Nadia Kulshina
CSE 527 notes
November 28, 2007

RNA search and motif discovery

Rfam is the RNA family data base; it was set up about 5 years ago. Rfam was started with 25 families and now it contains over 800 (and is way behind literature!). The curators get information about families, perform alignments and build covariance models (CM). They use the Viterbi algorithm to align sequences to the covariance model.

There are several problems with the database:
The families are overly narrow because the curators are very selective. There are known RNAs that don't have a secondary structure. It doesn't handle splicing because the CM doesn't handle it very well.
Sometimes a spliced mRNA is reverted to DNA and inserted back into the genome. But these "genes" are non-functional because they lack a promoter, they are termed pseudogenes. This happens to RNA, and it's sometimes hard to tell if it's a real gene or not.
Problems that can be dealt with are (these are addressed directly in prof. Ruzzo's lab):
- Speed and sensitivity
- Motif discovery

Ravenna: Genome scale, performs RNA search.
Typically a 100x speed up over raw CM with no loss in accuracy. It drops the representation of secondary structure in the model and the sequence is dealt with through HMM (going through HMM is dramatically faster!). Then it is possible to look in detail with the CM. HMM is built in a way that the CM score is a lower bound on the HMM score, so if the HMM score is below the threshold we are interested in, we lose nothing that the CM wouldn't reject anyway. But it is tricky to do it in a way that HMM doesn't throw out too little and CM will be run on a lot of data.

CMs are good, but slow.
It will take 10 years to go through all Rfam RNAs! So BLAST is used to find matches to the seed sequences, then the ones that pass some threshold are run in the CM (this takes up about a month). The downside, however, is that BLAST doesn't know anything about secondary structure. Ravenna, though, does know about secondary structure, so it filters better, but again, runs slower (the process takes ~2 months).

Simplified CM.
Convert CM to HMM. For each state in CM there are 2 states in HMM (half of them ordered in reverse). The key issue: 25 scores (25 emissions per state in CM) -> 10 scores (5 emissions per state, 2 states in HMM). Need: the log Viterbi scores in the CM are bounded by the

corresponding HMM scores. Viterbi finds the path in a string with highest probability. To get the upper bound that we want, it might be that probabilities don't sum up to 1, so it's not a probabilistic model, bit Viterbi still works..
Viterbi score (x) = max { } – best path
Forward score (x) = sum { } – the total over all paths
In HMM we get probabilities of 2 states emitting a better state, then sum. So the score in HMM is the upper bound in CM

Rigorous filtering.
Any score satisfying the linear inequalities give rigorous filtering (proof in slides). It is not necessarily the only path through HMM, but it's some path. The Viterbi HMM might not be the "corresponding" HMM path score, but Viterbi finds the best possible path.
Some scores filter better. E.g.:
$P_{UA} = 1 <= L_U+R_A$
$P_{UG} = 4 <= L_U+R_G$
One option would be to make them all 2, another option is: L_U=0, R_A =1, R_G =4. Assuming AGCU = 25%, the expected average is:

1) $L_U +(R_A + R_G)/2 = 4$
2) $L_U + (R_A + R_G)/2 = 2.5$

We should choose scores that drive the scoring down, then the filter works better. Now we want to choose the scores. There are several solutions that satisfy the inequalities. We can optimize the forward score, but it is difficult to optimize the Viterbi algorithm because it has a *max* in it. It is a heuristic optimization: ("forward ↓ => Viterbi ↓ =>filtered"), but it is still rigorous because it is "subject to score linear inequalities."
As a function of 800 parameters (L_i,R_i), under 0th order background model.
Calculating E (L_i,R_i)
What's the expected score when I arrive at a state? As a function of the variables, it's dynamic programming. Addition *vs. max* is great! Now minimize E (L_i,R_i). Calculate a symbolic expression for that state. Then calculate the symbolic derivatives:

We can't afford to write out the whole symbolic expression. So it has to be written out as an algorithm that evaluates it.

Estimated filtering efficiency (Rfam at 4th release, looked at 139 families)
Take family1 -> CM -> HMM -> feed to the database. 105 families had a filtering fraction $<10^{-4}$ (the table is in slides). The HMM runs 100x faster than CM. At a filtering fraction of 10^{-2}, about one percent of the database will be run through the CM, so the HMM time on the whole database approximately equals the CM time on the remaining 1%. For most families it works really well, but for some families it doesn't. The model is built individually for a family, so only one family is run at a time. When we build HMM we have to throw away the secondary structure, but we

know that it is very important. However, if the RNA family has some sequence conservation we can get good scores and we can get great speed ups. But for families of diverged sequences (secondary structure can still be conserved) HMM is very flat and not good.

Results: new ncRNAs?

For many families nothing new was found compared to BLAST. But found new members in certain families like S-box, hammerhead, and others. The question is: are they real? Maybe; e.g. putative new IREs (Iron Response Elements) are located near iron-processing genes (that's where these RNAs are involved). But if they are false positives that points to flaws in the CM model, so it's an informative result in either case.

Additional work:

So far there's been no use of secondary structure. So tried to add stuff that could capture secondary structure, but would still run fast. Just to enhance the filtering:

1. Sub-CM – pick out parts of secondary structure that are relatively small, but well conserved (structure, not sequence) and plug this small CM into HMM.
2. Extra HMM states remember mate. HMM doesn't enforce correlation, so split into 2 cases and score in 2 states separately. It's time consuming, so can be done on small structures only, e.g. hairpins.
3. Try lots of combinations of "some hairpins".
4. Chain together several filters.

E.g., we can now affordably process tRNAs – found ~5000 new potential tRNAs.

Heuristic filters.

Rigorous filters are optimized for the worst case. Using the Same sorts of HMM ideas, but optimizing scoring for "average" case, we can get significant additional speedup with modest loss in sensitivity. On some examples, BLAST has a sensitivity of 60% (meaning that 40% of true positions are thrown out). Heuristic filters are 99% sensitive. For BLAST you would have to pass a 1000x more data.

Motif discovery.

If we have a model we can search the genome at relatively good speed and accuracy. Now we want to automate the model discovery/construction phase – something like MEME for RNAs. (Recall that Rfam models all depend on hand-built alignments.) Suppose there are 10-20 unaligned sequences of ~1kb. It is hard to align 150 bp that are in common if they are surrounded by a lot of not-in-common stuff.

Approaches:

1) Align, then look at common structures. Find Watson-Crick pairing, then it's a good basis for predicting structure. But, compensatory substitutions reveal structure, while alignment penalizes them. Structure conservation vs sequence conservation.

2) Predict structures, then try to align them. But single sequence structure predictions are only 60% accurate. There's no good model/algorithm for aligning 2 structures.

Do both together – simultaneously align the sequence and do structure prediction. Possible (the Sankoff algorithm, but unfortunately it's an $O(n^6)$ algorithm, not $O(n^2)$ like the Smith-Waterman.