An Introduction to Hidden Markov Models

Markov and hidden Markov models have many applications in Bioinformatics. A quick search for "hidden Markov model" in Pubmed yields around 500 results from various fields such as gene prediction, sequence comparison, structure prediction, and more specialized tasks such as detection of genomic recombination, determining expression from genomic tiling microarrays, and many others. One might in fact rather ask "What are HMMs not good for?" (this will be discussed later in the unit). First, though, the unit will very briefly introduce the history of Markov and hidden Markov models and outline some common fields of their application.

HISTORY

The name *Markov* model refers to the Russian mathematician Andrei Markov (1856-1922) who studied sequences of mutually dependent random variables. For many years, his findings could not be used practically due to the sheer complexity of the calculations. With the advent of modern computers, however, this limitation diminished, and hidden Markov models (HMMs) quickly became popular as a tool for *supervised machine learning*. One of the first applications of HMMs was in the field of *speech recognition*. According to the classic tutorial on HMMs by Rabiner (1989), the basic theory of HMMs was outlined by Baum and colleagues (Baum and Petrie, 1966) in the late 1960s and further developed by different groups during the 1970s. In one of his early reviews on the subject, Eddy (1996) describes how HMMs were adopted in biology. According to him, Haussler, Krogh, and colleagues first realized that a range of profile methods already in use to describe protein families could be expressed in terms of HMMs. Baldi et al. also came up with a very similar concept for profile HMMs at the same time (Baldi et al., 1994). Soon after, several groups began to explore the potential of HMMs for sequence analysis, and later two of the most popular packages, SAM and HMMER, were released by groups in Santa Cruz and Cambridge, respectively.

COMMON APPLICATIONS

Finding signals in noisy data is the problem that Bioinformatics is concerned with most of the time. The signal can be a gene in a genomic sequence, a transcription factor binding site, a member of a protein family from a set of unknown proteins and so forth. For all of these problems, programs were written that used existing knowledge to filter the noise. For example, when it was known that there are certain transcription initiation motifs and stop codons, one could search for open reading frames in genomes. Most of the early programs were very simple in their design, they just looked for exact occurrences of small strings or used ad-hoc scoring schemes. They lacked a conceptional probabilistic framework to bind together different pieces of evidence and give meaning to the returned scores. It was often difficult to extend these programs or to calculate some kind of confidence value to test the quality of the output.

Markov models and HMMs offered a clean and well-established theory that applied to a wide range of existing problems. Many programs actually already implicitly used HMM-like structures, but employing arbitrary scores rather than probabilities. HMM theory sheds a new light on those programs, showing that seemingly different approaches were strongly related to each other from the underlying principles. HMMs also made it possible to interpret scores as probabilities. This greatly simplified the creation, extension, and evaluation of programs. As will be seen later, by just changing the model layout, a wide range of questions can be addressed, while the algorithms for parameter estimation and decoding remain the same.

Another great advantage of HMMs is that machine learning techniques can be used to characterize the statistical properties of a signal. Using a *data-driven* learning approach, a program can be allowed to induce a set of rules from a collection of manually classified cases. Those rules are then converted into an HMM and applied by another program to classify unknown cases.

Contributed by Benjamin Schuster-Böckler and Alex Bateman
Current Protocols in Bioinformatics (2007) A.3A.1-A.3A.9

Sequence Comparison

Detecting homologous sequences was one of the first applications in Bioinformatics. Soon after the first protein sequences were collected, algorithms were developed to compare them and evaluate their similarity. Tools like NEEDLE or BLAST compare pairs of sequences and predict whether the sequences are homologous.

In a pairwise alignment, all residue positions are treated the same and only judged in terms of the overall amino acid substitution rate. What a pairwise sequence alignment can not provide is a characterization of conserved residues within a *family* of related sequences. For example, an important active site residue is scored just the same as a rapidly mutating surface residue. *Multiple sequence alignments (MSA)* offer a handy visualization of a set of related sequences. From a multiple alignment, it can be easily seen which positions are more conserved than others. To test whether a new sequence belongs to a family, and how good the fit is, one needed to recalculate the whole alignment, which is relatively slow. Profile methods for homology detection were invented to allow searching for family members in sets of sequences. The position-specific scores allowed important residues to be given higher weight. But due to the ad-hoc nature of the scoring schemes, it needed an expert to assess which matches were significant. Furthermore, only experts with sufficient knowledge of the particular program and the biology of the sequences of interest were able to use the advantage of these programs.

