A Sequential Monte Carlo Algorithm for Coalescent Clustering

Dilan Görür DILAN@GATSBY.UCL.AC.UK
Yee Whye Teh YWTEH@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, UCL, London

Algorithms for automatically discovering hierarchical structure from data play an important role in machine learning. Teh et al. (2008) proposed a Bayesian hierarchical clustering model based on Kingman's coalescent (Kingman, 1982b) and proposed both greedy and sequential Monte Carlo (SMC) based agglomerative clustering algorithms for inference, the SMC algorithm having computational cost $O(n^3)$ per particle, where n is the number of data items. We build upon this work and propose a new SMC based algorithm for inference in the coalescent clustering model. Our algorithm is based upon a different perspective on Kingman's coalescent than that in (Teh et al., 2008), where the computations required to consider merging each pair of clusters at each iteration is not discarded in subsequent iterations. This improves the computational cost to $O(n^2)$ per particle. In experiments we show that our new algorithm achieves improved costs without sacrificing accuracy or reliability.

A regenerative race process perspective Kingman's coalescent (Kingman, 1982b) describes the family relationship between a set of haploid individuals by constructing the genealogy backwards in time. Ancestral lines coalesce when the individuals share an ancestor, and the genealogy is a binary tree rooted at the common ancestor of all the individuals under consideration. Let π be the genealogy of n individuals. There are $n-1$ coalescent events in π, ordered such that $i=1$ is the most recent one, and $i=n-1$ is the last when all ancestral lines are coalesced. Event i occurs at time $T_i < 0$ in the past, and involves the coalescing of two ancestors, denoted ρ_{l_i} and ρ_{r_i}, into one denoted ρ_i. Let A_i be the set of ancestors right after coalescent event i, and A_0 be the full set of individuals at the present time $T_0 = 0$.

We interpret each stage of the coalescent as a race between the $\binom{n-i+1}{2}$ pairs of individuals to coalesce. Each pair proposes a coalescent time, the pair with most recent coalescent time "wins" the race and gets to coalesce, at which point the next stage starts with $\binom{n-i}{2}$ pairs in the race. At stage i of the coalescent we have $n-i+1$ individuals in A_{i-1}, and $\binom{n-i+1}{2}$ pairs racing to coalesce. Each pair $\rho_l, \rho_r \in A_{i-1}$, $\rho_l \neq \rho_r$ proposes a coalescent time

$$t_{lr}|T_{i-1} \sim T_{i-1} - \text{Exp}(1), \tag{1}$$

that is, by subtracting from the last coalescent time a waiting time drawn from an exponential distribution of rate 1. The pair ρ_{l_i}, ρ_{r_i} with most recent coalescent time wins the race, and coalesces into a new individual ρ_i at time $T_i = t_{l_i r_i}$. At this point stage $i+1$ of the race begins, with some pairs dropping out of the race (specifically those with one half of the pair being either ρ_{l_i} or ρ_{r_i}) and new ones entering (specifically those formed by pairing the new individual ρ_i with an existing one). For the pairs (ρ_l, ρ_r) that did not drop out nor just enter the race, the distribution of t_{lr} conditioned on the fact that $t_{lr} < T_i$ (since (ρ_l, ρ_r) did not win the race at stage i) is $t_{lr}|T_i \sim T_i - \text{Exp}(1)$ due to the memoryless property of the exponential distribution, thus (1) still holds and *we need not redraw t_{lr} for the stage $i+1$ race.* In other words, once t_{lr} is drawn, it can be reused for subsequent stages of the race until it either wins a race or drops out.

We obtain the probability of the coalescent π as a product over the $i = 1, \ldots, n-1$ stages of the race, of the probability of each event "ρ_{l_i}, ρ_{r_i} wins stage i and coalesces at time T_i" given more recent stages. The probability at stage i is simply the probability that $t_{l_i r_i} = T_i$, and that all other proposed coalescent times $t_{lr} < T_i$, conditioned on the fact that the proposed coalescent times t_{lr} for all pairs at stage i are all less than T_{i-1}. Taking the product and with terms canceling,

$$p(\pi) = \prod_{i=1}^{n-1} \left(p(t_{l_i r_i} = T_i) \prod_{\substack{\rho_l \in \{\rho_{l_i}, \rho_{r_i}\}, \\ \rho_r \in A_{i-1} \setminus \{\rho_{l_i}, \rho_{r_i}\}}} p(t_{lr} < T_i) \right),$$

where the second product runs only over those pairs that dropped out after stage i. The first term is the probability of pair (ρ_{l_i}, ρ_{r_i}) coalescing at time T_i given its entrance time, and the second term is the probability of pair (ρ_l, ρ_r) dropping out of the race at time T_i given its entrance time.

