Machine Learning Engineer Nanodegree

Image Recognition Capstone Project

Andrey Semenov

I. Image recognition capstone project

Project Overview

The idea of the project has been taken from the description of one of project examples: Computer Vision Capstone Project. The goal of the project was to create a learning

agent in the form of bot for Telegram messenger which is able to recognize faces of famous people.

The set of basic techniques for image processing is described in the book Programming Computer Vision with Python. The basic methods and python libraries (PIL, NumPy, SciPy, Matplotlib) are described in details including the number of examples. Computer vision could be described as an field AI in which the computer is able to recognize various objects using different AI approaches from images or videos.

Face recognition is one of most challenging problems in computer vision and image processing. Good literature review of different approaches used for this problem could be found in the paper A Survey of Face Recognition Techniques by Rabia Jafri and Hamid R. Arabnia. Authors divide the techniques for the problems on the following: featured-based, statistical, AI based.

The face recognition problem has a number of practical applications:

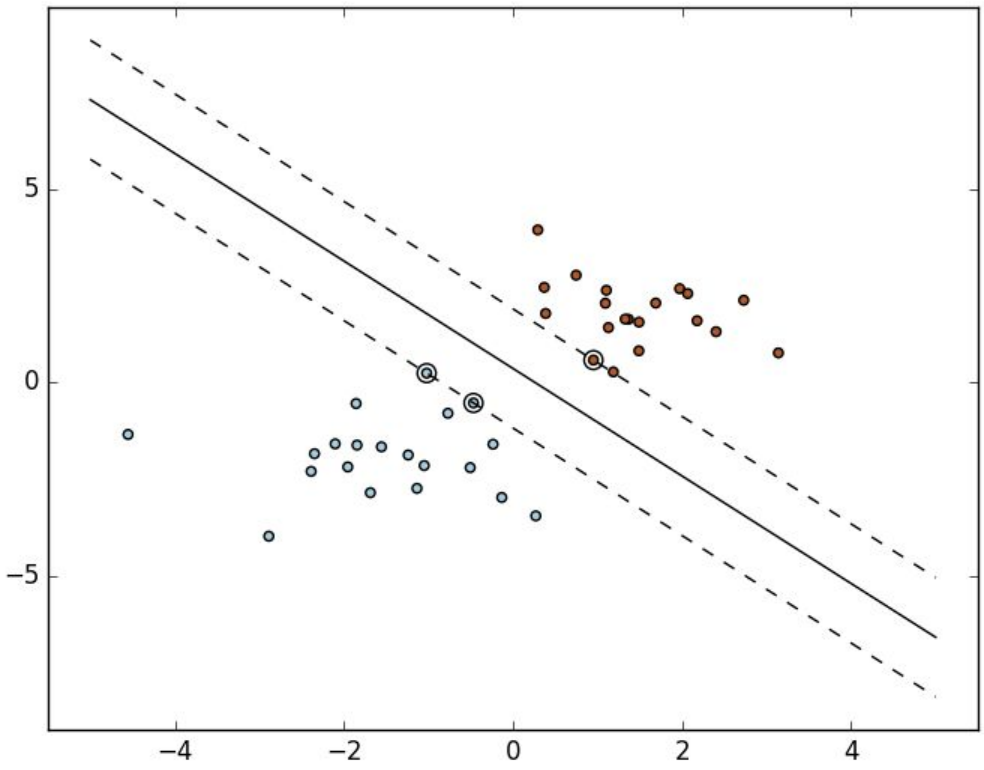
- Security surveillance
- General identity verification
- Entertaining applications (Apple Photos)

Problem Statement

The face recognition general problem can be formulated as follows: Given an input face image and a database of face images of known individuals, how can we verify or determine the identity of the person in the input image? More specific, Face Recognition or Face Identification is: given the picture of the face of an unknown person, identify the name of the person by referring to a gallery of previously seen pictures of identified persons.

The problem of face recognition is classification problem. Solution of the problems is a supervised by nature in the face identification problems, because we are comparing the unknown face with already known faces.
The goal for this particular project was formulation as: create the bot for Telegram messenger allowing to identify face of famous person. Python programming language has been chosen for the implementation of user application. Telegram messenger platform has been chosen for user interface development due to the open API. telepot framework was used for the bot development.

