Second Order Descent Methods for Estimating Branch Lengths in Phylogenetic Trees

Zach Stephens
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
zstephe2@illinois.edu

I. Introduction

The goal of statistical phylogenetics is to find the most probable relationship among a group of species (taxa) given their aligned DNA sequence data. Evolutionary relationships are described using *phylogenetic trees*, a bifurcating tree structure that encapsulates the hierarchal ancestry of a collection of organisms.

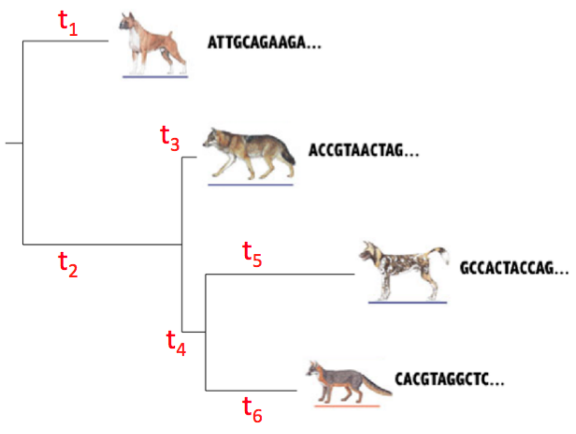


Figure 1. Example Phylogenetic Tree

A topology τ consists of the tree's structure and the distance between each node. The length of each branch in the tree ($\{t_1, \ldots, t_6\}$ in figure 1) is proportional to the *genetic distance* between a species and its ancestor. Larger distances imply more genetic dissimilarity (more nucleotide mutations) between the two sequences. Branch lengths can be any non-negative real number.

Finding an optimal topology is a computationally daunting task. The number of possible structures explodes as a factorial function of the number of taxa in the tree, thus it has become popular to approach the problem using heuristics and Monte-Carlo methods [1]. Sampling algorithms have been developed to traverse the space of possible tree structures [2], but in order to assess the likelihood of each candidate structure it is necessary to find the branch lengths that maximize the likelihood given that particular structure. This report explores this optimization sub-problem, that is, estimating the optimal branch lengths given aligned DNA sequence data and a fixed tree structure.

This work expands upon previously explored gradient descent methods (See ECE551 final project) which suffered from slow convergence and inefficient (also numerically unstable) gradient computations. This work seeks to address convergence issues by incorporating second order information, as well as a more stable and efficient derivative formulation.

The following sections of this report include an explicit formulation of the problem, additional background information, algorithm implementation details, and expressions for gradient / hessian elements. The algorithm is run on simulated and real-world datasets and results are discussed.

II. Background / Problem Formulation

For DNA sequence data each species is represented by a string of the characters A, C, G, T, standing for the nucleotides *Adenine, Guanine, Cytosine*, and *Thymine*, respectively. The dataset is arranged into an observation matrix $\mathbf{X}$ where each row designates a species, and each column is an observation site x consisting of the nucleotide present at that point in the string for each species.

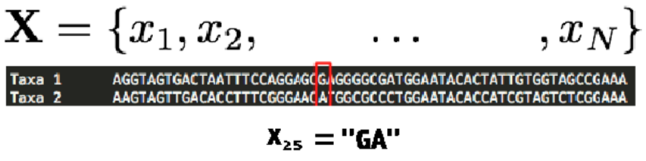

$\mathbf{X} = \{x_1, x_2, \quad \ldots \quad , x_N\}$
Taxa 1 AGGTAGTGACTAATTTCCAGGAGCGAGGGGCGATGGAATACACTATTGTGGTAGCCGAAA
Taxa 2 AAGTAGTTGACACCTTTCGGGAACATGGCGCCCTGGAATACACCATCGTAGTCTCGGAAA
x_{25} = "GA"

Figure 2. Example Observation Matrix

Because this work assumes the tree structure is fixed, let τ denote solely the collection of branch lengths:

