Ensemblign

William Heaton

6 December 2013

Introduction

Ensemblign is a novel multiple sequence alignment algorithm for estimating the MLE multiple alignment or sampling from the posteriori distribution of multiple alignments. It takes the approach of first computing all pairwise alignments and then combining them into a multiple alignment. But it improves on other algorithms with this approach because it benefits from the posteriori distribution of pairwise alignments generated by the forward/backward algorithm hidden markov model alignment outlined in Durbin and Eddy[2] and then computes the highest probability self consistent multiple alignment via a modified Karger's algorithm. Due to the fact that Karger's algorithm is a monte carlo approximation algorithm, we can also sample from the distribution of posteriori multiple alignments. Armed with this algorithm, biologists can sample not just MLE, but typical explanations of their multiple sequence data. But to be honest, I am not sure how good or fast this will be so I just need to code it up and test on some standard MSA benchmark data to see.

Motivation

Goals

Assumptions Every algorithm has a assumptions. Only the prudent ones list them.

- We assume that all sequences input have arrisen from the same source molecule with the variations coming from mutation or sequencing error. If this is not the case, then the math and the algorithm will not be correct.
- Because dissimilarities due to mutation and sequencing error will conform to different distributions, the scoring of each should be for that partucular case.
- I will list other assumptions here as they occur to me.

Pairwise Alignment

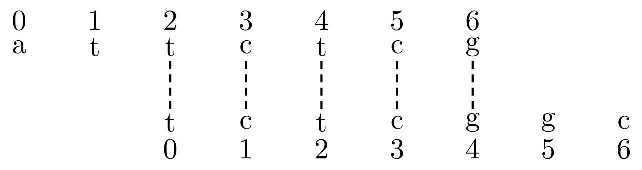
There exists a considerable literature on pairwise alignments that we need not discuss at length. The first description of the dynamic programming alignment algorithm was by Needleman and Wunsch[5] when was then generalized from global to local alignment by Smith and Waterman.[6]. But the pairwise alignment algorithm used for this project is the HMM formulation first described by ¡citation needed¿ and laid out in [2] chapter ¡look up chapter¿. The benefit of this latter method is that not just the single highest scoring alignment is generated, but for every pair of nucleotides that could be aligned between the two sequences, a probability is generated representing the posterior probability over all possible alignments of the two sequences weighted on the likelihood of those alignments.

First we will consider the case of non-HMM alignment and then we will extend it to the more complicated but more powerful case. These algorithms construct an ordered set of matched pairs of nucleotides (or pair of nucleotide and gap) between the two input sequences such that a score function on the level of error admitted by the alignment is optimized (maximized or minimized depending on scoring metric over all possible alignments). Since this is all previous work not novel to our method we will simply refer the reader to appropriate sources and provide one example.

Giving the ordered set of pairs for this alignment as

$$\{\{v_2^0, v_0^1\}, \{v_3^0, v_1^1\}, \{v_4^0, v_2^1\}, \{v_5^0, v_3^1\}, \{v_6^0, v_4^1\}\}$$

Figure 1: Pairwise Alignment.



The aligned pairs of monomers in such an alignment are likely to represent the same particular locus in the source molecule.

Later in this document it will become useful to diagram this alignment as a graph where the monomers are vertices and an edge represents the fact that those two monomers have been aligned. This seems rather trivial for the pairwise case but will become more complex later. An example of this can be seen in Figure 2.

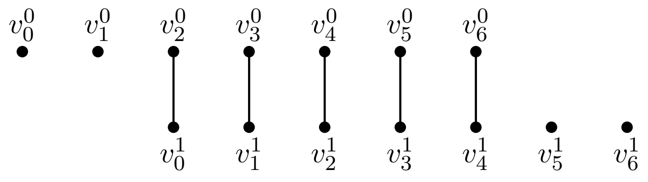

v_0^0 v_1^0 v_2^0 v_3^0 v_4^0 v_5^0 v_6^0
v_0^1 v_1^1 v_2^1 v_3^1 v_4^1 v_5^1 v_6^1

Figure 2: Pairwise Alignment: Graph Representation.

Multiple Alignment

We shall describe the homology between a set of sequences with a multiple alignment. The purpose of a multiple alignment is to match input nucleotides in sets that reflect the nucleotide occurrences on the source molecule from which the inputs originated. In structure, a multiple alignment is an ordered set of sets of nucleotides. Each set (**aligned point**) consists of at most one monomer on each measurement. Additionally, each monomer on each sequence is present in at most one set. Finally, the sets must obey the ordering principle: if monomers a, b, and c occur on the same input sequence such that b lies after a and before c, and each of a, b, and c is in an aligned set, the the aligned set containing b lies after the aligned set which contains a and before that which contains c in the multiple alignment. Intuitively, each aligned point consists of those monomers that "match," i.e. that are measurements of the same locus in the source molecule.

