Quick Overview of FastTree

Rayne Hernandez

July 12, 2017

Abstract

FastTree (FT) is a Minimum Evolution (ME) algorithm working on large alignments of sequence data to construct large phylogenies and estimate their reliabilities. FastTree employs Neighbor Joining (NJ) but unlike other NJ methods does not use a distance matrix. Instead it uses profiles stored at each node in the tree. It also speeds up NJ using a "top hits" heuristic and refines the tree using Nearest Neighbor Interchanges (NNI) to reduce tree length. FastTree also uses local bootstrapping at each split instead of resampling the whole sequence. Overall the algorithm takes $O(NLa + N\sqrt{N})$ memory and $O(N\sqrt{N}\log NLa)$ time.

1 Introduction

Neighbor Joining and Minimum evolution methods are the fastest and most scalable methods for infering phylogenies. However previous NJ methods rely one calculating a distance matrix of the distances between each pairs of nodes. In traditional NJ one starts with a star topology and and joins the best pair of nodes together according to some minimum evolution criterion. This greedy procedure considers all $O(N^2)$ choices each iteration taking a total of $O(N^3)$ time.

As sequence alignments grow larger, the cost of producing the distance matrix is growing faster than CPU speeds, causing even NJ methods to be too slow for large sequences. Likewise to calculate the reliability of the tree, most NJ methods use bootstrap where columns of the sequence are resampled and the whole tree is reconstructed and the proportion of correct splits counted. This can take 100 to 1000 times longer than the original computation.

2 Approach

FastTree makes four improvements over the traditional NJ method.

Firstly, it stores profiles for each individual node in the tree instead of storing a distance matrix. FastTree uses profiles to calculate the distances between nodes. Essentially each profile for each node stores a vector at each nucleotide position listing the frequencies of the characters in the children nodes at that position. The frequencies are the weighted averages of the characters in the children node at each position.

The intuition is that the average of the distances between sequences in two subtrees A and B equals the distance between profile(A) and profile(B) because profile(A) is the average of the sequences in A.

Secondly, FastTree reduces the number of joins considered in neighbor joining using a "Top Hits" heuristic. This reduces the time cost of NJ from $O(N^3)$ to $O(N\sqrt{N}\log NLa$ and the memory cost from $O(N^2$ to $O(NLa + N\sqrt{N}$.

Thirdly, FastTree refines the initial topology using Nearest Neighbor Interchanges. For each split ((A,B),(C,D)) FT considers the alternative topologies ((A,C),(B,D)) and ((A,D),(B,C)) and makes pick the one with the shortest tree length. This takes $O(N\log NLa$ additional time and $O(NLa)$ additional space.

Lastly, FT uses a local bootstrap at each internal split instead of resampling the whole tree. Local Bootstrapping resamples the columns of profiles of ((A,B),(C,D)) couting the fraction of resamples that support the split over the two alternative topologies considered in NNI. this provides an 100 fold speedup but that the same time only estimates the probability that the given split is correct given ((A,B),(C,D)) is the correct subtree.

The results are that FT is more accurate than other ME methods and that Local Bootstrap is a reasonably good indicator of whether the splits are correct.

3 Methods

3.1 Distance Between Sequences

Uncorrected distances between two sequences are calculated as such, d_u = fraction of positions that differ from eachother. To correct for multiple substitutions FT uses the Jukes-Cantor distance $d = -\frac{3}{4}\log\left(1 - \frac{4}{3}d_u\right)$. Protein sequences are more complicated.

3.2 Distance Between Profiles

FT uses profiles to estimate the average distance between the children of two nodes. To calculate FT first calculates the distance at each position as the average dissimilarity of the characters at that position. Then the uncorrected distance between profiles is the the weighted average of the distances at each position weighted by the proportion of of nongaps at each position.

The profile distance is identical to the average of uncorrected distances assuming no gaps. For sequences A and B

$$\Delta(AB, C) = \frac{d_u(A, B) - d_u(B, C)}{2}$$

This is not true for corrected distances or if sequences that have gaps.

$$\Delta(AB, C) \neq \frac{d(A, B) - d(B, C)}{2}$$

where each d(.,C) is a Jukes-Cantor log function. FT cannot compute averages of logs but instead computes the log of the averages, a reasonable approximation.