The coalescent can be used as a prior over binary trees in a model where we have a tree-structured likelihood

function for observations at the leaves. Let θ_i be the subtree rooted at ρ_i and $\mathbf{x}_i$ be the observations at the leaves of θ_i. Teh et al. (2008) showed that by propagating messages up the tree the likelihood function can be written in a sequential form:

$$p(\mathbf{x} \mid \pi) = Z_0(\mathbf{x}) \prod_{i=1}^{n-1} Z_{\rho_i}(\mathbf{x}_i|\theta_i), \tag{2}$$

where $Z_0(\mathbf{x})$ is an easily computed normalization constant in eq.(2) and the "local likelihood" term Z_{ρ_i} is a function only of the coalescent times associated with ρ_{l_i}, ρ_{r_i}, ρ_i and of the local messages sent from ρ_{l_i}, ρ_{r_i} to ρ_i (see (Teh et al., 2008) for further details).

Our sequential Monte Carlo algorithm for posterior inference is directly inspired by the regenerative race process described above. The idea is that the proposal distribution Q_{lr} is constructed taking into account the observed data.We use eq.(1) as the "local prior" and combine it with the "local likelihood" to obtain the density for the proposal distribution Q_{lr}:

$$q_{lr}(t_{lr}) \propto Z_{\rho_{lr}}(\mathbf{x}_{lr}|t_{lr}, \rho_l, \rho_r, \theta_{i-1}) p(t_{lr} \mid T_{c(lr)}) \tag{3}$$

where ρ_{lr} is a hypothetical individual resulting from coalescing l and r, $T_{c(lr)}$ denotes the time when it enters the race, $\mathbf{x}_{lr}$ are the data under ρ_l and ρ_r, and $p(t_{lr} \mid T_{c(lr)}) = e^{t_{lr} - T_{c(lr)}} \mathbb{I}(t_{lr} < T_{c(lr)})$ is simply an exponential density with rate 1 that has been shifted and reflected[1] The weight updates are given by

$$w_i = w_{i-1} \frac{Z_{\rho_i}(\mathbf{x}_i|\theta_i) p(t_{l_i r_i} = T_i)}{q_{l_i r_i}(t_{l_i r_i} = T_i)} \times \prod_{\substack{\rho_l \in \{\rho_{l_i}, \rho_{r_i}\}, \\ \rho_r \in A_{i-1} \setminus \{\rho_{l_i}, \rho_{r_i}\}}} \frac{p(t_{lr} < T_i)}{q_{lr}(t_{lr} < T_i)}, \tag{4}$$

with $w_0 = Z_0(\mathbf{x})$.

The proposal distribution in (Teh et al., 2008) has a form similar to eq.(3), but with the exponential rate at stage i being $\binom{n-i+1}{2}$ instead. This dependence means that at each stage of the race the coalescent times proposal distribution changes for each pair, leading to an $O(n^3)$ computation time. On the other hand, similar to the prior process, we need to propose a coalescent time for each pair only once when it is first created. This results in $O(n^2)$ computational complexity per particle[2].

[1] $\mathbb{I}(\cdot)$ is an indicator function returning 1 if its argument is true, and 0 otherwise.

[2] Technically the time cost is $O(n^2(m + \log n))$, where n is the number of individuals, and m is the cost of sampling from and evaluating eq.(3). The additional $\log n$ factor comes about because a priority queue needs to be maintained to determine the winner of each stage efficiently, but this is negligible compared to m.

Note that it may not always be possible (or efficient) to compute the normalizing constant of the density in eq.(3) which we need for weight updates eq.(**??**). In that case, we can use an approximation $\tilde{Z}_{\rho_{lr}}$ to $Z_{\rho_{lr}}$ instead.