Profile HMMs provided a probabilistic framework to describe a set of sequences by their observed properties. Many existing algorithms could be reinterpreted with respect to HMM theory. Within this framework it was now possible to compare and evaluate the scores of different programs. HMM packages like HMMER or SAM now allow non-specialists to use existing multiple sequence alignments to derive a profile HMM, without having to tweak any part of the process manually.

Once an HMM representing a set of homologous sequences has been created, it can be used to search for other potential members of this family in a fast and reliable way. The Pfam database provides a hand curated collection of sequence families represented as HMMs. More on Pfam can be found in *UNIT 2.5* (Identifying Protein Domains with the Pfam Database).

It should be mentioned here that HMMs have not replaced other methods. Many programs still exist and new ones are being developed which use other ways of representing similarity data. HMMs are not even necessarily the most sensitive method. What makes HMMs so popular is that they offer a very general, well-established framework that is maintainable and extendable.

Gene Finding

Several gene finding programs use HMMs to distinguish intergenic sequence, exons, introns, and other elements on raw genomic sequence (Kulp et al., 1996; Burge and Karlin, 1997; Lukashin and Borodovsky, 1998). Unlike in homology detection, these HMMs do not necessarily have to be trained on existing data but can be set up with general values, e.g., about the nucleotide frequencies in different elements. However, training the models on each species gives large gains in accuracy. A simple example of this will be seen below. One program, GeneMark.hmm, is further described in *UNITS 4.5 & 4.6* (Prokaryotic/Eukaryotic Gene Prediction Using GeneMark.hmm).

Birney et al. (2004) describe an algorithm that combines gene prediction and homology searching using a strictly probabilistic approach. Their program, called GeneWise, is able to identify regions in raw DNA sequence which, once translated and spliced, would be similar to a known protein family. This approach basically combines a simplified, GenScan-like gene-finder with HMMER-style profile HMMs. The strictly HMM based approach greatly facilitated this combination. Some more information is provided in *UNIT 2.5* (Identifying Protein Domains with the Pfam Database).

Fold Assignment and Structure Prediction

HMMs are routinely used in homology modeling to identify known folds in a target sequence. HMMs are popular in this area for their high sensitivity. After successfully identifying template structures, it is often necessary to improve the accuracy of the alignment. It can be advantageous to compare a profile of the target family to the profile of the fold, rather than just aligning the individual target sequence to the fold profile. Tools for profile-profile comparison and visualization have been developed and shown to be useful (Madera, 2005; Schuster-Böckler and Bateman, 2005; Söding, 2005).

Phylogeny and Function Prediction

An efficient way to infer the properties of an unknown protein is usually to search for evolutionarily related sequences which have some annotations. It has already been mentioned that HMMs are widely used for homology detection. *UNIT 6.9* describes the FlowerPower algorithm which uses an iterative clustering algorithm involving repeated HMM searches to cluster related sequences into subfamilies with distinct features. These subfamilies can then be used for functional inference.

MARKOV MODELS

One type of Markov chain that almost every biologist has come across at some point, probably without realizing it, are amino acid or nucleotide substitution matrices like PAM and BLOSUM. Quite generally, a Markov chain is a type of stochastic (random) process. It consists of several states (x) from a system (X), which represent observations at a specific point in time (t), and a set of *transition* probabilities (Pr). What distinguishes Markov chains from other stochastic processes is the *Markov property*, which states that Pr depends only on the current state of the system. In mathematical terms, this is usually written as shown in equation below.

$$Pr(X_{t+1}=x_j|X_1=x_1,X_2=x_2,...,X_t=x_i) \\ =Pr(X_{t+1}=x_j|X_t=x_i)$$

for all $t \geq 1$ and $x_1,...,x_n$ element of S, where S is the set of all possible states in the system. “|” in this context means “depending on”. The equation denotes that the transition probability of going from x_i to x_j at timestep $t+1$ does not depend on the states visited in the past.

Markov models are often represented by a graph in which states are shown as nodes and the transitions are edges. A graph showing the transition probabilities as defined by a nucleotide substitution matrix is shown in Figure A.3A.1B. Each nucleotide is a possible observation, and there are probabilities of substituting one for another. Imagine looking at one residue in a sequence, and checking it every 1 million years. Within each time-step, the residue can mutate to a different residue or remain the same. The model in Figure A.3A.1B shows exactly that. At every time step, one can transit from one state to another (including, in this case, going to the same state again), with a probability p_{ij} depending on the state one is currently in. The residue change from T to C in the alignment shown in Figure A.3A.1A then corresponds to a transition from state T to state C, which has a probability of p_{TC}. While it is theoretically possible to have different transition probabilities for a $T \rightarrow C$ and a $C \rightarrow T$ transition, substitution matrices like PAM or BLOSUM are always symmetrical.

