[3] Di Francesco F, Garnier J, and Munson PJ (1997) Protein topology recognition from secondary structure sequences: application of the hidden Markov models to the alpha class proteins. Journal of Molecular Biology 28 267(2) : 446-63.

[4] Eddy SR. (2001) HMMER: Profile hidden Markov models for biological sequence analysis (http://hmmer.wustl.edu/). 2001.

[5] Henderson J, Salzberg S, Fasman KH. (1997) Finding genes in DNA with a Hidden Markov Model. Journal of Computational Biology 4(2) :127-41.

[6] Krogh A, Brown M., Mian IS, Sjolander K, and Haussler D.(1994) Hidden Markov models in computational biology: Application to protein modelling. Journal of Molecular Biology, 235 :1501-1531.

[7] Krogh A, Larsson B, von Heijne G, Sonnhammer EL (2001) Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. Journal of Molecular Biology 305(3) : 567-80.

[8] Krogh A., Mian IS. and Haussler (1994) D. A hidden Markov model that finds genes in E. coli DNA. Nucleic Acids Resarch 22 : 4768-4778.

[9] Lukashin AV. and Borodovsky M. (1998) GenMark.hmm: new solutions for gene finding. Nucleic Acids Research 26(4) : 1107-1115.

[10] Rabiner LR A tutorial on hidden Markov models and selected applications in speech recognition. Proceeding of the IEEE 77 : 257-286.

[11] Clote P., Backofen R. (2000) Computational Molecular Biology An Introduction, John Wiley & Cons, LTD, Chichester UK.

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

this is 4096 parameters. For $k = 8$ (3 codons) the number of parameters grows to 262144. This number is to large to be estimated using genomic data for most microbial genomes. Thus one of frequently used programs GenMark uses $k = 5$. The microbial gene finding program Glimmer attempts to optimize the order used by adjusting it on fly to the amount of gathered data (the methods referred to as interpolated Markov Model).

Additional challenge in gene finding is introduced by horizontal gene transfer. Such horizontally transferred genes have different evolutionary history and consequently will have a different statistical signature. The authors of program GenMark.HMM applied to e.coli proposed a model that has two branches: one to referred to as typical to recognize what is assumed to be a typical e.coli genes and one atypical.

In the case of eukaryotic genome, the gene finding process is more complex. Let us focus on the gene region again. The main goal here is to distinguish introns and exons. One useful information is that qt the beginning of an intron there is a donor splice side at the end the accentor splice side. Both sites have characteristics (although short) signals. The region inside the two splice sides has different statistical properties that the coding (exon) regions. Since introns don't have the codon structure, a first order Markov models seems to be sufficient to model this region. The process of modeling the intron/exon structure of the eukaryotic genome is additionally complicated by the so called *frame shift*. Namely and intron can interrupt an exon at any time, not necessarily after a complete codon. The following exon has to continue from the next codon position. One way of resolving this problem (applied in the program GenScan), is to have tree separate models shifted by one position, each modeling a different frame shift.

6 Bibliographical notes and web servers

NOT FINISHED The first HMM gene finding algorithm, ECOPARSE, was designed specifically for E.coli (Krogh et. al).

References

[1] Burge, C. and Karlin, S. (1997) Prediction of complete gene structures in human genomic DNA. Journal of Molecular Biology 268: 78-94.

[2] Bystroff C, Thorsson V, Baker D. (2000) HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins. Journal of Molecular Biology. 301(1) : 173-90.

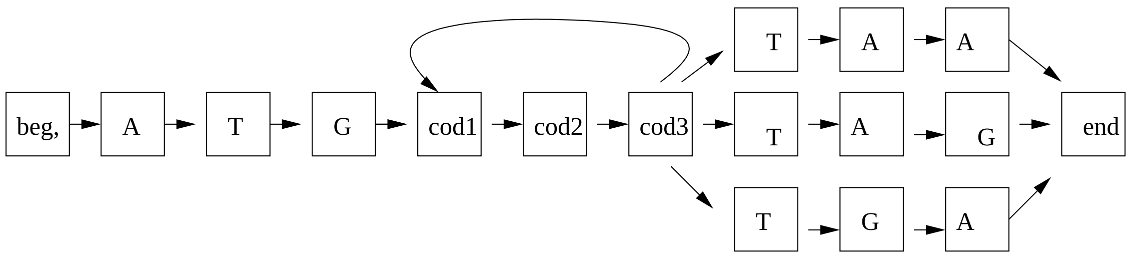

Figure 7: A diagram of a simple HMM for coding region in a prokaryotic DNA. cod1-3 correspond to three position of a codon. The standard start codon. ATG and end codons: TAA,TAG,TGA are assumed.

A probabilistic model associated with each state can be an HsMM, decision tree, neural network etc.

In a prokaryotic genome, the DNA sequence that encodes a protein is continuous and enclosed on a so called *open reading frame* (ORF): a DNA sequence of length $3k$ for some integer k that starts with the "start" codon and ends with an "end" codon and does not contain an "end" codon at positions $3i+1, 3i+2, 3i+3$ for any $i < k$. A simplified prokaryotic gene-finding HMM is depicted in Figure 7.

One obvious problem with this simple approach is that does not forbid a stop codon in the middle of a gene. Furthermore, it does not capture many of interesting statistics of the coding regions like codon usage specific for the organism, the distribution of occurrence of consecutive amino-acid pairs in the protein etc. All of the above are easily accommodated by adding memory to the Markv process. This is formally done by introducing higher order Markov models.

In Markov model of order k the transition probabilities depend k directly preceding states. Thus the transition probability matrix $a(i,j)$ describing the probability of moving form state i to j is replaced by a $k+1$ dimensional probability matrix: $a(i_1, i_2, \ldots, i_{k-1}, i_k, j)$ that defines the transition probability form i_k to j assuming that the last k visited states ware $i_1, \ldots i_k$. The k^{th} order Markov model for a coding region is usually no longer hidden. In addition to the "begin" and "end" states we have four sates, each corresponding to one nucleotide (each such nucleotide sate can be seen as a state that emits its label). If the model is of order 2 the transition probability depends on two previous sates and therefore the model can capture statistics related to one codon (e.g. codon preference for a given organism). A 5^{th} order Markov model can capture statistics related to two consecutive codons (e.g. preferential occurrence of certain amino-acid pairs, etc.). Obviously, the higher is the order, the more memory the model possesses and the more sophisticates statistics can be applied. The limit on the order of the model is imposed be the size of training data. The transition table model of order k has dimension $k+1$ thus it has 4^{k+1} parameters. For $k = 5$

