CSE 527
December 5, 2005
Tia Ghose

Class Notes

Review of RFAM—the RNA family data base

Basic picture of how RFAM is constructed:

1. Uses Multiple Alignment using data sources, typically crystal structure, although occasionally the results of model prediction.

2. Has secondary structure notation

3. These are used to train a covariance model, which enables them to come up with a score threshold and a window length.
    Longer window length takes longer, but sees more of the unusual family elements

Some limitations of RFAM technique:

1. Not using method of learning alignment from mutually information (as in Durbin/Eddy).
2. There are lots of examples across species, for example, with a highly conserved structure. In this case the mutual information won't pick up because there's relatively little sequence variation
3. Families tend to be overly narrow because it doesn't handle non-secondary or variant structures
    Example: the small nucleolar RNA's involved in guiding process that modify ribosomal RNA and have sequence complementarity to tRNA
4. Models cannot deal with splicing or RNA pseudogenes (i.e. nonfunctional code or copies of other code)
5. The traditional Covariance model's speed is rather bad.
6. Finally: if I don't know about, for instance, a particular element, for instance, the IRE, and I had a bunch of genes surrounding it and the sequence conservation is low, how would I find it?

Zasha Weinberg's Work

The basic premise is a state machine much like the Markov model. Paired states emit paired symbols that correspond to the particular base pair nucleotides

Motivation: The Covariance Models are good, but SLOW. So what RFAM curators actually have in place is this system:

RFAM→ EMBL→BLAST→CM

BLAST is a filtering program that takes a query and then scans the database for short exact matches of approximately 7-11 nucleotides in length.

Thus, there is a lower bound for exact match, and then it pieces together nearby exact matches so that you get globally good matches. They take seed members of a family and BLAST each of those against the database, then passes it on. The filter removes the unpromising pairs. This reduces their computational time 100 fold.

*BUT, BLAST throws away some good stuff :

What if, for instance, you have a short sequence that isn't 11 nucleotide base pairs long?

What if you have an RNA family that has divergence but with compensatory mutations that preserve the structural regions?

Zasha's work represents an improvement.

It filters EMBL, and anything it doesn't like is junk, just like BLAST. It's a little slower than BLAST, but you don't throw anything away that you want, because it's not throwing anything away the CM would find.

Simplified Model

Based on covariance model. The state machines are summarized as a single state that emits nucleotide pairs or a gap

We lose some information this way.

The CM (covariance model) can be approximated by a hidden markov model of two staets. The information lost is such that, for instance, in the state machine model, if we emit an A, we know the other side will emit a U, but in the Markov Model, these two are uncoupled)

We split the chain into two pathways. Now, we must question—well, is this approximation valid?

We need to show that the log of the Viterbi scores of the CM are always less than or equal to those of the HMM

So we can say that we are bounded above by the HMM scores: if the HMM scores are set to infinity threshold, then all the sequences pass, but obviously, that's not a very good filter.

To establish rigor, we must ensure a suitably high threshold, but obviously practicality dictates that we want the smallest allowable threshold.

Viterbi Forward Score

For any nucleotide sequence, we can say that the Viterbi Score(x)=max{score(Π)/Π emit x}

Whereas, the Forward score(x)=sum{score(Π)/Π emit x)

So, we must have it such that the log Viterbi scores is less than or equal to the HMM scores, and we can institute the following condition to ensure this is so.

$P_{AA} <= L_A + R_A$ etc. for all two base pair combinations (on a state by state basis)

Note, that these probabilities don't have to correspond to actual probabilities, i.e. the sum of the probabilities need not equal 1.

Any scores that satisfy this linear inequality will always give the rigorous filtering, because the HMM model is not throwing anything away that the CM would keep.

Some scores are better than others. I am trying to split the scores associated with a pairstate for the HMM's, and there are many ways I can satisfy these inequalities, so which way is best?

Optimizing

For any nucleotide sequence X,

Viterbi Score(x)=max{score(Π)/Π emit x}

Forward score(x)=sum{score(Π)/Π emit x)

The expected score is:

E(Li, Ri) = The Sum over all sequences x(Forward-score(x)*Pr(x))

Where Pr(x) assumes independence of subsequent trials.

So, we want to choose the variables that minimize the expected value. We are optimizing for ow forward scores because it's an approximation to Viterbi, but because it's a summation, we can take its derivative.

To minimize the expected score we don't calculated numerically, but rather keep everything symbolic. We do the partial derivatives to get a convex optimization, leaving L_a etc. as a variable to see how the function changes as L_a is changed.

(side note: A convex optimization problem has only a single local optimum, so it's guaranteed to converge).

So How does the program actually perform?

Less than 1/10,000 of the initial set survive the first filtering process. Thus, over the CM, we save 100 fold of comp time.

But, for all the ones in the range of .01-.99, it's not practical to do this type of filtering, so for these cases you would fall back of BLAST (or use the more elaborate methods described below).

It does catch some that BLAST has discarded (see slides for details!). But, for some families, like the snoRNA with little sequence variation, BLAST actually performs fine. Plus, even though we can be sure that some of these are false positives, they are sequence matches that the CM would have found anyways.

Take home message: this algorithm is much much faster than the CM for most cases, but for others, it is not worth using. We need to modify the algorithm so that it runs with more predictable, realistic limits on comp time

Additional Work

Obviously, for some families the new method did terribly. For instance, tRNAs are diverse and have a distinct 2ndary structure, but again, we threw all that information out when we adopted the HMM.

So can we quickly reincorporate 2ndary structure information?

The trick is to decide which filters to chain together with which sequences. It is thus a shortest path problem, where we are optimizing some objective function to select which filters are most useful to run.

Idea:

1. Run Sub CM
2. Remember HMM states

These would work for only short hairpins, since it adds in a sub covariance model.

Even with these filters, there are a few families that couldn't be done quickly.

Back to Original Conditions

Our initial limitation was one of rigorous filtering—that is we are optimizing for the worst case scenario. But, perhaps we can improve the speed by throwing out that basic guarantee?

We could use profile HMMs but with scoring based not on linear inequality but chosen to optimize the average case.

After that, the program runs 10 times more quickly! We still have to consider whether the data we are throwing out is meaningful or not, and balance the two considerations.