The comparison of phylogenetic reconstruction using five different methods

Lívia Qian
KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science
liviaq@kth.se

Abstract

The aim of this paper is to briefly describe and compare the performance of some widely accepted and one lesser-known method for phylogenetic reconstruction. Instead of trying to be a comprehensive guide summarizing the theoretical background of phylogenetics, it focuses on the main principles of creating simple phylogenetic trees and the conclusions I reached while testing these methods. While my primary goal was not to optimize them and making them scale to more complex data, approximating the time complexity of each algorithm was a crucial part of the experiments besides measuring the accuracy of the generated trees.

Introduction

Phylogenetics is the study of the evolutionary history of species and groups of species. There are multiple forms of expressing the relationship between different groups, one of them being the phylogenetic tree (or evolutionary tree), a diagram containing all the taxa in the form of leaves. Common traits can be observed using DNA sequences or morphology (the latter being generally more complex due to the many characteristics I certain species can have). Phylogenetic trees can be rooted and unrooted; the former is a version representing not only the similarity between each taxon but the ancestral relationships as well, while the latter form only illustrates the connection between the taxonomical units.

Methods

There are multiple steps included in creating a phylogenetic tree; DNA sequences need to be of the same length, therefore a proper sequence alignment is needed before most of the algorithms can be used. There are three main categories of tree-creating algorithms: distance-based, maximum parsimony and maximum likelihood methods. While all these techniques are fairly popular, distance methods are usually recommended when computational time is an important factor; in other cases, parsimony and likelihood are preferred as they are more rigorous and have the ability to explore different combinations.

Sequence alignment

DNA sequencing is the process of determining the order of nucleoids in the DNA; there are four bases — adenine, guanine, cytosine, and thymine — represented by characters A, G, C, T whose order needs to be determined in order to gain relevant information about the molecule at hand. Sequence alignment is a way of arranging multiple DNA sequences into a matrix so that similar regions are grouped together — that is, regions that show a high degree of similarity in multiple sequences are placed under-

neath each other. As this process makes the characters in the sequences shift, a special character (usually a "-") is used to fill in the gaps.

Neighbor joining

Neighbor joining is one of the distance-based methods created by Saitou and Sei in 1987 [1]. A distance matrix is needed for all of the calculations in this method; this matrix should contain the pairwise distances between the input sequences according to a predefined distance metric. This matrix reduces in size after each step because of the clustering nature of the method.

This is one of the simplest and fastest methods in phylogeny, but it may not produce accurate results.

WPGMA

Originally attributed to Sokal and Michener [2], WPGMA (weighted pair group method with arithmetic mean) is a hierarchical clustering method that uses a distance matrix in a similar fashion as the neighbor joining method. Each step consists of the following:

1. Find the smallest element in the distance matrix. Let's say that it is $d(a, b)$ and that it belongs to elements a and b (both of them could be clusters). Let's create a parent node called p.

2. We need to calculate branch lengths $\delta(a, p)$ and $\delta(b, p)$. If both a and b are leaves, these lengths are simply

$$\delta(a, p) = \delta(b, p) = \frac{d(a, b)}{2} \tag{1}$$

   If a or b are clusters, the height of the subtree belonging to them needs to be deducted from the amount mentioned before. Let's assume that a is a cluster and that c is a leaf of the subtree belonging to it. Then

$$\delta(c, p) = \frac{d(a, b)}{2} \tag{2}$$

$$\begin{aligned} \delta(a, p) &= \delta(c, p) - \delta(c, a) = \\ &= \frac{d(a, b)}{2} - \delta(c, a) \end{aligned} \tag{3}$$

   In the case where a is a leaf, c could be represented by a itself and $\delta(c, a)$ would be zero, therefore this is a generalized formula that can be used in every situation.

3. After cluster $p = (a, b)$ is created, it should be integrated into the distance matrix as a new element, thus both a new column and a new row should be created (the columns and rows belonging to a and b can be deleted). Let's take an arbitrary element q. The distance between q and p can be calculated as

$$d(p, q) = d((a, b), q) = \frac{d(a, q) + d(b, q)}{2} \tag{4}$$

This is repeated until there is only one node left (the third step is not needed in the last iteration).

UPGMA

Also created by Sokal and Michener, UPGMA (unweighted pair group method with arithmetic mean) is highly similar to WPGMA. The only difference lies in that it keeps track of the number of elements in each cluster, making the result itself unweighted. In Step 3, UPGMA uses cluster weights, resulting in the following distance matrix update rule:

$$d((a, b), q) = \frac{d(a, q) \cdot m + d(b, q) \cdot n}{m + n} \tag{5}$$

where a consists of m and b consists of n elements. This is called *proportional averaging*.

