Report

Required libraries

- numpy (for mathematical operations for matrixes as well as for managing data e.g. dot product or handeling training and validation data)
- pylab (not in use anymore)
- sklearn (for SVM's)
- mathplotlib (used for working with plotting and images)
- PIL (for working with IMages)
- os (for opening folders)
- pickle (for saving and reading data from files)
- random (for random values)
- functools (for functional programming)
- time (used to show the time)
- pandas (used for ploting and working with images)
- seaborn (used for plotting)
- keras (just used for the CNN)

Programs and the usage of them

- cnn.py
- hogFeatures.py
- pca s-sne.py
- svms.py

cnn.py

This is used for training a Convolutional neuronal network.

Requirements:

trainningData folder with trainingDataComplete files

Outputs:

A trained cnn model, which is saved in trained-models under the name *trained-CNN*

createPyramid.py

This is used for creating a pyramid. The pyramid is a structure, which scales an image. After each scale, a 96 times 96 window slides over the picture and creates pictures out of it. These pictures are then used to generate HOG-Data, which can afterwards be fed into a pre-trained model. After the model says, that there is a face on the picture or not, the boundingboxes can be drawn.

why different scales?

Through the different scales, faces in varying sices can be found. Because the models are trained, that an face has the perfect size and is centered on the window. Therefore we slide pixelwise over the given image.

Requirements:

Path_to_image variable should contain a string, to which image it goes

Output:

a *pyramid* file consisting of an structure to save the cordinates, the image and the cropped images.

Why scaling the image?

After the scaling, a 96 times 96 "filter" goes over the image and cre

hogFeatures.py

This is used for generating the hogFeatures and saving them to files/ Or it is used to create a pyramid-hog file from a pyramid file, which can later be used to give it to the models

Requirements:

../FDDB-folds/FDDB-fold-01-ellipseList.txt to ../FDDB-folds/FDDB-fold-10-ellipseList.txt as well as the corresponding images given my the *path* in those files\ as well as the textdocument you want to read the data from

Outputs:

traingDataSize * 900 dimensional hog-vector of the image. currently not saved to the trainingData folder, because it's processing just one textdocument at a time

pca s-sne.py

This is used to generate pca, s-sne as well as a pca-s-sne image for showing the distribution of the data

Requirements:

trainingData folder with trainingDataHog-files

Outputs:

A diagram, how the data is distributed / separated

SVM's.py

This is used for training the logistic regression (either via lagevin dynamics or SGD) and the SVM's with different kernels (lin, rbf, poly, sigmoid)

Requirements:

- trainingData/traingDataComplete - folders
- trained-models/modelName (for a pretrained model)

Outputs:

the accuracy of the models, as well as the model stored in a trained-models/file

loadImageAndCrop.py

This is used for cropping the image to the size we want, as well as putting pixels in the corners, if a face is on the corner of an image and saving it to trainingDataComplete

> **Requirements:**
>
> textdocuments of faces with labels, as well as the images
>
> **Outputs:**
>
> returns cropped images, which might differ, if the face was at the edge of an image

put data together.py

This is mainly used because the whole training data gets saved in the normal folder. Therefore it gets stored in the *trainingData* folder and also multiple files get merged to just one to have more data in one file to organize the space a bit better

Folders

- trained-models
  - trained-linear-SVM
  - trained-poly-SVM
  - trained-rbf-SVM
  - trained-sigmoid-SVM
- trainingData
  - trainingDataComplete-folder-X-to-X
  - trainingDataHog-folder-X-to-X

Learning Rate

If the learning rate is to high, it doesn't converge to the minimum

Retrain the model

just set the newDataset parameter in the init file to *True* and afterwards the functions will be trained !might take a while!

Why my program is split in many small parts

The reason is, that I'm using my cpu. therefore I had to be able to program and let some part of the program run at the same time.

accuracy:

```
    - logReg: -3.8881760566118436e-05
    - linearSVM: 99.93728372021727
    - rbfSVM: 99.95991290486299
    - polySVM: 99.91998133676822
    - sigSVM: 99.72075119566051
    - LDA: -3.8881760566118436e-05
    - cnn: 98.8629529048119
> The accuracy of LDA might be so low, because I didn't use the True inverse of the matrix (because sometimes the determinant of the covariance matrix is 0)
> my accuracy of the logReg might be so low, because my loss function might not be good enough o I didn't use enough training iterations. But for smaller datasets, the accuracy is better.
> Futhermore because of **rounding errors in determining the accuracy of the models**, the value got negative
```

How to use my program?

load images and crop them according to how it is described in the slides

change the path_to_FDDB variable according to how your filestructure is. the numbers 1, 4 in *createTrainingDataFomFolders(1,4)* mean, that all folders from 1 to 4 are loaded, copped and saved

generate Hog features

comment *getHogFromPyramid* out and instead uncommend hogFromFolder() change the *path* and *labels* variables according to the file The *output_path* variable is used to name the outputh path, with the lable attatched

training the models.

The *trainingDataFile* as well as the *Files* are defining the path to the trainingData. In the method *initEverything* are the models already predefined.

> The *loadModel* function takes many parameter. A name SVM,LogReg,LDA; a kernel, if one can be choosen [linear | rbf | poly| sigmoid] and in case of the *LDA* even if it should be trained with lagevinDynamics. Last but not least, there is a parameter, if the model should be trained on a new Dataset, if no is chosen, it loads an pretrained model.

cnn

with the variables *tcn* and *p*, a path to the trainingdata can be choosen **IMPORTANT: THE CNN IS TRAINED NOT TRAINED WITH HOGDATA** else it wouldn't work. It gets trained via just preprocessed images by the loadImagesAndCrop function. Without a graphics card, it takes a long time to compute.

Testing the models with an own picture

firstFirst run createPyramid.py on the file, be shure, that the variable *path_to_image* is correct. After running this code, hogFeatures.py should be run, but this time the *getHogFromPyramid()* function with the right path. Afterwards run the faceDetection. I use a pyramid sliding window, but this does not work as good as expected. Because it's not using all the different sices and all the different pixel, because my laptop couldnt handle it