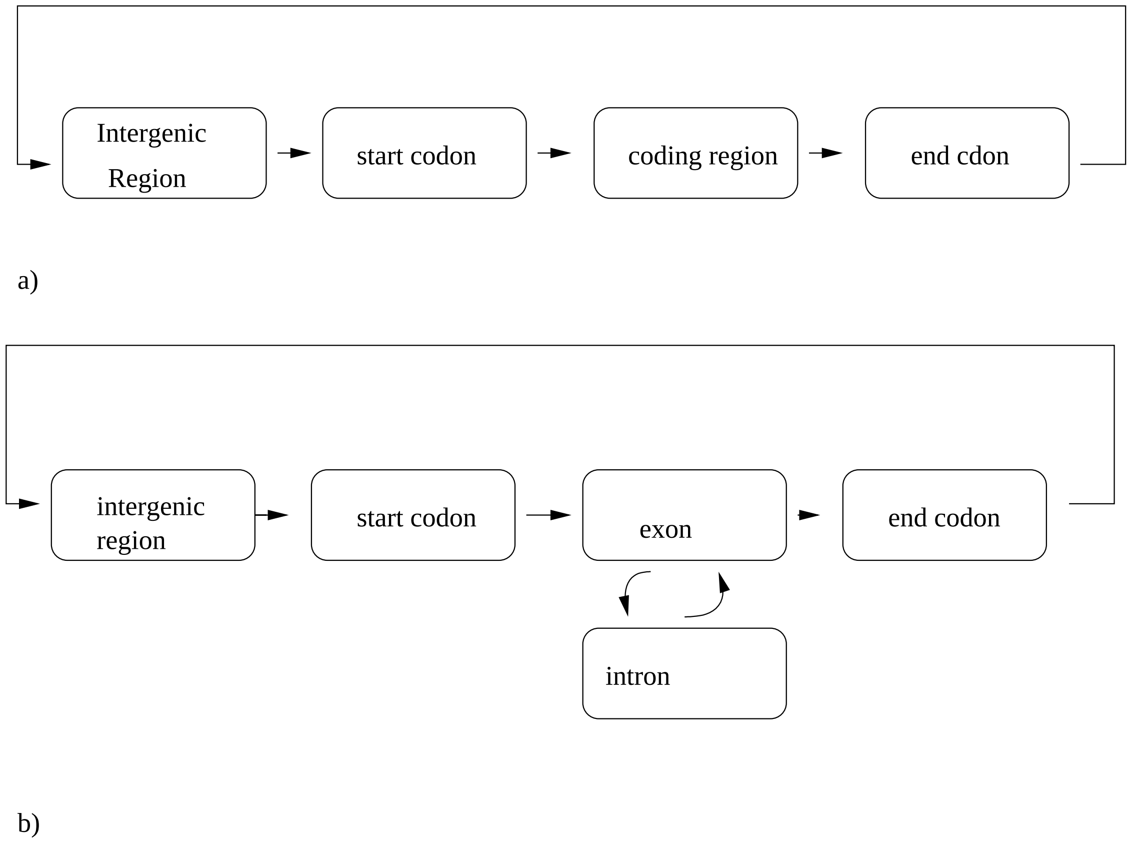

Figure 6: Simplified view of the genome for prokaryotes (a)and eukaryotes (b)

of known structures. Each I-side motif is then represented as a chain of Markov states where adjacent positions are represented by transitions. Overlapping motifs are represented by brunching structures. For each state, there are four categories of emission a symbol corresponding respectively to amino-acid resides, secondary structure, backbone angle region, and structural context (e.g. hairpin, middle strand etc). The initial success of the method was measured in terms of correctly predicted secondary and super secondary structure and 3D context.

Like any machine learning models, the power of HMM models increases with the amount of statistical data collected. The limitation of the model is its is linearity and thus inability to capture non-local dependences.

5.3 HMM based approach to gene finding

The goal of gene finding/prediction is recognizing genes within a DNA sequences. Here we will not be focusing on the details of any particular approach, but we will present basic ideas behind the use of HHM models in gene prediction. Due to the differences in the structure of eucaryote and prokaryote genomes the structure of HMM for these two groups are different. A simplified view of the genome organization is presented on the figure 6. Most of the regions denoted by one block can be further subdivided if we want to capture more details: e.g. promoter region within the itragenic part, donor and acceptor splice sides within an intron, etc.

The underlying assumption that makes HMM approach possible is that the regions described by blocks on Figure 6 are statistically different. The high level diagram like presented on Figure 6 can be in fact viewed as a graphical depiction of generalization of a HMM known as *Hidden semi-Markov Model* (HsMM). In an HsMM each state generates an entire sequence of symbols rather than a single symbol. Thus state has associated with it a length distributions function and a stochastic method to generate a sequence. When a state is visited the length of the sequence is randomly determined form the length distribution associated with this state. Formally, a Hidden semi-Markov Model is described by a set of four parameters (Q, π, a, f, E):

- A finite set Q of states.
- Initial state probability distribution π .
- Transition probabilities $a(i, j)$
- For each state q a length distributions function f_q defining the distributions of the length of the sequences generated bu q
- Fore each state q a probabilistic models E_q, according to which output strings are generated upon visiting state q.

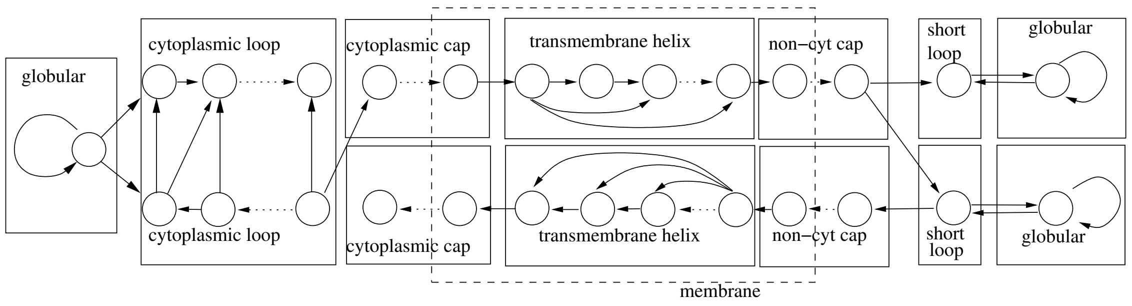

Figure 5: HMM for the transmembrane helical domain

Following construction of an HMM for the transmembrane helical regions - HMMs for transmembrane β-barrels ware proposed. Unlike transmembrane helices, the strands in a transmebrane β-barrels do not correspond to hydrophobic stretches of residues. Only one side of β-barrel is hydrophobic: the “inside” forms a pore which is makes contact with water. Although the details of a model are different than for α-helices the top level idea remains the same: assuming that each state of the model corresponds to a position of amino-acid in the structure and dividing the states of the model into groups corresponding to structural subunits (Liu, Martelli)

The HMM approach has been also tried for fold recognition for globular proteins. The fold recognition problem is a decision problem: given a fold and a protein sequence, does the protein in its native state assume a given fold? The linear nature of the model dictates that the main focus of this approach is on recognition of local structural patterns. Such approach cannot account for long-range interactions between distant protein residues. However, it has been shown that secondary structure information alone: order of secondary structure, their types, lengths and length of loop regions already provides significant discriminatory information (cite Przytycka, other citations) thus it should be not surprising that this approach has some success. In particular, Kevin Karplus proposed an extension of HMM to a multi-track HMM. A multi-track HMM has the same basic structure as a regular HMM but instead of one table of emission probabilities it has two independent tracks defined by two emission tables: one for amino acid emission and the second for emission of the secondary structure descriptor for a given residuum (strand, helix, loop) (cite CASP5 predictions).

