Facutly of information technology
Brno University of Technology

SUR – Machine learning and recognition
Target person detector

Šimon Sedláček (xsedla1h)
Marek Sarvaš (xsarva00)
Dávid Špavor (xspavo00)

25/04/2020

Content

1 Prerequisites **2**

2 Usage **2**

2.1 Specifying the paths to data directories 2

2.2 Classification output 2

3 Image classifier **3**

3.1 PCA/LDA gaussian classifier 3

3.1.1 The idea behind model 3

3.1.2 Data augmentation and preparation 3

3.1.3 Model review 3

4 Speech classifiers **4**

4.1 Gaussian Mixture Model classifier 4

4.2 CNN speech classifier 5

4.2.1 Data preparation 5

4.2.2 The model 5

4.2.3 Model review 6

5 References **7**

Preface

Our project consists of three classifiers, two of which are meant for audio classification and the other one for images. We have chosen to implement three methods, where each have some advantages and disadvantages.

1 Prerequisites

The classifiers are written in Python 3.7. We also used some functions from the ikrlib.py module but were forced to alter some of the functions in order to be able to use them in Python 3. Our altered version of ikrlib.py is present in the SRC/ folder. As for the libraries used, please make sure to install the latest versions of the following python libraries by executing this command:

```
pip install numpy scipy imageio tensorflow keras -U
```

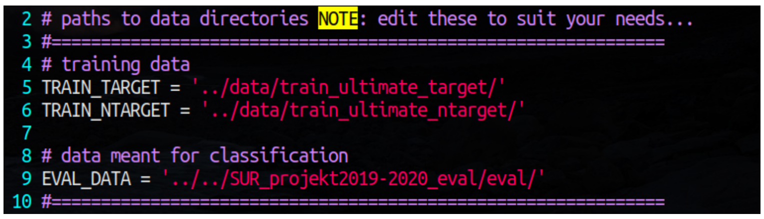
2 Usage

The main classifier modules are named:
– `speech_GMM.py`
– `speech_cnn.py`
– `image_pca_lda_gauss.py`
The classifiers can be run by executing:

```
python3.7 %classifier_name% [--train] [--eval]
```

2.1 Specifying the paths to data directories

In the header of each of the main classifier modules, there is a section which contains three constants that need to be edited before using the classifier. There are two constants used to specify the path to the training data: ‘`TRAIN_TARGET`’ and ‘`TRAIN_NTARGET`’. The other constant, ‘`EVAL_DATA`’ is used to specify the path to the data that is to be classified.



```
# paths to data directories NOTE: edit these to suit your needs...
#=============================================================
# training data
TRAIN_TARGET = '../data/train_ultimate_target/'
TRAIN_NTARGET = '../data/train_ultimate_ntarget/'

# data meant for classification
EVAL_DATA = '../../SUR_projekt2019-2020_eval/eval/'
#=============================================================
```

Figure 1: Specified paths to data directories

2.2 Classification output

When classifying data, each of the classifiers produces a *.txt file which contains the results of the classification process. The names of these files for each classifier are as follows:
– `speech_GMM.py` → `speech_GMM.txt`
– `speech_cnn.py` → `audio_convolutionalNN.txt`
– `image_pca_lda_gauss.txt` → `iamge_pca_lda_gaussian.txt`

3 Image classifier

3.1 PCA/LDA gaussian classifier

File: `image_pca_lda_gauss.py`

3.1.1 The idea behind model

Applying LDA to our image dataset was the first thing we wanted to implement just to see where this approach would take us and whether the results would at all be plausible. Immediately, we realized this approach would not work because of the size of our dataset - the dimensionality of our data was too large and we had too few images to work with.

To tackle this problem, we decided to try to use PCA for dimensionality reduction, which should enable us to then use LDA on our dataset - and the results were quite surprising. The LDA was able to spread our training data so that it was perfectly splittable in that one resulting dimension, too well actually - the log-likelihoods we were getting from our resulting gaussian distributions were too negative and the threshold we would have to set in order to classify any data at all would change drastically with the smallest change of the classifier parameters.

3.1.2 Data augmentation and preparation

We decided to implement some data augmentation methods (including rotations, zoom, shifts, lighting shifts,...) and in order to further reduce the dimensionality of our data, we would crop each image by 13 pixels from each side (resulting in a 54*54 image). This was done to throw away some of the "redundant" information in the image - mostly the surroundings of the classified face. This is by no means a very elegant way to get rid of some of the redundant information and we are aware of it, however we decided to keep it in in the end as it was giving us slightly better results on the test data.

The score for each picture is calculated as the difference of log-likelihoods that it fits to the target or non-target gaussian distribution. The spread of the score values is not that big so choosing a suitable hard decision threshold turned out to be a bit of a challenge - in the end we settled on the threshold value of 2.0 as it was giving us the best results on the evaluation data.

3.1.3 Model review

This approach turned out to give us some surprising results given its simplicity. A more suitable approach to image classification would surely be something in the realm of convolutional neural networks - which we tried to implement but regretfully with not much success as the model would overfit on the training data while training. Also, some of the methods regarding the data augmentation and preparation could be improved - most notably the hard-trimming of the picture edges is not a very consistent nor robust way of easing the classification process for the model.

