BISTRO: Bayesian Importance Sampling for Phylogenetic Inference

BL,CSL

Phylogenetics studies the evolutionary relationships between species, typically visualized in a tree or phylogeny. DNA sequences are used as input data to estimate the phylogenetic tree that links species through common ancestry. This estimation can be performed through a variety of methods such as maximum parsimony (reference), maximum likelihood (reference) or bayesian inference (reference).

In the Bayesian framework, the goal is to obtain the posterior distribution of trees, branch lengths and other model parameters. However, this posterior distribution does not have an explicit form. Furthermore, it is impossible to obtain samples directly from this distribution, so MCMC methods are widely used (references) to generate a Markov chain with the posterior distribution as stationary distribution.

The posterior sample generated by MCMC can then be used to do inference on parameters of interest, such as identifying trees or splits with higher posterior probabilities, or computing posterior means and credibility intervals of numerical model parameters.

The downside of MCMC methods is that its performance rapidly deteriorates as the parameter space increases due to slow or poor mixing. In phylogenetic analyses, the tree space increases dramatically with the number of species. Then, MCMC methods need a very big chain in order to navigate the huge tree space. In addition, the MCMC moves keep most of the parameters constant, when proposing a new state, and thus the resulting chain is highly dependent. This situation could result in a decreased effective sample size as we need a chain with millions of generations to generate few independent observations.

With these limitations in mind, we propose an importance sampling method to generate independent samples from the posterior distribution in the phylogenetic setting. We introduce the program BISTRO (Bayesian Importance Sampling for TRees O...) for Bayesian phylogenetic inference through importance sampling. We compare the performance of MrBayes (reference) and BISTRO in a variety of simulated and real-life datasets, and conclude that this new sampling scheme is more efficient as proven by a bigger effective sample size (ESS). However, we found difficulties when applying BISTRO to big datasets (more than 25 taxa).

First, we recall the main concepts of importance sampling in section I, and then we describe the methods applied to phylogenetics in section II. In section III, we apply BISTRO to different simulated and real-life datasets, and compared its performance to MrBayes in terms of ESS. In section IV, we discuss the difficulties to apply BISTRO to big trees.

I. Importance Sampling Background

Let $X \sim f(x)$, and suppose that we want to calculate the expectation of a function $h(X)$ under $f(x)$:

$$E_f(h(X)) = \int h(x)f(x)dx := \mu$$

If the integral does not have a closed solution, we can estimate μ with the mean from a random sample $X_1, X_2, ..., X_n \sim f(x)$:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} h(X_i)$$

Problem: It can be hard (or impossible) to get random samples from $f(x)$.

Solution: Sample from an easier density $g(x)$ such that inference of $h(X)$ under $f(x)$ can be approached as inference of $h(X)w(X)$ under $g(x)$ for a weight function $w(x) = f(x)/g(x)$ defined when both $f(x)$ and $g(x)$ are *normalized* densities (we will discuss the *unnormalized* case in next subsection). That is,

$$\begin{aligned} E_f(h(X)) &= \int h(x)f(x)dx \\ &= \int h(x)\frac{f(x)}{g(x)}g(x)dx \\ &= \int h(x)w(x)g(x)dx \\ &= E_g(h(X)w(X)) \end{aligned}$$

Algorithm 1: Importance Sampling

Goal: Estimate $\mu = E_f(h(X))$, and $\sigma^2 = Var_f(h(X))$.

1) Sample independently $Y_1, Y_2, ..., Y_m \sim g(x)$
2) Define the weight function: $w(x) = f(x)/g(x)$
3) Mean estimate: $\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} h(y_i)w(y_i)$
4) Variance estimate: $\hat{\sigma}^2 = \frac{1}{m}\sum_{i=1}^{m}(h(y_i)w(y_i) - \hat{\mu})^2$

Standard error: Since $Y_1, Y_2, ..., Y_m \sim g(x)$ is an independent sample, we can compute the variance of the estimator as

$$Var_g(\hat{\mu}) = \frac{1}{m}Var_g(h(X)w(X)) = \frac{\sigma^2}{m}$$

Unnormalized case

The usual approach of importance sampling assumes that you have the normalized versions of both $f(x)$ and $g(x)$. In many real-life applications, this is not true.