Perhaps more surprising is the attempt to use the HMM machinery for novel structure prediction. The idea implemented in the program HMMSTR [**?**] is to use an HMM do describe a general grammar for protein structure. The building blocks are local structural motifs common to diverse protein families. These, so called “I-sites” are taken from a library of short sequence motifs obtained by clustering form a non-redundant data base

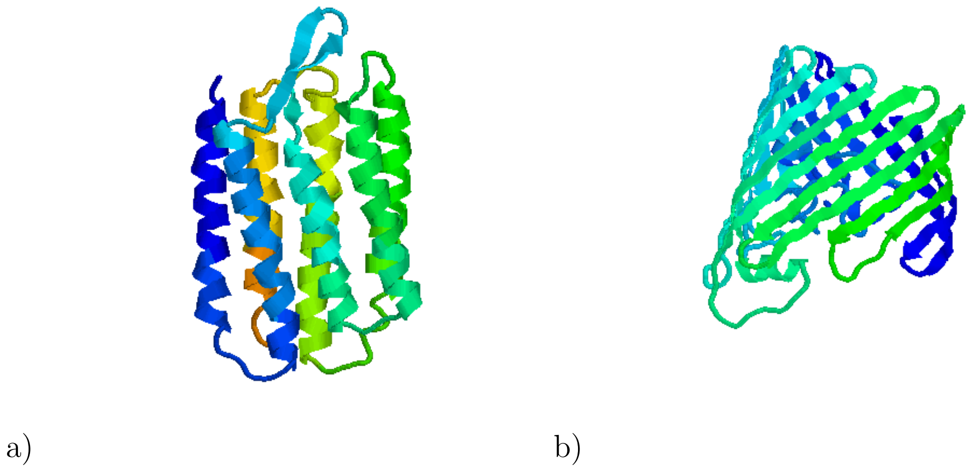

Figure 4: A transmembrane helical bundle (a) and a transmembrane barell (b)

by machine learning methods. Simple (non HMM based) methods for recognizing A transmembrane α-helix simply search for long sequences of hydrophobic residues. Such long hydrophobic sequences are unlikely to occur in globular water solvable proteins. At the same time, hydrophobic residues interact favorably with the lipids in the membrane. Such transmembrane helical bundle can be described by a simple grammar: $(CH\overline{C})^n|(\overline{C}HC)^n$, where H is a transmebrane helix, C is a part of polypeptide chain embedded in cytoplasm (the cytoplasmic loop), $\overline{C}$ is the non-cytoplasmic part embedded outside of the cell or inside an organelle(the non-cytoplasimc loop) and H is a transmembrane helix. The characteristic aspect of cytoplasmic portions is that they are usually enriched in positively charged residues (argenine and lysine). This is often referred to "positive inside" rule. Unfortunately the non-cytoplasmic loops can contain whole globular domains which can also contain positively charged residues. The second obstacle is the existence of the so called signaling peptides that are also characterized by starches of hydrophobic residues and the models has to be able to tell apart the trnasmembrane helices and these signaling peptides.

The basic architecture of the HMM model proposed by Sonhammer at al is given if figure **??**. Each residuum of a protein corresponds to a unique state of the model. The sates of the model are divided into groups that depend on the position of a residuum relative to the membrane. The states in each group have the same parameters. Additional complication encountered in the construction of this HMM comes with the training process: the precise position of helixes are often not known. The solution to this problem proposed by Sonnhammer and other was to predict precise boundaries as a part of training process (see the references for details).

Tg-ATAT
TaaATAT
T- -ATA-

Table 3: The alignment constructed for the set TGATAT,TAATAT, TATA using the HMM model the example

Then the corresponding most likely paths are "begin", M1, I1, M2, M3, M4, M5, "end"; "begin", M1, I1, I1, M2, M3, M4, M5, "end"; "begin", M1, M2, M3, M4, D5, "end". The symbols generated in the same matching state are aligned; D5 in the sequence of last path is translated to a deletion in the matching column number 5. Note that first and second sequences have insertions between state M1 and M2. The insertions are of different length. The HMM does not specify how subsequences generated in the insertion states are to be aligned. Writing the unaligned letters in lower case the resulting alignment is:

Note that the techniques developed in this section can be also in used to produce multiple alignment of a set of unaligned sequences. Namely, all what needs to be done is to train a generic profile HMM using sequences that are to be aligned and the Baum-Welch training. After the model is trained, we can find the most likely path for each sequence and derive multiple sequence alignment as described above. In practice HMM are often constructed from a "seed" alignment: a trusted alignment for a subset of sequences in the family.

Finally, we need to point to an important technical detail. Namely, algorithms for HMM described in the previous section need to be adapted to the situation where some states are silent. Recall that these algorithms assumed that in the t^{th} time step the t^{th} element of the sequence is generated. This is not true if we have silent states. We will not go into technical details of the modifications but only note that they are is quite similar to the modifications we introduced to convert the pair-wise sequence alignment algorithm with constant gap penalty to an algorithm that allows afine gap penalty.

5.2 Protein Structure recognition and prediction

Following the success in sequence analysis, HMM are increasingly applied to prediction and/or recognition of protein 3D structure. One of first steps in this direction was an HMM model for predicting transmembrane helices in protein sequences **??**. In general, transmembrane proteins fall into two classes: α-helical bundles and transmembrane β barrels. A trans-membrane α-helical bundle consists of a number of helices, each of which is spans the membrane (compare figure 4 a). Transmembrane β-barrels consist of an even number of strands, each strand spanning the membrane and the strands are positioned so that they form the surface of a "barrel" (see Figure 4 b).

