ISMB-95
ROBINSON COLLEGE, CAMBRIDGE

Tutorial Programme
Sunday 15 July 1995

TUTORIAL T4

Hidden Markov Models and Other Machine Learning Approaches in Computational Molecular Biology

(Pierre Baldi)

HIDDEN MARKOV MODELS AND OTHER MACHINE LEARNING APPROACHES IN COMPUTATIONAL MOLECULAR BIOLOGY

Pierre Baldi

Caltech

- A. NOTES

1. Introduction
2. Hidden Markov Models
3. Learning Algorithms
4. Protein Apllications
5. DNA Applications
6. Limitations and Optimality of HMMs
7. Hybrid HMM/NN Architectures
8. Conclusion
9. References

- B. SLIDES

1. Introduction
2. Statistical Modeling and Induction
3. Neural Networks
4. Hidden Markov Models
   4a. Protein Applications
   4b. DNA Applications
5. Stochastic Grammars

- C. REPRINT

A. NOTES

1 INTRODUCTION

A wealth of protein and DNA primary sequences is being generated by genome and other sequencing projects. Computational tools are increasingly needed to process this massive amount of data, to organise and classify sequences, to detect weak similarities, to separate coding from non-coding regions, and to reconstruct the underlying evolutionary history (see, for instance, Jurka and Milosavljevic (1991), Karlin and Brendel (1992), Lawrence et al. (1993), von Heijne (1987), Waterman (1986) and Thorne et al. (1991) for references and background). Sequence analysis sheds light on structure and function and greatly contributes to our understanding of biological systems. Computational problems in molecular biology, such as multiple alignments, classification, data base searches, and phylogenetic reconstruction will be briefly reviewed in the tutorial (see slides).

Although very useful, conventional computer science algorithms have not been able to solve all these problems, partly because of a lack of a general theory and because of our poor understanding of evolutionary tinkering. As pointed out in Rumelhart et al. (1995), machine learning approaches (e.g. Neural Networks, Hidden Markov Models, Belief Networks) on the other hand, are ideally suited for domains characterised by the presence of large amounts of data and a lack of a comprehensive underlying theory. The fundamental idea behind these approaches is to *learn the theory from the data*, through a process of model fitting. These algorithms indeed consist in first selecting a generic model for the data, with a large number of parameters, and then adjusting the parameters iteratively so as to optimise a certain measure of how well the model fits the data. Before we can proceed with molecular biology applications, however, we must first examine what is the proper general framework for machine learning approaches.

The fundamental problem in machine learning is the same as in scientific reasoning in general, as well as statistical modeling : to come up with a good model for the data. That is to *infer* a good model, or hypothesis, from the available data. The difference is just in emphasis: in machine learning the search for a good model should be somewhat automatic, within a large class of possible representations. There exists a logically consistent, almost universally accepted, framework for *deduction*, pervasively used in all our digital computers, namely Boole's algebra. What is perhaps less well known and more surprising is that there is an equally logically consistent and compelling framework for *inference*, known as the Bayesian framework (Cox (1964), Jaynes (1986)). Since the fundamental problem in inference, scientific reasoning, or machine learning is to infer a model or a hypothesis from the data, it is essential that one be able to compare different models and to rank them. That is, in science, one ought to be able to decide whether to prefer model (or hypothesis) M_1 or M_2 given a certain amount of background information D. Furthermore, such preference should be transitive. Given D, if M_1 is better than M_2, and M_2 is better than M_3, then M_1 is a better model than M_3. It is then easy to see that preferences should be representable by a real number. Furthermore, it is then possible to write a very small set of common sense axioms (the Cox-Jaynes axioms) so that preference for model M, under background information D can be expressed uniquely by a single number, denoted $p(M|D)$, between 0 and 1. p represents our degree of belief or confidence in model or hypothesis M. It turns out that under the Cox-Jaynes axioms, p must satisfy all the rules of probability theory. In particular, p satisfies:

Sum rule:

$$p(A|B) = 1 - p(\bar{A}|B) \tag{1.1}$$

Product Rule:

$$p(A,B|C) = p(A|C)p(B|A,C) \tag{1.2}$$

Bayes Theorem:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{1.3}$$

Thus $p(M|D)$ is also called the probability of M given the data D.

Therefore the basic engine of inference consists in first selecting a class of models, or a hypothesis space, and then in cranking the rules of probability theory to estimate numbers such as $p(M|D)$. Naturally, the Bayesian approach allows one also to compare models across different classes.

In machine learning approaches, models are often parametrised by a large vector of parameters w, so that we can write $M = M(w)$, or just w for a model. A suitable goal is often to find the most probable model, that is

$$\arg\max_w p(w|D) = \arg\max_w p(D|w)\frac{p(w)}{p(D)} \tag{1.4}$$

or, in equivalent form, by taking logarithms:

$$\arg\max_w \log p(w|D) = \arg\max_w \log p(D|w) + \log p(w) - \log p(D) \tag{1.5}$$

This is also called MAP (maximum a posteriori) estimation. $p(D|w)$ is the data likelihood. $p(w)$ is the prior on w and is equivalent, from an optimisation standpoint, to the inclusion of a regulariser term in the objective function. The normalising constant $p(D)$ does not depend on w, and is irrelevant for this optimisation. ML (maximum likelihood estimation) is the maximization of the likelihood term only. With uniform prior, MAP reduces to ML. It is important to notice that, in Bayesian inference, the proper focus should be first on the distribution p, over the entire space of models, rather than on a single model associated with the peak of this distribution. For instance in in a regression problem with $y = f_w(x)$, the proper Bayesian prediction of y given an input x is obtained by integrating over all possible models

$$E(y) = \int f_w(x)p(w|D)dw \tag{1.6}$$

If the distribution p is peaked around its mode, a good approximation to (1.6) can be obtained from the mode only. One important observation is that the background information D is not closed: as more data is acquired and experiments are performed, D is enlarged and the distribution p should in principle be recalculated. It is easy, using the rules of probability, to provide update rules for $p(M|D_1, D_2)$.

The Bayesian theoretical framework is simple enough to describe. In practice, however, the calculation of $p(w|D)$, of its mode, or, even worst, of expectations such as (1.6), can be numerically very intensive or intractable. One must resort to approximation methods. Monte Carlo methods to approximate expectations and optimise objective functions are important in this regard. This is also where important connections between the fields of machine learning, statistical inference, and statistical mechanics arise. In many practical situations, one is lead to approximate the mode of $p(w|D)$ using iterative methods such as gradient descent, the EM (Expectation- Maximization) algorithm, simulated annealing, and their variants. It is this optimisation process that is often equated with "learning". Learning algorithms and Monte Carlo methods will be discussed in more detail in the tutorial (see slides).

Armed with a solid probabilistic foundation for statistical modeling and inference, we can now turn to the problems of interest in computational molecular biology. Depending on the problem considered, different classes of models are possible. In the tutorial, and if time permits, we will review four classes of models: Hidden Markov Models (HMMs), artificial Neural Networks (NNs), Belief Networks (BNs), and Stochastic Grammars, such as Stochastic Context Free Grammars (SCFGs) and the more general Stochastic Graph Grammars (SGGs) (see slides). When dealing with DNA and protein primary sequences, HMMs are one of the most flexible and powerful class of models, particularly well suited for several basic computational tasks, such as multiple alignments and data base searches. In the rest of these notes, we will concentrate on HMMs, leaving the other model classes for the tutorial presentation (see slides). The emphasis in the notes is on the theory of HMMs, and how to apply them to problems in molecular biology. Examples of specific applications will be given during the tutorial.

2 HIDDEN MARKOV MODELS

HMMs are a class of statistical models for time series with a wide range of applications. Whereas traditionally HMMs have been applied mostly to speech recognition (Levinson et al. (1983), Rabiner (1989)), in recent years their scope has been extended to many other domains, such as handwriting recognition, financial market prediction, ion channel biophysics (Ball and Rice (1992)), neuronal spike train analysis, and protein/DNA modeling (Baldi et al. (1994), Krogh et al. (1994)). Here, we first briefly review HMMs and how HMMs can be applied to biological sequences, starting with protein families and then moving to genes.

A first order discrete HMM is completely defined by a set of states S, an alphabet of m symbols, a probability transition matrix $\mathbf{T} = (t_{ij})$, and a probability emission matrix $\mathbf{E} = (e_{iX})$. The model is intended to describe a stochastic system that evolves from state to state, while randomly emitting symbols from the alphabet. When the system is in a given state i, it has a probability t_{ij} of moving to state j, and a probability e_{iX} of emitting symbol X. The model is called hidden because what is observed is the output string of symbols from the system and one of the goals is to gather information about the hidden set of transitions that may have led to its production. HMMs can also be viewed as Stochastic Regular Grammars in the Chomsky language hierarchy.

As in the application of HMMs to speech recognition, a family of related primary sequences can be seen as a set of different utterances of the same word, generated by a common underlying HMM with a left-right architecture, i.e. once the system leaves a given state it can never return to it. Examples of standard architectures used in our experiments can be seen in Fig. 1 and in the slides. For the corresponding alphabets, $m = 4$ in the case of DNA or RNA sequences, one symbol per nucleotide, and $m = 20$ in the case of proteins sequences, one symbol per amino acid. Common knowledge about evolutionary mechanisms suggests to introduce three classes of states (in addition to the start and end states): the main states, the delete states and the insert states with $\mathbf{S} = \{start, m_1, ..., m_N, i_1, ..., i_{N+1}, d_1, ..., d_N, end\}$. N is the length of the model. Usually, it is set equal to the average length of the sequences in the family being modeled. Alternatively, N can be iteratively adjusted during learning, as in (Krogh et. al (1994)). Prior to any learning, the transition and emission parameters of a model can be initialized uniformly, at random or according to any other desirable distribution. The main and insert states always emit a letter of the alphabet, whereas the delete states are mute. The linear sequence of state transitions $start \rightarrow m_1 \rightarrow m_2 .... \rightarrow m_N \rightarrow end$ we call the backbone of the model. Corresponding to each main state, insert and delete states are needed to model insertions and deletions, with respect to the backbone. Self loops on the insert states allow for multiple insertions. Architectural

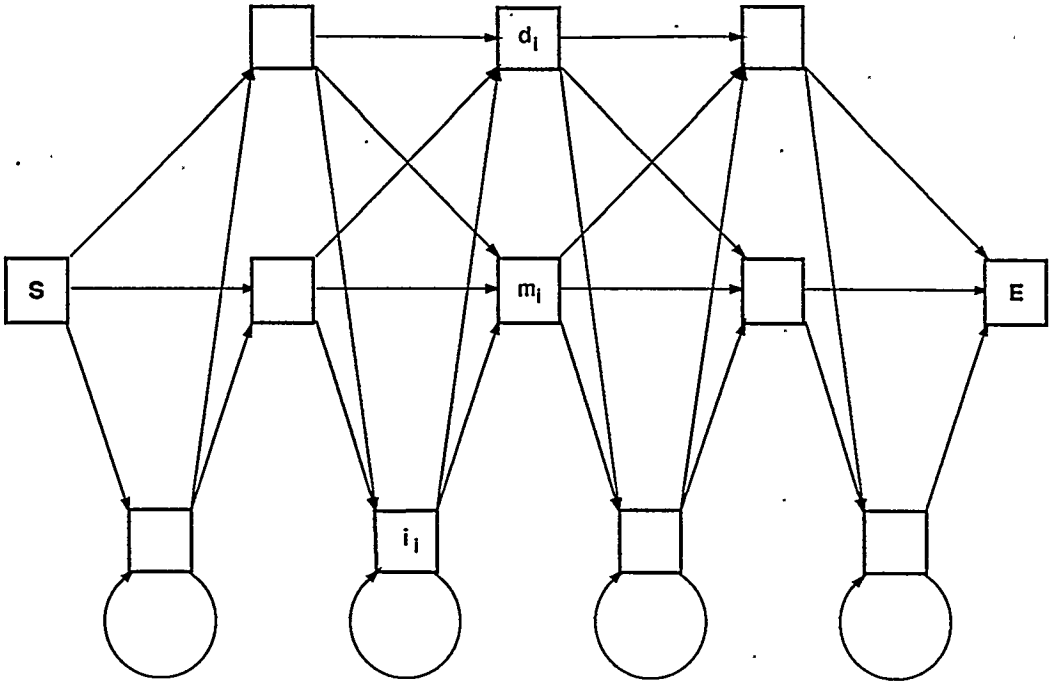


Figure 1: Basic HMM architecture for DNA or protein modeling. S start state. E end state. m_i main state along the backbone. i_i insert state with loops for multiple insertions. d_i delete state which is mute.

variations are possible and may be taylored to particular problems when additional information is available.

3 LEARNING ALGORITHMS

The most important aspect of HMMs is that they are adaptive: given a set of training sequences, the parameters of a model can be iteratively modified to optimize the fit of the model to the data according to some measure, usually the product of the likelihoods of the sequences. Different algorithms are available for HMM training, such as the classical Baum-Welch algorithm (Baum (1972), Rabiner (1989)), which is a special case of the more general EM algorithm in statistical estimation (Dempster et al. (1977)), and different forms of gradient descent and their approximations (for instance Baldi and Chauvin (1994)). Learning algorithms can be distinguished depending on the function being optimized, and how the optimization is being carrried (EM/gradient descent, on-line/off-line, uniform/non-uniform initialization, with or without the Viterbi approximation). Here we give a brief overview of the main algorithms used in the experiments to be presented, using the general Bayesian framework.

Given a set of training sequences $O = \{O_1, ...O_K\}$ and a HMM $M(y)$, where y is a vector of parameters containing all the emission and transition probabilities, a reasonable goal is to find the most likely value of y given O, i.e. to maximize the quantity $p(M(y))|O)$. Using Bayes rule,

$$p(M(y))|O) = \frac{p(O|M(y))p(M(y))}{p(O)} \tag{2.1}$$

where $p(O|M(y))$ is the data likelihood and $p(M(y))$ is the prior on the model. This is the standard framework of maximum a posteriori estimation (MAP). By taking negative logarithms

of both sides and noticing that $p(O)$ does not depend on y, our goal becomes the minimization of

$$-\log p(O|M(y)) - \log p(M(y)) \tag{2.2}$$

At this point, it is standard to assume that the sequences are independent and therefore $p(O|M) = p(O_1|M)...p(O_K|M)$. So that finally we want to find

$$\min_y -\sum_k \log p(O_k|M(y)) - \log p(M(y)) \tag{2.3}$$

The effect of the prior is thus equivalent to the introduction of a regularizer in the objective function. If the prior term is neglected, which is course also equivalent to assuming a uniform prior on the models, then the goal reduces to standard maximum likelihood (ML) estimation.

For the time being, let us concentrate on the likelihood term or, equivalently, on ML. In general, there is no way of directly finding a value of y which minimizes the negative log-likelihood of the data. So one has to resort to some iterative procedure. The best known is the Baum-Welch or EM algorithm. The EM update equations are usually computed in batch mode (across all sequences) and given by

$$t_{ij} = \frac{n_{ij}}{n_i} \quad \text{and} \quad e_{iX} = \frac{m_{iX}}{m_i} \tag{2.4}$$

where $n_i = \sum_j n_{ij}$, $m_i = \sum_X m_{iX}$ and n_{ij} and m_{iX} are the expected counts derived from a double dynamic programming procedure, known as the forward-backward algorithm (see Rabiner (1989) for additional details). Thus the EM algorithms resets the parameters to their most likely value, given the data and the current value of the parameters. A classical theorem shows that each EM steps increases the likelihood of the sequences, until a (possibly local) maximum is reached.

In the examples to be presented in the tutorial, we have also extensively used a form of gradient descent analysed in Baldi and Chauvin (1994). More specifically, we first reparametrize the model using normalized exponentials and a new set of variables w_{ij} and v_{iX}

$$t_{ij} = \frac{e^{w_{ij}}}{\sum_k e^{w_{ik}}} \quad \text{and} \quad e_{iX} = \frac{e^{v_{iX}}}{\sum_Y e^{v_{iY}}} \tag{2.5}$$

This reparametrization has two advantages: (1) modification of the w's and v's automatically preserves the normalisation constraints on the original emission and transition probability distributions; (2) transition and emission probabilities can never reach the absorbing value 0. The on-line gradient descent on the negative log-likelihood are:

$$\Delta w_{ij} = \eta(n_{ij} - n_i t_{ij}) \quad \text{and} \quad \Delta v_{iX} = \eta(m_{iX} - m_i e_{iX}) \tag{2.6}$$

where η is the learning rate, $n_i = \sum_j n_{ij}$ (resp. $m_i = \sum_Y m_{iY}$), n_{ij} (resp. m_{ij}) are again expected counts derived by the forward-backward procedure, this time for each single sequence if the algorithm is to be used on-line. On-line algorithms do not require memorizing the contribution of each sequence and have the added advantage of being smoother. They also perform a sort of stochastic gradient descent and, as such, may be able to avoid being trapped in some local minima.

3.1 VITERBI ALGORITHM

The Viterbi algorithm is a recursive dynamic programming procedure (similar to the forward propagation) that computes the most likely path associated with a given sequence in a given

model. This can be done efficiently in $O(N^2)$ steps. In certain applications, especially if the Viterbi paths tend to dominate all the other ones, the likelihood of the Viterbi path is used as an approximation to the full likelihood $p(O|M)$. Alternatively, one may define the likelihood of a sequence as being the likelihood associated with its Viterbi path and proceed accordingly in the previous learning algorithm. This has been particularly effective in the case of protein family modeling, probably because of the particular significance attached with the optimal paths.

