Laboratory exercise for hidden Markov models

As an integral part of this laboratory exercise, an experiment with a hidden Markov model is performed (simulated and analyzed), and calculations are performed using the HMM Toolbox for Matlab. Before performing the exercise yourself, it is necessary to study the instructions in the document *"Guidelines for working with Markov models in Matlab"*, and especially those parts that relate to specific examples of working with the HMM Toolbox. The individual tasks from this exercise will follow the content and the chapter structure from these guidelines.

Personalization of tasks for laboratory exercise, method of submitting solutions for automatic verification and instructions for preparation of the final exercise report

Moodle is used to automatically check and score your results of performing this laboratory exercise in Matlab. Although all students solve the same HMM model experiment, each student will access their personalized version of the task through Moodle.

In addition to typing the required numerical solutions into Moodle for each sub-task for the purpose of automatically checking your results, the entire process of solving this lab exercise needs to be documented through a report and a Matlab script. The report is submitted through the Moodle environment in .pdf format, and separately, the Matlab script is submitted as an attachment to the report in .zip format. These need to be made as follows.

When creating a report for each sub-task it is necessary:

1) indicate the sub-task (appropriate numbering),
2) copy personalized sub-task text from the Moodle environment,
3) copy from Matlab a snippet of program code used to solve the sub-task,
4) highlight the results of code execution,
5) answer questions, comment results, etc. (label REPORT) depending on the text of the sub-task.

Matlab script should contain:

1) appropriate initializations of all necessary variables, which will depend on the given parameters of the model, more precisely specific issues in individualized Moodle sub-tasks,
2) all used Matlab commands for calculating the solution of individual sub-tasks,
3) detailed comments on the sections of code used.

As an example of a properly commented Matlab script, its common beginning is shown below, which includes the initialization of the transition probability matrix and the initial state probability vector in accordance with the described experiment and guidelines:

```
clear;
pack;
addpath(genpath('HMMall')) % Add a path to the function library

% ============================================================
% HMM model state designations
```

```
% We have three biased dice of which we always roll only one.
% Model states are indices of the biased dice used

% Initial state probability vector (for t=1)
% determined by rolling an unbiased dice:
prior0=[
    1  % First dice  (if '1' is observed)
    2  % Second dice (if '2' or '3' is observed)
    3  % Third dice  (if '4', '5' or '6' is observed)
]/6;

% Number of states of the HMM model
Q=size(prior0,1);

% ------------------------------------------------------------------
% Probability matrix of state transitions
%
% a11 a12 a13
% a21 a22 a23
% a31 a32 a33

% For a stochastic state change experiment, parameter
% M is used to define the transition probabilities in
% new state in transition matrix A, where states are necessary
% changed cyclically due to the forced structure of the transition matrix.
M= ...;  % Here you define M from your personalized task.

% Create a state transition probability matrix
% (with a cyclic state change structure, because
% transitions 1-> 3, 2-> 1 and 3-> 2 are prohibited)
transmat0=[
M-1 1 0  % P(1|1) P(2|1) P(3|1)
0 M-1 1  % P(1|2) P(2|2) P(3|2)
1 0 M-1  % P(1|3) P(2|3) P(3|3)
]/M;
```

Description of a random experiment used in the exercise

The selected stochastic experiment is based on throwing three biased dices with possible throwing outcomes from 1 to 6. These dice are "tuned" in such a way that the first dice will, on average, give the number "1" in half of all throws, and the second dice will similarly give the number "3" in half of all throws, while the third will give the number "5" in the same way in half of all throws. The probabilities of the remaining throwing outcomes are much smaller, but they are not equal, but depend on the individual dice for each of the remaining 5 possible numbers. The outcome of the dice roll is visible to viewers, while what is not visible to them is the information which of the three biased dice was used in each roll, so the index of the dice thrown (1st, 2nd or 3rd dice) represents the hidden state of this model.