Between the two groups the first is easier to recognize and was the first to be approached

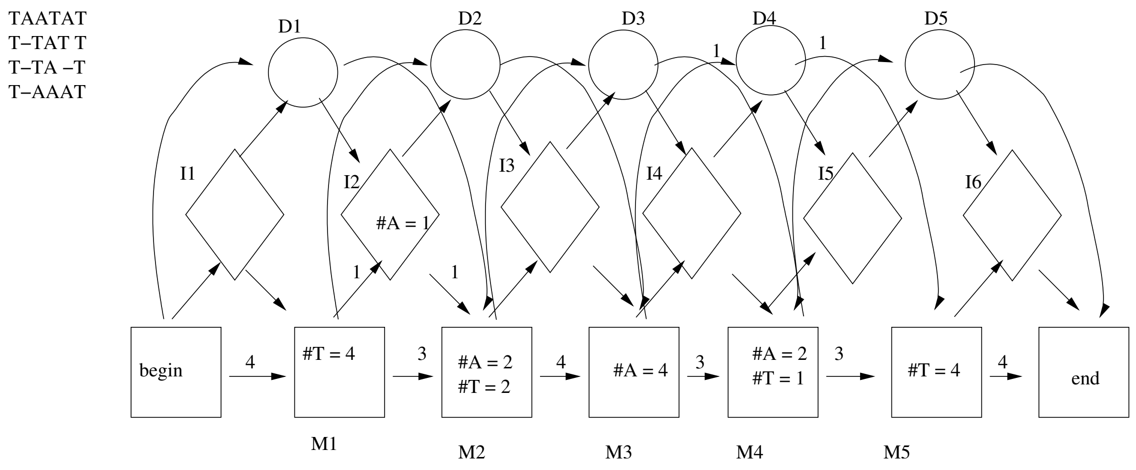

Figure 3: Profile HMM for a toy sequence family. $M1 - M5$ are matching states, $I1 - J6$ insertions sates and $D1 - D5$ delete states. The are the counts obtained in the training procedure performed under assumption of known paths

a set of unaligned sequences from this family we can us the HMM to construct a multiple alignment for this set.

To illustrate parameters estimation for an alignment, consider the following hypothetical toy sequence on Figure 3. We will use a model with 5 matching states and let the matching states correspond to columns 1,3,4,5,6 (compare Fig 3, ignore the labels for the moment). Then for each sequence in the family we have a unique path in the model. For example the path for sequence TAATA sequence is: "begin", M1, I2, M2, M3, M4, M5, "end"; for sequence T-TAAT is "begin" M1, M2, M3, M4, M5, "end"; for sequence T-TA-T: "begin" M1, M2, M3, D4, M5, "end" and for sequence T-AAAT: "begin" M1, M2, M3, M4, M5, "end". Thus we use the training method with known paths. The counts of each transitions end emissions are given as the labels. The rest of the values are zero (thus our training set is not sufficiently large). Now the transition and emission probabilities are set using equations (15) and (16).

To illustrate the method of computing multiple sequence alignment given a model, assume that we have given a complete HMM M (with all the parameters) and we would like to find a multiple sequence alignment for a set of sequences from the family modeled by M. Do so, find using the Viterbi algorithm the most likely path for each sequence. These most likely paths can be converted into multiple alignment in a natural way: positions from different sequences that go through the same matching state are aligned (they are in the same column in the multiple alignment), all position that are generated in the insert state are inserted and all the delete states correspond to deletion.

For example, assume that the sequences to be aligned are TGATAT,TAATAT, TATA.

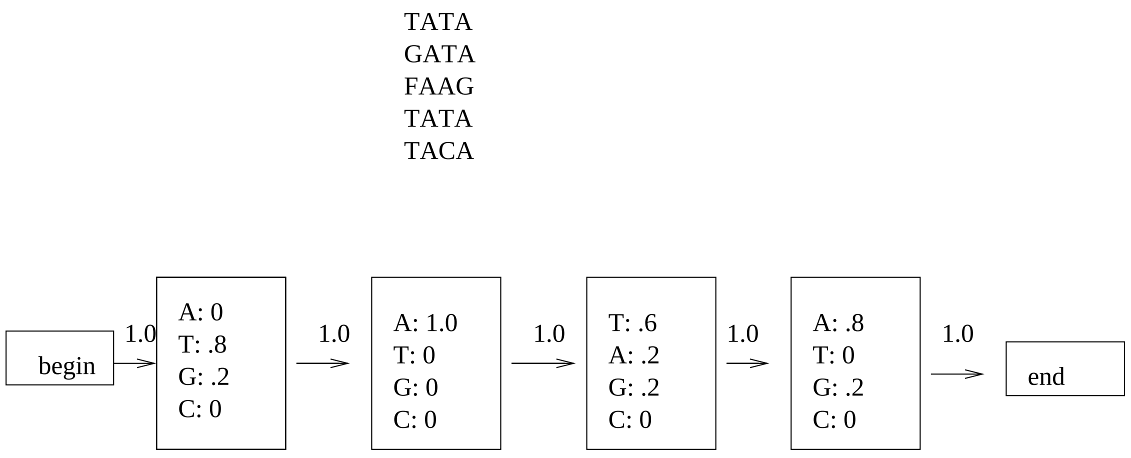

Figure 2: An HMM for a sequence family with a gapless alignment. Pseudo-counts are not added.

has no gaps. The profile of such set of sequence can be represented by a linear HMM model, with the number of states (in addition to the "begin" and the "end" sates) equal to the length of the profile as illustrated on Figure 2. The only allowed transitions are form the state corresponding to one alignment column to the state corresponding to the next alignment column. The transition probability is equal to one. The emission probabilities for the state corresponding to a given column are equal to the frequency of amino acids in that column. Anther than the terminology. Such HMM is equivalent to a profile representation of the family.

I assume that the reader knows what is a profile

Including insertions and deletions to the multiple alignment requires a more interesting HMM topology. The model proposed by Krogh *et.al* has 3 types of nodes: match, insertion and deletion (compare Figure 3 where match, insertion and deletion states are represented respectively with rectangles, diamonds and circles). The number of match states often is set to the average length of the sequences in the family. The insertion state allow for inserting residues that do not correspond to matching states. The delete states are special: they do not emit any symbols. The need for delete states may seem at first non-obvious: we could have edges going from every insert/match to any insert/mach state that occur further in the model. However, there are two reasons for which delete states are useful: they reduce the number of edges in the model and they have a natural interpretation in terms of sequence alignment. Namely, each delete state corresponds to deleting a single residuum.

The design of a profile HMM makes an explicit connection between the model and the multiple sequence alignment of the members of the modeled family. Given a multiple alignment the estimation the parameters of the model is done using the algorithm for training with known paths. The reverse is also true: given a profile HMM for a sequence family and

In step 3, the new parameters are assigned to the model. The new values of $a(i,j)$ and $e(i,j)$ are computed from values (21) and (22) using equations (15) and (16)

It can be shown that Expectation Maximization method converges to a local maximum. More precisely, method guarantees convergence of the target function; here maximum expectation generating the sequences from the training set. It does not guarantees convergence of the values of the parameters. In practice, the convergence criterion is usually set as until the change is the parameter estimates becomes negligible (or a local maximum is reached).

The fact that Expectation Maximization method converges to a local maximum and not necessarily to the global maximum is a reason for concern, especially for large models that are likely to contain multiple local maxima. Several known heuristics, like simulated annealing or gradient descent method, are often used to increase the probability of reaching the global maximum.