3.3 Distances between Internal Nodes

After joining two nodes A and B traditional NJ sets the distance between AB and another node C as

$$d_u(AB, C) = \frac{d_u(A, C) - d_u(B, C) - d_u(A, B)}{2}$$

FT instead computes

$$d_u(AB, C) = \Delta(AB, C) - u(AB) - u(C)$$

where $\Delta(i, j)$ is the profile distance and $u(i)$ is the "up-distance" function representing the average distance of the node from its children. Roughly $u(i) = 0$ for leaf nodes and $u(ij) = \Delta(i, j)/2$ for internal nodes. The real process uses weighted averages for calculating $u(ij)$. The profile distance and NJ distance are actually equivalent assuming no log correction and no gaps.

3.4 Neighbor-Joining Criterion

NJ selects the join that minimizes the criterion $d_u(i, j) - r(i) - r(j)$ where r(i) is the average "out distance" representing the the average distance of node i to every other node. Traditional NJ uses

$$r(i) = \sum_{k \neq i} \frac{d_u(i, k)}{(n - 2)}$$

Traditional NJ computes all out-distances before joins costing $O(N^2)$ time. FT only computes each out distance as needed in $O(NLa)$ time by maintaining a "total profile" T which is just the average of all active profiles. Then then out distance is calculated using

$$\sum_{k \neq i} \Delta(i, k) = n\Delta(i, T) - \Delta(i, i)$$

where $\Delta(i, i)$ can be interpreted as the average distance between the children of i.

No log corrections are used for NJ because this actually reduces the accuracy.

3.5 Selecting the Best Join using "Top Hits"

FastTree uses a top hits heuristics to reduce the number of joins considered at each step. A list of closest m neighbors according to the NJ criterion is maintained for each node where $m = \sqrt{N}$. Top hits lists are generated for each node in the following manner. For the first node A the top 2m top hits are calculated and then the top hits for every node B in A's m top hits are calculated by comparing them to the other 2m nodes for A. This process is iterated until every node has a top hits list. This takes $O(N\sqrt{N}L)$ time and $O(N\sqrt{n})$ space.

During NJ top hits lists are maintained for joined nodes. After a join the new nod top hits list is made by comparing its profile to it children's top hits list. Inactive joined nodes are removed from the all lists and replaced with their ancestor. Top hits are also periodically refreshed to improve accuracy. FT also remembers the "best-join" for each node.

To find a join during each iteration of NJ, FT finds the top m "best-joins" and sorts them into a list. It then recomputes the NJ criterion for these joins and selects the best one. FT also does a local hill climb to find a potentially better join, ie. given join AB, FT considers A-TopHits(B) and B-TopHits(B). In theory NJ using these heuristics takes $O(N\sqrt{N}\log NLa)$ time

3.6 Nearest Neighbor Interchanges

After the initial topology is created, FT compares each split to the two alternate topologies. Ie. given the split ((A,B),(C,D)) FT considers ((A,C),(B,D)) and ((A,D),(B,C)) and choses ((A,B),(C,D)) if

$$d(A,B) + d(C,D) < min(d(A,C) + d(B,D), d(A,D) + d(B,C))$$

FT does $\log_2 N + 1$ rounds of NNI. Overall this takes an extra $O(NLa)$ time and $O(NLa)$ space.

3.7 Local Bootstrap

Resamples the profiles for each split and counts the proportion of resamples that support the split.

3.8 Branch Lengths

Branch lengths are computed at the end given by the formulas

$$d(AB,CD) = \frac{d(A,C) + d(A,D) + d(B,C) + d(B,D)}{4} - \frac{d(A,B) + d(C,D)}{2}$$

for internal branches and

$$d(A,BC) = \frac{d(A,B) + d(A,C) - d(B,C)}{2}$$

for leaves A.