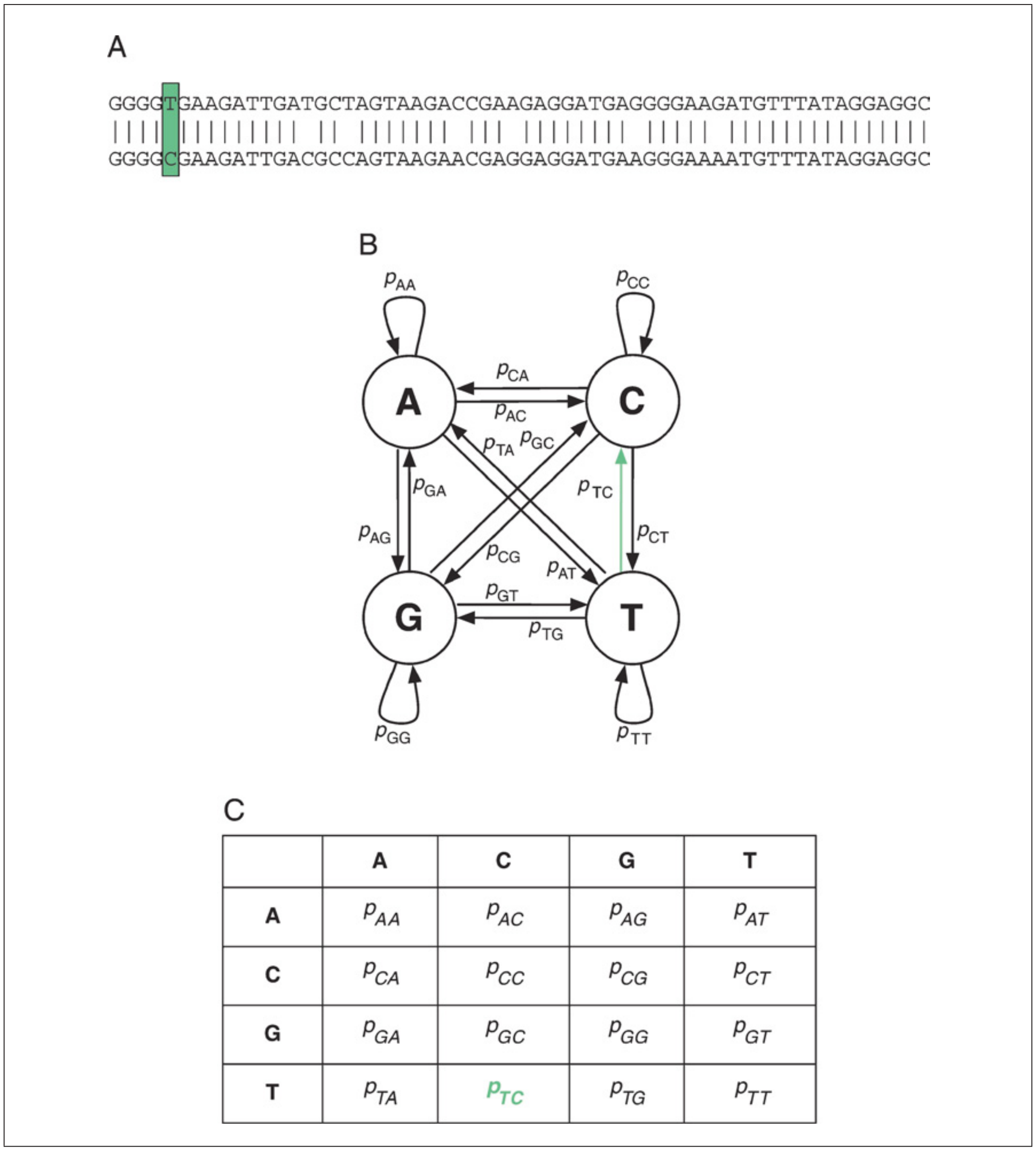
In this example, the Markov property demands that what a residue will mutate to in the future does not depend on what residues were observed in the past. This means that if it is known that another homologous sequence exists with a C residue observed at the specified position, that does not change the probability of a $T \rightarrow C$ transition. It should be mentioned here that relationships between neighboring residues are also not accounted for in any way. Certainly, these are major simplifications, but the benefits usually outweigh the drawbacks.

