Chapter 1

Hidden Markov models

Abstract

This chapter will take you through all of the steps of HMMs and make you aware of all of the important components which you should be aware of when using them, but not go into the dynamic programming and algorithms necessary to write your own software. For this recommend the book. Strong emphasis on their use for remote protein homology detection

1.1 Introduction

Hidden Markov models (known as HMMs) can be applied as a probabilistic approach to recognition problems. Their formulation is such that they are particularly well suited to the recognition of strings of indeterminate length, which are often problematic for other approaches. Nucleotide sequences of DNA and amino acid sequences of proteins fit this description. During evolution they are subject to random alterations which change their length by inserting and deleting sequence. Recognition has been a central problem for understanding and using biological sequences for some time, but the advent of the genome sequencing projects has made it ever more important. HMMs have become one of the most powerful and popular tools for genome analysis.

1.1.1 Applications

HMMs were first widely used in speech recognition, where the words are considered as strings of audio signal, the length of which is dependent on accent and enunciation. They are now applied to a wide range of biological recognition problems. Their strength lies in the recognition of distant relationships, analogous to the detection of a very weak signal. As a consequence they may succeed where other methods fail to detect anything, but not be as well suited in situations of high similarity.

In this chapter the use of HMMs for remote protein homology detection will be described in detail, although it should be noted that there are other uses of HMMs for biological sequence analysis which are also important. The most successful trans-membrane helix predictors use HMMs, and they have also been used for the detection of CpG islands, regulatory regions, and for gene prediction. Despite the emphasis, information is of general relevance and in the resources section at the end of the chapter the other major HMM applications are represented.

1.1.2 Performance

During evolution protein sequences are duplicated and subsequently altered. In this way families of proteins evolve from a common ancestor into groups of related proteins which usually have similar functions. HMMs can be used for homology detection of these family members, and hence their ancestry and approximate function can be determined from their amino acid sequence alone. This is essential in for the analysis of the large number of genome sequences, which would take decades to carry out experimentally on a similar scale. Nucleotide sequences can also be used but amino acid sequences are used in all examples and descriptions in this chapter.

HMMs and some other methods, for example PSI-BLAST which is essentially very similar, are able to detect relationships between a sequence and a sequence family. By using the information contained in a family of sequences, HMMs are able to detect more remote homology than methods which merely compare a sequence to another single sequence. Pair-wise methods such as BLAST compare individual sequences and although they are less successful in detecting remote homology, they are computationally less expensive and therefore are still often useful. There are also more expensive methods for homology detection, often involving the use of three-dimensional structure information (e.g. threading or ab initio) which may perform better. However these methods do not perform much better and are so much more computationally expensive that they may not be applicable on a genomic scale, whereas HMMs are far superior to pair-wise methods and are easily applicable on a genomic scale.

A profile of a family of sequences can be constructed in the from of an HMM. This profile is constructed from a multiple sequence alignment of the members of the family and should characterise the important common features of the family. This profile is then compared to unknown sequences to determine whether or not they fit the profile and are members of the family which the model represents.

The SCOP all against all test

The primary sequence of members of the same family often have diverged to the point where they have any amino acids in common, but they always retain the same three-dimensional structure arrangement in the deeply buried hydrophobic core. This allows family membership to be unambiguously identified from the

protein's structure. The SCOP database classifies all proteins of known structure into their constituent families. It is possible to assess performance of a method by using it to compare all of the amino acid sequences of the structures, then using the structure-based classification to mark the detected relationships as true or false. Obviously the object of any method is to get as many true positives and as few false positives as possible. Figure 1 shows that in the region of a 1% error rate the SAM-T99 hidden Markov model procedure performs better than the PSI-BLAST profile method, and that both profile methods perform far better than BLAST which is a pair-wise method. The other methods mentioned earlier are too computationally expensive to include in this assessment.

1.2 What is a hidden Markov model?

A hidden Markov model is a Markov chain with hidden states providing a general probabilistic model for sequences.

Why "hidden"?

A Markov chain is a collection of states linked by transitions between the states. In a simple Markov chain the different states in the chain represent the possible outcomes that a unit of the sequence can have. For example a general Markov chain modeling DNA would have four states, one for each nucleotide; it would have transitions from each state to every other state, and from every state to itself. Any sequence can be traced through the model by passing from one state to the next via the transitions.

In a hidden Markov model the states are less simple, and instead of having a single known outcome they can have several possible outcomes. The model is called "hidden" because the outcome of any given state is uncertain. For example a general hidden Markov model modeling DNA would have one state for each position in the sequence, each state having four possible outcomes; it would have transitions from each position to the next. Any sequence can be traced through the model, this time by passing from one state to the next in a line via the transitions.