Let $X \sim f(x) = c_1 f_0(x)$, and we again want to estimate $E_f(h(X))$. We sample independently $Y_1, Y_2, ..., Y_m \sim g(x) = c_2 g_0(x)$. Unlike in the previous setting, we compute the weights with the *unnormalized* densities: $w_0(y_i) = f_0(y_i)/g_0(y_i)$.

This unnormalized case is different from the normalized importance sampling case. Here we do inference of $h(X)$ directly, but with a weighted sample $Y_1, Y_2, ..., Y_m$ with weights $\tilde{w}_0(y_1), \tilde{w}_0(y_2), ..., \tilde{w}_0(y_m)$ with $\tilde{w}_0(y_j) = \frac{w_0(y_j)}{\sum_{i=1}^m w_0(y_i)}$. On the contrary, in the normalized importance sampling case, we do inference of $h(X)w(X)$ with a random sample from $g(x)$.

Algorithm 2: Unnormalized Importance Sampling

Goal: Estimate $\mu = E_f(h(X))$, and $\sigma^2 = Var_f(h(X))$.

1) Sample independently $Y_1, Y_2, ..., Y_m \sim g(x)$
2) Define the weight function: $w_0(y_j) = f_0(y_j)/g_0(y_j)$, and the normalized weight as $\tilde{w}_0(y_j) = \frac{w_0(y_j)}{\sum_{i=1}^m w_0(y_i)}$.
3) Mean estimate: $\hat{\mu} = \sum_{i=1}^m h(y_i)\tilde{w}_0(y_i)$
4) Variance estimate: $\hat{\sigma}^2 = \sum_{i=1}^m (h(y_i) - \hat{\mu})^2 \tilde{w}_0(y_i)$

The estimate $\hat{\mu}$ is sometimes called *self-normalized importance sampling estimate*.

Standard error: Since the sample $Y_1, Y_2, ..., Y_m$ with weights $\tilde{w}_0(y_1), \tilde{w}_0(y_2), ..., \tilde{w}_0(y_m)$ is no longer an independent sample (because weights add up to 1), the variance estimate is then (Owen, 2013)

$$\widehat{Var}(\hat{\mu}) = \sum_{i=1}^m \tilde{w}_0(y_i)^2 (h(y_i) - \hat{\mu})^2.$$

Diagnostics

Importance sampling diagnostics are not clear-cut rules. One common approach is to compute the effective sample size:

$$n_e = \frac{\left(\sum_{i=1}^n w(y_i)\right)^2}{\sum_{i=1}^n w(y_i)^2}.$$

If the effective sample size is too small, then one or few weights could be too large compared to the others, and importance sampling is not as efficient as expected.

II. Importance Sampling for Phylogenetics

Let $\theta = (T, t, \mathbf{Q}(\pi, r))$ be the parameters of interest in the phylogenetic setting, where T represents a tree topology, t represents the vector of branch lengths and $\mathbf{Q}(\pi, r)$ represents the rate matrix for the GTR model (reference) as a function of the base frequency vector $\pi = (\pi_A, \pi_C, \pi_G, \pi_T)$ and the transition rate parameters $s = (s_{AC}, s_{AG}, s_{AT}, s_{CG}, s_{CT}, s_{GT})$. Note that we have a different parametrization than MrBayes (add model parametrization here) Let X denote the DNA sequences as input data.

We want to generate independent samples from the posterior distribution $p(\theta|X)$ (the $f(x)$ density in section I). Thus, we need to find a density $g(\theta|X)$ such that we can generate samples from g instead of p.

The success of importance sampling relies on choosing a g that is close enough to the density of interest: centered in the same place (no bias) and with heavier tails. We will focus in finding a g that ressembles the likelihood $L(X|\theta)$, instead of the posterior $p(\theta|X)$, since we can always choose a flat prior.