Changes of hidden states are defined by the transition matrix A of the model, with some transitions not allowed in this experiment, i.e., having zero probability. Specifically, if the first dice is used in the current roll, then the same dice, or second dice, will be used in the next roll, but the third dice cannot be selected. Analogously, if the model is in the state of throwing the second dice, it can remain in that state, or go into the state of the third dice, but it cannot return to the state of the first dice. Finally, when it enters the state of the third dice, it can cyclically transition to the state of the first dice, or remain in the existing state, but cannot

return to the state of the second dice. In conclusion, this model necessarily cyclically goes through its hidden states 1-> 2-> 3-> 1, with the possibility of remaining in the current state. The probability of maintaining the same state is (M-1) / M, while the probability of transition to the next cyclic state is 1/M.
REPORT: Sketch a state diagram for the model described in this way.

In order to define the initial state of this model, i.e. only for the first roll, an additional unbiased dice is used as follows: if the outcome of the roll of that unbiased dice is "1", the first biased cube will be used for the first roll, for outcomes "2" or " 3" the second dice will be used, while in the case of the remaining three outcomes," 4 "to" 6 ", the third biased dice will be thrown first.

Below are detailed descriptions of the sub-tasks you need to solve for your specific example with additional notes related to the preparation of the report. This description is general, while specific parameters' values and specific required results are defined in each student's individualized Moodle assignments. For example, in the personalized Moodle task of each student, different output probabilities of observation of all possible outcomes are defined for all three biased dices. Additionally, the parameter M will be different, which determines the transition probability of the model to the new state. Each student will also receive individualized sub-tasks of the task of this laboratory exercise, to which they will have to answer by entering the exact numerical value of the obtained solution in the appropriate field in Moodle exam, which will be explained in more detail later in this document.

Sub-task 1 - Complete definition of the HMM model in Matlab

Based on the specified observation frequencies of individual dice outcomes and based on the state retention parameter M of your Moodle task, it is necessary to append the Matlab script template to fully describe the described HMM model of this experiment including the probability matrix of output symbol observations.

Sub-task 2 - Determining the log-likelihood of the observation of given output symbol sequences for a given HMM model

In the second sub-task in Moodle, two specific observed sequences of output symbols (outcomes of throwing biased dice) are given. Insert these arrays into the Matlab script as the initialization of the corresponding two-row matrix (which you will also use in the following sub-tasks) and use the appropriate function to calculate the log-likelihood of the observations of both given sequences, given the model from sub-task 1. Enter the obtained log-likelihood of the first and second sequence in the required fields in Moodle.
REPORT: Can you explain the reason why the second sequence is less likely to be observed than the first, given the described model, by comparing the observed symbols? Describe in words. Calculate and type in Moodle how many times the second sequence is less probable than the first.

Sub-task 3 - Calculation of forward and backward probabilities for all hidden states of the model and all time moments of observation

For the first observed sequence from the second sub-task it is necessary to apply the algorithms: "Forward" and "Backward" and calculate the forward (alpha) probabilities and

backward (beta) probabilities for all three hidden states and all observation instances for the given model L (specified in the first sub-task). In order to verify your solution, the individualized sub-task in Moodle asks you to enter the calculated alpha and beta probabilities for some randomly chosen time moments and some randomly selected model states, so enter the required probabilities calculated by the *fwdback* function in the appropriate fields in scientific (exponential) number notation. Important: you must not activate probability scaling when calling a function, i.e. you must define…, 'scaled', 0) in a function call; as done in the example in the guidelines.
REPORT: Explain and show in the report how you can use the alpha probabilities from the last time step to determine the log-likelihood of the whole observation, or how you can use the calculated backward probability beta from the first time step for the same purpose, and compare the results with those from the sub-task 2 for the first observed sequence.

Sub-task 4 - Decoding hidden states using the Viterbi algorithm

Using the Viterbi algorithm (by using corresponding *viterbi_path* function), it is necessary to determine the most likely sequence of hidden states of the model only for the first observed sequence from the second sub-task. In order to verify the obtained solution, it is necessary to enter the decoded states of the model for the first three and for the last three time steps of the first observation in Moodle.

