Hidden Markov Models

Monica Franzese and Antonella Iuliano, Institute for Applied Mathematics "Mauro Picone", Napoli, Italy

Introduction

Hidden Markov models (HMMs), named after the Russian mathematician Andrey Andreyevich Markov, who developed much of relevant statistical theory, are introduced and studied in the early 1970s. They were first used in speech recognition and have been successfully applied to the analysis of biological sequences since late 1980s. Nowadays, they are considered as a specific form of dynamic Bayesian networks, which are based on the theory of Bayes. HMMs are statistical models to capture hidden information from observable sequential symbols (e.g., a nucleotidic sequence). They 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 a HMM, the system being modelled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters. A good HMM accurately models the real world source of the observed real data and has the ability to simulate the source. A lot of Machine Learning techniques are based on HMMs have been successfully applied to problems including speech recognition, optical character recognition, computational biology and they have become a fundamental tool in bioinformatics: for their robust statistical foundation, conceptual simplicity and malleability, they are adapted fit diverse classification problems. In Computational Biology, a hidden Markov model (HMM) is a statistical approach that is frequently used for modelling biological sequences. In applying it, a sequence is modelled as an output of a discrete stochastic process, which progresses through a series of states that are 'hidden' from the observer. Each such hidden state emits a symbol representing an elementary unit of the modelled data, for example, in case of a protein sequence, an amino acid. In the following sections, we first introduce the concepts of Hidden Markov Model as a particular type of probabilistic model in a Bayesian framework; then, we describe some important aspects of modelling Hidden Markov Models in order to solve real problems, giving particular emphasis in its use in biological context. To show the potentiality of these statistical approaches, we present the stochastic modelling of an HMM, defining first the model architecture and then the learning and operating algorithms. In this work we illustrate, as example, applications in computational biology and bioinformatics and, in particular, the attention is on the problem to find regions of DNA that are methylated or un-methylated (CpG-islands finding).

Stochastic Process

The basic idea of the process modelling is to construct a model of a process starting from a set of sequences of events typically generated by the process itself. Subsequently, the model could be also used to discover properties of the process, or to predict future events on the basis of the past history. From a general point of view, a model can be used for three main purposes: describing the details of a process, predicting its outcomes, or for classification purposes, i.e., predicting a single variable k, which takes values in a finite unordered set, given some input data $x=(x_1,\ldots,\ x_n)$. Against the deterministic model, which predicts outcomes with certainty, with a set of equations that describe the system inputs and outputs exactly, a stochastic model represents a situation where uncertainty is present. In other words, it's a model for a process that has some kind of randomness. The word "stochastic" derives from the Greed and means *random* or *chance*. A deterministic model predicts a single outcome from a given set of circumstances. A stochastic process is a sequence of events, in which the outcome at any stage depends on some probabilities. It means that a stochastic model predicts a set of possible outcomes weighted by their likelihoods, or probabilities. In modelling stochastic processes the key role is played by time; in fact, the stochastic model is a tool for predicting probability distributions of potential outcomes by allowing a random variation in its inputs over time. A stochastic process is defined as a collection of random variables $X=\{X_t{:}t\in T\}$ defined on a common probability space, taking values in a common set S (the *state space*), and indexed by a set T, often either N or $[0,\ \infty)$ and thought of as *time* (discrete or continuous respectively) (Oliver, 2009).

Markov Processes and Markov Chains

Important classes of stochastic processes are Markov processes and Markov chains. A Markov process is a process that satisfies the Markov property (*memoryless*), i.e., it does not have any memory: the distribution of the next state (or observation) depends exclusively on the current state. Formally, a stochastic process $X(t)$ is a Markov process, if it has the following properties:

1. The number of possible outcomes or states is finite.
2. The probabilities are constant over time.
3. It satisfies *memoryless* property.

$$P\{X(t_{n+1})=x_{n+1}|X(t_n)=x_n\ldots..X(t_1)=x_1\}=P\{X(t_{n+1})=x_{n+1}|X(t_n)=x_n\}$$

for any choice of time instants t_i, con $\mathrm{i}=1,\ldots,\ n$ where $t_j>t_k$ for $j>k$.