The optimal paths of each sequence can be used during training by computing approximate counts of transitions and emission, in batch mode, and then use the EM update equations 2.4. Alternatively one can use another algorithm discussed in (Baldi and Chauvin (1994)). At each iteration, transition and emission probabilities are increased along the Viterbi path of the corresponding sequence. Specifically, at each step along a Viterbi path, and for any state i on the path, the parameters of the model are updated according to

$$\Delta w_{ij} = \eta(T_{ij} - t_{ij}) \quad \text{and} \quad \Delta v_{iX} = \eta(E_{iX} - e_{iX}) \tag{2.7}$$

where η is the learning rate. $T_{ij} = 1$ (resp. $E_{iX} = 1$) if the $i \rightarrow j$ transition (resp. emission of X from i) is used, and 0 otherwise. In the case of a loop, as for the insert states, (2.7) must be repeated every time the loop is traversed. The new parameters are therefore updated incrementally, using the discrepancy between the frequencies induced by the training data and the probability parameters of the model. These update rules must be repeated for each training sequence until equilibrium It can be shown that this algorithm approximates a gradient descent procedure on the negative log-likelihood of the sequences given the model. As such, it can be expected to converge also to a (possibly local) maximum likelihood estimator. Finally, even if Viterbi algorithm is not used for training, it is essential for aligning any sequence to the model, or for aligning two sequences to each other. Both tasks are chieved by calculating the corresponding most likely paths.

3.2 REGULARIZATION

Once an architecture has been chosen, there are still many ways by which prior information can be incorporated into the structure of the weights, their initialisation, and/or the learning algorithm. Here, we consider only one of the most simple class of priors one can use for both the emission e_{iX} and transition t_{ij} probabilities of a model, namely Dirichlet priors. Dirichlet priors are also considered in Krogh et al. (1994a). By definition, a Dirichlet distribution on the probability vector $\vec{p} = (p_1, ..., p_K)$, with parameters α and $\vec{q}$, has the form

$$D_{\alpha\vec{q}}(\vec{p}) = \frac{\Gamma(\alpha)}{\prod_i \Gamma(\alpha q_i)} \prod_i p_i^{\alpha q_i - 1} = \prod_{i=1} p_i^{\alpha q_i - 1} / Z(i) \tag{2.8}$$

with $\alpha, p_i, q_i \geq 0$ and $\sum p_i = \sum q_i = 1$. For such a Dirichlet distribution, $E(p_i) = q_i$, $\mathrm{Var}(p_i) = q_i(1 - q_i)/(\alpha + 1)$ and $\mathrm{Cov}(p_i p_j) = -q_i q_j/(\alpha + 1)$. The parameters α determines how peaked is the distribution around its mean $\vec{q}$. To allow for maximum flexibility, each state should have its own prior on both emissions and transitions. In the most general case, this prior is not necessarily a Dirichlet distribution but could be more complicated, for instance a mixture of Dirichlets. Assuming as usual that all the individual state priors are independent, the total prior on the model has the form

$$\prod_{i \in \mathrm{S}} D_{\alpha(i)\vec{q}_i}(t_{ij}) \prod_{i \in \mathrm{E}} D_{\beta(i)\vec{r}_i}(e_{iX}) \tag{2.9}$$

where S (resp E) is the set of all states (resp. of all emitting states, i.e. main and insert states).

To avoid having too many parameters, it is natural to link the different priors so as to have the same Dirichlet transition distribution out of all main states, and similarly for insert and delete states, as well as for emission Dirichlet for the main and insert states. We then need to choose the 5 corresponding parameters α's (3) and β's (2), and the 5 corresponding parameter vector $\vec{q}$'s (3) and $\vec{r}$'s (2). For the emissions, it is natural to initialize the vectors $\vec{r}$ uniformly, or from the average composition of the family being modeled. Limited experiments seem to indicate that in many cases this choice does not make much of a difference. So that in general we initialize emission parameters uniformly. Notice also that when negative logarithms of the priors are taken, the quantities $\beta r_i - 1$ appear classically as regularization constants. Thus, when a Dirichlet prior is centered on a uniform distribution, the strength of the corresponding regularizer can be made arbitrarily small, by choosing the corresponding β as close to $1/r_i = 1/r$ as desired. In all the other cases, however, and especially so for a Dirichlet which is centered on a vector $\vec{r}$ far away from uniform, it is impossible to arbitrarily reduce the influence of the corresponding regularizer, on the objective function, by varying β. The same holds true for transition priors.

For the transitions, and again after limited experimentation and for simplicity, we have also often used a uniform Dirichlet prior on the transitions out of the delete and insert states. This in conjunction with a choice of the corresponding α which eliminates altogether the effect of the corresponding prior or regularization term. Alternatively, one can use a Dirichlet prior which favors the transition from a delete, or an insert state, back to the corresponding main state on the backbone. For the main state transitions, however, we have found it essential to introduce a prior which tends to favor the backbone. This is particularly true with the architecture of Fig. 1. In this architecture, the main states and the insert states have the same fixed fan-out of 3. Therefore, everything else being equal, there is no preference for emitting symbols from the backbone, rather than from the insert states. This must be corrected at initialization time, and/or by including an appropriate prior. In the following experiments, we have often initialized all the transitions uniformly but used a Dirichlet prior on the main to main transitions, typically with $\alpha = 3.01$ and $\vec{q} = (1.01/3.01, 1/3.01, 1/3.01)$. Thus the function to be optimized contains a regularizer term associated with the probability of the backbone in the form:

$$\min_y - \sum_{k=1}^{K} \log p(O_k|M(y)) - \gamma \sum_{i=0}^{N} \log t_{m_i m_{i+1}} \qquad (2.10)$$

where $t_{m_i m_{i+1}}$ is the transition probability between two consecutive main states ($m_0 = start$ and $m_{N+1} = end$). γ is the the regularization constant that controls the relative influence of the prior term and the data likelihood term ($\gamma = 0.01$). Naturally, other priors could be investigated, based on statistical information, such as substitution matrices (Baldi (1995)), or any structural or functional information (hydrophobicity, motifs,...).

4 PROTEIN APPLICATIONS

Regardless of the training algorithm used, once a HMM has been successfully trained on a family of primary sequences, it constitutes a model of the entire family, and can be used in a number of different tasks. First, for any given sequence, we can compute its likelihood according to the model, and also its most likely path using the Viterbi algorithm. A multiple alignment results immediately from aligning all the optimal paths of the sequences in the family. The model can also be used for classification, i.e. to decide whether a given sequence belongs to the family or not. This can be done by comparing the likelihood of the sequence, according to the model, to

the likelihoods of the sequences in the family. Entire data base of sequences can be searched in this fashion (Krogh et al.(1994), Baldi and Chauvin (1994)). The method can also be adapted to fragments and data base mining problems. The model can also be used to detect motifs, by examining the profile of the emission entropy (Blahut (1987)) across the model, and sub-families, by examining how different protein paths cluster throughout the model. As in speech applications, HMMs can also be combined in a hierarchical and modular fashion to cover, for instance, motifs, domains, sub-families, families and superfamilies. So far, HMMs have been applied to several protein families including globins, immunoglobulins, kinases, growth factors, G-protein-coupled receptors (GPCRs), EF hand, aspartic acid proteases and HIV membrane proteins. In all cases, the HMMs models have been able to perform well on all previous tasks yielding, for instance, multiple alignments that are comparable to those derived by human experts and published in the literature. Complete details and results cannot be reported here and can be found in the references given at the end. Detailed examples from the GPCR family (Baldi and Chauvin 1994) will be given in the tutorial.

5 DNA APPLICATIONS

HMMs can of course also be applied to DNA problems in a similar way, by using instead an alphabe with $m = 4$ letters. HMMs have been used to detect coding regions in bacterial DNA (Krogh et al. (1994)). In our own work, we have applied HMMs to model exons, introns, and splice sites in human DNA (Baldi et al. (1994, 1995). At this stage, the main purpose was not to build an efficient program for gene finding and gene parsing, but rather to see whether new statistical patterns would emerge from the application of a new method. The most interesting result so far has been the detection of a particular periodic pattern, besides the reading frame, present in exons and in the flanking intron regions. The pattern, which has the consensus: non-T, (A or T), G and a minimal periodicity of roughly 10 nucleotides, is not a consequence of nucleotide statistics in the three codon positions, nor of the well known nucleosome positioning signal. In Baldi et al. (1995), we discuss the possible implications of this pattern and its relation to DNA bending and curvature. The analysis of periodic patterns requires the introduction of new HMM architectures containing loops. These recent results will be covered in the tutorial (see slides).

6 LIMITATIONS AND OPTIMALITY OF HMMs

In spite of their success, HMMs for biological sequences, and other applications, have two weaknesses. First, they have a large number of unstructured parameters. In the case of protein models, a typical architecture (Fig. 1) has a total of approximately $49N$ parameters ($40N$ emission parameters and $9N$ transition parameters). For a typical protein family, N is of the order of a few hundreds, resulting immediately in models with over 10,000 free parameters. This can be a problem, especially when only few sequences are available for training, not an uncommon situation in early stages of genome projects. It should be noted, however, that a single sequence should not be counted as a single training example. Each letter, and each succession of letters, in the sequence should be considered as a "training example" for the HMM parameters. Thus a typical sequence provides of the order of 2N constraints, and 25 sequences or so provide a number of examples in the same range as the number of HMM parameters.

In our experience, we have derived good multiple alignments with sometimes as little as 35 sequences in a family. We also conjecture that a HMM , trained with only two sequences and the

proper regularisation, should be able to yield optimal pairwise alignments. More generally, it has been noticed several times in the connectionist literature that certain models are "well adapted" to certain tasks, in the sense that little overfitting is observed, even with an unfavorable ratio of number of parameters to number of training examples. We believe that HMMs are well adapted for protein modeling, and in some sense, to be discussed below, optimal. But the fact remains that some improvement should be possible in situations where little training data is available. Furthermore, the HMM parameters have no structure, and no explicit relations among them. The hybrid HMM/NN architectures described in the next section provide a solution to these problems.

A second limitations of first order HMMs, is their inability to deal with long range dependencies. By definition, the emission distributions in a first order HMM depend only on the current state. This can obviously be a problem, whenever local statistics are highly context dependent. Most interesting problems will exhibit variable correlations, often over several scales. For instance, when X is emitted from state i in a sequence O, it is generally followed by Y at state j, and when X' is emitted from i, in a different sequence O', it tends to be followed by Y' at j. A single HMM has a fixed emission vector at i, and a fixed emission vector at j. Therefore it cannot capture such correlations. Detecting variable correlations is of course linked to problems of self-organization and classification. For instance, in the simple example given above, the data contains at least two different subclasses of sequences, associated wtih the $X - Y$ and $X' - Y'$ pairings.

Proteins fold into complex 3D shapes, essential to their function. Subtle long range dependencies in their polypeptide chains exist that are not immediately apparent in the primary sequences alone. It may seem surprising that good models can be derived using simple first order Markov processes. One partial explanation for this is that HMMs can capture those effects of physical long range interactions that manifest themselves in a more or less *constant* fashion, across a family of sequences. For instance, suppose that, as a result of a particular folding structure, two distant regions of a protein have a predominantly hydrophobic composition. Then this pattern is present in all the members of the family, and will be learnable by a HMM. On the other hand, a *variable* long range interaction such as: "a residue X at position i implies a residue f(X) at position j" cannot be captured by a first order HMM, as soon as f is sufficiently complex. [Note that a HMM may still be capable of capturing certain variable long range interactions. For instance, in the $X - Y/X' - Y'$ example above, the 2 sub-classes of sequences in the family could be associated with 2 types of paths in the HMM where, for instance, $X - Y$ are emitted from main states and $X' - Y'$ are emitted from insert states]. Although these dependencies are important, their effects do not seem to have hampered the HMM approach. Indeed, in the example above, consider the standard case of a data base search with a HMM trained on a protein family. To fool the model, sequences would have to exist having all the same first order properties associated with all the emission vectors (i.e. the right statistical composition at each position) similar to the sequences in the family, but with a $X' - Y$ or $X - Y'$ association, rather than a $X - Y$. This is highly unlikely, and current experimental evidence shows that HMM performance in sequence data base mining is excellent.

A slightly different point of view is to consider, for a given protein family, a hierarchy of models. All the models have the same structure, and are entirely defined by a fixed emission distribution vector at each position. These model can also be seen as factorial distributions over the space of sequences. One model, within this class, is when the emission vector is constant at all positions, and equal to the average amino acid composition of the family, or of the data base that is being searched. This is the standard model that is used for comparison in many statistical discrimination tests. Any good model must fair well against this one. Within the

factorial class, the best model is the one that assigns a different, but fixed, emission distribution to each column, identical to the one derived from a good multiple alignment. Likewise, it is also essentially equivalent to a well trained HMM, since a good alignment can be derived from a HMM, and vice versa, the parameters of a HMM can be estimated from a good alignment. Therefore in a sense HMMs are optimal within this limited hierarchy of models that assign a fixed distribution vector at each position. It is *impossible* to capture dependencies of the form $X - Y$ and $X' - Y'$ within this hierarchy, since this would require, in its more general form, *variable* emission vectors at the corresponding positions, together with a mechanism to link them in the proper way. We now consider one possible direction for overcoming the HMM limitations by using hybrid HMM/NN architectures. This section requires some previous background on NNs.

7 HMM/NN HYBRID ARCHITECTURES

7.1 BASIC IDEA

In a general HMM, an emission or transition vector parameter θ is a function of the state i only. That is $\theta = f(i)$. The first basic idea to derive HMM/NN hybrid architectures, is to have a NN sitting on top of the HMM, for the computation of the HMM parameters, i.e. for the computation of the function f. NNs are universal approximators and therefore can represent any f. More importantly perhaps, NN representations of the parameters enable the flexible introduction of many possible constraints, and the control of model complexity.

For simplicity, in the rest of these notes we discuss emission parameters only, but the approach extends immediately to transition parameters as well. In the reparametrisation of (2.5), we can consider that each one of the HMM emission parameters is calculated by a small NN, with one on/off input, no hidden layers and, in the case of protein models, 20 softmax (or normalized exponential) output units. The connections between the input and the outputs are the v_{iX}. This can be generalized immediately by having arbitrarily complex NNs, for the computation of the HMM parameters. The NNs associated with different states can also be linked with one or several common hidden layers. In general, we can consider that there is one global NN connecting all the HMM states to their parameters. The architecture of the network should be dictated by the problem at hand. In the case of a discrete alphabet however, such as for proteins, the emission of each state is a multinomial distribution, and therefore the output of the corresponding network should consist of M softmax units.

As a concrete example, consider the following hybrid HMM/NN architecture.

1. Input layer: one unit for each state i. At each time, all units are set to 0, except one which is set to 1. If unit i is set to 1, the network computes e_{iX}, the emission distribution of state i.
2. Hidden layer: H hidden units indexed by h, each with transfer function f_h (logistic by default) with bias b_h ($H < M$).
3. Output layer: M softmax units or weighted exponentials, indexed by X, with bias b_X.
4. Connections: $\alpha = (\alpha_{hi})$ connects input position i to hidden unit h. $\beta = (\beta_{Xh})$ connects hidden unit h to output unit X.

For input i, the activity in the h-th unit in the hidden layer is given by:

$$f_h(\alpha_{hi} + b_h) \tag{7.1}$$

The corresponding activity in the output layer is

$$e_{iX} = \frac{e^{-[\sum_h \beta_{Xh} f_h(\alpha_{hi}+b_h)+b_X]}}{\sum_Y e^{-[\sum_h \beta_{Yh} f_h(\alpha_{hi}+b_h)+b_Y]}} \tag{7.2}$$

A number of points are worth noticing:

- The HMM states can be partitioned into different groups, with different networks for different groups. In the limit of one network per state, with no hidden layers (or with $H = M$ hidden units), one obtains the architecture used in Baldi (1994), where the HMM parameters are reparametrised as normalised exponentials In protein applications, for instance, one can use different NNs for insert states and for main states, or for different group of states along the protein sequence corresponding for instance to different regions (hydrophobic, hydrophilic, alpha-helical, etc...).

- HMM parameter reduction can easily be achieved using small hidden layers with H hidden units, and H small compared to N or M. With a typical HMM for protein and the NN described above, with H hidden units and considering only main states, the number of parameters is $H(N+M)$ in the HMM/NN architecture, versus NM in the corresponding simple HMM. For protein models, this yields roughly HN parameters for the HMM/NN architecture, versus 20N for the simple HMM.

- The number of parameters can be adaptively adjusted to variable training set sizes, merely by changing the number of hidden units. This is useful in environments with large variations in data base sizes, as in current molecular biology applications. The total number of protein families is believed to be on the order of a thousand (Chotia (1992)). One can envision building a library of HMMs, one model per family, and update the library as the data bases grow.

- Because the number of parameters can be significantly reduced, training of hybrid architectures, along the lines described below, is also faster in general.

- The entire bag of well-known connectionist tricks can be brought to bear on these architectures including: higher order networks, radial basis functions and other transfer functions, multiple hidden layers, sparse connectivity, weight sharing, weight decay, gaussian and other priors, hyperparameters and regularization, to name only the most commonly used. Many sensible initialization and structures can be implemented in a flexible way. For instance, by allocating different number of hidden units to different subsets of emissions or transitions, it is easy to favor certain classes of paths in the models, when needed. In typical HMM architectures for molecular biology applications, one must in general introduce a bias favoring main states over insert states, prior to any learning. It is easy also to tie different regions of a protein that may have similar properties by weight sharing, and other types of long range correlations.

- By setting the output bias to the proper values, the model can be initialized to the average composition of the training sequences, or any other useful distribution.

- Classical prior information in the form of substitution matrices is also easily incorporated. Substitution matrices (for instance Altschul et al. (1991)) can be computed from data bases, and essentially produce a background probability matrix $P = (p_{XY})$, where p_{XY} is the probability that X be changed into Y over a certain evolutionary time. P can be implemented as a linear transformation in the emission NN.

- HMMs with continuous emission distributions can easily be incorporated in the HMM/NN framework. The continuous emission distributions can be represented in the output layer in many ways, for instance in the form of samples, moments and/or mixture coefficients. In the classical mixture of gaussian case, means, covariances and mixture coefficients can be computed by the NN. Likewise, additional HMM parameters, such as exponential parameters to model the duration of stay in any given state, can be calculated by a NN.

- Finally, by looking at the structure of the weights and the activity of the hidden units, it may be possible to detect certain patterns in the data.

