Jae Yoon; Mahshad Farnoush; Jagadeesh Meesala

Facial Recognition with PCA and Classifiers as Bayes and KNN

Priniciples Of Machine Learning

2021

Abstract

We apply facial recognition techniques on the DATA and POSE data sets. "DATA.mat" consists of cropped 24x21 pixel images of 200 subjects, 3 images each. We first try dimensional reduction techniques on the data-set using principal component analysis (PCA). Have tried different dimensional reduction values and derived optimal value with high accuracy. Then applied classifiers k-NN and Bayes to classify neutral vs facial expression and similarly applied PCA followed by k-NN and Bayes classifiers on "POSE.mat" data set and computed the accuracy of classification.

1 Introduction

Implementation of Facial Recognition using k-NN and Bayes classifiers with principal component analysis(PCA) as optimization technique.

2 Results

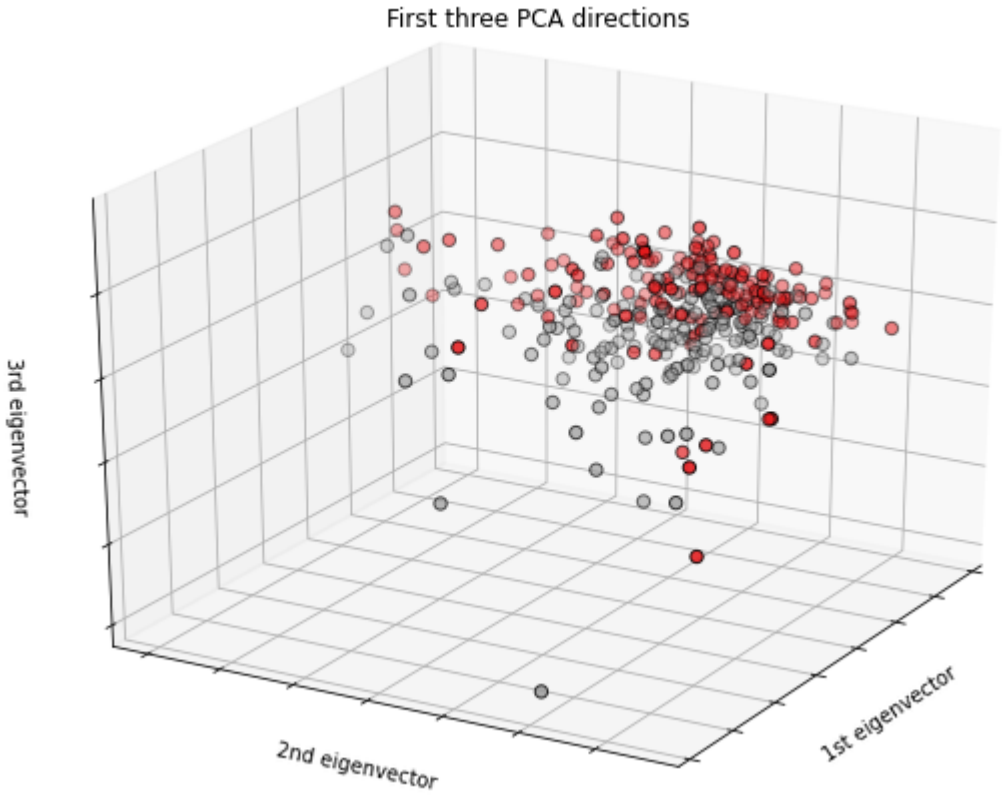
2.1 Experiments with data.mat

PCA followed by Bayes

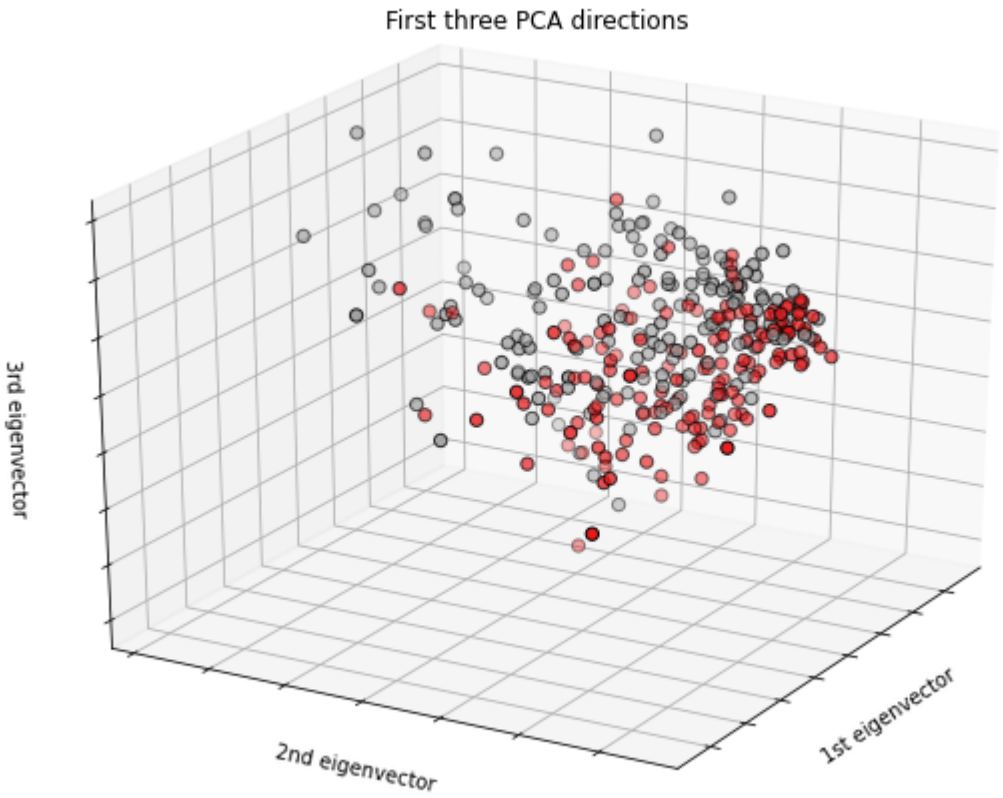
The dimensionality reduction offered by PCA is a very useful step for visualizing and processing high-dimensional datasets, while still retaining as much of the variance in the dataset as possible. Because PCA is translation variant, the faces must be frontal and well aligned on facial features such as the eyes, nose and mouth. PCA is useful for dimensionality reduction if the size of the training set is too small for the number of dimensions of the data. But if you are using all of the principal components, PCA won't improve the results of your linear classifier -if your classes weren't linearly separable in the original data space, then rotating your coordinates via PCA won't change that. The problem with image representation is its high dimensionality. Two-dimensional p x q gray scale images span am = pq-dimensional vector space, so an image with 24x 21 pixels lies in a 504-dimensional image space. The PCA method finds the directions with the greatest variance in the data, called principal components. By varying the number of principal components used, the dimensionality can be brought down as required.The following table gives the classification accuracy for PCA followed by the Bayes Rule for different number of principal components. It is seen that the accuracy does not vary by a large amount by increasing the number of principal components which means that most of the energy of the signal is contained within the first 20 principal components.