Markov chain is a specific Markov process with a finite or countable state-space. Considering a set of states, $S = \{s_1, \dots ., s_r\}$, we describe Markov chain as a process that starts in one of these states and moves successively from one state to another. Each move is called *step*. If the chain is currently in state s_i, then it moves to state s_j at the next step with a probability denoted by p_{ij}; this probability does not depend upon which states the chain was in before the current state. The probabilities p_{ij} are called transition probabilities and are defined as the probabilities that the Markov chain is at the next time point in state j, given that it is at the present time point at state i. The matrix P with elements p_{ij} is called the transition probability matrix of the Markov chain. Since the state space is countable (or even finite), we can use the integers Z or a subset such as Z_+ (non-negative integers), the natural numbers $N = \{1, 2, 3, \dots\}$ or $\{0,1,2,\dots, m\}$ as the state space. We refer to Markov chains as *time homogeneous* or having stationary transition probabilities. Then, the probability of the transition from i to j between time point n and $n+1$ is given by conditional probability function $P(X_{n+1}=j \mid X_n=i)$. We will assume that the transition probabilities are the same for all time points so that there is no time index needed on the left hand side. Given that the process X_n is in a certain state, the corresponding row of the transition matrix contains the distribution of X_{n+1}, implying that the sum of the probabilities over all possible states equals one. The transition probability matrix (see **Table 1**) contains a conditional discrete probability distribution on each of its rows. Formally,

$$p_{ij} \geq 0, \ \forall\, i, j \in S$$

$$\sum\nolimits_{j \in S} p_{ij} = 1, \ \forall\, i \in S$$

In general, we define Markov chain as kth-order Markov model, when the probability of X_{j+k} conditioned on all previous elements in the sequence is identical to the probability of X_{j+k} conditioned on the previous k elements only:

$$P(X_{j+k} | X_1, \dots, X_j) = P(X_{j+k} | X_{j+k-1}, \dots, X_j)$$

In particular, in zeroth-order Markov chain $k=0$. It means that the variables X_i are independent, i.e., $P(X_j \mid X_1,\dots,X_{j-1}) = P(X_j)$.

It is always possible to represent a time-homogeneous Markov chain by a transition graph. Then, any transition probability matrix P (see **Table 1**) can be visualized by a transition graph, where the circles are nodes and represent possible states s_i, while edges between nodes are the transition probabilities p_{ij} (see **Fig. 1**).

Markov chains are probabilistic models, which can be used for the modelling of sequences given a probability distribution and then, they are also very useful for the characterization of certain parts of a DNA or protein string given, for example, a bias towards the AT or GC content. DNA sequences consist of one of four possible bases {A, T, C, G} at each position. Each sequence is always read in a specific direction, from the 5′ to the 3′ end. Each place in the sequence can be thought of as a state space, which has four

Table 1 Transition probability matrix

		Time t+ *1*				
	State	S_1	S_2	S_3	S_4	*Sum*
Time t	S_1	p_{11}	p_{11}	p_{12}	p_{13}	1
	S_2	p_{21}	p_{21}	p_{22}	p_{23}	1
	S_3	p_{31}	p_{31}	p_{32}	p_{33}	1
	S_4	p_{41}	p_{41}	p_{42}	p_{43}	1

$p_{ij} = p(q_{t+1} = S_j | q_t = S_i)$

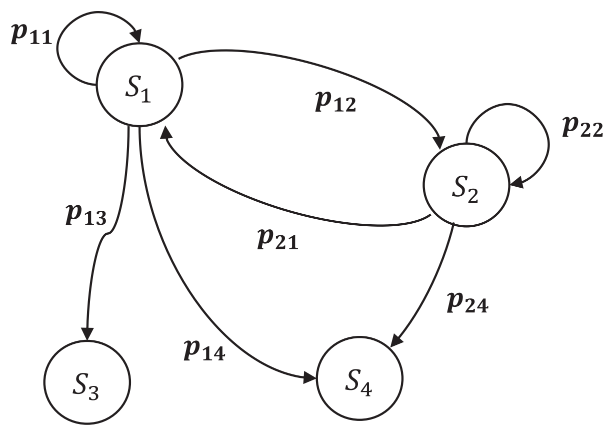

Fig. 1 Transition probability graph.

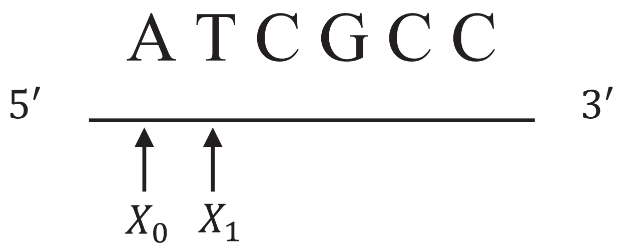

Fig. 2 Nucleotide sequence (5′ to 3′) as a first-order Markov Chain.