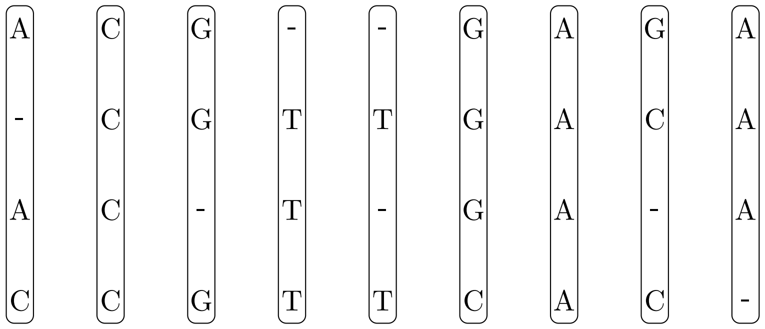


Figure 3: Multiple Alignment

Graph Conception of a Multiple Alignment

Recall the graph representation of pairwise alignments from figure 2. We will now extend this to a multiple alignment. The aligned points that make up a multiple alignment are equivalence classes, such that every pair of monomers in such a set has the relation "are homologous." A **graph** can built representing these pairwise relations, wherein a vertex v_j^i represents the j^{th} monomer on sequence i and the (undirected) edge (v_j^i, v_l^k) represents that v_j^i and v_l^k are homologous with respect to the sequence of common origin. By way of notation, $v_j^i.m = i$ and $v_j^i.e = j$. Because, by definition, those monomers in a given aligned point are homologous to one another and to no other monomers, each connected component in this graph is fully connected and consists of the monomers in one aligned point.[1]

Multiple Alignment as the Union of Pairwise Alignments

Let us call this multiple alignment graph G, consisting of a set of vertices V and a set of edges E. We can, for each pair j and l, define $E_{jl} = E_{lj} = (u, v)$ in E such that $u.m = j$ and $v.m = l$. Since each pair of monomers (u, v) in E comes from exactly one pair of different sequences, the set of all E_{jl} is a partitioning of E. E_{jl} defines a **pairwise** alignment

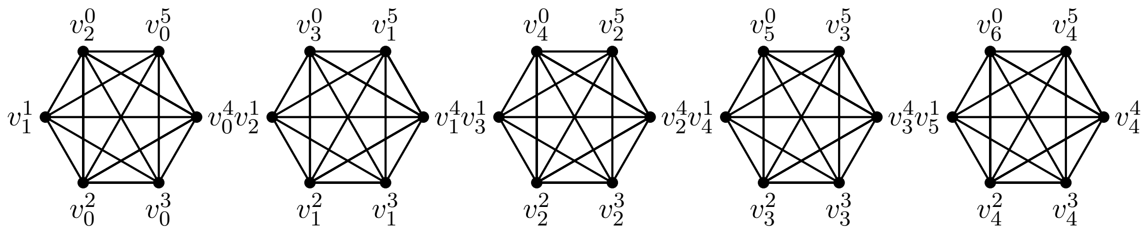


Figure 4: Multiple Alignment: Graph Representation. As you can see this results in a series of clique subgraphs.

between sequences j and l, consisting of the pairs of homologous monomers between these two sequences. That E_{jl} is a partitioning of E is also to say that E is the union over all such pairwise alignments. So one way to determine the perfect multiple alignment between a collection of sequences is to take the union of perfect pairwise alignments.

But as a result of sequencing error or mutation, a given pairwise alignment is rarely perfect. Consider two sequences, x and y, with monomers $x_{1 \ldots m}$ and $y_{1 \ldots n}$. Each monomer derives either from a source molecule site γ or from an insertion. In the latter case, the monomer is not homologous to a monomer on any other sequence and a perfect pairwise alignment will not include this monomer in a matched pair. In the former case, this monomer will be matched if and only if the other sequence has an monomer deriving from γ. So the ordered set of aligned points in a perfect pairwise alignment consists of one matched pair for each monomer in the intersection of true monomers in the two sequences.

Modified-Karger's Algorithm for Contradiction Separation

Suppose that we have performed pairwise alignments between all pairs of input sequences. Consider the set of edges E' (and the graph $G' = V, E'$) formed by taking the union of these imperfect pairwise alignments. E' differs from the perfect solution E in its missing and extra edges. Most importantly, the extra edges mean that E' has edges between what would be separated components in E. (Additionally, some edges are missing within what would be connected components in E. These missing edges are less of a concern under the assumption that we have at least several sequences because it is very unlikely that enough

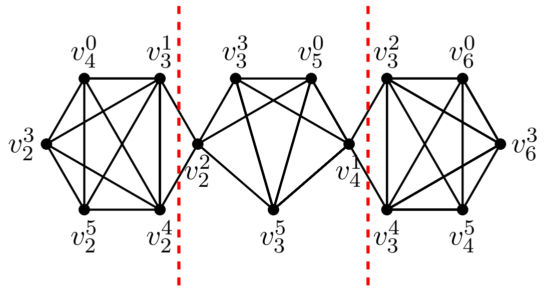


Figure 5: Multiple Alignment: Graph Representation. This alignment graph contains multiple **contradictions**. Note that the vertices have been rearranged spatially for ease of viewing the graph. The broken red lines indicate the edges that must be cut in order to obtain the contradiction-free multiple alignment that best explains all of the pairwise alignments.

edges might be missing to separate a connected component into two or more components.) In order to recover E as best possible, we set about removing the extra edges from E'.