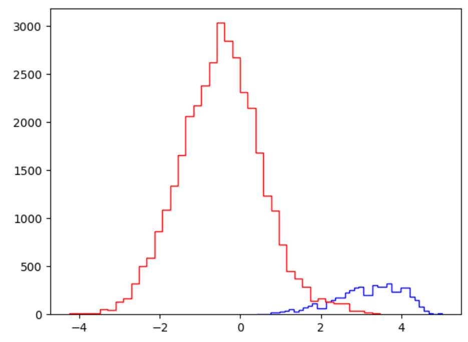


Figure 2: Image data dispersion after applying PCA and LDA

4 Speech classifiers

4.1 Gaussian Mixture Model classifier

File: `speech_GMM.py`

After applying PCA and LDA on audio with not so great results we decided to use GMM method instead. Using `train_gmm` and `logpdf_gmm` from `ikrlib.py` GMM showed better results on the first try with no dimension reduction. You can see in Figure 3 that after applying PCA and LDA on voice dataset, data dispersion is not great, therefore classification wouldn't be so accurate.

First update was on how many components should each class (target/non-target) have. Starting on small numbers like 3 went up to 40 while reading some articles on optimal number of components, best and most consistent results in regard to given training time was around 20. This number of components lead to very good results for that small data set. Log-likelihood scores sometimes more inconsistent but had nice values and nothing extreme. However to obtain low miss rate threshold had to be set a bit higher around 200.

Secondly we decided to cut off initial 100 frames to get rid of redundant silence from the beginning of every record but this alone had no significant impact. Huge difference was made by removing silence from whole record. This was accomplished by calculating mean record energy from 1st mfcc feature then compare mean record energy with newly calculated energy from 20 frames. If the mean energy of current 20 frames was lower than mean record energy the 20 frames where removed.

After this trained GMM model was more consistent with higher hit rate on such small train data set. When GMM model was trained on `target_train` and `non_target_train` data sets, we get 100% hit on `target_dev` and 98% hit on `non_target_dev` with really nice log-likelihoods where threshold is set only to 100 to make these results.

Further improvements may be bigger training data set or usage of other mfcc features.

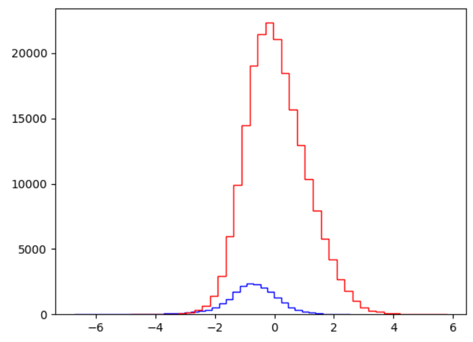


Figure 3: Voice data dispersion after applying PCA and LDA

4.2 CNN speech classifier

File: `speech_cnn.py`
The second audio classifier we implemented utilizes convolutional neural networks to classify the speaker. We used the Keras API for tensorflow to implement this neural network.

4.2.1 Data preparation

The first step before using the recordings to train the network or classifying them is to prepare our data. The main aspect of this process is the removal of silence in the recordings. The silence removal is performed on each recording separately.

First, we cut off the first 100 frames of the recording to get rid of the occasional 'pop' at the start as well as the initial silence. Then, we calculate the mean value of the "zeroth" MFCC coefficient for the recording. After that we scan the recording ten frames at a time and we calculate the mean value of the same MFCC coefficient on these ten frames. If this value is smaller than the overall mean value of the recording, those ten frames are considered as silence and are discarded.

4.2.2 The model

The main idea behind our model is that even though the MFCC frames already contain speech features over some time period, taking multiple of these frames could tell us more about the way the target speakers speak and maybe how they pronounce things - basically we wanted to take advantage of the fact, that the frames are correlated.

Because of this, we take each recording and create groups [1] of 50 frames each with an overlap of 10 frames. We are aware, that 50 frames represent a rather large time period but given the amount of training and test data we had, these parameters produced the best results. These recording segments are what our network is trained on and it also uses them to classify the speakers.

The network itself consists of two convolutional layers and one dense layer before the output layer, plus some additional max-pooling and dropout layers. Details of the model can be viewed in the source code.

When classifying the recording, we simply look at the amount of these 50-frame segments that get classified as the target speaker and otherwise. The overall score of the recording is simply the difference between these two values. We ended up setting the threshold for the hard decision to -20 as it was giving us the most consistent and positive results on our test data.

[1]https://arxiv.org/pdf/1702.02289

4.2.3 Model review

The challenging thing with neural networks is their craving for training and evaluation data. We found it quite difficult to make alterations in the model parameters because we only had limited amount of data - also because of this, the result of the training process can be inconsistent and random. We had to retrain the network multiple times in order to get decent evaluation results.

I am quite sure that, apart from the parameters of the neural network itself, the most important parameter is the size of the feature groups. The 50 frames we are using now can seem like a stretch but it simply performed the best while evaluating the model. I'm sure, given more training data, something like the segment size of 10 frames would work much better as the network would be classifying much smaller speech fragments.

Also, I think that trying a different model like an LSTM instead of a convolutional network could improve the results as recursive neural networks can better suited for continuous types of data like text and speech.

5 References

SUR lectures
GMM optimal components
GMM