$\{t_i, \dots, t_{2(T-1)}\}$. So our goal is to find:

$$\begin{aligned}\tau_{MLE} &= \arg\max P(\mathbf{X} \mid \tau)\\ &= \arg\max \log P(\mathbf{X} \mid \tau)\end{aligned}$$

$P(\mathbf{X} \mid \tau)$ is also called the *Phylogenetic Likelihood Function*, or PLF. Assuming independence among sites we can write the log-likelihood as the summation:

$$\log P(\mathbf{X} \mid \tau) = \sum_{x \in \mathbf{X}} \log P(x \mid \tau)$$

The likelihoods $P(x \mid \tau)$ can be computed in the same fashion as a bayesian network with the same structure, that is, each node in the tree is conditionally independent given its ancestor. The joint distribution of the sequence data and the inner nodes is given as:

$$P(x, \text{nodes} \mid \tau) = \prod_{\text{nodes}} P(\text{node}_i \mid parent(\text{node}_i), t_i)$$

For example, consider the simple two-species tree:

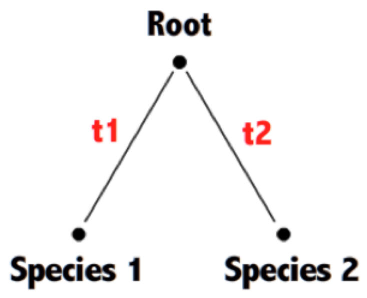


Figure 3. Two-Species Tree

Which has joint the distribution:

$$P(x, n_{root}, \mid \tau) =$$

$$P(n_{root}) \cdot P(x_1 \mid n_{root}, t_1) \cdot P(x_2 \mid n_{root}, t_2)$$

We can arrive at the desired $P(x \mid \tau)$ by marginalizing over the inner nodes, in this example just n_{root}:

$$P(x \mid \tau) = \sum_{n_{root} \in \{A,C,G,T\}} P(x, n_{root} \mid \tau)$$

We now have the likelihood in terms of things we can compute. $P(n_{root} = a) = P(a)$ is the prior probability of seeing nucleotide a in the dataset and can be estimated directly from the data. $P(n_i = a \mid n_j = b, t_i)$ is the probability of nucleotide b mutating into nucleotide a over time t_i. This term is evaluated by specifying a Markov model of nucleotide substitution. In this work the simple Jukes-Cantor '69 [3] model is chosen, this model assumes nucleotides mutate into each other with an equal rate:

$$\mathbf{Q} = \begin{pmatrix} -3a & a & a & a \\ a & -3a & a & a \\ a & a & -3a & a \\ a & a & a & -3a \end{pmatrix}$$

Each element Q_{ij} of the matrix describes the rate that nucleotide i mutates into j. So therefor:

$$P(n_i = a \mid n_j = b, t_i) = e^{Qt_i}{}_{ba} = P(t_i)_{ba}$$

$P(t_i)$ is called the *transition probability matrix*. Using this matrix we can rewrite $P(x, \tau)$ for our two-species example in a vectorized form:

$$P(x \mid \tau) = \pi^T (P(t_1)L_1 \circ P(t_2)L_2)$$

$$\pi = \begin{pmatrix} P(A) \\ P(C) \\ P(G) \\ P(T) \end{pmatrix} L_i = \begin{pmatrix} P(n_i = A \mid parent(n_i), t_i) \\ P(n_i = C \mid parent(n_i), t_i) \\ P(n_i = G \mid parent(n_i), t_i) \\ P(n_i = T \mid parent(n_i), t_i) \end{pmatrix}$$

Where $\circ$ denotes the Hadamard product. The vectors L_i are called *partial likelihood vectors*.

