Faster likelihood calculations on trees

Bret Larget*
Department of Mathematics
and Computer Science
Duquesne University
College Hall 440
Pittsburgh, PA 15282
USA
E-mail: larget@mathcs.duq.edu
Phone: (412) 396-6469
FAX: (412) 396-5197

Donald L. Simon
Department of Mathematics
and Computer Science
Duquesne University
College Hall 440
Pittsburgh, PA 15282
USA
E-mail: simon@mathcs.duq.edu
Phone: (412) 396-6472
FAX: (412) 396-5197

Key Words: phylogeny, likelihood, dynamic programming, DNA sequence analysis

Running head: Faster likelihood calculations on trees

Bret Larget and Donald Simon's research is supported by NSF Grant DBI-9723799.

Abstract

Calculating the likelihood of observed DNA sequence data at the leaves of a tree is the computational bottleneck for phylogenetic analysis by Bayesian methods or by the method of maximum likelihood. Because analysis of even moderately sized data sets can require hours of computational time on fast desktop computers, algorithmic changes that substantially increase the speed of the basic likelihood calculation are significant. It has long been recognized that the contribution to the likelihood at sites with identical patterns is the same and need only be computed once for each unique pattern. We note that sites whose patterns are not identical on the entire tree may be identical on subtrees, and hence partial likelihood calculations made for one site may be stored and used for calculations at another. The bookkeeping and memory requirements are large, but not too excessive for current desktop computers. Timed calculations on many genuine data sets indicate that the computational algorithm we present in this paper for likelihood calculations on trees result in decreases in running time by a factor ranging from 1.1 to 5.2 for data sets considered. There is reason to believe similar increases in speed would be more generally realized on alternative trees and data sets.

Introduction:

Saving partial calculations for the likelihood at one site for use in the calculation of the likelihood at another site can greatly decrease the computational time necessary to evaluate the likelihood of a phylogenetic tree. We have taken advantage of this in our free software package Bayesian Analysis in Molecular Biology and Evolution (BAMBE). To the best of our knowledge, all other publicly available free and commercial likelihood-based phylogeny programs do not do so, including likelihood-based programs in the widely used packages PHYLIP, FastDNAML, and PAUP*. We describe an algorithm for computing the likelihood of DNA sequences on trees, but the same principle could be used to increase the speed of likelihood calculations of amino acid sequences as well.

Method:

Our calculation of the likelihood of observing s aligned data sequences on a phylogenetic tree ψ modifies the recursive "pruning" algorithm described in Felsenstein (1983). We assume that ψ is a rooted binary phylogenetic tree and that all sites evolve independently according to a Markov base substitution model M that partitions the sites into c categories and includes separate submodels M_k for sites in category k, $1 \leq k \leq c$. Each submodel has its own set of parameters.

We designate the set of all unique site patterns within the kth category $\mathcal{U}_k$ and the union of these sets $\mathcal{U}$. Order the n_k unique site patterns in $\mathcal{U}_k$ into some fixed ordering $\mathcal{O}_k$ with the jth unique site pattern in the ordering representing $n_k(j)$ sites in the category. Let n be the total number of unique site patterns. The base in the sequence for taxon i in the jth unique site pattern in $\mathcal{U}_k$ is $x[i, j, k]$. The entire set of observed data is x.

According to M_k, we may calculate the transition probability $p_k(b_1, b_2, t)$ that a site in the kth category with base b_1 evolves to base b_2 along a branch with length t. We label the root of the tree ρ and assume a specific left/right orientation for the two subtrees rooted at each internal node so we may refer to the left child, right child, and parent of a node u by $\ell(u)$, $r(u)$, and $\sigma(u)$ respectively when these designations are appropriate. If u is a leaf, it is associated with taxon $s(u)$. Each node u except ρ is connected by a branch of length t_u to its parent. The subtree rooted at u is ψ_u. The pattern of data for all taxa at the jth site in $\mathcal{U}_k$ is $x[\cdot, j, k]$, while the same pattern of data restricted to taxa in ψ_u is $x[\cdot \restriction u, j, k]$.

