Exercises for
Pattern Analysis
Amir Davari, Dalia Rodriguez Salas, Johannes Bopp, Lina Felsner
Assignment 4, published on July 8, 2019

General Information:

Lecture:	Mon 16:15h – 17:45h and Wed 10:15h – 11:45h (H16)
Exercises:	Tue 14:00 – 15:00, 15:00 – 16:00 (02.151-113) and
	Thu 16:00 – 18:00 (0.01-142, 00.156-113, 02.151b-113)
Certificate:	Oral exam at the end of the semester
Contact:	amir.davari@fau.de & dalia.rodriguez@fau.de

HMMs: Biometric Signature Verification

Exercise 1 Preliminary Remarks

In this exercise, we aim to (partially) reproduce a work on biometric signature verification. During training, the task is to fit a Hidden Markov Model to handwritten signatures of a person. During testing, the task is to determine for a new signature whether it belongs to that person or to someone else who imitates that signature.

This exercise loosely builds on a paper by Fierrez *et al.*, "HMM-based on-line signature verification: Feature extraction and signature modeling", Pattern Recognition Letters, vol. 28, Aug. 2007, pp. 2325–2334. The paper is also available in studOn.

Exercise 2 Using hmmlearn

You will likely need to locally install `hmmlearn`. To do so, please run

```
pip install --user setuptools
pip install --upgrade --user hmmlearn
```

for integration into python 2, or

```
pip3 install --user setuptools
pip3 install --upgrade --user hmmlearn
```

for integration into python 3. Note that both pip and pip3 will write about 190MB in your home directory (i.e., in `/.local`) — if you are short on space you can check your available space with `quota -v` or analyze the use of your space, with a call from within your home directory to `du | sort -n -r | head -30` or variants thereof.

Exercise 3 Dataset

We will use some writer data from the "mobisig" dataset. You can access this data from any computer of the Computer Science Cip pool at `/proj/ciptmp/sichries/pa_mobisig`. Let us use signatures from up to 10 writers, denoted as `user_0` to `user_9`. Each signature is stored in an individual CSV file. "Original" signatures are from the writer herself or himself. "Imitated" signatures are from an imposter who only pretends to be that writer, i.e., they should not be considered for training.

Exercise 4 Feature Extraction and Processing

The purpose of this task is to prepare the signature data for classification in a HMM. Each signature file provides several feature information: the time stamp of a stroke, x- and y-position, pen pressure p, velocity in x- and y-direction, and acceleration in x- and y-direction, and other information that we will not consider.

The task is to use the available features to compute the seven-dimensional feature vector from Eqn. (3) of the paper. To this end, you will notice that x, y, p are identical to the features in the paper. The remaining four features are relatively straightforward to calculate with the equations in the bullet list above Eqn. (3).

There is one notable caveat for the feature computation: if during your calculation a division by 0, or a log of 0 occurs, the result in that dimension will be `nan` instead of a numeric value — this can really screw up your result. One standard approach in pattern recognition to deal with such issues is to add a small epsilon to "dangerous" calculations.

The final feature vector should be mean-free and with unit variance (confer Sec. 2.5 in the paper).

We found in our own experiments that you can omit the feature rotation (Eqn. (2)) and a resampling of the timestamps to equidistance. These two steps would only be required if our goal would be to achieve a performance that is competitive to what the authors reported. Don't worry about this for now, let us just aim to be better than guessing.

Exercise 5 HMM Preliminary Remarks

Have a look at the `hmmlearn` API documentation and tutorial.

Considering the HMM that we discussed in the lecture, and considering our feature set, you will notice a difference: in the lecture, we assumed that an observation o_i and the associated output probability $b_{s_j}(o_i)$ of state s_j is a categoric value. In our dataset, we do not have such clearly defined categorial values. To accomodate this, `hmmlearn` offers that each state internally holds a parametric probability density estimate of "its" features. In other words, a state fits during training a Gaussian density or a Gaussian Mixture model to a subsequence of the features. During testing, it evaluates the probability that a sample at time step t is drawn from that distribution.

Therefore, you should either use `hmm.GaussianHMM` or `hmm.GMMHMM` from the `hmmlearn library` to achieve this internal representation. Both of these models can be constrained in their complexity. We made good experiences with a commonly used constraint, namely to set the covariance matrices to be diagonal matrices with the (preset) parameter `covariance_type='diag'`.

