ITX Integrative Transcriptomics
Prof. K. Nieselt,
Institute for Bioinformatics and Medical Informatics Tübingen
Prof. S. Nahnsen,
Institute for Bioinformatics and Medical Informatics Tübingen

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Lecture: Grundlagen der Bioinformatik — SoSe 2022

Assignment 9 (20 points)

Hand out: Thursday, June 30
Hand in due: Thursday, July 7 18:00

Direct inquiries via the ILIAS forum or to your respective tutor at:
Mathias Witte Paz: iizwi01@uni-tuebingen.de
theresa-anisja.harbig@uni-tuebingen.de
meret.haeusler@student.uni-tuebingen.de
jules.kreuer@student.uni-tuebingen.de
simon.heumos@qbic.uni-tuebingen.de

1. Viterbi Algorithm by hand (2P)

Given the following Hidden Markov Model with a state alphabet $Q = \{b, P, N, e\}$ and an emission alphabet $\Sigma = \{R, Y\}$. The transition and emission probability matrices, resp., are given as

	P	N	e
b	0.7	0.3	0
P	0.2	0.7	0.1
N	0.7	0.2	0.1

	b/e	R	Y
b/e	1	0	0
P	0	0.4	0.6
N	0	0.3	0.7

Your task is to compute the most probable path for the sequence RYY using the Viterbi algorithm (by hand). For this, you need to compute the Viterbi variable values $v_l(i) = e_l(x_i) \max_{k \in Q}(v_k(i-1)p_{kl})$ for each of the states l in Q.

We have pre-filled the matrix. E.g. $v_P(1) = e_P(R) \max_k(v_k(0)p_{kP}) = 0.4 \cdot 1 \cdot 0.7 = 0.28$. You are asked to finalize the missing entries of the matrix and provide the decoded (path) sequence via traceback.

		sequence				
	V	b	R	Y	Y	e
States	b	1	0	0		
	P	0	0.28	0.0378		
	N	0	0.09			
		0	1	2	3	4

2. Profile HMMs (6P)

Draw the state diagram and estimate the parameters of a profile HMM for the following multiple alignment of DNA sequences:

```
A A - T G
T G - G C
T C T T G
A C T G C
A G - C G
A - - C G
A G C G -
T G - C -
T C C G G
T C - C G
```

Note that a heuristic rule suggests that an alignment column with a fraction of gap symbols below 50% corresponds to a match state of a profile HMM, otherwise it belongs to an insert state.

3. **Supervised training** (2 P)

Given a hidden Markov model $M = \{\Sigma, Q, P, E\}$, for which the parameters of the transition matrix P and emission probability matrix E need to be trained. For the approach of supervised training we sometimes have the problem of overfitting. To avoid this we first count the number of times each particular transition or emission occurs in the training sequences

P_{kl}: Number of transitions from state k to l
$E_k(b)$: Number of emissions of b in state k

We then obtain the maximum likelihood (ML-) estimators for (P, e) by adding plus 1 to each observation and normalize appropriately (this is also called adding a pseudo count with the *Laplace* rule). How do the final ML estimators p_{kl} and $e_k(b)$ look like? Write down the corresponding general formula.

Practical Assignments

4. **Decoding data using the Viterbi algorithm** (10P)

Please implement a method that applies the Viterbi algorithm to a given sequence of symbols and returns the decoded sequence of states. For this task you can use the `hmm_viterbi.py` as a template. This file should be able to read a FASTA file and run the Viterbi algorithm for each of them by using a given HMM model. You can get familiar with the format for the HMM model by looking at the file `cpg.hmm` or in the script on page 163. For each sequence from the input FASTA file that has been decoded using the Viterbi algorithm, your script should return an output similar as the following:

```
Symbols: ACTGTGACGTGT...          (original symbols, 60 char per line)
Viterbi: acTGTGAcgtgt...          (decoded states, 60 char per line)
```

To solve this task:

(a) Complete the function `read_hmm` from the template `hmm_handler.py`. It should receive the path of the file containing the HMM model. The information of the file (states, symbols and matrices) can be saved as properties of the HMM object. **Hint:** You can adapt the reader of the previous assignment to also read the emission states and matrix.

(b) Complete the function `runViterbi` by implementing the four steps of the algorithm: *Initialization*, *recursion*, *termination* and *traceback*. You could use for each step one helper function (see template).

(c) Implement the function `prettyPrinting` that formats the given sequences to match the desired output.

(d) **Testing your implementation:** Apply your program to the four sequences in the FASTA file `input_hmm.fasta` using the HMM model from the file *cpg.hmm*. Export the output as a `txt`-file and include it in your hand in.