With hybrid HMM/NN architectures, in general the M step of the EM algorithm cannot be carried analytically. Here, we describe two simple training algorithms for HMM/NN architectures. As in the case of simple HMMs, we using gradient descent on the likelihood and on the likelihood of the Vietrbi paths. Learning can be on-line or off-line. Here we give the on-line equations (batch equations can be derived similarly). So, for a single sequence O, we need to compute the partial derivatives of $\ln p(O)$, or $\ln p(\pi(O))$, where $\pi(O)$ is the Viterbi path of sequence O, with respect to the parameters α, β and b of the network.

7.2 GRADIENT LEARNING ON LIKELIHOOD

Let $Q(O) = \ln p(O)$. If $m_{iX}(O)$ is the normalized count for the emission of X from i for O, derived using the forward-backward algorithm (Rabiner (1989)), then we have

$$\frac{\partial p(O)}{e_{iX}} = \frac{m_{iX}(O)}{e_{iX}} p(O) \tag{7.3}$$

so that

$$\frac{\partial Q}{\partial e_{iX}} = \sum_{k=1}^{K} \frac{m_{iX(O)}}{e_{iX}} \tag{7.4}$$

The partial derivatives with respect to the network parameters α, β and b can be obtained by the chain rule, that is by back-propagating through the network for each i. The resulting on-line learning equations are (for each i and O)

$$\begin{cases} \Delta\beta_{Xh} = \eta[m_{iX}(O) - e_{iX} m_i] f_h(\alpha_{hi} + b_h) \\ \Delta b_X = \eta[m_{iX}(O) - e_{iX} m_i] \\ \Delta\alpha_{hj} = \delta_{ij}\eta f_h'(\alpha_{hi} + b_h)[\sum_Y (m_{iY}(O) - e_{iY} m_i)\beta_{Yh}] \\ \Delta b_h = \eta f_h'(\alpha_{hi} + b_h)[\sum_Y (m_{iY}(O) - e_{iY} m_i)\beta_{Yh}] \end{cases} \tag{7.5}$$

with $\delta_{ii} = 1$, and $\delta_{ij} = 0$ for $j \neq i$. The full gradient is obtained by summing over all sequences O_k and all main states i. Thus, for instance,

$$\frac{\partial Q}{\partial \alpha} = \sum_O \sum_{i=1}^{N} \frac{\partial Q(O)}{\partial \alpha}(i) \tag{7.6}$$

and similarly for β, and the biases. It is worth noticing, as in (Baldi and Chauvin (1994)), that these learning equations are slightly different from those that would result by back-propagating on the local cross-entropy error measure, between the emission distribution e_{iX} and the target distribution m_{iX}/m_i, derived from the forward-backward algorithm.

7.3 VITERBI LEARNING

Here $Q(O) = \ln p(\pi(O))$. The component of this term that depends on emission from main states, and thus on α, β and b, along the Viterbi path $\pi = \pi(O)$ is given by

$$-\sum_{(i,X)\in\pi} \ln e_{iX} = -\sum_{(i,X)\in\pi} T_{iX} \ln \frac{e_{iX}}{T_{iX}} = -\sum_{i\in\pi}\sum_{Y} T_{iY} \ln \frac{e_{iY}}{T_{iY}} \tag{7.7}$$

where T_{iX} is the target, namely $T_{iX} = 1$ if X is emitted from main state i in $\pi(O)$, and 0 otherwise. Thus, remarkably, computing the gradient of $Q(O) = -\ln p(\pi(O))$ with respect to α, β and b is equivalent to computing the gradient of the *local* cross entropy

$$H(T,E) = -\sum_{i\in\pi} H(T_i, e_i) = -\sum_{i\in\pi}\sum_{Y} T_{iY} \ln \frac{e_{iY}}{T_{iY}} \tag{7.8}$$

between the target output and the output of the network, over all i in π. This cross entropy error function, combined with the softmax output unit, is the standard NN framework for multinomial classification (see, for instance, Rumelhart et al. (1995)). In summary, the relevant derivatives can be calculated on-line both with respect to the sequences $O_1, ..., O_K$ and, for each sequence, with respect to the Viterbi path. For each sequence O, and for each main state i on the Viterbi path $\pi = \pi(O)$, the corresponding contribution to the derivative can be obtained by standard back-propagation on the cross-entropy error function $H(T_i, e_i)$. Therefore, the Viterbi on-line learning equations, similar to (7.5), are given by:

$$\begin{cases} \Delta\beta_{Yh} = \eta(T_{iY} - e_{iY}) f_h(\alpha_{hi} + b_h) \\ \Delta b_Y = \eta(T_{iY} - e_{iY}) \\ \Delta\alpha_{hj} = \delta_{ij}\eta f_h'(\alpha_{hi} + b_h)[\sum_Y \beta_{Yh}(T_{iY} - e_{iY}) \\ \Delta b_h = \eta f_h'(\alpha_{hi} + b_h)[\beta_{Xh}(1 - e_{iX}) - \sum_{Y\neq X} \beta_{Yh} e_{iY}] \end{cases} \tag{7.9}$$

for $(i,X) \in \pi(O)$, with $T_{iX} = 1$, and $T_{iY} = 0$ for $Y \neq X$. The full gradient is obtained again by summing over all sequences O_k, and all main states i present in the corresponding Viterbi paths $\pi(O_k)$. Thus, for instance,

$$\frac{\partial Q}{\partial \alpha} = \sum_{O} \sum_{i\in\pi(O)} \frac{\partial H(T_i, e_i)}{\partial \alpha} \tag{7.10}$$

and similarly for β and the biases.

In (7.5) and (7.9), the HMM dynamic programming and the NN backpropagation components are intimately fused. These learning algorithms can also be seen as GEM (Generalized EM) (Dempster et al. (1977)) algorithms. They can easily be modified to accomodate different target functions, such as MAP optimisation with inclusion of priors.

But no matter how complex the NN component, the final model for the data remains so far a single HMM. A single HMM defines a probability distribution over the space of all possible alphabet sequences. Only an extremely small fractions of such distributions can be represented by a reasonably constrained [1] HMM. HMMs, or the equivalent multiple alignment, essentially generate the manifold of factorial distributions. In this sense, a HMM already provides a nice and compact representation of a distribution over the space of all possible sequences. A given family

[1] Any distribution can be represented by a single *exponential size* HMM, with a start state connected to different sequences of deterministic states, one for each possible alphabet sequence, with a transition probability equal to the probability of the sequence itself.

of proteins defines also a distribution D over the space of all possible amino acid sequences. Thus our problem can also be viewed as an attempt to approximate D with a factorial distribution F. A properly trained HMM defines a close to optimal factorial approximation F. We have seen that for many practical purposes, and in particular for data base mining, we can expect factorial approximations to perform very well. HMM/NN provide a further improvement on the flexibility and compression rate of such parametrisations. In particular, they enable the introduction of structure and constraints on the HMM parameters in a flexible way, and solve the problem of parameter reduction. They are inadequate, however, for capturing variable correlations. For this, one must consider more general HMM/NN hybrid architectures, where the underlying statistical model is a *set* of HMMs, or a single HMM that can be *modulated*, as a function of input or context.

To model the simple $X-Y/X'-Y'$ example described above, four different emission vectors are needed: e_i, e_j, e'_i and e'_j. Each one of these vectors must assign a high probability to the letters X, Y, X' and Y' respectively. More importantly, there must be some kind of memory, i.e. a mechanism to link the distributions at i and j so that e_i and e_j are used for sequence O, and e'_i and e'_j are used for sequence O'. The combination of e_i and e'_j (or e'_i and e_j) should be rare or not allowed, unless required by the data. Thus e_i and e_j must belong to a first HMM, and e'_i and e'_j to a second HMM, with the possibility of switching from one HMM to the other, as a function of input sequence. Alternatively, there must be a single HMM, but with variable emission distributions, modulated again by some input.

In both cases, we consider that the emission distribution of a given state depends not only on the state itself, but also on an additional stream of information I. That is now $\theta = f(i, I)$. Again in a multiple HMM/NN hybrid architecture this more complex function f can be computed by a NN. Depending on the problem, the input I can assume many different forms. One possibility, is to consider I as a representation of contextual information. When feasible, I can even be equal to the currently observed sequence O itself. Local connectivity in the NN can also ensure that only local context be taken into consideration. Other inputs are however possible, over different alphabets. An obvious candidate in protein modeling tasks would be the secondary structure of the protein (alpha helices, beta sheets and coils). In general, I could also be any other array of numbers representing useful information for the HMM modulation. Examples of such architectures will be given at ISMB95.

8 CONCLUSION

We have briefly reviewed the general Bayesian framework for statistical modeling and inference. The framework provides a solid probabilistic foundation for machine learning approaches to problems in computational molecular biology. The framework has been applied to the theory of HMMs for sequence analysis, in an elementary way. In most cases, we have just focused on trying to obtain a single model, with a high likelihood. We have not made use of the distribution over the entire space of models, mostly for computational reasons. Even in the case of a pairwise alignment problem, a full Bayesian approach would require integrating over *all* possible pairwise alignments. Likewise, in these notes, we have not attempted to compare HMMs to other model classes in a rigorous Bayesian way. In spite of all these approximations, HMMs have emerged has a flexible and powerful tool for modeling sequence consensus, multiple alignments, and data base searches. Specific examples of applications to protein and DNA modeling are given in the tutorial slides and, in more detail, in the references at the end. We have also examined the optimality of HMMs and their limitations, as well as a possible direction of research to overcome these limitations, by using hierarchical hybrid HMM/NN models. Other relevant generalisations

of HMMs for machine learning approaches to computational molecula biology include Stochastic Context Free Grammars, Stochastic Graph Grammars, and Belief Networks.

9 PARTIAL REFERENCES BY TOPIC

General Background and Computational Problems

Multiple Alignment/Classification/Data Base Searches/Phylogenetic Reconstruction

Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. (1990) Basic Local Alignment Search Tool. Journal of Molecular Biology, **215**, 403-410.

Altschul, S. F., Carroll, R. J. and Lipmann, D. J. (1989) Weights for Data Related by a Tree. Journal Molecular Biology, **207**, 647-653.

Altschul, S. F. (1991) Amino Acid Substitution Matrices from an Information Theoretic Perspective. Journal of Molecular Biology, **219**, 1-11.

Bairoch, A. (1993) The PROSITE Dictionary of Sites and Patterns in Proteins, its Current Status. Nucleic Acid Research, **21**, 3097-3103.

Barton, G. J. and Sternberg, M. J. E. (1987) A Strategy for the Rapid Multiple Alignment of Protein Sequences Confidence Levels from Tertiary Structure Comparisons. Journal of Molecular Biology, **198**, 327-337.

Barton, G. J. and Sternberg, M. J. E. (1987) Evaluation and Improvements in the Automatic Alignment of Protein Sequences. Protein Engineering, **1**, 89-94.

Cardon, L. R. and Stormo, G. D. (1992) Expectation Maximization Algorithm for Identifying Protein-binding Sites with Variable Lengths from Unaligned DNA Fragments. Journal of Molecular Biology, **223**, 159-170.

Carrillo, H. and Lipman, D. (1988) The Multiple Sequence Alignment Problem in Biology. SIAM Journal of Applied Mathematics, **48**, 5, 1073-1082.

Chan, S. C., Wong, A. K. C. and Chiu, D. K. Y. (1992) A Survey of Multiple Sequence Comparison Methods. Bulletin of Mathematical Biology, **54**, 4, 563-598.

Chotia, C. (1992) One Thousand Families for the Molecular Biologist. Nature, **357**, 543-544.

(33) Churchill, G. A. (1989) Stochastic Models for Heterogeneous DNA Sequences. Bulletin of Mathematical Biology, **51**, 1, 79-94.

Dayhoff, M. O. (1978) A Model of Evolutionary Change in Proteins. Detecting Distant Relationships. In: Atlas of Protein Sequence and Structure, M. O. Dayhoff Editor, National Biomedical Research Foundation: Washington, D. C., 345-358.

Dayhoff, M. O., Barker, W. C. and Hunt, L. T. (1983) Establishing Homologies in Protein Sequences. Methods in Enzymology, **91**, 524-545. Academic Press.

Doolittle, R. F. (1981) Similar Amino Acid Sequences: Chance or Common Ancestry? Science, **214**, 149-159.

Felsenstein, J. (1981) Evolutionary Trees from DNA Sequences: a Maximum Likelihood Approach. Journal of Molecular Evolution, **17**, 368-376.

Feng, D. F., Johnson, M. S. and Doolittle, R. F. (1985) Aligning Amino Acid Sequences: Comparison of Commonly Used Methods. Journal of Molecular Evolution, **21**, 112-125.

Feng, D. F. and Doolittle, R. F. (1987) Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees, Journal of Molecular Evolution, **25**, 351-360.
Fitch, W. M. and Margoliash, E. (1967) Construction of Phylogenetic Trees. Science, **155**, 279-284.
Fitch, W. M. (1971) Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology. Systematic Zoology, **20**, 406-416.
Fitch, W. M. and Smith, T. F. (1982) Implications of Minimal Length Trees. Systematic Zoology, **31**, 1, 68-75.
Green, P., Lipman, D. Hillier, L., Waterson, R., States, D. and Claverie, J. M. (1993) Ancient Conserved Regions in New Gene Sequences and the Protein Databases. Science, **259**, 1711-1716.
Gusfield, D. (1993) Efficient Methods for Multiple Sequence Alignment with Guaranteed Error Bounds. Bulletin of Mathematical Biology, **55**, 1, 141-154.
Henikoff, S. and Henikoff, J. G. (1992) Amino Acid Substitution Matrices from Protein Blocks. PNAS USA, **89**, 10915-10919.
Henikoff, S. and Henikoff, J. G. (1994) Protein Family Classification Based on Searching a Database of Blocks. Genomics, **19**, 97-107.
Higgins, D. G., Bleasby, A. J. and Fuchs, R. (1992) CLUSTAL V: Improved Software for Multiple Sequence Alignment. Computer Applications in the Biosciences, **8**, 189-191.
Jurka, J. and Milosavljevic, A. (1991) Reconstruction and Analysis of Human Alu Genes. Journal of Molecular Evolution, **32**, 105-121.
Karlin, S. and Brendel, V. (1992) Chance and Statistical Significance in Protein and DNA Sequence Analysis. Science, **257**, 39-49.
Karlin, S., Brendel, V. and Bucher, P. (1992) Significant Similarity and Dissimilarity in Homologous Proteins. Molecular Biological Evolution, **9**, 1, 152-167.
Klotz, L. C. Komar, N., Blanken, R. L. and Mitchell, R. (1979) Calculation of Evolutionary Trees from Sequence Data. PNAS USA, **76**, 9, 4516-4520.
Lawrence, C. E. and Reilly, A. A. (1990) An Expectation Maximization (EM) Algorithm for the Identification and Characterization of Common Sites in Unaligned Biopolymer Sequences. Proteins: Struct. Funct. Genet., **7**, 41-51.
Lawrence, C.E., Altschul, S. F., Boguski, M.S., Liu, J.S., Neuwald, A. and Wootton, J.C. (1993) Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment. Science, in press.
Lipman, D. J., Altschul, S. F. and Kececioglu, J. D. (1989) A Tool for Multiple Sequence Alignment. PNAS USA, **86**, 4412-4415.
Maier, D. (1978) The Complexity of Some Problems on Subsequences and Supersequences. Journal of the Association for Computing Machinery, **25**, 2, 322-336.
Martinez, H. M. (1988) A Flexible Multiple Sequence Alignment Program. Nucleic Acid Research, **16**, 1693-1691.
Myers, E. W. (1991) An Overview of Sequence Comparison Algorithms in Molecular Biology. Technical Report 91-29, Department of Computer Science, The University of Arizona, Tucson, Arizona and in: Calculating the Secrets of Life: Applications of Mathematical Sciences in Molecular Biology. Eric Lander and Walter Gilbert Editors. National Academy Press; Washington D.C. (to appear).
Needleman, S. B. and Wunsch, C. D. (1970) A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. Journal of Molecular Biology, 48, 443-453.
Pearson, W. and Lipman, D. (1988) Improved Tools for Biological Sequence Comparison. PNAS USA, **85**, 2444-2448.

Probst, W. C., Snyder, L. A, Brosius, J., Schuster, D. I. and Sealfon, S.C. (1992) Sequence Alignment of the G-Protein Coupled Receptor Superfamily. DNA and Cell Biology, **11**, 1, 1-20.
Rao, J. K. M. (1987) New Scoring Matrix for Amino Acid Residue Exchanges Based on Residue Characteristic Physical Parameters, Int. J. Pept. Protein Res., **29**, 276-281.
Risler, J. L., Delorme, O., Delàcroix, H. and Henaut, A. (1988) Amino Acid Substitutions in Structurally Related Proteins. A Pattern Recognition Approach. Determination of a New and Efficient Scoring Matrix. Journal of Molecular Biology, **204**, 1019-1029.
Schuler, G. D., Altschul, S. F. and Lipman, D. J. (1991) A Workbench for Multiple Alignment Construction and Analysis. Proteins, Structure, Function and Genetics, **9**, 180-190.
Sellers, P. H. (1974) On the Theory and Computation of Evolutionary Distances. SIAM Journal Applied Mathematics, **26**, 4, 787-793.
Smith, R. F. and Smith, T. F. (1990) Automatic Generation of Primary Sequence Patterns from Sets of Related Protein Sequences. PNAS USA, **87**, 118-122,
States, D. J., Gish, W. and Altschul, S. F. (1991) Improved Sensitivity of Nucleic Acid Database Searches Using Application-Specific Scoring Matrices. Methods: A Companion to Methods in Enzymology, **3**, 1, 66-70.
Strosberg, A. D. (1991) Structure-Function Relationship of Protein Belonging to the Family of Receptors Coupled to GTP Binding Proteins. European Journal of Biochemistry, **196**, 1, 1-10.
Subbiah, S. and Harrison, S. C. (1989) A Method for Multiple Sequence Alignment with Gaps. Journal of Molecular Biology, **209**, 539-548.
Taylor, W. R. (1986) Identification of Protein Sequence Homology by Consensus Template Alignment. Journal of Molecular Biology, **188**, 233-258.
Taylor, W. R. (1987) Multiple Sequence Alignment by a Pairwise Algorithm, CABIOS, **3**, 81-87.
Thorne, J. L., Kishino, H. and Felsenstein, J. (1991) An Evolutionary Model for Maximum Likelihood Alignment of DNA Sequences. Journal of Molecular Evolution, **33**, 114-124.
Vingron, M. and Argos, P. (1990) Determination of Reliable Regions in Protein Sequence Alignments. Protein Engineering, **3**, 7, 565-569.
Vingron, M. and Argos, P. (1991) Motif Recognition and Alignment for Many Sequences by Comparison of Dot Matrices. Journal of Molecular Biology, **218**, 33-43.
von Heijne, G. (1987) Sequence Analysis in Molecular Biology-Treasure Trove or Trivial Pursuit. Academic Press, San Diego, CA.
Waterman, M. S. (1984) General Methods of Sequence Comparison. Bulletin of Mathematical Biology, **46**, 4, 473-500.
Waterman, M. S. (1986) Multiple Sequence Alignment by Consensus. Nucleic Acid Research, **14**, 9095-9102.
White, J. V., Stultz, C. M. and Smith, T. F. (1993) Protein Classification by Non-Linear Optimal Filtering of Amino-Acid Sequences. Preprint.
Zuckerkandl, E. (1975) Journal of Molecular Evolution, **7**, 1.

