UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

SEMINAR

Using Hidden Markov Models for DNA Sequence Alignment

Jelena Glasovac, Hrvoje Radoš
Mentor: *Mirjana Domazet Lošo*

Zagreb, January 18, 2026.

CONTENTS

1. Introduction **1**

2. Seminar **2**

2.1. The Model 2

2.1.1. Model Training 4

2.1.2. Model Predictions 6

2.2. Results 7

2.2.1. Time and memory usage 7

2.2.2. Alignment comparisons 7

3. Conclusion **10**

4. Bibliography **11**

5. Abstract **12**

1. Introduction

Probabilistic graphical models (PGM) describe casual connections between a group of variables, each represented by a node. These nodes are connected by a set of relations. While this works for simple problems, in reality the problems we intend to describe with such models are sufficiently complex for a major issue to occur. When multiple variables have relations to multiple other variables, the complexity of the model increases rapidly, making PGM-s close to unusable.

To combat this, PGM-s introduce a set of latent variables which help describe the relations between multiple variables while reducing the actual number of relations in the model. The latent variables act as a hidden layer of states, therefore earning the name of a Hidden Markov Model (HMM).

Some examples of HMM usage include predictions in finance, speech recognition, and DNA and protein analysis in bioinformatics. The topic of this seminar is exactly that; the usage of HMM-s in DNA alignment.

Despite many mechanisms of repairing mistakes in the process of DNA duplication within a cell, mutations can occur as a result of mistakes which can but do not have to be influenced by the environment in which the organism exists. This creates differences in the DNA sequences between the organisms of the same species, which are also inherited by future generations. By studying genetic material found in multiple organisms of the same species and comparing them to each other, we can attempt to find the most likely sequence of changes in DNA which led to this outcome.

To effectively do this, the observed DNA chains need to be properly aligned. Due to the random nature of mutations, there exists no model which can determine the perfect alignment. That is why probabilistic models are used to find the most likely alignment instead.

In this seminar, an HMM model is implemented, trained and used for alignment of multiple pairs of DNA sequences from HIV virus. The results are then compared to alignments constructed using Needleman-Wunsch (NW) algorithm and discussed in terms of quality as well as time and space usage.

2. Seminar

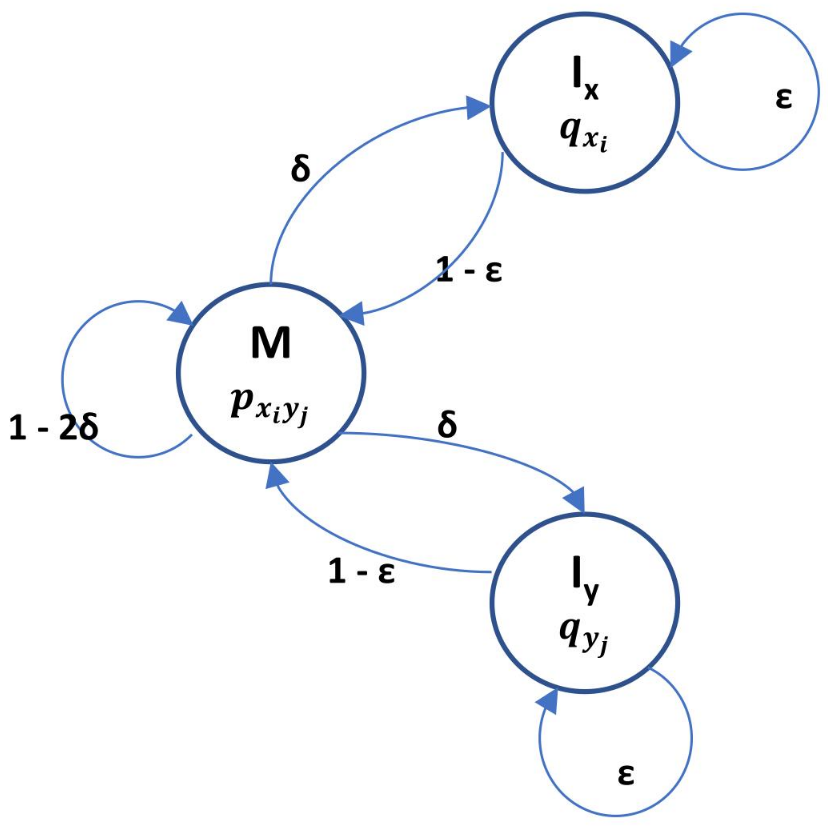
When comparing two DNA sequences x and y, it is important to understand which are the possible mutations which can occur and result in differences. There are two types of mutations taken into consideration in this problem. First type of mutations are point mutations cause a change in a single base. For example, a single Adenine is replaced by a single Thymine. These mutations aren't problematic for alignments on their own because they don't change the positions of other bases in the chain. Second type are frameshift mutations which include insertions and deletions. This means in a chain one base is either added or lost during transcription. These are the problematic mutations for aligning chains because they shift the positions of other bases and cause mismatches in the entirety of following chain.

