CSE527 Notes, Lecture 18 11/29/04

David W Richardson

December 2, 2004

1 RNA Sequence Analysis Using Covariance Models

This lecture looked at the paper by Sean Eddy and Richard Durbin entitled ”RNA Sequence Analysis Using Covariance Models.” The paper appeared in *Nucleic Acids Research* in 1994. The paper describes the construction of a probabilistic model for RNA families (mainly tRNA). This model is known as a ”Covariance Model” (CM) and is roughly an analogous model to Stochastic Context-Free Grammars. CMs are generalizations of profile HMMs where simultaneous emissions of symbols represent base-pairs in the RNA secondary structure. The paper also describes algorithms for training and searching databases with these profile CMs. Training can be performed on either aligned or unaligned sequences. The training algorithm complements the Nussimov/Zuker-style folding algorithm we’ve seen in previous lectures.

The main results of the paper are as follows. Given a CM for a family of tRNAs, searching for more tRNA members is very accurate. The model construction is comparable to, although not quite as good as, models constructed by human experts. Their results also show quantitative information on the importance of pseudoknots and other tertiary features. It turns out that such structural features are not that important for effective alignment and searching.

The basic idea behind a CM is straightforward. Suppose we are given a RNA sequence and its secondary structure. A CM models this sequence and its structure by using a tree-like state machine. There are states that correspond to emitting unparied nucleotides, and states for emitting paired nucleotides. Special ”bifurcation” states represent junctions between multiple helices in the secondary structure. Given such a tree, you simply walk the tree in depth-first order to get the RNA’s primary sequence. Paired nucleotides in the secondary structure correspond to pairs emitted by the same state in the tree. This idea is analogous to emitting nested pairs in a context-free grammar. An example of such a binary tree is shown in figure 1.

However, in general we want to model a family of RNA sequences and their structure. To do this, we can assign probabilities of emitting letters and pairs at states, and probabilities of inserting and deleting letters. This results in a full model which is a generalization of HMMs. Instead of a simple HMM, we now have a ”guide tree” where nodes in the tree fill the same role as nodes in a HMM. There are many node types possible in the CM. Some of these types are listed below:

1. left singlet node - this node corresponds to the top 2 ”A o -” nodes in the binary tree of figure 1. This node has 3 possible states with emission probabilities on letters: MATL (match left), INSL (insert left), and DEL (delete). These states correspond to the various ways that a single letter could be emitted with no ”right” match from this node. This is why it is called a left singlet node.
2. right singlet node - this is the dual of a left singlet node.
3. bifurcation node - this node corresponds to a junction between multiple helices in the secondary structure. The node has 2 children nodes, each of which are transitioned to with probability 1.

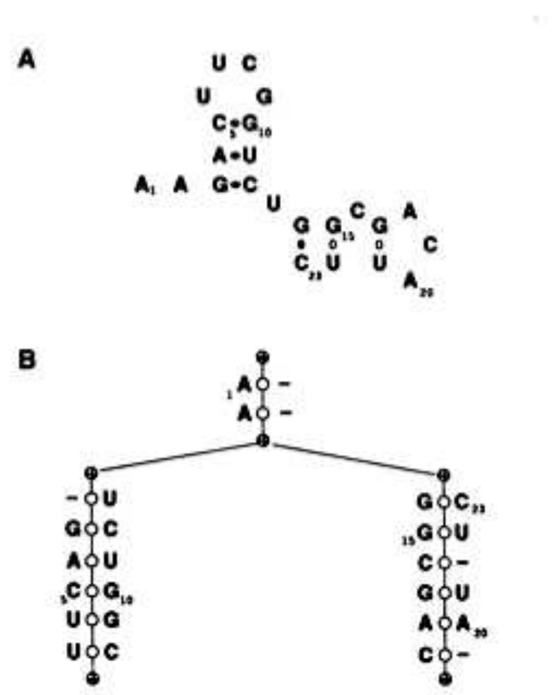

Figure 1. (A) An example RNA structure. (B) An ordered binary tree description of that RNA structure. The tree includes dummy begin, end, and branching (bifurcation) nodes in addition to pairwise and singlet nodes that account for sequence.

Figure 1: An example RNA structure along with its ordered binary tree description from the original paper.

4. pairwise node - this node, along with the bifurcation node, encompasses the key reason why CMs are more general than HMMs. A pairwise node has a special "match" state which emits 16 possible base-pairs, each with some probability. This corresponds to a base pairing in the RNA secondary structure. There are also MATL and MATR states corresponding to when part of the base-pair match is missing. Other states such as INSL and DEL are also possible in this node.

5. begin and end nodes - these are special nodes that mark the beginning and end of a helix.

An example of a CM showing these nodes and states is shown in figure 2. This figure is also directly lifted from the paper.

