INM431 Machine Learning
Tutorial 9 - Solutions
Hidden Markov Models (HMMs)

Note: Please download and extract the tutorial-9.zip file found in the ML module Moodle page for Section 9. Then open Matlab and go to the path of the extracted folder. A list of pmtk functions related to HMMs can be found in hmm.html .

1. Discrete HMM demo

a. Open file hmmDiscreteTest.m and run it by typing hmmDiscreteTest in Matlab.

b. Inspect the code in hmmDiscreteTest.m: note how a discrete HMM can be constructed, by pre-defining a transition matrix, a vector of prior probabilities, and an emission matrix. Useful functions:
- hmmSample(): on generating sequences of observations and hidden states using an HMM.
- hmmFit(): can be used to learn HMM parameters using the EM/Baum-Welch algorithm by using a sequence of observations as input.
- hmmLogprob(): returns the log-probability of an HMM given a sequence of observations. Useful for comparing models and for creating HMM-based classifiers.
- hmmMap(): decoding - finds the most probable sequence of hidden states given a sequence of observations (Viterbi algorithm).

2. Gaussian HMM demo

a. Open file hmmGaussTest.m and run it by typing hmmGaussTest in the Matlab command line.

b. Inspect the code in hmmGaussTest: note that the dataset is composed of several sequences of 13-dimensional features computed from audio files. These features are called MFCCs (for info: they stand for “Mel-frequency cepstral coefficients”) and are used frequently in automatic speech recognition applications. The dataset consists of several utterances of the words “four” and “five”. Also note that a continuous 2-state HMM is created which uses two 13-dimensional Gaussians for the emission probabilities (this can be seen by typing model.emission in the Matlab command line).

3. Discrete HMM exercise

Sarah is a Data Science student who does one of the following activities in her free time every working day: cycle, shop, or watch a movie (she can do only one activity per day). Her choice of activity is however constrained by the weather, which might be rainy or sunny. Assume that the probability for the first working day’s weather being rainy is 0.2, and that the weather changes every day according to the following transition table (rows stand for the previous day, columns for the next one):

	Rainy	Sunny
Rainy	0.3	0.7
Sunny	0.4	0.6

Also assume that her choice of activity depends on the weather, following the probabilities of the table below:

	Rainy	Sunny
Cycle	0.1	0.6
Shop	0.4	0.3
Movie	0.5	0.1

a. Construct a discrete HMM in Matlab importing the above probabilities.

```
nHidStates = 2;
model.pi = [0.2 0.8];
model.A = [0.3 0.7; 0.4 0.6];
model.emission = tabularCpdCreate([0.1 0.6; 0.4 0.3; 0.5
0.1]');
model.type = 'discrete';
```

b. Generate a possible sequence of activities for 5 days, along with the corresponding weather.

```
[observed, hidden] = hmmSample(model, 5, 1);
```

c. Mary is having a conversation with Mark on last week's activities, and mentions the following activities for the last 5 days: cycle, movie, movie, shop, cycle. What is the probability of observing this sequence?

```
obs = [1 3 3 2 1];
disp(exp(hmmLogprob(model, obs)));
```

d. Given the sequence of activities from question (c), what is the most probable weather pattern during these 5 days?

```
seq = hmmMap(model, obs);
```

4. GMM-HMM exercise

Use again the speech data from question 2, which contain features corresponding to utterances of the words 'four' and 'five'. Load the data using *load('speechDataDigits4And5');* The goal of this exercise is to create a continuous/Gaussian HMM classifier – and test that it works.

a. Train two GMM-HMMs, one per word class. As training data, use ‘train4’ and ‘train5’ for each word class (once the dataset is loaded into Matlab, these structures appear in the variable workspace). Model each training subset as a GMM-HMM using a mixture of 5 Gaussians.

```
load('speechDataDigits4And5');
model_4 = hmmFitEm(train4', 5, 'gauss', 'verbose', true,
'maxIter', 100);
model_5 = hmmFitEm(train5', 5, 'gauss', 'verbose', true,
'maxIter', 100);
```

b. Perform testing using the test set ‘test45’ (again part of the variable workspace when loading the dataset). The test set contains 252 utterances, with corresponding labels stored in variable ‘labels’.

- Compute the log-probability for the test set for each trained HMM.
- For each test sample (out of 252), compare the two computed log-probabilities. The HMM that maximises the log-probability is chosen as the class prediction for that sample.
- Using the ‘labels’ class annotation, compare the predicted labels using the above step with the annotated labels. Compute a classification accuracy metric as the percentage of the ratio of correctly predicted labels over the total number of test samples.

```
logprob_4 = hmmLogprob(model_4, test45);
logprob_5 = hmmLogprob(model_5, test45);
for i=1:length(test45)
     if(logprob_4(i)>logprob_5(i)) predictedLabel(i) = 4; else
predictedLabel(i) = 5; end;
end
count = 0;
for i=1:length(test45)
     if(labels(i)==predictedLabel(i)) count = count+1; end;
end
disp(['Classification accuracy: '
num2str(100*count/length(test45)) '%']);
```