When aligning pairs of DNA sequences it is crucial to recognize insertions and deletions as special cases in which an empty slot is added to the corresponding chain to combat the issue caused by frameshift. Deletions can also be observed as the insertions into the opposite chain and this is how they will be discussed from now on in this seminar.

2.1. The Model

The model needs three hidden states to accurately describe the situation: Match (M) state which handles both the correct matches and the point mismatches, InsertX (Ix) state which handles the case where there is a base inserted into the X sequence, and InsertY (Iy) state which handles the case where there is a base inserted into the Y sequence. The relations between these states are shown on the picture 1. The probabilities of going from one state to another are calculated in the training process with the limitations that not all relations in the model exist. There is no direct relation between states Ix and Iy as shown in the figure 2.1

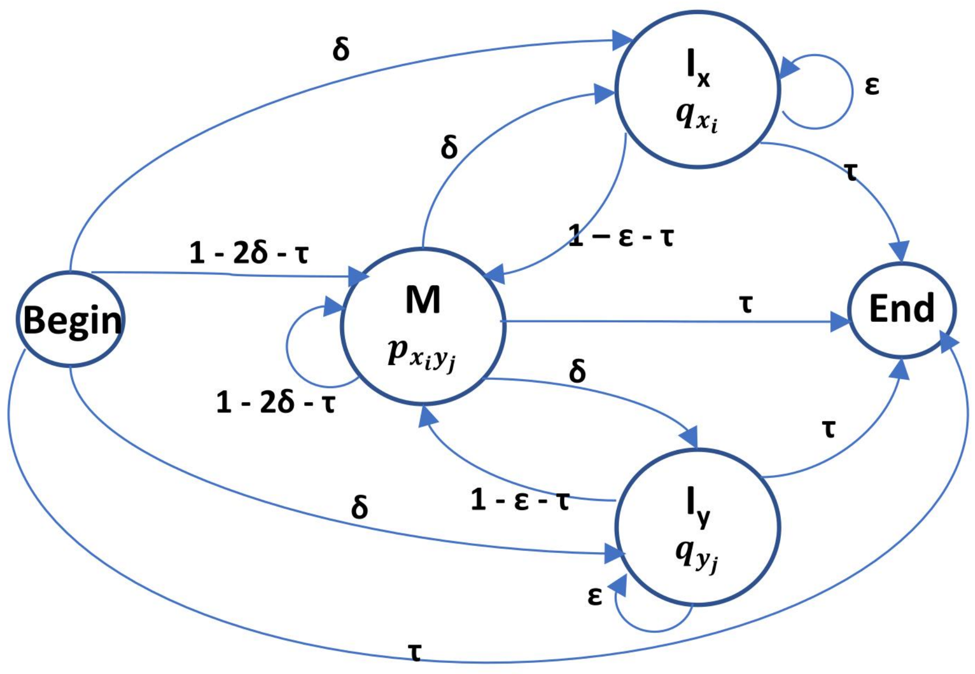
Figure 2.1: HMM scheme



The base model is further expanded by adding a begin state and an end state as can be seen in figure 2.2. From begin state, it is possible to go to any other state while end state is final state and the only transition from it is one which leads back to itself. The image below shows the scheme of final model implemented in this project. Starting probabilities of the relations are shown in the image with parameters $\delta = 0.3$, $\epsilon = 0.4$ and $\tau = 0.1$.