Markov chains can usually be expressed as a matrix, where each row and column corresponds to a state and the value in the cell is the transition probability between the states (see Fig. A.3A.1C). Representing a Markov chain as a Matrix allows a number of useful operations. In the case of substitution matrices, for example, one can use matrix multiplication to extrapolate substitution matrices for any arbitrary time-distance from the one matrix that was measured. Also, because the matrix consists purely of probabilities, one can calculate confidence measures analytically.

HIDDEN MARKOV MODELS

So, what is hidden in a hidden Markov Model? In our earlier example, a state represents exactly one residue, and this residue will be observed every time the state is passed. But there are cases where the actual state the process is in at a given time is not known. In the introduction, it was mentioned that HMMs can be used for gene prediction. Following the example from Eddy (2004), imagine one is interested in identifying splice sites in a genomic sequence. Simply put, it is stated that every residue in a genome is either in an exon (E), intron/intergenic (I), or a 5′ splice site (5). A Markov chain describing this model is shown in Figure A.3A.2A. The difference to our earlier example is that the sequence of visited states can not be observed directly. In fact, the most likely sequence of hidden states is very often what one is interested in.

A hidden Markov model consists of a set of states just like a simple Markov model. The difference is that every hidden state *emits* a nucleotide when it is visited. Therefore, every state “contains” a probability distribution which defines the emission probability. Just like the transition probabilities, the emission probabilities are again independent of the past.



```
GGGGTGAAGATTGATGCTAGTAAGACCGAAGAGGATGAGGGGAAGATGTTTATAGGAGGC
|||| ||||||||| || ||||||| ||| |||||||| ||||| |||||||||||||||
GGGGCGAAGATTGACGCCAGTAAGAACGAGGAGGATGAAGGGAAAATGTTTATAGGAGGC
```

	A	C	G	T
A	p_{AA}	p_{AC}	p_{AG}	p_{AT}
C	p_{CA}	p_{CC}	p_{CG}	p_{CT}
G	p_{GA}	p_{GC}	p_{GG}	p_{GT}
T	p_{TA}	p_{TC}	p_{TG}	p_{TT}

Figure A.3A.1 A simple Markov model for the transition probabilities as defined by a nucleotide substitution matrix. **(A)** Example of a short DNA sequence alignment with the transition depicted in green.**(B)** A simple Markov model for DNA residue substitutions. Every circle represents a state, and arrows denote probabilities to make a transition into the state at the end of the arrow in the next time step. A stochastic process like this is called a Markov chain if the transition probabilities *p* do not change depending on the states visited in the past. The transition observed in Panel (A) is highlighted in green. **(C)** A transition matrix for the model shown in (B). For color version of this figure see *http://www.currentprotocols.com*.

A model like the one in Figure A.3A.2A can be thought of as a generator or simulator of whatever it represents, in this case DNA containing splice sites. It can be used to create "random" strings of nucleotide letters. Our example starts at the starting state "S", then passes from one state to the other, choosing the next state according to the transition probabilities on the arrows. When a state is reached, a random letter according to the distribution defined for the state is emitted. This continues until the end state "N" is reached. An example of a state path and the emitted sequence is shown in Figure A.3A.2B. The recorded sequence of emitted residues from the path through the model should have the characteristics of a splice site as was defined, because all paths from start to end must pass through the "5" state.

