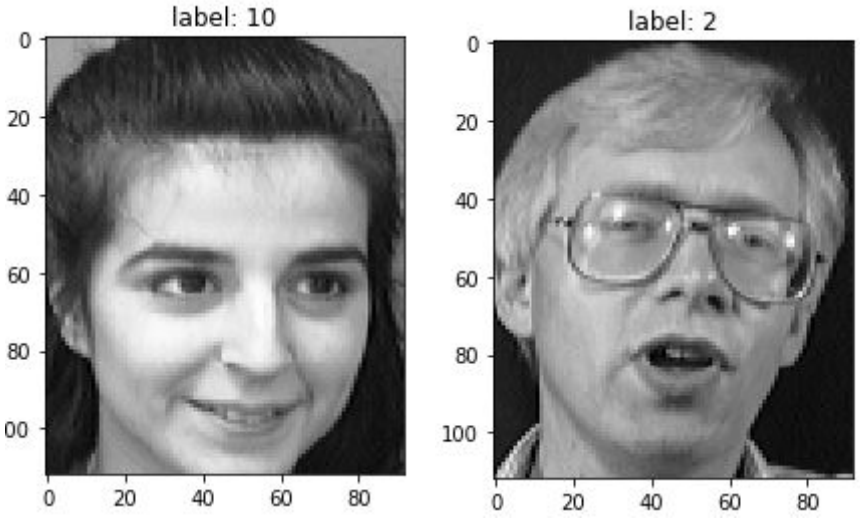
Report - Face Recognition.

Problem overview.

Find solution for face identification problem on ORL dataset.
Dataset link here.
The files are in PGM format. The size of each image is 92x112 pixels, with 256 grey levels per pixel. The images are organised in 40 directories (one for each subject), which have names of the form sX, where X indicates the subject number (between 1 and 40). In each of these directories, there are ten different images of that subject, which have names of the form Y.pgm, where Y is the image number for that subject (between 1 and 10).

Few examples of dataset:



Algorithm explanation

I use MSE method (Euclidean distance) for solving face identification problem.

We use two methods for vectorization dataset:

- preprocesing_images - return file names of all images from directory dataset
- create_data_matrix - return matrix (vectorized dataset) and labels

We have vectorized dataset, shape 400 - 10304, where each row represent image and vector **labels** where each component correspond label of image.

Method split_dataset is used for splitting dataset (80% train and 20% test). I select 2 elements from each classes
After splitting we have train_set (320 - 10304) and test_set (80 - 10304)

Steps:

1) take first row (vector **v**) from test_set.
2) subtract **v** from each row train set (vector **delta**)
3) calculate distance = ||**delta**|| - euclidean norm. For example, **delta** is vector [x1,x2,...x10304], then ||**delta**||=(x1**2+x2**2+...+xn**2)**0.5.
4) As a result we have vector **distance** [distance1, distance2, …, distance10304]
5) Find **index** minimum value in vector **distance**: argmin(**distance)**

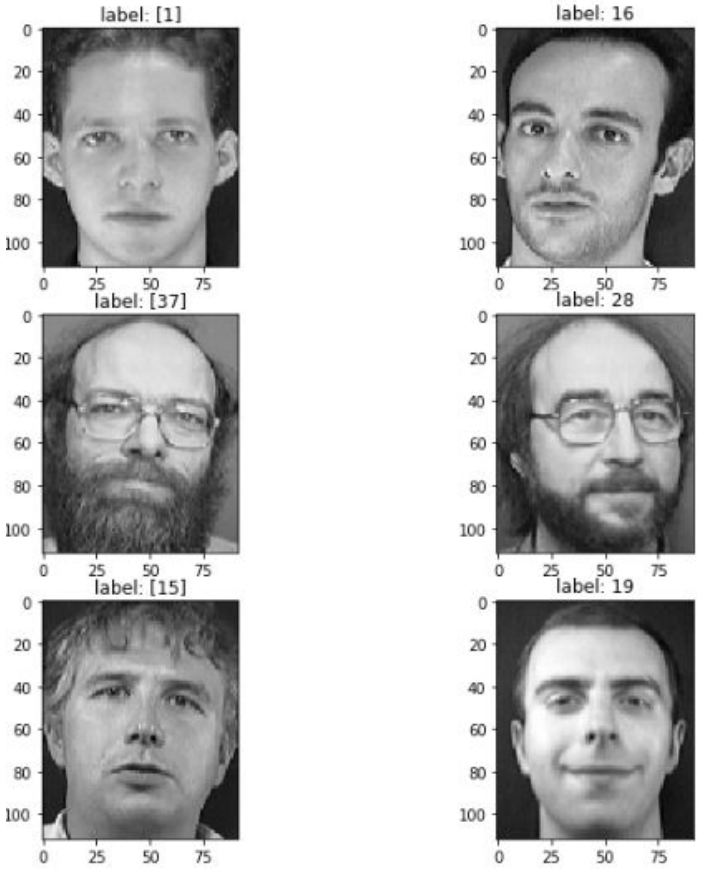
6) Find label that correspond **index: label[index]**
7) Calculate Accuracy: acc = (true_answ/test_labels.shape[0])*100