Machine learning component of application (database of faces, functions) was taken from sklearn face_recognition. In the solution of the problems the principal component analysis will be used to decrease the dimension of the problem. And Support Vector Machine (SVM) algorithm will be used for the solution of classification problem. A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.



Metrics

Different models build on training dataset were evaluated by the following criteria classification report:

- Recall score

Recall score=tp/(tp+fn). Where *tp* is the number of true positive outcomes on testing data set and *fn* is false negative outcomes. The recall is an ability of classifier to find all positive outcomes.

- Precision score

Precision score=tp/(tp+fp). Where *tp* is the number of true positive outcomes on testing data set and *fp* is false positive outcomes. The closer the precision to 1, the better the model.

- F1 score

F1 score is a harmonic average of precision and recall score (F1=2*(precision*recall/(precision+recall)))

Metrics above have been chosen instead of accuracy due to the better description of predictive ability of classifier due to the widely known accuracy paradox in which the model with highest accuracy is not the best one.

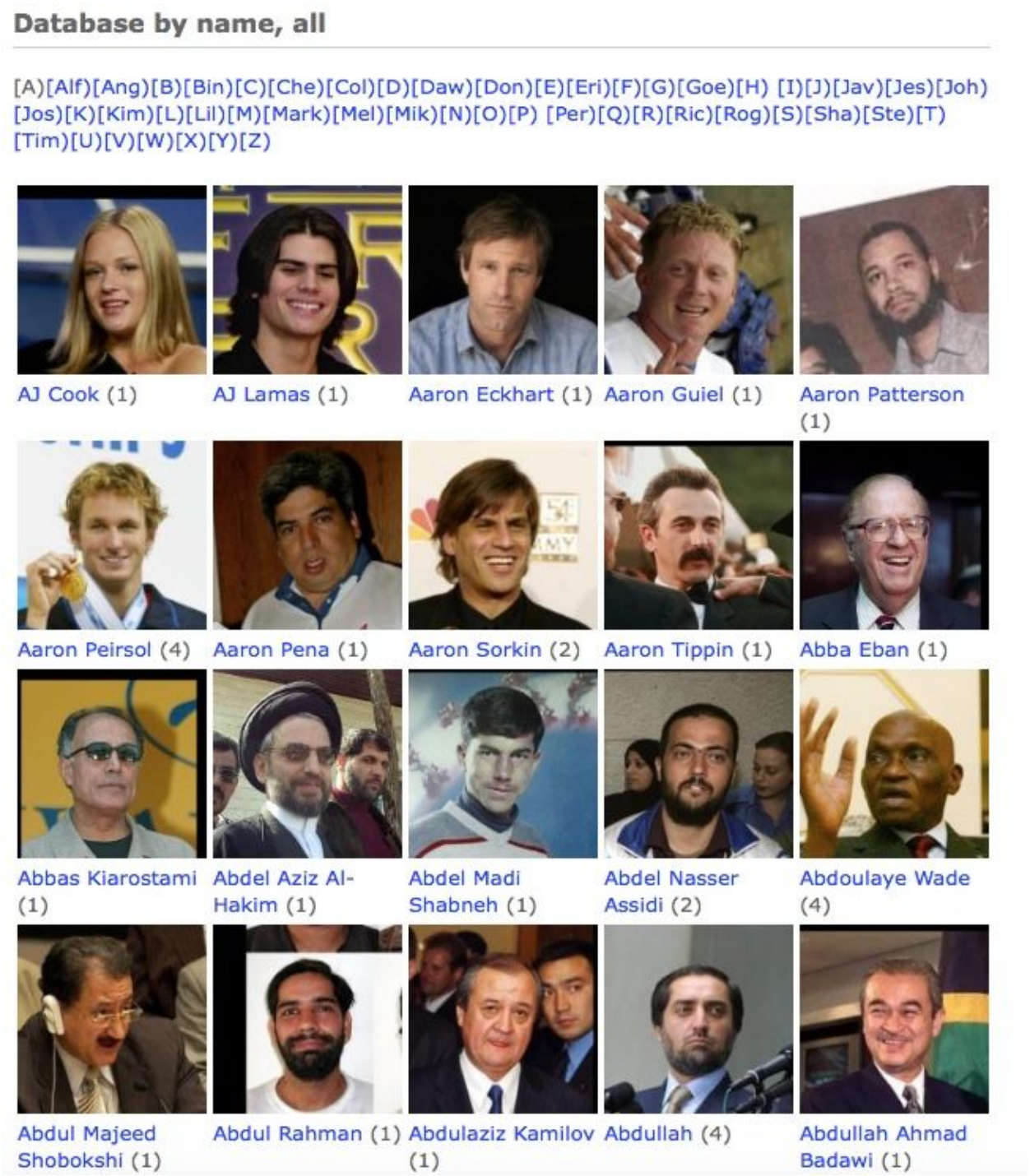
II. Analysis

Data Exploration

Publically available faces dataset was used. Dataset of labeled faces in wild contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set. In different parts of the project, I've varied the minimum number of photos per person for the analysis, and used the portion of dataset in the range from 1288 samples to 4325. Each picture is centered on a single face. Each pixel of each channel (color in RGB) is encoded by a float in range 0.0 - 1.0. The original images are 250 x 250 pixels, but the default slice and resize arguments reduce them to 62 x 74. The dataset is described in details in the paper let.

Exploratory Visualization

The dataset is a named series of famous faces people:



Many persons have more than 1 image (in brackets).

Algorithms and Techniques

There are three main techniques to be used for the solving this image recognition task:

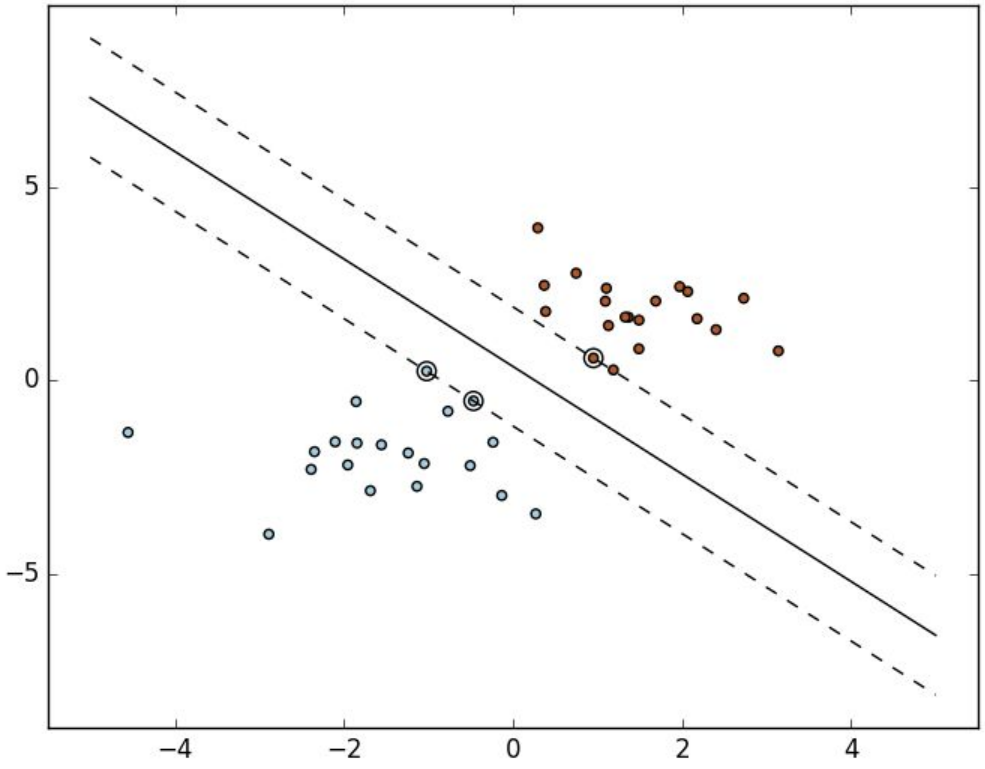
1) Principal Components Analysis PCA.