Table 2 Transition probability matrix for the ATCG sequence

$$\begin{bmatrix} A & 0.3 & 0.2 & 0.2 & 0.3 \\ T & 0.1 & 0.2 & 0.4 & 0.3 \\ C & 0.2 & 0.2 & 0.2 & 0.4 \\ G & 0.1 & 0.8 & 0.1 & 0 \end{bmatrix}$$

possible states {A, T, C, G}, where each state of the nucleotide is sequentially dependent on the nucleotide adjacent and immediately upstream of it, and only that nucleotide. Each position in the sequence can be represented with a random variable X_0, $X_1 \ldots, X_n$ that takes a value of one of the states in the state space for a particular place in the sequence. In a Markov Chain of zero order, the current state (or nucleotide) is totally independent of the previous state, so it's no memory and every state is untied. For the first order Markov Chain the case is different, because the current state actually depends only on the previous state. In the **Fig. 2** a first-order Markov chain is shown: the sequence goes in the 5′ to 3′ direction and $X_0 = A$, $X_1 = T$ and so forth. In this case, we can express the probability of this sequence using the Markov Property as follows:

$$P(X_5 = C | X_4 = C,\ X_3 = G,\ X_2 = T,\ X_1 = T,\ X_0 = A) = P(X_5 = C | X_4 = C)$$

and its transition probability matrix is shown in **Table 2**.

Hidden Markov Model

Now, we can define the Hidden Markov Models as probabilistic models, in which sequences are generated from two coexistent stochastic processes: the process of moving between states and the process of emitting an output sequence, characterized by Markov property and the output independence. The first one is a Markov model represents by a finite set of states, which generates the sequence of states of variables, specified by the initial state probabilities and state transition probabilities between variables; the second is characterized by the emission of one character of a given alphabet from each state, with a probability distribution that only depends from the state. The sequence of state transitions is a hidden process; it means that the variable states cannot be directly observed but they are observed through the sequences of emitted symbols, therefore the name *Hidden Markov Model*. Then, an Hidden Markov Model is defined by states, state probabilities, transition probabilities, emission probabilities and initial probabilities. They constitute the architecture of an HMM. Formalizing the definition, an HMM is a quintuple (S, V, π, A, B), characterized by the following elements (Rabiner and Juang, 1986a):

- $S = \{S_1, \ldots .., S_N\}$ is the set of *states*, where N is the number of states. The triplet (S, π, A) represents a Markov chain; the states are hidden and we never observe them directly.
- $V = \{v_1, \ldots .., v_M\}$ is the *vocabulary*, the set of symbols that may be emitted.
- $\pi : S \rightarrow [0,1] = \{\pi_1, \ldots \ldots, \pi_N\}$ is the *initial probability distribution* on the states. It gives the probability of starting in each state. We expect that

$$\sum\nolimits_{s \in S} \pi(s) = \sum_{i=1}^{N} \pi_i = 1$$

- $A = (a_{ij})_{i \in S,\ j \in S}$ is the *transition probability* of moving from stateS_i to state S_j. We expect that $a_{ij} \in [0,1]$ for each S_i and S_j, and that $\sum\nolimits_{i \in S} a_{ij} = 1$, for each S_j.
- $B = (b_{ij})_{i \in V,\ j \in S}$ is the *emission probability* that symbol v_i is seen when we are in state S_i.

HMMs provide great help when there is the need of modelling a process in which there is not a direct knowledge about the state in which the system is. The key idea is that an HMM is a sequence "generator". In general, we talk about emission of observable events because we can think of HMMs as generative models that can be used to generate observation sequences. Algorithmically, a sequence of observations $O = o_1, \ldots\ldots, o_T$, with $o_t \in V$ can be generated by an HMM, described by the algorithm in **Fig. 3** (Rabiner and Juang, 1986a). Two assumptions are made by the model. The first is called the Markov assumption and represents the memory of the model; it means that the current state is dependent only on the previous state; formally:

$$P\left(q_t | q_1^{t-1}\right) = P(q_t | q_{t-1})$$

The second is the independence assumption, i.e., the output observation at time t is dependent only on the current state and it is independent of previous observations and states:

$$P\left(o_t | o_1^{t-1}, q_1^t\right) = P(o_t | q_t)$$

A simple HMM for generating a DNA sequence is specified in **Fig. 4**. In this model, state transitions and their associated probabilities are indicated by arrows; and symbol emission probabilities for A, C, G, T at each state are indicated below the state. For clarity, we omit the initial and final states as well as the initial probability distribution. For instance, this model can generate the state sequence given in **Fig. 5** and each state emits a nucleotide according to the emission probability distribution. The logical idea of an HMM suggests the potential ability of this approach for modelling problems in computational biology. In particular, Baldi and Brunak (1998a) define three main groups of problems in computational biology for which HMMs have been useful. The first problem is the multiple alignments of DNA sequences, which is more difficult using a dynamic programming approach. The second is to discover periodic sequences within specific regions of biological data from the knowledge of consensus patterns. The third is the problem to classify each nucleotide according to which structure it belongs. HMMs have also been used in protein

```
– Set t=1
– Choose an initial state $S_i(t)$, according to the initial state probability distribution $\pi$
– while $t \le T$ do
      Choose $O_t = V_k$, according to the emission probability $b_{ik}$ in state $S_i$
      if $t \le T$
            Transit to a new state $S_j(t+1)$, according to the transition probability
            distribution $a_{ij}$ for state $S_i$
            Set $S_i = S_j$
      end if
      Set $t = t + 1$
  end while
```

Fig. 3 Generator algorithm for a sequence of observations by an Hidden Markov Model.