This leads us to Felsenstein's celebrated dynamic programming algorithm [4] [5] that allows us to compute $P(x \mid \tau)$ for an arbitrary tree structure by traversing the tree in post-order (from leaves to root), and recursively computing:

$$Li = P(t_{c_1})L_{c_1} \circ P(t_{c_2})L_{c_2}$$

Where c_1 and c_2 denote the children of node i. The partial likelihood vectors for leaf nodes are determined directly by the sequence data, for example if node$_i$ is a leaf node:

$$\text{if } x_i = A: \quad \text{if } x_i = C: \quad \text{if } x_i = G: \quad \text{if } x_i = T:$$

$$L_i = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad L_i = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad L_i = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad L_i = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Once the algorithm traverses all the way to the root, the likelihood $P(x \mid \tau)$ can be computed:

$$P(x \mid \tau) = \pi^T L_{root}$$

$$\log P(\mathbf{X} \mid \tau) = \sum_{x \in X} \log(\pi^T L_{root})$$

Now that we have an explicit formulation for the log-PLF we can obtain expressions for gradient and hessian elements for use in an iterative descent algorithm.

III. Algorithm

In general there is no closed-form expression for τ_{MLE}, so common approaches include iterative methods such as gradient descent, Newton's method, and expectation maximization [6] [7]. This work explores Newton's method approaches, that is:

$$\tau \leftarrow \tau + \eta H^{-1} \nabla_\tau \log \text{PLF}(\mathbf{X}, \tau)$$

To use this update equation we first need to procure expressions for the gradient elements $\frac{d \log \text{PLF}}{dt_i}$ and the hessian elements $\frac{d^2 \log \text{PLF}}{dt_i^2}$, $\frac{d^2 \log \text{PLF}}{dt_i t_j}$. Furthermore, to impose the non-negativity constraint we clip any negative assignment:

$$t_i \leftarrow \max(0, t_i + \eta H^{-1} \tfrac{d}{dt_i} \log \text{PLF}(\mathbf{X}, \tau))$$

A. Analytic Expression for Gradient and Hessian

By inspection it is easy to see that the only component of the PLF that depends on t_i is its transition probability matrix $P(t_i) = e^{Qt_i}$, which has derivatives $P'(t_i) = Qe^{Qt_i}$ and $P''(t_i) = Q^2 e^{Qt_i}$. Returning to our two-species example:

$$P(x \mid \tau) = \pi^T (P(t_1)L_1 \circ P(t_2)L_2)$$

We see that the derivative with respect to t_1 is simply:

$$\tfrac{d}{dt_1} P(x \mid \tau) = \pi^T (P'(t_1)L_1 \circ P(t_2)L_2)$$

We see that the derivative of the PLF with respect to a particular branch length is just the PLF function called with a slightly different topology. Let $\tilde{\tau}_i$ denote a topology τ where $\tilde{P}(t_i) \leftarrow P'(t_i)$. Similarly let $\tilde{\tau}_{ii}$ denote τ with $\tilde{P}(t_i) \leftarrow P''(t_i)$, and $\tilde{\tau}_{ij}$ denote τ with $\tilde{P}(t_i) \leftarrow P'(t_i)$ and $\tilde{P}(t_j) \leftarrow P'(t_j)$. The forms of the derivatives in general are:

$$\tfrac{d}{dt_i} P(x \mid \tau) = P(x \mid \tilde{\tau}_i)$$

$$\tfrac{d^2}{dt_i^2} P(x \mid \tau) = P(x \mid \tilde{\tau}_{ii})$$

$$\tfrac{d^2}{dt_i t_j} P(x \mid \tau) = P(x \mid \tilde{\tau}_{ij})$$

But what we are really looking for are the derivatives of the log-PLF, so by applying some basic calculus:

$$\tfrac{d}{dt_i} \log P(x \mid \tau) = \frac{P(x \mid \tilde{\tau}_i)}{P(x \mid \tau)}$$

$$\tfrac{d^2}{dt_i^2} \log P(x \mid \tau) = \frac{P(x \mid \tilde{\tau}_{ii}) P(x \mid \tau) - P(x \mid \tilde{\tau}_i)^2}{P(x \mid \tau)^2}$$

$$\tfrac{d^2}{dt_i t_j} \log P(x \mid \tau) = \frac{P(x \mid \tilde{\tau}_{ij}) P(x \mid \tau) - P(x \mid \tilde{\tau}_i) P(x \mid \tilde{\tau}_j)}{P(x \mid \tau)^2}$$