Method **model_face_recognition** performs identification employee.
It consist of:

- identify_employee - load unknown photos and identify employee (MSE algorithm)
- accuracy - calculate accuracy
- show_errors - visualize errors of model

Results without data normalization.

```
Accuracy for test dataset = 96.25
```



Optimization / Normalization steps.

There are two approaches to normalize dataset:

- subtract_mean - for each feature in the dataset calculate mean value and subtract it from the feature.
- standard_norm - normalize each image separately: calculate mean from data entry (one face image), subtract mean from original image and divide it by standard deviation of the image.

Table of different types normalization and corresponding accuracy:

Accuracy	subtract_mean	standard_norm
by features	95	95
by samples	95	95

Data normalization didn't improve accuracy in our case.

PCA (Principal Component Analysis)

One of the most important applications of PCA is for speeding up machine learning algorithms.
I realize method ‘my_pca’.

Steps of PCA algorithm:

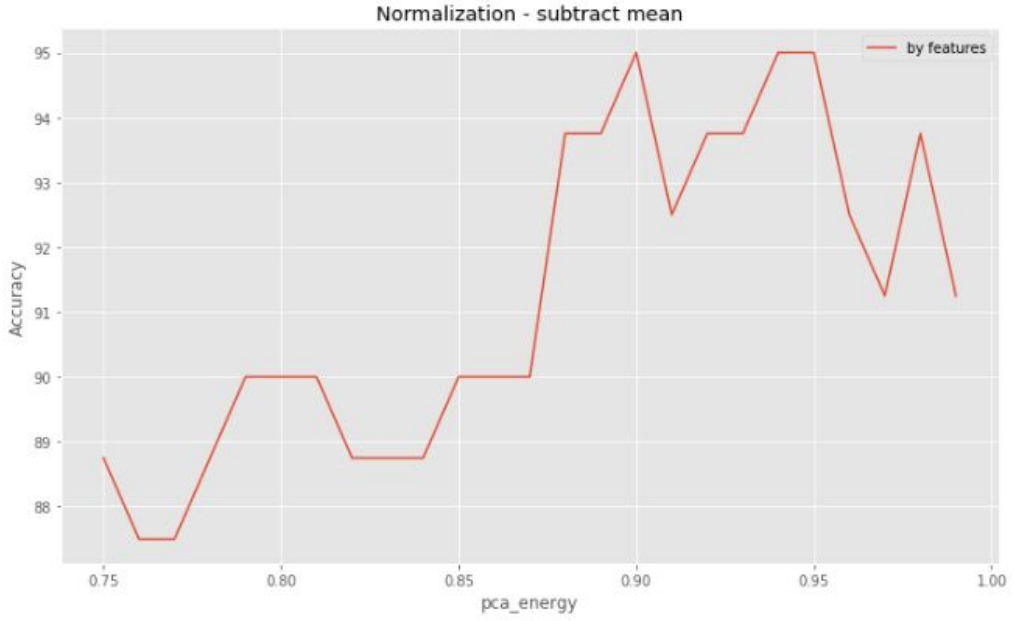
- Before using PCA we need to standardize dataset: *Normalize(X)*.
- Compute the the Covariance Matrix. *Cov_matrix = X * X.T*
- Compute the eigenvalues λi and eigenvectors vi of Cov_matrix
- Order the eigenvectors descending by their eigenvalue.The k principal components are the eigenvectors corresponding to the k largest eigenvalues.
- Compute transform matrix into PCA subspace, *W= important_vec * X*
- Projecting dataset into the PCA subspace: *X_pca = W *X.T*

We have different values of pca energy (0.95, 0.98, and 0.99).
Before applying PCA, I use ‘subtract mean’ normalization by features.
Below you can see results of using PCA:

Table accuracy for different values of pca_energy

pca_energy	number of components	accuracy
0.99	263	91.25
0.98	228	93.75
0.95	160	95

The best accuracy 95% for pca energy=0.95.
The shape of dataset after projection into PCA subspace equal 320 - 160.
You can see more details on graph, where I change pca_energy in range (0.75 - 1):



We can reduce dataset to lower dimensional then original dataset and have similar accuracy.
In this case we can work with only 160 features instead of 10304 !