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. Detailed and clear explanation of linear algebra basics behind PCA are presented in the paper Principal component analysis by Aapo Hyv arinen.Principal components are the linear combination of vectors that contain as much of the variance in the input data as possible. In order to found the principal components, we need first to create covariance matrix of input data. Then found the eighteen values of covariance matrix. Eighteen vectors of covariance matrix will be our principal components. So, the in the basis of eighteen vectors the covariance matrix is orthogonal. So mathematically speaking, the task of finding principal component is the task of calculating eighteen vectors of the covariance matrix. By transforming the observation matrix into the basis of principal components, we are minimizing the correlation among the data.

It allows to decrease the dimension of the problem for face recognition by leaving only most important features (corresponding by first n eighteen values of covariance matrix) and removing correlated data. But how many principal components to keep? Different number of features (n=5,10,50,100,150,200,250) was compared in order to achieve optimal number of principal components.

2) Machine learning problems in such case is a supervised learning problems. The model is supposed to be trained trained on the part of the dataset (training dataset), tested on testing data set and used on images received from the user. For supervised learning part the Support Vector Classifier was used SVM. In the supervised learning project, the SVM classifier got one of the highest scores.

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.



The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

3) In order to make the learning agent able to learn new images in addition to existing dataset, if the image is not recognized, the image will be stored in the dataset. And in the next request, the classifier will take into account the stored image received from user.

Benchmarks

As a benchmark for the proposed analysis the image-restricted results with no outside data could be used for different techniques used on the same data lfw dataset benchmark:

Technique	Mean classification accuracy and standard error of the mean
Eigenfaces, original	0.6002 ± 0.0079
Nowak, original	0.7245 ± 0.0040
Nowak, funneled	0.7393 ± 0.0049
Hybrid descriptor-based, funneled	0.7847 ± 0.0051
3x3 Multi-Region Histograms (1024)	0.7295 ± 0.0055
Pixels/MKL, funneled	0.6822 ± 0.0041
APEM (fusion), funneled	0.8408 ± 0.0120
APEM (fusion), funneled	0.8408 ± 0.0120
MRF-MLBP	0.7908 ± 0.0014
Fisher vector faces	0.8747 ± 0.0149
Eigen-PEP	0.8897 ± 0.0132
MRF-Fusion-CSKDA	0.9589 ± 0.0194
POP-PEP	0.9110 ± 0.0147
Spartans	0.8755 ± 0.0021

III. Methodology

Data Preprocessing

A few data preprocessing steps was used for the dataset:

- The time of the bot response is crucial. So in order to manage the part of dataset on which the model is trained, the min_faces_per_person variable is using in order to take the faces with the >threshold number of pictures. All images from dataset are resizing with resize=(0,1] ratio parameter.
- The next step is splitting the data between training and testing data set. For the existing data set the ratio of 3:1 was used, for the bot image recognition all images are using as testing data set and only 1 received from user as testing data set.
- After receiving the image from user, the image is processing:

```
pil_im=Image.open(image)
image_resized=pil_im.resize((w,h))
```

- Also image is converting to 1D numpy array:

```
face = array(image_resized.convert("L"), "f")
face_1D= np.transpose(face.ravel())
```

Implementation

1. PCA was implemented by the following way.

```
# Compute a PCA (eigenfaces) on the face dataset (treated as unlabeled
# dataset): unsupervised feature extraction / dimensionality reduction
n_components = 150
print("Extracting the top %d eigenfaces from %d faces"
    % (n_components, X_train.shape[0]))
t0 = time()
pca = PCA(n_components=n_components, svd_solver='randomized',
      whiten=True).fit(X_train)
print("done in %0.3fs" % (time() - t0))
eigenfaces = pca.components_.reshape((n_components, h, w))
print("Projecting the input data on the eigenfaces orthonormal basis")
t0 = time()
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
print("done in %0.3fs" % (time() - t0))
```

In the beginning the number of principal components was chosen (150 in this example). After that the training and testing datasets were converted into the space of principal components.

