Lecture 6

Sequence Similarity, continued

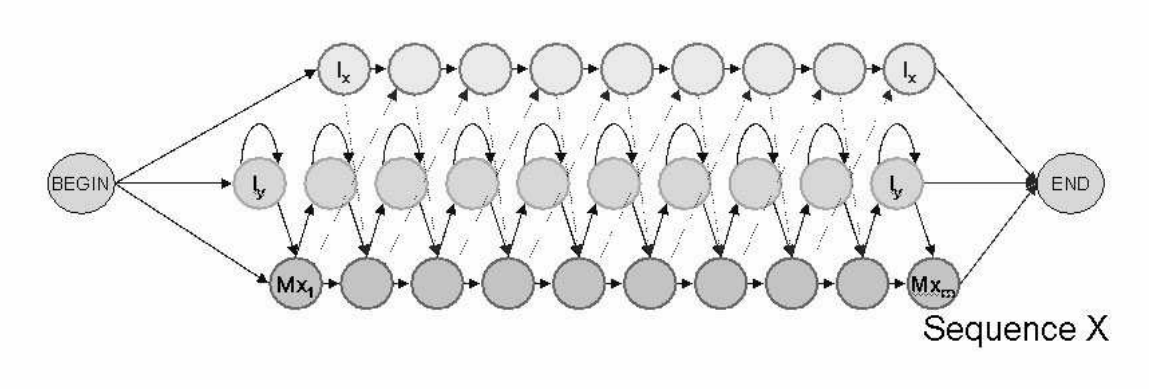
Scribe: Andrei Aron

Acknowledgement: Images have been copied from the class slides, text is based on the class lecture.

The Viterbi Alignment Algorithm (continued)

The 3 state automaton described in Lecture 5 can also be interpreted as a Pair Hidden Markov Model (HMM) by assigning probabilities to the transitions. The alignment that generates the highest joint alignment probability is chosen.

Another way of representing the model is by imagining we're threading the sequence Y through sequence X. The current y index may be gapped, we may skip an x or we can align y to some x:



The probabilities associated with transitions in this model are computed based on a database of known, curated alignments. This model can actually yield additional statistics, such as posteriors for individual instance alignments/gappings. Here is an example query: "What's the probability that x_i and y_j align, given that sequences x and y align?" Such queries can also be computed by using Dynamic Programming techniques.

Database Search Tools

BLAST - Fast database search:

First published in "Journal of Molecular Biology" in 1990 by Altschul, Gish, Miller, Myers and Lipman, "Basic local alignment search tool", or BLAST, is one of the most cited papers in computational biology (Google Scholar alone shows it's been cited 16 thousand times).

BLAST approximates the results of the Dynamic Programming algorithm, only without the associated time cost.

BLAST first creates a dictionary of query words (the word length is typically 4 for amino-acids and 11 for nucleotides), which it sorts in linear time by using RADIX. It then runs once through the database, matching the query words above a certain similarity threshold. There are two approaches for matching words in the database: the first method creates an inverted index of words in the query and iterates through the database once in order to find potential matches; the second method uses a Direct Finite Automaton to find matches in the database. The second method was preferred by the authors of the BLAST paper.

BLAST then creates a list of database location candidates, which it then attempts to extend to the left and to the right. (The first version of BLAST did not allow gaps.) The results are then ranked according to the similarity score.

The word length can be increased, improving sensitivity and trading off speed.

PSI-BLAST

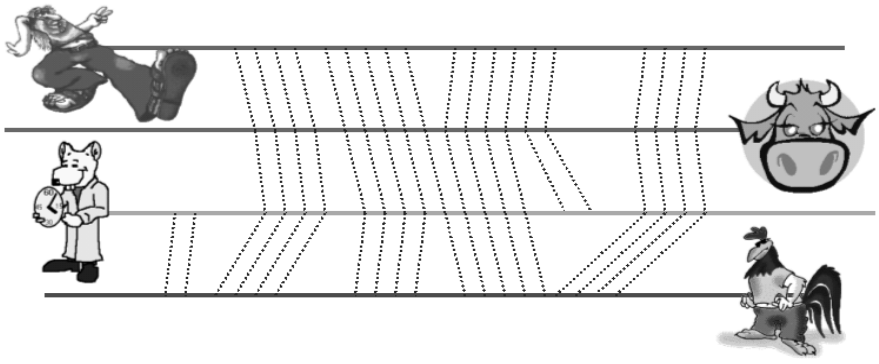
Position specific iterative BLAST was created in 1997. PSI-BLAST represents an extension of BLAST where position specific scoring is used. What this means is that when looking for word matches in the database, we create a "profile" or family for the words we're looking for. Once we found all matches within a certain significance threshold, we use the obtained profiles to refine the search by repeating the procedure. This allows us to find more significant matches. The profiles are represented as substitution matrices.