Maximum parsimony

Maximum parsimony is an umbrella term for all the methods that are based on the principle

that a model is better than another if it implies fewer evolutionary events. In these methods, the number of evolutionary events is represented by a cost called the *parsimonious score*. There are two major categories: *exact* methods are the ones that consider the score of all possible trees, thus giving a global minimum, and *heuristic* methods are those that use some kind of technique to find a significant subset of all the trees and produces a somewhat sub-optimal result. As exact methods are often computationally infeasible, it is usually recommended to use a heuristic method that is let to run for a sufficient amount of time.

Fitch's algorithm can be used to calculate the score of an input tree: the first-pass of the algorithm starts from the leaves and goes up to the root in order to determine the parsimonious score and possible sets of bases at each internal node, while the second-pass goes from top to bottom and helps in determining a hypothetical tree. In each step of the first phase, the algorithm takes a look at the bases belonging to the children nodes (which consist of one element in the case of the leaves), and then, assigns their intersection to the parent node if they have bases in common. If they do not have any base in common, the parent node gets the union of the children's sets and the parsimonious score is increased by 1. This algorithm is repeated for each site (position in the sequences) in all input sequences and the score is summed up in the end.

There are a couple of popular heuristic methods that can help in exploring new trees, for example nearest neighbor interchange (NNI), subtree pruning and regrafting (SPR), or tree bisection and reconnection (TBR). NNI is the simplest one but is used for unrooted binary trees; the current tree may produce two new trees if all four branches connecting to one of its internal branches are swapped in every possible combination. In SPR, a certain subtree is removed from the current tree and gets reinserted elsewhere. TBR does the same as SPR, but instead of reinserting the subtree in the form it was in originally, it tries every possible way of connecting the two subtrees (similarly to NNI, it works for unrooted trees only). These methods are highly random, but they should be able to approximate the original distribution of trees given that they have enough time to run.

Branch-and-bound, an exact method that filters out a certain number of trees, may reduce the running time of the exhaustive search — an exact method that goes through all possible trees — significantly. This makes use of the fact that adding a new edge to a tree increases its parsimonious score, thus avoids going for trees that are bigger than the current one.

Maximum likelihood

Maximum likelihood, like in many other cases where probabilistic methods can be used, makes use of the basics of Bayesian statistics and looks at the probability of a certain sequence given a model (the model can be freely chosen). It may have a high algorithmic complexity as evaluating one sequence in itself may be computationally intensive, let alone multiple sequences. There are Bayesian methods (e.g., MCMC) that build upon ML; the major difference between these two groups of tree-building algorithms is that Bayesian methods take prior knowledge into account.

Self-growing tree algorithm or self-organizing tree algorithm

This one is based on a paper published in 1997 by Dopazo and Carazo [3] and is a combination of the Kohonen self-organizing map [4] and the growing cell structures algorithm of Fritzke [5]. This is an unsupervised learning network that starts out as a tree consisting of a small number of nodes (the paper mentioned two sister nodes) and then alternates between growing and adapting to the input sequences until it is fully grown and every taxonomical unit is assigned a proper place. The strictest exit condition guarantees that every input sequence is

associated to a unique cell.

First, the input sequences need to be converted to one-hot encoding. Secondly, the tree's initial node(s) and the corresponding weight matrices that the encoded input sequences will ultimately be compared to need to be created; in order to distinguish inner nodes from leaves, the authors of the paper mention that it would be best to call them nodes and cells, respectively. In each step, nodes are considered "closed", meaning that they cannot be assigned sequences after they transition from being a cell to a node. After the first few cells are initialized with numbers ranging from 0 to 1, the alternating phases begin to take place. The first phase is called adaptation, which is basically what characterizes Kohonen's self-organizing maps: the input points are compared to all available cells and those that are closest to each of the inputs are updated, along with their neighborhood (in the paper, finding the winning cell and updating its neighborhood is referred to as a *presentation*). After running this for a number of epochs, the inputs are mapped to appropriate positions in the output space, namely, the cells that they are the closest to according to a predefined metric. This distance metric is defined as

$$d_{S_jC_j} = \frac{\sum_{r=1}^{A} S_j(r,l) \cdot C_i(r,l)}{L} \tag{6}$$

where S_k is sequence k, C_i is cell i, A is the number of characters in the alphabet (in our case, four) and L is the length of the sequences.