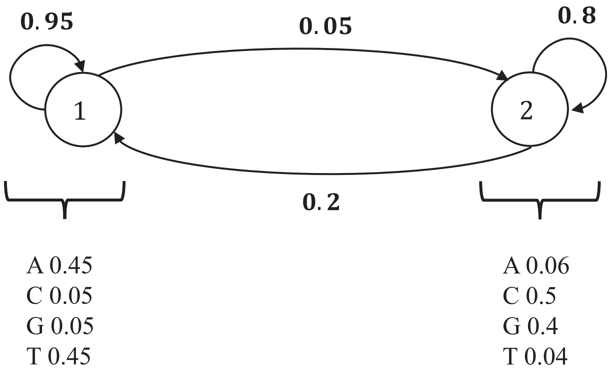

Fig. 4 A Hidden Markov model generator for DNA sequence.

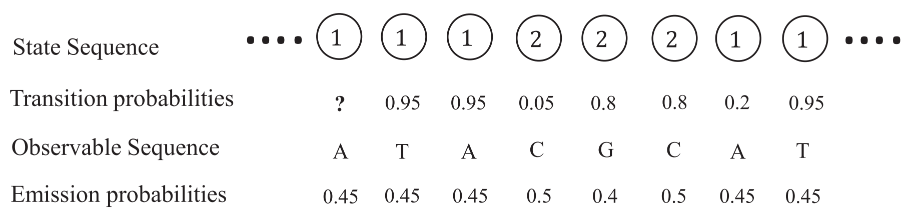

Fig. 5 Scheme for HMM emission probabilities and states.

profiling to discriminate between different protein families and predict a new protein's family or subfamily and in the problem of gene finding in DNA.

Statistical Inference in Hidden Markov Models

Once the architecture of an HMM has been decided, to analyze and describe data in almost all applications of HMMs, three distinct questions must to be solved:

1. What is the probability of an observed sequence according to a given HMM?
2. How to find the optimal state sequence that the HMM would use to generate the observed sequence?
3. How to find the structure and parameters of the HMM that best accounts for the data?

In general, in order to use the HMMs in application contexts, for example, in computational biology, to solve these questions, we need to be able to:

1) *Evaluate* the likelihood of the model given the observations, i.e., to compute the probability of the observation sequence for a model.
2) *Decode* the most likely state sequence given the observations, i.e., to find the optimal corresponding state sequence given the observation sequence and the model.
3) *Learning or training* to estimate the model parameters (initial probabilities, transition probabilities, and emission probabilities), that best explains the observation sequences given the model structure, describing the relationships between variables.

Evaluation Problem

Given a sequence of observations and an HMM, to solve the first problem, we would like to be able to compute the probability (or likelihood), $P(O|\lambda)$, that the observed sequence is produced by the model. This problem could be viewed as one of evaluating how well a model predicts a given observation sequence and thus allow us to choose the most appropriate model from a set. For this purpose, the forward-backward algorithm and the result can be used (Rabiner and Juang, 1986a; Stratonovich, 1960a). We consider the probability of the observations O for a specific state sequence Q

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(o_t|q_t,\lambda) = b_{q_1}(o_1) \times b_{q_2}(o_2)\ldots.b_{q_T}(o_T)$$

and the probability of the state sequence

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots. a_{q_{T-1} q_T}$$

The join probability that O and Q occur simultaneously is simply the product of above two terms $P(O|Q, \lambda)$ $P(O|Q, \lambda)$. Then, we can obtain the probability of the observations given the model by summing this join probability over all possible state sequences

$$P(O|\lambda) = \sum_{Q} P(O|Q,\lambda)P(Q|\lambda) = \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \ldots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