2. SVM classification implementation.

```
# Train a SVM classification model
print("Fitting the classifier to the training set")
t0 = time()
param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5],
              'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1], }
clf = GridSearchCV(SVC(kernel='rbf', class_weight='balanced'), param_grid)
clf = clf.fit(X_train_pca, y_train)
print("done in %0.3fs" % (time() - t0))
print("Best estimator found by grid search:")
print(clf.best_estimator_)
```

SVC classifier from sklearn package was used with variable parameters ‘C’ and ‘gamma’.

3. Recognizer bot implementation in telegram.

Telepot framework was used and class Bot was created. Bot Recognizer was created by BotFather in Telegram messenger. Received Token was used for authorization. Bot is handling messages by method: on_chat_message. In case of receiving an image, the image is storing and used for recognition.

Refinement

In order to found the best solution using PCA, the following number of components was used: (5,10,50,100,150,200,250). And results log for 12 persons with the highest number of pictures are below. The performance of the model depends upon number of PC could be compared by the average precision/recall/F1 score. Total dataset size:

n_samples: 1288
n_features: 1850
n_classes: 7

1. **n_components=5**

Best estimator found by grid search:
SVC(C=50000.0, cache_size=200, class_weight='balanced', coef0=0.0,

decision_function_shape=None, degree=3, gamma=0.1, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)

precision	**recall**	**f1-score**
0.41	0.34	0.36

2. **n_components=10**

Best estimator found by grid search:
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
decision_function_shape=None, degree=3, gamma=0.1, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)

precision	**recall**	**f1-score**
0.50	0.48	0.49

3. **n_components=50**

Best estimator found by grid search:
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
decision_function_shape=None, degree=3, gamma=0.01, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)

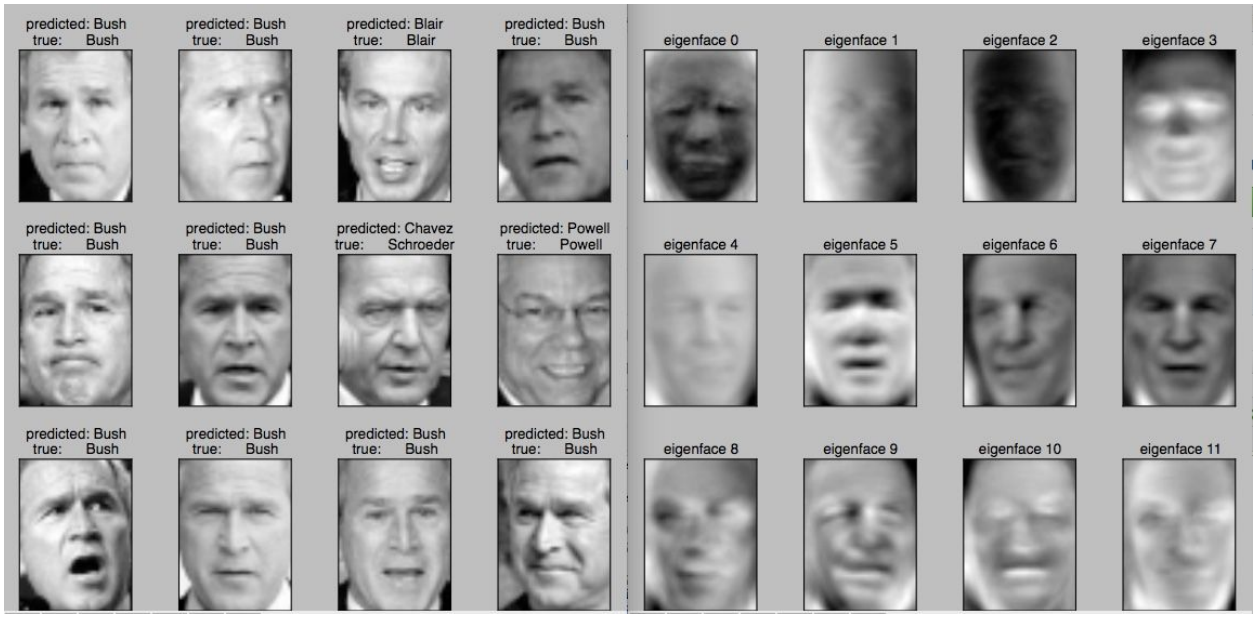
precision	**recall**	**f1-score**
0.82	0.82	0.82