1.2.1 A sequence emitter

The above trivial examples of a Markov chain and a hidden Markov model show how any sequence can be traced out, but it is not clear yet how this is useful. To understand how HMMs are used they must first be thought of as a sequence generator, it will then be explained how a sequence generator is useful. Let us call the states described so far 'emission' states and define them as emitting a single nucleotide, the hidden states are capable of emitting any single nucleotide. The chain or the model can emit any sequence of nucleotides, infinite in the case of the chain, and the finite length of the model in that case. In the case of the Markov chain there are probabilities associated with each transition, and

if these are not equal, then the chain is more likely to emit sequences with a certain composition and arrangement than others. The hidden Markov model has probabilities in each state associated with the different possible outcomes, and if these are not equal, the model will be more likely to emit sequences with certain outcomes in certain positions. In this way models can be profiles representing a family of sequences with the common features characterised in the probabilities.

All of the probabilities are independent, therefore any emission is not governed by previous emissions. Higher order Markov models are possible in theory but are computationally intractable for biological problems.

A model may seem to be of limited use as a sequence generator, but it can be used to answer the following question "given a known group of sequences, is another given sequence a member of that group or not?". Firstly a model is constructed to represent the group by choosing the probabilities such that the model is that which is most likely to have generated the given group of sequences. Then it is possible to calculate the probability that the model would generate the given sequence, thus giving the probability that the sequence is a member of the initial group.

1.2.2 States and transitions

In the introduction to this chapter an emphasis was put on the fact that the formulation of the hidden Markov model lends itself very well to sequences of indeterminate length. So far only emission states have been discussed, but now let us introduce some other states into the hidden Markov model which will bring it closer to sequences observed in nature. Firstly there are two trivial states in the model at the beginning and the end, called the 'start' and 'stop' states. These are obviously the points at which a model starts and stops generating the sequence.

Insertions

The emission states described so far of which there are one for each position are all called 'match' states, but there are other types of emission states called 'insert' states. These may appear between two match states with transitions (and their associated probabilities) from the preceding match state, and to the following match state. Like the match states they are also emit, and have associated probabilities for each possible outcome. In this way if the transition from the preceding match state is taken to the insert state instead of the transition to the following match state, it will emit an extra outcome in between the two match states. Adding an extra transition from the insert state to itself allows the insertion of any number of extra outcomes. The probabilities of the transitions govern the likelihood of an insertion taking place, and the length of the insertion.

Deletions

There are also states which are not emission states called 'delete' states, which do not emit, and which merely pass on to other states via their transitions. Putting a delete state in parallel with a match state, and transitions to and from match states before and after, allow the match state to be skipped by passing through the delete state instead. Delete states adjacent to each other have transitions between each other allowing the skipping of more than one match state in a row. The transition probabilities govern the likelihood and length of a deletion.

1.2.3 Architecture

Figure 2 shows the typical arrangement of states and transitions in a hidden Markov model used for sequence homology in biology. From now on let us consider models representing protein families, which have 20 probabilities in each match and insert state, one for each amino acid. The match states will generally have very different emission probabilities from each other depending on the local environment of the peptide at the corresponding position in the protein. Insert states will generally have similar generic emission probabilities as they affect only the amino acid composition of the length of inserted sequence. The transition probabilities from match to insert are analogous to the 'gap open' penalties in pair-wise methods such as BLAST, and the transitions from insert states to themselves are analogous to gap 'extension'penalties.

1.3 Model generation

1.3.1 Alignments

To generate an HMM representing a protein family a multiple sequence alignment of that family is used. The single most important factor in determining the quality of an HMM is the alignment from which it is built. There are two factors: the number and type of sequences in the alignment, and how well the sequences are aligned. Models are improved by having a larger number of sequences in the alignment which must all be true homologues to each other. As a rough guide, a good model can usually be constructed from an alignment of 60 sequences. The more diverse the sequences are the better; a large number of very similar sequences contains less information than fewer very dissimilar ones. Obviously the better the alignment of the sequence positions into columns, the better the model which is built from that alignment. Ways of generating alignments for model building are described in section 1.5.

In the most simple case a match state is created for each column of an alignment. In the match state the probabilities for each amino acid are proportional to the observed frequencies in the corresponding column of the alignment. Obviously the probabilities of all amino acids must add up to one. However usually

alignments will have some deletions and insertions, which are better handled by making use of the other states as well.

