Hidden Markov Models in Bioinformatics

Alexander J Ohrt - UPC BarcelonaTech

December 6, 2021

Contents

1 Introduction **2**

2 Objectives **2**

3 Theoretical Background **2**
3.1 Markov Chains . . . 2
3.1.1 Markov Chain Topologies . . . 4
3.2 Hidden Markov Models . . . 4
3.3 Basic Problems for HMMs . . . 5
3.3.1 The Evaluation Problem . . . 6
3.3.2 The Decoding Problem . . . 6
3.3.3 The Training Problem . . . 7
3.4 HMM Variants . . . 8
3.4.1 Standard HMM . . . 8
3.4.2 Profile-HMM . . . 8
3.4.3 Pair-HMM . . . 10
3.4.4 Context-Sensitive HMM . . . 11

4 Applications in Bioinformatics **14**
4.1 Pairwise Sequence Alignment . . . 14
4.2 Multiple Sequence Alignment . . . 14
4.3 Motif Representation (or Identification?) . . . 15
4.4 Prediction of Function . . . 16
4.5 Segmentation . . . 16
4.6 Protein Homology Detection . . . 16
4.7 Gene Prediction / Genomic Annotation (?) / Gene Finding (?) . . . 17
4.8 Protein Sequence Classification . . . 17
4.9 Protein Structure Prediction . . . 17
4.10 Base-calling . . . 17
4.11 Modeling DNA Sequencing Errors . . . 17
4.12 ncRNA Identification . . . 18
4.13 RNA Structural Alignment . . . 18

5 Examples in R **18**
5.1 CG-islands and the "Fair Bet Casino" . . . 18

1 Introduction

These two first parts should be reviewed (perhaps written when done with the rest of the report) This report will present the uses of Hidden Markov models (HMMs) in Bioinformatics. It will explain and highlight why they are populaar alternative tools for solving a wide range of problems in the field. In order to understand why HMMs are useful, and why they can be used effectively, some background on stochastic processses and Markov Chains is needed. Moreover, for completeness, a brief theoretical construct of some different HMMs will be done. In the remaining part of the report, a range of problems in Bioinformatics will be presented, first solved without the use of HMMs, then contrasted with the HMM based solutions. This will hopefully highlight why HMMs are effective alternatives. Note that this report is by no means extensive; there exists a plethora of theory and problems where the HMMs can be used.

2 Objectives

The main objective of this report is to explain why HMMs are very widely used in Bioinformatics. Several problems in the field of study will be presented, with common solutions. First of all, solutions that are not based on HMMs will be shown, with their strengths and weaknesses. These will be contrasted with solution based on HMMs, which in all cases yield alternative methods which oftentimes perform better in some respect. For completeness, the reader should get a quick introduction to the vast theoretical background for HMMs, which includes a quick look at Markov Chains, the Hidden Markov models and different types of them. Furthermore, the most important algorithms for working with HMMs in practice are presented. For the experienced reader, the section on the theoretical background may be skipped. Finally, some practical examples in the open-source programming language R will be developed.

3 Theoretical Background

Basic knowledge in statistics and probability theory is assumed. Any introductory book on probability and statistics will do, but a quick refresher can be found in the two first chapters of [7].

3.1 Markov Chains

A Markov Chain (MC) $\mathbf{X} = \{X_t\}$ is a stochastic process which is characterized by one important property; given the value of X_t, the values of X_b, $b > t$ are not influenced by the values of X_a, $a < t$, where $a, b \in \mathbb{Z}$ decide what *order* of MC we are working with. Choosing $b = t + 1$ and $a = t - 1$ yields the type of *Markov property* we will be concentrating on in this case, which is referred to as a *first-order* Markov property. More precisely, we have that

$$\mathrm{P}(X_{t+1} = j | X_0 = i_0, X_1 = i_1, \ldots X_{t-1} = i_{t-1}, X_t = i) = \mathrm{P}(X_{t+1} = j | X_t = i) =: \mathrm{P}_{ij}^{t,t+1},$$

where the superscripts highlight an important point; the *one-step transition probability* does not only depend on the state, but also on the time when the transition occurs. In our case, we will concentrate on *time-homogeneous* MCs, which means that the transition probabilites are stationary in time. Thus, we drop the superscripts in the notation and denote the transition probabilities as

$$\mathrm{P}_{ij} = \mathrm{P}(X_{t+1} = j | X_t = i).$$

In a MC, the random variable X can only take values from a pre-specified set called *states*. We will concentrate on the simplest type of state space, which yields a so-called *discrete-time* MC. This is the case when the set of states is finite or countable set and whose time index is $T = (0, 1, 2, \ldots)$. Thus,

we have described the assumptions made when working with the simplest form of MC, which we call a *discrete-state time-homogeneous first-order MC* [7]. These lay the foundation of the Hidden Markov models that we will study in the remainder of the report.

Let us define the notation that will be used in the remainder. First of all, let the random vector $\mathbf{X} = (X_0, X_1, \ldots, X_L)^T$ represent a MC of length $L+1$. In the following we will always assume that $\mathbf{X}$ is of finite length $L+1$. A realization of the random vector will be denoted in lower case, i.e. $\mathbf{x} = (x_0, x_1, \ldots, x_L)^T$, following standard notation in most probability or statistics courses. What exactly this means will become clear later. A discrete-time MC consists of a finite set of states, which will be denoted by $\mathcal{H} = \{h_0, \ldots, h_m\}$, where $m+1 = \text{card}(\mathcal{H})$ are the amount of states in the model. Why the letter $\mathcal{H}$ is chosen will hopefully become clear when studying the Hidden Markov models. The MC transitions between each of the states in $\mathcal{H}$, based on the set of *transition probabilities*, which where denoted by P_{ij} above. In the following, the notation $\mathcal{T}_{ij}$ will be adopted instead, for ease of recalling that they are transition probabilities. These are summarized in a $m \times m$-matrix $\mathcal{T}$, where each position $\mathcal{T}_{ij}$ represents the probability of a transition from state i to state j, where $i, j \in \mathcal{H}$. Written in mathematical notation

$$\mathcal{T}_{ij} = \mathrm{P}(X_{t+1} = j | X_t = i), \quad i, j \in \mathcal{H}.$$

As noted, the term *time-homogeneous* refers to the fact that $\mathcal{T}$ does not change as time goes on, i.e. as the MC is generated, but stays fixed during the entire simulation of the chain. In mathematical notation, this assumption means that

$$\mathcal{T}_{ij} = \mathrm{P}(X_{t+1} = j | X_t = i), \quad i, j \in \mathcal{H}, \forall t \in [0, L].$$

Moreover, the term *first-order* refers to an important assumption in this model, which makes it possible to define $\mathcal{T}$ as we did, which is that the transition probability from the previous state to the current state only depends on the previous state. For example, a time-homogeneous second-order MC would depend on the two previous states when transitioning to the current state, which makes the model more general, but also more complicated. Note that the initial state of a MC is determined based on the initial distribution of the states, commonly denoted by $\pi = (\pi(h_0), \ldots, \pi(h_m))$. The assumptions make calculations of probabilities in the MCs very simple, since they are strong independence assumptions.