4. **n_components=100**

Best estimator found by grid search:
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
decision_function_shape=None, degree=3, gamma=0.005, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)

precision	**recall**	**f1-score**
0.86	0.86	0.86

Example of visual analysis below: prediction vs true name. And display of first 11 eighteen vectors:



5. **n_components=150**

Best estimator found by grid search:
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
decision_function_shape=None, degree=3, gamma=0.005, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
Predicting people's names on the test set
done in 0.060s

precision	**recall**	**f1-score**
0.86	0.85	0.85

6. **n_components=200**

Best estimator found by grid search:
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
decision_function_shape=None, degree=3, gamma=0.001, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)

precision	**recall**	**f1-score**

0.85	0.85	0.85

As it could be seen from the beginning, in case of low number of PC (5,10) the dimension is not enough to recognize faces and scores are poor. But starting from 50, the average F1/Precision/Recall scores are increasing, which means that model is improving the predictive ability. The local maximum is found for n=100 components (all average scores are equal 0,86) and later increase of number of components to 150 and 200 only a little bit decrease average scores. It is mean, that 100 components are allowing to extract all information from images needed for this task.

IV. Results

Model Evaluation and Validation

The final model was constructed for famous people with >40 images in the database and 100 principal components and have the following parameters. The number of 100 principal components were selected due to the maximization of F1 score in the analysis above. The log of model creation is following:

```
Total dataset size:
n_samples: 1867
n_features: 1850
n_classes: 19
Extracting the top 100 eigenfaces from 1400 faces
done in 0.810s
Projecting the input data on the eigenfaces orthonormal basis
done in 0.057s
Fitting the classifier to the training set
done in 54.465s
Best estimator found by grid search:
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.005, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Predicting people's names on the test set
done in 0.114s
```

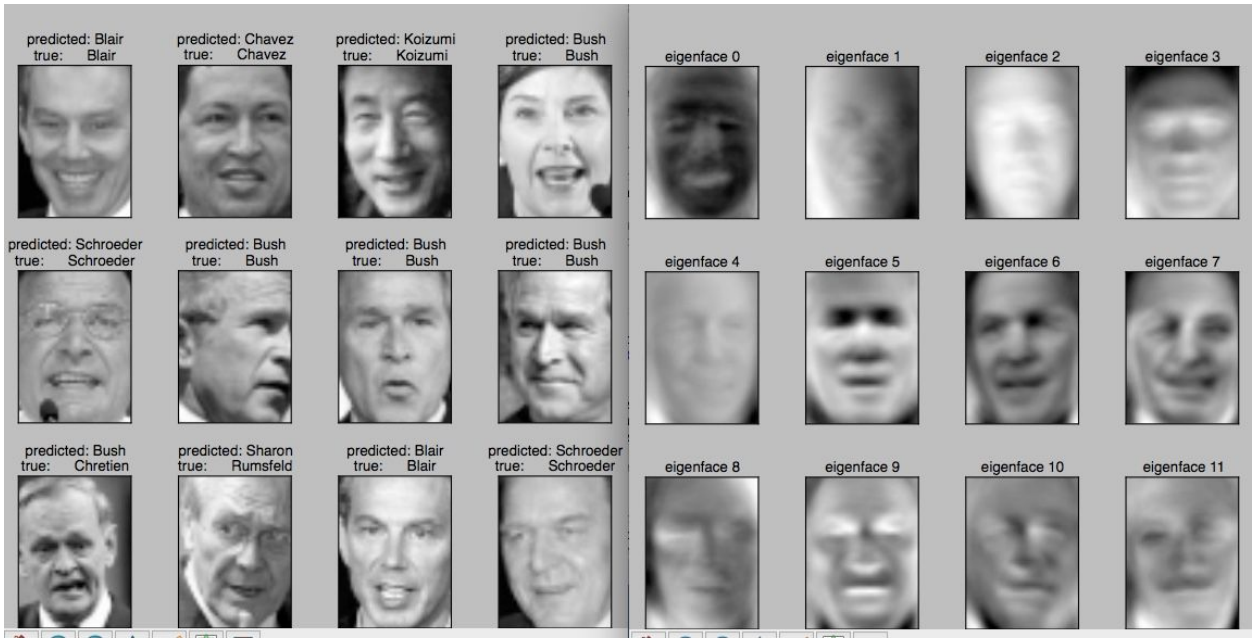
precision	**recall**	**f1-score**	**support**	**Mean classification accuracy**