To evaluate the last equation the forward-backward approach, that reduced complexity, is used. The key idea is to process a sequence, consider a forward or backward loop that processes it one element at a time. The sequence $X(1\ldots T)$ is broken into two parts, a "past" sequence $X(1\ldots t)$ and a "future" sequence $X(t+1\ldots T)$. In the Hidden Markov Model, each symbol emission and each state transition depend only on the current state; there is no memory of what happened before, no lingering effects of the past. This means that we can work on each half separately. Splitting the sequence into two parts, the inductive calculation on t can be used: if t advances from 1 towards T, it is called *forward* calculation, while if t is decremented down from T towards 1, it is called the *backward* calculation. In formal terms, the forward algorithm calculates the probability of being in a state i at time t and having emitted the output $o_1,\ldots,o_t$. These probabilities are named the *forward variables*. By calculating the sum over the last set of forward variables, one obtains the probability of the model having emitted the given sequence. Both the forward algorithm and the backward algorithm involve three steps: initialization, recursion (or induction), and termination. We define the forward probability variable α as the probability of the partial observation sequence $o_1,\ldots,\ o_t$ and state s_i at time t:

$$\alpha_t(i) = P(o_1 o_2 \ldots. o_t, q_t = s_i|\lambda)$$

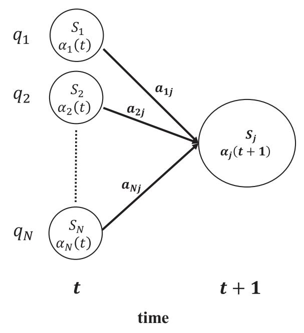

Fig. 6 The recursion step of the forward algorithm.

The forward algorithm is as follows:

1. *Initialization*
   Calculate the forward probability at the first position $\alpha_1(i) = \pi_i b_i(o_1),\ 1 \le i \le N$
2. *Recursion*
   Compute the forward probability

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}),\ 1 \le t \le T-1,\ 1 \le j \le N$$

   The recursion step is the key to the forward algorithm (see Fig. 6). For each state s_i, $\alpha_j(t)$ stores the probability of arriving in that state, having observed the observation sequence up until time t.
3. *Termination*
   When $i = N$, the forward recursion stops. Then, the probability of the whole sequence of observations can be found by summing the forward probabilities over all the states at the final variable

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

This approach reduces the complexity of calculations involved to $N^2 T$ rather than $2TN^T$. Similarly, backwards algorithm can be defined. It is the exact reverse of the forwards algorithm, with the backwards variable

$$\beta_t(i) = P(o_{t+1} o_{t+2} \ldots . o_T, q_t = s_i | \lambda)$$

as the probability of the partial observation sequence from $t+1$ to T , starting in state s_i.

Decoding Problem

The aim of decoding is to find the optimal state sequence associated with a given observation sequence. There are several possible ways to solve this problem. One possible optimality solution is proposed by the *Viterbi algorithm* (Viterbi, 1967a). The Viterbi algorithm uses a dynamic programming approach to find the most likely sequence of states Q given an observed sequence O and model λ. It works similarly to the forward algorithm. The goal is to get the most likely path through the HMM for a given observation; then, only the most likely transition from a previous state to the current one is important. Therefore, the transition probabilities are maximized at each step, instead of summed. To implement this solution, we define the variable

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2 \ldots, q_t = s_i, o_1, o_2 \ldots . o_t | \lambda)$$

It is the probability of the most probable state path for the partial observation sequence. Corresponding to each node, two storage variables are used to cache the probability of the most likely path for the partial sequence of observations and the state at the previous variable to lead this path, denoted $\delta_t(i)$ and $\psi_t(i)$, respectively. The Viterbi algorithm consists of four steps: initialization, recursion, termination, and backtracking. It is described as follows:

1. *Initialization*
   Calculate $\delta_1(i) = \pi_i b_i(o_1),\ 1 \le i \le N$ and set $\psi_1(i) = 0$
2. *Recursion*
   Calculate

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(o_t),\ 2 \le t \le T,\ 1 \le j \le N$$

and record the state

$$\psi_t(j) = \arg\max_{1 \le i \le N} \left[\delta_{t-1}(i)a_{ij}\right],\ 2 \le t \le T,\ 1 \le j \le N$$

3. *Termination*

   The recursion ends when $i=N$. The probability of the most likely path is found by

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

   The state of this path at variable N is found by

$$q^* = \arg\max_{1 \le i \le N} [\delta_T(i)]$$

4. *Backtracing*

   The last observation frame at time T is needed in order to decode the global optimal state path from T back to the first time index. The state of the optimal path at variable i is found by

$$q_t{}^* = \psi_{t+1}(q_{t+1}{}^*),\ t = T-1,\ T-2, \ldots, 1$$

   The backtracking allows the best state sequence to be found from the back pointers stored in the recursion step.