Information Theory, Bayesian Modeling and Statistics

Blahut, R. (1987) Principles and Practice of Information Theory. Addison Wesley.
Cover, T. M. and Thomas, J. A. (1991) Elements of Information Theory. John Wiley, New York.
Cox, R. T. (1964) Probability, frequency and Reasonable expectation. American Journal of Physics, **14**, 1-13.

Gull, S. F. (1989) Developments in Maximum Entropy Data Analysis. In: Maximum Entropy and Bayesian Methods. J. Skilling Editor, Kluwer, Dordrecht, 53-71.
Jaynes, E. T. (1986) Bayesian Methods: General Background. In: Maximum Entropy and Bayesian Methods in Statistics, J.H. Justice Editor, Cambridge University Press, Cambridge, 1-25.

HMMs and EM Algorithm

Baldi, P. and Chauvin, Y. (1994a) Smooth On-Line Learning Algorithms for Hidden Markov Models. Neural Computation, **6**, 2, 305-316.
Baldi, P. (1995) Hidden Markov Models and Substitution Matrices. Journal of Computational Biology. In press.
Ball, F. G. and Rice, J. A. (1992) Stochastic Models for Ion Channels: Introduction and Bibliography. Mathematical Bioscience.
Baum, L. E. (1972) An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. Inequalities, **3**, 1-8.
Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal Royal Statistical Society, B **39**, 1-22.
Levinson, S. E., Rabiner, L. R. and Sondhi, M. M. (1983). An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition. The Bell System Technical Journal, **62**, 4, 1035-1074.
Rabiner, L. R. (1989) A Tutorial on Hidden Markor Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, **77**, 2, 257-286.

HMMs and Protein Applications

Baldi, P., Chauvin, Y., Hunkapiller, T. and McClure, M. (1994) Hidden Markov Models of Biological Primary Sequence Information. PNAS USA, **91**, 3, 1059-1063.
Baldi, P. and Chauvin, Y. (1994) Hidden Markov Models of the G-Protein-Coupled Receptor Family. Journal of Computational Biology, **1**, 4, 311-335.
Eddy, S. R., Mitchinson, G. and Durbin, R. (1995) Maximum Discrimination Hidden Markov Models of Sequence Consensus. Journal of Computational Biology. In press.
Krogh, A., Brown, M., Mian, I.S., Sjolander, K. and Haussler, D. (1994) Hidden Markov Models in Computational Biology: Aplications to Protein Modeling. Journal of Molecular Biology, **235**, 1501-1531.

HMMs and DNA Applications

Baldi, P., Brunak, S., Chauvin, Y., Engelbrecht, J. and Krogh, A. (1994) Hidden Markov Models of Human Genes. In: Advances in Neural Information Processing Systems, **6**, J. D. Cowan, G. Tesauro and J. Alspector Editors, Morgan Kaufmann, San Francisco, CA, 761-768
Baldi, P., Brunak, S., Chauvin, Y., Engelbrecht, J. and Krogh, A. (1995) Hidden Markov Models for Human Genes. Technical Report, Division of Biology, Caltech.
Baldi, P., Brunak, S., Chauvin, Y., Engelbrecht, J. and Krogh, A. (1995) Periodic Sequence Patterns in Human Exons. Proceedings of the 1995 Conference on Intelligent Systems for Molecular

Biology (ISMB95), in Cambridge (UK).
Krogh, A., Mian, I.S. and Haussler, D. (1994) A Hidden Markov Model that Finds Genes in *E. Coli* DNA. Nucleic Acid Research, **22**, 4768-4778.

Neural Networks and Computational Molecular Biology Applications

Brunak, S., Engelbrecht, J. and Knudsen, S. (1991) Prediction of Human mRNA Donor and Acceptor Sites from the DNA Sequence. Journal of Molecular Biology, **220**, 49-65.
Lapedes, A., Barnes, C., Burks, C., Farber, R. and Sirotkin, K. (1989) Application of Neural Networks and Other Machine Learning Algorithms to DNA Sequence Analysis. In: Computers and DNA, SFI Studies in the Sciences of Complexity, vol. VII, G. Bell and T. Marr Editors, Addison-Wesley.
Qian, N. and Sejnowski, T. (1988) Predicting the Secondary Structure of Globular Proteins Using Neural Network Models. Journal of Molecular Biology, **202**, 865-884.
Rumelhart, D. E., Durbin, R., Golden, R. and Chauvin, Y. (1995) Back-propagation: the Theory. In: Back-propagation: Theory, Architectures and Applications. Y.E. Chauvin and D.E. Rumelhart Editors, Chapter 1, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Hybrid HMM/NN

Baldi, P. and Chauvin, Y. (1995) Hierarchical Hybrid Modeling, HMM/NN Architectures, and Protein Applications. Neural Computation. In press. Bengio, Y. and Frasconi, P. (1995) An Input-Output HMM Architecture. In: Advances in Neural Information Processing Systems, **7**, IEEE), J. D. Cowan, G. Tesauro, and J. Alspector Editors, Morgan Kaufmann, San Francisco, CA.

Stochastic Grammars and RNA Applications

Sakakibara, Y., Brown, M., Hughey, R., Saira Mian, I., Sjölander, K., Underwood, R. C. and Haussler, D. (1994) Stochastic Context-Free Grammars for tRNA Modeling", Nucleic Acid Research, **22**, 5112-5120.
Searls, D. B (1992) The linguistics of DNA. American Scientist, **80**, 579-591.

Belief Networks

Heckerman, D. (1995) A Tutorial on Learning Bayesian Networks. Technical Report MSR-TR-95-06, Microsoft Corporation, Redmond, WA.
Pearl, J. (1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, CA.

B. SLIDES

I. INTRODUCTION

Examples of Computational Problems

- Physical and Genetic Maps
- Pairwise and Multiple Alignments
- Motif Detection/Discrimination/Classification
- Data Base Searches and "Mining"
- Phylogenetic Tree Reconstruction
- Gene Finding and Gene Parsing
- Secondary Structure Prediction
- Tertiary Structure Prediction

Multiple Alignment

- No definition of what a good alignment is (low entropy, detection of motifs).
- The multiple alignment problem is NP complete.
- Pairwise alignment can be solved efficiently by dynamic programming in $O(N^2)$ steps.
- For K sequences of average length N, dynamic programming scales like $O(N^K)$, exponentially in the number of sequences.
- Problem of variable scores and gap penalties.

Machine Learning

- Extract information from the data automatically (inference) via a process of model fitting (learning from examples).

- Model Selection: Neural Networks, Hidden Markov Models, Stochastic Grammars,...

- Model Fitting: Gradient Methods, Monte Carlo Methods, ...

- Machine learning approaches are most useful in areas where there is a lot of data but little theory.

II. STATISTICAL MODELING AND INDUCTION

- Bayesian framework for induction: we start with hypothesis space and wish to express relative preferences in terms of background information I (the Cox-Jaynes axioms).

- **Axiom 0:** Transitivity of preferences.

- **Theorem 1:** Preferences can be represented by a real number. $\pi(H)$

- **Axiom 1:** There exists a function f:

$$\pi(\bar{H}) = f(\pi(H))$$

- **Axiom 2:** There exists a function F:

$$\pi(A, B) = F(\pi(A), \pi(B|A))$$

- **Theorem 2:** There is always a rescaling w such that $p(H) = w(\pi(H))$ is in $[0, 1]$.

Probability as Degree of Belief

- **Sum rule:**

$$p(H|I) = 1 - p(\bar{H}|I)$$

- **Product Rule:**

$$p(A, B|I) = p(A|I)p(B|A, I)$$

- **Bayes Theorem:**

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

- Induction form:

$$p(Model|Data) = \frac{p(Data|Model)p(Model)}{p(Data)}$$

- Equivalently:

$$p(Model|Data, I) = p(Data|Model, I)\frac{p(Model|I)}{p(Data|I)}$$

- Sequential aspect: can have $D = (D_1, ..., D_n)$ and $I = (D_1, ..., D_{n-1})$. How to update probabilities in light of new data.
- Fundamental computational problem: calculate $p(Model|Data)$ and corresponding expectations. Intractability and need for approximations.

Model Fitting and Prediction

- Model Fitting and Model Comparison [Learning]

$$\max_M p(M|D) = p(D|M)\frac{p(M)}{p(D)}$$

- Equivalent form:

$$\max_M \log p(M|D) = \log p(D|M) + \log p(M) - \log p(D)$$

- $M = M(w)$ $w =$ vector of parameters

$$\min_w \log p(M(w)|D) = -\log p(D|M(w)) - \log p(M(w)) + \log p(D)$$

- Maximum A Posteriori (MAP), Maximum Likelihood (ML), Priors and Regularization.

- Bayesian Prediction. Need to integrate across models. Example (deterministic case) $y = f_w(x)$

$$E(y) = \int f_w(x) P(w|D) dw$$

- Hierarchical Modeling:

$$P(w) = \int P(w|\alpha) P(\alpha) d\alpha$$

- Fundamental computational problem: calculate $p(Model|Data)$ and corresponding expectations. Intractability and need for approximations.

Model classes: Examples

- Binomial/Multinomial Models
- Neural Networks
- Hidden Markov Models
- Stochastic Grammars
- Belief Networks

Expectations, Optimization and Learning Algorithms

- The goal is to compute the expectation of $< f(x_1, ..., x_n) >$:

$$< f(x_1, ..., x_n) >= \sum_{(x_1, ..., x_n)} P(x_1, ..., x_n) f(x_1, ..., x_n)$$

This can be done using, for large N, using the approximation

$$< f(x_1, ..., x_n) > \approx \frac{1}{N} \sum_{(x_1, ..., x_n)} f(x_1, ..., x_n)$$

provided $(x_1, ..., x_n)$ are sampled according to their distribution $P(x_1, ..., x_n)$.

- To sample from P, it is often necessary to use a Monte Carlo Method (Gibbs Sampling or Metropolis Algorithm)

Gibbs Sampling and Metropolis Algorithm

- Problem: sample $(X_1, ..., X_n)$ according to $P(X_1, ..., X_n)$

- Gibbs Sampling and Metropolis Algorithm are Markov Chain Methods.

- Gibbs Sampling: iteratively sample each single variable, conditioned on the most recent value of all the other variables.

Starting from $(x_1^t, ..., x_n^t)$
select x_1^{t+1} according to $P(X_1 | X_2 = x_2^t, ..., X_n = x_n^t)$
select x_2^{t+1} according to $P(X_2 | X_1 = x_1^{t+1}, ..., X_n = x_n^t)$
.....
select x_n^{t+1} according to $P(X_n | X_1 = x_1^{t+1}, ..., X_{n-1} = x_{n-1}^{t+1})$

- Metropolis Algorithm: iteratively generate random perturbations of the current state and accept or reject them according to a certain probability Q (usually derived from statistical mechanics/downhill steps are alway accepted/uphill steps are accepted probabilistically depending on the temperature).

- Symmetric global probabilistic version of the algorithm:

Starting from $x^t = (x_1^t, ..., x_n^t)$
randomly generate a new point $x^* = (x_1^*, ..., x_n^*)$ according to a probability distribution R satisfying everywhere $R(x, x^*) = R(x^*, x)$ and $R(x, x*) \neq 0$
if $P(x*) \geq P(x)$, then accept x^*, i.e. $x^{t+1} = x^*$
if $P(x*) < P(x)$, then accept x^* with probability $Q = \frac{P(x^*)}{P(x)}$

Statistical Mechanics

- $S = (x_1, ..., x_n)$ state of a system with energy $E(S) = f(x_1, ..., x_n)$

- Boltzmann-Gibbs distribution (maximal entropy under observed $< f >$):

$$P(S) = \frac{e^{-\beta E(S)}}{Z}$$

  Z is the partition function (normalizing constant) and $\beta = 1/kT$

- Metropolis Algorithm:

Starting from $S^t = (x_1^t, ..., x_n^t)$
randomly generate a new point $S^* = (x_1^*, ..., x_n^*)$ according to a probability distribution R satisfying everywhere $R(S, S^*) = R(S^*, S)$ and $R(S, S*) \neq 0$
if $E(S*) \leq E(S)$, then accept S^*, i.e. $S^{t+1} = S^*$
if $E(S*) > E(S)$, then accept S^* with probability $Q = e^{-\beta[E(S^*) - E(S)]}$

- Optimization by Simulated Annealing: In order to optimize f, we can view it as the "energy" function of a statistical mechanical system. At low temperature, the Boltzmann-Gibbs distribution is dominated by the ground state so that, as $\beta \to \infty$:

$$< S >= \sum_S S \frac{e^{-\beta E(S)}}{Z} \approx S_{opt}$$

- Simulated annealing combines the Metropolis algorithm with a schedule for lowering the temperature.

Learning

- Optimize $p(M(w)|D)$ [NP complete problem].
- Gradient Methods (gradient descent, conjugate gradient, back-propagation,...)

$$w(t+1) = w(t) - \eta \frac{\partial E}{\partial w}$$

- Monte Carlo Methods: Simulated Annealing
- Other Methods: Expectation/Maximization (EM) Algorithm

III NEURAL NETWORKS

- Early Neural Networks: one layer of hidden units, sigmoidal transfer functions, LMS error

- Modern Neural Networks: general class of bayesian statistical models

- N=network D=data=$\{(x_i, y_i)\}$ [input/output pairs usually independent of each other

- N is deterministic

- D is stochastic

- output of network=$out(x_i)$=$f(in(x_i))$=average or expectation of y_i given x_i

- Goal: to maximize $P(N|D)$ or to minimize L

$$L = -\log P(D|N) - \log P(N) + \log P(D)$$

- $logP(N)$ corresponds to the prior on the network and leads for instance to weight sharing, weight decay, and so on.

- The likelihood term has the form

$$E = -\log P(D|N) = -\sum_i \log P(y_i|x_i)$$

- Additional hypothesis can be made regarding $P(y_i|x_i, N)$.

- If $P(y_i|x_i, N)$ is gaussian, then the output transfer function should be linear, and E is the least mean square error function.

- If $P(y_i|x_i, N)$ is binomial, then the output transfer function should be sigmoidal, and E is the cross-entropy (or KL) distance.

- If $P(y_i|x_i, N)$ is multinomial, then the output transfer functions should be normalized exponentials ("soft-max"), and E is the cross-entropy (or KL) distance.

- Generalized linear models: Bayesian analysis suggests the proper choice of transfer function as well as error function. Analysis can sometimes be extended to hidden layers.

IV HIDDEN MARKOV MODELS

A Hidden Markov Model is completely defined by

- A set of states;
- An alphabet of symbols;
- A transition probability matrix $T = (t_{ij})$;
- An emission probability matrix $E = (e_{ix})$.