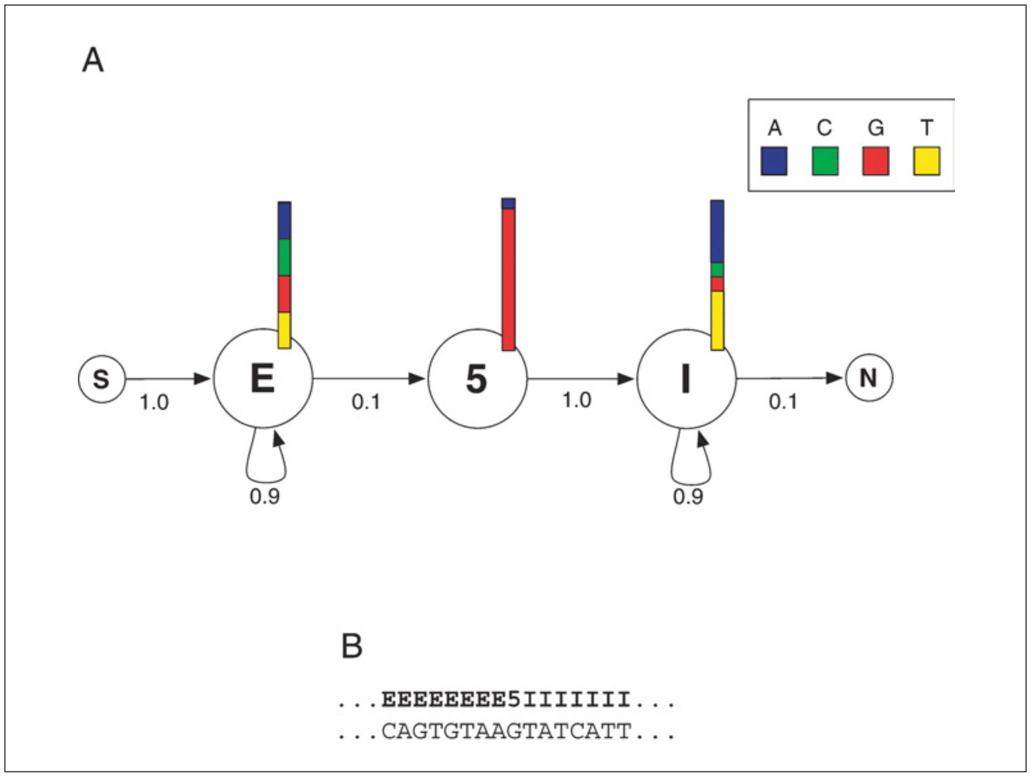


Figure A.3A.2 A simple hidden Markov model for splice site recognition (Eddy, 2004). **(A)** The circles again denote states, in this case exon (E), intron (I) or 5′ splice site (5). The edges represent transition probabilities to move from one state to another in each time step. "Time" in this context refers to the residue position we are in, not actual time. The sequence of visited states cannot be observed, rather a nucleotide distribution that is typical for one of the 3 states is seen, here shown as a bar shaded in 4 colors. The proportion of each color reflects the probability of observing this nucleotide when the state is visited. **(B)** An example of a sequence (bottom) and its hidden state path (top, bold font). Passing through the model and emitting a residue according to the given emission probabilities creates a sequence like the one shown. Vice-versa, one can calculate the probability that an observed sequence was created by the given state path. For color version of this figure see *http://www.currentprotocols.com*.

Usually one would not want to create a random sequence, unless it is necessary to know whether some sequence is a splice site. In other words, the question is: Given an observation, what is the most likely sequence of hidden states that would have created this observation? The so-called *Viterbi* algorithm finds the state-path with the highest probability given an observed sequence. For this purpose, it is useful to think of an HMM as a generating model: The Viterbi algorithm compares the probability of all paths through the model that could generate the observed sequence and chooses the most likely one.

Another question that is particularly important for profile HMMs representing a sequence family is: How likely is it that a given model generated an observed sequence? This kind of analysis is often called *posterior decoding*. One can find the probability of generating a given sequence from an HMM by summing over the probabilities of all paths that could create this sequence. Efficient algorithms for these and other problems are implemented in all major HMM packages. More on the details of these algorithms can be found in Durbin et al. (1998).

PROFILE METHODS FOR SEQUENCE ANALYSIS

Profile hidden Markov models are a way to describe families of related sequences. It was previously shown how a hidden Markov chain is made up of a series of states which emit a signal according to a state-dependent distribution. In our earlier example, there were 3 states (E, I, and 5). A schematic outline of a profile HMM (as implemented in the HMMER package) is shown in Figure A.3A.3. In profile HMMs, the number of states varies depending on the length of the sequences in the family. The layout is strictly repetitive: The three types of states in a profile HMM are *Match*, *Insert*,

