Exercises for
Introduction to Machine Learning
Paul Stoewer, Johannes Enk, Xingjian Kang, Yu Gao, Oliver Korn
Task 5: deadline: 14th July 2023

Pattern Recognition Lab

Exercise 05: Face Recognition

In this task, you are supposed to implement a face recognition pipeline. First, you will extract features from the given images and use them to fit a classifier. Then, to check whether your classifier works, its performance will be evaluated on unseen test images. Additionally, you will reconstruct a given image using your generated eigenfaces.
The algorithm should work with an offline database as well as on a live video. Therefore, the *Main.py* class has a flag called *onlineDetection*. Test your implementation first with the offline version, as this version will be checked by the unit tests.

Hint: This exercise is closely related to the provided Jupyter-notebook. Please first check out the lecture slides and understand the notebook before implementing this exercise.

Hint: you will need to train and use an SVM from sklearn. You probably do not know how to do it, and it's perfectly fine, don't worry. Search on your favorite search engine "sklearn svm", open the result corresponding to the documentation of the library, and have a look at the methods for the SVC class – you will need to use two of them.

Exercise 1 Read first

The python files contain a lot of information which will help you solving this exercise. Please, read quickly[1] the python files in order to have an idea of their content before proceeding further. Oh, and have you read the exercise introduction and the two useful hints?

Exercise 2 Eigenfaces.py

Within the module *eigenfaces.py*, the main tasks are generating eigenfaces from a set of images, reconstructing an image using a subset of these eigenfaces and training a classifier with the eigenface-coefficients as features. These tasks are to be implemented in the 3 methods *process_and_train*(), *reconstruct_image*() and *classify_image*(). Note, that the provided skeleton further modularizes the code into helper functions to ease the error detection provided through the unit tests. Make sure not to change the method *create_database_from_folder*() as it is needed to conduct the classification of your live video.

process_and_train()

In this method, you need to calculate the eigenfaces using the provided training face images. Plot one of the generated eigenfaces to check your implementation. Use these eigenfaces as features to train the classifier. Return the requested number of eigenfaces as a matrix with the eigenfaces as row vectors. **Pay attention**

[1] Of course, read these files more carefully while solving the exercises!

to the different shapes of the matrices. Use the helper methods to fulfil the task.

reconstruct_image()

In this method, you need to reconstruct a given image by computing the weighted sum of the provided eigenfaces with their coefficients. The coefficients are generated by calculating the similarity between an eigenface and the input image. The number of used eigenfaces for reconstruction can be played with to visualize the impact on the reconstruction result. You can change the number of eigenfaces to get a better understanding of this relationship in *Main.py*.

If you use enough eigenfaces for the reconstruction, then the output image should be almost identical to the input (or totally identical if you used all eigenfaces). If not, then you have an issue somewhere.

classify_image()

In this method, you should predict the label of a given image by using the trained model. To do so, compute the features of the input image and feed them into the classifier. Return the output of the classifier without modifying it. In the main function, a complete test set is predicted. The output is printed and should resemble Fig. 1. **As the test set is chosen randomly, the scores can vary!** Just try a few times.

```
Training done in 4.156s
Prediction of 305 faces done in 1.483s
Prediction time per face 0.005s
                   precision    recall  f1-score   support

     Ariel Sharon       0.73      0.92      0.81        24
     Colin Powell       0.82      0.82      0.82        56
  Donald Rumsfeld       0.92      0.63      0.75        35
    George W Bush       0.86      0.93      0.89       125
Gerhard Schroeder       0.83      0.80      0.81        30
       Tony Blair       0.84      0.74      0.79        35

         accuracy                           0.84       305
        macro avg       0.83      0.81      0.81       305
     weighted avg       0.84      0.84      0.84       305
```

Figure 1: Example of scores of the SVM with C = 0.025 and a linear kernel.