Other Blast Variants

TBlastX "compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database using the BLAST algorithm" (2). For example, one of the six-frame translations of "5' ACCGTTA 3'" is TGGCAAT (3's complement)

Other versions include: BLAST-N, used for genomic sequences, BLAST-P used for proteins, BLAST-X for translated genome versus proteins and TBLASTN for proteins versus translated genomes.

Multiple Sequence Alignment



One of the reasons why multiple sequence alignments are necessary lies within the protein phylogenies. The events that define the phylogenies are: duplication (within the same species) and speciation (when two or more species diverge).

There are two types of similarity encountered in multiple sequence alignments: weak similarity and helix similarity.

But what is a multiple sequence alignment? What we want is to take a number of sequences and introduce gaps (but not actually allow gaps in all positions) in such a way that in the end, all sequences have the same length, and the global scoring function is maximized. Many times, aligning multiple sequences also yields better pair-wise alignments. Due to the fact that we need an evolution model in order to make accurate alignment predictions, the Scoring Function is, unfortunately, still an area of research. However, a good heuristic scoring function is the weighted sum of induced pair-wise alignment scores. The induced pair-wise alignment is the pair-wise alignment obtained from the corresponding multiple sequences alignment, by removing matching gaps. The weights are increasing with evolutionary tree distance:

$$S(m) = \sum_{k<l} w_{kl} s(m_k, m_l)$$

The reason for the weighting of the pair-wise sums is that some parts may be more densely sampled than others, thus incorrectly weighing more.
Here is an example:

```
x:  AC-GCGG-C
y:  AC-GC-GAG
z:  GCCGC-GAG
```

Induces:

```
x: ACGCGG-C;   x: AC-GCGG-C;   y: AC-GCGAG
y: ACGC-GAC;   z: GCCGC-GAG;   z: GCCGCGAG
```

Any multiple sequences alignment naturally generates a profile (as discussed above). Essentially, a profile represents a set of statistical distributions for each position in the final aligned sequences, computed by simple ML:

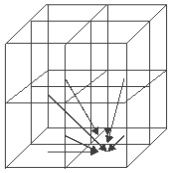
```
     -  A  G  G  C  T  A  T  C  A  C  C  T  G
     T  A  G  -  C  T  A  C  C  A  -  -  -  G
     C  A  G  -  C  T  A  C  C  A  -  -  -  G
     C  A  G  -  C  T  A  T  C  A  C  -  G  G
     C  A  G  -  C  T  A  T  C  G  C  -  G  G

A       1           1        .8
C    .6          1     .4  1     .6 .2
G          1 .2                 .2        .4  1
T    .2                1    .6              .2
-    .2       .8                      .4 .8 .4
```

Multidimensional Alignment Algorithms

Multidimensional DP is a generalization of the Needleman-Wunsh algorithm. The total score for a multiple alignment is the sum of column-wise scores:

$$S(m) = \Sigma i\ S(mi)$$

The dynamic programming algorithm for two sequences can be generalized to work for any number of sequences. Let's say we have **N** sequences, **i**, **j** andk are positions within the sequence, and let's denote **F** as the optimal score function. Let's also assume the final length (including gaps) is **L**. Here is an illustrative example, for the case of 3 sequences:



$$\begin{aligned} F(i,j,k) = \max\{ & F(i-1,j-1,k-1)+S(x_i, x_j, x_k), \\ & F(i-1,j-1,k\ \ )+S(x_i, x_j, -\ ), \\ & F(i-1,j\ \ ,k-1)+S(x_i, -, x_k), \\ & F(i-1,j\ \ ,k\ \ )+S(x_i, -, -\ ), \\ & F(i\ \ ,j-1,k-1)+S(\ -, x_j, x_k), \\ & F(i\ \ ,j-1,k\ \ )+S(\ -, x_j, x_k), \\ & F(i\ \ ,j\ \ ,k-1)+S(\ -, -, x_k)\ \} \end{aligned}$$

The problem is that now, we have a DP matrix of size L^N and we have $O(2^N)$ neighbors, so we get a time complexity of $O(2^N L^N)$ which is huge.
The generalization of the pair HMM for the 3-state DFA is even worse, having a running time of $O(4^N L^N)$.

