating the projection of the first K eigenvectors onto the image:

$$\omega_{ij} = u_j^T \Phi_i.$$

For recognition of an unknown image, we center the image by subtracting Ψ and calculate the weight for each eigenface. With the projected weights we can test against our training set weights to determine the feature.

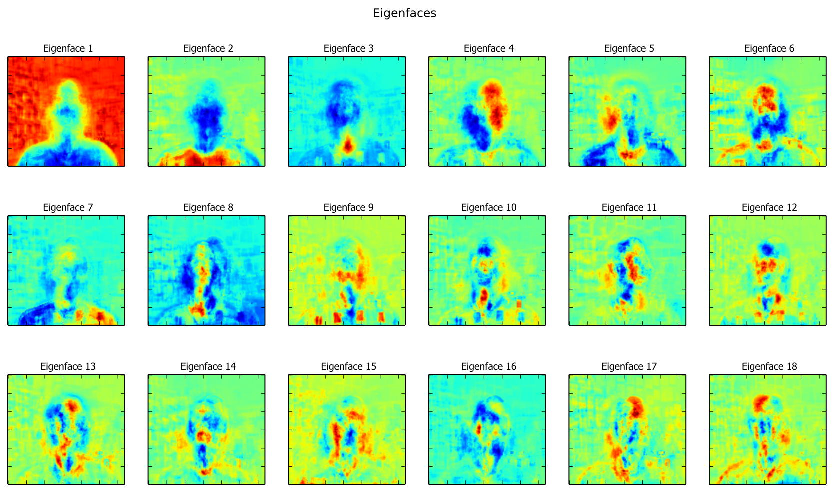

Figure 1: The most significant 18 eigenfaces.

3 Classifiers

3.1 Nearest Neighbors

We employ two different nearest neighbor techniques: Mahalanobis distance and distance weighted kNN. Mahalanobis distance is defined as [2]:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{x}')},$$

where $\mathbf{C}$ is the covariance matrix of $\mathbf{x}$ and $\mathbf{x}'$. Mahalanobis distance performs better than Euclidean distance when different dimensions of a vector need different weights, such as the eigenfaces. This is due to the facts that Mahalonobis distance accounts for the covariance of different dimensions and Euclidean distance suffers from the curse of dimensionality.

In our kNN implementation we choose $k = 3$ and use distance weighting. Our kNN uses standard Euclidean distance to find the nearest neighbors. Even though Euclidean distance suffers from the curse of dimensionality, weighting the neighbors by distance reduces the noise [2].

3.2 GMM & SVM

We use two clustering techniques, GMM and SVM. We use both in a semi-supervised fashion with a radial basis function kernel where the kernel is defined as [2, 3]:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right).$$

3.3 LDA

LDA can be used as a classifier with a linear decision boundary. The decision boundaries are created using class conditional densities from the data. In particular we used LDA with Gaussian densities. LDA has no parameters to tune, and it is inherently multi-class making it an appropriate choice for separating features with more than two classes [2, 3].

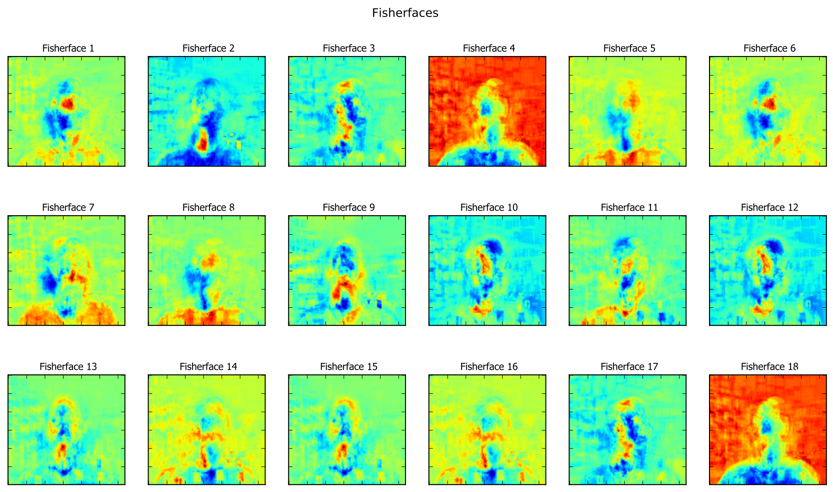

Figure 2: The first 18 Fisherfaces.

LDA, like PCA, performs supervised dimensionality reduction, and Fisherfaces are the subspace consisting of most discriminant bases. Unlike eigenfaces, Fisherfaces highlight the differences among classes [4].