Sub-task 5 - Determining the log-likelihood of observations along decoded Viterbi paths

Repeat the determination of the Viterbi state sequence for the second observed series from sub-task 2 as well, and for both series calculate the log-likelihoods of the observation but only along the decoded "optimal" Viterbi paths.
REPORT: Compare the obtained results with those from sub-task 2 where the total log-likelihood for all possible hidden state paths was found.
The difference between log-likelihood across all paths and log-likelihood along the Viterbi path for both observed sequences should be found and entered in Moodle.
REPORT: What does the sign of these differences tell us? Discuss the results obtained in the report. Could you also calculate the likelihoods of the observation of complete observed sequences (in full length) along all possible individual paths of the state lattice, as described in the guidelines? If not, why not? Explain.

Sub-task 6 - Determining the likelihood of observations for the truncated sequence and finding the most likely individual state paths

For the first observed sequence from sub-task 2 it is necessary to determine the total likelihood of observation of the shortened sequence, i.e. only for the first four observed output symbols o1, o2, o3 and o4. For this purpose, you need to use the earlier solution from the third sub-task in which you have determined all the probabilities of the model, but for the complete sequence.
REPORT: Explain how you obtained the likelihood of observing a truncated output sequence. Enter in Moodle exam the value of likelihood (not log-likelihood!) in exponential notation of observing the first four output symbols.

Then determine the Viterbi path again, but for this truncated observation sequence, and calculate and enter in Moodle what proportion of the observation likelihood (normalized to 1) is achieved along the Viterbi path in relation to all possible state paths of the model.
REPORT: Were you allowed to use the solution from sub-tasks 4 and 5 to find this Viterbi solution of a shortened sequence? Explain the answer.
To check the results, enter the found Viterbi state path in Moodle for the first four observed symbols of the first sequence.
Finally, it is necessary to calculate the likelihood of observation of the first four output symbols, but along all possible individual paths of the state trellis, according to the example from the guidelines. Enter in Moodle how many of these individual state paths there are in total. Based on the calculated likelihood of the individual state paths, determine how many of the paths of all of them are not possible at all, i.e. enter in Moodle the number of paths that have zero likelihood of observing the truncated sequence.
REPORT: Explain the reason.
Sort the paths from the most likely to the least likely and enter in Moodle what share of the total likelihood of the observation (normalized to 1) is cumulatively achieved along the first five most likely paths of this sorted list.
REPORT: Copy the determined list of most likely state paths. Is there a shortened Viterbi route among them? Can different state pathways have the same likelihood? What does it depend on?

Sub-task 7 - Generate observations for a given model

It is necessary to generate multiple random sequences of the observed output symbols (nex different sequences), where each sequence should be of length T time samples (symbols). To generate data, use the *dhmm_sample* function according to the guidelines, with the HMM model parameters from your individual subtask 1. In your individualized Moodle subtask, the desired number of sequences (nex) and their length (T) are specified. Save this observation matrix as it will be used extensively in subsequent sub-tasks.
Important: In order for the automated verification algorithm to uniquely verify your simulation results, it is essential that the verification uses identical generated sequences as you will generate and use during your execution. Each new call to this *dhmm_sample* function will generate a completely new set of random observations, since this function uses the built-in Matlab pseudo-random number generator. To ensure that the generated sequence is identical to the expected one, it is necessary to initialize this pseudo-random number generator to an identical initial (seed) value. Be sure to do this with the *rng('default')* command before calling the function *dhmm_sample* to generate observations. Your success in generating observations will be checked in the next sub-task.
REPORT: Document the data generation procedure.

Sub-task 8 - Determination of long-term statistics of observed symbols and comparison with their theoretical expectations

For the observations generated in subtask 7, it is necessary to experimentally determine the probabilities of observing all output symbols using similar examples from the guidelines. For the first observed sequence from subtask 7, which is of length T symbols, enter the observation number of each output symbol, 1 to 6, in Moodle, which you will determine with the *hist* function.

After that, it is necessary to determine the theoretical expectations of the long-term probabilities of observing the output symbols. In doing so, first determine the stationary state distribution (pi_stac) by successively multiplying the given transition matrix A by itself T times, and then based on this long-term state probability statistic, and based on the output probability observation matrix B, determine the expected stationary observation probabilities of all output symbols (1 to 6), all in accordance with the example in the guidelines. To verify the accuracy of your solutions, enter the long-term probability of a certain required state of the model in Moodle, as well as the long-term probability of observing a certain required output symbol. The required state and the requested symbol will be specified in your individualized sub-task in Moodle.