1.3.2 Transition probabilities

Often when an alignment is generated not all regions are aligned, but there are regions along the sequence which were not possible to align. For example a group of proteins will conserve the same secondary structure elements, but may not have the same structure in their loops, and furthermore the loops may vary a great deal in length from one protein to the next. In this case the regions of secondary structure will be aligned, but some loop regions are unaligned.

Implicit state assignment

If the alignment specifies which parts are aligned and which are not, then the columns which are specified as aligned are used to build match states. Each unaligned region of the alignment is used to build a single insert state between the last match state before the unaligned region begins, and the first match state after the region ends. The more sequences there are in the alignment which have residues in the unaligned area, the higher the transition probability from the match to that insert state. The longer the unaligned region is, the higher the transition probability will be from insert to itself. Just as the emission probabilities from any state must add up to one, all of the transition probabilities leading from a particular state must also add up to one. Similarly match to delete transition probabilities are affected by the observed deletions. In an aligned region, the transition probability from the previous match state to a delete state as opposed to the match state, is affected by the corresponding column in the alignment. The transition to the delete state will be higher the more sequences there are in the alignment with no residue aligned in that column.

Background rates

There may be many columns in the alignment which have a residue aligned in every sequence, and many continuous aligned regions with no unaligned regions between aligned columns. Using this information only in the way described above would lead to some zero transition probabilities to delete and insert states. It is important when building a model to not only use the observed information from the alignment, but also to use prior information as is discussed in more detail in 1.3.3. What is meant by this is that we have prior information on protein sequences in general before seeing the observed information in the alignment. We know for example that an insertion or a deletion is possible anywhere in a new sequence, even if we have not yet already observed one in a sequence (in the alignment) at that position. For this reason it is usually undesirable to have any transitions with zero probability. Using knowledge of insertions and deletions gathered from large numbers of sequences, the background rates

can be calculated. These background rates are combined with the observations from the alignment, the ratio between them depending on the significance of the observations (number of sequences in the alignment). In this way there are never any transitions with zero probability.

Automatic state assignment

An alignment may not have the aligned and unaligned regions specified. This may be because the method used to generate the alignments forces the alignment of everything regardless of similarity, or simply that the information is discarded. In this situation it is not clear how to build the match and insert states in the model. The simplest solution to this problem is to declare all columns which have an aligned residue in more than half (or some other proportion) of the sequences as aligned, and these become match states. Those remaining are represented by insert states.

A more sophisticated method is called MAP match-insert assignment. This is a dynamic programming algorithm which essentially amounts to generating the combination of match and insert states which are most likely to have produced the observed alignment. This is very good, and in theory could be applied to alignments which have the aligned and unaligned regions specified, by discarding the information and using MAP to choose the match and insert states. This may work in some situations with untrusted alignments, but for good quality alignments better models are built by using the alignment implicitly.

1.3.3 Emission probabilities

In the same way that zero probabilities of transitions are undesirable, zero probabilities in a match or insert state for the emission of a given amino acid is undesirable. If the model is going to be able to match new distant homologous members of the family, it will have to allow for the possibility of amino acids in positions where none is observed in the alignment.

Pseudo-counts

The most straight forward way to avoid zero emission probabilities is to add pseudo-counts to the observed counts of amino acids in each column of the alignment. By adding a single count to each of the 20 amino acids observed in a column of the alignment, there is a non-zero probability of every amino acid. Furthermore the greater the number of sequences in the alignment the less the contribution from the pseudo-counts will be. Using single pseudo-counts is called 'Laplace's rule'.

In reality all amino acids are not equal. Some are more common than others and therefore more likely to be substituted into a position in the alignment. So better than single pseudo-counts is to use the frequency of amino acids in nature to add fractional pseudo-counts proportional to the background distribution. Extending this still further it is possible to not only use our prior knowledge

of the background distribution, but of the similarity between amino acids. The likelihood of certain amino acids replacing others depends on their properties, such as hydrophobicity, charge, and size. There are substitution matrices such as the BLOSSUM and PAM matrices which can be used to give a probability of any amino acid being substituted by any other. These substitution matrices are normally used by pair-wise comparison methods for scoring the observed substitutions between two proteins. A substitution matrix can be used to generate pseudo-counts whose distribution is dependent on the observed amino acids in the alignment. The proportions used for pseudo-counts can be generated by taking the distribution of substitution probabilities from the matrix, for each amino acid observed in the alignment, and averaging them.