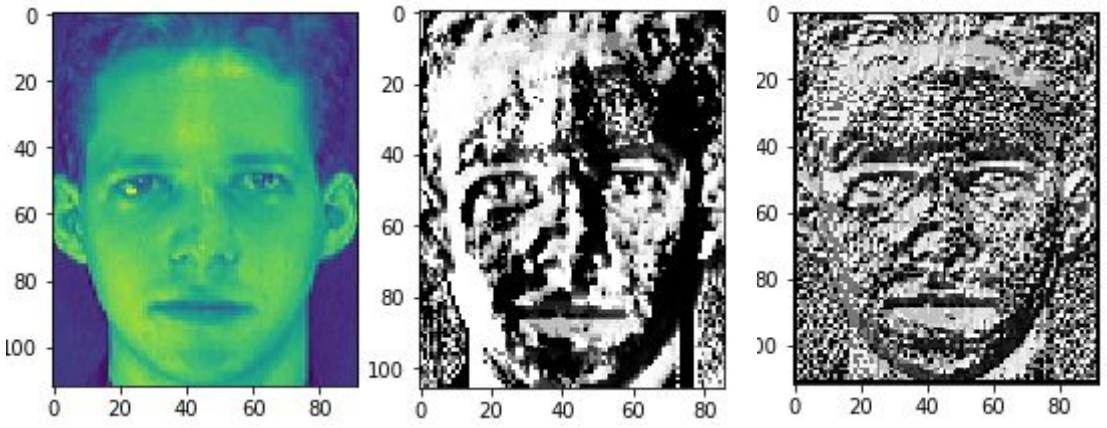
LBP (Local Binary Pattern)

The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. Take a pixel as center and threshold its neighbors against. If the intensity of the center pixel is greater-equal its neighbor, then denote it with 1 and 0 if not.
I have 2 free parameters, the number of blocks(divide the images into N*N blocks), the length of radius(R).
Before implementation LBP algorithm, you have to standardize dataset, please use method 'standard_norm', that transforms data to have a mean of zero and a standard deviation of 1
Description of LBP algorithm:

- divide image into N*N blocks
- for each pixel take surrounding neighborhood with radius R
- threshold it to construct a set of binary digits
- converting it into a decimal representation
- build LBP histogram for each block (bins = the number of uniform patterns)
- concatenate LBP histograms (for each blocks) into one feature for image
- use this feature for face recognition.
- standardize dataset, please use method 'standard_norm' by samples (axis=1), that transforms data to have a mean of zero and a standard deviation of 1
- use chi-squared distance to compare two LBP histograms

Examples images after implementation LBP transformations:

original (N=1, R=3) (N=1, R=1)



I use chi-square distance for solving face identification problem.

Accuracy for combinations of N,R. N=1,2,...6, R=1,2...5

Features	Acc	N*N	R
59	83.75	1	1
243	91.25	1	2
555	83.75	1	3
995	82.5	1	4
1563	68.75	1	5
236	**96.25**	**2**	**1**
972	93.75	2	2
2220	95	2	3
3980	90	2	4
6252	88.75	2	5
531	**96.25**	**3**	**1**

2187	96.25	3	2
4995	**97.5**	**3**	**3**
8955	**96.25**	**3**	**4**
14067	92.5	3	5
944	**97.5**	**4**	**1**
3888	**96.25**	**4**	**2**
8880	95	4	3
15920	92.5	4	4
25008	90	4	5
1475	95	5	1
6075	95	5	2
13875	87.5	5	3
24875	83.75	5	4
39075	72.5	5	5
2124	93.75	6	1
8748	88.75	6	2
19980	85	6	3
35820	78.75	6	4
56268	67.5	6	5
Features	**Acc**	**N*N**	**R**

The best result accuracy is **97.5%** for 2 set of parameters:

- N=3, R=3 (4995 features)
- N=4, R=1 (944 features)

Additional steps.

Let's use interpolation for finding surrounding neighborhood and compare accuracy.

LBP parameters	interpolation off	interpolation on
N=3,R=3	97.5	97.5
N=4,R=1	97.5	98.75

The accuracy **98.75** (LBP parameters N=4,R=1) is the best result, but interpolation is high computational cost operation

Apply PCA after LBT transformation

Table of accuracy for LBP parameters: N=3,R=3 and different pca energy

pca energy	0.95	0.98	0.99
Features	13	37	71
Accuracy	**87.5**	**91.25**	**93.75**

References and sources

https://docs.opencv.org/3.1.0/da/d60/tutorial_face_main.html
https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60
https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

Author: Vasyl Shandyba
15.04.2019