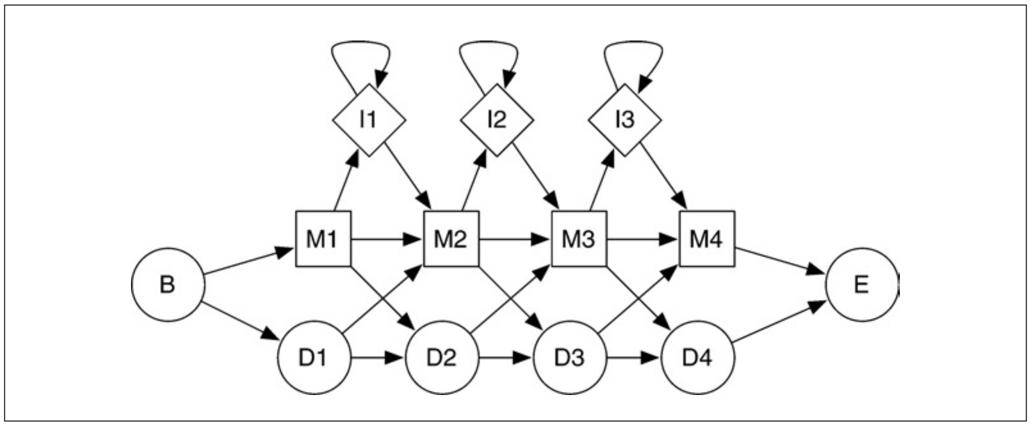


Figure A.3A.3 Schematic representation of a profile hidden Markov model of length 4. The states are begin (B), end (E), match (M), insert (I), and delete (D). Rectangular states (M and I) emit amino acids or nucleotides, round states are silent. A sequence family is represented by a characteristic distribution of amino-acid emission probabilities at every match or insert state. Delete states are equivalent to gaps.

and *Delete*. Match states represent informative positions in a family, while insert states allow insertions of small stretches of nonspecific sequence. Delete states correspond to gaps, where a match position is skipped in a member of the family. With every step, the process moves from left to right, to the next distinct residue position, until the end state has been reached.

The parameters of a profile HMM, its emission and transition probabilities, have to be "learned" before it can be applied. This is done by taking a multiple sequence alignment of a core set of family members and deriving the conservation of positions and residues from it, see Figure A.3A.4. The match states in the profile represent conserved positions in the multiple alignments. Positions in the multiple alignments which contain gaps for most member sequences transform to a high probability of passing through the corresponding delete state. Similarly, regions which are indistinguishable to the background amino acid distribution are modeled as insert states.

Deriving the parameters of the HMM from a sequence alignment necessarily means that a profile HMM can only be as good as this underlying alignment. If, for example, an alignment contains numerous orthologous sequences, the resulting HMM could be biased towards a particular subset of the family. The HMM might then not be able to find more distant family members with a distinctly different residue composition. While software packages like HMMER or SAM try to correct for bias by too closely related sequences in the alignment, great care should still be taken when compiling a training set for a protein family.

PAIR HMMS

A pair HMM is a process that maps one alphabet to another. The beauty of pair HMMs is that the concept can be applied to all sorts of alignment or mapping problems. Comparison of two profile HMMs can be expressed as a pair HMM (Madera, 2005). Birney et al. use pair HMMs in their GeneWise algorithm to link the output of a basic gene-prediction algorithm with a profile HMM search (Birney et al., 2004). Pair HMMs have been used successfully for gene prediction when two related sequences are known (Meyer and Durbin, 2004).

As an example, pairwise sequence alignment can be expressed in terms of pair HMMs, too. Figure A.3A.5 shows the model for an HMM that corresponds to a pairwise alignment. The core of the model consists of a

Figure A.3A.4 *(at right)* The Leucine Rich Repeat family as a multiple sequence alignment and as an HMM. The columns of the HMM and the positions in the profile HMM that correspond to them are shown by black bars. The bottom part shows an HMM logo. HMM logos resemble sequence logos (Schuster-Böckler et al., 2004). Each position in the HMM corresponds to a column in the logo. Match states are white, insert states are red. The width of the column reflects how likely it is to skip it by going through the delete state. The height of the letters shows their frequency relative to the overall information content of the state. It can be seen how the information in the HMM mirrors the composition of the sequence alignment. For color version of this figure see *http://www.currentprotocols.com*.