Prior libraries

The most successful method of using prior information is to use Dirichlet mixtures. Rather than using the overall background distribution of amino acids, it is possible to use a mixture of different distributions drawn from different local environments in proteins. For example at a position in a protein where there is currently a small hydrophobic (buried) residue, it is unlikely that a large hydrophilic residue will be substituted. A prior library is a collection of Dirichlet distributions derived from a large number of observed amino acids in different natural local environments. These distributions represent our prior knowledge of what substitutions are expected to be likely at different positions. For each position in the alignment a mixture of these Dirichlet distributions taken from the library is used. The proportions with which they are mixed is derived by estimating the likelihood for each prior (Dirichlet mixture) that it could have generated that column in the alignment. A library of 20 automatically generated priors has been shown to perform well.

1.3.4 Sequence weighting

It was mentioned earlier that ideally the alignment should contain many sequences, and that they should be as diverse as possible whilst remaining related. Due to the nature of evolution via duplication, there may be some very closely related sequences and some very distantly related ones. When building a model from the alignment this is taken into consideration. Until now all sequences were described as being weighted equally, but models can be improved by weighting them. If every sequence had equal contribution to the probabilities in the model, then it would be biased toward closely related sequences.

Let us imagine an alignment of a family of related sequences, where divergence and duplication have taken place. In this family there are several divergent subgroups, each consisting of more closely related proteins. Let us say that one of these subgroups is larger than all of the others put together, which is a common situation. If all sequences are given equal weight then the model will represent the characteristics of the large subgroup too strongly. As a consequence the HMM will not model the true divergence of the family. What is

wanted is something like equal weighting of subgroups without losing the lesser, but still useful information contained in the differences between closely related members of subgroups.

There are various ways of weighting the sequences. Many are based on tree representations of the sequences, taking the contribution to the probabilities in the model from the branch lengths. Other weighting schemes are based on similarity distances between sequences, or by using the affect of the weights on the model to maximise its ability to discriminate the sequences from a random model. However the scheme used by most methods is to maximise the sum of the entropy in the model. This amounts to making the statistical spread of the model as broad as possible, hence making it as general as possible. More general models will match more distant homologues.

1.4 Sequence scoring

The most common use of HMMs is for remote homology detection. Usually an HMM is first built to represent a protein family and then searched against a database of sequences to discover new members of the family. Each sequence in the database is compared to the model and assigned a score; the ability of these scores to separate true from false homologues, and to accurately predict the level of certainty determines the success.

1.4.1 Algorithms

There are two common algorithms for sequence scoring: Viterbi, and EM. Both of these can be solved using dynamic programming which is very efficient and makes the large calculation computationally practical. The Viterbi algorithm is faster than the EM algorithm, but the EM algorithm performs slightly better.

Viterbi

The Viterbi algorithm is also known as the 'best path' algorithm. For a given model and a given sequence, the Viterbi algorithm calculates the most probable path through the states and transitions of the model. This path gives the probability of the model generating the sequence, and hence a score which is related to the likelihood of the sequence in question being a member of the family which the model represents.

Expectation maximisation

The expectation minimisation (EM) algorithm is also known as the 'all paths' algorithm. It is possible for a model to generate the same sequence via different paths through the states and transitions of the model. The EM algorithm calculates the sum of the probabilities across all paths rather than just the most probable path.

1.4.2 Local or global

Scoring a model against a sequence can be done in several ways depending on what is known about the model and the sequence.

Global

The simplest form is global scoring which directly compares the whole model to the whole sequence, and should be used when it is known that the model and the sequence are comparable (and hence have a similar length). Global scoring starts at the begin state, follows a path via transitions to the first emission state which emits the first residue of the sequence being scored. It continues through the model generating the entire sequence until it follows a path via transitions from the emission of the last residue in the sequence to the end state.

Global/local

Many proteins, especially in eukaryotes, are multi-domain proteins. Domains are evolutionary units (building blocks) of proteins which are duplicated and recombined to form many multi-domain proteins with different combinations of these domain sub-units. Multi-domain proteins vary both in the number and type of domains from which they are built. As a consequence sequence searches will not always be comparing like proteins. For example if there is a family of single domain proteins for which an HMM is built, searching this model against all of the sequences in a genome using global scoring is inappropriate. If this domain also appears as a sub-unit of a multi-domain protein, then the model will assign a bad score to the whole protein using global scoring, despite containing a homologous domain. This is a very common situation in genome analysis. For example the human genome contains tens of thousands of proteins, over half of which are multi-domain, and they are built from combinations of domains belonging to only hundreds of families.