Learning Problem

How can we adjust the HMM parameters in a way that a given set of observations, called *training set*, is represented by the model in the best way for our purposes? Training involves adjusting the transition and output probabilities until the model sufficiently fits the process. These adjustments are performed using techniques to optimize $P(O|\lambda)$, the probability of observed sequence $o_1,\ldots, o_t$, given model λ over a set of training sequences. How do we learn the parameters of an HMM from observations? Depending on the application, the "quantity" that should be optimized during the learning process differs. There isn't known way to analytically solve this problem. Therefore iterative procedures or gradient techniques for optimization can be used. Here we will only present the iterative procedure. Therefore, we can choose an HMM such that $P(O|\lambda)$ is locally maximized. In literature, we can find several optimization criteria for learning. We present an iterative procedure, the Baum-Welch or Expectation Maximization (EM) method (Rabiner and Juang, 1986a; Baldi and Brunak, 1998a), which adapts the transition and output parameters by continually estimating these parameters until $P(O|\lambda)$ has been locally maximized. The EM algorithm is an iterative method to find the maximum likelihood estimate (MLE). It will let us train both the transition probabilities A and the emission probabilities B of the HMM. It works by computing an initial estimate for the probabilities, then using those estimates to compute a better estimate, and so on, iteratively improving the probabilities that it learns. This iterative algorithm alternates between performing an expectation step (E-step) and a maximization step (m-step). In an E-step, the expected number of times each transition is computed (the expected log-likelihood) and emission is used for the training set. In an M-step, the transition and emission parameters, that maximize the expected log-likelihood in the E-step are updated, using re-estimation formulas. The EM algorithm includes three steps of initialization, a series of iterations, and termination. In order to describe how to re-estimate HMM parameters, we first define the probability $\xi_t(i, j)$ of being in state S_i at time t and in state S_j at time $t+1$, as following:

$$\xi_t(i,j) = P\left(q_t = S_i, q_{t+1} = S_j | O, \lambda\right)$$

1. ***Iteration***

   Each cycle of EM iteration involves two steps, an E-step followed by an M-step, alternately optimizing the log-likelihood with respect to the posterior probabilities and parameters, respectively. In E-step we calculate the posterior probability of latent data using the current estimate for the parameters (π, α, β) of the model λ at time t. To perform the E-step, the expected log-likelihood function is maximized under the current estimated parameters and is computed, by using the estimated posterior probabilities of hidden data. In M-step: the new parameters that maximize the expected log-likelihood found in the E-step are estimated. From the definitions of forward and backward variables, we can calculate $\xi_t(i, j)$, which represents the posterior state probabilities of a variable and of the state combinations of two adjacent variables. We can write

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}$$

   where the numerator term is $P(q_t=S_i,\ q_{t+1}=S_j|O,\ \lambda)$ and $P(O|\lambda) = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)$ is the proper nor malization factor for $\xi_t(i,j)$. We need to introduce another auxiliary variable, $\delta_t(i)=P(q_t=s_i,\ O|\lambda)$. It is the probability of being in the state S_i at time i, given the observation sequence and the model. In forward and backward variables this can be expressed by

$$\delta_t(i) = \frac{\alpha_t(i)\beta_{t+1}(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}$$

Then, we can see that the relationship between $\delta_t(i)$ and $\xi_t(i, j)$ is given by

$$\delta_t(i) = \sum\nolimits_{j=1}^{N} \xi_t(i,j), \; 1 \le i \le N, \; 1 \le t \le T$$

If we sum over $\delta_t(i)$ over the time index t, we obtain the expected number of transitions from state S_i, while, summing $\xi_t(i, j)$ over the time, we obtain the expected number of transitions from state S_i to state S_j. Then, we can use again the Baum-Welch method to re-estimate the HMM parameters as follows

$$\overline{\pi}_i = \textit{expected frequency in state } S_i \textit{ at time } t = 1$$

$$\overline{a}_{ij} = \frac{\textit{expected number of transitions from state } S_i \textit{ to state } S_j}{\textit{expected number of transitions from state } S_i}$$

$$\overline{b}_i(k) = \frac{\textit{expected number of times in state } j \textit{ and observing symbol } v_k}{\textit{expected number of times in state } j}$$

2. ***Termination***

If we apply this procedure iteratively, (Baum and Sell, 1968a; Baum *et al.*, 1970a) using $\overline{\lambda}(\overline{\pi}, \overline{\alpha}, \overline{\beta})$ in place of λ, and we repeat the calculation of the parameters, we can improve that the probability of O being observed from the model until convergence is reached. The final result of this re-estimation procedure is called maximum likelihood estimate of the HMM. The re-estimation parameters can be derived by maximizing the auxiliary function

$$Q(\lambda, \overline{\lambda}) = \sum_{Q} P(Q|O, \lambda) \log\left[P(O|Q, \overline{\lambda})\right]$$

over λ. The maximization of $Q(\lambda, \overline{\lambda})$ leads to increased likelihood

$$\max_{\overline{\lambda}} Q(\lambda, \overline{\lambda}) \Rightarrow P(O|\overline{\lambda}) \ge P(O|\lambda)$$