The extra edges in E' have the property that they introduce **contradictions**. Let us define a contradiction as a connected component in G' that contains two or more different vertices from the same sequence. This means that the multiple alignment implicit from G' currently counts two monomers on one sequence as arising from a single monomer in the underlying true sequence of the source molecule. This is always an error because each aligned point in our multiple alignment should correspond to a particular monomer γ in the sequence of common origin and it is impossible for two sites on the same sequence to be homologous to the same γ. One important step towards fixing E' is to separate these **contradictory components** into non-contradictory ones. ¡add figure detailing an example of contradictory pairwise alignments¿

Under the presumption that most edges in E' are correct, a natural way to fix these contradictions is to find a min-cut such that no contradictions remain. We use a novel, modified version of Karger's Algorithm to find this min-cut. Karger's Algorithm is a probabilistic algorithm that finds the min-cut of a graph by finding strongly connected components and then severing them from one another. These strongly connected components are discovered by "contracting" edges at random until only two nodes remain. To contract an edge is to remove the edge and combine its end nodes into one node retaining all other edges therefore allowing multiple edges between two nodes. The selected cut itself is the set of all edges remaining when no further contraction is allowed. This process can of course fail to find the min-cut. Consider for instance that the first edge contracted at random happens to be an edge that is in the min cut. For this reason we must perform this succession of contractions many times in parallel and selecting the minimum cut among these iterations, the algorithm offers high probability of finding the min-cut. Because strongly connected components contain far more edges than other areas of the graph, these areas get contracted earlier with higher probability. This convenient property of highly connected components allows for very high probability of finding the real min-cut with surprisingly few iterations. This run time can be further improved by a modification known as the Karger-Stein algorithm. In our case, we impose the additional constraint that no two vertices representing monomers on the same sequence can be contracted. This constraint is natural because the fully-contracted vertices at the end of Karger's Algorithm are identical to the **aligned points** our multiple alignment seeks to recover. As such, our modified Karger's Algorithm contracts edges at random without violating constraints until no contractions are allowed under the constraints. The min-cut selected by this modified Karger's Algorithm represents a likely selection of the extra edges in E' and, in any case, results in an ordered set of non-contradictory connected components that best explain the set of pairwise alignments.[4][3] As a simplified example, take the following set of fragments.

Karger's Algorithm, along with our modifications, is uniquely suited to this problem. No other formulation of the min-cut problem (network flow, etc.) can be so simply extended with our constraints. The operation of Karger's Algorithm consists of a probabilistic sampling of the space of graph cuts. By introducing our constraints, we simple decline to sample the portion of the space that contains contradictions. And, if this was not clear, in the simple case in which E' contains no contradictions to begin with we do nothing and just take it as the multiple alignment.

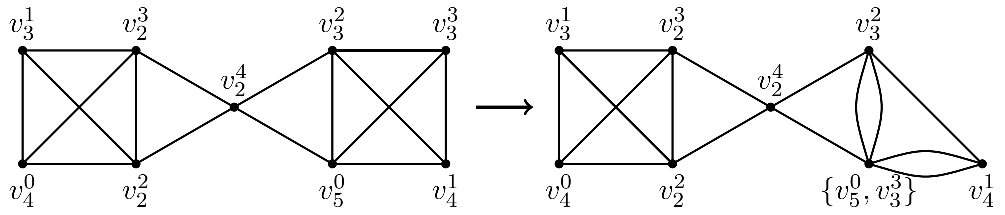


Figure 6: Contraction on edge from v_5^0 to v_3^3. Edges are drawn at random and contracted if they do not have labels of the same fragment (the superscript here). This process continues until either there are 2 nodes left in the graph and the remaining edges are the "cut" or no more edges can be contracted given our constraint. This process is then repeated several times and the cut with the fewest cut edges is selected as the most likely.

Conclusions

References

[1] Eduardo Corel, Florian Pitschi, and Burkhard Morgenstern, *A min-cut algorithm for the consistency problem in multiple sequence alignment*, Bioinformatics **26** (2010), no. 8, 1015–1021.

[2] Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison, *Biological sequence analysis: probabilistic models of proteins and nucleic acids. cambridge univ*, 1998.

[3] David R. Karger, *Minimum cuts in near-linear time*, STOC, 1996, pp. 56–63.

[4] David R. Karger and Clifford Stein, *A new approach to the minimum cut problem*, J. ACM **43** (1996), no. 4, 601–640.

[5] S. B. Needleman and C. D. Wunsch, *A general method applicable to the search for similarities in the amino acid sequence of two proteins.*, Journal of molecular biology **48** (1970), no. 3, 443–453.

[6] T. F. Smith and M. S. Waterman, *Identification of common molecular subsequences.*, Journal of molecular biology **147** (1981), no. 1, 195–197.