For each node u except ρ we denote by $f_k(u, b, j)$ the probability of realizing the

pattern $x[\cdot \restriction u, j, k]$ given the parent of u has base b at the jth unique site pattern in $\mathcal{U}_k$. This partial likelihood is not identical to the "fragmentary likelihood" described in Felsenstein (1983) because it conditions on the parent's base instead of the base at u. This change decreases the number of calculations when reusing calculations from other sites. The partial likelihoods satisfy the recursive relationships

$$f_k(u,b,j) = \begin{cases} p_k(b, x[s(u), j, k], t_u) & \text{if } u \text{ is a leaf} \\ \sum_{a \in \mathcal{B}} p_k(b,a,t_u) f_k(\ell(u), a, j) f_k(r(u), a, j) & \text{if } u \text{ is an internal non-root node} \end{cases} \tag{1}$$

where $\mathcal{B}$ is the set of four nucleotide bases. We express the log likelihood of the tree as

$$\log L(x|\psi, M) = \sum_{k=1}^{c} \left(\sum_{j=1}^{n_k} n_k(j) \log \left(\sum_{b \in \mathcal{B}} \pi_b^{(k)} f_k(\ell(\rho), b, j) f_k(r(\rho), b, j) \right) \right) \tag{2}$$

where $\pi_b^{(k)}$ is the stationary probability of base b in M_k.

The values of $f_k(u,b,j)$ and $f_k(u,b,j')$ will be equal whenever $x[\cdot \restriction u, j, k] = x[\cdot \restriction u, j', k]$. For each category k and node u there is an equivalence class of sites with identical restricted site patterns. We choose the canonical site of this equivalence class to be the one that comes first in ordering $\mathcal{O}_k$. We separate the equivalence classes into those with constant and non-constant restricted site patterns and order the $e_k(u)$ equivalence classes from non-constant restricted site patterns according to the ordering in $\mathcal{O}_k$ of the corresponding canonical sites. We denote the common partial likelihoods for all sites in the ith equivalence class with non-constant restricted site patterns to have values $g_k(u,b,i)$ where $1 \leq i \leq e_k(u)$. If the restricted site pattern is constant for base b', we denote the common partial likelihood value $h_k(u,b,b')$.

The partial likelihoods are determined by the canonical partial likelihoods by

$$f_k(u,b,j) = \begin{cases} g_k(u,b,o_k(u,j)) & \text{if } x[\cdot'|u,j,k] \text{ is non-constant} \\ h_k(u,b,b') & \text{if } x[\cdot'|u,j,k] \text{ is constant for base } b' \end{cases} \tag{3}$$

where $o_k(u,j)$ is the equivalence class corresponding to non-constant site pattern $x[\cdot'|u,j,k]$. Our algorithm calculates and stores the canonical partial likelihoods so that the partial likelihoods $f_k(u,b,j)$ are calculated only when site j is canonical for category k and node u. Figure 1 illustrates the ideas presented to this point in the paper.

[Figure 1 should appear about here]

The Algorithm:

We represent a site with a bit vector with four bits per taxon, one for each choice of base. Using the ordering A,G,C,T, the second site in Figure 1, 'ATTTGA', has bit vector '1000 0001 0001 0001 0100 1000'. A gap would be represented by '1111', while the symbol 'R' for an undetermined purine would be represented by '1100'. The algorithm implemented in BAMBE reuses calculations from sites with gaps and indeterminate bases that we supress in this paper to increase the clarity of the presentation of the main idea.

At each node u we define a mask, m_u, which is a bit vector with four bits per taxon that indicates the taxa in the subtree rooted at u. The mask for node u in Figure 1 whose subtree consists of taxa 1 and 5 is '1111 0000 0000 0000 1111 0000'. With the notation "$|_{\text{or}}$" to mean bitwise inclusive or and "&" to mean bitwise and, we note that

$$m_u = m_{\ell(u)} \,|_{\text{or}}\, m_{r(u)} \tag{4}$$

for any non-leaf node u. In addition, sites restricted to the subtree rooted at node u satisfy the equations

$$\begin{aligned} x[\cdot \,|\, u, j, k] &= x[\cdot \,|\, \sigma(u), j, k] \;\&\; m_u & (5) \\ &= x[\cdot, j, k] \;\&\; m_u & (6) \end{aligned}$$

The only exception is that the right hand side of Equation (5) is undefined when $u = \rho$.

We describe the general approach of the algorithm here. A complete description with full implementation details would be excessively long. Readers interested in these details are welcome to examine the source code in the BAMBE package.