neutral	***smiling***
41	3
1	35

Table 1: confusion matrix



PCA with Scaler



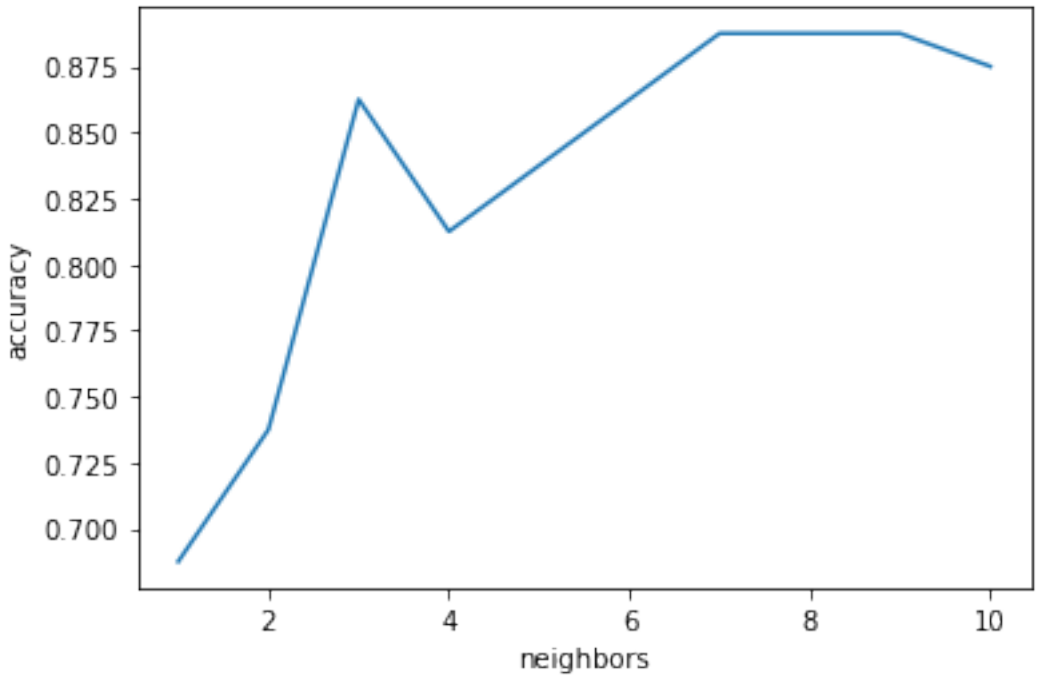
PCA without Scaler

PCA followed by k-NN

The following table gives the classification results after PCA is applied to the image vector and by using the kNN classifier. Results with k=1,2 and 3 neighbors are reported with varying number of principal components. Here we see an increase in the classification accuracy as the number of principal components used in the data representation increase.

$accuracy$	$correct$	$misclassified$
0.6875	55	25
0.7375	59	21
0.8625	69	11
0.8125	65	15
0.8875	71	10
0.875	70	10