A MC is completely specified by π and $\mathcal{T}$ (where $\mathcal{H}$ is indirectly given in both these two quantities). Once these quantities are specified, one can make calculations concerning the MC. The probability of a realization $\mathbf{x} = (x_0, x_1, \ldots, x_L)^T$ taking place can be calculated recursively as

$$\begin{aligned}
\mathrm{P}(X_0 = x_0, X_1 = x_1, \ldots, X_L = x_L) &= \mathrm{P}(X_0 = x_0, X_1 = x_1, \ldots X_{L-1} = x_{L-1}) \\
&\quad \cdot \mathrm{P}(X_L = x_L | X_0 = x_0, X_1 = x_1, \ldots X_{L-1} = x_{L-1}) \\
&= \mathrm{P}(X_0 = x_0, X_1 = x_1, \ldots X_{L-1} = x_{L-1}) \\
&\quad \cdot \mathrm{P}(X_L = x_L | X_{L-1} = x_{L-1}) \\
&= \mathrm{P}(X_0 = x_0, X_1 = x_1, \ldots, X_{L-1} = x_{L-1}) \cdot \mathcal{T}_{x_{L-1}, x_L} \\
&= \mathrm{P}(X_0 = x_0, X_1 = x_1, \ldots, X_{L-2} = x_{L-2}) \cdot \mathcal{T}_{x_{L-2}, x_{L-1}} \cdot \mathcal{T}_{x_{L-1}, x_L} \\
&\quad \vdots \\
&= \pi(x_0) \cdot \mathcal{T}_{x_0, x_1} \cdot \ldots \cdot \mathcal{T}_{x_{L-2}, x_{L-1}} \cdot \mathcal{T}_{x_{L-1}, x_L}
\end{aligned}$$

where the second inequality holds because of the Markov property. Also, note that $\pi(x_0)$ refers to the probability that the initial state is x_0. Hence, it is apparent that calculating the probability of a specific realization of the MC is very simple.

Could discuss stationarity and other properties also, if I think it is relevant for the rest

For a more rigorous treatment of stochastic processes and MCs, the reader is referred [7].

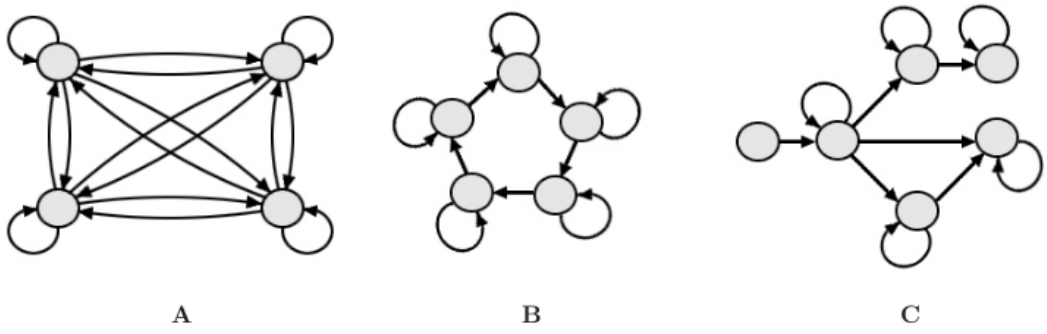

Fig. 2 Some existing HMM topologies. **A.** a fully connected HMM; **B.** a circular HMM; **C.** a left-right HMM.

Figure 1: Examples of the three highlighted topologies stolen from Choo.

3.1.1 Markov Chain Topologies

Perhaps not that important. The topology of a MC refers to which state transitions are permitted and prohibited, i.e. what the values in the transition probability matrix are. There exists many different ways of constructing MCs. As inspired by Choo and colleagues, three different topologies will be highlighted here [2]. Figure 1 gives some simple graphical explanations of these models.

A *fully connected model*, as the name suggests, yields a complete directed graph. This means there are no zero entries in the transition probability matrix, except the possibility of zero entries in the diagonal, which would mean that a loop in the given state is not possible.

A *circular model* is

A *left-right model* is

3.2 Hidden Markov Models

Yoon gives a very nice first introduction to Hidden Markov models (HMMs): "A hidden Markov model (HMM) is a statistical model that can be used to describe the evolution of observable events that depend on internal factors, which are not directly observable." [3]. A HMM, simply put, is a model that comprises of two different parts. The first part is the sequence or the result that can be observed. A possible observed sequence could be a sequence of nucleotides or it could be a chain of events, like 20 dice throws in a row. The second part is a MC, which is *hidden*, i.e. it cannot be observed. This very simple idea gives a very flexible model, which can be used for many different types of problems. The main goal is often to infer the characteristics of the hidden MC from the observed sequence.

In a HMM, a hidden sequence of states is generated, according to the MC. This sequence will not be observable and will in (almost) all practical cases be considered as unknown. Each of the states creates, or *emits*, an observed value. All these observed values are what make up the observed sequence. The emission follows a multinomial distribution, where each state follows such a distribution with different parameters. An important assumption for the HMM we will deal with is that the parameters of each multinomial only depends on the state to which it belongs [2]. This means that the emitted value only depends on its respective hidden state and not on other hidden states. A HMM thus represents a doubly stochastic process: the current underlying state in the MC is stochastic, with an added layer of stochasticity in the observed value. Note that each of the hidden states should be able to produce the same symbols, i.e. the same observable values, but in different frequencies [4].

Following the same notation as earlier, let $\mathbf{X}$ be the random vector that represents the underlying MC of length $L+1$, let $\mathcal{T}$ be the transition probability matrix, let $\mathcal{H}$ denote the hidden states, where

$m = \text{card}(\mathcal{H})$, and let π be the initial distribution of states. In addition to this, we need to define some notation for the observable sequence in the HMM. First of all, let the random vector $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_L)^T$ represent an observable sequence of length $L+1$. As earlier, $\mathbf{y} = (y_0, y_1, \ldots, y_L)^T$ will represent a realization of $\mathbf{Y}$. Note that both the hidden and the observed sequence are of length $L+1$, since it is assumed that each state emits exactly one symbol. The *symbol alphabet*, i.e. the set of symbols that can be emitted from each state, will be denoted by $\mathcal{S} = \{s_1, \ldots, s_M\}$, where $M = \text{card}(\mathcal{S})$. The *emission probabilities*, i.e. the probabilities used in each multinomial distribution in each state, are usually arranged in a matrix as well. Denote by $\mathcal{E}$ the emission probability matrix. $\mathcal{E}$ is of dimension $m \times M$, where each row in the matrix contains the emission probabilities (of each of the symbols) for a given state. Written in mathematical terms

$$\mathcal{E}_{ij} = \text{P}(Y_t = j | X_t = i), \;\; i \in \mathcal{H}, j \in \mathcal{S}.$$

Time-homogeneity is also assumed in the stochastic process of emission, which means that these emission probabilities are the same $\forall t \in [0, L]$.