The concept of neighborhood is trickier than when the output space is a 1D or 2D space; in the case where the winning cell's sister is a cell, the winning cell, the sister cell and their common ancestor (mother cell) all get updated — it should be noted that all three of them are assigned different update rates. However, if the sister is actually a subbranch of the tree and consists of multiple nodes and cells, only the winning cell is updated.

The second phase is the growing of the tree. As it is mentioned in the paper [3], "the growing of the network takes place in the cell having higher resources. This cell gives rise to two new descendant cells [...] and transforms itself into a node. At this moment, its upper node becomes a grandmother node, and thereafter, it does not receive any more updating. The two new cells are, in principle, identical to the node which generated them".

The tree stops growing when the resource value of the cell with the highest value is smaller than a predefined threshold — this threshold should be zero if the goal is to map each sequence to a unique cell, or rather, a sufficiently small number in order to avoid inaccuracies in numerical calculations. The concept of resource value is related to the distance between cells and input points, and is defined as

$$R_i = \frac{\sum_{k=1}^{K} d_{S_kC_i}}{K} \tag{7}$$

where R is the resource value belonging to cell i, S_k is sequence k, C_i is cell i and the summation is done over the K sequences associated to cell i.

The criteria used for monitoring the convergence of the network relies on the definition of the total error, ϵ, "defined as the summation of the distances of each sequence to the corresponding winning cell after an epoch" [3]. This helps in setting an exit condition for the adaptation process — e.g., the process can be ended when the relative increase of the error falls below a small threshold. The relative increase in error is

$$\left| \frac{\epsilon_t - \epsilon_{t-1}}{\epsilon_{t-1}} \right| < E \tag{8}$$

What is also interesting is that there is an extended update rule that is based on Kohonen's original formula:

$$C_i(\tau + 1) = C_i(\tau) + \eta_{t,\tau,i} \cdot [S_j - C_i(\tau)] \tag{9}$$

and $\eta_{t,\tau,i}$ is defined as a factor dependent on the number of cycles among others:

$$\eta_{t,\tau,i} = \alpha_i \cdot \frac{1-t}{M_t} \cdot (1 - b\tau) \tag{10}$$

where α_i is a coefficient dependent on the role of the current cell, t is the total number of presentations, M_t is the maximum number of presentations allowed ($\mu \times A \times L$) and b is the slope for the reduction of the interaction as the number of presentations, τ, increases within a cycle.

Implementation

I implemented the algorithms mentioned in the previous section using Python in combination with NumPy and Pandas. For reference and sanity tests, I used Biopython's implementation of some of the algorithms mentioned above. Since some of the data structures needed were not feasible to implement within the scope of the project, I decided to use Biopython's corresponding classes to facilitate the work. These consist of `Bio.SeqIO`, `Bio.Align.MultipleSeqAlignment`, `Bio.Alphabet` and `Bio.Phylo.BaseTree`. These were needed to read in data, create sequence alignments, fill in the gaps in the sequence alignments and create the trees in a form that can be visualized easily, respectively.

Regarding phylogeny, I decided to use unrooted trees because they are generally more accurate and more easily comparable in a lot of cases. Biopython's `BaseTree` can easily be visualized with `Bio.Phylo.draw`, a function that takes into account features like branch length and custom labels.

The methods I implemented are neighbor joining, UPGMA, WPGMA, maximum parsimony and SOTA. In the case of maximum parsimony, I implemented an exact algorithm that uses exhaustive search to find all the trees (with an additional option for branch-and-bound) and a heuristic algorithm that uses SPR. The self-organizing tree algorithm relies on a couple of hyperparameters; these need to be tuned before extensive testing.

Data

Experiments

Hyperparameters of SOTA

Comparison with regard to running time

Comparison with regard to accuracy

References

[1] The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, July 1987.

[2] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

[3] Joaquín Dopazo and José María Carazo. Phylogenetic Reconstruction Using an Unsupervised Growing Neural Network That Adopts the Topology of a Phylogenetic Tree. *Journal of Molecular Evolution*, 44(2):226–233, February 1997.

[4] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[5] Bernd Fritzke. Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, January 1994.

[6] Alexandre De Bruyn, Darren P. Martin, and Pierre Lefeuvre. Phylogenetic reconstruction methods: An overview. In *Methods in Molecular Biology*, pages 257–277. Humana Press, December 2013.

[7] Arjun B. Prasad, Marc W. Allard, and Eric D. Green and. Confirming the phylogeny of mammals by use of large comparative sequence data sets. *Molecular Biology and Evolution*, 25(9):1795–1808, May 2008.

[8] Martin Vingron, Jens Stoye, Hannes Luz, and Roland Wittler. Algorithms for phylogenetic reconstructions. February 2002-2009. Lecture notes from Bielefeld University.