HOMEWORK 1: HIDDEN MARKOV MODEL

VLADIMIR SHCHUR

You can use any language, but C/C++ or Python are preferred. If you code in Python, *pandas* module is not allowed. Upload your code to `https://gist.github.com/`. The final report can be part of a jupyter notebook or typeset with LaTeX.

M is the number of states, K is the number of possible emissions, L is length.

- Set transition ($M \times M$) and emission ($M \times K$) matrices and beginning distribution (vector of length M).
- HMM random generator: generate a sequence x of length L given the model parameters.
- Use Viterbi algorithm to find the most probable path π given x and HMM parameters.
- Find the probability $P(x)$ using forward algorithm (store the full matrix of forward probabilities $f_k(x_i)$ in order to use it for posterior decoding later on).
- Implement the backward algorithm.
- Implement posterior decoding.
- Implement Baum-Welch training.

Using the previous results, build the HMM for unfair casino example.

- Simulate occasionally unfair casino: one dice has equal probabilities of each face $P(i) = 1/6, i = 1, 2, 3, 4, 5, 6$ and another dice has probability $P(6) = 0.5$ and all other outcomes have probability 0.1. The probability to switch from the fair dice to the loaded dice is 0.05, the probability to switch from the loaded to the fair dice is 0.1. The beginning distribution is the stationary distribution of the underlying Markov chain.
- Plot and compare the simulated path and Viterbi most likely path.
- Plot posterior probability $P(fair)$. Compare with the regions of the simulated path where loaded dice was used.
- Use Baum-Welch training to estimate HMM parameters (transition and emission probabilities, beginning distribution). Try different parameter initialisations and sequence length, analyse the rate of convergence.
- Write a brief discussion on the HMM performance (e.g. qualitative analysis of Viterbi and posterior decoding based on the plots. You can also try different values of L and M in order to observe the scalability of these algorithms (optional).

Hint. For long sequences, underflow can happen. Use either renormalisation technique (see *Durbin et al.*), or log-space computations for the Viterbi and forward/backward probabilities to protect it from underflow ($\log AB = \log A + \log B$). For summations, assuming

that $A \geq B$, in the log space we have

$$\log(A+B) = \log(e^{\log A} + e^{\log B}) = \log\left(e^{\log A}(1 + e^{\log B - \log A})\right),$$

hence

$$\log(A+B) = \log A + \log(1 + e^{\log B - \log A}).$$