To use these models, we have to set up a specific model structure and assign probabilities to the model's parameters. Doing so defines a probability distribution on strings and secondary structures modeled by the CM.

The paper describes how to train a CM to a set of training sequences. The training algorithm starts with a trusted multiple alignment of RNA sequences. It then estimates the parameters for the CM using this

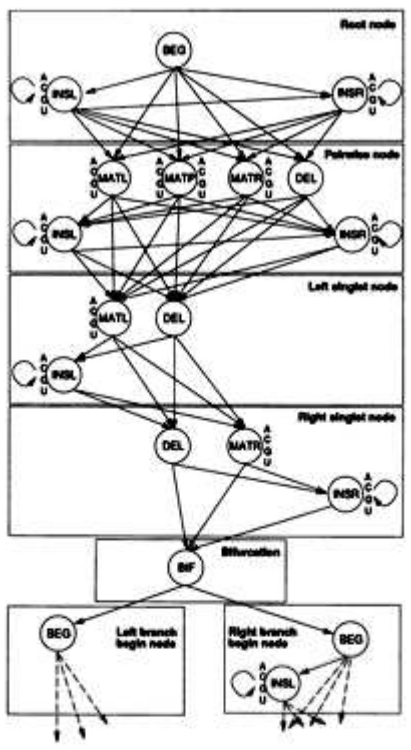

Figure 2. The seven distinct types of nodes from Figure 1 are broken up into states as shown. There are seven different kinds of states in all (bifurcation BIF, begin BEG, insert-left INSL, insert-right INSR, match-pairwise MATP, match-left MATL, match-right MATR, and delete DEL). State transition probabilities are indicated by arrows. States which have singlet or pairwise symbol emission probabilities are indicated by 'ACGU' beside the state.

Figure 2: An example CM with labeled nodes and states from the original paper.

alignment. This is done by essentially looking at the viterbi paths through the CM for each sequence to give the probability of the model's transitions. If no alignment is provided, the training algorithm performs outer EM iterations to determine the model's parameters. This consists of starting with a random alignment, then estimating the model's parameters, then re-aligning the sequences to this new model, and repeating this process until convergence. It turns out that this iterative EM process works quite well for unaligned training sequences.

But how do you define the original structure of the model? While this problem remains a bit of a black art (as it is with HMMs), the authors propose the following solution that uses the notion of *mutual information*. Mutual information calculates the amount of information we gain about the letter in column i of the alignment if we know the identify of the letter in column j. If f_{xi} is the frequency of nucleotide x in column i of the alignment, then we define the mutual information of column i and j as:

$$M_{ij} = \sum_{xi,xj} f_{xi,xj} * log_2 \frac{f_{xi,xj}}{f_{xi} * f_{xj}} \tag{1}$$

where $f_{xi,xj}$ is the joint (pairwise) frequency of one of the sixteen possible base pairs observed in columns i and j. Basically, M_{ij} measures how much the joint frequency distribution deviates from the distribution expected if the two columns i and j varied independently. For RNA, M_{ij} varies between 0 and 2 bits. M_{ij} is maximal if columns i and j appear completely random when observed individually, but are perfectly correlated.

As we'll see in the next lecture, this mutual information scoring can be used to create a guess of the CM structure using a dynamic programming algorithm.

The paper describes how to use a CM for performing multiple alignments. The basic idea is to use a variant of the ”inside algorithm” analogous to the forward algorithm we saw for HMMs. The details are slighty mucky, but straightforward enough to understand. See page 2081 of the paper for more information.

The paper also presents an algorithm for using a profile CM to search databases for homologous RNAs. This basically consists of determining the probability that a given string is generated by the CM (as we did with HMMs). Given a model, the algorithm scans a genome, calculating the viterbi alignment of the substring to the model. If the probability of the viterbi path through the CM compared to the probability of the path through a null model exceeds some threshold, the sequence in question is a ”hit” and is considered to be a homologous RNA. The bonus feature of this search algorithm is that since the viterbi calculation also gives you the path through the model, you can deduce the secondary structure of this newly found RNA based on what states were match states in the model. This provides for RNA secondary structure prediction for new sequences.

So does this search and alignment technique work? The answer is yes, it works pretty well. While alignment results compared to ClustalV where suboptimal, the authors show a strong comparison between the ability of trained CMs and the program tRNASCAN to find tRNAs in a test-set. It should be noted that in 1994, the year this paper was published, tRNASCAn was the champion of tRNA prediction programs. The best-trained CM built from a trusted alignment of 1415 tRNA sequences outperformed tRNASCAN with respect to both sensitivity and specificity. In fact, the best-trained CM yielded better than 99.8% true positives with less than .2 false positives per Megabase.