Starting state is the begin state and the only final state is the end state.

Figure 2.2: expanded HMM scheme



2.1.1. Model Training

Training requires a dataset of already aligned sequences, which serve as observations for the model. The goal of training is to estimate the model parameters λ that maximize the likelihood of observed sequences. Since the state sequence is hidden, direct maximization is intractable. Instead, an iterative expectation–maximization (EM) procedure known as the Baum–Welch algorithm is used. The Baum–Welch algorithm relies on the forward and backward variables. The forward variable $\alpha_t(i)$ is defined as

$$\alpha_t(i) = \mathbb{P}(Y_1 = o_1, Y_2 = o_2, \ldots, Y_t = o_t, X_t = i \mid \lambda),$$

Y_t is a random variable that corresponds to the emission at time t and o_i is some value that the HMM emitted. Furthermore, X_t is a random variable that corresponds to the state in which HMM find itself at time t.
$\alpha_t(i)$ can be computed recursively as

$$\alpha_1(i) = \pi_i e_i(o_1),$$

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) e_j(o_{t+1}).$$

Where N is the number of states of the HMM and $e_i(o_j)$ is the probability of emitting o_j while in state i.

Similarly, the backward variable $\beta_t(i)$ is defined as

$$\beta_t(i) = \mathbb{P}(Y_{t+1} = o_{t+1}, Y_{t+2} = o_{t+2}, \dots, Y_T = o_T \mid X_t = i, \lambda),$$

with recursion

$$\beta_T(i) = 1,$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} e_j(o_{t+1}) \beta_{t+1}(j).$$

These formulas can be obtained by using the graph notation associated with HMM's in the context of PGM's.
Using the forward and backward variables, the expected state occupancy probability $\gamma_t(i)$ is computed as

$$\gamma_t(i) = \mathbb{P}(X_t = i \mid \mathbf{Y}, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{k=1}^{N} \alpha_t(k)\beta_t(k)}.$$

The expected state transition probability $\xi_t(i, j)$ is defined as

$$\xi_t(i, j) = \mathbb{P}(X_t = i, X_{t+1} = j \mid \mathbf{Y}, \lambda),$$

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} e_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_t(k) a_{kl} e_l(o_{t+1}) \beta_{t+1}(l)}.$$

In the maximization step, the HMM parameters are re-estimated using the expected sufficient statistics computed in the expectation step. The updated initial state distribution is given by

$$\pi_i^{\text{new}} = \gamma_1(i).$$

The state transition probabilities are updated as

$$a_{ij}^{\text{new}} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}.$$

For discrete observation symbols, the emission probabilities are updated as

$$e_j^{\text{new}}(o) = \frac{\sum_{t=1}^{T} \mathbb{I}(Y_t = o)\gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)},$$

where $\mathbb{I}(\cdot)$ denotes the indicator function.

The Baum–Welch algorithm's convergence is only guaranteed to a local maximum of the likelihood function. In this seminar, the algorithm is terminated after 50 iterations. In practice, numerical underflow occurs when computing forward and backward variables. This issue is addressed by dividing $\alpha_t(i)$ with $A_t = \sum_{j=1}^{N} \alpha_{t-1}(j)$ and dividing $\beta_t(i)$ with $B_t = \sum_{j=1}^{N} \beta_{t+1}(j)$.

2.1.2. Model Predictions

In this project, a pair of DNA sequences can be aligned using the trained model. Viterbi algorithm is implemented and used to compute the most probably alignment of the two sequences.

To do this, three matrices Vm, Vix and Viy are defined, one each for M, Ix and Iy states. Element by element, these matrices are filled to represent the maximum probability that current state is the one which corresponds to the matrix which is being filled and that previously observed symbols have been emitted.

Initially, values Vm(i,0), Vm (0,i), Vix(i,0) and Viy(0,j) were set to 0 with the exception being Vm(0,0) which is set to 1. The matrices are then filled row by row, column by column.

