Sheet04 - Exercise for MA-INF 2218 Video Analytics SS18
Submission on 15.06.2018
Video segmentation with HMM

In this assignment we will compute the most probable sequences for a video input and segment video input into several temporal chunks using Viterbi decoding. We will furhter train a GMM/HMM to improve recognition on the given sequences. To segment video sequences into multiple chunckswe need a list of occuring action classes (here: SIL (the silence/background class), take_bowl, pour_cereals, pour_milk, stir_cereals, take_cup, pour_coffee, spoon_powder, stir_milk). Each action class is represented by one HMM. A video sequence is made up a combination of multiple HMMs (e.g. making a coffee = SIL, take_cup, pour_coffee, pour_milk, SIL).

Exercise 1.1
In the folder *data/exc1* you are given a set of pretrained HMMs, consisting of a HMM definition file in hmm_definition.vector, showing the number of states for each HMM. The class mapping can be found in hmm_definition.dict. The hmm_state_definition.vector maps the classes to the respective states. For all HMMs, we assume a left-to-right feed-forward model, allowing only self-transistion or transitions to the next state. The transition probabiltities are 0.9 for self-transistions and 0.1 for transistions to the next state. The observation probablitites are modeled by a Gaussian mixture for each state described by the mean in GMM_mean.matrix and variance in GMM_var.matrix. The input data are the framewise features of three videos (making cereals, making coffee, making choclate milk). Features (DT+FV+PCA, 64dim) are stored in the three *.npy files. The files containes three feature sequences, in which each frame is represented by one 64 dimensional descriptor. There are three possible paths for those videos given by the three grammar files *.grammar.
Inference: Write a programm to compute the maximum probability of the three videos for the three given paths (9 results overall). The alignment of the input frames to the HMM states given by

$$\arg\max_{s_1,\ldots,s_T} p(s_1,\ldots,s_T|\mathbf{x}^T) = \arg\max_{s_1,\ldots,s_T} \prod_{t=1}^{T} p(x_t|s_t)\cdot p(s_t|s_{t-1}), \tag{1}$$

where each s_t is a state of the original action class HMMs, $\mathbf{x}^T$ is the input feature sequence of each video $\mathbf{x}^T = \{x_1,\ldots,x_T\}$ with $x_t \in \mathbb{R}^m$ ($m = 64$ in our case) as feature vector at frame t. To compute $p(x_t|s_j)$, the observation probability is modeled by a multivariate Gaussian distribution defined as

$$p(x_t|s_j) = \frac{1}{\sqrt{(2\pi)^l|\Sigma_j|}} e^{-\frac{1}{2}(x_t-\mu_j)^\intercal \Sigma_j^{-1}(x_t-\mu_j)} \tag{2}$$

with l denoting the dimension of the input sequence $\mathbf{x}$, μ_j the l-dimensional mean vector, and Σ_j the $l \times l$ covariance matrix of the Gaussian model for state s_j. Since we model the observation probability by a single component Gaussian, the observation probability is parametrized only by the mean μ_j and the covariance matrix Σ_j corresponce to the variance of the related feature distribution. The above maximization can be solved efficiently using the Viterbi algorithm.

Output the negative loglikelyhood probablitites for all paths for each video. Which path is the most probale one for each video?

Hint: Using negative loglikelyhood probablitites during computation helps to avoid numerical instabilities. To faciliate inference, consider each path as one 'big' HMM. You can save time by precomputing observation probabiltities for all frames over all states in advance. *(8 Points)*

Exercise 1.2 For each sequence *.npy compute and output the framewise states as they occur in the most probable path. Map the states to their respective action classes. Evaluate the accuracy of your programm by comparing the output to the ground truth actions in *.gt . Output the mean-over-frames accuracy ($MoF = n_{correctFrames}/n_{allFrames}$). *(4 Points)*

Exercise 2 You are now able to decode sequences with respect to states. You will now run a simple Viterbi training for a set of HMMs. The training data consists of a set of video based features in the folder *data/exc2* with respective state labels. For initialisation, we start with a uniform segmentation of the actions according to the number of HMM states that can be found in *_initStates.npy. The overall procedure is as follows:

- Initialize GMM/HMM model with uniformly segmented states.
- Decode training samples and ouput the new aligment.
- Updating GMM/HMM parameters according to new states.

The training is done by iterating between steps two and three. To initialize the HMMs, please use the segmentation provided by *_initStates.npy to compute parameters for single Gaussian. Repeat steps two and three 5 times. After each iteration, run a decoding of the test data (from Exercise1) on the new model and report MoF accuracy. What do you observe? Write a short comment. *(8 Points)*