A HMM is completely specified by π, $\mathcal{T}$ and $\mathcal{E}$, where the state space $\mathcal{H}$ and the symbol space $\mathcal{S}$ are indirectly determined by the other quantities [1]. Once these quantities are specified, one can make calculations concerning the HMM. As already noted, calculating the probability of a given realization $\mathbf{x}$ is

$$\text{P}(\mathbf{X} = \mathbf{x}) = \pi(x_0) \prod_{t=2}^{L} \mathcal{T}_{x_{t-1}, x_t}. \tag{1}$$

In a similar fashion, the assumption of independence between states when generating symbols makes it possible to find the probability of the observed sequence simply by taking the product of all appropriate emission probabilities. In mathematical terms, the probability of seeing the sequence $\mathbf{y}$, given the state sequence $\mathbf{x}$, is

$$\text{P}(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) = \prod_{t=1}^{L} \mathcal{E}_{x_t, y_t}. \tag{2}$$

Finally, combining these results, the joint probability of the sequences $\mathbf{x}$ and $\mathbf{y}$ is

$$\text{P}(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = \text{P}(\mathbf{X} = \mathbf{x}) \text{P}(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) = \pi(x_0) \mathcal{E}_{x_0, y_0} \prod_{t=2}^{L} \mathcal{E}_{x_t, y_t} \mathcal{T}_{x_{t-1}, x_t}. \tag{3}$$

Keep in mind that the products in equations (1), (2) and (3) usually become very small in practice. This can give numerical instability when computing the probabilities. A common trick to solve this is to work with the logarithm of the probabilities. In this way, the multiplicative properties become additive, which mitigates some problems that may occur in the calculations. This is especially important when implementing the algorithms that will be discussed later [4].

Note that we here assume that the hidden state sequence $\mathbf{x}$ is known, which is usually not the case in practice. The state sequence needs to be inferred from the observed sequence, which is one of the main problems of a HMM, which will be explained in the following. For a more rigorous treatment of this topic, although in the topic of speech recognition, the reader is referred to [10].

3.3 Basic Problems for HMMs

http://jedlik.phy.bme.hu/ gerjanos/HMM/node6.html

As Yoon [3] and Rabiner [10] point out in their respective articles, there are three issues that need to be resolved in order to be able to use HMMs in practical applications. In the following, these issues will be presented, together with algorithms that are used to address the problems.

A given HMM will in the following be denoted by $\theta = (\pi, \mathcal{T}, \mathcal{E})$. This means that when we refer to a HMM θ, this means that the HMM is fully specified by the three parameters and that the parameters are known. Moreover, for ease of notation, $\mathrm{P}(\mathbf{Y} = \mathbf{y})$ will be denoted as simply $\mathrm{P}(\mathbf{y})$ in the following.

3.3.1 The Evaluation Problem

The first issue that needs to be addressed is: how can the probability $\mathrm{P}(\mathbf{y}|\theta)$ be calculated? That is, how can one find the probability of observing the realization $\mathbf{y}$ from a given HMM θ? If the exact underlying state sequence was known, this would have been easily calculated using equation (3). However, the underlying state sequence cannot be observed, which means that there may exist many different underlying state sequences that can yield the same observed sequence $\mathbf{y}$. Thus, the first solution that comes to mind would be to use the law of total probability to calculate

$$\mathrm{P}(\mathbf{y}|\theta) = \sum_{\forall \mathbf{x}_i \in \mathcal{H}^n} \mathrm{P}(\mathbf{y}, \mathbf{x}_i|\theta) = \sum_{\forall \mathbf{x}_i \in \mathcal{H}^n} \mathrm{P}(\mathbf{x}_i|\theta)\mathrm{P}(\mathbf{y}|\mathbf{x}_i, \theta), \tag{4}$$

where $\mathcal{H}^n$ is the space of all possible state sequences that can yield $\mathbf{y}$. Of course, $\mathcal{H}^n$ can be very large, which makes this calculation computationally infeasible. In fact, $\mathcal{H}^n$ may consist of m^L sequences, since these are all the possible state sequences. Luckily, the *forward algorithm* exists, which uses dynamic programming to solve this issue in a computationally feasible manner. This algorithm has time complexity $\mathcal{O}(Lm^2)$, which is a significant improvement over the exponential solution by straightforward use of equation (4). The forward algorithm is given below.

Algorithm 1 Forward Algorithm

$V \leftarrow \text{array}((m+1) \times (L+1))$
for $i = 0, 1, \ldots, m$ **do**
  $V(i, 0) \leftarrow \pi(h_i)\mathcal{E}_{h_i, y_0}$
end for
for $i = 1, 2, \ldots, L$ **do**
  for $j = 0, 1, \ldots, m$ **do**
    $V(j, i) \leftarrow \mathcal{E}_{h_j, y_i} \sum_{k=0}^{m} V(k, i-1)\mathcal{T}_{h_k, h_j}$
  end for
end for
return $\mathrm{P}(\mathbf{y}|\theta) \leftarrow \sum_{k=0}^{m} V(k, L)$

Note that the forward algorithm gives a *confidence measure* on the observed sequence - how confident are we that the observed sequence could have been generated by the model θ. This problem is also sometimes referred to as the *scoring problem*, since $\mathrm{P}(\mathbf{y}|\theta)$ is a way to score a new observed sequence based on the given HMM [3]. If the calculated probability is lower then some "background distribution" of similar sequences, then the probability of the sequence being emitted by the HMM is low and one would conclude that the HMM is not a good model for the observed sequence. However, if the probability is high, then we can conclude that the HMM can be used in the situation. When this is verified, we tackle the *decoding problem*.

3.3.2 The Decoding Problem

The second issue that needs to be addressed is: what state path $\mathbf{x}$ maximizes the probability of emitting the observed sequence $\mathbf{y}$, i.e. how can the *optimal state path* be found? This follows the principle of maximum likelihood, which says that we should choose the hidden state path that maximizes the likelihood of yielding the observed sequence. Our best bet when trying to infer the hidden state path from the observed sequence is exactly the one that gives the maximum probability of obtaining the observed sequence. In other words, the hidden state sequence that best explains the observed sequence is the optimal state path

$$\mathbf{x}^* = \arg\max_{\mathbf{X}} \mathrm{P}(\mathbf{X}|\mathbf{y}, \theta). \tag{5}$$

Notice that this is equivalent to finding the state sequence that maximizes $\mathrm{P}(\mathbf{X}, \mathbf{y}|\theta)$, because of the definition of conditional probability

$$\mathrm{P}(\mathbf{X}|\mathbf{y}, \theta) = \frac{\mathrm{P}(\mathbf{X}, \mathbf{y}|\theta)}{\mathrm{P}(\mathbf{y}|\theta)}.$$

Again, the first solution that comes to mind would be to compare all possible state sequences m^L, and use equation (3), but this is still computationally infeasible in most cases. Therefore, the *Viterbi algorithm*, which also is based on dynamic programming, exists. The time complexity of the Viterbi algorithm is $\mathcal{O}(Lm^2)$, which is the same as for the forward algorithm. This is a large improvement over the exponential time-complexity of the straightforward solution. The Viterbi algorithm is given below.