This is where an issue presents itself. Due to the length of the sequences to be aligned, the probabilities quickly become numbers far too small to accurately perform calculations. Instead, the calculations are changed to be performed using logarithms. Despite the numbers in matrices losing their meaning, they retain relations between each other and can be used for accurately finding the path which offers maximum probability, although that probability can no longer be read. All initializations have since been changed to become logarithms of their original values.

Matrix emissionP stores probabilities that an emission j is emitted if current state is state i. Matrix transitionP stores the probabilities of going from state i to state j. The calculation for each element after the change is as follows.

$$V_m(i,j) = log(emissionP(x_i, y_i)) + \max \begin{cases} log(transitionP(M,M)) + V_m(i-1,j-1) \\ log(transitionP(I_x,M)) + V_{ix}(i-1,j-1) \\ log(transitionP(I_y,M)) + V_{iy}(i-1,j-1) \end{cases}$$

$$Vix(i,j) = log(emissionP(x_i, -)) + \max \begin{cases} (transitionP(M,Ix)) + Vm(i-1,j) \\ (transitionP(Ix,Ix)) + Vix(i-1,j) \end{cases}$$

$$Viy(i,j) = log(emissionP(-, y_i)) + \max \begin{cases} transitionP(M,Iy)) + Vm(i,j-1) \\ transitionP(Iy,Iy)) + V^iy(i,j-1) \end{cases}$$

While the matrices are being calculated, it is important to take note of previous state for each element so that optimal path can be reconstructed later. In this implementation, another three matrices have been used to keep track of those. The backtracking is done by reading the pair of sequences from behind and following the path of previous states all the way from end state to the begin state. Begin and end states are marked by emissions (-,-) so these two and match state move sequence pointers to positions x-1, y-1 for each step while states Ix and Iy move pointers to positions x-1,y and x,y-1 respectively.

This way, the most likely path is turned into a possible alignment of the pair of sequences.

2.2. Results

The results of our implementation of HMM were compared to the results of NW algorithm. While HMM has its basis in observations of large training set of already aligned sequences, NW algorithm uses a simple scoring method to find the optimal alignment under the assumption that the chosen scoring method accurately represents reality.

2.2.1. Time and memory usage

Firstly, the comparison of time and memory usages will be presented and discussed, then the quality of results will be discussed. The simplicity of NW algorithm makes it one of the first algorithms to be found as a solution to this specific problem. It requires little time and memory to be highly effective. As a machine learning model, HMMs require lengthy training time, higher memory usage and offers no compensation for this in the average alignment time of pairs of DNA sequences and required memory as can be seen in table 2.1.

Table 2.1: Time and memory usage

	Training		Alignment	
	time /s	memory /???	Align time pair/s	Align memory /???
HMM	449.618		1.7	
NW	-	-		

2.2.2. Alignment comparisons

Secondly, the alignments calculated by the two methods are compared to each other by marking the number of mismatching base pairs between them. Then, the difference is scored by taking the average number of mismatching base pairs and dividing it by an estimated length of the sequences. This score is presented along with median number of mismatches in the table 2.2 for different scoring methods of NW algorithm. NW recognizes different scores for match, for mismatch and insertion/deletion. The most common scoring method is (1,-1,-2) meaning one point is awarded for a math while one point is deducted for a mismatch and two for an insertion or deletion. Except these scores, two additional scoring systems have been defined; one which punishes insertions and deletions less (1,-1,-1) and one which punishes them more (1,-1,-10).

Different scoring of insertions are the most interesting in this case because M state of HMM covers both matches and mismatches, while the probabilities of transitions to Ix and Iy states determine the likelihood of insertions and deletions appearing. It is due to this that by changing NW scoring for insertions, it is expected to find a system which is closer or further away from the HMM model.

As can be seen in the table 2.2, relative difference is the smallest when comparing alignments using standard NW scoring. The outcome isn't surprising. This scoring has been made a standard because it offers satisfying results and has been proven highly effective at creating realistic alignments. The difference is inevitable because of the probabilistic approach used in aligning with Viterbi algorithm but keeping it low between two vastly different methods indicates both are valid solutions to the problem of aligning sequences.