Case Study

We consider an application of HMMs to the finding problem of the CpG islands in a DNA sequence (Ron, Lecture Notes). For this purpose, we define CpG islands as regions of DNA with a high frequency of CpG sites, where a cytosine nucleotide is followed by a guanine nucleotide in the linear sequence of bases along its $5' \rightarrow 3'$ direction. The CG pair of nucleotides is the most infrequent dinucleotide in many genomes. This is because cytosines C in CpG dinucleotides are vulnerable to a process, called methylation, that can change with a high chance in mutating to a T. The methylation process is suppressed in areas around genes and hence these areas contain a relatively high concentration of the CpG dinucleotide. Such regions are called CpG islands, whose length varies from few hundreds to few thousands bases, with a GC content of greater than 50% and a ratio of observed-to-expected CpG number above 60%. The presence of a CpG island is often associated with the start of the gene (promoter regions) in the most mammalian genome and thus the presence of a CpG island is an important signal for gene finding. Then, we consider an HMM for detecting CpG islands in a DNA sequence. In this case, the model contains eight states corresponding to the four symbols of the alphabet $\sum = \{A, C, G, T\}$, where the states and emitted symbols are shown in Fig. 7 and we consider two possible conditions: a DNA sequence is a CpG island (labeled $+$) or a DNA sequence is non-CpG island (labeled -). Considering a DNA sequence $X = (x_1, \ldots, x_L)$, the question is to decide whether X is a CpG island.

Results

In the case study we ask if given a short sequence, is it from a CpG island or not. To answer this question, we can estimate the transition probabilities from statistical data about CpG islands and non-CpG islands and then we can build two Markov chains, one for each. Then given a sequence, we compute the probability p of obtaining the sequence in the CpG island Markov chain, and the probability q of obtaining the sequence in the non-CpG island Markov chain. The odds ratio or log-odds ratio of these two

State:	A^+	C^+	G^+	T^+	A^-	C^-	G^-	T^-
Emitted Symbol:	A	C	G	T	A	C	G	T

Fig. 7 Hidden Markov Model for identification of CpG islands and non-CpG islands.

Table 3 Transition probability matrix for CpG islands (+) and non-CpG islands (−)

$$p^{+} = \begin{bmatrix} & A & C & G & T \\ A & 0.18 & 0.27 & 0.43 & 0.12 \\ C & 0.17 & 0.37 & 0.27 & 0.19 \\ G & 0.16 & 0.34 & 0.37 & 0.13 \\ T & 0.08 & 0.36 & 0.38 & 0.18 \end{bmatrix}$$

$$p^{-} = \begin{bmatrix} & A & C & G & T \\ A & 0.30 & 0.20 & 0.29 & 0.21 \\ C & 0.32 & 0.30 & 0.08 & 0.30 \\ G & 0.25 & 0.25 & 0.29 & 0.21 \\ T & 0.18 & 0.24 & 0.29 & 0.29 \end{bmatrix}$$

Table 4 Transition probability matrix in the CpG islands for HMM

$a_{\pi_i,\pi_{i+1}}$	A^+	C^+	G^+	T^+	A^-	C^-	G^-	T^-
A^+	$0.18p$	$0.27p$	$0.43p$	$0.12p$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$
C^+	$0.17p$	$0.37p$	$0.27p$	$0.19p$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$
G^+	$0.16p$	$0.34p$	$0.37p$	$0.13p$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$
T^+	$0.08p$	$0.36p$	$0.38p$	$0.18p$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$
A^-	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$0.30q$	$0.20q$	$0.29q$	$0.21q$
C^-	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$0.32q$	$0.30q$	$0.08q$	$0.30q$
G^-	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$0.25q$	$0.25q$	$0.29q$	$0.21q$
T^-	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$0.18q$	$0.24q$	$0.29q$	$0.29q$

probabilities can be used to determine whether the sequence is coming from a CpG island or not. Then, for the CpG island Markov chain, we estimate a_{ij}^{+} as follows

$$a_{ij}^{+} = \frac{c_{ij}^{+}}{\sum_k c_{ik}^{+}}$$

where c_{ij}^{+} is the number of times nucleotide j follows nucleotide i in the sequences labeled $+$. For the non-CpG island Markov chain, a_{ij}^{-} is estimated in a similar way. Now given a sequence X, we can compute $p(x)$ for each Markov chain; denoted these by $p(x|+)$ and $p(x|-)$, we can use the log-odds ratio $\log\frac{p(x|+)}{p(x|-)}$ to determine if X is coming from a CpG island or not; in fact, if $\log\frac{p(x|+)}{p(x|-)} > 0$, the sequence X is coming from a CpG island. Assuming that the transitions from the start state and to the end state are the same in both cases, the log-odds ratio can be expressed as following:

$$\log\frac{p(x|+)}{p(x|-)} = \log\frac{\prod_{i=0}^{n} a_{x_i x_{i+1}}^{+}}{\prod_{i=0}^{n} a_{x_i x_{i+1}}^{-}} = \sum_{i=1}^{n-1} \log\frac{a_{x_i x_{i+1}}^{+}}{a_{x_i x_{i+1}}^{-}}$$