Global/local (model/sequence) scoring allows the first emission from the model to begin at any point in the sequence, and for the last emission to end at any point. In this way the model matches the best-scoring region of the sequence. This should also allow for more than one match to the sequence; many multi-domain proteins contain repeats of the same domain. Usually after the model matches a region of the sequence that region is eliminated and the next best-scoring region is searched for, allowing multiple matches of a single model to different regions of a single sequence.

Local

Fully local scoring finds the best-scoring match of any region of a model to any region of the sequence. Local model scoring can be achieved in two ways. One way is to built a model with a local-scoring architecture. By adding transitions from every match state to the end state and from the begin state to every match state, it allows the path through the model to jump in and out at any point.

The other way is to calculate the scores on sub-regions of the model as part of the sequence searching procedure. Both approaches essentially amount to the same thing.

Fully local scoring not only has the advantage that no prior knowledge of the domain composition of the model or the sequence is required, but also improves remote homology detection. Parts (e.g. loops) of very distantly related proteins may have evolved beyond all recognition whilst the core of the protein is retained. Local scoring allows the best score for those parts of the protein which have been retained, without penalising the overall score by forcing the variable regions to match to each other when they may no longer have anything in common. The longest possible match will still however always be sought, because longer matches are less likely to occur by chance and therefore are higher-scoring.

Local scoring should be used for most applications. Other scoring should only be used if the model and sequence are known to be comparable in some way, in these cases the use of this prior knowledge to select the type of scoring may improve results.

Local/Global

Local/global (model/sequence) scoring is the same as fully local scoring except it forces the first emission to be the first residue of the sequence, and the last emission to be the last of the sequence. This is only useful for scoring fragments of sequence.

1.4.3 Null models

Some sequence patterns and compositions are more common in nature than others. Models built to represent families which have common types of sequences will be more likely to match other sequences than models built from unusual families. A very common problem with sequence comparison methods is that false matches with high scores are given to low complexity sequences and sequences with repeats. For example a protein family which has amino acid distributions which are close to the background, might score highly an unrelated sequence which only has common residues, because the chance of being able to find suitable paths through the model emitting the correct sequence will be higher.

To overcome this problem a null model is introduced. The null model represents the alternative to the given family profile model, and the scores for that model are offset against the null model. This means that a model which is close to the null requires a stronger match to get a high score from the comparison than a model which is very far from the null.

Fixed null

The most simple form of the null model is the equivalent of the family profile model with the probabilities replaced by those from the background distribution. This null model can be thought of as the broadest possible model representing all natural protein sequence.

Geometric mean

Using the fixed null model will improve the discrimination of the family model against common sequences, but there may still be cases where unrelated sequences are able to match a model because they have a similar sequence composition to the model, even if that sequence composition is not necessarily common. This will happen far less frequently than with common sequences, but is still a possibility. To combat this problem a null model can be defined from the family model. Rather than taking the probabilities from the background distribution, the null model probabilities are calculated as the geometric mean of the probabilities across all states of the family model. This way the null model has the same sequence composition as the family model.

Reverse null

There is still a possibility that a model might have a pattern of important characteristic residues, small in number and thus not affecting the composition much, yet likely to occur in other proteins. For example proteins with disulphide bonds between cysteine residues, could have those cysteines so highly conserved that it becomes the most important characteristic of the profile (family model). There are likely to be some other unrelated proteins which also have some disulphide bonds between cysteines, and which can be matched to the relevant states in the model. The reverse null model is the model in reverse. The symmetry means that the score of the sequence to the reverse null model is the same as the score of the reverse sequence to the ordinary family model. The resulting score is calculated simply by scoring the sequence against the model and then subtracting the score that the same sequence in reverse gets to the model. In the example of the proteins with highly conserved cysteines, the cysteines will match in both directions, and so the difference between the forward and reverse scores will depend on the rest of the protein.

Using the reverse null model has the additional advantage of generating accurate E-values (see 1.4.4) directly, whereas using the other null models means that E-values must be calculated from an ad hoc estimation of the distribution of scores (see 1.4.4). The reverse null has the disadvantage that the sequence must be scored in both directions requiring double the computation.

1.4.4 E-values

Expectation values (E-values) are a general value not specific to HMMs and are supplied by many sequence comparison methods. In theory E-values from any

model, and even any method should be comparable.

What E-values mean

Since HMMs are a probabilistic method, statistically one expects occasional chance errors. E-values are the expectation of the number of errors per query. If an HMM is used to score a database of sequences (a query), then the expected number of sequences unrelated to the model (errors), which have the same score or better than any sequence, is given by its E-value. It is important to note that an E-value is only true in the context of the query, for example a query on a large database will have more errors than a query on a small database searched with the same model. E-values can be used to decide what error rate you are prepared to allow in your results, but they say nothing about the success rate.

