Lecture 14

Using Interpolated Context Models to Find Genes

February 22, 2000
Notes: Gretta Bartels

14.1. Problems with Markov Chains for Finding Genes

The Markov chain is an effective model for finding genes, as described in Section 13.3. However, such a tool is not 100% accurate. The problem is that a kth order Markov chain requires 4^{k+1} probabilities in each of three reading frame positions. There is a tension between needing k large to produce a good gene model, and needing k small because there is insufficient data in the training set. For example, when $k = 5$ as in GeneMark, we need $3 \times 4^6 \approx$ 12,000 6-mers to build the model. Each of these 12,000 6-mers must occur often enough in the training set to support a statistically reliable sample.

Some 5-mers are too infrequent in microbial training sets, yet some 8-mers are frequent enough to be statistically reliable. Section 14.2 describes a gene finder that was designed to have the flexibility to deal with these extremes.

14.2. Glimmer

Glimmer [1, 2] is a gene prediction tool that uses a model somewhat more general than a Markov chain. In particular, Glimmer 2.0 [1] uses what the authors call the *interpolated context model* (ICM).

The *context* of a particular residue consists of the k characters immediately preceding it. A typical context size might be $k = 12$. For context $C = b_1 b_2 \ldots b_k$, the interpolated context model assigns a probability distribution for b_{k+1}, using only as many residues from C as the training data supports. Furthermore, those residues need not be consecutive in the context.

Glimmer has three phases for finding genes: training, identification, and resolving overlaps.

14.2.1. Training Phase

As in Sections 12.1 and 13.3, Glimmer uses long ORFs and sequences similar to known genes from other organisms as training data for the model. For each of the three reading frame positions, consider all $(k+1)$-mers that end in that reading frame position. Let X_i be a random variable whose distribution is given by the frequencies of the residues in position i of these $(k+1)$-mers.

In general, we will not have sufficient training data to use all k residues from this context to predict the $(k+1)$st residue b_{k+1}. Our goal is to determine which variable X_j has most correlation with X_{k+1}, and use

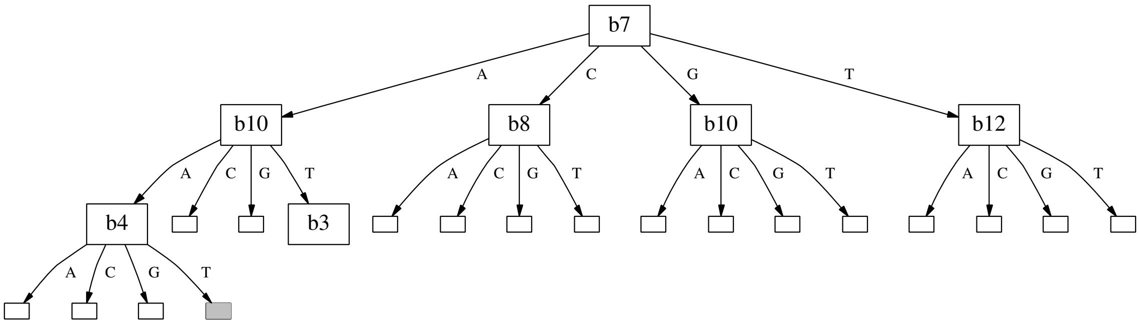

Figure 14.1: Interpolated context model tree.

it to predict b_{k+1}. The mutual information of Definition 11.3 is used to make this determination. We first find the maximum among the mutual information values

$$I(X_1; X_{k+1}), I(X_2; X_{k+1}), \ldots, I(X_k; X_{k+1}).$$

Suppose X_j maximizes this mutual information. Then the jth residue b_j will be used first to predict the value of b_{k+1}.

