Hidden Markov Models for Genome Analysis

Leonardo Gori

May 2022

1 Introduction

Hidden Markov Models (HMMs) are statistical models capable of capturing hidden information from observable sequential symbols (e.g., a nucleotidic sequence). These models have many applications in sequence analysis, in particular, to predict exons and introns in genomic DNA, identify functional motifs (domains) in proteins (profile HMM), align two sequences (pair HMM). In the context of a hidden Markov model (HMM), the forward algorithm is used to calculate a **belief state**: the probability of a state at a certain time, given the history of evidence. The goal of the project is to implement the hidden Markov model algorithm. The history of the whole project is preserved on GitHub. The activity performed during the project is presented in section 2, while in section 3 describes the proposed algorithm. Finally, the actual implementation of the project is described in section 4.

2 Summary of the performed activities

Before the implementation of the program, an analysis of the problem has been performed. In particular, the attempt made was understanding the concepts and the components used for the correct execution of the algorithm.

In the very beginning, the concepts of pairwise alignment, Markov chains and hidden Markov models, the Viterbi algorithm and the forward algorithm have been gathered from chapter 2 and 3 of [DEKM98].

Once gained confidence with the theory behind these models and algorithms, as a second stage of the analysis, the attempt of understanding the Pairwise alignment using HMMs in section 4 of [DEKM98] was made with less difficulties.

Subsequently, following the extract by Benjamin ([Ben17]) and the precious information by [RBAA18], the actual implementation of the algorithm has been made.

Finally, the implementation of the algorithm was made in order to provide parallel independent computation for improving the performances of the computation of the

3 Algorithm's description

The forward algorithm is a dynamic programming algorithm that is based on the pair HMM theory for the computation of the overall alignment quality between a candidate haplotype and an input read. For what concerns pair HMMs, they can be informally described by two properties:

- The transition probability from one state to another, for all the states contained in a well pre-defined set of states S
- For each state in S, its emission distribution of all the values that are contained in a well pre-defined set of values V

For the purpose of sequence alignment, a special kind of HMMs, defined as **pair HMMs** can be defined by:

- a set of states $S = \{M, I, D\}$ and their transition probabilities
- a set of nucleotide basis $N = \{A, C, G, T\}$, from which V is derived as the set containing all the nucleotide pairs $(a, b), \forall\ a, b \in N$ and, for each of them, its emission probability $\mathcal{P}_s(a, b)$ for each state $s \in S$

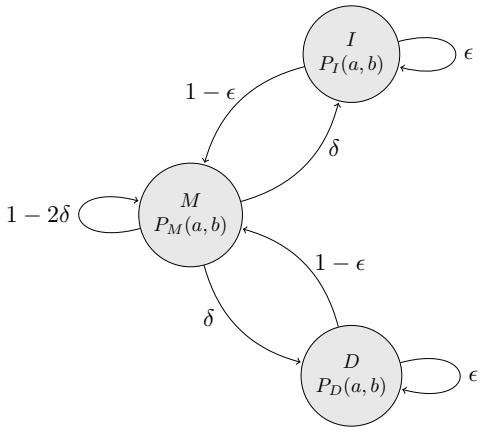

Figure 1: Naive representation of a Pair HMM, composed by its transmission states and emission probabilities

The hidden states are represented as M, D and I. When in state M, the pair-HMM emits symbols from both sequences, implying that the symbols may align to each other. When in state I, it emits one symbol from sequence X, and a blank symbol - meaning no symbol from sequence Y, indicating an insertion in sequence X. Similarly D state represents a deletion in sequence X (or an insertion in sequence Y)

The probability of the particular alignment is the product of the corresponding state transition probabilities. To find the overall alignment probability, we need to find the sum of the probabilities of all such alignments between the two sequences. However, if we follow a bruteforce approach to evaluate each possible sequence alignment, it can be computationally expensive as there can be a large number of possible alignments between two sequences.

In order to improve the performances, the forward algorithm is used to efficiently compute the overall alignment probability, through a dynamic programming approach. We define the **state transition matrix** T, containing the transition probabilities of switching from one state to another (i.e.: T_{MI} is the transition probability of switching from a matching state M to an insertion state I). Furthermore, we define $\mathcal{R}$ and $\mathcal{H}$ as the **read** and **haplotype** sequences, and we refer to $\mathcal{R}_i$ and $\mathcal{H}_j$ respectively as the $i-th$ and $j-th$ elements of the read and haplotype nucleotidic sequences.

Algorithm 1 Pair HMM algorithm

Initialize:

- $M_{i,0} = I_{i,0} = D_{i,0} = 0, \forall\ 0 \leq i \leq |\mathcal{R}|$
- $M_{0,j} = I_{0,j} = 0, \qquad \forall\ 0 \leq j \leq |\mathcal{H}|$
- $D_{0,j} = 1/n, \qquad \forall\ 0 \leq j \leq |\mathcal{H}|$

for $1 \leq i \leq |\mathcal{R}|$ **do**
  for $1 \leq j \leq |\mathcal{H}|$ **do**
    $M_{ij} = \mathcal{P}_M(\mathcal{R}_i, \mathcal{H}_j) \cdot (M_{i-1,j-1}T_{MM} + I_{i-1,j-1}T_{IM} + D_{i-1,j-1}T_{DM})$
    $I_{ij} = M_{i-1,j}T_{MI} + I_{i-1,j}T_{II}$
    $D_{ij} = M_{i,j-1}T_{MD} + D_{i,j-1}T_{DD}$
  end for
end for
Total likelihood $P(\mathcal{R}|\mathcal{H})$ is $\sum_j (M_{\mathcal{R},j} + I_{\mathcal{R},j})$.