Algorithm 2 Viterbi Algorithm

$V \leftarrow \text{array}((m+1) \times (L+1))$
$P \leftarrow \text{array}((m+1) \times (L+1))$
for $i = 0, 1, \ldots, m$ **do**
  $V(i, 0) \leftarrow \pi(h_i)\mathcal{E}_{h_i, y_0}$
  $P(i, 0) \leftarrow -1$
end for
for $i = 1, 2, \ldots, L$ **do**
  for $j = 0, 1, \ldots, m$ **do**
    $V(j, i) \leftarrow \mathcal{E}_{h_j, y_i} \max_{k=0,1,\ldots,m}\{V(k, i-1)\mathcal{T}_{h_k, h_j}\}$
    $P(j, i) \leftarrow \arg\max_{k=0,1,\ldots,m}\{V(k, i-1)\mathcal{T}_{h_k, h_j}\mathcal{E}_{h_j, y_i}\}$
  end for
end for
return $\mathrm{P}(\mathbf{X}|\mathbf{y}, \theta) \leftarrow \max_{k=0,1,\ldots,m} V(k, L)$; $x_L^* = \arg\max_{k=0,1,\ldots,m} V(k, L)$

Note that $\mathbf{x}^*$ is simple to find based on x_L^*; the optimal path $\mathbf{x}^*$ starts at x_L^* and backtracks all the way to x_0^* by using $P(\cdot, \cdot)$. The backtracking ends when the value -1 is found in P, i.e. when the beginning of the sequence has been reached.

Also notice that both the forward and the Viterbi algorithms can easily be log-transformed in order to gain greater numerical stability, by taking logarithms of all values and switching multiplications with additions.

An important point to make before moving forward is that both the evaluation problem and the decoding problem are related to whether or not an observed sequence can be reasonably modeled by means of a given HMM θ. But how can this model be found in the first place? This is what is often referred to as the *training problem*, which will be covered next.

3.3.3 The Training Problem

The third issue that needs to be addressed is: how can the HMM parameters be reasonably chosen based on a set of observed sequences? For example, we have a set of sequences $\chi = \{\mathbf{y}_1, \ldots, \mathbf{y}_G\}$, where each $\mathbf{y}_i$, $i \in [1, G]$ is an observed sequence of symbols, that we want to represent with a HMM. The difficulty that needs to be solved is how the parameters of the HMM can be estimated based on the set χ. This is what is typically called the *training* or *learning* problem, analogously to the need for training any other machine learning model. There exists no optimal way to train the HMM from a limited number of finite observation sequences, but there exists algorithms that can find local maximums in the observation probability. Some examples are the Baum-Welch algorithm, standard

gradient based methods from optimization or simulation with Monte Carlo expectation maximization (MCEM) [3]. For detailed treatments of how the training prblem can be solved, the reader is referred to [10], [11] or [12].

3.4 HMM Variants

There exists many variants of what we will call the standard HMM. For completeness, the standard HMM will be specified first, before the most widely used types of HMMs in bioinformatics will be presented. Note that this list is by noe means exhaustive.

3.4.1 Standard HMM

The *standard HMM* is the simplest form of a HMM, which has the most restricting assumptions. In fact, the assumptions described in section 3.2 all belong to what we will call the standard HMM. Concisely restated, the hidden MC is a discrete-state time-homogeneous first-order MC. Also, often one assumes that the MC is *ergodic*, which means that every state can be reached from any other state in a finite amount of states [10]. Moreover, each state in $\mathcal{H}$ generates exactly one output and can emit every symbol in $\mathcal{S}$. Importantly, the emitted value from each states depends only on that respective state.

All the assumptions made in the simple model restricts the usefulness of the HMM. For example, standard HMMs do not deal well with correlations between residues, since they assume that each residue depends only on one underlying state [1]. Since the Markov property is assumed, an HMM has no way of "remembering" any distant correlation between the symbols in $\mathcal{S}$. Moreover, Choo et. al writes that "the linear nature of HMM also makes it difficult to capture higher-level information or correlations among amino acids" [2]. These are only some examples of how restricted the standard HMMs are.

Despite the fact that the strong assumptions of the standard HMM restrict the flexibility of the model, one can only imagine how the assumptions can be tweaked to create a new type of HMM. Some examples of how changes in the assumptions can mitigate some om the problems of the standard HMM are shown next.

3.4.2 Profile-HMM

*Profile-HMM*s (pro-HMMs) have specific architectures, which make them suitable for modeling sequence profiles [3]. Because of this, in the context of bioinformatics, the topology of pro-HMMs are best explained with multiple sequence alignment in mind. Two simple ways to think of a pro-HMM is as an abstract description of a protein family or a statistical summary of multiple sequence alignment [4]. A pro-HMM is constructed to not contain any cycles. Moreover, a pro-HMM consists of three different types of hidden states: match states (M_k), insert states (I_k) and delete states (D_k). These are used to describe symbol frequencies, insertions and deletions, respectively, at each specific position in a sequence. A great example on how to build a pro-HMM, which helps to clarify the ideas, can be found in [3]. Can I just steel this example? The illustration from the example is shown in 2.

The example shows how a pro-HMM can be built from a multiple sequence alignment. The idea is that the pro-HMM can be used as a profile to classify new observations; either the new observation fits into the family of sequences that have been aligned, or it does not fit and should be aligned with some other profile. The M_k's are used to indicate match states, i.e. they represent the case where a symbol in a new observation matches with the state of the pro-HMM. The emission probabilities from each match state are easily estimated by using the frequencies of each symbol in the match state. The ungapped HMM can represent ungapped sequences. The states I_k and D_k are added so that the HMM can represent sequences that contain gaps, which makes the pro-HMM a much more powerful model compared to the simpler models that describe a multiple sequence alignment, like regular expressions and PSSM, which we will come back to in section 4.3. It is important to note that the delete states D_k

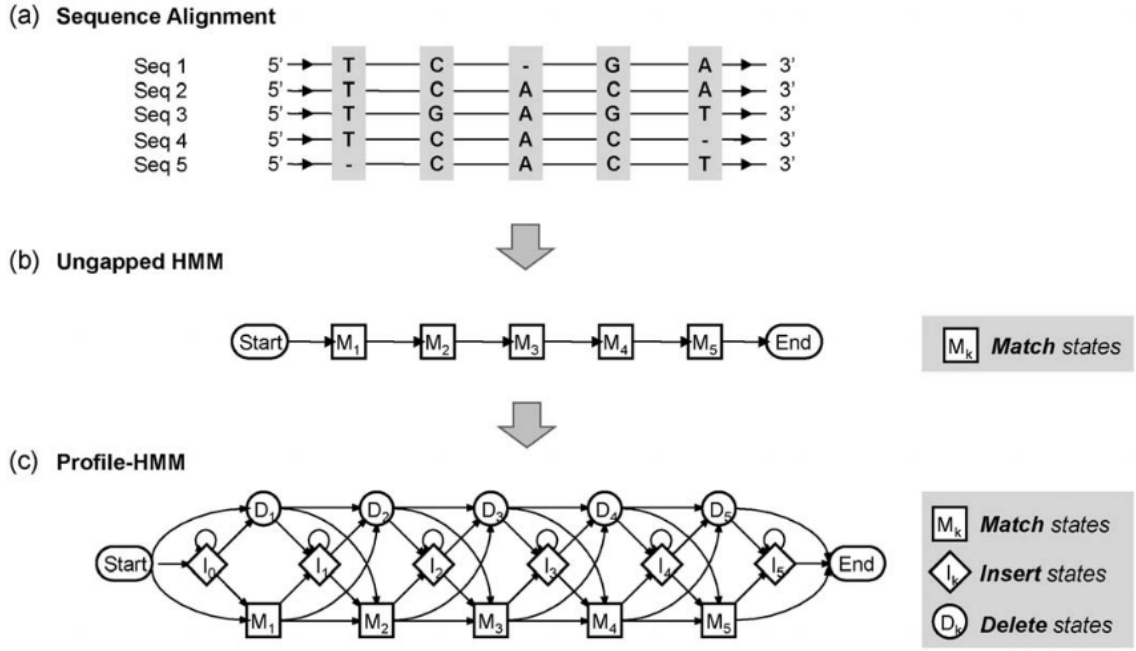

Fig. (2). Profile hidden Markov model. (**a**) Multiple sequence alignment for constructing the profile-HMM. (**b**) The ungapped HMM that represents the consensus sequence of the alignment. (**c**) The final profile-HMM that allows insertions and deletions.

