Pattern and Speech Recognition Tutorial
Exercise 10

Exercise 1 (3 points)

To start with hidden Markov models (HMM), you will implement a simulation algorithm. Later you can use this as a reference to compare results with.

1. Chose a suitable data structure for (finite) HMMs. Justify your choice.

2. Implement a function `random_select`($[p_1, \dots, p_n]$) which takes a sequence $\mathbf{p} \in \mathbb{R}^{n+1}_{\geq 0}$, selects randomly an element $p_i \in \mathbf{p}$, and return its index i.

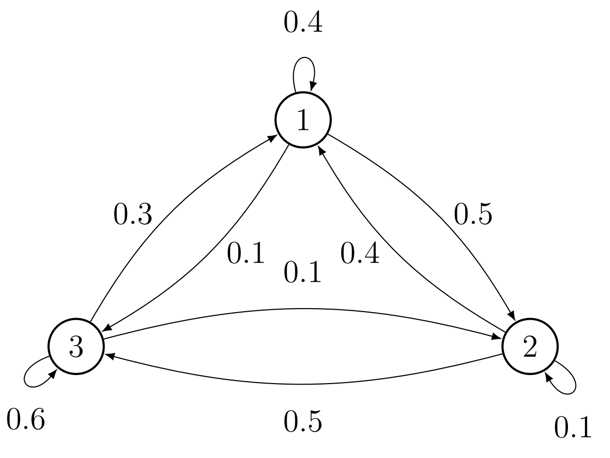
   Assume that $\mathbf{p}$ represents a probability, that is, $\sum_{p_i \in \mathbf{p}} p_i = 1$ and $\forall i \; p_i > 0$. Construct the function such that the probability of selecting p_i (resp. returning i) is equal to p_i.

   For instance, for $\mathbf{p} = [\frac{1}{2}, \frac{1}{4}, \frac{1}{4}]$ your function should return 0 with probability $\frac{1}{2}$ and 1,2 with probability $\frac{1}{4}$ (assuming you start indexing with 0). You can use this function to randomly select successor states, emissions, and an initial state in your simulation.

3. Implement a function `simulate(hmm, max_t)` which simulates a HMM `hmm` for `max_t` (`max_t` $\in \mathbb{Z}_{\geq 0}$) transitions.

   This includes, randomly selecting an initial state (once), a successor state and an emission symbol (for each step), follow slide 29. Return a sequence of pairs $[(\pi_0, o_0), (\pi_1, o_1), \dots]$ where $\pi_i \in Q$ is the i–th hidden state and $o_i \in \Sigma$ is the i–th emitted symbol.

4. Apply `simulate` to simulate the HMM from figure 1 for `max_t` $= 100$. Include some example outputs to your report file.



state	a	b	c
1	0.6	0.1	0.3
2	0.5	0.4	0.1
3	0.05	0.15	0.8

Figure 1: hidden states (l) and emission matrix (r); $Q = \{1, 2, 3\}$, $\Sigma = \{a, b, c\}$, initial distribution: $1 : \frac{1}{3}$, $2 : \frac{1}{3}$, $3 : \frac{1}{3}$.

Exercise 2 (7 points)

1. Implement the Viterbi–algorithm (slide 27 ff.). That is, write a function `viterbi(hmm,`$[o_0, \ldots, o_n]$`)` which determines the likeliest sequence of hidden states $\pi_0, \ldots, \pi_n \in Q^{n+1}$ for a given sequence of observed symbols $o_0, \ldots, o_n \in \Sigma^{n+1}$, and for a given HMM. `viterbi` should return the likeliest sequence and its corresponding probability.

   You can assume that your HMM (and the observation sequence) is small enough to not produce underflows (i.e. there is no need to use logarithms to store numbers).

2. Apply `viterbi` to the HMM in figure 1 and $\mathbf{o}$ = abbabcc

3. Consider the sequence $\mathbf{s}$ which contains the likeliest state at each position for a given observation sequence $\mathbf{o}$, i.e. $s_i = \underset{k \in Q}{\operatorname{argmax}} P(\pi_i = k|\mathbf{o})$. Find a HMM and an observation sequence such that $\mathbf{s}$ and the likeliest sequence $\mathbf{v}$ (i.e. the results of the Viterbi–algorithm) are completely disjoint. That is, $\forall\, i, j\; s_i \neq v_j$. Justify your answer.

4. How (why, and to which extent) can one use `simulate` to approximate the result of `viterbi`? Implement a function which tries to find the likeliest sequence of hidden states using a large number of simulation runs. Can you do something similar to estimate $\mathbf{s}$? Explain.

Exercise 3 (2 bonus points)

Implement a function `backward(hmm, o)` which computes the backward–probabilities (denoted by $b_k(i)$ in the slides) for each state k and position i and apply it to the HMM from figure 1 with $\mathbf{o}$ = cbacba.

Submission architecture

You have to generate a **single ZIP file** respecting the following architecture:

```
tutorial1_<matriculation_nb1>_<matriculation_nb2>_<matriculation_nb3>
|
+--- source
|      |
|      +------- file 1
|      +------- file 2
|      +------- ...
+--- rapport.pdf
+--- README.txt
```

---

where

- **source** contains the source code of your project,
- **rapport.pdf** is the report where you present your solution with **the explanations (!)** and the plots,
- **README** which contains group member informations (name, matriculation numbers and emails) and a **clear** explanation about how to compile and run your source code

The ZIP filename has to be :

`tutorial10_<matriculation_nb1>_<matriculation_nb2>_<matriculation_nb3>.zip`

You have to choose between the following languages **python** or **matlab**. Other languages won't be accepted.

Some hints

We advice you to follow the following guidelines in order to avoid problems :

- Avoid building complex systems. The exercises are simple enough.
- Do not include any executables in your submission, as this will cause the e-mail server to reject it.

Grading

Send your assignment to the tutor who is responsible of your group:

- Gerrit Gromann `gerritgr@gmail.com`
- Sbastien Le Maguer `slemaguer@coli.uni-saarland.de`
- Kata Naszdi `b.naszadi@gmail.com`

The email subject should start with `[PSR TUTORIAL 10]`