4 Experiment

4.1 Setup

The dataset that we use for this study can be downloaded online[1]. The data consists of about 2000 images containing four different characteristic features: person, orientation, mood and eyewear. Also, the dataset comes with three different resolutions.

We train our algorithm on the training set, which corresponds to 60% of our dataset, and we choose classifiers for different features by running our algorithm on the validation set, 15% of our dataset. After choosing the best performing classifiers on the validation set, we run it on the test set.

[1]`http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-8/faceimages/faces/`

We implement the PCA algorithm in Python[2]. In our dataset we have three different resolutions (120×128, 60×64, and 30×32) for the same set of images.

4.2 Results

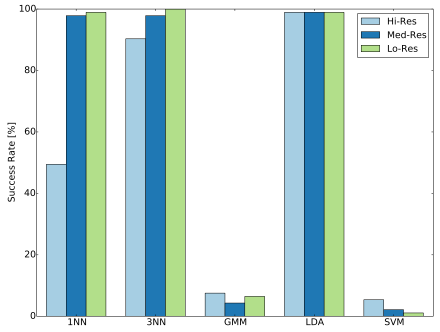

Figure 3: Success rates for predicting the person.

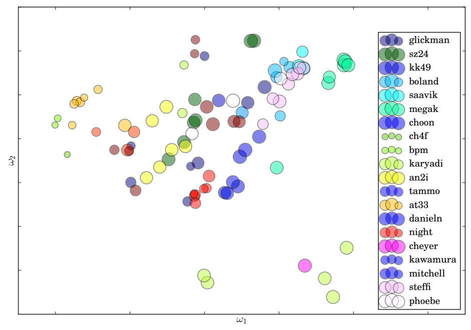

Figure 4: Scatter plot of two dominant principal components for validation set.

Each image is identified based on the algorithm used and success rates are shown in Figure 3. LDA's performance is uniform on all different resolutions even though the highest successs rate is for 3NN with 100% on the low resolution images. In light of this fact we choose LDA as the classifier for person on the test set. Figure 4 clearly shows the clustering of faces in the validation data. The size of the marker is proportional to the total Mahalanobis distance from the trained characteristic weights.

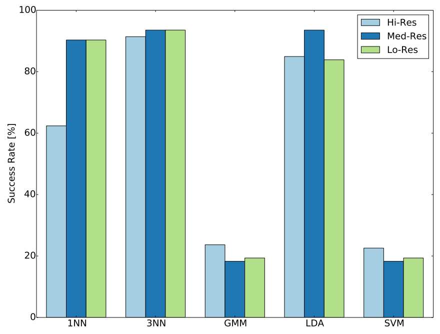

Figure 5: Success rates for predicting the orientation.

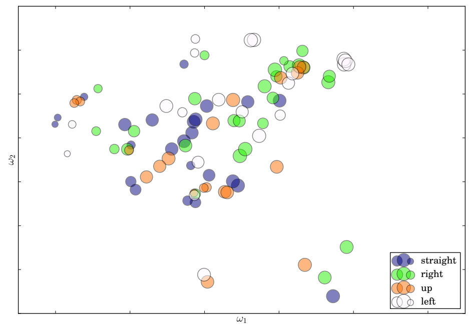

Figure 6: Scatter plot of two dominant principal components for validation set.

In Figure 5 it can be seen that 3NN performs best across the board. Naturally, we choose 3NN as the classifier for predicting orientation on the test data.

[2]Python 2.7.5

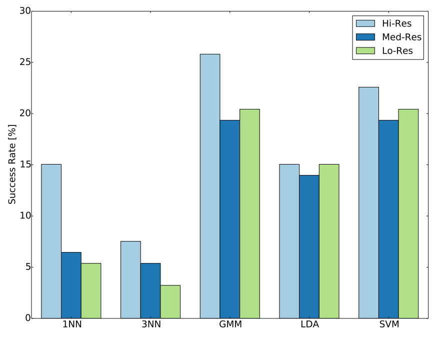

Figure 7: Success rates for predicting the mood.

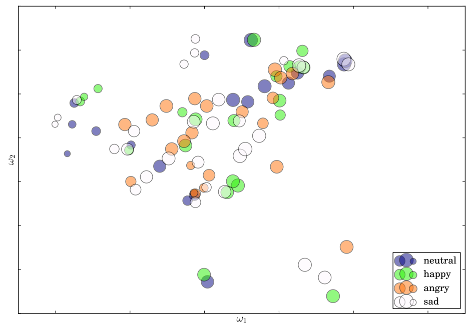

Figure 8: Scatter plot of two dominant principal components for validation set.