The importance sampling density $g(\theta|X)$ has three parts since θ has three parts: topology, branch lengths and rate matrix. We explain each part of g in the next subsections:

$$g(\theta|X) = g_1(\mathbf{Q}) g_2(T|\mathbf{Q}) g_3(t|T, \mathbf{Q})$$

Density for $\mathbf{Q}$

First, we generate a rate matrix $\mathbf{Q}$ by generating separately a vector of base frequencies $\pi \sim Dirichlet(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ and the vector of rates $s \sim Dirichlet(\beta_1, \beta_2.\beta_3, \beta_4, \beta_5, \beta_6)$. To choose the parameters of the Dirichlets, we need to find unbiased estimates of π and s. This proved to be a difficult task. Estimating π by the observed base frequencies and s by the observed pairwise counts would yield biased estimates. Estimating Q for a fixed initial topology with branch lengths would also yield biased estimates. This is due to the fact that we need an estimate of Q *averaged* over different topologies and branch lengths. We provide a partial solution by using MCMC on the fixed Neighbor-Joining (NJ) tree (from Jukes-Cantor distances) to sample branch lengths and Q. We then use the sample of Q to provide estimates of center and variance for the Dirichlet densities.

Algorithm 3: Parameters of the proposal density for $\mathbf{Q}$

1) Obtain the NJ tree from the JC distances
2) Use MCMC on that fixed to obtain estimates of mean and variance for π: $(\hat{\pi}_1, \hat{\pi}_2, \hat{\pi}_3, \hat{\pi}_4)$ and s: $(\hat{s}_1, \hat{s}_2, \hat{s}_3, \hat{s}_4, \hat{s}_5, \hat{s}_6)$
3) Compute the parameters of the proposal Dirichlet densities: $\alpha_i = c_p \hat{\pi}_i$ for $i = 1, 2, 3, 4$ and $\beta_j = c_s \hat{s}_j$ for $j = 1, 2, 3, 4, 5, 6$, where c_p, c_s correspond to scaling factors that depend in the MCMC variance

For the importance sampling scheme, we sample $\pi \sim Dirichlet$ and $s \sim Dirichlet$, and then construct $\mathbf{Q}$ like this:

$$q_{ij} = \frac{s_{ij}}{2\pi_i}$$
$$q_{ii} = -\sum_i q_{ij}$$

Density for T *given* $\mathbf{Q}$

The proposal density for tree topologies depend on the clade distribution (Larget, 2013), but first we need an estimate of this clade distribution. With the MCMC mean estimate of Q, we use the GTR distance matrix between sequences to obtain a NJ tree. We then bootstrap the sites to get a bootstrap sample of NJ trees. Originally, we wanted to use this sample of NJ trees to estimate the clade distributions, but the bootstrap sample of trees has two problems: one, it is too spread out and therefore, it contains many clades that have too low posterior probability, and two, sometimes it lacks important clades with high posterior probability that never appear in the bootstrap sample. (do we solve this with the mean tree approach?)

To overcome these problems, we decide to weight trees according to its distance from a center tree. The procedure is like this: from the bootstrap sample, we compute a mean tree (reference? bret mean published somewhere?) and calculate the distance of each bootstrap tree to the mean tree. Each tree will then be assigned a weight of the form $\exp -\lambda d(T_{mean}, T)$, for a constant λ and for the distance $d(T_{mean}, T)$ defined as in (reference). In this way, trees that are close to the mean tree will have higher weight, and thus its clades will be sampled more often when using the clade distribution. This idea is based on the assumption that the mean tree is close to the MLE tree (which has not been proven, but see for a slightly different mean tree reference Megan Owen).

Algorithm 4: Estimation of clade distribution

1) Obtain a sample of NJ bootstrap trees with GTR distances for the MCMC mean of $\mathbf{Q}$
2) Compute the mean tree of the bootstrap sample as in (reference)
3) Compute the distance of each bootstrap tree to the mean tree as in (reference), and weight each tree by $\exp -\lambda d(T_{mean}, T)$
4) Use the weighted sample of bootstrap trees to estimate the clade distribution as in (Larget, 2013)

For the importance sampling scheme, we will sample one topology from the clade distribution (add details here on how?).