Since the computation of the values that are placed on the anti-diagonal path of the matrices is independent one to another, then a parallel computation of all the elements that respect this property can be performed for each loop in the algorithm. This behaviour is proposed as a goal in the implementation of the code.

4 Implementation

The code is entirely written in C++ programming language, with the use of the following libraries and APIs (omitting the standard ones):

- random: used for the random generation of sequences and the random definition of state transition probabilities
- algorithm: used for the shuffling of sequences, used for randomization purposes

Since the computation of the components that build up the matrices, the execution of the algorithm has been developed in a parallel fashion, by making use of the **OpenMP** APIs.

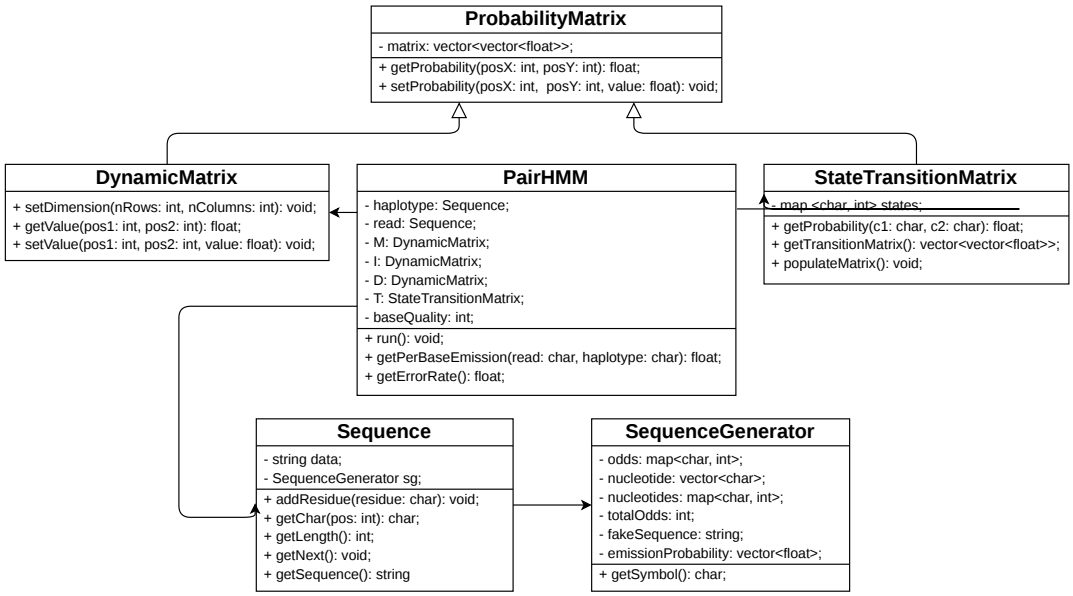

Figure 2: High level UML class diagram reporting the classes that have been defined for the implementation of the algorithm for the analysis of genomic sequences.

The main classes that build up the project are:

- **Sequence**: class that represents the sequence of nucleotides, it contains the string of characters that compose the sequence and the class SequenceGenerator
- **SequenceGenerator**: class that defines a random emission probability distribution of a sequence of nucleotides. Currently, an instance of the class Sequence is randomly generated according to its SequenceGenerator.
- **ProbabilityMatrix**: class that represents a generic matrix of floating point values, from which the classes DynamicMatrix and StateTransitionMatrix inherit common attributes and methods. DynamicMatrix adds the possibility of adding rows and columns dynamically, while StateTransitionMatrix provides a series of states, and a mapping between them and the indexes of the matrix
- **PairHMM**: the class that implements the pair HMM forward algorithm, it encloses 2 instances of the class Sequence (one for defining the read sequence, and one for defining the haplotype sequence), 1 instance of the class StateTransitionMatrix (for defining matrix T), and 3 instance of the class DynamicMatrix (for the definition of matrices M, I and D)

5 Conclusions

Actually, the implementation that is proposed on GitHub performs a parallel execution of the algorithm along the elements placed on the anti-diagonal line of the matrices. However, some loop cycles are actually performed for checking the validity of the accessed elements. Therefore, a part of the performance that is gained by exploiting the parallel execution by multiple threads, is lost by the actual implementation. Once gained confidence with the OpenMP APIs, a better implementation of the algorithm is supposed to be made on the next version of the repository.

References

[Ben17] David Benjamin. Pair HMM probabilistic realignment in HaplotypeCaller and Mutect. `https://github.com/broadinstitute/gatk/blob/master/docs/pair_hmm.pdf`, August 2017.

[DEKM98] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[HMR+17] Sitao Huang, Gowthami Jayashri Manikandan, Anand Ramachandran, Kyle Rupnow, Wenmei W. Hwu, and Deming Chen. Hardware acceleration of the pair-hmm algorithm for dna variant calling. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '17, page 275–284, New York, NY, USA, 2017. Association for Computing Machinery.

[RBAA18] Shanshan Ren, Koen Bertels, and Zaid Al-Ars. Efficient acceleration of the pair-hmms forward algorithm for gatk haplotypecaller on graphics processing units. *Evolutionary Bioinformatics*, 14:1176934318760543, 2018. PMID: 29568218.