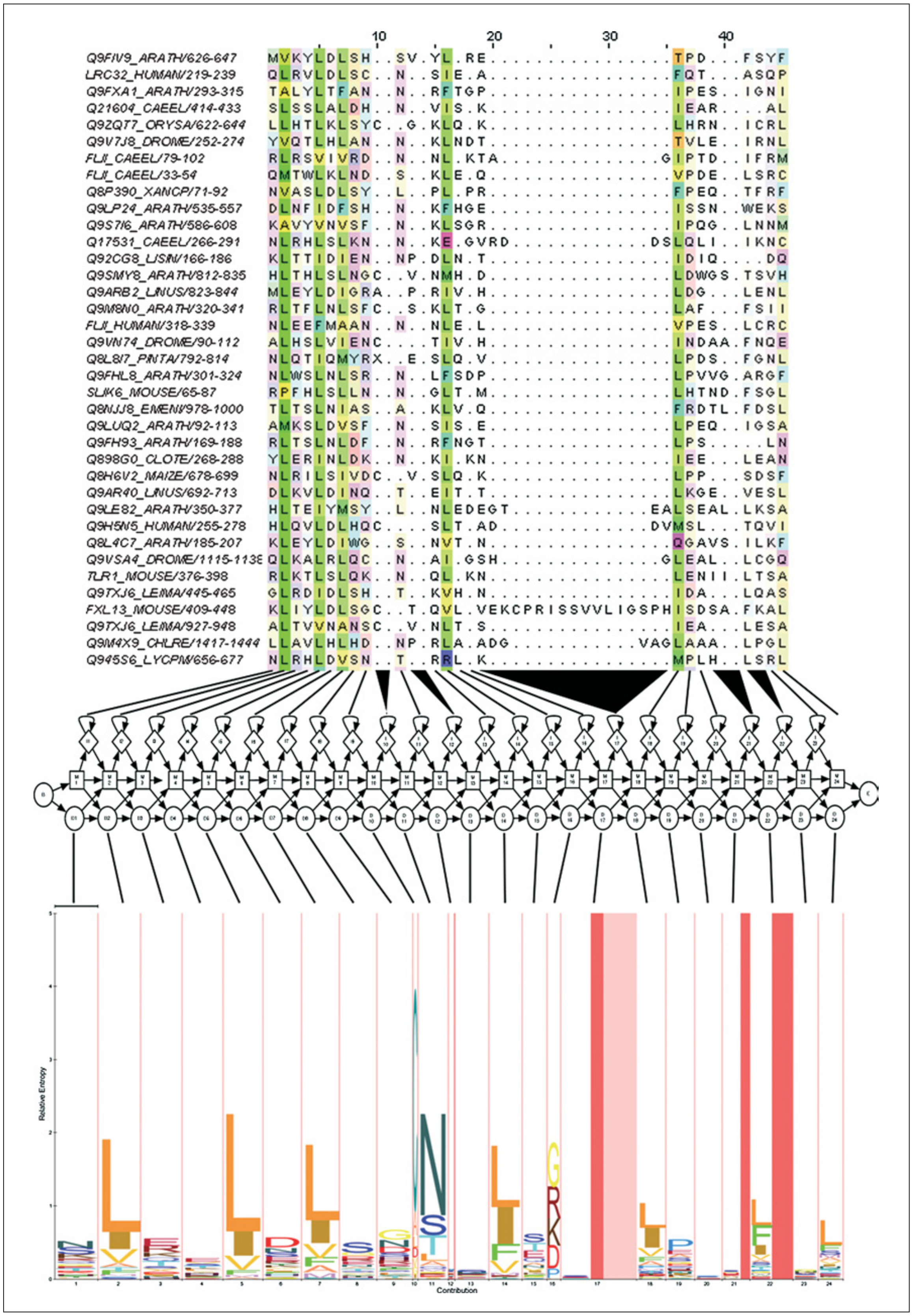


Figure A.3A.4 Legend at left.

