# Exercise 5

Deadlines: Monday 2008.10.06 (final, no student correction)

Write, comment and hand in a script that performs tasks 2-7 in this exercise (preferably by email to xffhello@gmail.com). Use comments (% at the start of a line) to answer the questions. There are some hints in a not so random order at the end of the exercise.

Matlabs Statistics Toolbox contains the following functions for HMM :

hmmdecode -- Calculates the posterior state probabilities of a sequence
hmmgenerate -- Generates a sequence for a hidden Markov model
hmmestimate -- Estimates the parameters for a Markov model
hmmtrain -- Calculates the maximum likelihood estimate of hidden Markov model parameters
hmmviterbi -- Calculates the most likely state path for a hidden Markov model sequence

**Task 1:**

Read and understand the documentation for each of these functions (type help <function> or doc <function> at the prompt).

**Task 2:**

Use hmmgenerate to generate 50 sequences 10 letters long from the 2 state HMM specified below (in the format used by the Matlab functions), collect the paths as well:

Model:

```
transitionProbabilities =
[0.95,0.05;
0.10,0.90];

emissionProbabilities =
[1/3, 1/3, 1/3;
 1/4, 1/4, 1/2;];
```

All of the Matlab functions used assume that the first state is state 1 with probability 1 *before* the sequence is observed (i.e. the first state in the path is chosen from the transition probabilities from state 1).

**Task 3:**

Pick a couple of the generated sequences (and paths) to estimate transition and emission parameters for each of the models assuming the paths were known using hmmestimate. How well do they agree with the true values? Note that hmmestimate will return e.g. a 1x1 matrix for the transition probabilities if only one state is visited in the path.

**Task 4:**

Pick a couple of the sequences you generated in **Task 2**. Use hmmviterbi to calculate the most probable path for this sequence. Compare this to the true path. What is the percentage of correct assignments? Would you consider that useful performance?

**Task 5**:

Use hmmdecode to determine the probability of being in either state along the sequence (use the same sequences as in **Task 4)**. Is the most probable state equal to the state of the Viterbi path at all times?

**Task 6:**

Use hmmtrain to estimate the transition and emission probabilities of each of the models from the sequence data generated in **Task 2**. Try Viterbi-learning and Baum-Welch as well as different initial guesses for emission and transition probabilities.

How well do the estimated transition and emission parameters agree with the true values? What happens if you use the true values as initial values? Would you expect to get the true values back? Are the training algorithms sensitive to the initial guess?

What happens when Viterbi training is used and one of the states is never visited? How could this problem be solved?

**Task 7:**

This task will emulate a practical application of HMMs for sequence classification.

1. Generate two training sets of 20 sequences, 10 letters long from each of the two models below (in Matlab format as in **Task 2**).

Model 1:

Transitions1 =

[0.95,0.05;
0.10,0.90];

Emissions1 =
[1/3, 1/3, 1/3;
1/4, 1/4, 1/2];

Model 2:

Transitions2 =
[0.7, 0.15, 0.15;
0.10, 0.4, 0.5;
0.8, 0.1, 0.1];

Emissions2 =
[1/4, 1/4, 1/2;
1/3, 1/3, 1/3;
4/6, 1/6, 1/6];

2. Estimate transition and emission parameters for each of the models using hmmtrain on the sequences generated in (1).

3. Generate a test set of 5 sequences, 10 letters long from each of the two models.

4. Use the parameters estimated in (2) and hmmdecode to calculate the probability of the sequences from Model 1 being generated by Model 1 and Model 2. Are the sequences more probable for Model 1 or Model 2? Repeat but calculate the probabilities for the test set drawn from Model 2.

5. Repeat (4) but use the true parameters in hmmdecode. How is the performance now?

6. Can you see any dangers with comparing two models of different complexity (explanatory power)?

**Not so random hints:**

```
transitions = ...
[0.8,0.2; ...
0.10,0.90];

emissions = ...
[1/3, 1/3,  1/3; ...
 1/4, 1/4,1/2;];

len = 10;

help cell



for i=1:50
    [seqs{i}, states{i}] = hmmgenerate(len,transitions,emissions);
end

[TR, E] = hmmestimate(seqs{2},states{2})
[TR, E] = hmmestimate(seqs{3},states{3})


viterbiStates = hmmviterbi(seqs{1},transitions,emissions)
states{1}

statesProbabilities = hmmdecode(seqs{1},transitions,emissions)
viterbiStates

jnk = rand(2);
trGuess = jnk./(sum(jnk, 2) * ones(1,2));
jnk = rand(2,3);
eGuess = jnk./(sum(jnk, 2) * ones(1,3));

[estimatedTransitions, estimatedEmissions] = hmmtrain(seqs,trGuess,eGuess,'algorithm',
'BaumWelch')
[estimatedTransitions, estimatedEmissions] = hmmtrain(seqs,trGuess,eGuess,'algorithm',
'Viterbi')

[estimatedTransitions, estimatedEmissions] = hmmtrain(seqs,transitions,emissions,'algorithm',
'BaumWelch')
[estimatedTransitions, estimatedEmissions] = hmmtrain(seqs,transitions,emissions,'algorithm',
'Viterbi')

Transitions1 = ...
[0.95,0.05; ...
0.10,0.90];

Emissions1 = ...
[1/3, 1/3, 1/3; ...
 1/4,  1/4,  1/2];

Transitions2 = ...
[0.75,0.15, 0.15; ...
```

```
0.10, 0.4, 0.5;...
0.8, 0.1, 0.1];

Emissions2 = ...
[1/4, 1/4, 1/2; ...
1/3, 1/3, 1/3; ...
4/6, 1/6, 1/6];


for i=1:20
    [trainSeqs1{i}] = hmmgenerate(len,Transitions1,Emissions1);
end



for i=1:20
    [trainSeqs2{i}] = hmmgenerate(len,Transitions2,Emissions2);
end

[eTr1, eE1] = hmmtrain(trainSeqs1,trGuess,eGuess,'algorithm', 'BaumWelch')

jnk = rand(3);
trGuess = jnk./(sum(jnk, 2) * ones(1,3));
jnk = rand(3,3);
eGuess = jnk./(sum(jnk, 2) * ones(1,3));

[eTr2, eE2] = hmmtrain(trainSeqs2,trGuess,eGuess,'algorithm', 'BaumWelch')


for i=1:5
    [testSeqs1{i}] = hmmgenerate(len,Transitions1,Emissions1);
end


for i=1:5
    [testSeqs2{i}] = hmmgenerate(len,Transitions2,Emissions2);
end

for i=1:5
    disp(i)
    [statesProbabilities, logP] = hmmdecode(testSeqs1{i},eTr1,eE1);
    exp(logP)
    [statesProbabilities, logP] = hmmdecode(testSeqs1{i},eTr2,eE2);
    exp(logP)
end
```