B. Finite Difference Approximation

Even though the analytic expression for the gradient is fairly cheap to compute, finite-difference methods were also explored. The central difference approximations were used:

$$\text{Let } f(x, \tau) = \log P(x \mid \tau)$$

$$\tfrac{d}{dt_i} f(x, \tau) \approx \frac{f(x, \tilde{\tau}_{i_+}) - f(x, \tilde{\tau}_{i_-})}{h}$$

$$\tfrac{d^2}{dt_i^2} f(x, \tau) \approx \frac{f(x, \tilde{\tau}_{i_+}) - 2f(x, \tau) + f(x, \tilde{\tau}_{i_-})}{h^2}$$

$$\tfrac{d^2}{dt_i t_j} f(x, \tau) \approx \frac{f(x, \tilde{\tau}_{i_+ j_+}) - f(x, \tilde{\tau}_{i_+ j_-}) - f(x, \tilde{\tau}_{i_- j_+}) + f(x, \tilde{\tau}_{i_- j_-})}{h^2}$$

Where $\tilde{\tau}_{i_+}$ denotes topology τ with $\tilde{P}(t_i) \leftarrow P(t_i + \frac{h}{2})$, and $\tilde{\tau}_{i_-}$ denotes τ with $\tilde{P}(t_i) \leftarrow P(t_i - \frac{h}{2})$. It has been found that the hessian estimates are inaccurate when the branch values approach zero [8], so for this work the finite difference approximation is only used as an alternative way to compute the gradient.

C. Levenberg-Marquardt Approximation for the Hessian

The Levenberg-Marquardt approximation was included in these experiments as a computationally efficient estimate of the hessian matrix:

$$g_n = \nabla_\tau \log P(x \mid \tau) \qquad H \approx \sum_{n=1}^{N} g_n (g_n)^T$$

The inverse to this matrix can be found efficiently via the matrix inversion lemma:

$$H_0^{-1} = \alpha I$$

$$H_i^{-1} = (H_{i-1})^{-1} + \frac{(H_{i-1}^{-1} g_i)(H_{i-1}^{-1} g_i)^T}{1 + (H_{i-1}^{-1} g_i)^T g_i}$$

D. Computational Complexity

Below is a brief comparison of the computational complexity of each approach, where **T** is the number of taxa, and **N** is the number of sites (length of DNA sequences). Each invocation of $P(x \mid \tau)$ is $O(T)$.

Gradient Computation:

- Analytic: $2N(T-1)$ PLF invocations.
- Finite-Difference: $4N(T-1)$ PLF invocations

Hessian Computation:

- Analytic: $3N(T-1)^2$ PLF invocations + $O(T^3)$ inversion
- L-M Approx: $3N$ matrix-vector multiplies ($O(T^2)$)

IV. Experimental Results

The descent algorithm was run on synthetic and real datasets to compare the effects of incorporating analytic and estimated second-order information. The synthetic data was created by randomly constructing a reference sequence of a desired length (each nucleotide chosen with equal probability), then randomly mutating bases with a specified probability for each synthetic species. Thus we would expect the most probable tree topology to be one that reflects each species sharing a common ancestor. In these first experiments such a dataset was constructed with $T = 4$, $N = 2000$.

Figure 4 shows a comparison of first order and second order descent approaches given the same initialization. Figures 5 shows the initialized topology, and Figure 6 shows the final topology reached by the top-performing method (finite-difference gradient descent).

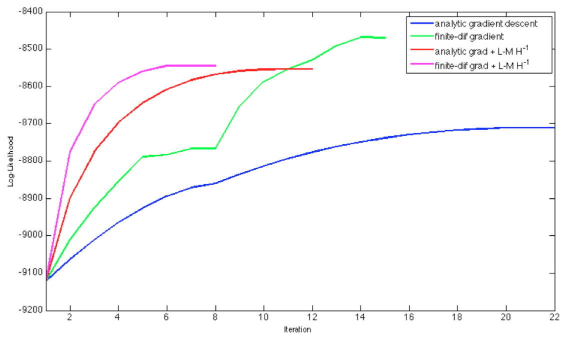


Figure 4. Log-Likelihood (LL) vs. iteration comparison.