Density for t given $T, \mathbf{Q}$

After drawing a random topology T, we initialize all the branch lengths with the NJ distances (and 0.00001 as minimun length in case of negative distance), and then estimate the MLE distance for each branch at a time using the likelihood function

$$L(t) = \prod_k \sum_x \sum_y \pi_x P_{xy}(t) P(A_k|x) P(B_k|y)$$

where the product is over sites, the two sum are over the state of the internal nodes x and y, $P_{xy}(t)$ is the transition probability from x to y in time length t given by the GTR model, and $P(A_k|x), P(B_k|y)$ are the probability of the subtrees given the state of the internal nodes.

When estimating the MLE for a given branch length, we keep all the other branch lengths in the subtrees constant at their current values. Since the NJ branch lengths are very different from the MLE branch lenghts, we need to do several MLE passes to get accurate branch length estimates.

After all branch lengths are set at their MLE, we order the nodes in postorder, and traverse the tree from leaves to root. For a given cherry, we jointly sample the two branch lengths leading to leaves with a Gamma distribution centered at the joint MLE and with variance given by the observed 2×2 information matrix from the likelihood:

$$L(t_1, t_2) = \prod_k \sum_y \pi_y \sum_{x_1} \sum_{x_2} P_{yx_1}(t_1) P_{yx_2}(t_2) * P(A_k^{(1)}|x_1) P(A_k^{(2)}|x_2) P(B_k|y).$$

The observed information matrix is negative the inverse of the Hessian matrix evaluated at the MLE. (add figure)

This joint density allows us to account for the correlation of sister edges, which is an important component of the likelihood under certain situations (see section IV). Furthermore, the term $P(B_k|y)$ allows us to include all the data in the likelihood, not only the data in the subtrees of x_1 and x_2.

Algorithm 5: Joint Gamma sampling

Goal: Generate a joint Gamma random vector with mean $\mu = (\mu_1, \mu_2)$ and covariance matrix Σ

1) Obtain the Cholesky decomposition $\Sigma = LL^T$, with L a lower triangular matrix
2) Generate $T_1 \sim Gamma(\alpha_1, \lambda_1)$ with
$$\alpha_1 = \frac{\mu_1^2}{L_{11}^2} \qquad \lambda_1 = \frac{\mu_1}{L_{11}^2}$$
3) Compute $z_1 = \frac{T_1 - \mu_1}{L_{11}}$
4) Generate $T_2|T_1 \sim Gamma(\alpha_2, \lambda_2)$ with
$$\alpha_2 = \frac{(\mu_2 + L_{21} z_1)^2}{L_{22}^2} \qquad \lambda_2 = \frac{(\mu_2 + L_{21} z_1)}{L_{22}^2}$$

For the importance sampling scheme, we simply traverse the tree and sample the branch lengths of sister edges $(t_1, t_2) \sim JointGamma$.

Computation of the weights

We computed the likelihood of the data given the drawn $\theta = (\mathbf{Q}, T, \lambda)$ and evaluated the importance sampling density $g(\theta|X)$ to obtain the weight:

$$w(\theta) = \frac{L(\theta|X)}{g(\theta|X)}.$$

Finally, we repeated the process to obtain an independent sample $\theta_1, \theta_2, ..., \theta_n \sim g(\theta|X)$ with the unnormalized weights $w(\theta_1), w(\theta_2), ..., w(\theta_n)$. Note that since the likelihood used is not a normalized density, we are in the case of self-normalizing importance sampling estimates, and thus, we need to normalize the weights:

$$\tilde{w}(\theta_j) = \frac{w(\theta_j)}{\sum_{i=1}^n w(\theta_i)}.$$

todo: - describe h functions of interest: indicator of tree

III. Examples

here we present examples

IV. Discussion

References

Larget, B. 2013. The estimation of tree posterior probabilities using conditional clade probability distributions. Systematic Biology **62**:501–511.

Owen, A. B. 2013. *Monte Carlo theory, methods and examples.*