GENOME-SCALE ALGORITHM DESIGN
by Veli Mäkinen, Djamal Belazzougui, Fabio Cunial and Alexandru I. Tomescu
Cambridge University Press, 2nd edition, 2023
www.genome-scale.info

## Exercises for Chapter 7. Hidden Markov models (HMMs)

7.1 Argue that $O(|H|)$ working space suffices in Problem 7.3 to compute the final numerical value (not the actual solution path) with the Viterbi algorithm, and in Problem 7.4 with the forward and backward algorithms.

7.2 To traceback the solution of Viterbi algorithm, the naive solution requires $O(n|H|)$ working space. Show that this can be improved to $O(\sqrt{n}|H|)$ by sampling every $\sqrt{n}$th position in $S$, without asymptotically affecting the running time of the traceback.

7.3 The standard algorithm for the multiplication of two matrices of $m$ rows by $m$ columns takes time $O(m^3)$. More sophisticated approaches achieve time $O(m^\omega)$ with $\omega < 3$ (the current record is $\omega \leq 2.38$ and $\omega = 2$ is the best one can hope for). Suppose that we have to compute many instances of the forward and backward algorithms on different sequences, but on the same HMM. Show how to use fast matrix multiplication to improve the running time of such a computation.

7.4 Given an HMM and a sequence $S = s_1 \cdots s_n$, derive an $O(n|H|)$ time algorithm to compute, for all $h \in H$,

$$\operatorname*{argmax}_i \{\mathbb{P}(P, s_1 \cdots s_n) \mid P = p_0 p_1 \cdots p_n p_{n+1} \in \mathcal{P}(n) \text{ and } p_i = h\}. \tag{1}$$

7.5 Multiplication is the source of numerical errors in HMM algorithms, when the numbers become too large to fit into a computer word. Show how the Viterbi algorithm can be implemented using a sum of logarithms to avoid these numerical problems.

7.6 For the forward and backward algorithms the sum of logarithms conversion is not enough for numerical stability. Browse the literature to find a solution for this problem.

7.7 The flexibility of choosing the states, transitions, emissions, and their probabilities makes HMMs a powerful modeling device. So far we have used a *zeroth-order Markov model* for emission probabilities (probabilities only depended on the state, not on the sequence context). We could just as well use *first-order Markov chains* or, more generally, *$k$-th order Markov chains*, in which the probability depends on the state and on the last $k$ symbols preceding the current one: $\mathbb{P}(s_i \mid s_{i-k} \cdots s_{i-1}) = \mathbb{P}(s_i \mid s_1 \cdots s_{i-1})$.

Notice that the states of the HMM are independent, in the sense that each state can choose a Markov chain of a different order from that of the Markov chain for its emission probabilities. In addition to the use of different order Markov chains, we could adjust how many symbols are emitted in each state. Use these considerations to design a realistic HMM for *eukaryote gene prediction*. Try to take into account intron/exon boundary di-nucleotides, codon adaptation, and other known features of eukaryote genes. Consider also how you can train the HMM.

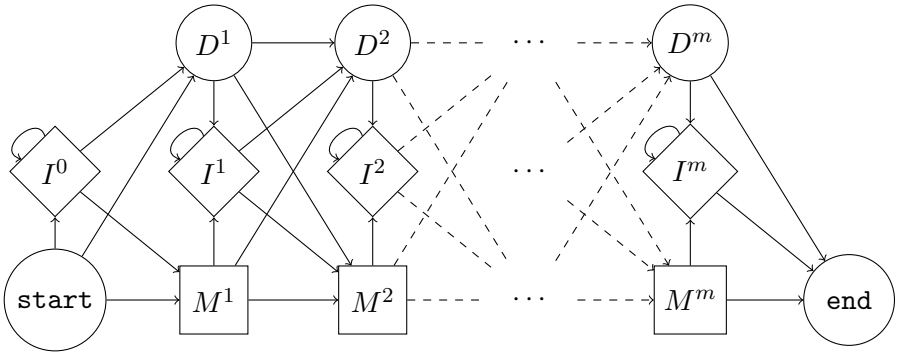


Figure 1: Profile HMM illustration without showing the transition and emission probabilities.