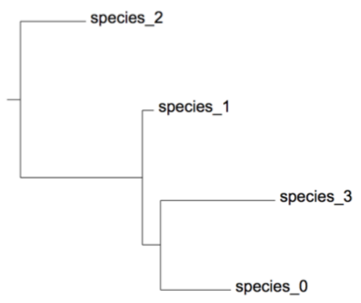


Figure 5. Randomly initialized tree. LL: -9122

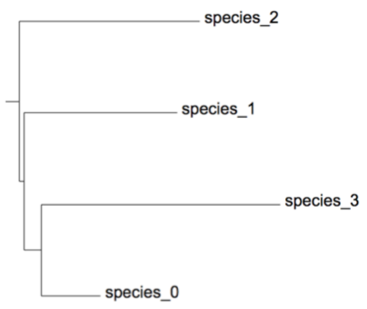
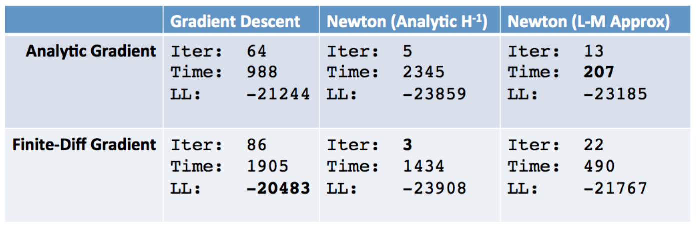


Figure 6. Final tree. LL: -8470

To further compare various attributes of each method every combination of gradient and hessian computations was performed on the same initialization of another synthetic dataset ($T = 20$, $N = 1000$, $LL_0 = -23966$). Figure 9 shows the iteration count, computation time, and final log-likelihood for each method after it was run to convergence. The algorithm was halted after the change in log-likelihood between iterations fell below a threshold (0.01). To prevent ”overstepping” into poor regions of the parameter space the step size was recursively halved if the log-likelihood is decreased after an iteration.



	Gradient Descent	Newton (Analytic H^{-1})	Newton (L-M Approx)
Analytic Gradient	Iter: 64 Time: 988 LL: -21244	Iter: 5 Time: 2345 LL: -23859	Iter: 13 Time: **207** LL: -23185
Finite-Diff Gradient	Iter: 86 Time: 1905 LL: **-20483**	Iter: **3** Time: 1434 LL: -23908	Iter: 22 Time: 490 LL: -21767

Figure 7. Comparison of gradient / hessian methods

As expected the methods that incorporated the hessian converged in an order of magnitude fewer iterations, though with a significant additional computational cost per iteration The usage of analytic gradients and hessian led to the fastest convergence, but surprisingly methods that used approximations ended up converging to a better optima.

In addition to simulated data the algorithms were run on a real dataset consisting of shrub DNA from the Pinaceae family. For this dataset $T = 20$, $N = 2838$. Each combination of gradient / hessian computations was run as above, and the method that resulted in the highest final log-likelihood was chosen. Surprisingly it was again the finite-difference gradient that outperformed the others.