4.3 The Viterbi training

In some application a different training algorithm, based on the Viterbi algorithm for most likely path is used. This approach is often referred to as the *Viterbi training*. The idea is to replace the computation of $A(i,j)$, $E(i,j)$ that is expected number of transitions and emissions by the following computation: First using the Viterbi algorithm find the for each training sequence the most likely path and then compute number the number of emissions and transitions as in the case of the algorithm with known path. The training process is then iterated. Although theoretically less sound, some HMM for biological sequences are trained this way. For example, an HMM data base for sequence motifs Meta-MEME [**?**] is trained using the Viterbi training arguing that the Viterbi path is interpretable in terms of the evolution of the given sequence.

5 Applications

Hidden Markov models found numerous applications in computational biology. In this section we sketch three such applications: to modeling protein sequence families, protein structure prediction/recognition and gene finding. The detailed description of the construction of each HMM, it's training, evaluating the performance and further improvements are beyond the scope of this book. Thus we focus on presenting the main ideas and basic directions.

5.1 Profile HMM

In 1994 (DOUBLE CHECK) Krogh *et.al.* introduced an HMM architecture that models protein families. Assume, that we have given a set of sequences from some sequence family together with a multiple alignment of its members. First assume that this multiple alignment the frequency of observing symbol σ_j in the database (indepdendntly of i). More advanced methods include data-dependent pseudocoutns and Dirichlet mixtures (Manfred J.Sippl. Calculation of conformational ensembles from potentials of mean force: an approach to the knowledge base prediction of local structure in globular proteins, JMB 213: 859-833), 1990; Karpulus).

4.2 Baum-Welch training

The HMM training becomes more challenging if the path of states that model should use to generate each of S_i in the training set $\mathcal{S}$ is unknown. The main strategy in this case is to start with some prior probability distribution for emission and transition probabilities and then iteratively improve the model using the training set data so that expectation of generating the sequences form the training set increases. More precisely, the Expectation Maximization method approaches this problem as follows:

1. Assign initial values to the parameters according to assumed prior distributions.
2. For each sequence in the training set, compute the expected number of times each transition/emission is used. This can be done efficiently using the algorithms described in the previous section.
3. Estimate new parameters of the model using the expected values from step 2.
4. Repeat steps 2 and 3 until an convergence criterion is reached.

With the tools developed in section 2 step 2 is straightforward. Namely for the k^{th} sequence in the training let $a_t^k(i,j)$ be the probability that the process visits state i in step t and state j is step $t+1$. Similarity, let $\tilde{e}_t^k(i,j)$ be the probability that symbol σ_j is emitted in time t at state i for sequence S^k. Using the forward and backward variables we have:

$$\tilde{a}_t^k(i,j) \quad = \quad P[q_t = i, q_{t+1} = j | S^k, M] = \frac{F^k(t,i)a(i,j)e(j,t+1)B(t+1,j)}{P[X^k|M]} \tag{19}$$

$$\tilde{e}_t^k(i,j) \quad = \quad P[q_t = i, S_t^k = \sigma_j | S^k, M] = \{ \begin{array}{ll} \frac{F(i,j)B(i,j)}{P[X^k|M]} & \text{if } S_t^k = \sigma_j \\ 0 & \text{otherwise} \end{array} \tag{20}$$

Then the expected number of times for transition and emission probabilities can be computed summing the respective probabilities over all time steps and over all sequences in the training set:

$$A(i,j) = \sum_{k=1}^{N} \sum_{t=0}^{\text{len}(X^k)} \tilde{a}_t^k(i,j) \tag{21}$$

$$E(i,j) = \sum_{k=1}^{N} \sum_{t=0}^{\text{len}(X^k)} \tilde{e}_t^k(i,j) \tag{22}$$

resulting model will be biased towards this particular group. To avoid this effect a *weighting schemes* are usually applied to the training data. For example the training data may be divided into clusters of sequences which have pair-wise similarity above certain threshold and each sequence is assigned a weight one over the cardinality of the cluster it belongs to. In this way all clusters will have equal weight in the training process.

Weighting of training sequences - refer to the same issue elsewhere in the book. If not defined elsewhere it needs to be expanded

4.1 Training an HMM under assumption of known paths

Let $\mathcal{S}$ be a training set and let for every sequence $S \in \mathcal{S}$ path of states that the model should take while generating S is known. Under this assumption the training process is relatively simple. Namely, let $E(i,j)$ be the total number of times (over all training sequences) when a symbol σ_j is emitted in state i. Similarly, let $A(i,j)$ be the total number of times (again over all training sequences) when the transition form state i to state j takes place. Then $e(i,j)$ and $a(i,j)$ are estimated as follows:

$$e(i,j) = \frac{E(i,j)}{\sum_k E(i,k)} \tag{15}$$

$$a(i,j) = \frac{A(i,j)}{\sum_k A(i,k)} \tag{16}$$

The fundamental question that one needs to consider during the training process is whether the training set contains enough data to estimate correctly transition and emission probabilities. Lack of data leads to over-fitting of the model: the model cannot properly generalize the training data. If no statistics can be extracted, then the model can represent only the training data. Note that the number of all parameters is $O(n^2 + nm)$ thus may be quite big.

One technique to deal with this problem is to use *pseudocounts*. Pseudocunts are values added to counters $E(i,j)$ and $A(i,j)$ to avoid zero probabilies. Let $\hat{a}_{i,j}$ and $\hat{e}_{i,j}$ be the corresponding pseudocoutns values for an HMM. Pseudocouts correspond to prior knowledge on corresponding probability distributions. In this context we can assume that for each i $\hat{a}_{i,j}$ and $\hat{e}_{i,j}$ are normalized to sum up to one. Then equation 15 and 16 become respectively:

$$\begin{aligned} e(i,j) &= \frac{E(i,j) + \alpha\hat{e}_{ij}}{\sum_k E(i,k) + \alpha} \qquad (17) \\ a(i,j) &= \frac{A(i,j) + \beta\hat{a}_{ij}}{\sum_k A(i,k) + \beta} \qquad (18) \end{aligned}$$

where α and β are "scaling" parameters that reflect the weight we put on the pseudocouts.

Explain Dirchlet mixtures here ?

The simplest method of assigning pseudocounts is simply setting them uniformly. However, in the case of emission probabilities natural pseudocount values are to let $\hat{e}_{i,j}$ be equal to

Unfortunately, the probability of the sequence $P[S]$ as well as the probability of the model $P[M]$ are unknowable. The standard approach is to consider odds the probability of model M, to the probability of a null model N:

$$\frac{P[M|S]}{P[N|S]} = \frac{P[S|M]P[M]}{P[S]}\frac{P[S]}{P[S|N]P[N]} = \frac{P[S|M]}{P[S|N]}\frac{P[M]}{P[N]} \tag{13}$$

The null model is a model that attempts to model all sequences in the universe of sequences considered (e.g. all protein sequences). The sequence fits a model M if $\frac{P[M|S]}{P[N|S]} > 1$. Practical considerations dictate using log-likelihoods instead of the odds ratios. In particular, this circumvents the numerical issues related to dealing with very small values $P[S|M]$ and $P[S|N]$. Using the log-likelihoods we the condition for S fitting the model M can be rewritten as:

$$-\log(P[S|M]) > \log\frac{P[M]}{P[S|N]P[N]} \tag{14}$$

In the inequality above the $\frac{P[N]}{P[M]}$ is the prior estimation of the relative probabilities of the model M and the null model. If we assume that model N generates the sequences from the sequence universe with uniform distribution and that size of the universe is equal to the size of the sequence data base that then $\frac{1}{P[S|N]}$ is the size of the data base.