Figure 2: Example from [3]

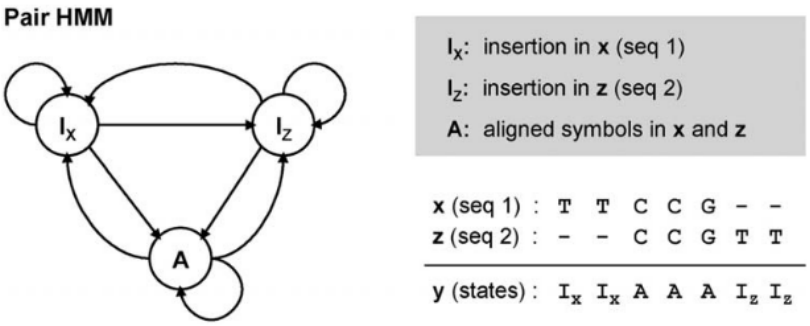

Fig. (3). Example of a pair hidden Markov model. A pair-HMM generates an aligned pair of sequences. In this example, two DNA sequences **x** and **z** are simultaneously generated by the pair-HMM, where the underlying state sequence is **y**. Note that the state sequence **y** uniquely determines the pairwise alignment between **x** and **z**.

Figure 3: Example from [3]

are silent states; they are placeholders for missing symbols in the consensus sequence of the alignment when compared to a shorter new observed sequence.

Note that the pro-HMM parameters can be set in two different ways [13]; The first option is to set the parameters from a pre-aligned set of sequences, simply by counting state transitions and emissions, before converting the counts to probabilities. This is how the parameters in the previous example would be set. The second option is to use a set of sequences that is not aligned before training the pro-HMM. This problem is thus analogous to running a multiple sequence alignment before building the model. In this case, algorithms mentioned in section 3.3.3 are used. Since these algorithms are only local optimizers it is always advisable to build HMMs on pre-aligned data when this is possible [13].

3.4.3 Pair-HMM

In the context of bioinformatics, a *pair-HMM* (p-HMM) is especially useful for finding sequence alignments and evaluating the significance of the alignments [3]. To contrast the standard HMM, a p-HMM generates an aligned pair of sequences, instead of only one sequence. An example is shown in [3] Just steel this example as well!? The illustration from the example is shown in 3. The example presents a very simple p-HMM, with hidden states I_x, I_y and A. This hidden MC produces the two observed sequences denoted by $\mathbf{x}$ and $\mathbf{z}$, where I_x emits an unaligned symbol in sequence $\mathbf{x}$ and I_z emits an unaligned symbol in $\mathbf{z}$. Additionally, state A generates an aligned pair of two symbols, i.e. it inserts two identical symbols, each inserted into each of the observed sequences.

Since there is a one-to-one relationship between a hidden state sequence $\mathbf{x}$ of a p-HMM and the alignment between two observed sequences, the alignment problem reduces to finding the optimal state path in the hidden MC. This can be found by a simple variation of the Viterbi algorithm and has time complexity $\mathcal{O}(L_{\mathbf{a}}L_{\mathbf{b}})$, where $\mathbf{a}, \mathbf{b}$ are the two sequences that are aligned and $L_{\mathbf{a}}, L_{\mathbf{b}}$ are their respective lengths. The p-HMM model is an improvement over the classical sequence alignment methods, since it can be used to compute the alignment probability of the pair of sequences indepently of a specifif alignment. A problem in the classical methods is how one should choose a punctuation scheme in order to find a biologically meaningful optimal alignment of the sequences. In cases where this is difficult to choose, it is more meaningful to compute the probability that the sequences are related, which a p-HMM makes possible [3]. This probability can be calculated by a slight modification of the forward algorithm. The improvement that a p-HMM gives with respect to pairwise sequence alignment will be further discussed in section 4.1.

3.4.4 Context-Sensitive HMM

As noted in section 3.4.1, the standard HMM cannot properly model correlations between residues, which makes the model unsuitable for several applications in biology, for example when dealing with RNA sequences [3]. In order to apply the HMM-methodology successfully to such situations, one needs to extend the standard HMM to allow for pairwise correlations between non-adjacent symbols in the observed sequence. This is where the *context-sensitive HMM*s (cs-HMMs) are appropriately applied.

The main difference between cs-HMMs and the standard HMMs is that the former can use information about earlier emissions to adjust the emission probabilities at future states. This information is referred to as the "context". Thus, it is possible to model correlation between non-adjacent states in a cs-HMM. These HMMs use three different types of states: single-emission states (ξ_i), pairwise-emission states (ϕ_i) and context-sensitive states (γ_i). ξ_i are very similar to the normal states in ordinary HMMs, since they have usual emission probabilities and do not use any additional information for emitting symbols. ϕ_i have fixed emission probabilities, and, in addition, they store the symbols that are admitted in memory. γ_i first retrieves the emitted symbol from ϕ_i, before adjusting its emission probabilities based on this retrieved context. Thus, we can state the emission probability for this context-sensitive state as

$$\mathrm{P}(y_j|y_k, x_k, x_j) = \mathrm{P}(y_j \text{ emitted at } x_j \text{ given that } y_k \text{ was emitted at } x_k), \quad j > k$$

where y_j is the emitted symbol at a context-sensitive state x_j, y_k is the emitted symbol at a pairwise-emission x_k and $j > k$, i.e. the MC transitions to the context-sensitive state some time after the pairwise-emission state. Using the fact that y_k is independent of x_j, the joint emission probability of y_k and y_j can be calculated

$$\mathrm{P}(y_k, y_j|x_k, x_j) = \mathrm{P}(y_k|x_k)\mathrm{P}(y_j|y_k, x_k, x_j) = \mathcal{E}_{x_k,y_k}\mathrm{P}(y_j|y_k, x_k, x_j).$$

Thus, using pairs of states (ϕ_i, γ_i) allow modeling of pairwise symbol correlation by specifying the emission probabilities that appear in equation (3.4.4). Notice that these two states always exist in pairs, since ϕ_i are needed to describe the emission probabilities at γ_i. Moreover, the model is built such that it is never possible to transition to a context-sensitive state before transitioning to the corresponding pairwise-emission state [3].

