Hidden Markov Models
Speech Communication Laboratory

Franz Pernkopf
Graz University of Technology, Laboratory of Signal Processing and Speech Communication
Inffeldgasse 16c, 8010 Graz, Austria

March 9, 2016

Please provide the Name and Matrikelnummer of each team member on the report.

1 HMMs

Go through the tutorial "Hidden Markov Models" and download the accompanying MATLAB programs and data.

1. Load the Hidden Markov Models (HMMs) $\mathbf{\Theta}_1, \ldots, \mathbf{\Theta}_4$, and make a sketch of each of the models with states and transition probabilities. The parameters of the Markov models and of the Gaussian emission pdfs are stored in the file `data.mat`. Each HMM $\mathbf{\Theta}_i$ is stored as an object, e.g., `hmm1`, with fields `hmm1.trans`, `hmm1.pi`, `hmm1.means`, and `hmm1.vars`. The `trans` field contains the transition matrix $\mathbf{A}$, and the `pi` field the prior probability vector $\boldsymbol{\pi}$. The `means` field contains a matrix composed of mean vectors of the Gaussian emission pdfs, where each column of the matrix corresponds to one state of the HMM (to access the mean vector $\boldsymbol{\mu}$ of, e.g., the second state from `hmm1` use: `hmm1.means(:,2)`. The `vars` field contains a 3 dimensional array of covariance matrices, where the third dimension corresponds to the state (e.g., to access the covariance matrix $\mathbf{\Sigma}$ of state 1 use `hmm1.vars(:,:,1)`).

2. Generate samples from the HMMs `hmm1`, `hmm2`, `hmm3`, and `hmm4` and plot them with `plotseq` and `plotseq2`. In the resulting plots, the obtained sequences are represented by a yellow line where each point is overlaid with a colored dot. The different colors of the dots indicate the state from which a particular sample has been drawn.
```
>> % Example:  generate a sequence of length T (e.g.  100) from HMM1
>> [X,ST] = sample_ghmm(hmm1,T)
>> plotseq(X,ST) % View of both dimensions as separate sequences
>> plotseq2(X,ST,hmm) % 2D view with location of Gaussian states
```

   Draw several sequences for each HMM and compare. Compare the MATLAB figures with your sketch of the models and add (some of) them to your homework elaboration. Moreover, explain: What is the effect of the different transition matrices of the HMMs on the sequences obtained? Hence, what is the role of the transition probabilities in the HMM?

3. Pattern recognition with HMM's: Classify the sequences X_1, X_2, X_3, X_4, given in the file `Xdata.mat`, in a maximum likelihood sense with respect to the four Markov models $\mathbf{\Theta}_1, \mathbf{\Theta}_2, \mathbf{\Theta}_3$, and $\mathbf{\Theta}_4$. Use the function `loglik_ghmm` to compute the log-likelihood $\log p(X_i|\mathbf{\Theta}_j)$ of a sequence X_i with respect to a HMM $\mathbf{\Theta}_j$. Store the results in a matrix (they will be used in the next section).

```
>> load Xdata
>> % Example:
```

```
>> logLike(1,1) = loglik_ghmm(X1,hmm1)
>> logLike(1,2) = loglik_ghmm(X1,hmm2)
...
>> logLike(i,j) = loglik_ghmm(Xi,hmmj)
...
```

Filling the `logLike` matrix can be done automatically with the help of loops:

```
>> for i=1:4,
     for j=1:4,
      stri = num2str(i);
      strj = num2str(j);
      eval([ 'logLike(' , stri , ',' , strj , ')=...
            loglik_ghmm(X' , stri , ',hmm' , strj , ');' ]);
  end;
end;
```

You can find the maximum of each row in the matrix with the MATLAB function `max`:

```
>> for i=1:4;
    [v,index] = max(logLike(i,:));
    disp(['X',num2str(i),' -> HMM',num2str(index)]);
end
```

Sequence	$\log p(X_i\|\mathbf{\Theta}_1)$	$\log p(X_i\|\mathbf{\Theta}_2)$	$\log p(X_i\|\mathbf{\Theta}_3)$	$\log p(X_i\|\mathbf{\Theta}_4)$	Most likely model
X_1					
X_2					
X_3					
X_4					

4. Viterbi decoder: Write a MATLAB function `[loglik,path] = vit_ghmm(data,HMM)` to implement the Viterbi decoding algorithm to find the most likely state sequence Q for a given observation sequence X_i for HMMs with Gaussian emission probabilities. Use the function `mk_ghmm_obs_lik` to calculate the observation probabilities (Gaussian) for each state and time step. Please perform the calculations in the log domain, where the multiplications of the probabilities for the parameters δ and ψ become additions. So you can prevent numerical instabilities for long sequences.

```
[loglik,path] = vit_ghmm(data,HMM)
% Compute the path and the log likelihood of a given model and
% observation sequence
%
% INPUT
%      data ... containing a sequence of observation
%         size(data)=[Number of features of each obs., length of sequence]
%      HMM is an object containing
%       HMM.trans = state transition probability matrix;
%       HMM.pi = prior probability vector;
%       HMM.means = mean vectors of Gaussian emission pdf for each state;
%       HMM.vars = covariance matrices of Gaussian em. pdf for each state;
% OUTPUT
%       loglik ... log likelihood of the most likely path for the data
%       path ... most likely path
```