4 Training of an HMM

Assume that we have given the topology of a hidden Markov model, M but not its parameters. Let $\mathcal{S}$ be a set of representative sequences form the modeled family. i.e. *training set*. The goal of the training process is given the set $\mathcal{S}$, assign parameters to M so that it maximize (under prescribed topology) the probability of generating sequences in $\mathcal{S}$. Formally, training problem is described as follows:

Given a training set $\mathcal{S} = \{S_1, ..S_N\}$ and a topology of an HMM, M, find emission and transition probabilities that maximize the likelihood of generating $S_1, \ldots, S_N$ by the model. The usual assumption is that $S_1, \ldots S_N$ are independent and therefore $P(S_1, \ldots S_m|M) = \Pi_i(S_i|M)$ and thus we seek parameters assignment that maximize $\Pi_i(S_i|M)$.

The details of the training step depend on whether for the paths of states, which the HMM model should use to generate S_i are known in advance. If they are known then the training step is very simple. Otherwise more elaborated methods are applied.

In practical applications, the independent sample condition is rarely met. The training set often contains nearly identical sequences. This may lead to a skewed parameter estimates. Namely, if within a training set there is a large group of very similar sequences, then the at any given time using any path of states. That, for any given state k and time step t we would like to compute the probability that the HMM is in state k at time t: $P[q_t = k|S, M]$.

This problem can be solved by a combination of the forward algorithm presented before with an almost symmetric *backward algorithm* that we are about to introduce. The backward variable, $B(k,t)$ is equal to the probability of generating the subsequence $s_{t+1}, \ldots s_m$ using state k as the starting state and ending at the usual "end" state $n+1$. The backward variable is computed similarly as the forward variable but the algorithm is executed in the "backward" direction (model with begin/end):

$$\begin{aligned} B(n+1,n+1) &= 1 \\ B(k,n+1) &= 0 \text{ for } k < n+1 \\ B(n+1,t) &= 0 \text{ for } t < T \\ B(k,t) &= \sum_j B(j,t+1)a(j,k)e(j,s_{t+1}) \quad \text{otherwise} \end{aligned} \tag{9}$$

For completeness, we include the variant of the forward algorithm for the model wit begin/end states:

$$\begin{aligned} F(0,0) &= 1 \\ F(k,0) &= 0 \text{ for } k \neq 0 \\ F(0,t) &= 0 \text{ for } t \neq 0 \\ F(k,t) &= e(k,s_t)\sum_j F(j,t-1)a(j,k) \text{ for } t > 1 \end{aligned} \tag{10}$$

By definition of $F(k,t)$ and $B(k,t)$ follows that:

$$P[q_t = k|S, M] = (F(k,t)B(k,t))/P[S|M] \tag{11}$$

Thus $P[q_t = k|S, M]$ for all time steps t and alls sates k can be computed in $O(n^2T)$ time.

3 Scoring

One of the most important applications of HMM in biological sequence analysis is to use the model to discriminate whether or a given sequence belongs to the family. Formally, the question is: given a sequence S and a model M what is the probability $P[M|S]$ that the sequence S comes form the model M. Recall that the forward algorithm form section 2 computed $P[S|M]$ not $P[M|S]$. The two quantities are related by Bayes rule:

$$P[M|S] = P[S|M]\frac{P[M]}{P[S]} \tag{12}$$

$$\begin{aligned} V(k,0) &= 0 \text{ for } k \neq 0 \\ V(0,t) &= 0 \text{ for } t \neq 0 \\ V(k,t) &= e(k,s_t)\max_j V(j,t-1)a(j,k) \text{ for } t,k>0 \end{aligned} \tag{8}$$

However to annotate a sequence we need the path p^* itself. Computation the sequence of states this path is done using the technique presented in the context of recovering the optimal alignment from the dynamic programming table constructed for computing optimal alignment score. Namely, let $\text{argmax}_j(V(j,t)a(j,k))$ be the index of state j that maximizes the value $V(j,t-1)a(j,k)$. Thus argmax gives the pointer to the last but one vertex on the most likely paths likely path that generates subsequence $s_1,\dots,s_t$ and ends at state k at step t. This pointer is stored as $ptr(k,t)$ The Viterbi algorithm thus is therefore summarized as follows:

The Viterbi Algorithm:

```
1. $V(0,0) = 1$;
   forall $k \neq 0$ $V(k,0) = 0$;
   forall $t \neq 0$ $V(0,t) = 0$;

2. for $t = 1$ to $T$ do
   for $k = 1$ to $n$ do
   $V(k,t) = e(k,s_t)\max_j V(j,t-1)a(j,k)$
   $ptr(k,t) = \text{argmax}_j(V(k,t), a(j,k))$

3. $P[S|p^*, M] = \max_j V(j,T)$
   $ptr(n+1, T+1) = \text{argmax}_j(V(j,T))$

4. $p^*$ = the reverse of the path from state $T+1$ to state 0 using pointer $ptr$.
```

The algorithm takes $O(n^2T)$ time.

2.3 Computing the probability of visiting a given state at a given time step: the Forward-Backward algorithm

Finding the most likely path is one way to find the positions in the sequence that correspond are generated in interesting states of the model. However, if the sequence can be generated using a number of roughly equivalent paths such a most likely path may not be very informative. What may be more interesting is to find the probability of visiting any particular state

$S_t =$	1	1	2	6
$F[k,t]$	$t=1$	$t=2$	$t=3$	$t=4$
$k=1$ (A)	$.5 \times \frac{1}{6}$ $= 0.83$	$.125 \times .9 \times \frac{1}{6}+$ $= .01875$	.00385	.00039
$k=2$ (B)	$.5 \times \frac{1}{4}$ $=.125$	$.083 \times 1.0 \times \frac{1}{4}+$ $.125 \times .1 \times \frac{1}{4} = .0239$	.0026	.00096
$P[S\|M] =$				0.00135

Table 2: Computing the matrix F in the Forward algorithm.

2.2 Computing the most likely path: the Viterbi algorithm

Recall that a sequence S can be usually generated using deferent paths of states. It is often interesting to know with of the possible paths is the most likely one. Since the states of an HMM usually correspond to biologically relevant positions in the sequence, such most likely path can be used to annotate the sequence S. The process of finding the most likely path is also called *aligning the sequence to the model.*

The *most likely path* for generating a sequence S in model M is the path, p^*, that maximizes the probability of generating the sequences:

$$P[S|p^*, M] = \max_p P[S|p, M] \tag{7}$$

Thus although the states of a hidden Markov are not directly observable, the most likely path provides information about the most likely sequence of such "hidden" states. In particular, based on Table 1, we know the most likely path to generate the sequences 1,1,2,6 is A, B, A, B.