A simple example of a cs-HMM, which is a modified version of an example appearing in [3], is given in figure 4. The model has three different states: unconsistent notation! S_1 is the only single-emission state, P_1 is the only pairwise-emission state and C_1 is the only context-sensitive state. The MC first transitions to P_1 and emits one or more symbols which are stored in a queue. When the MC eventually transitions to C_1, a symbol is retrieved from the queue and the emission probabilities in C_1 are adjusted in such a way that the same symbol is emitted. Moreover, in this model, the transition probabilities are adjusted such that the MC transitions only to C_1 as long as the queue is not empty. When the queue is empty, the MC transitions to the end state and the simulation is over. Thus, this simple example can be used to create repeating sequences of two different formats. Firstly, the sequence

$$\mathbf{y} = (y_1, y_2, \ldots, y_{(L+1)/2}, y_1, y_2, \ldots, y_{(L+1)/2})^T,$$

of even length $L+1$ can be generated by the model. In this case, the first $(L+1)/2$ underlying states are P_1 and the next $(L+1)/2$ are C_1. Secondly, the sequence

$$\mathbf{y} = (y_1, y_2, \ldots, y_{L/2}, y_0, y_1, y_2, \ldots, y_{L/2})^T,$$

of odd length $L+1$ can be generated by the model. In this case, the first $L/2$ underlying states are P_1, the underlying state for the $L/2+1$ emitted symbol y_0 is S_1 and the underlying states for the

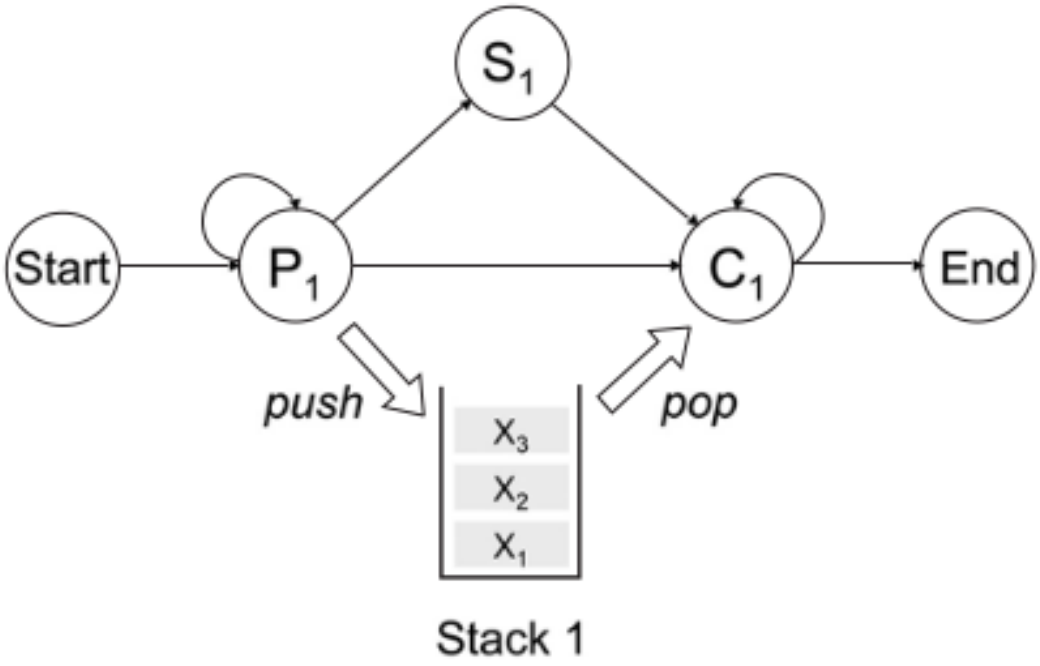

Fig. (4). A context-sensitive HMM that generates only symmetric sequences, or palindromes.

Figure 4: Example from [3]. I changed the example a bit, generating repeating sequences instead of for palindromes, as Yoon did. Changing the stack (LIFO) for a queue (FIFO) gives repeating patterns in the sequences, instead of palindromes. .

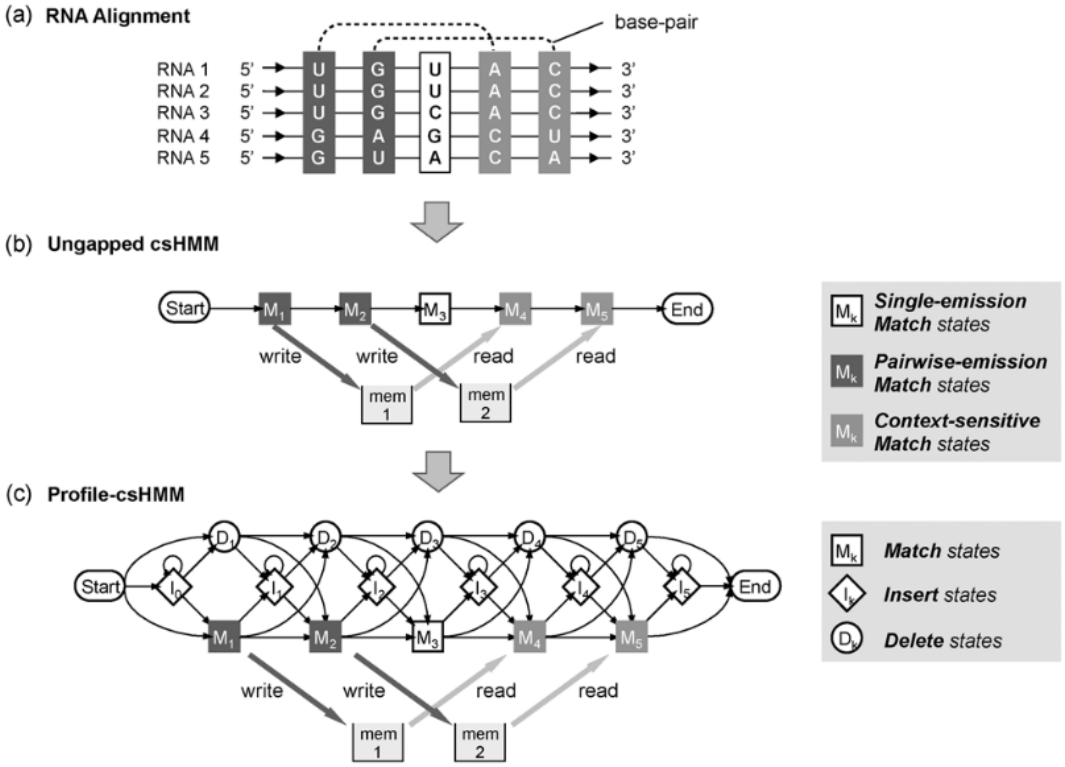

Fig. (5). Constructing a profile-csHMM from a multiple RNA sequence alignment. (**a**) Example of an RNA sequence alignment. The consensus RNA structure has two base-pairs. (**b**) An ungapped csHMM constructed from the given alignment. (**c**) The final profile-csHMM that can handle symbol matches, insertions, and deletions.

Figure 5: Example from [3]. Tenker at jeg stjeler dette eksempelet, da det er veldig godt (eller bare henviser), men lager de to andre litt selv etter hvert!

last $y_{L/2}$ are C_1. Notice that changing the FIFO queue for a LIFO enables generation of palindromes instead of repeating sequences from the model [3].

Recall that pro-HMMs were used to make profiles that represent multiple alignments of sequences, which in the context of biology can be DNA or proteins (sequences of aminoacids). Imagine that we want to make profiles that take column-wise correlations in the multiple alignment into account. This can be done relatively easily by combining the ideas of the pro-HMMs and the cs-HMMs, into what is called *Profile Context-Sensitive HMM*s (pro-cs-HMMs). Like for the pro-HMMs studied earlier, the pro-cs-HMMs contain match states (M_k), insert states (I_k) and delete states (D_k). The new addition is that the match states can be of three different types ($\xi_i^m, \phi_i^m, \gamma_i^m$), to account for symbols that are pairwise correlated. The single-emission match states ξ_i^m are used to represent columns in the multiple alignment that are uncorrelated with the others, while the pairing of (ϕ_i^m, γ_i^m) are used to model pairwise correlation between symbols in different columns. An example, taken from [3], is shown in figure 5 Not described yet.