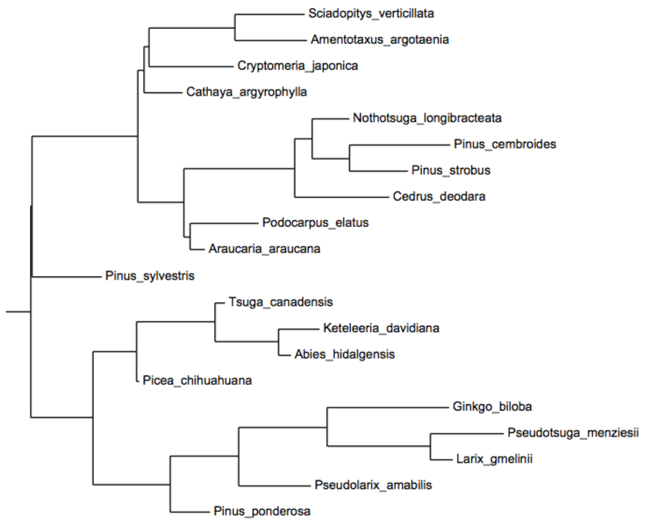


Figure 5. Randomly initialized tree: Pinaceae dataset. LL: -24146

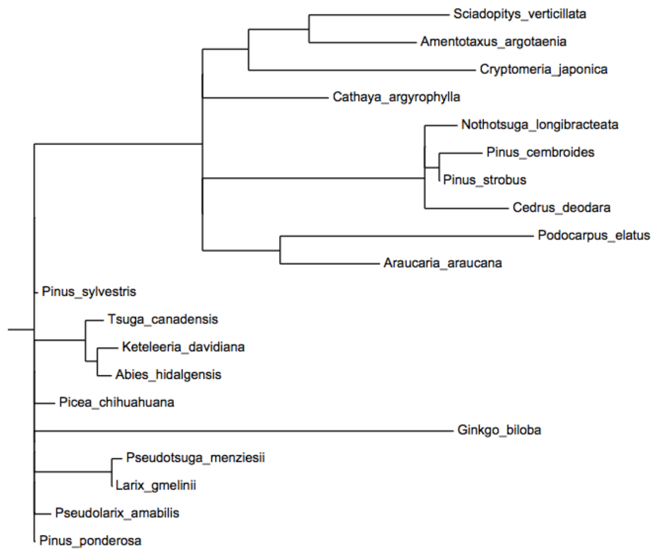


Figure 6. Final tree: Pinaceae dataset. LL: -17461

The resulting tree elucidates the relationships between the species more clearly than the synthetic datasets. There are more obvious hierarchal groupings, most notably between the two clusters of species separated by the root. The significant increase in likelihood is reassuring that the descent algorithm is exploring a decent portion of the parameter space.

V. Conclusions

This report investigated the application of second order descent methods for estimating the optimal branch lengths in a phylogenetic tree. Expressions for the derivatives of the log-PLF were presented and used in an iterative algorithm that was successfully run on synthetic and real-world data.

It came as no surprise that the incorporation of the hessian into the standard gradient descent update equation led to an order of magnitude faster convergence at the expense of additional computation time. A compromise was found by using the Levenberg-Marquardt approximation of the hessian, resulting in fast convergence with low computational cost. It is very interesting however, that by using the finite-difference approximation for the gradient the algorithm often ended up at a better local optima when compared to using the analytic expressions. One possible explanation is that by heading ”not exactly” in the direction of the gradient we might be more likely to deviate from the closest local maxima and more thoroughly explore the parameter-space, eventually converging on an even better optima.

References

[1] Huelsenbeck, J. P. and F. Ronquist. *MRBAYES: Bayesian inference of phylogeny*. Bioinformatics 17:754-755. 2001.

[2] Larget, B. and Simon, D. *Markov Chain Monte Carlo Algorithms for the Bayesian Analysis of Phylogenetic Trees*. 16 (6). pp. 750-9. 1999.

[3] T. Jukes and C. Cantor, *Evolution of Protein Molecules*. New York: Academic Press. 21-132, 1969.

[4] J. Felsenstein, *Phylogenies and the comparative method*. American Naturalist 125: 1-15, 1985.

[5] J. Felsenstein, *Evolutionary trees from DNA sequences: a maximum likelihood approach*. Journal of Molecular Evolution 17: 368-376, 1981.

[6] I. Mayrose et al., *Best branch lengths using expectation maximization*. 2005.

[7] A. Siepel. *Expectation maximization for combined phylogenetic and hidden markov models*. 2002.

[8] M. Reis, Z. Yang *Approximate Likelihood Calculation on a Phylogeny for Bayesian Estimation of Divergence Times*.
Mol. Biol. Evol. 28(7): 2161-2172, 2011.