0.79	0.78	0.77	467	0.777

The model fitting time is less than 60 s, which make the model available for usage in the user application. Model is able to recognize 19 famous people and tell to which of them the provided photo is similar. As well the model mean accuracy is equal to 0,777 which is in the mid distribution results of provided benchmarks, which makes the model statistically applicable. All of mentioned above make model robust.

Justification

Based on the avg/total scores for precision, recall, F1>0.77 the model is able to recognize the face for 19 famous people. As it is shown below, the faces predicted are correct for the majority of predictions.



V. Conclusion

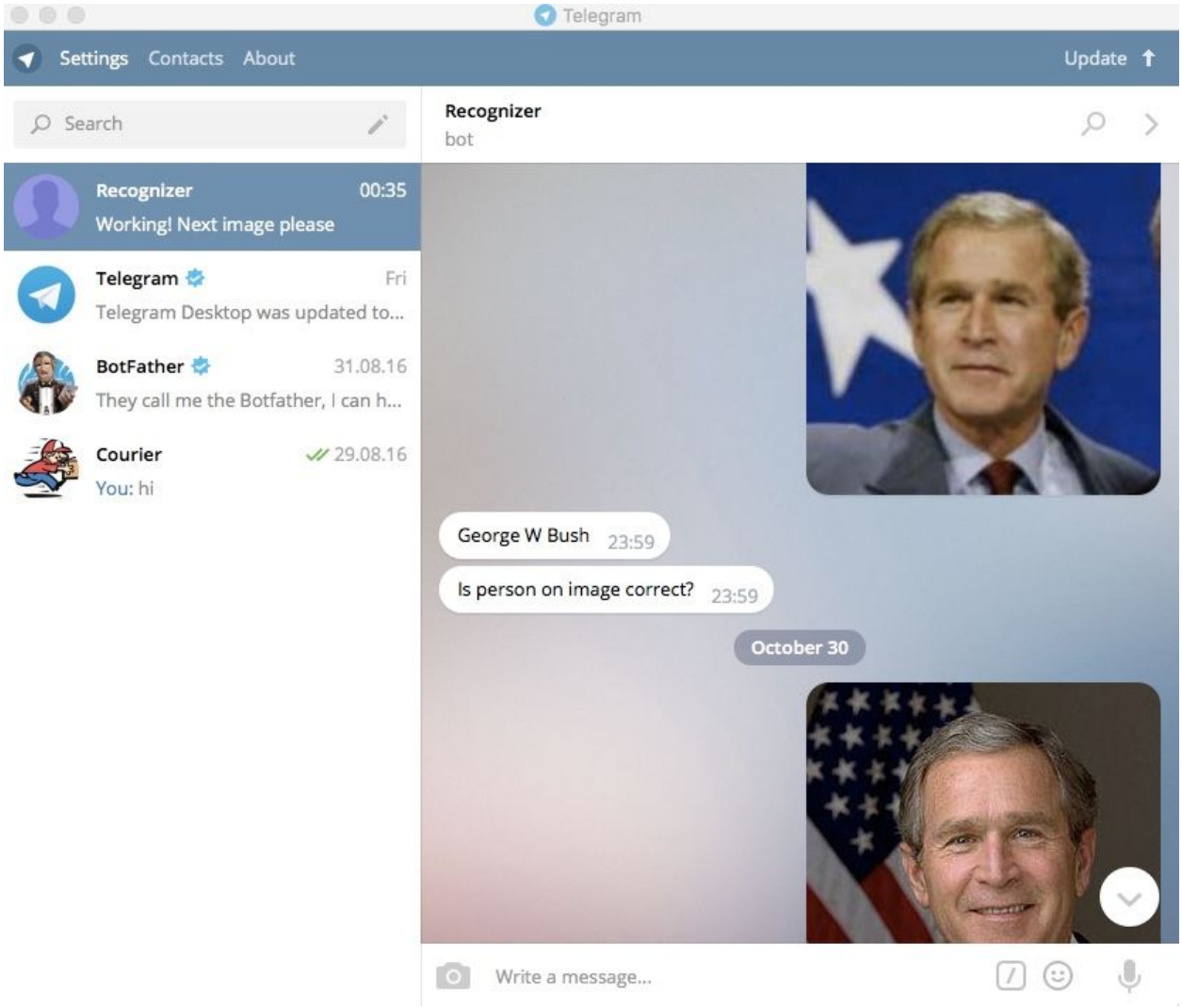
Face recognition problems is represents quite difficult problem due to the limitation in calculation power, images database and current approaches to image processing. But use of modern filtering (PCA) and supervised learning algorithms (SVM) could found the solution of the problem in case of sufficient image database and time available. Apter the