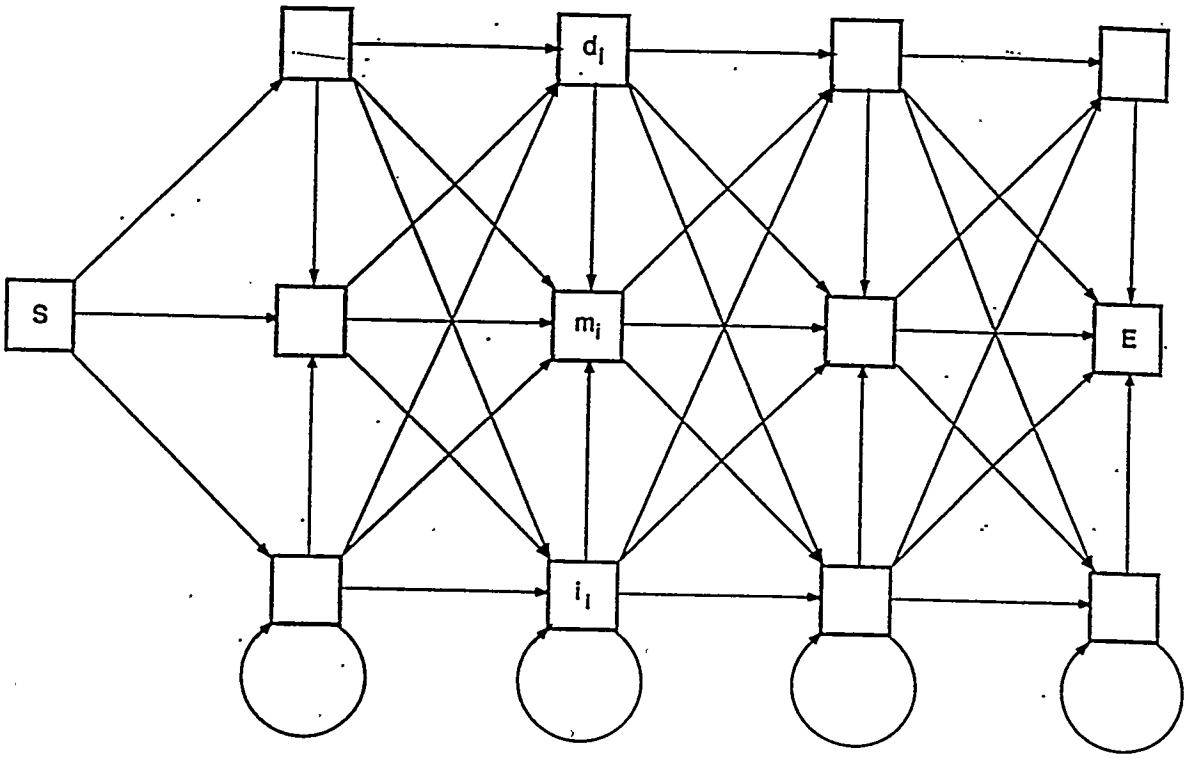
Basic Ideas

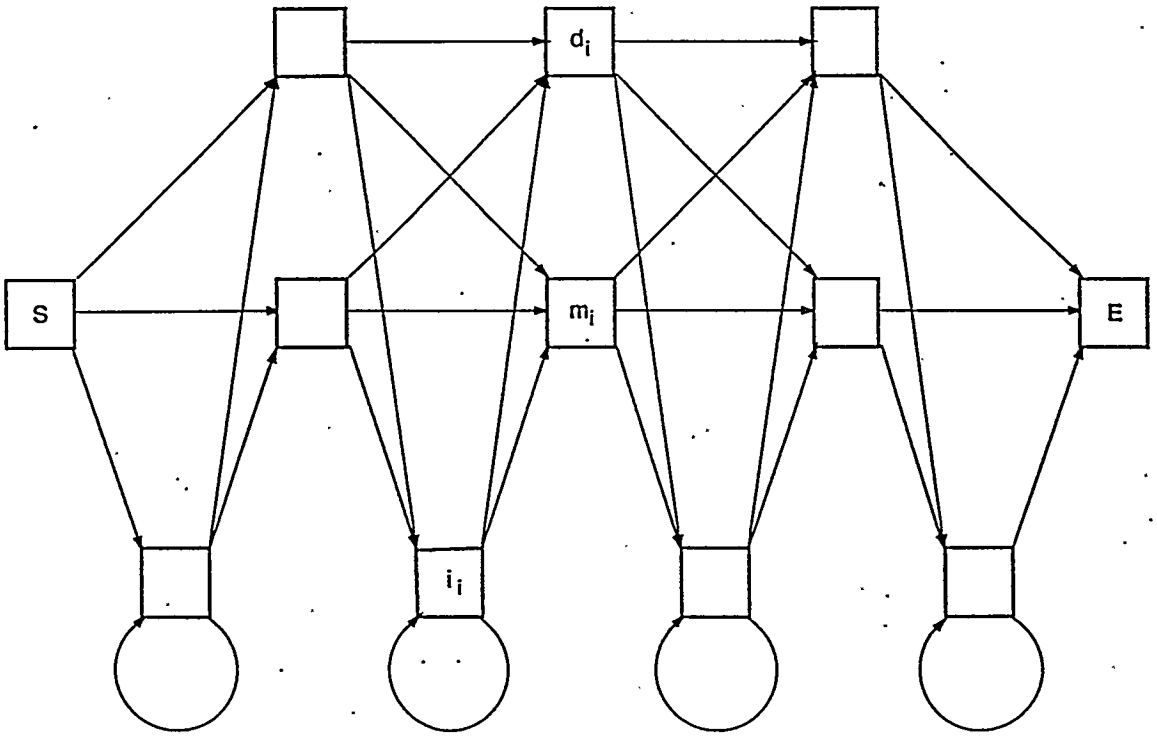
- As in speech recognition, use Hidden Markov Models (HMM) to model a family of related biological primary sequences.

- As in speech recognition, use a left to right HMM: once the system leaves a state it can never reenter it. The basic architecture consists of a main backbone chain of main states, and two side chains of insert and delete states.

- The parameters of the model are: the transition probabilities t_{ij} and the emission probabilities e_{iX}. These parameters are adjusted during training from examples.

- After learning, the model can be used in a variety of tasks including: multiple alignments, detection of motifs, classification, data base searches.


S
d_l
m_i
i_l
E


S
d_i
m_i
i_i
E

Three Basic Algorithms for HMMs

- The forward algorithm: given a sequence $O = O_1, ..., O_T$, recursively computes

$$\alpha_t(i) = P(s_i, O_1, ..., O_t/M)$$

- The backward algorithm: given a sequence $O = O_1, ..., O_T$, recursively computes

$$\beta_t(i) = P(s_i, O_{t+1}, ..., O_T/M)$$

- The Viterbi algorithm: given a sequence $O = O_1, ..., O_T$, recursively computes the most likely path π through the system, consistent with O (maximizes $P(\pi, O/M)$).

- Each one of these algorithms is based on dynamic programming and scales like $O(N^2)$.

Learning

The Baum-Welch algorithm:

- Compute the expected number n_{ij} (resp. m_{ij}) of times the $i \to j$ transition (resp. letter j emission from state i) is used, using the forward-backward algorithm.

- Reset the model parameters to the observed frequencies induced by the data according to

$$t_{ij}^{+} = \frac{n_{ij}}{n_i} \quad \text{and} \quad e_{ij}^{+} = \frac{e_{ij}}{e_i}$$

- The Baum-Welch algorithm converges to a possibly local maximum likelihood estimator! (special case of EM algorithm).

On-Line Learning Algorithms

- First, use a "normalized exponentials" representation for all the model parameters

$$t_{ij} = \frac{e^{\lambda w_{ij}}}{\sum_k e^{\lambda w_{ik}}} \quad \text{and} \quad e_{iX} = \frac{e^{\lambda v_{iX}}}{\sum_Y e^{\lambda v_{iY}}}$$

- Learning equations (and similarly for the emission parameters):

$$\Delta w_{ij} = \eta(\frac{n_{ij}}{n_i} - t_{ij})$$

$$\Delta w_{ij} = \eta(n_{ij} - n_i t_{ij})$$

$$\Delta w_{ij} = \eta(T_{ij} - t_{ij})$$

  where $T_{ij} = 0$ or 1 is the target determined by the corresponding Viterbi path.

- Properties: smooth, on-line or batch, with or without the Viterbi best path approximation, 0 probabilities are not absorbing, convergent (special case of a GEM algorithm / approximation to gradient descent on negative log-likelihood).

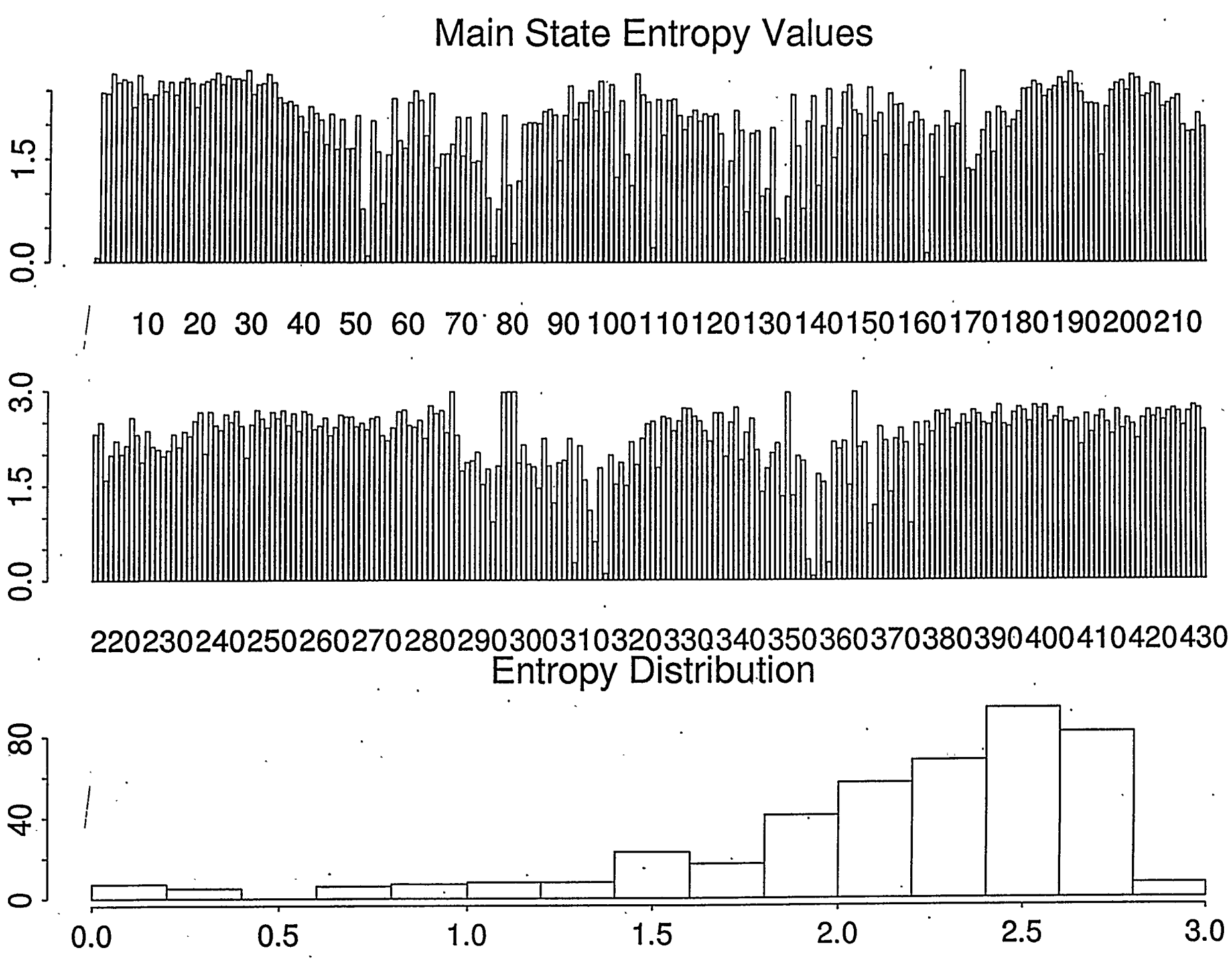
Immunoglobulins

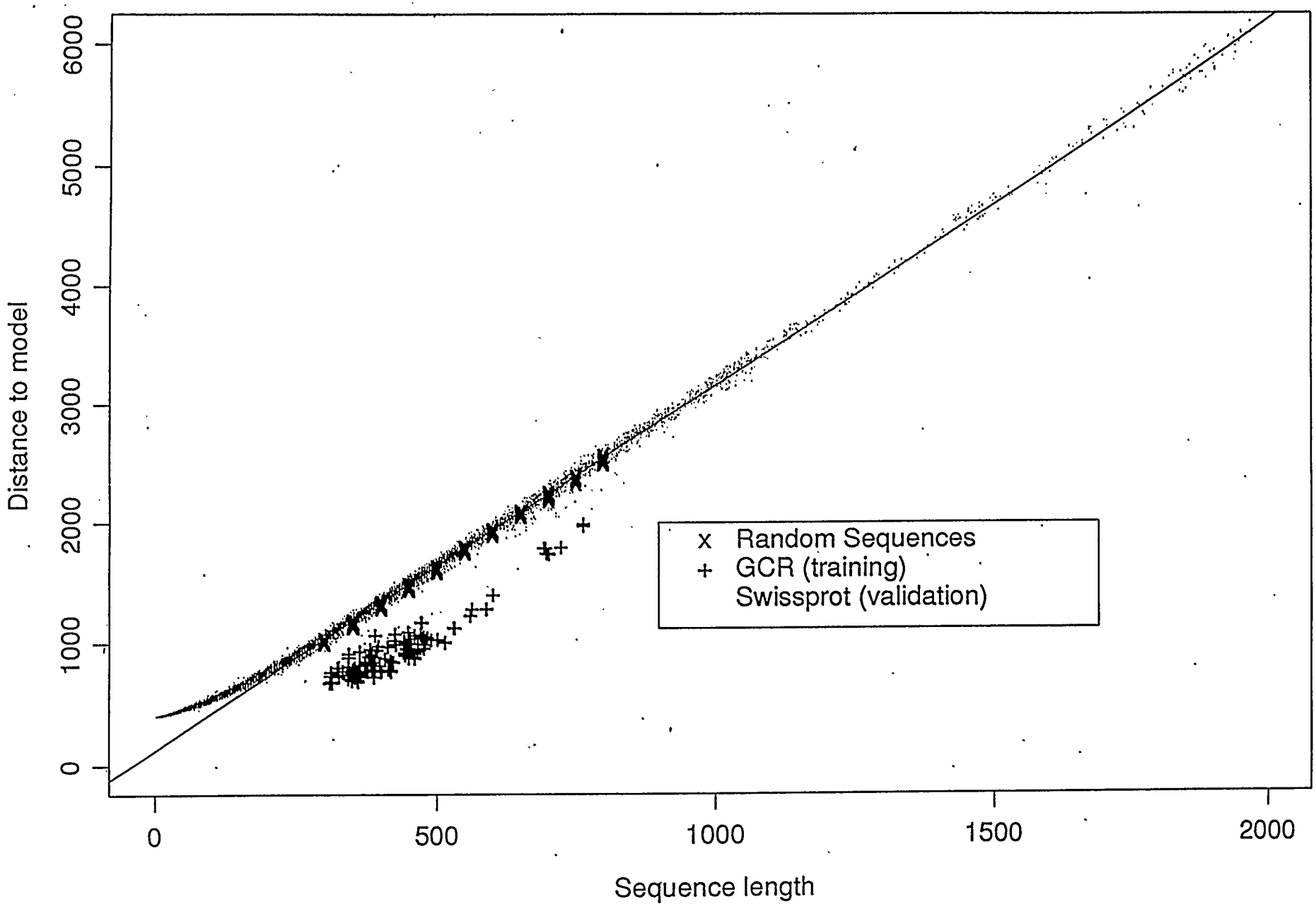
- 294 sequences (V regions) with minimal length 90, average length 117 and maximal length 254.
- Model described here has been trained with a random subset of 150 sequences.

```
                                        1           2           3         4
PH0106    mklpvrllvlmfwipasssDvVMTQTPLSLpvSLGDQASISCRSSQSLVHSngnTYLNWYLQ--KAGQS--p-KL
B27563    ----------------------LQQPGAELv-KPGASVKLSCKASGYTFTN---YWIHWVKQ--RPGRGLE-WIG
MHMS76    ------------------------ESGGGLv-QPGGSMKLSCVASGFTFSN---YWMNWVRQ--SPEKGLE-WVA
D28035    mefglswiflvailkgvqcEvRLVESGGDLv-EPGGSLRVSCEVSGFIFSK---AWMNWVRQ--APGKGLQ-WVG
D24672    ------------------------------------ISCKASGYTFTN---YGMNWVKQ--APGKGLK-WMG
PH0100    ------------------LvQLQQSGPVLv-KPGTSMKISCKTSGYSFTG---YTMSWVRQ--SHGKSLE-WIG
B27888    ------------------EvMLVESGGGLa-KPGGSLKLSCTTSGFTFSI---HAMSWVRQ--TPEKRLE-WVA
PL0160    ------------------QvQLQQSGPGLv-KPSQTLSLTCAISGDSVSSns-AAWNWIRQ--SPSRGLE-WLG
E28833    ------------------DvVMTQTPLSLpvSLGDQASISCRSSQSLVRSngnTYLHWYLQ--KPGQP--p-KL
D30539    ------------------EvKLVESGGGLv-QSGGSLRLSCATSGFTFSD---FYMEWVRQ--PPGKSLE-WIA
C30560    ------------------QvHLQQSGAELv-KPGASVKISCKASGYTFTS---YWMNWVKQ--RPGQGLE-WIG
AVMSX4    ------------------EvKLLESGGGLv-QPGGSLKLSCAASGFDFSR---YWMSWVRQ--APGKGLE-WIG
C30540    ------------------EvKLVESGGGLv-QPGGSLRLSCATSGFTFSD---FYMEWVRQ--PPGKRLE-WIA
PL0123    ------------------EvQLVESGGGLv-QPGGSLRLSCAASGFTFSS---YWMSWVRQ--APGKGLE-WVA
H36005    ------------------EvQLVESGGGLv-KPGGSLRLSCAASGFTFSN---AWMNWVRQ--APGKGLE-WVG
PH0097    ------------------DvKLVESGGGLv-KPGGSLKLSCAASGFTFSS---YIMSWVRQ--TPEKRLE-WVA
I37267    gsimg-------------vQLQQSGPELv-KPGASVKISCKTSGYTFTE---YTMHWVKQ--SHGKSLE-WIG
A25114    ------------------DvHLQESGPGLv-KPSQSLSLTCSVTGYSITRg--YNWNWIRR--FPGNKLE-WMG
D2HUWA    ------------------RlQLQESGPGLv-KPSETLSLTCIVSGGPIRRtg-YYWGWIRQ--PPGKGLE-WIG
A30539    ------------------EvKLVESGGGLv-QPGGSLRLSCATSGFTFSD---FYMEWVRQ--PPGKRLE-WIA
..........................................................*.....................................

          5         6         7         8         9         0                   1
PH0106    LI-YKV---SNR-FSGVPDRFSGSG--SGTDFTLKISRVEAEDLGIYFCSQ-------------------------
B27563    RI-DPNSGGTKY-NEKFKNKATLTINKPSNTAYMQLSSLTSDDSAVYYCARGYDYSYY---------AMDYWGQGT
MHMS76    EIrLKSGYATHY-AESVKGRFTISRDDSKSSVYLQMNNLRAEDTGIYYCTRPGV-------------PDYWGQGT
D28035    QIkNKVDGGTIDYAAPVKGRFIISRDDSKSTVYLQMNRLKIEDTAVYYCVGNYTGT-----------VDYWGQGT
D24672    WI-NTYTGEPTY-ADDFKGRFAFSLETSASTAYLQINNLKNEDTATYFCARGSSYDYY---------AMDYWGQGT
PH0100    LI-IPSNGGTNY-NQKFKDKASLTVDKSSSTAYMELLSLTSEDSAVYYCARPSYYGSRnyy------AMDYWGQGT
B27888    AI-SSGGSYTFY-PDSVKGRFTISRDNAKNTLYLQINSLRSEDTAIYYCAREEGLRLDdy------AMDYWGQGT
PL0160    RT-YYRSKWYNDYAVSVKSRITINPDTSKNQFSLQLNSVTPEDTAVYYCARELGDA-----------FDIWGQGT
E28833    LI-YKV---SNR-VSGVPDRFSGSG--SGTDFTLKISRVEAEDLGVYFCSQSTHV--------------------
D30539    ASrNEANDYTTEYSASVKGRFIVSRDTSQSILYLQMIALRAEDTAIYYCSRDYYGSSYw--------YFDVWGAGT
C30560    EI-DPSNSYTNN-NQKFKNKATLTVDKSSNTAYMQLSSLTSEDSAVYYCARWGTGSSWg--------WFAYWGQGT
AVMSX4    EI-NPDSSTINY-TPSLKDKFIISRDNAKNTLYLQMSKVRSEDTALYYCARLHYYGY----------AAYWGQGT
C30540    ASrNKAHDYTTEYSASVKGRFIVSRDTSQSILYLQMNALRAEDTAIYYCARDADYGSSshw------YFDVWGAGT
PL0123    NI-KQDGSEKYY-VDSVKGRFTISRDNAKNSLYLQMNSLRAEDTAVYYCAR-------------------------
H36005    RIkSKTDGGTTDYAAPVKGRFTISRDDSKNTLYLQMNSLKTEDTAVYYCTTDRGGSSQ----------GDYWGQGT
PH0097    TI-SSGGRYTYY-SDSVKGRFTISRDNAKNTLYLQMSSLRSEDTAMYYSTASGDS------------FDYWGQGT
I37267    GI-NPNNGGTSY-NQKFKGKATLTVDKSSSTAYMELRSLTSEDSAVYYCARRGLTTVVaksy-----YFDYWGQGT
A25114    YI-NYDGS-NNY-NPSLKNRISVTRDTSKNQFFLKMNSVTTEDTATYYCARLIPFSDGyyedyy--AMDYWGQGT
D2HUWA    GV-YYTGS-IYY-NPSLRGRVTISVDTSRNQFSLNLRSMSAADTAMYYCARGNPPPYYdigtgsddGIDVWGQGT
A30539    ASrNKANDYTTEYSASVKGRFIVSRDTSQSILYLQMNALRAEDTAIYYCARDYYGSSYvw------YFDVWGAGT
..........................................................*.....................................

            2
PH0106    ------tthvpptfgggtkleikr-
B27563    SVTVSS-------------------
MHMS76    TLTVSS-------------------
D28035    LVTVSS-------------------
D24672    SVTVSS-------------------
PH0100    SVTVSSak-----------------
B27888    SVTVS--------------------
PL0160    MVTVSS-------------------
E28833    -------------------------
D30539    TVTVSS-------------------
C30560    LVTVSA-------------------
AVMSX4    LVTVSAe------------------
C30540    TVTVSS-------------------
PL0123    -------------------------
H36005    LVTVSS-------------------
PH0097    TLTVSSak-----------------
I37267    TLTVSS-------------------
A25114    -------------------------
D2HUWA    TVHVSS-------------------
A30539    TVTVSS-------------------
.....................................
```