Notice in Figure 7 that most of the classifiers perform worse than random on predicting the mood. Still GMM and SVM perform with less bias than the other the other classifiers, so we select GMM as the classifier for mood. The reason for this can be partially explained by Figure 8 in which clustering among classes is absent.

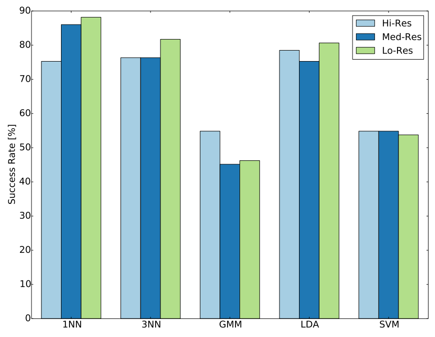

Figure 9: Success rates for predicting the eyewear.

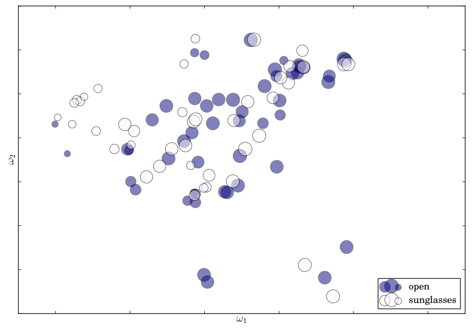

Figure 10: Scatter plot of two dominant principal components for validation set.

Figure 9 shows that 1NN is the most successful classifier for detecting eyewear, so we choose 1NN to use on the test data.

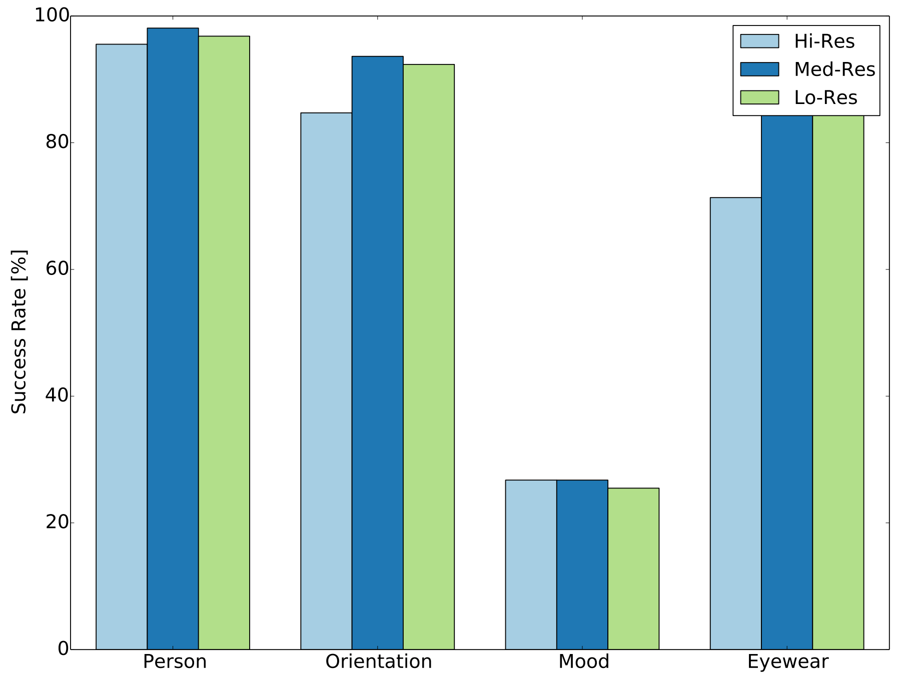

Figure 11: Success rates for the test data.

As can be seen in Figure 11, we achieve more than 70% success rates in predicting person, orientation, and eyewear for all resolutions. All of our classifiers perform worse than random in predicting mood. However, GMM performs as well as random.

5 Conclusion

In this study we examine the performance of different classifiers on predicting different facial features. We find that some classifiers perform better compared to others on each facial feature. Nonetheless, we cannot successfully predict mood with any of the tested classifiers.

The results show that there are better classifiers for different features. Classifier selection should be done using validation data to achieve the best performance.

References

[1] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Jan 1991.

[2] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*. The MIT Press, 2012.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] P.N. Belhumeur, J.P. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, 1997.