launch of code in *index.py* file the folowing actions will be executed (detailed explanation in readme.md):

1. Dataset of pictures of famous people will be uploaded to the machine.
2. The PCA and SVM algorithms will be used for the training of the model for image recognition and defining metrics.
3. The Telegram bot code for Recognizer bot will be launched.
4. As soon as the bot will receive image from any user with picture, the bot will predicted to whom the face is similar.

Free-Form Visualization

The user interface of application is looks like this:



The image could be send to bot and bot will advice to which famous person the face is similar.

Reflection

The were two particularly interesting and difficult at the same time parts of the project:

1. Image recognition techniques. The whole set of instruments for image recognition: filtering, image representation in the form of arrays, PCA transform, etc... were quite new for me. Great help for me was coverage of PCA on this example in the education materials.

2. Bot user interface. Thanks to open API it was possible to build user interface for the application available on different platforms: mobile, desktop, etc... Interesting part was in building the bot and handling the messages.

Main lessons learned from this project for me are the techniques for image processing and image data analysis. It was first time when I've used image recognition techniques and works with images as with arrays of data. It allows me to solve the real problems. As well PCA shows good applicability to extraction of unique information from images and reduction of problems dimension. SVM classifier validate its applicability to the complex classification problems.

Improvement

Derived model is robust enough in case of the big number (>50 images) per person. In case of decreasing the threshold of images per person to 10 for example, the model has the following performance:

```
Total dataset size:
n_samples: 4324
n_features: 1850
n_classes: 158
Extracting the top 100 eigenfaces from 3243 faces
done in 1.293s
Projecting the input data on the eigenfaces orthonormal basis
done in 0.136s
Fitting the classifier to the training set
done in 620.780s
Best estimator found by grid search:
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.005, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Predicting people's names on the test set
done in 2.009s
precision    recall  f1-score   support
```

precision	**recall**	**f1-score**	**support**
0.51	0.51	0.47	1081

So there are 2 clear issues:

- Decrease of scores in case of using more images in the dataset and less images per person for model training. But still the derived model have scores >0.5.

- The training of the model is taking >10 minutes which makes it hard to reeducate the model after adding the images to the dataset.

The solution of those problems could significantly improve the project. In order to do it there are 2 possible solutions:

1. Train the model (for example complex neural network) not in real time. For example spent few days and huge database of image and then use it for all face recognition without training on newly received images from the user. Or train it periodically, let's say every couple of days.
2. Increase of accuracy could be achieved by increasing number of images per person. 10 images is not enough, but 40 (as it is show before) allows to make model using SVM classifier with good F1 score.
3. Other option could be use of different classification method: neural networks, decision trees, etc... They could improve the results.