G-Protein-Coupled Receptors (GPCR's)

- 145 sequences with minimum length 310, average length 430 and maximal length 764.
- Model used here has been trained with 143 sequences (3 sequences contained undefined symbols) using the Viterbi best path approximation
- Multiple alignment very similar to the one in Probst et al. (1992); the 7 transmembrane domains (alpha helices) and the conserved residues are clearly visible.


Main State Entropy Values
0.0
1.5
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210
0.0
1.5
3.0
220 230 240 250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430
Entropy Distribution
0
40
80
0.0
0.5
1.0
1.5
2.0
2.5
3.0


Distance to model
0
1000
2000
3000
4000
5000
6000
0
500
1000
1500
2000
Sequence length
x Random Sequences
+ GCR (training)
Swissprot (validation)

Analytical Expressions

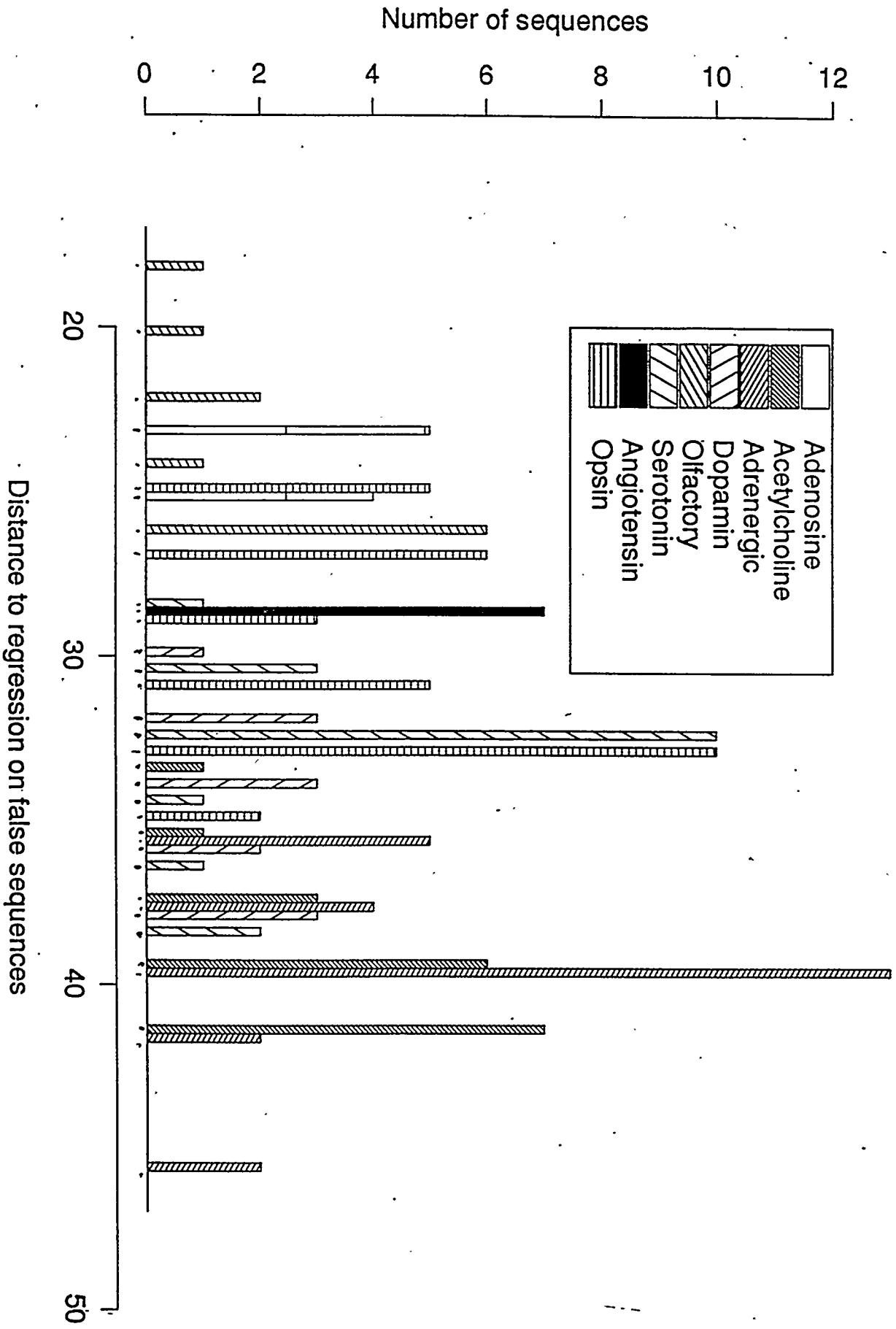
- Slope of scores of random sequences with fixed source $P = (p_x)$ (determined by optimal insert state k):

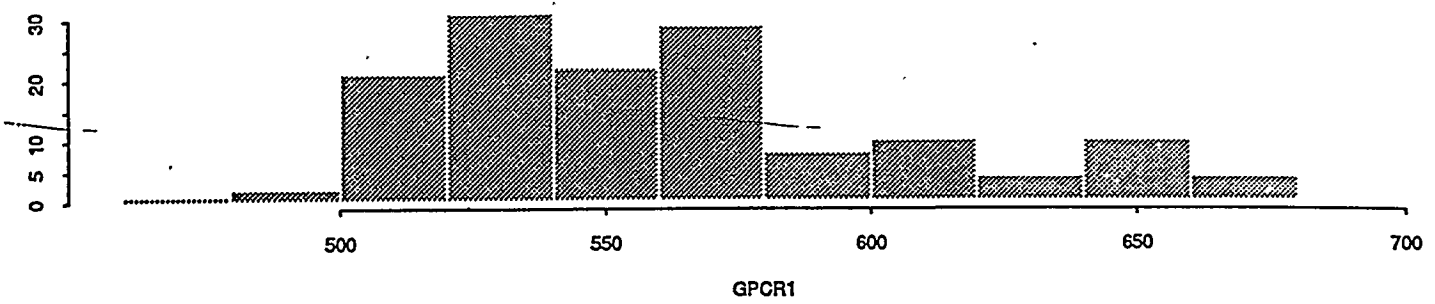
$$\min_k[-\ln t_{kk} - \sum_x p_x \ln e_{kx}]$$

- Variance of Scores:

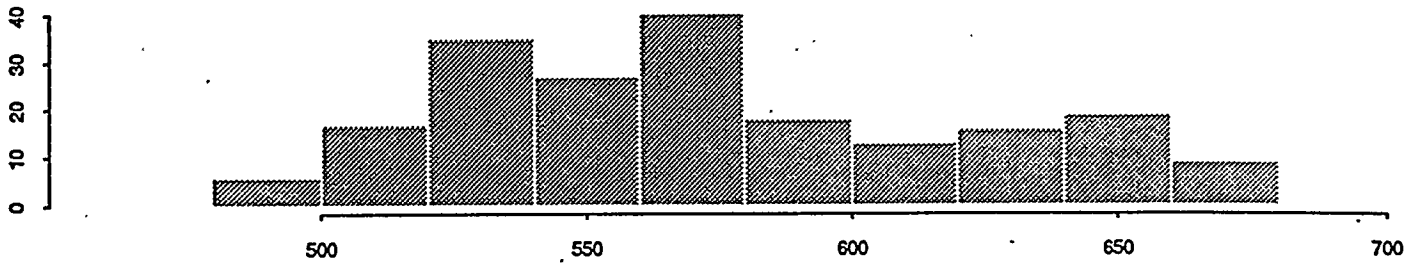
$$l[E_P(\ln^2 e_{kx}) - E_P(\ln e_{kx})] = l\mathrm{Var}_P[\ln e_{kx}]$$

- Central Limit Theorem for large length l (scores are normally distributed)

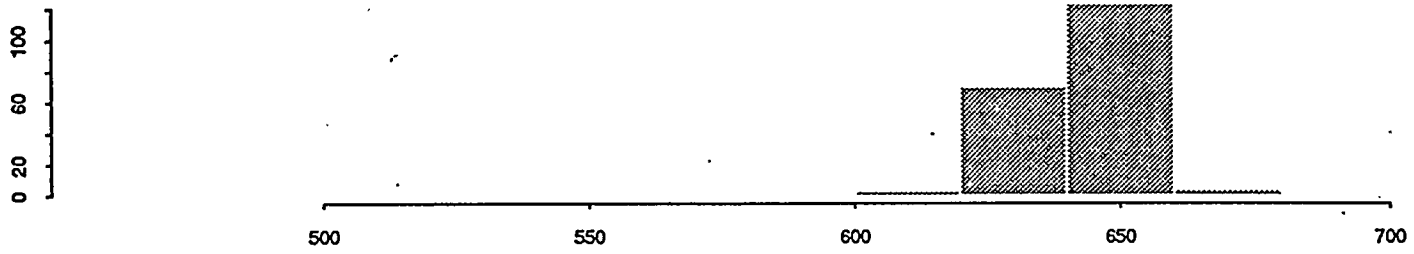

Number of sequences
0
2
4
6
8
10
12
Distance to regression on false sequences
20
30
40
50
Adenosine
Acetylcholine
Adrenergic
Dopamin
Olfactory
Serotonin
Angiotensin
Opsin



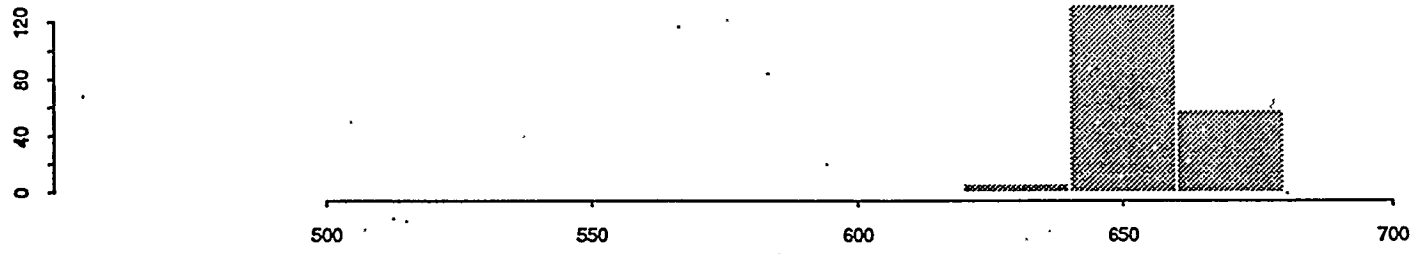
GPCR1



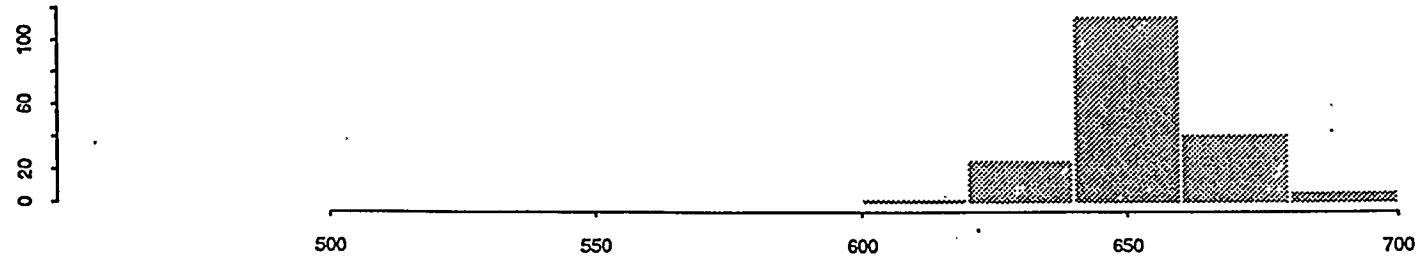
GPCR2



RAN



KIN



SWISS

Limitations of HMMs

- Large Number of Parameters/Small Families (Overfitting)
- Long Range Correlations/First Order Markov Model (Underfitting)

Caveats

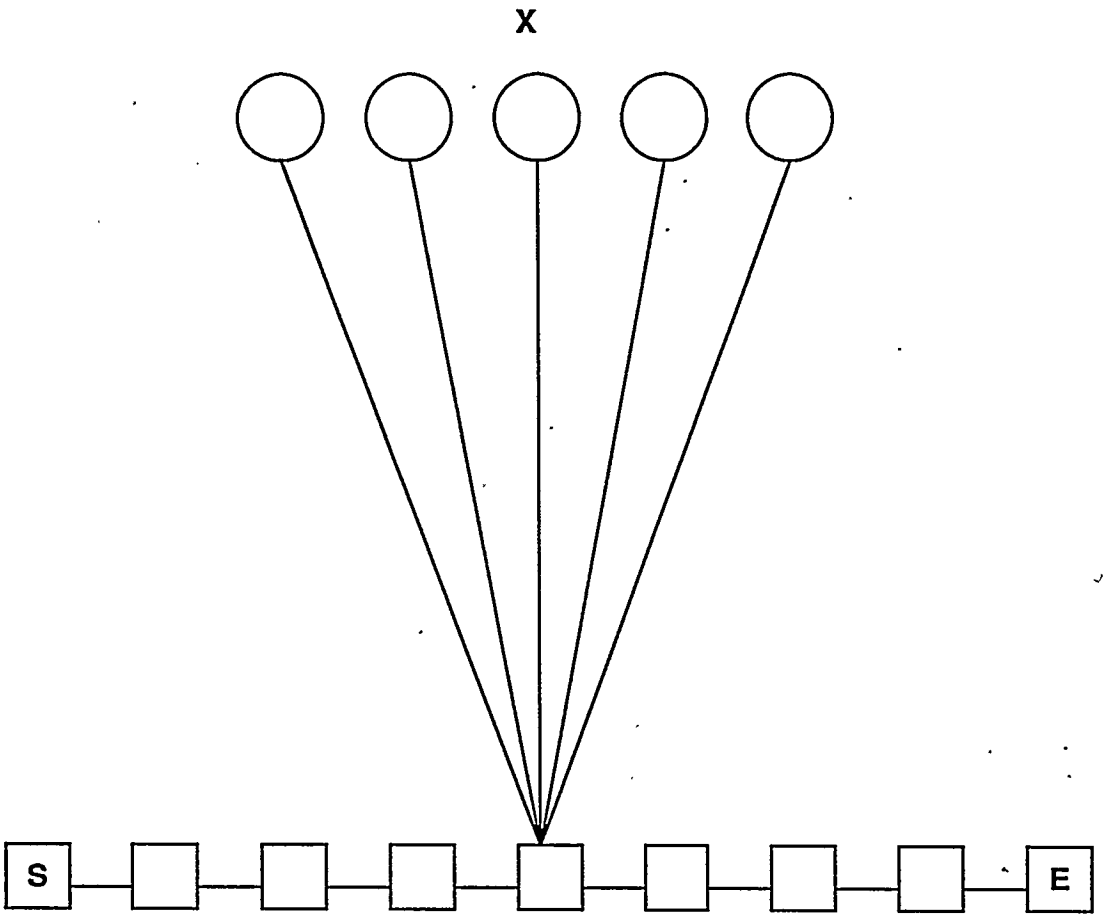
- HMMs seem to be well constrained.
- HMMs have been trained on as little as 35 sequences. HMMs could be trained on 2 sequences only.
- Regularization.
- Constant long range correlations are easily captured by HMMs. Simple variable long range correlations can also be captured ($X \rightarrow Y$ and $X' \rightarrow Y'$) in some cases.
- Variable long range correlations are rarely the problem in data base mining.
- Model hierarchy. HMMs are equivalent to good multiple alignments. HMMs are equivalent to factorial distributions over the space of sequences. Within this class, well trained HMMs are optimal.
- For long range correlations (i.e. when the best factorial approximation is not good enough), and if one is to stay with HMMs, one must have multiple HMMs or a way of modulating a single HMM as a function of input sequence (classification).