The value $P[S|M]$ is computed using the dynamic programming algorithm, known as *Vitrebi algorithm.* The basic variable, $V(k,t)$, in the Virterbi algorithm is be equal to the probability of the most likely path that generates subsequence $s_1, \ldots, s_t$ and ends at state k at time step t. Then the probability of the most likely path is $\max_k V(k,T)$ for the variant without specified "end" state or $P[S|p^*, M] = V(n+1, T+1)$ if $n+1$ is the "end" state. In subsequent presentation we assume an HMM model with begin/end states.

The recurrence for computing the value of $V(k,t)$ is closely related to the recurrence for computing $F(k,t)$ in (5) except we are no longer interested in the sum of probabilities over all possible paths but in the most likely path, thus the summation in (5) is replaced by max. Consequently, the recurrence for computing $V(k,t)$ is given by the following formula:

$$V(0,0) = 1$$

2.1 Probability of generating a sequence by an HMM: the Baum-Welsh algorithm

As we pointed out before, the knowledge of the probability of generating of a sequence S by an HMM M alone does not suffice to discriminate between members and non-members of the modeled family. However, as we will see in section 3 this value in combination with additional information provides the basis for such discrimination.

Computing $P[S|M]$ directly from the definition (4) is not feasible due to an exponential number of possible paths (compare also the list in Table 1). Fortunately, the problem admits a dynamic programming formulation. Let $F(k, t)$ denote the probability of generating subsequence $s_1, ...s_t$ using a path that ends at state k. Observe that before entering state k in time step t any other state could be visited in step $t-1$. By the definition, the probability of visiting state j in step $t-1$ (and generating in this state symbol s_{t-1}) is $F(j, t-1)$. The probability of the move from state j to k is $a(j, k)$. Thus $\sum_j F(j, t-1)a(j, k)$ is the probability of reaching state k in step t. That sum multiplied by $e(k, s_t)$ (the probability of emitting symbol s_t in state k) is exactly equal to $F(k, t)$. Thus we have the following recurrence:

$$\begin{aligned} F(k,1) &= e(k,s_1)\pi[k] \\ F(k,t) &= e(k,s_t)\sum_j F(j,t-1)a(j,k) \text{ for } t>1 \end{aligned} \tag{5}$$

If there is no special "end" state then probability of generating the sequence S by model M is given by (with end state that would simply be $F[n+1, T+1]$):

$$P[S|M] = \sum_j F(j,T) \tag{6}$$

The recursion (5) can be readably implemented using dynamic programming technique. Matrix F is filled in the column by column fashion where each column if filled in the increasing order of k value (see Table 2). The algorithm takes $O(n^2m)$ time: there are nm values of F and computing each value involves computing a sum over n previously computed values. Since for a fixed model, n is a constant the running time is linear in the length of the sequence m.

The variable $F(k, t)$ is also called the *forward variable* and the whole algorithm the *forward algorithm*. The terminology comes form the fact that this algorithm combined with a symmetric *backward algorithm* is used as a building block for in the algorithm that computes the solution to a related problem described in a later subsection.

generating sequence S following path p. That is:

$$\begin{aligned} P[p|M] &= \pi[q_1]a(q_1,q_2)a(q_2,q_3)...a(q_T,q_T) \quad (2) \\ P[S|p,M] &= e(q_1,s_1)e(q_2,s_2)...e(q_{T-1},s_T) \quad (3) \end{aligned}$$

Finally, the probability $P[S|M]$ of generating sequence S, by an HMM, M is defined as:

$$P[S|M] = \sum_p P[S|M,p] \quad (4)$$

While designing a HMM for a biological sequence family, we typically start with a set of related sequences. The topology of the model is usually set manually based on the understanding of the modeled family, while the parameters of the model are adjusted automatically to maximize the probability of generating the input set of sequences. The process of adjusting parameters called the *training process* and the input sequence is called *the training set.*

An HMM defines a probability distribution over sequences of symbols form Σ. An HMM that models a sequence family is assumed to generate the members of the modeled family with a relatively high probability while the probability of generating an unrelated sequence is to be relatively low. We need to keep in mind that the probability of generating a sequence S by model M, $P[S|M]$, alone does not suffice to discriminate between members of the modeled family and sequences that are unlikely to belong to the family. The values of $P[S|M]$ over all sequences S sum up to one thus if we have two different sequence families modeled by two different models where one family is much larger then the other then the individual members of the large family have a smaller probability of being generated than the individual members of the small family. A method of discriminating between members and non-members of a family is described in section 3.

In the consecutive sections we describe the basic algorithms for HMMs, the scoring methods, the training methods and we finish with practical applications of HMMs to molecular biology.

2 Basic algorithms for HMM's

An HMM model describing a sequence family can be used in a number of ways. Most frequently it is used to find other members of the family. Furthermore, as we will see in the section 5, an HMM for a biological sequence family is usually designed in such a way that states correspond to biologically significant positions in the sequence family e.g. conversed residua, exons in a DNA sequence etc. An HMM can be used to annotate a sequence, that is to predict where these special positions are located in the sequence. In this section we describe algorithms that support these applications.

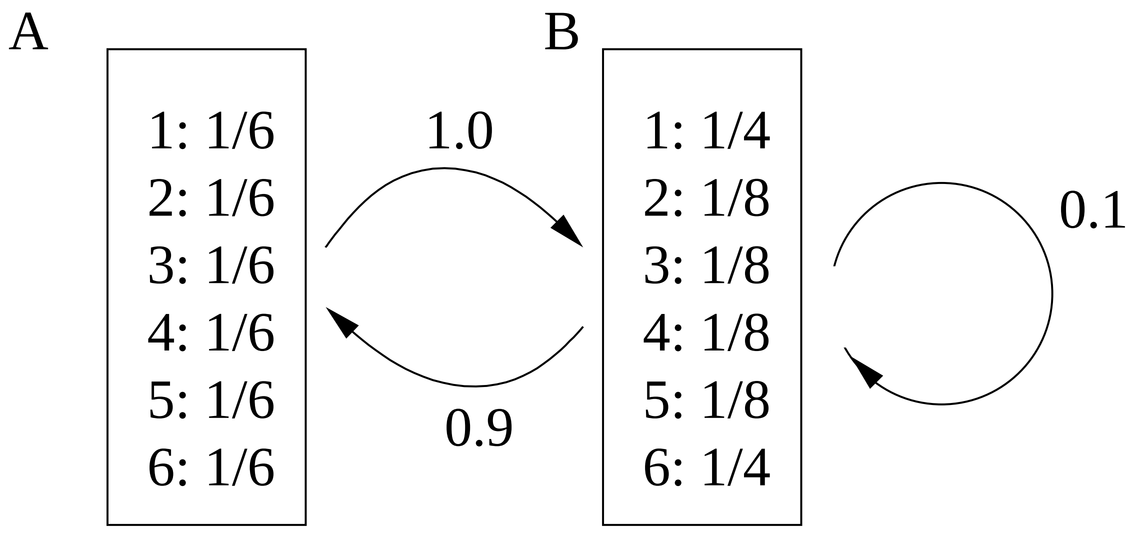

Figure 1: A graphical representation of the HMM for the Example 1. The states are represented by rectangles. The emission probabilities for each state are given inside corresponding rectangles and the edges are labeled with transition probabilities.