Calibration

The errors produced by HMM scores follow an extreme value distribution. One way of estimating E-values is to score the model against a large number of random sequences (assumed unrelated to the model, and therefore all errors) and plot the distribution. An extreme value distribution can be fitted to the results and used to estimate the E-values of scores produced by the model. The process of scoring random sequences and fitting the distribution is often called model calibration. This method has the disadvantage that many random sequences must be scored to get an accurate estimation of the distribution, and this is computationally expensive.

Reverse null E-values

Another approach is to score the reverse of the sequence as well as the sequence itself; this is the reverse null model mentioned earlier. Since the reverse sequence has the same error distribution as the forward sequence, subtracting the reverse score from the forward score cancels out the component of 'error' from the distribution. Both forward and reverse scores are expected to be drawn from the same Gumbel distribution which has two parameters, however the difference of two variables drawn from the same Gumbel distribution follows a simpler one-parameter distribution. It appears to be the case empirically that the parameter is the same for all models and equal to one, so the distribution of the differences in forward and reverse scores is known. The E-value can then be directly estimated from the query conditions (database size), without the need for ad hoc model calibration requiring the scoring of random sequences and the fitting of the distribution.

1.4.5 Hardware acceleration

Some companies have developed hardware acceleration for HMMs. There are chips called 'field programmable gate arrays' which are configurable to use dynamic programming algorithms and boards are manufactured with many of

these chips. These chips are also able to run the inner loop of complex algorithms in one clock cycle, whereas conventional processors require tens of thousands of cycles. Since most HMM search problems are naturally massively parallel, the parallelisation is relatively easy and many of these boards can be implemented together with some post-processing software, effectively giving thousands of processors in a single machine. Although the technology is very expensive, it is still much cheaper than the equivalent power in conventional hardware. These accelerators open a realm of investigative possibilities on a large genomic scale which was previously only possible for pair-wise methods and is unlikely to become possible for most other sequence comparison methods.

1.5 Multiple sequence alignments

1.5.1 From models

Multiple sequence alignments are required for model building, but HMMs can also be used to produce multiple sequence alignments. Good quality multiple sequence alignments are important because they give information beyond mere homology. For example three-dimensional models of proteins can be constructed from sequence alignments if one of the sequences has a known structure. If a sequence is found to be homologous to another for which the three-dimensional structure has been experimentally determined, the alignment can be used to replace the side-chains of the corresponding amino acids on the backbone of the known template structure. Multiple sequence alignments are particularly useful because the patterns of conservation indicate the evolutionary constraints which in turn suggest the functional and structural constraints on the proteins in the aligned family.

The multiple sequence alignments produced by HMMs are not true multiple sequence alignments in the way that those produced by ClustalW are. ClustalW compares every sequence to every other sequence, whereas HMM aligning compares every sequence to the model independently so that the alignment between sequences is by proxy. Adding new sequences to a ClustalW alignment will add new information which may alter the alignment of other sequences; adding new sequences to an HMM alignment never changes the alignment of any sequences relative to each other.

An alignment between a sequence and an HMM can be described by the path through the states of the model. Every residue in the sequence which is emitted by a match state will be aligned to that position in the model, match states which are omitted via a delete state will be empty columns of the alignment, and residues emitted by insert states will merely be padded out between the columns corresponding to match states. Subsequent sequences are aligned in the same way, so that there is a column in the alignment for each match state filled by the residues of the sequences which have been emitted by that state. Figure 3 shows a multiple sequence alignment generated by an HMM. The numbering across the top corresponds to the match states of the model, the columns of which

contain upper case residues (aligned) or dashes where the path has passed via a delete state emitting nothing. The lower case residues are generated by insert states (unaligned), and these parts of the alignment are just padded out between the numbered columns filling in the other sequences with dots to maintain the alignment as a whole.

Algorithms

To create alignments to HMMs the path through the model is required. This can be calculated using the Viterbi algorithm for calculating the best path described in 1.4.1 . The EM algorithm calculates the sum across paths, which does not give a single path which is required for an alignment. The EM algorithm can however be used to calculate the most likely path, which is much slower than calculating the best path but slightly better. The EM algorithm is first used to evaluate all possible paths, calculating the probability of each residue appearing in each model state and the probability of each transition being used. A second (Viterbi-style) dynamic programming is then performed on these probabilities to find the most likely path (see, for example, Holmes and Durbin in Recomb98).

1.5.2 For models

As mentioned earlier the single most important factor in model building is the multiple sequence alignment. There are many sources of alignments so only some of the more important ones are discussed here.