5. Use your function `vit_ghmm` to compute the most likely paths for the sequences $X_1, \ldots X_4$ with respect to each model $\boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_4$. Also compute the log-likelihoods $\log p^*(X_i|\boldsymbol{\Theta}_j)$ along the most likely paths found by the Viterbi decoder. Note down your results below. Compare with the log-likelihoods $\log p(X_i|\boldsymbol{\Theta}_j)$ obtained in the previous section with the function `loglik_ghmm(...)`:
   `>> diffL = logLike-logLikeViterbi`

   Log-likelihoods along the best path:

Sequence	$\log p^*(X_i\|\boldsymbol{\Theta}_1)$	$\log p^*(X_i\|\boldsymbol{\Theta}_2)$	$\log p^*(X_i\|\boldsymbol{\Theta}_3)$	$\log p^*(X_i\|\boldsymbol{\Theta}_4)$	Most likely model
X_1					
X_2					
X_3					
X_4					

   Difference between log-likelihoods of a sequence given the model and log-likelihoods along the best path found by the Viterbi algorithm, $\log p(X|\boldsymbol{\Theta}_i) - \log p^*(X|\boldsymbol{\Theta}_i)$:

Sequence	HMM1	HMM2	HMM3	HMM4
X_1				
X_2				
X_3				
X_4				

   Is the likelihood along the best path a good approximation of the real likelihood of a sequence given a model ?

2 HMMs applied to speech

HMMs with Gaussian mixture emission pdfs should be trained and used for recognition of utterances of English digits from 'one' to 'five'. Go through the tutorial "Mixtures of Gaussians" and download the accompanying MATLAB programs and data.

1. Load the signals into MATLAB using `load digits`, and play them with the MATLAB functions `sound` or `wavplay`, e.g., `sound(three15)`.

   Process the speech signals to get parameters (features) suitable for speech recognition purposes. For the signals loaded from `digits.mat` this is done using the function `preproc()` (without arguments). This function produces a cell array `data_N{}` for each digit `N` holding the parameter vectors (mel-frequency cepstral coefficients, 12-dimensional) for each training signal (e.g., `data_1{2}` holds the sequence of parameter vectors for the second example of digit 'one'), as well as a cell array `testdata_N{}` for each digit `N` for each test signal. `preproc()` uses functions that are part of VOICEBOX, a MATLAB toolbox for speech processing.

   The sequences of parameter vectors have different length (as also the speech signals differ in length!), that is why we can not store all sequences for training or testing in one array.

2. Train one HMM for each digit. Training of HMM parameters (emission pdfs using the EM algorithm, as well as prior and transition probabilities) is done using the function `train`:
   `>> [HMM] = train(data,K,Ns,nr_iter)`

where `Ns` denotes the number of HMM states, `K` the number of Gaussian mixtures in the emission pdfs, and `nr_iter` the number of iterations. The function trains left-to-right HMMs with covariance matrices of the Gaussian mixtures in diagonal form.

3. Determine the recognition rate on the test signals. To test the model use the function `recognize` as described in the tutorial, e.g., for digit 1:
   `[r1,L1,p1] = recognize(testdata_1,HMM_1,HMM_2,HMM_3,HMM_4,HMM_5)`

4. In your report note down your chosen settings, intermediate results and considerations, and the recognition results (recognition rate for each digit, and for the whole set). Which digit seems more easy to be recognized correctly? Which digits get easily confused during recognition? Use different values for the number of states (`Ns=2...5`) and the number of Gaussian mixture components (`K=1...3`). How do these numbers affect the computational effort for training and the recognition rate? Which values for `Ns` and `K` do you think are optimal? Please present your recognition results (for each digit and for the whole data set) as well as the time for training the models as tables/figures (depending on the choice of `Ns` and `K`). For measuring the training time use the matlab commands `tic` and `toc`.

5. Why do we use diagonal covariance matrices for the Gaussian mixtures? What assumption do we take, if we do so? Please modify the function `train` to train left-to-right models with a full covariance matrix. Again, do experiments similar as in task 4 for `Ns=2...5` and `K=1...3` and present the tables for the recognition rate and the computational costs for training. Give also a table for the number of parameters we have to train in this case compared to the left-to-right HMM using diagonal covariance matrices for the Gaussian mixtures (as in task 4). Is there a connection between the number of parameters we have to train, the number of training samples we have available, and the recognition performance? Please present a short discussion on that in your report.

6. What kind of model do we have if we have just `Ns=1` state in our HMM. Please give the equation $P(X|\mathbf{\Theta})$ for this model in terms of $\mathbf{\Theta} = \{\pi, \mathbf{A}, \mathbf{B}\}$.