Fortunately, there exists a simpler way of performing the alignment, by making use of the evolution tree. The alignment is called "Progressive Alignment". Here, there are two possible situations:

1. The evolution tree is known (for instance, by noticing sequential similarities between rat, mouse, human and using them to build the tree). In this case, in the order of the tree, at each node, we align exactly two sequences or two profiles. We then propagate the resulting alignment. There is a problem with this approach, however: the algorithm is greedy – once we align XY for instance, when looking at XZ, we can no longer change the XY alignment.

2. If the evolution tree is unknown, then we need to compute all pair-wise alignments, build the distance matrix (which defines evolutionary distance based on the computed pair-wise alignments), and use a clustering algorithm in order to re-create the evolution tree. Finally, the evolution tree can be used just like in the first case.

Some heuristics may be used in order to improve alignments: Iterative refinement, A* search, Consistency and Simulated Annealing for instance.

We shall briefly analyze Iterative Refinement as an alignment improving heuristic. Iterative Refinement addresses the problem of the immutability of alignments made earlier in the tree.

The algorithm is due to Barton and Stenberg and is extremely simple: remove a sequence from the original set, align the others, and then align the one we removed to the others; once all are aligned, repeat the procedure by removing the next sequence; repeat the procedure until convergence is achieved.

This algorithm works well in general, but there are cases when it does not work at all.

Another variant of this algorithm is the Tree Partition algorithm. Here, we pick sequences from a the original set, and then randomly decide to put them in the left or right sub-trees.

Multiple Alignment Resources

LUSTALW is the most widely used software. It has high throughput, and it is accurate. Unfortunately it only scales to a few hundred sequences. **MUSCLE** scales to thousands of proteins – which is important for instance when comparing histoms, or when comparing HIV characteristics for many patients. **PROBCONS** only scales up to one hundred and it is slower, but it is the most accurate.

MUSCLE's secret lies in the first two steps: since the phylogeny tree is unknown, it needs to do all pair-wise alignments in order to build the tree. However, MUSCLE avoids this quadratic alignment cost. MUSCLE creates a dictionary and gets a crude idea of how many common terms the sequences have; based on this crude idea, it creates a candidate tree, D_{DRAFT} which it then refines. The running time for this procedure is given by the number of sequence pairs times linear time in L, so we get:

$$(N-1)^2 * l^2$$

MUSCLE's time is $O(N^2 * l)$ so it is smaller by almost a factor of "l".

PROBCONS on the other hand, relies on consistency by parameterizing a probability distribution over all possible alignments of all pairs of sequences. It uses a HMM/DFA similar to the one used in pair-wise alignment. The Viterbi algorithm is then used in order to determine the highest probability alignment. Unfortunately, this is not necessarily the most accurate alignment. Therefore, Maximum Expected Accuracy is used instead.

A good analogy is that of the lazy teacher. The teacher gives his students an exam comprised of one question with a True/False answer. The teacher doesn't actually know the answer himself, but needs to build an answer key based on the answers of his students. The equivalent of Viterbi would be that the teacher chooses the student with the highest previous grade, and decides that whatever his answer here, on this test, it must be correct. MEA computes instead the weighted vote (where the weights are a function of the previous grades) of the whole class – so the best student would get more weight than any other one student, but would not decide by himself the answer.

Here is an example of how to compute accuracy; say we got this alignment:

A	A	C	_	C	C
A	A	T	G	G	C
good	good	bad	50/50	unclear	good

Other References:

[1] SF Altschul, W Gish, W Miller, EW Myers, DJ Lipman - J. Mol. Biol, 1990, "BLAST"
[2] http://www.incogen.com/public_documents/vibe/details/tblastx.html