In the experiments, we considered clustering of discrete data using the independent-sites parent-independent mutation model over multinomial vectors in (Teh et al., 2008), and we used an approximation to $Z_{\rho_{lr}}$ to construct the proposal distribution. We compared the performance of our algorithm `SMC1` to `SMC-PostPost` of (Teh et al., 2008) on a synthetic data set cmposed of 15 binary 12-dimensional vectors. There is overlap between the features of the data points however the data does not obey a tree structure, which will result in a multimodal posterior. Since both SMC algorithms are exact in the limit, the values should converge as we add more particles. We can check convergence by observing the variance of likelihood estimates of multiple runs. The variance should shrink as we increase the number of particles. Figure 1 shows the change in the estimated likelihood as a function of number of particles. From this figure, we can see that the computationally cheaper algorithm `SMC1` is more efficient also in the number of particles as it gives more accurate answers with less particles. The

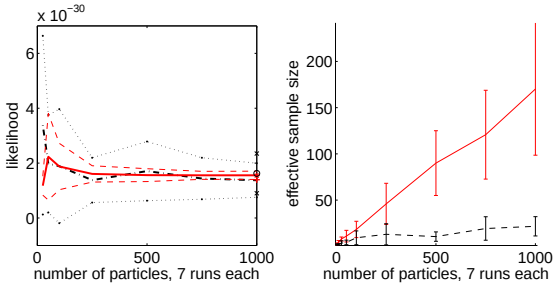

Figure 1. The change in the likelihood (left) and the effective sample size (right) as a function of number of particles for `SMC1` (solid) and `SMC-PostPost` (dashed). The mean estimate of both algorithms are very close, with the `SMC1` having a much tighter variance. The variance of both algorithms shrink and the effective sample size increases as the number of particles increase.

distribution of the particle weights is important for the efficiency of the importance sampler. Ideally, the weights would be uniform such that each particle contributes equally to the posterior estimation. If there is only a few particles that come from a high probability region, the weights of those particles would be much larger than the rest, resulting in a low effective sample size. For the synthetic dataset, the effective sample size of `SMC-PostPost` is very poor, and that of `SMC1`

is much higher, see Figure 1.

We also applied our algorithm to the same Indo-European subset of the phylolinguistic data in (Teh et al., 2008), that is 44 languages with 100 binary features. Unfortunately, on this dataset, the effective sample size of both algorithms is close to one. A usual method to circumvent the low effective sample size problem in sequential Monte Carlo algorithms is to do resampling, that is, detecting the particles that will not contribute much to the posterior from the partial samples and prune them away, multiplying the promising samples. There are two stages to doing resampling. We need to decide at what point to prune away samples, and how to select which samples to prune away. As shown by (Chen et al., 2005), different problems may require different resampling algorithms. We tried resampling using Algorithm 5.1 of (Fearnhead, 1998), however this did not improve the final effective sample size for either algorithms on this data set. Note that in the recursive calculation of the weights in `SMC1` , we are including the effect of a pair only when they either coalesce or cease to exist for the sake of saving computations. Therefore the partial weights are not really informative about the state of the sample and the effective sample size does not really give any explanation about whether the current sample is good or not. In fact, we did observe oscillations on the effective sample size calculated on the weights along the iterations, i.e. starting off with a high value, decreasing to virtually 1 and increasing later before the termination, which also indicates that it is not clear which of the particles will be more effective eventually. An open question is how to incorporate a resampling algorithm to improve the efficiency.

References

Chen, Y., Xie, J., & Liu, J. (2005). Stopping-time resampling for sequential monte carlo methods. *Journal of the Royal Statistical Society, 67*.

Fearnhead, P. (1998). *Sequential Monte Carlo method in filter theory*. Doctoral dissertation, Merton College, University of Oxford.

Gilks, W., & Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics, 41*, 337–348.

Kingman, J. F. C. (1982a). The coalescent. *Stochastic Processes and their Applications, 13*, 235–248.

Kingman, J. F. C. (1982b). On the genealogy of large populations. *Journal of Applied Probability, 19*, 27–43. Essays in Statistical Science.

Teh, Y. W., Daume III, H., & Roy, D. M. (2008). Bayesian agglomerative clustering with coalescents. *Advances in Neural Information Processing Systems*.