Structural alignments

The most reliable high quality sequence alignments are the result of a detailed analysis of proteins of known structure. By superimposing three-dimensional structures on each other a very accurate correspondence between positions in the sequences can be obtained. Furthermore it is possible to make alignments of sets of extremely divergent proteins because although their sequences may have diverged beyond the point of having any residues in common, their core structure is maintained. The ability to recognise more distant homologues using structural comparison methods rather than purely sequence-based methods enables the inclusion of more divergent members. It is usually necessary to include sequences of unknown structure in the structure-based alignment to increase numbers. This can be done by using automatic sequence search methods (such as BLAST) with very certain score thresholds to ensure true homology, which can be added to the alignment using the close similarity in combination with knowledge of key residues and structural context obtained from the structural analysis of the family.

The disadvantages of using hand curated structural alignments are that they are very laborious to produce, and require experimentally determined three-dimensional structures which are not always available, especially in sufficient

quantities. There are automatic procedures for structural alignment which eliminate the need for doing it by hand, but these are less accurate and reliable, and will sometimes fail altogether in the case of very divergent proteins.

Automatic methods

There may already be sequence alignments available from databases or literature, but in the absence of reliable and up to date data they must be generated from scratch. There are two steps in automatic sequence alignment generation: finding the diverse set of related family members, and creating the multiple alignment. Usually homologues are sought by searching the sequences against a large non-redundant database (e.g. NRDB90), which consists of all the major sequence resources concatenated and filtered to eliminate highly similar (redundant) sequences. There are many freely available programs and software packages for both of these tasks and it is not the purpose of this chapter to list and review them all. It should be said however that it is important that whichever method is used, the sequence homologues should be selected with a high degree of certainty, otherwise possible errors will be propagated to the model. Another pit-fall to be avoided is the inclusion of low-complexity or repeat sequences, which can be masked using a program such as 'seg'. Due to the short-comings of fully automatic procedures, combining automatic procedures with hand curation is common practice, and most HMM-based databases adopt this to a greater or lesser degree.

SAM-T98

There is more than one software package for using HMMs for remote homology detection, but there is only one which does not require alignments for model-building, because it creates its own using an iterative procedure similar to PSI-BLAST. The SAM package comes with a procedure called 'T98' which was later improved and renamed 'T99'. This procedure creates multiple alignments of homologous sequences for the purpose of model building. The procedure begins with a seed, which can be a single sequence or a set of sequences, aligned or not. The procedure then uses a large non-redundant database of sequences from which to extract homologues by an iterative process to add to a growing alignment.

The default T98 procedure follows these steps:

1. Using the initial sequence(s) and the WU-BLASTP procedure (http://blast.wustl.edu/blast), a search of a large non-redundant protein database is carried out to find two sets of sequences. The first set consists of close homologues of the query sequence: those that match it with E-values of 0.00003 or less, and these are used to create the initial alignment (step 2). The second set of sequences consists of those that match the query with E-values of 500 or less and, therefore, will probably include more distant homologues of the query sequence.

2. Sequences from the non-redundant database in the set of close homologues are aligned with each other and with the seed sequence.

3. An initial HMM is built from the alignment of sequences created in step 2.

4. This model is then used to search the second set from step 1 for more homologues, which are added to the first set and re-aligned to create a new and larger alignment from which a new and better model can be built. The initial HMM is extended with additional homologues by repeating steps 2,3, and 4 for four iterations. In step 1, the use of low E-values by WU-BLAST and a strict HMM scoring threshold, ensures that only close homologues are used. In the four iterations, thresholds for the HMM score are gradually decreased making the model gradually more general.

5. From the final alignment produced by the iterations of steps 2,3, and 4, a model is built using one of the scripts provided by the SAM package. These scripts filter and weight the sequences in the alignment before building the model.

As well as being fully automatic, this iterative process produces very general models which are excellent for remote homology detection.

1.6 Resources

These resources are all computer-based utilities and databases available either for download or access via the internet. This section is not a review of all that which is available, but indicates important resources for the most common uses, and provides a starting point for those wishing to use HMM technology in practical applications.

1.6.1 Software

These are software packages which it is possible to download and install so that they can be run locally on private computers.

Remote protein homology detection and alignment

These are the software packages which carry out the functions which have been the focus of this chapter, and which have implemented the features which have been described.