7.8 *Profile HMMs* are an extension of HMMs to the problem of aligning a sequence with an existing multiple alignment (profile). Consider for example a multiple alignment of a protein family:

```
AVLSLSKTTNNVSPA
AV-SLSK-TANVSPA
A-LSLSK-TANV-PA
A-LSSSK-TNNV-PA
AS-SSSK-TNNV-PA
AVLSLSKTTANV-PA
```

We considered the problem of aligning a sequence $A$ against a profile in the context of progressive multiple alignment in Section 6.6.4, and the idea was to consider the multiple alignment as a sequence of columns and apply normal pair-wise alignment with proper extensions of substitution and indel scores. Consider $A =$ `AVTLSLSTAANVSPA` aligned to our example profile above, for example, as follows:

```
AVTLSLS--TAANVSPA
AV-LSLSKTTN-NVSPA
AV--SLSK-TA-NVSPA
A--LSLSK-TA-NV-PA
A--LSSSK-TN-NV-PA
AS--SSSK-TN-NV-PA
AV-LSLSKTTA-NV-PA
```

Here we have added two gaps to the sequence and two gap columns to the profile following the "once a gap, always a gap" principle.

Profile HMMs are created using *inhomogeneous* Markov chains, such that each of the columns will form separate match, insertion, and deletion states, and transitions go from left to right, as illustrated in Figure 1. Match and deletion states emit the columns of the profile, so they do not contain self-loops. Insertion states emit symbols from the input sequence, so they contain self-loops to allow any number of symbols emitted between states that emit also columns of the profile.

Since the resulting HMM is reading only one sequence, the Viterbi, forward, and backward algorithms are almost identical to the ones we have studied so far. The

only difference is that deletion states are *silent* with respect to the input string, since they do not emit any symbol.

a) Modify the Viterbi recurrences to handle both emitting and silent states.

b) Derive the Viterbi recurrences specific to profile HMMs.

7.9 Derive a local alignment version of profile HMMs.

7.10 *Pair HMMs* are a variant of HMMs emitting two sequences, such that a path through the HMM can be interpreted as an alignment of the input sequences. Such pair HMMs have a *match* state emitting a symbol from both sequences simultaneously, and symmetric *insertion* and *deletion* states to emit only from one input sequence.

a) Fix a definition for pair HMMs and derive the corresponding Viterbi, forward, and backward recurrences. *Hint.* The result should look very similar to Gotoh's algorithm for global alignment with affine gap costs from Section 6.4.4.

b) Apply a derivation for pair HMMs similar to the ones we used in obtaining relation (7.13), in order to define the probability of $a_i$ aligning to $b_j$ over all alignments of $A = a_1 \cdots a_m$ and $B = b_1 \cdots b_n$.

c) Let $p_{ij}$ denote the probability derived above to align $a_i$ to $b_j$. We say that the *most robust alignment* of $A$ and $B$ is the alignment maximizing the sum of values $p_{ij}$ over $i, j$ such that the $a_i \to b_j$ substitution is part of the alignment. Derive a dynamic programming algorithm to compute this most robust alignment.

## Additional exercises not in the book

7.11 Assume a set of DNA sequences with coding/non-coding labeling. *Implement* a program that trains the emission/transition probabilities for the coding/non-coding HMM considered at the book, given the training data.

7.12 *Implement* the Viterbi algorithm in the special case of the coding/non-coding HMM. Implement also the tracing back of the optimal path and deduce the labels for some example sequence. If you did the previous assignment, train the HMM with some other example sequence.

7.13 Familiarize yourself with the HMM package of Biopython: `http://biopython.org/DIST/docs/api/Bio.HMM-module.html`. Explain how the provided functions match the description in the book.

7.14 Use Biopython to run the Viterbi algorithm on the coding/non-coding HMM.