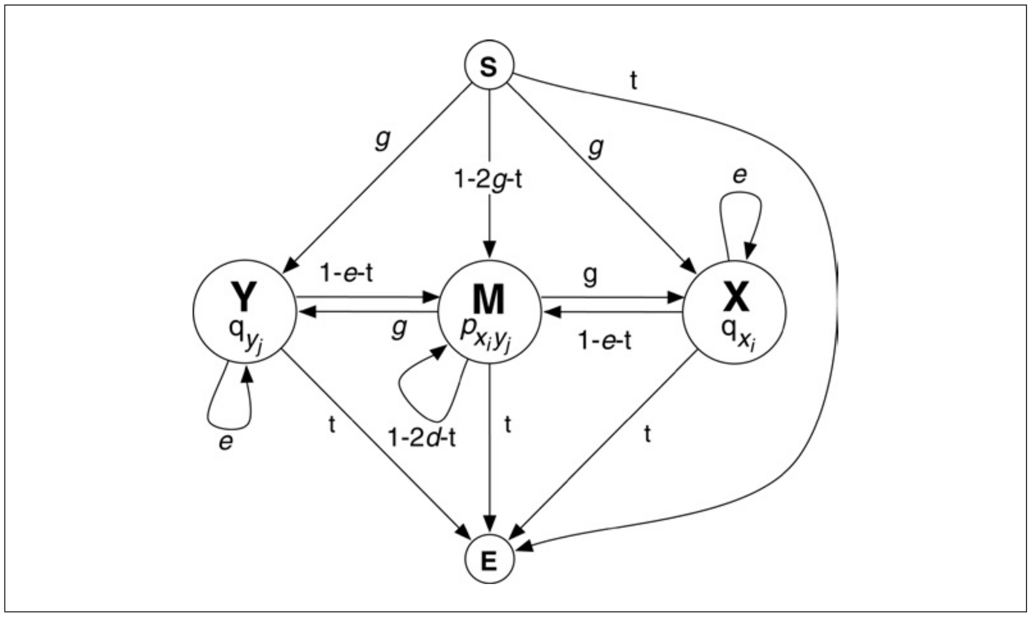


Figure A.3A.5 A pair HMM for global pairwise alignment with affine gap penalties as described by Durbin et al. (1998).

match state M that emits a *pair* of symbols. The X and Y states are gaps in either sequence. The start S and end E states complete the model. The model generates sequences of pairs of symbols, or gaps in one of the sequences. The probability to emit a certain pair of symbols $p_{x_i y_j}$ is identical to a substitution score. g in the model is the gap opening probability, e is the gap extension probability.

DRAWBACKS

What are hidden Markov models *not* good for, then? As mentioned before, care has to be applied when using machine learning to estimate parameters from existing data. A prediction can only be as good as the data used for training. When there are very few training data available, other methods can sometimes perform better, especially if they make use of expert knowledge that was built into the algorithm directly.

The most important limitation of HMMs results from the Markov property itself. Because HMMs are *memoryless*, it is not possible to model dependencies between distant events. In practice this means that one cannot, for example, model the dependency between two nucleotides in an RNA sequence that form a base pair in the secondary structure, but are separated by an unknown number of residues in the primary structure. There are other methods that can deal with such situations, see *UNITS 12.1 & 12.5*, but they increase the computational complexity significantly.

CONCLUSIONS

The number of applications of HMMs in the field of Bioinformatics is large and still growing. Sophisticated algorithms, complex models, and a specialized terminology sometimes gives people the impression that HMMs are an overly complicated piece of machinery. It is the hope of the authors to have opened the "black box" of HMMs a little, shedding some light on what they do and how they achieve it. After all, HMMs can make life easier for bioinformaticians and the users of the programs they develop.

LITERATURE CITED

Baldi, P., Chauvin, Y., Hunkapiller, T., and McClure, M.A. 1994. Hidden Markov models of biological primary sequence information. *PNAS* 91:1059-1063.

Baum, L.E. and Petrie, T. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Stat.* 37:1554-1563.

Birney, E., Clamp, M., and Durbin, R. 2004. GeneWise and Genomewise. *Genome Res.* 14:988-995.

Burge, C. and Karlin, S. 1997. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* 268:78-94.

Durbin, R., Eddy, S.R., Krogh, A., and Mitchison, G. 1998. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge, U.K.

Eddy, S.R. 1996. Hidden Markov models. *Curr. Opin. Struct. Biol.* 6:361-365.

Eddy, S.R. 2004. What is a hidden Markov model? *Nat. Biotechnol.* 22:1315-1316.

Kulp, D., Haussler, D., Reese, M.G., and Eeckman, F.H. 1996. A generalized hidden Markov model for the recognition of human genes in DNA. *Proc. Int. Conf. Intell. Syst. Mol. Biol.* 4:134-142.

Lukashin, A.V. and Borodovsky, M. 1998. GeneMark.hmm: New solutions for gene finding. *Nucl. Acids Res.* 26:1107-1115.

Madera, M. 2005. Hidden Markov models for detection of remote homology. PhD thesis, University of Cambridge, MRC Laboratory of Molecular Biology, May 2005.

Meyer, I.M. and Durbin, R. 2004. Gene structure conservation aids similarity based gene prediction. *Nucl. Acids Res.* 32:776-783.

Rabiner, L.R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77:257-286.

Schuster-Böckler, B. and Bateman, A. 2005. Visualizing profile-profile alignment: Pairwise HMM logos. *Bioinformatics* 21:2912-2913.

Schuster-Böckler, B., Schultz, J., and Rahmann, S. 2004. HMM Logos for visualization of protein families. *BMC Bioinformatics* 5:7.

Söding, J. 2005. Protein homology detection by HMM–HMM comparison. *Bioinformatics* 21:951-960.

Contributed by Benjamin Schuster-Böckler and Alex Bateman
Wellcome Trust Sanger Institute
Hinxton, Cambridge, United Kingdom