FIND TREE LIKELIHOOD:

Step 1. Find and count the number of occurences of unique site patterns in each category.

Step 2. Propogate masks recursively from leaves toward the root.

Step 3. Find canonical sites for each node recursively from the root toward the leaves.

Step 4. Calculate partial likelihoods for each canonical site and each category at each non-root node.

Step 5. Evaluate the likelihood of the tree by Equation (2).

We give more details on each step of the algorithm.

Step 1. FIND UNIQUE SITE PATTERNS:

This step is executed only once, at the time the data is read, because the calculations do not depend on the tree or the model. The algorithm calculates $n_k(j)$ for $1 \leq k \leq c$, $1 \leq j \leq n_k$ and creates bit vectors $z_k^{(1)}(\rho)$, $z_k^{(2)}(\rho)$, $\ldots$, $z_k^{(e_k(\rho))}(\rho)$ for the unique non-constant site patterns in each category. The collection of such bit vectors in category k for a node u is $\mathcal{Z}_k(u)$.

Step 2. PROPOGATE MASKS:

1. Set masks m_u for all leaf nodes u.

2. Recurse from leaves to the root to find m_u for all nodes u by Equation (4).

Step 3. Find Canonical Sites:

1. For each category k:

   (a) Beginning with the children of the root, recurse from parents to children to find the canonical non-constant sites at each node and link the canonical sites of the child to the corresponding canonical sites of the parent. At each non-root node u:

      i. In a single pass through $\mathcal{Z}_k(\sigma(u))$, create a list of (possibly not unique) restricted non-constant site patterns at node u by applying Equation (5) to each $z \in \mathcal{Z}_k(\sigma(u))$. Suppose z corresponds to the jth unique site in category k. If the restricted site pattern z & m_u is constant for base b', point $f_k(u, j, b)$ to the value in $h_k(u, b, b')$ for each b. Otherwise, add the restricted site pattern to the list.

      ii. Sort the list of remaining non-constant restricted sites, if any. Pass through the sorted list collecting all identical site patterns under the restriction and for all such patterns which correspond to say, the j'th unique site in category k, point $f_k(u, j', b)$ to $g_k(u, b, j^*)$ where j^* is the unique site in category k that corresponds to the first occurrence of the site pattern.

Step 4: Calculate Partial Likelihoods:

1. For each category k:

   (a) Recurse from the leaves to the children of the root and evaluate the canonical partial likelihoods. For node u:

      i. Evaluate $h_k(u, b, b')$ for each b and b'.

      ii. Evaluate $g_k(u, b, j)$ for each canonical j with respect to category k and node u.

Step: 5 Evaluate the Likelihood:

1. For each site category k:

   (a) Evaluate the inner sum in Equation (2).

2. Evaluate the outer sum in Equation (2) to obtain the log likelihood of the tree.

Examples and Discussion:

Sorting restricted site patterns at each node is the most costly part of the algorithm. For genuine data sets, many of these restricted site patterns will be constant. By separating the constant restricted site patterns before sorting, we create a smaller list to sort and obtain greater efficiency.

In practice, the likelihood of each of a large sequence of trees will be evaluated. Steps 2 and 3 are only necessary when the current tree topology differs from the previous one. Further efficiency may be obtained by recalculating masks, finding canonical sites, and calculating partial likelihoods only for portions of the tree where these change. For speed comparisons, we make two tests: the time required to complete steps 2–5 and the time required to complete steps 4–5 in FIND TREE LIKELIHOOD. In the program `mcmc` in BAMBE, the evaluation routine executes steps 2–5 for the entire tree, after the GLOBAL tree update algorithm, and executes only steps 4–5 after updating model parameters while leaving the tree fixed .

To test the speed of our algorithm, we calculated the likelihood of trees sampled from the posterior distribution for several data sets. The data sets we studied were: 32 cichlid fish with aligned sequences of 1044 base pairs from protein coding mitochondrial DNA (Kocher, *et al.*, 1995; Mau, Newton, Larget, in press); 13 pocket gophers and 13 parasitic lice with aligned sequences of 379 base pairs from mitochondrial DNA encoding cytochrome oxidase I (Hafner *et al.*, 1994; Newton, Mau, Larget, in press); and 31 mammals (14 whales and 17 artiodactyles) using 1140 base pairs of aligned cytochrome b (Adachi and Hasegawa, 1995; Larget and Simon, submitted). A Bayesian analysis of

each data set with the program `mcmc` in the software package BAMBE produced a tree and estimated model parameters.