Table 2.2: Average relative difference and median relative difference

Scoring method	Average relative difference	Median relative difference
(1,-1,-2)	2.9%	2.4%
(1,-1,-1)	4.9%	4.6%
(1,-1,-10)	10%	7.2%

Changing insertion scoring, though, lead to an increase in relative difference between the two alignments. This is expected for the increase in insertion score because in reality insertions and deletions are less likely than point mutations and far more dangerous for the organism in question. In the process of implementing and testing the HMM, it has been shown they are so unlikely that a significantly higher amount of smoothing was needed to avoid the probability of their appearance dropping to 0 than the amount of smoothing needed to allow mismatches. This signals they must be scored differently in order to create realistic alignments using NW algorithm.

The surprising part may be how little of a difference it makes in the end, but it becomes understandable after taking a glance at the alignments in question - the vast majority of base pairs are matching and match the algorithm will always prioritize matching as much as possible so the difference between mismatches and insertions doesn't result in a large relative difference. The decrease of insertion score, however, shows that it is necessary to keep the possibility of insertions available more often because although they are rare, if they are too harshly punished, longer matching sequences may be turned into mismatches because of frame shift to avoid a single insertion.

It is the main difference between the two approaches of DNA sequence alignment that the results highlight; NW algorithm uses a rather simple approach which relies on a good estimate of relative likelihoods of appearance of different mutations while HMM uses real data to accurately represent likelihoods. While this difference is relatively small, it is a tradeoff between the accuracy in representing reality and time/memory usage.

This would lead us to believe HMM will always deliver better results, though the issue is that training of this model needs a large amount of already aligned sequences to be effective. If it is trained on a set of data received by using an algorithm which doesn't represent reality well, the benefits fail to present themselves. In this case, though, the training data has been taken from an official source and will be assumed to be the closest possible to an optimal alignment. Under this assumption, the conclusion can be made that HMM offers a good alternative to NW algorithm, which guarantees alignments based on observations instead of man-made estimates.

3. Conclusion

In this paper, a HMM implementation has been presented as a method of solving the problem of DNA sequence alignment. The model has been described in detail, including the training process and the usage of Viterbi algorithm in calculating the most likely alignments. It has been noted that different levels of smoothing are necessary to ensure usability of the HMM model.

The alignments suggested by the model has been compared to the optimal alignments computed using NW algorithm with different scoring systems. The results show the highest level of similarity between alignments created using HMM implementation and using NW algorithm with the standard scoring system of (1, -1, -2) which suggests the two methods produce comparable results in most situations. Despite this, the key difference between the two has been underlined. NW offers a method to get an optimal alignment under the assumption that we can estimate the likelihood of different mutations beforehand, while a HMM is trained on a set of data which is presumed to be correctly aligned and avoids man-made estimates.

Further testing could include comparing the results achieved from HMM with results made using other algorithms or comparing alignments to the source alignments that have been used to generate the testing dataset.

The major area in which improvements are needed, though, is the time and memory usage. While it is impossible to achieve the same level of simplicity as NW algorithm, there is space for improvement. Instead of using vectors in c++, it would be more efficient to use arrays for all matrices that have fixed sizes. The number of operations within loops could be decreased, especially ones which are the most used.

4. Bibliography

5. Abstract

In this paper, alignment of DNA sequence pairs is done using a Hidden Markov model (HMM). The model is discussed in detail, including the training process utilizing Baum–Welch algorithm and the usage of Viterbi algorithm to calculate the most likely alignment of sequences. Appropriate smoothing techniques are discussed as a necessary measure to ensure usability of the model.

The results of this method are then compared to results obtained using NW algorithm with different scoring systems demonstrating the most similarities with the classic scoring system (1, -1, -2). The fundamental differences between the two approaches have been discussed despite the similarities in results.

The time and memory usage of HMM-based approach has been highlighted as the key weakness of this approach and has been suggested as the area in which further improvements may be needed.