To determine which position has the next highest correlation, we do not simply take the second highest mutual information from the list above. The identity of the next position instead depends on the value of the first selected residue b_j. Glimmer builds a tree of influences on X_{k+1}, illustrated in Figure 14.1, as follows. The $(k+1)$-mers from this reading frame position are partitioned into four subsets according to the residue b_j. Then we repeat the mutual information calculation above for each of these subsets. In the example of Figure 14.1, X_7 was found to have the greatest mutual information with X_{k+1}, and the $(k+1)$-mers were partitioned according to the value of residue b_7. For those with b_7 = A, X_{10} was found to have the greatest mutual information with X_{k+1}, and they were further partitioned into four subsets according to the value of residue b_{10}.

A branch is terminated when the remaining subset of $(k+1)$-mers becomes too small to support further partitioning. Each such leaf of the tree is labeled with the probability distribution of X_{k+1}, given the residue values along the path from the root to that leaf. For example, in the tree shown in Figure 14.1, the leaf shaded gray would be labeled with the distribution

$$Pr(X_{k+1} = x \mid X_7 = A \ \&\ X_{10} = A \ \&\ X_4 = T).$$

Note how this tree generalizes the notion of Markov chain given in Definition 13.1.

14.2.2. Identification Phase

Once the interpolated context model has been trained, the identification phase begins. Given a candidate ORF x, compute the probability of each residue b_{k+1} of x by following the appropriate path in the tree that corresponds to b_{k+1}'s position in the reading frame. Here is a rough algorithm for the identification phase:

1. Pick the tree for the correct reading frame position.

2. Number the residues in the context of b_{k+1}, so that the previous residue is b_k, the one before that b_{k-1}, and so on.

Figure 14.2: Overlapping gene candidates. The arrow points to the $3'$ end of the sequence.

3. Trace down the tree, selecting the edges according to the particular residues in the sequence, until reaching a leaf. Read the probability $\Pr(X_{k+1} = b_{k+1} \mid \ldots)$ from that leaf.

4. Shift to the next residue in the sequence and repeat.

The product of these probabilities (times the probability of the first k residues of x, as in Section 13.2) yields the probability that x was generated by this interpolated context model. To take the length of x into account, combine these probabilities by adding their logarithms, and normalize by dividing by the length of x. Select x for further investigation if this score is above some predetermined threshold.

Note: Glimmer actually stores a probability distribution for X_{k+1} at every node of the tree, not just the leaves, and uses a combination of the distributions along a branch to predict b_{k+1}. This is where the modifier "interpolated" would enter, but we will not discuss this added complication. See the papers [1, 2] for details.

14.2.3. Resolving Overlap

In the final phase, Glimmer resolves candidate gene sequences that overlap. The main flexibility in resolving overlap is the possibility of shortening an ORF by choosing a different start codon. Suppose sequences A and B overlap, and that A has the greater score. There are four possibilities for the way they overlap, as depicted in Figure 14.2. Each arrow in that figure points to the $3'$ end of the sequence.

(a) Suppose A and B overlap as shown in Figure 14.2(a). Moving either start codon cannot eliminate the overlap in this case. If A is significantly longer than B, then reject B. Otherwise, accept both A and B with an annotation that they have a suspicious overlap.

(b) Suppose A and B overlap as shown in Figure 14.2(b). If moving B's start codon resolves the overlap in such a way that B still has great enough score and great enough length, accept both. If not, proceed as in Case (a).

(c) Suppose A and B overlap as shown in Figure 14.2(c). If the score of the overlap is a small fraction of A's score, and moving A's start codon resolves the overlap in such a way that A still has great enough length, accept both. Otherwise reject B.

(d) Suppose A and B overlap as shown in Figure 14.2(d). Move B's start codon until the overlap scores higher for B than for A. Then move A's start codon until the overlap scores higher for A than for B. Repeat until there is no more overlap, and accept both.

If there are more than two overlapping sequences, treat the overlaps in decreasing order of score. In this way, if A overlaps B and B overlaps C, and A has the highest score, then B may be rejected before its overlap with C would cause C to be rejected.

References

[1] A. L. Delcher, D. Harmon, S. Kasif, O. White, and S. L. Salzberg. Improved microbial gene identification with GLIMMER. *Nucleic Acids Research*, 27(23):4636–4641, 1999.

[2] S. L. Salzberg, A. L. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548, 1998.