For each data set and associated tree, we generated fifteen simulated data sets using nucleotide base probabilites and transition/transversion parameters set to the means of their posterior distributions. For five of the fifteen simulated data sets for each tree we slowed the rate of evolution by a factor of ten, for five we used the mean overall rate from the posterior distribution, and for five we increased the rate by a factor of ten. We also analyzed the genuine data. In every case, we timed the recalculation of the likelihood of the tree 5,000 times using FIND TREE LIKELIHOOD steps 2–5, using FIND TREE LIKELIHOOD steps 4–5, and using the standard algorithm that does not share any calculations between unique sites patterns and computes the contribution for each unique site pattern only once.

The ratios of time using the standard algorithm to time using FIND TREE LIKELIHOOD are shown in Table 1. The median speed increase in this small study is by a factor of 2.5.

[Table 1 should appear about here.]

We have demonstrated that FIND TREE LIKELIHOOD significantly increases the speed of likelihood calculations on several examples. Our algorithm will show the most improvement on large trees with a rather low overall substitution rates, so there are many distinct near constant site patterns. Even on smaller trees with extremely high substitution rates, our algorithm is at least three times as fast as the standard algorithm when executing

Data set	Genuine Data		Slow Rate		Medium Rate		Fast Rate	
	2–5	4–5	2–5	4–5	2–5	4–5	2–5	4–5
Fish	1.5	4.3	1.9	3.7	1.5	4.4	0.9	3.1
Gophers	1.2	3.4	1.7	2.6	1.2	3.6	0.9	3.3
Lice	1.1	3.2	1.8	3.3	1.1	3.5	0.9	3.2
Whales	1.2	5.2	2.3	6.0	1.3	5.6	0.8	3.5

Table 1: **Increases in speed across several data sets.** Each number is the ratio of the time required to calculate a tree likelihood 5,000 times without the dynamic programming algorithm to the time required with the algorithm using either steps 2–5 or 4–5. All calculations in a row were for data on the same tree sampled from the posterior distribution after a Bayesian analysis of the genuine data. Each number in the genuine data column represents a single data set. The numbers in the fourth through sixth columns are medians of five different data sets generated according to the HKY85 model (Hasegawa *et al.*, 1985). The slow rate, medium rate, and fast rates are respectively one tenth, one, and ten times the posterior mean rate in the Bayesian analysis. There was very little variabilitity in the time necessary for calculations over simulated data sets from the same tree. The reported median value usually differed from observed maximums and minimums by 0.1 and never differed by more than 0.2.

steps 4–5 for the examples we considered.

The precise size of the increase in speed is specific to data sets and trees. We allocate extra storage arrays for partial likelihood calculations of

$$\text{sizeof}(\text{double}) \cdot ((2 \cdot s - 1) \cdot (8 \cdot n + 32 \cdot c)) + \text{sizeof}(\text{double} *) \cdot (2 \cdot s - 1) \cdot n.$$

For our largest data set, that of whales, this storage space requirement was less than 2.4 Mbytes. With standard desktop computers commonly available today, these memory requirements are not excessive, even for problems with substantially more sites and taxa than we have considered here.

It is clear that dynamic programming can greatly increase the speed of likelihood calculations on trees at reasonable costs of memory, bookkeeping, and code complexity. We advocate that other developers of software for the analysis of molecular sequence data by likelihood-based methods use and/or modify the algorithm presented here to increase the speed of their programs.

Acknowledgments

We acknowledge the support of the National Science Foundation through grant DBI-9723799. Work on this manuscipt began while the first author participated in the program "Biomolecular function and evolution in the context of the genome project" at the Isaac Newton Institute for Mathematical Sciences at Cambridge University in August, 1998. He thanks the NSF for supporting his participation in this program. We thank Masami Hasegawa, who provided the whale, fish, gopher, and louse data sets, and Ziheng Yang, who distributes the primate data set in his PAML software package.