The Sequence Alignment and Modeling system (SAM) is available under license which is awarded free of charge for academic use. This is the most complete software package for HMMs in this area. The SAM package (http://www.cse.ucsc.edu/research/compbio/sam.html) contains programs for model building from multiple sequence alignments, sequence searching (model scoring), and sequence alignment. Most importantly

the SAM package contains the T99 software which can build a model from a single seed sequence, used to iteratively search a large database for homologues, build a multiple alignment from the homologues, and from that a model representing the family of sequences to which the original seed belongs. SAM also has several helper scripts and an abundance of features and programs for expert users. SAM uses EM scoring with the reverse null model by default, and takes the architecture from the alignment when model building. The SAM package is not obviously the most popular due to the licensing and a past reputation for being less easy to install and use than HMMER.

The HMMER package (http://hmmer.wustl.edu) is possibly the most popular, and has open source code under the GNU public license. The HMMER package is similar to SAM, and contains the same basic components with the exception of the T99 iterative procedure. Independant multiple sequence alignments are required to build models. HMMER uses Viterbi scoring by default, and estimates the model architecture from the alignment for model building. HMMER models need to be calibrated for the generation of E-value scores.

Comparing SAM and HMMER default procedures for model building and scoring, shows that on average SAM performs slightly better, but that HMMER is faster at scoring although much slower at building models if calibration time is included.

SAM and HMMER are both excellent popular packages, but it is worth also mentioning HMMpro (http://www.netid.com) which is commercially produced and META-MEME (http://metameme.sdsc.edu) which is motif-based.

Gene finding

The Wise2 software package (http://) is used extensively on the human genome and others. It is focused on comparing DNA sequences at the level of its conceptual translation, regardless of error and introns. The GeneWise part of the package makes use of the Dynamite code generation software for HMMs and compares an HMM profile to DNA sequence, modeling the occurance of introns. GeneMark (http://opal.biology.gatech.edu/GeneMark) is another HMM based software package for gene prediction worth mentioning.

Trans-membrane helix prediction

Trans-membrane helices can be predicted with relatively high accuracy, the authors claim well above 90%. TMHMM (http://www.cbs.dtu.dk/krogh/TMHMM) is one of the best performing methods and uses HMMs. The model is cyclic and has seven states: helix core, helix caps on either side, loop on the cytoplasmic side, two loops for the non-cytoplasmic side, and a globular domain state in the middle of each loop.

HMM Topology of Proteins (HMMTOP at http://www.enzim.hu/hmmtop) is another hidden Markov model based trans-membrane predictor.

General HMM tools

There are some general mathematical tools available to carry out some of the basic calculations which have been discussed, most of which are directed at speech recognition applications, but may still be useful to those wishing to get involved in programming but do not wish to write their own code to implement the algorithms from scratch. The GNU Hidden Markov Model programming library (http://www.zaik.uni-koeln.de/~hmm/ghmm) provides most of the components, and is freely and publicly available.

There is Matlab software package called HMM toolbox (http://www.cs.berkeley.edu/~murphyk/Bayes/hmm.html) with links to other Matlab HMM resources. This software is also distributed under the GNU public license, so is freely and publicly available.

1.6.2 Databases

There are a large number of databases which can be accessed via the internet, and a more complete listing can be obtained from the Nucleic Acids Research database issue which comes out in January of every year. It is worth mentioning however some of the most popular databases which all use libraries of hidden Markov models.

SUPERFAMILY

The SUPERFAMILY database (http://supfam.org) is a library of hidden Markov models representing all proteins of known structure. It is based on the SCOP (http://scop.mrc-lmb.cam.ac.uk/scop) domain classification of proteins at the superfamily level. The SUPERFAMILY database primarily uses the SAM software but provides models in both SAM and HMMER formats. The library of models has been searched against all completely sequenced genomes and the assignments are available on the web site, as well as a sequence search facility and a service providing multiple alignments. SUPERFAMILY is available freely for academic use.

PFAM

The PFAM protein families database (http://www.sanger.ac.uk/Software/Pfam) is a highly curated database of sequence families. The database provides annotation of the families as well as multiple alignments and hidden Markov models built from them. The information in this database is very accurate and has a good coverage. PFAM uses the HMMER package, and is freely available.

SMART

The Simple Modular Architecture Research Tool (SMART at http://smart.embl-heidelberg.de) is similar to PFAM except that it covers only domain families found in signalling, extracellular, and chromatin-associated proteins. SMART

also uses the HMMER software. There is a web server for searches but the models are only available via the InterPro consortium.

TIGRFAMs

The TIGRFAMs (http://www.tigr.org/TIGRFAMs) database is another curated sequence-based HMM library based on the clustering of sequences from microbial genome projects. The families are classified by their functional similarity. This database uses the HMMER software and the models are available under the GNU public license.