path p	$P[p\|M]$	$P[S\|p]$	$P[p\|M] \times P[S\|p]$
A,B,A,B	.45	.00173	.00078
A,B,B,A	.045	.00086	.000039
A,B,B,B	.005	.0013	.0000065
B,A,B,A	.405	.00086	.00035
B,A,B,B	.045	.0013	.000058
B,B,A,B	.045	.0026	.000117
B,B,B,A	.005	.0013	.0000065
$P[S\|M] =$			0.00135

Table 1: The paths that can be used to generate sequence $S = 1, 1, 2, 6$; their probabilities; probabilities of generating S assuming given path and the probability $P[S|M]$ of generating S by M.

It is often convenient to extend the set of states of an HMM to contain states 0 and $n+1$ defined to be respectively the "begin" state and the "end" state. These extra "begin" and "end" states do not emit any symbols. The process modeled by such an HMM with begin/end states always starts in the begin state and ends in the end state. In particular, there is no need to include vector π in the HMM definition. We will refer to this variant as HMM with begin/end states end state explicitly when we use this variant of the model.

An HMM is usually visualized as a directed graph with vertices corresponding to the states and the edges representing pairs of states with non-zero transition probability $a(i, j)$ of moving from state i to j. A simple HMM, visualizing the example described below, is shown in Figure 1. The graph (without the emission /transition information) defines the *topology of the model* while the emission and the transition probabilities defines the *parameters of the model.*

Example 1. Consider a two-player game where each player has his own dice. Player A has a fair dice while player B has an unfair dice: 1 and 6 come with probability $\frac{1}{4}$ each and the probabilities of the remaining numbers (2,3,4,5) are $\frac{1}{8}$. Furthermore, assume that player B cheats: namely whenever absentminded A is not looking he makes another turn. This happens with probability 0.1. Player A is fair and always makes only one turn. The probability of starting the game from any of the two players is 0.5. Assume that an observer cannot see who is throwing the dice but he can see the results of consecutive dice outcomes. The game can be modeled with an HMM (compare Figure 1). The two players correspond to two states of the model. Each state is emitting symbols from alphabet $\{1, \ldots, 6\}$ with probabilities depending on the properties of the dice which the corresponding player is using. The initiation probability vector is defined by $\pi[A] = \pi[B] = \frac{1}{2}$. The transition probabilities are 1 for transition from player A to B, 0.9 for transition from B to A, and 0.1 for transition from B to B.

Assume that the observer sees the sequence $\{1, 1, 2, 6\}$. How likely he is to see such an outcome? Note that an output sequence can typically be produced in many ways using different paths of states. For example, the sequence $\{1, 1, 2, 6\}$ can be produced using any of the following path of states: (A, B, A, B), (A, B, B, B), (A, B, B, B), (A, B, A, B), (B, A, B, B), (B, B, A, B), (B, B, B, A) and (B, B, B, B). Obviously these sequences of states are not all equally likely (compare Table 1.) To any fixed sequence of states we can assign probability of generating the given output sequence following this particular path. The probabilities of generating the sequences $\{1, 1, 2, 6\}$ using all possible state paths are listed in Table 1. The probability of generating $\{1, 1, 2, 6\}$ is equal to the sum of all these probabilities.

Formally, given an HMM M, a sequence $S = s_1, ..., s_T$, and a path of states $p = q_1...q_T$ the probability of generating S using path p in model M, $P[S, p|M]$, is equal to the product:

$$P[S, p|M] = P[p|M]P[S|p, M]. \tag{1}$$

where $P[p|M]$ is the probability of the selecting the path p and $P[S|p, M]$ the probability of

1 Introduction

Many important tools in biological sequence analysis relay on statistical information derived from a set of related sequences. For example, analysis of a multiple alignment of a set of homologous sequences reviles which amino acid positions are conserved throughout the evolution and which are not. The conserved amino acids are likely to be functionally and/or structurally important. Such information is often used for fine-tune searches for distant homologues. The substitution pattern for less conserved positions also provides important statistical information, which can be used, for example, in secondary structure prediction.

A hidden Markov model (HMM), a concept initially developed for speech recognition [?], has proven to be a very suitable tool for representing statistical information extracted from a set of related sequences. Current applications of HMMs in computational biology include, among others, modeling protein families (Eddy 2001, Krogh et al, 1993), gene finding, (Krogh et al. 1994, Burge et al, 1997, Lukashin et al, 1998, Henderson et al 1997, Salezberg, 1998), prediction transmembrane helices (2001) Krogh, tertiary structure prediction (Bystroff, 2000; DiFrancesco et al 1997).

In a hidden Markov model, a sequence (e.g. a protein sequence, or a DNA sequence) is modeled as an output generated by a stochastic process progressing through discrete time steps. At each time step, the process outputs a symbol (e.g. an amino acid or a nucleotide) and moves from one of a finite number of states to another. Thus similarly to a Markov process, an HMM is described by a set of states and a transition probabilities matrix defining the probability of moving from one state to another. Additionally the definition of an HMM requires an *emission probability matrix* defining the probability of emitting a given character at a given state. The adjective "hidden" refers to the fact that, in contrast to a Markov process, the outcome of the statistical process modeled my an HMM is the sequence of symbols generated and not the path of states followed by the process.

A first order hidden Markov model is defined formally as a tuple $M = (Q, \Sigma, \pi, a, e)$ where:

- $Q = \{1, ..., n\}$ is a finite set of states;
- π is vector of size n defining the starting probability distribution, that is $\pi[i]$ is the probability of starting the process at state i;
- $\Sigma = \sigma_1, \sigma_2, \ldots, \sigma_m$ is a finite alphabet, i.e. the set of output symbols;
- a is an $n \times n$ matrix of transition probabilities, namely $a(i, j)$ is the probability of moving from state i to state j.
- e is an $n \times m$ matrix of emission probabilities, namely $e(i, j)$ is the probability of generating symbol σ_j in state i.

HMM

Teresa Przytycka

February 4, 2004

Undefined concepts used: sequence homology, amino-acid substitution, conserved amino-acids, Markov Process, globular protein, hydrophobicity, fold recognition, family profile, membrane proteins, exons, introns, promoters, neural networks, gradient descent, simulated annealing.

Contents