References:

Adachi, J., and Hasegawa, M. 1995. Phylogeny of whales: dependence of the inference on species sampling. *Molecular Biology Evolution* 12, 177–179.

Felsenstein, J. 1995. PHYLIP (Phylogeny Inference Package), Version 3.572c. Computer program distributed by the University of Washington, Seattle.

Hafner, M.S., Sudman, P.D., Villablanca, F.X., Spradling, T.A., Demastes, J.W., and Nadler, S.A. 1994. Disparate rates of molecular evolution in cospeciating hosts and parasites. *Science*, 265, 1087–1090.

Kocher, T.D., Conroy, J.A., McKaye, K.R., Stauffer, J.R., and Lockwood, S.F. 1995. Evolution of NADH dehydrogenase subunit 2 in East African cichlid fish. *Molecular Phylogenetics and Evolution* 4, 420–432.

Mau, B., Newton M.A., And Larget, B. 1996. Bayesian phylogenetic inference via Markov chain Monte Carlo methods. *Biometrics* in press.

Newton, M., Mau, B., And Larget, B. 1997. Markov chain Monte Carlo for the Bayesian analysis of evolutionary trees from aligned molecular sequences. *Proceedings of the AMS-IMS-SIAM joint summer research conference on statistics and molecular biology.*

Simon, D.L., and Larget, B. 1998. Bayesian Analysis in Molecular Biology and Evolution (BAMBE), version 1.01 beta. Department of Mathematics and Computer Science, Duquesne University. (http://www.mathcs.duq.edu/larget/bambe.html).

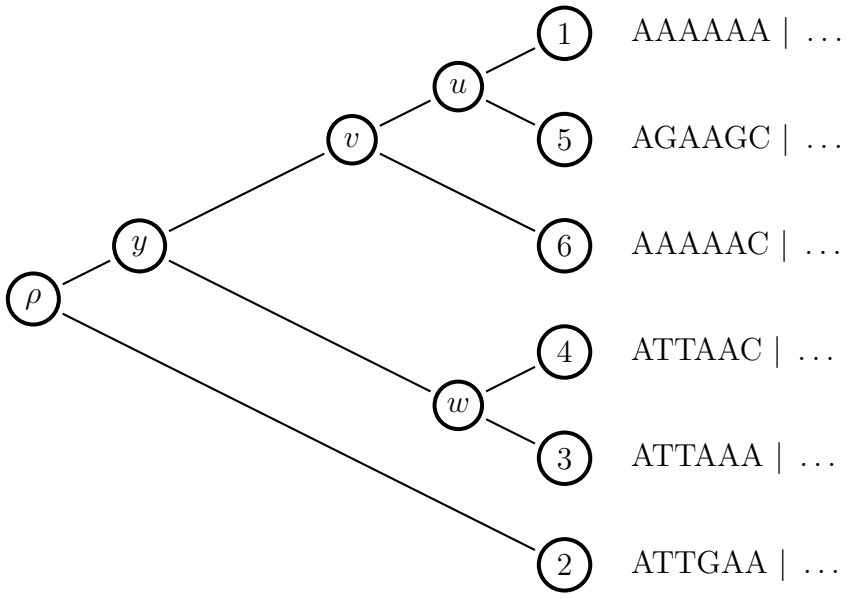

Figure 1: **An example tree and sequence data.** For this tree, $n(1) = 6$ and we order the unique sites in $\mathcal{U}_1$ from left to right. The site pattern $x[\cdot, 3, 1]$ = 'ATTTAA' restricted to ψ_v is $x[\cdot \restriction v, 3, 1]$ = 'A***AA'. The three canonical restricted site patterns in category 1 for node v are 'A***AA','A***GA', and 'A***CC'}. The three sites with non-constant restricted site patterns are mapped into $m_1(v) = 2$ equivalence classes by $o_1(v, 2) = o_1(v, 5) = 1$ and $o_1(v, 6) = 2$. The partial likelihoods satisfy $f_1(v, b, 1) = f_1(v, b, 3) = f_1(v, b, 4) = h_1(v, b, A)$, while $f_1(v, b, 2) = f_1(v, b, 5) = g_1(v, b, 1)$ and $f_1(v, b, 6) = g_1(v, b, 2)$ for each base b. While we display a tree which obeys the molecular clock, the algorithm we present is unchanged for trees without a molecular clock.