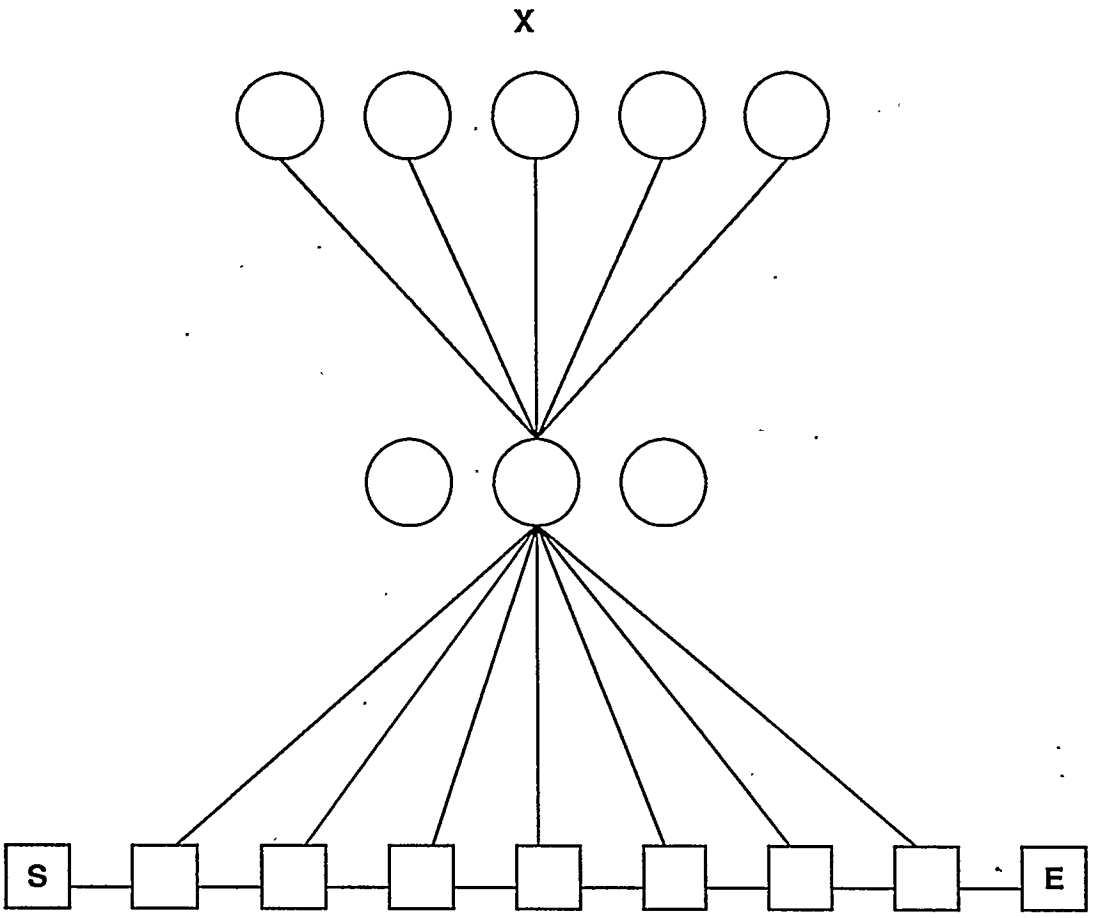
Normalized Exponential Representation

$$e_{iX} = \frac{e^{v_{iX}}}{\sum_Y e^{v_{iY}}}$$

$$t_{ij} = \frac{e^{w_{ij}}}{\sum_k e^{w_{ik}}}$$

- No limitations on v's or w's during learning
- Avoid 0 emission or transition probabilities
- Equivalent to having one or several independent little softmax neural networks for each HMM state


X
S
E


X
S
E

Hybrid Hierarchical Modeling

- Two General Ideas: compute and modulate the parameters of a probabilistic model via another model, typically a neural network.
- First derived in the case of HMMs (HMM/NN hybrid architectures) for protein modeling applications.

General Setting

- Data D.
- Class of Probabilistic Models $M(\theta)$, parametrised by θ.
- Goal: approximate $P(\theta|D)$ and possibly its mode.
- Problem 1: Model is too complex, resulting in overfitting.
- Problem 2: Model is too simple, resulting in underfitting.

Solution

- Problem 1: Reparametrise the model in the form:

$$\theta = F(w)$$

- Problem 2: Change model class or use multiple models with:

$$\theta = F(I)$$

where I is some input or context.

- In the most general case, one can combine the two together to have:

$$\theta = F(w, I)$$

- Of course, F can be computed by another model, for instance a Neural Network (universal approximation properties). This yields hierarchical hybrid M/NN architectures. These steps can also be iterated hierachically several times.

- Unified training algorithms (likelihood is "back-propagated").

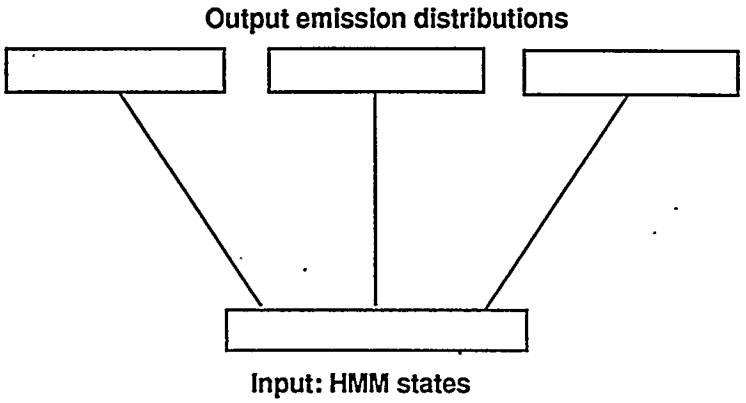
Hybrid HMM/NN Architectures

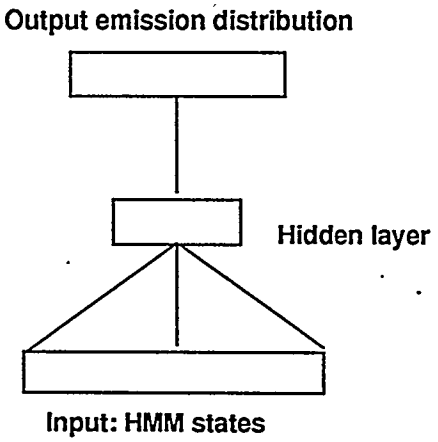
- In the case of a single HMM model, compute the transition and/or emission parameters using one or several Neural Networks.
- HMM states can be partitioned into different groups. For instance main states/insert states or hydrophobic/hydrophilic. regions.
- Output biases can be used to initialize to average family composition.
- By varying the number of hidden units, one can get backbone regularization, parameters reduction, and automatic fit to evolving sequence data bases.
- Great flexibility: different possible NN architecture and priors or regularizers (multiple layers, sparse connectivity, weight sharing, weight decay, mixtures,...).
- Analysis of weights and hidden units may reveal interesting patterns.
- Integrated learning algorithms that blend HMM learning with NN back-propagation.
- Training can be faster (less parameters).

- Applicable to other domains (HMMs with continuous emissions or transit parameters become easy).

- Different from previous hybrid architectures, where NNs are often used in early stages (preprocessing, feature extraction) and HMMs in later stages (word/language models).

- Most simple HMM/NN architectures do not solve the variable long range correlation problems, since ultimately they yield a single HMM. However, it is possible to have additional input, and/or hidden and/or output units to modulate a single HMM or to produce multiple different HMMs. Related to self-organisation and classification.


Output emission distributions
Input: HMM states


Output emission distribution
Hidden layer
Input: HMM states

Multiple HMM Models

- Mixtures of HMM Experts (n HMMs $M_1, ..., M_n$):

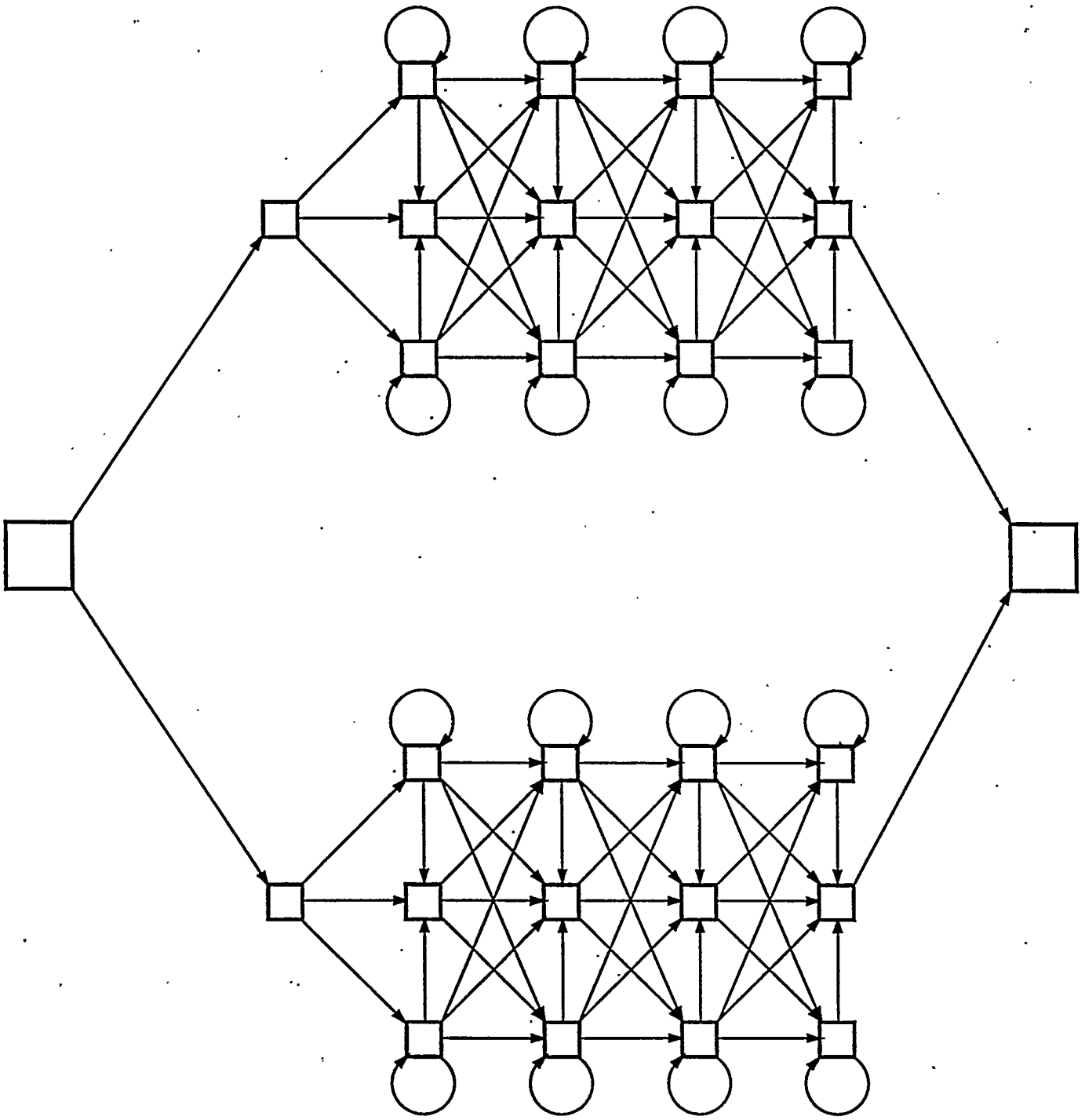
$$P(O) = \sum_{i=1}^{n} \lambda_i P_{M_i}(O)$$

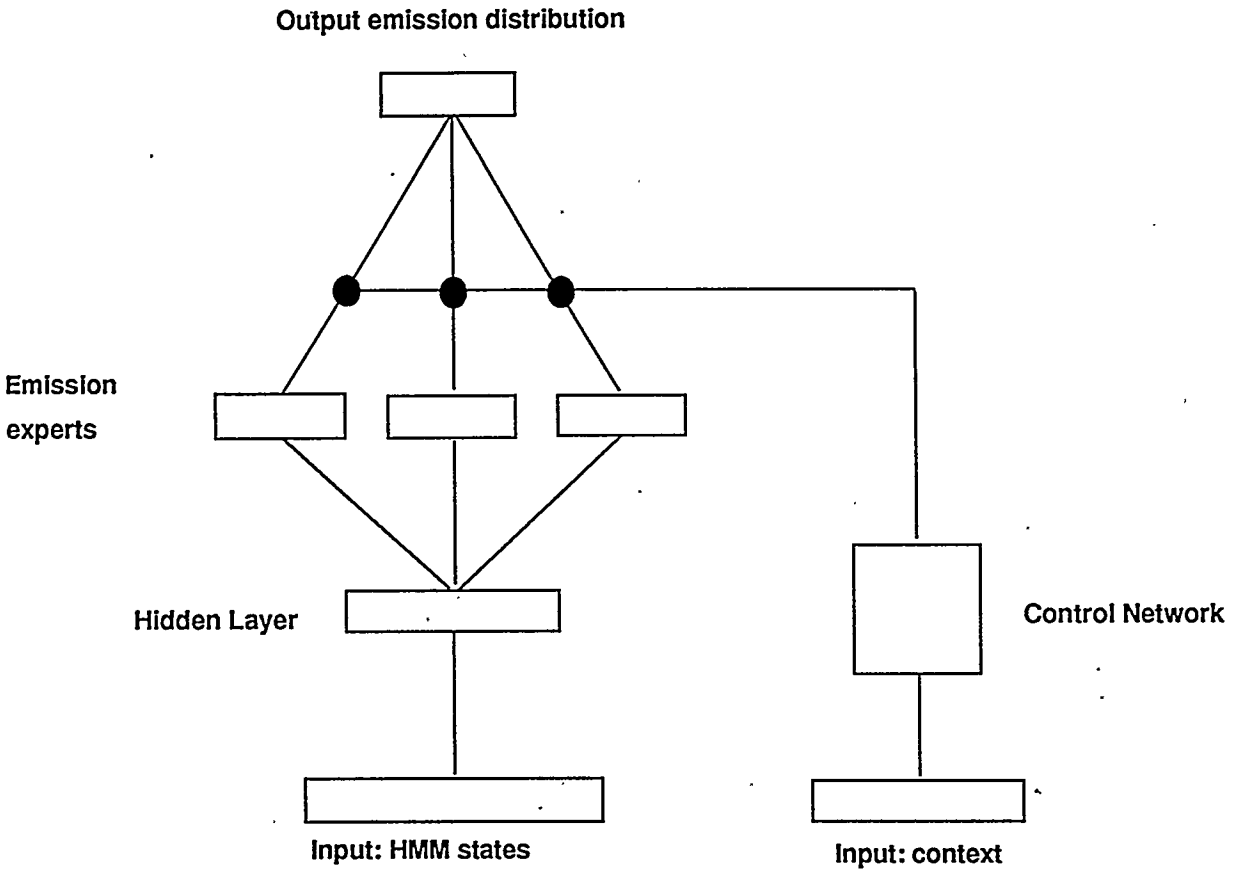
- Mixtures of Emission Experts:

$$P(i, X, I) = \sum_{j=1}^{n} \lambda_j(i, X, I) P_j(i, X, I)$$

$$P(i, I) = \sum_{j=1}^{n} \lambda_j(i, I) P_j(i, I)$$

$$P(i, I) = \sum_{j=1}^{n} \lambda_j(I) P_j(i)$$

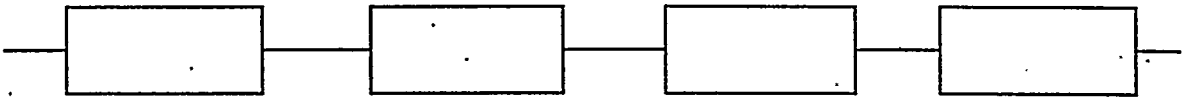



Output emission distribution
Emission
experts
Hidden Layer
Control Network
Input: HMM states
Input: context

Gene Modeling experiments

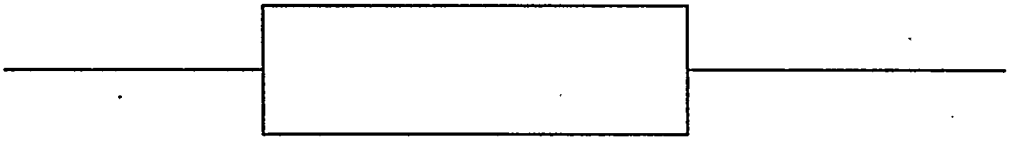
- Several experiments have been conducted using training sets consisting of exons only, flanked exons, flanked acceptor sites, flanked donor sites, reversed flanked exons and random sequences of similar composition.