REPORT: Discuss the obtained long-term probabilities of individual states, and output symbols. What would a degenerate HMM model look like with the same long-term statistics of output symbol observations?

Finally, determine the empirical long-term probabilities of symbol observation (using the hist function) by averaging the number of symbol occurrences across all nex experiments.

REPORT: Compare them with the just-calculated expected long-term output symbol statistics.

In Moodle, enter the maximum absolute value of difference between the empirical and theoretical probabilities of the output symbols maximized over all 6 output symbols.

Sub-task 9 - Calculation of log-likelihood of observation of individual generated observations based on a given model

For each of the random sequences generated in sub-task 7, it is necessary to calculate the log-likelihood of the observations with the given model, i.e. with the same model used to generate these observations. After that, calculate the maximum, minimum and mean value of log-likelihood averaged over all nex observed sequences, and enter the obtained results in Moodle.

REPORT: Why do the likelihoods of individual sequences differ?

Sub-task 10 - Perform the HMM model parameter training procedure

Based on all observation sequences generated in sub-task 7, it is necessary to calculate two new HMM models using the *dhmm_em* function. Important: in both cases, limit the number of iterations of the EM procedure to a maximum of 200, and set the threshold of relative change of log-likelihood to the previous iteration, for termination of the training to 1E-6 by adding these options to the *dhmm_em* function calls:

```
…,'max_iter', 200, 'thresh', 1E-6);
```

For the first HMM model, the initialization of the model parameters for the initial iteration of the EM procedure should be completely random, as explained in the guidelines in the program section that begins with the comment „`% initial guess of parameters`“.

Important: in order for the automated verification of your results to use identical random parameters of the initial model (prior1, transmat1 and obsmat1), reset the pseudo-random number generator to the initial value with the *rng ('default')* command before executing this command block.

For the second HMM model, use the specified model parameters (from sub-task 1) to initialize the EM process (i.e. the training is initialized with the same model used to generate training observations), so that the new "fine-tuned" model can better describe these specific observed sequences.

In Moodle, enter the number of iterations required to estimate the parameters of the HMM model by the EM procedure for both models (first and second).
REPORT: Explain the difference in the number of iterations in the report. The actual accuracy of your calculation of the model parameters will be verified in the next sub-task.

Sub-task 11 - Comparative evaluation of the given model, random model and trained models on the same data used for training

As illustrated in the guidelines, it is necessary to compare the modeling performance of the observation sequences generated in subtask 7 with all available HMM models, by calculating the log-likelihood of observing all generated sequences using the *dhmm_logprob* function. For the specified model, this has already been done in subtask 9, by calculating the log-likelihood of each individual generated sequence. Now we want to repeat this for the specified model as well as the two new trained HMM models from sub-task 10, that differ from each other only in the way the training process was initialized, but now you should find the total log-likelihood of observing all generated sequences from the sub-task 7 simultaneously (not for individual sequences).
As a "bad" model for baseline comparison, you shall use the HMM model with completely random parameters, the same one that was used to initialize the first of two new "optimal" HMM models in the previous sub-task 10 (prior1, transmat1 and obsmat1).
Important: make sure that the parameters of this random model are indeed generated immediately after the initialization of the pseudo-random number generator to be identical to those used in the automated verification of your results.
In Moodle, enter the obtained log-likelihood of the observation for: the given (specified) model, for the “bad” random model, and for the two new HMM models defined in subtask 10: the optimal model with random initialization and the optimal model initialized with the given model.
REPORT: Explain the relationship of the log-likelihood of individual sequences from subtask 9 to the total log-likelihood of all sequences for the given model. Discuss the obtained results of the new “optimal” models in comparison with the log-likelihood of observing the same sequences for the given model. Show and compare the actual estimated values of the parameters of the new trained models (matrices A, B, pi) with the given model. How do you explain the differences in the parameters of these models? Is a verification of the estimated model on the same dataset used for training an appropriate procedure? How should the right model training and validation process be implemented?