We consider, as example, the short sequence CGCG; the transition probabilities for each of the two chains shown in Table 3. From the Table 3, we note that for the '+' Markov chain (CpG islands), the transition probabilities to C and G are higher. The log-odds ratio for this sequence is $\log\frac{0.27}{0.08} + \log\frac{0.34}{0.25} + \log\frac{0.27}{0.08} > 0$. Therefore, in this case, CGCG is coming from a CpG island. Instead, to verify if given a long sequence, does it contain a CpG island or not, we can incorporate both models (CpG islands and non-CpG islands) into one model. Then, we build a single Markov model consisting of both chains (+) and (−) described above as sub-chains, and with small transition probabilities between the two sub-chains. As before, we can estimate the transition probabilities between the two sub-chains by relying on known annotated sequences with all their transitions between CpG and non-CpG islands. is that there it not a one-to-one correspondence between the states and the symbols of the sequence. For instance, the symbol C can be generated by both states C^{+} and C^{-}. For our purpose, we use an HMM (see Fig. 7): a sequence $X=(x_1,\ldots, x_L)$ does not uniquely determine the path in the model and the states are hidden in the sense that the sequence itself does not reveal how it is generated. In this model, we define the probability for staying in a CpG island as p and the probability of staying outside as q, then the transition probabilities can be described in Table 4, derived from the transition probabilities given in Table 3 under the assumption that we lose memory when moving from/into a CpG island, and that we ignore background probabilities. In this particular case, the emission probability of each state X^{+} or X^{-} is exactly 1 for the symbol X and *0* for any other symbol.

Conclusions

HMM methods are used to solve a variety of biological problems, such as, for example, gene prediction, protein secondary structure prediction; in the last years, HMM-based profiles was applied to the protein-structure prediction and large-scale genome sequence analysis. In this paper we have presented an introduction to statistical approach of an HMM to show as these methods provide a conceptual framework for building complex models, just by drawing an intuitive picture. In particular, we have described the three problems of this theory with applications to the problem of finding specific patterns in biological sequences.

See also: Deep Learning. Introduction to Biostatistics. Natural Language Processing Approaches in Bioinformatics. Nonlinear Regression Models. Stochastic Processes

References

Baldi, P., Brunak, S., 1998a. Bioinformatics – The Machine Learning Approach. Massachusetts Institute of Technology.
Baum, L.E., Petrie, T., Soules, G., Weiss, N., 1970a. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann. Math. Statist. 41 (1), 164–171. doi:10.1214/aoms/1177697196.
Baum, L.E., Sell, G.R., 1968a. Growth transformations for functions on manifolds. Pacific J. Math. 27 (2), 211–227.
Oliver, K., 2009. Probability Theory and Stochastic Processes With Applications Paperback. Overseas Press India Private Limited: New Delhi
Rabiner, L., Juang, B., 1986a. An introduction to hidden Markov models. IEEE ASSP Magazine 3 (1), 4–16.
Shamir, R. Lecture notes: Algorithms in molecular biology – Hidden Markov models. Tel Aviv University – Blavatnik School of Computer Science. Available at: http://www.cs.tau.ac.il/∼rshamir/.
Stratonovich, R., 1960a. Conditional Markov processes. Theory of Probability & Its Applications 5 (2), 156–178.
Viterbi, A.J., 1967. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. In: IEEE Transactions on Information Theory, vol. 13, pp. 260–269.

Further Reading

Chung, K.L., 1998a. Markov Chains With Stationary Transition Probabilities, second ed. Berlin: Springer-Verlag.
Doob, J.L., 1953. Stochastic Processes: Stochastic Modelling for Systems Biology. New York, NY: John Wiley& Sons.
Howard, R.A., 1971a. Dynamic Probabilistic Systems, vol. 1. New York, NY: John Wiley and Sons.
Taylor, H.M., Samuel, K., 1998. An Introduction to Stochastic Modeling, third ed. San Diego, CA: Academic Press.
Revuz, D., 1984a. Markov Chains, second ed. Amsterdam: North-Holland.