Table 2: k-NN accuracy



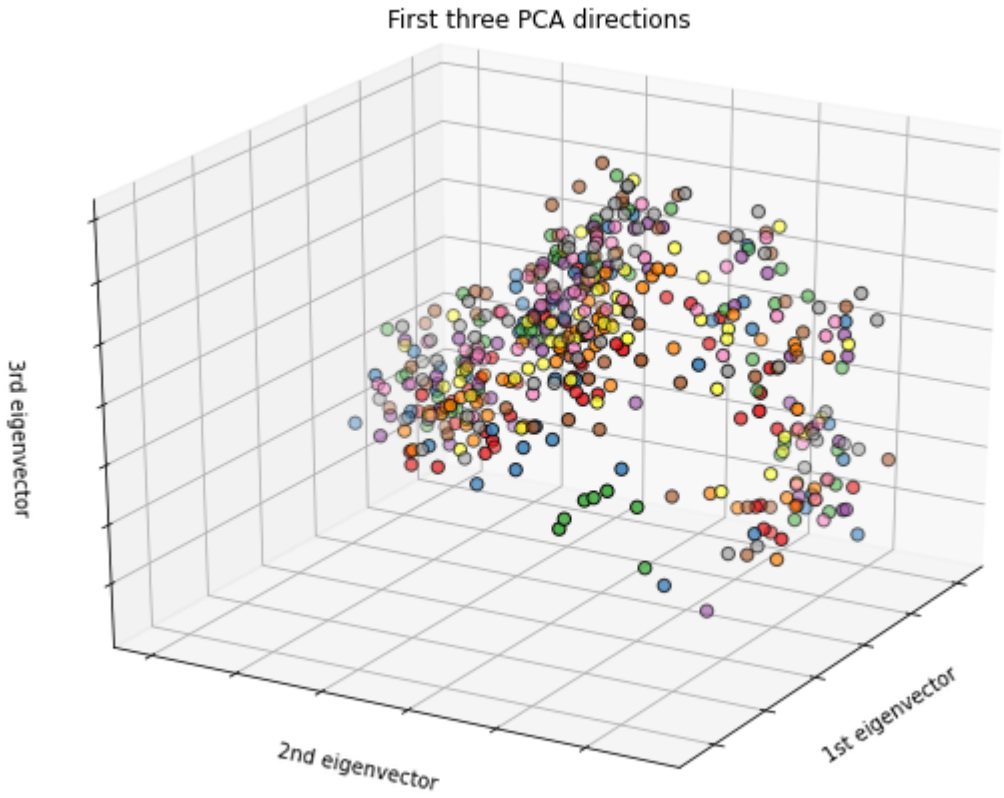
k-NN accuracy

Observations

- The Naive Bayes' classifier GaussianNB fits on the data better than Nearest Neighbors Classifier(k-NN) and achieves an accuracy of 0.925, which is better than the best accuracy achieved by the Nearest Neighbors classifier(k-NN). If we apply the Scaler on training set as a pre processing steps for PCA, we get higher accuracy which is 0.95, which is better accuracy without using.
- After trying with different n-components on PCA algorithm, the best value found was 15. Below that number we achieved worse accuracy as most of the information from the features was lost. Thus, the accuracy were not at best. If we train using the Scaler, we get higher accuracy on GaussianNB, but we get less accuracy on k-NN.

2.2 Experiments with pose.mat

(First) 9 images per subject are used for training and (last) 4 for testing. In general, we can see that the classification accuracy for every method are more than the previous data set as the number of training/testing samples are more.



PCA

PCA followed by Bayes

Used the first 10 poses of each subject for train and last three poses as test.

The naive bayes classifier does best when the PCA's n_compenents=16, changing it only worsens nb_test_accuracy

PCA followed by k-NN

Observations

- As we can see that with higher neighbor the accuracy declines. Also I found that with PCA `n_components`=10 the accuracy was worse and there was a significant improvement upon increasing it.

Further Processing

The data was not normalized so I normalized it. Also the `TRAIN_POSE_COUNT` was very deterministic of the final accuracy. `TRAIN_POSE_COUNT` = 7 proved to

accuracy	*correct*	*misclassified*
0.7475	305	103
0.4975	203	205
0.5073	207	201
0.4975	203	205

Table 3: k-NN accuracy

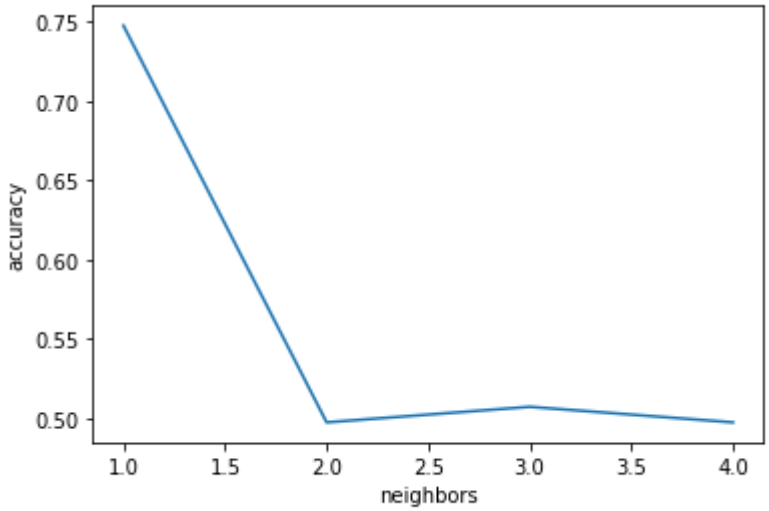


Table 4: k-NN accuracy

provide the best accuracy with the available data.

3 Code Files

The following notebook files are used for pre-processing and classification application:

- Jagadeesh Meesala
  - github link
- Jae Yoon
  - codelabs-code
- Mahshad Farnoush
  - codelabs link:

4 References

[**matplotlib tutorial**] https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html
[**sci-kit PCA documentation**] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.P
[**sci-kit k-NN**] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.htm
[**sci-kit Bayes**] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
[**sci-kit confusion matrix**] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_m