STRUCTURE OF EUCARYOTIC GENES



exon intron

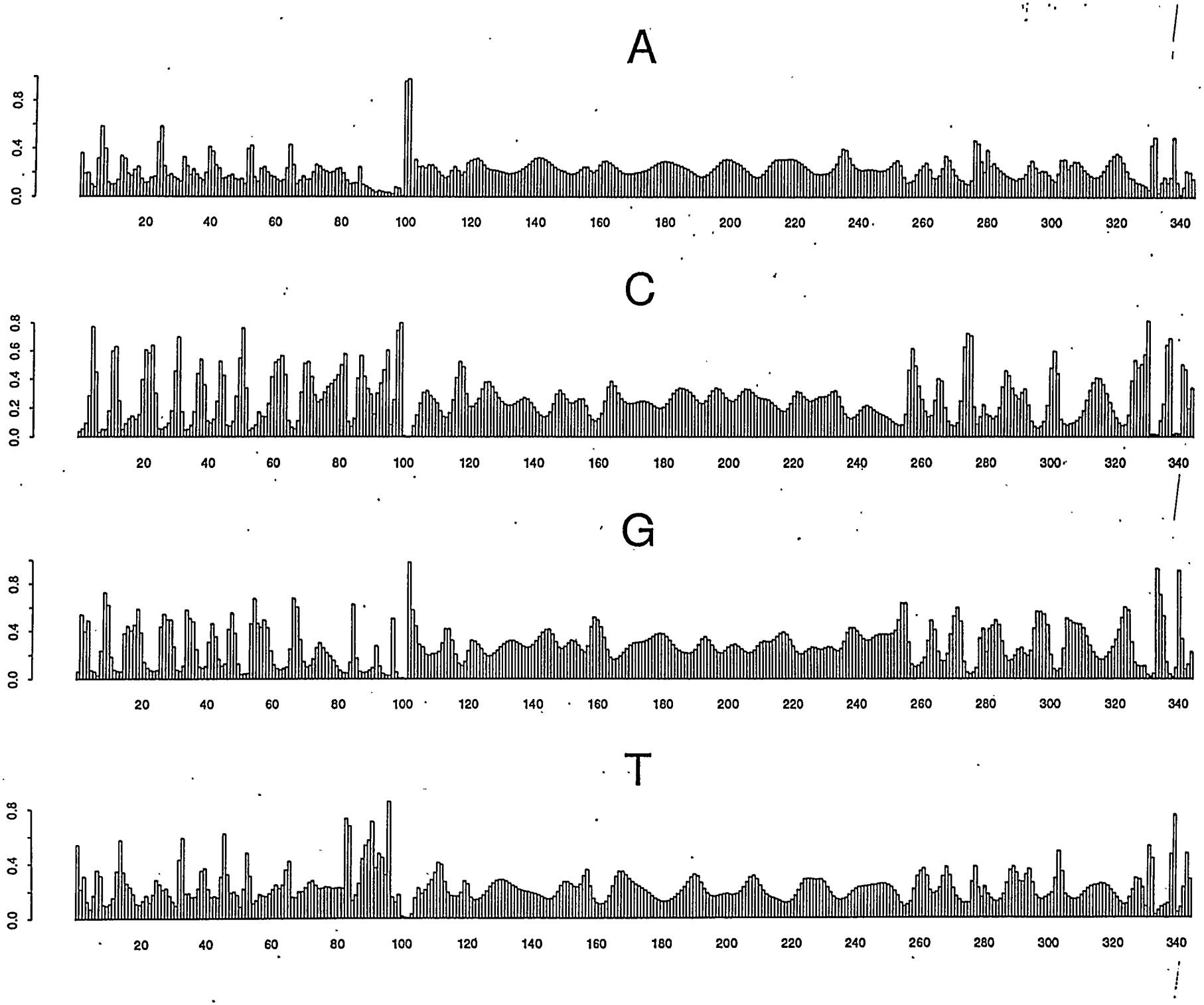
EXON

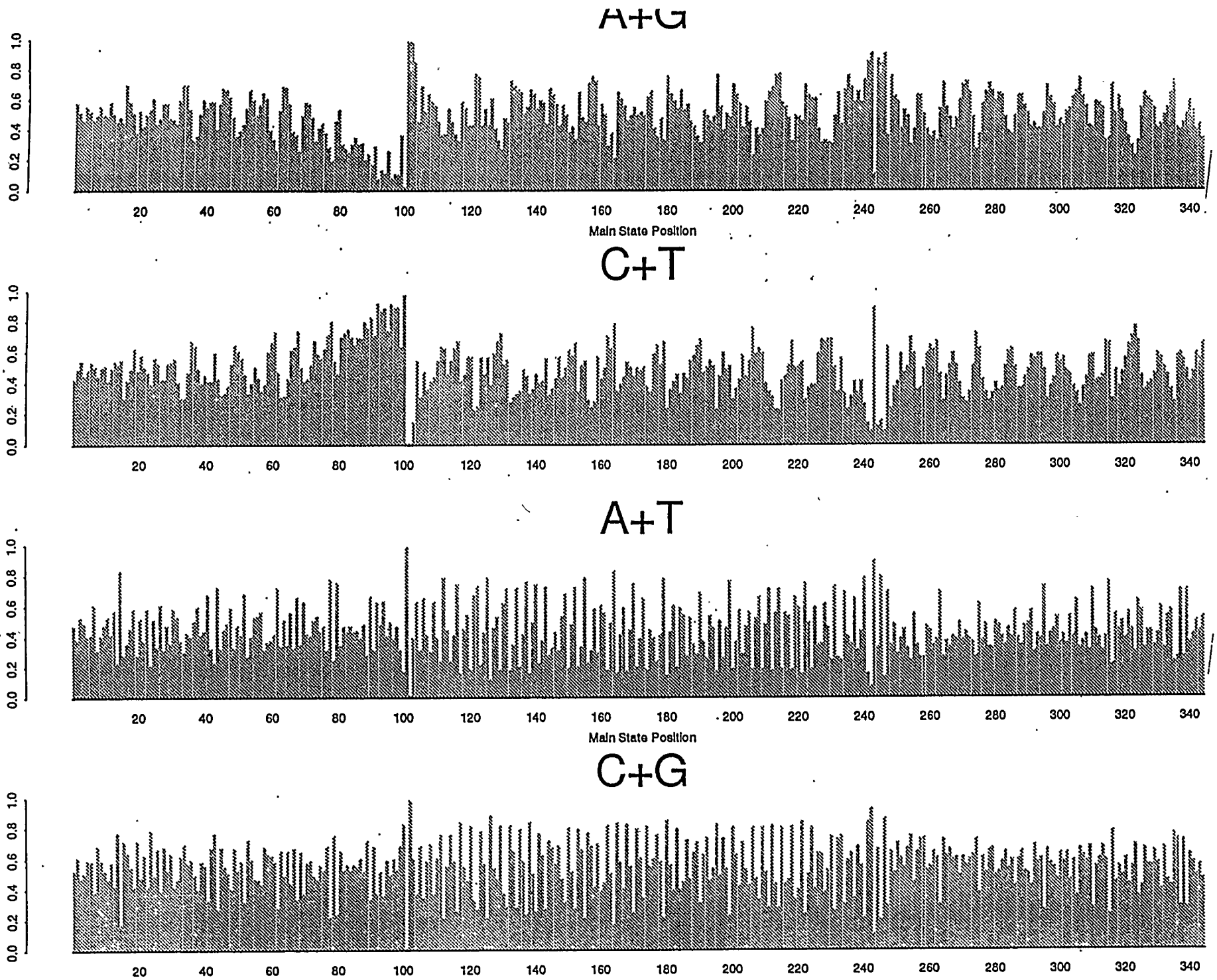


3' splice site
acceptor site

5' splice site
donor site

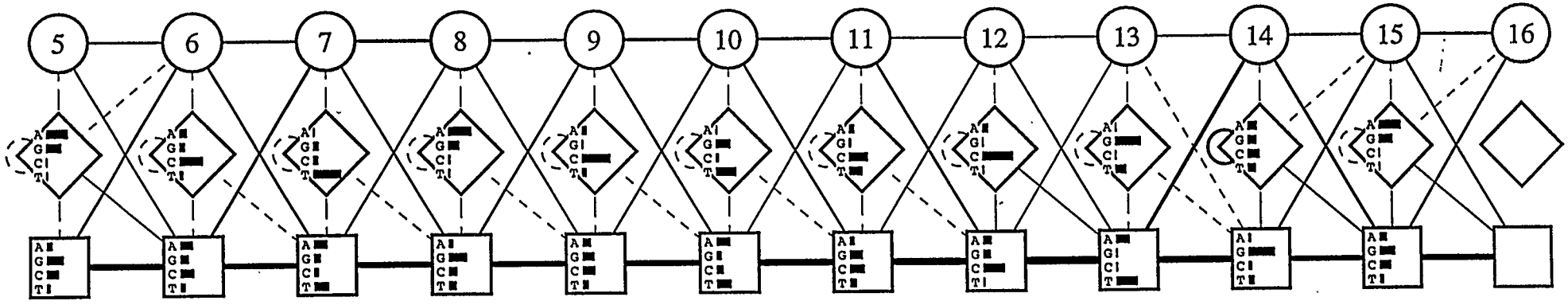
CONSENSUS SEQUENCES


A
0.0
0.4
0.8
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
C
0.0
0.2
0.4
0.6
0.8
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
G
0.0
0.4
0.8
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
T
0.0
0.4
0.8
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340


A+G
C+T
A+T
C+G
0.0 0.2 0.4 0.6 0.8 1.0
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300 320 340
Main State Position

Model Segment

This is the repeated segment of the model after training



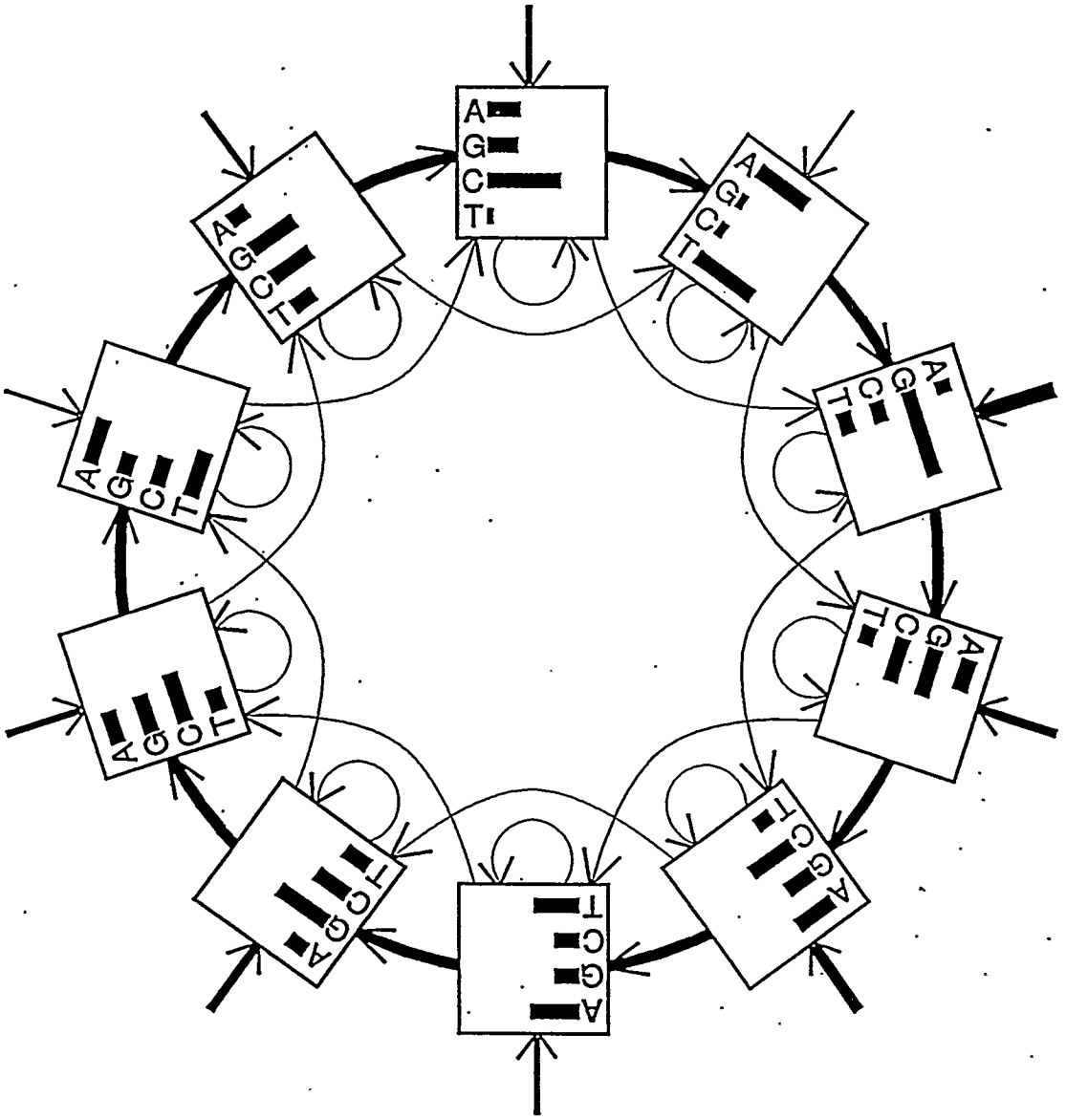
Notice the distinct pattern [^T][AT]G in states 12-14.

Compare with entropy plot.

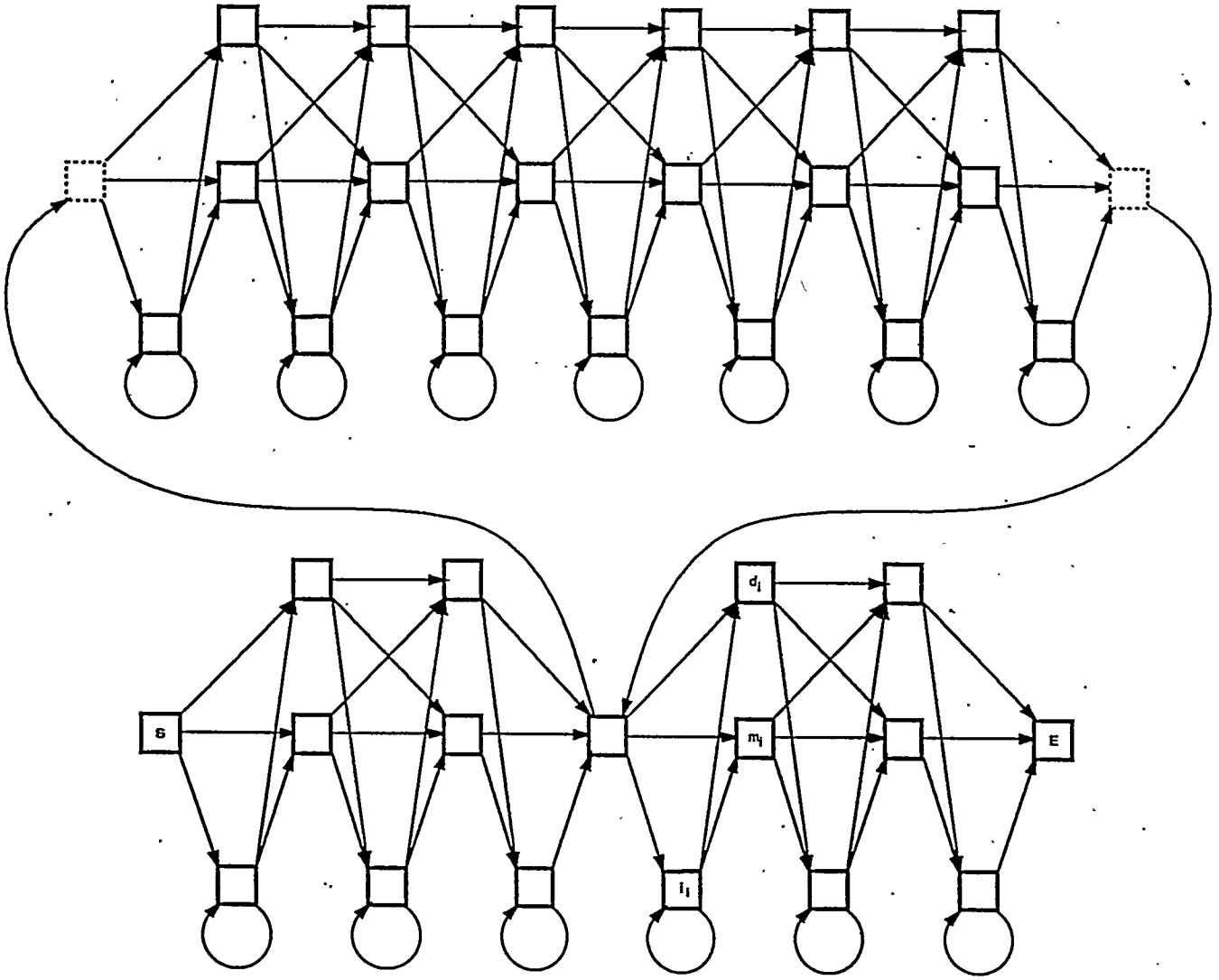
Model Scores

	NLL, training	NLL, testing	# parameters
Standard model with randomized seqs	203.2	200.3	2550
Standard model with real sequences	198.8	196.4	2550
Tied model with real sequences	198.6	195.6	340

NLL = negative log likelihood.



Model trained on 500 random exons. Thicknes of arrows from 'outside' show the probability of starting in the state.


S
d_j
m_j
i_j
E

STOCHASTIC GRAMMAR

- A formal grammar is entirely defined by an alphabet, a set of non-terminal sysmbols, and a set of production rules.
- Chomsky Hierarchy (Regular Grammars, Context-Free Grammars).
- Regular Grammars (RG):

$$x \rightarrow Ax \quad \text{and} \quad x \rightarrow A$$

- Context-Free Grammars (CFG):

$$x \rightarrow xx \quad \text{and} \quad x \rightarrow AxA$$

- Other Grammars: Graph Grammars (GG)
- Stochastic Grammars (SRG, SCFG, SGG): Probabilistic production rules.

RNA Modeling

- Watson-Crick base pairing:

$$x \rightarrow AxU$$

$$x \rightarrow UxA$$

$$x \rightarrow CxG$$

$$x \rightarrow GxC$$

- Examples.