We also made good experiences with constraining the model to a left-right HMM (this is also what the paper does). This can be done by pre-setting the transition matrix to a left-right model, as shown in the `hmmlearn` tutorial. Note that if you do this, then you have to prevent that the transition matrix is freshly re-initialized by the library, i.e., you have to omit `'t'` from the parameter `init_params`.

The same applies to the vector of starting probabilities: the paper sets these to $(1, 0, \ldots, 0)$. Do this analogously (again, please refer to the tutorial), and drop its initialization from `init_params`.

Particularly when experimenting with the more complex `hmm.GMMHMM`, we found that it helps convergence if transition probabilities and starting probabilities are completely excluded from training, i.e., also not refined later. To achieve this, drop `t` and `s` from the not-so-excellently-named parameter `params`.

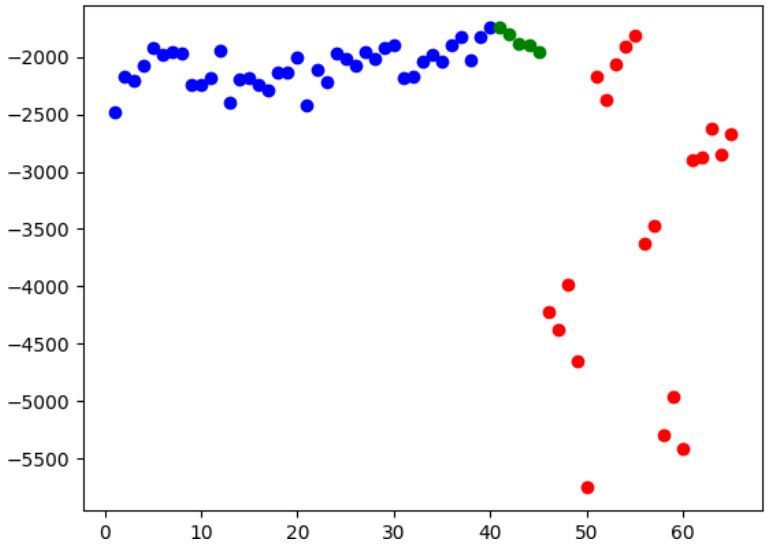
(as a sidenote, you probably see from these caveats that training can be a bit brittle with increasing model complexity. This is a general rule in statistical model fitting — which is why so many Ph.D. students are spending so much time on deep networks, our most complex models to date.)

Exercise 6 HMM Results per Signature

The task here is to choose a `GaussianHMM` or a `GMMHMM` and fit it to the data. To do this, split the 45 `original` signatures from a writer directory into training and testing. The training should contain at least 25 signatures to prevent overfitting. Start with a modest number of states, e.g., 4 or 5, to get some first results.

The paper proposes to statically segment the input sequency into H segments (Eqn. (9)), where H is the number of states in the model. Additionally, they pre-initialize their `GMM` model with the k-means algorithm. Ignore this! You can achieve some reasonable results by just feeding whole signatures to the training process. To this end, consult the tutorial: you will need a 2-D list where one row contains the seven features per time step. All time steps and multiple feature sequences are just concatenated.

Also ignore the threshold in Eqn. (11). Instead, compute the log-probabilities individually for each signature in one user directory using the method `score` from `hmmlearn`. Create a plot with the matching score per training signature in blue, each original testing signature in green, and each imitated signature in red. It should look like one of our example results, here computed on 40 training samples, 5 testing, and 20 imitated:



In this picture, "success" is defined as having a set of imitated signatures with lower scores than the green or blue signatures: if we had a good threshold, we could reject the imitations!

Exercise 7 HMM Model Selection

Let us now mimic the model selection problem. To this end, switch over to a `GMMHMM` (unless you already did so). The two interesting quantities of interest shall be the number of states H and the number of mixture components M in

the GMM representation.

Our goal is to obtain a list of results that are a simplified version of Tab. 1 in the paper. As a simplification, we will not report error rates. Instead, calculate the average matching scores on the test set for original and imitated signatures. If the average score for the originals is larger than for imitated, we count it as success. Which combinations of H and M work better, which work worse for your implementation?

Questions and demonstration of your results:

Considering that we are approaching now the end of the semester, you can use the exercises until (including) Thursday July 18 to ask questions.

The studon registration for demonstrations opens on July 9 at 10am.

You can book a date for the demonstrations of results at July 16, July 18, July 23 or July 25. Thus, there is an overlap of one week between questions and demonstrations where you can either decide to finish via demonstration, or to keep going by asking questions. Whenever you successfully demonstrated your results, you successfully passed! We hope that you enjoyed the exercises, and that they were useful in deepening your understanding of the material!