For a detailed treatment of cs-HMMs, the reader is referred to [14].

4 Applications in Bioinformatics

Some more applications listed on the Wikipedia page: `https://en.wikipedia.org/wiki/Hidden_Markov_model#Applica`

4.1 Pairwise Sequence Alignment

Pairwise sequence alignment is done to infer functional, structural and/or evolutionary relationships between two sequences. Such an alignment can be done both locally and globally and there exists a variety of methods of doing it. Optimal algorithms exist, such as the Needleman-Wunsch algorithm for optimal global alignment and the Smith-Waterman algorithm for optimal local alignment. These algorithms are based on dynamic programming and can be time consuming to use. Therefore, algorithms based on heuristics were developed, where the two most popular algorithms are called BLAST and FASTA. All the mentioned algorithms work well for highly similar sequences, but produce mediocre results for highly divergent sequences [2]. Profile based analysis was developed to improve these results, in which HMMs play a crucial role.

A big issue when it comes to finding the best alignment between sequences using the previously mentioned algorithms is that one has to define a reasonable scoring scheme for ranking the possible alignments. It can be difficult to define this scoring scheme, since different scoring schemes perform better or worse in different applications, depending on how similar the sequences are. Thus, the major drawback of these algorithms is the dependence on the scoring scheme, which to a large degree dictates how the alignment is produced, which in turn limits what type of relationship one is able to find between sequences. The p-HMMs introduce an alternative method of finding relationships between a pair of sequences, which distances itself from ad-hoc scoring schemes.

When using p-HMMs the pairwise sequence alignment problem is tackled as a stochastic process, as used by Smith et al [5] Skumles artikkelen og sjekk at det faktisk stemmer! Because the p-HMM generates pairs of observations, it can be used to calculate the relationship between two sequences independent of a specific alignment, simply by using the forward algorithm [2].

Goal: Infer functional similarity Could be merged with "Prediction of Function" perhaps?.

Pair Hidden Markov models on tree structures (PHHMTS) can be used for aligning trees. Since most RNA secondary structures can be represented as trees, this provides a useful framework for aligning RNA sequences.

4.2 Multiple Sequence Alignment

Multiple sequence alignment is commonly used to find conserved regions in a group of sequences and/or predicting protein structures. In contrast to the pairwise sequence alignment problem, there does not

exist any optimal algorithms to solve the multiple alignment problem. However, there exists a variety of heuristic methods for solving it. One possible solution is to reduce the multiple alignment problem down to a set of pairwise comparisons between the sequences, in an orderly fashion, in order to end up with the multiple alignment in the end. Here one can choose to use either the optimal algorithms or the heuristic algorithms when performing the pairwise alignment. One type of methods that employs this paradigm is referred to as progressive sequence alignment. More details on a progressive algorithm that uses Needleman-Wunsch for the pairwise alignment can be found in [6]. Some commonly available implementations of this solution include the T-Coffee and the Clustal package [8]. Another type of methods that uses this paradigm is referred to as iterative alignment. These methods are slightly different than the progressive methods, since they allow realignment of the pairwise sequences during multiple iterations, where the progressive methods depends highly on the initial pairwise alignment of the first two sequences [9]. One popular openly available implementation of an iterative method is MUSCLE [8].

As in the case of pairwise alignment, HMMs provide powerful alternatives to these other methods for multiple alignment. In this problem, pro-HMMs have been particularly successful [2]. Continuing to read in Choo.

Pro-HMMs have been applied to this problem with much success. Is connected to the Viterbi algorithm also.

Må virkelig få strukturert dette skikkelig! Vanskelig å dele opp alt slik jeg tenkte opprinnelig! Mulig jeg bør samkjøre teori og applications i større grad, selv om jeg egentlig ville ha dem separert!

`http://pfam.xfam.org/`

Pro-HMMs are defined in order to solve the problem of multiple alignment of sequences. All new sequences can efficiently be aligned against this pro-HMM. They also facilitate quick assignment of protein function. These pro-HMMs are commonly regarded as a summary of a multiple alignment of sequences or as a model for a family of such sequences [4].

Since pro-HHMs are an abstract representation of a multiple alignments, they can be used to produce pairwise or multiple alignments as well. Thus, aligning a sequence with a pro-HHM is equivalent to aligning the sequence to many, many other sequences, which where used to establish the pro-HMM in the first place [4]. E.g. PFAM is a free online repository, that store pro-HMMs of many known protein families. PROSITE is another database that stores pro-HMMs.

The pro-HMMs are very useful in the context of searching for homologues. Given a pro-HMM that represents a family of sequences, one can use this model to search in a database of sequences, in order to find additional homologues to the family. Similary, given a database of pre-built pro-HMMs (like PROSITE or PFAM), we can look for matching profiles to a symbol sequence. Thus, we can use this database to classify and annotate the given sequence.

Pro-HMMs can also be used to compare two different multiple sequence alignments (sequence profiles). This can be beneficial for detecting remote homologues [3]. "These profile HMMs are also what makes it possible to assign protein function quickly, and can be regarded both as a summary of a multiple alignment and as a model for a family of sequences" [4].

Many multiple sequence alignment algorithms also use p-HMMs [3]. The most widely used approach based on p-HMMs is called progressive alignment.

4.3 Motif Representation (or Identification?)

A motif is a recurring pattern in DNA or proteins, which is assumed to be related to biological function. Thus a motif can be a sequence of nucleotides or of amino-acids. Finding such motifs is interesting to a biologist because *transcription factor binding sites* (TFBS) appear as such motifs in sequences. These are regulatory sequences that control gene transcription, which is important information for a biologist because they repress or promote the expression of many other genes [4].

There are several ways of representing such sequence motifs, where the performance of each method generally depends on what type of motifs one wants to represent. For shorter, ungapped motifs of fixed length, methods like *consensus sequences*, *regular expressions* (RegEx), *sequence logos* and *position*

specific scoring matrix (PSSM), in ascending order of complexity, are commonly used. Note that these can also be seen as different ways of visualizing or representing multiple sequence alignments, which are useful tools in practice, since the alignments themselves are not very easy in use.

The drawbacks of these methods are that they do not work well for longer motifs with variable length gaps. This is where the profile-HMMs shine in comparison. The HMMs work well for all types of motifs, including the short and fixed length motifs, but they are more complicated models. Thus, even though the HMMs always will produce good results, given that they can be built correctly in the specific case, one should not always use these models because of their complexity. Always keep the principle of parsimony in mind; for competing explanations, or models, where all are reasonable, one should always choose the simplest, with the least amount of assumptions. Hence, even though the HMMs are great tools also for motif identification, they should be used when appropriate.

But how can the HMMs be used to represent motifs with variable length gaps? The profile-HMMs have gained much traction for this problem, since it yields very reliable results. How they work can be seen easily from the example presented about the profile-HMMs in the section on theoretical background, since the example talks about how to represent a multiple sequence alignment as a profile-HMM. The same principle explained in the example can be used when working with much larger and more complicated alignments.

After constructing these representations, they can be used to search for sequences that belong to the same family as the aligned sequences, since sequences that have the same motifs may share the same functions. Good idea to have this as a different section compared to the multiple sequence alignment-section?

Note that the problem of finding the motifs is not discussed in detail here, as this is a much more complicated problem compared to simply representing the motifs. However, this problem is highly similar to multiple local alignment, which has been explained in detail earlier. An example of such an algorithm is given in chapter 10.3 in [4], which is a variant of the Gibbs sampling algorithm based on PSSMs.

EXTRA, if needed: Wikipedia: When a motif appears in the exon of a gene, it can encode a "structural motif" of a protein. Outside of gene exons, there exists regulatory sequence motifs

4.4 Prediction of Function

HMMs are used to make probabilistic statements about the function of proteins and thus, they can also be used to assign proteins to families of unknown function [4]. Is this a step in Genome Annotation? Perhaps they should be merged then! Or multiple sequence alignment?

4.5 Segmentation

Segmentation is about defining exact boundaries between distinct regions with different chemical properties. Moreover, segmentation is about defining larger sequences of heterogeneous nucleotide in genome and to identify biological features that are responsible for the heterogeneity that is observed [4]. Some classical methods are ...

HMMs can be used effectively for segmentation as well... They can help to define regions of gene and protein sequences with various chemical properties [4].

Example 4.2 in [4] shows an example of segmentation using HMMs.

In the setting of segmentation, the hidden states are interpreted as different types of the sequence and the hidden alphabet is typically very small. The underlying MC is cyclic in this case, allowing returns to the same state, i.e. to the same type of sequence, many times during the simulation [4].

4.6 Protein Homology Detection

Goal: Determine which proteins are derived from a common ancestor.

Not sure if the following fits in here. Characterize sets of homologous proteins (gene families) based on common patterns in their sequence. This allows us to determine if a new protein belongs to a certain family or not [4]. In thi case HMMs can be used to provide a more flexible characterization of sequence patterns. This, in comparison to the simpler way of using multiple alignment to construct a PSSM, also works well for cases which include gaps of variable length, which is a case where the multiple alignment method does not work well [4]. This type of homology detection is done with profile HMMs. "profile HMMs encode position-specific information about the frequency of particular amino acids as well as the frequency of insertions and deletions in the alignment. They are constructed from multiple alignments of homologous sequences" [4]. Since pHHMs are an abstract representation of a multiple alignments, they can be used to produce pairwise or multiple alignments as well. Thus, aligning a sequence with a pHHM is equivalent to aligning the sequence to many, many other sequences, which where used to establish the pHMM in the first place.

Feature-based Profile HMMs can be used to improve the performance of remote protein homology detection [3].

This is just an application of multiple sequence alignment, no? So perhaps those two can be merged?

4.7 Gene Prediction / Genomic Annotation (?) / Gene Finding (?)

Is Gene Prediction and Genomic Annotation (and Gene Finding) the same problem? Two different words for the same name or are they slightly different problems, with different goals in mind?

HMMs are employed to find eukaryotic genes and to find pseudogenes, which look like functioning genes except for some misplaced stop codons. They are very useful for these problems because of their flexibility [4].

If the hidden states in the HMM can be inferred, the genome can be better annotated or one can understand the dynamics of the genome better [4].

Genomic annotation: Generalized HMMs [2].

Eukaryotic genes can be modeled using HMMs [3].

Pair HMMs can be used for gene prediction [3].

4.8 Protein Sequence Classification

Profile HMMs (analagously to multiple sequence alignment).

4.9 Protein Structure Prediction

Is connected to homology detection. More in [2].

Profile HMMs can be used to "model sequences of protein secondary structure symbols: helix (H), strand (E) and coil (C)" [3]. This model can be used to recognize the three-dimensional fold of new protein sequences based on their secondary protein structure predictions.

Also page 79 in Introduction to Mathematical Methods in Bioinformatics.

Eddy 1998 mentions this use of profile-HMMs as well. It has a part on protein fold recognition that could be interesting to use!!

4.10 Base-calling

[3]

4.11 Modeling DNA Sequencing Errors

[3]

4.12 ncRNA Identification

[3]

4.13 RNA Structural Alignment

As explained when talking about context-censitive HMMs (csHMMs), profile-csHMMs can be used to perform structural alignment of RNA and performing similarity searches, analagously to how the profile-HMMs can be used to perform multiple alignment of DNA or proteins and performing similarity searches in these cases. RNA sequence analysis is often of high computational complexity, because the alignment algorithms have to deal with base-pair correlations in the sequences that may be very complicated. More about this problem in [3].

5 Examples in R

5.1 CG-islands and the "Fair Bet Casino"

"An introduction to bioinformatics algorithms" - Jones and Pevzner Page 388.
Hay una tarea sobre esto tambien, descargado en pdf en "Books/bioinfo".
The reference list should be alphabetically ordered later!

References

[1] Sean R. Eddy (2004) *What is a hidden Markov model?*, Nature Biotechnology, Volume 22, Number 10, 1315-1316.

[2] Khar H. Choo, Joo C. Tong, Louxin Zhang (2004) *Recent Applications of Hidden Markov Models in Computational Biology*, Geno. Prot. Bioinfo, Vol. 2, No. 2, 84-96.

[3] Byung-Jun Yoon (2009) *Hidden Markov Models and their Applications in Biological Sequence Analysis*, Bentham Science Publishers Ltd., Current Genomics, Vol. 10, No. 6, 402-415.

[4] Nello Christianini, Matthew W. Hahn (2006) *Introduction to Computational Genomics: A Case Studies Approach*, Cambridge University Press.

[5] Smith, L., et al. (2003) *Hidden Markov models and op- timized sequence alignments*, Computational Biology and Chemistry, Vol. 27, 77-84.

[6] Feng DF, Doolittle RF (1987) *Progressive sequence alignment as a prerequisite to correct phylogenetic trees*, J Mol Evol. Vol. 25, No. 4, 351-360. doi: 10.1007/BF02603120. PMID: 3118049.

[7] Mark A. Pinsky, Samuel Karlin (2011) *An Introduction to Stochastic Modeling*, Fourth Edition, Elsevier

[8] DENNE MÅ FØRES OPP ORDENTLIG!

[9] DENNE MÅ FØRES OPP ORDENTLIG!

[10] Lawrence R. Rabiner (1989) *A tutorial on hidden Markov models and selected applications in speech recognition*, Proceedings of the IEEE, Vol. 77, No. 2, 257-286

[11] Alexander Isaev (2006) *Introduction to Mathematical Methods in Bioinformatics*, Springer-Verlag

[12] R. Durbin, S. Eddy, A. Krogh, G. Mitchison (1998) *Biological Sequence Analysis*, Cambridge University Press

[13] Sean R. Eddy (1998) *Profile hidden Markov models*, Bioinformatics Review, Volume 14, Number 9, 755-763.

[14] Byung-Jun Yoon, P. P. Vaidyanathan (2006) *Context-Sensitive Hidden Markov Models for Modeling Long-Range Dependencies in Symbol Sequences*, IEEE TRANSACTIONS ON SIGNAL PROCESSING, Volume 54, Number 11, 4169-4184.