Interactive Hierarchical Clustering using Bayesian Nonparametrics

Sharad Vikram
Computer Science and Engineering
University of California, San Diego
svikram@cs.ucsd.edu

Sanjoy Dasgupta
Computer Science and Engineering
University of California, San Diego
dasgupta@cs.ucsd.edu

Abstract

A widely-used class of algorithms to understand data is hierarchical clustering, but it is often difficult to reconcile the results of these algorithms with hierarchies constructed by humans. Interaction, or querying humans for constraints on the data, is a popular solution for addressing this discrepancy. In this paper, we propose using *leave-one-out* interactions to achieve better hierarchies by integrating the interactions into a Bayesian nonparametric posterior distribution over hierarchies.

1 Introduction

Hierarchical clustering is a popular task in machine learning. In general, a dataset is modeled by a tree (typically binary) where the leaves of the tree represent individual data points and internal nodes represent groupings of the data. Different levels of the hierarchy correspond to different flat clusterings of the data, growing coarser as we move up the tree towards the root.

There are a wide variety of algorithms for hierarchical clustering. Single-linkage and complete-linkage clustering are both examples of agglomerative clustering where a hierarchy is built by merging leaf nodes from the bottom-up. Spectral clustering is an example of the opposite paradigm of divisive clustering, where the hierarchy is defined in terms of splits over the entire dataset. Furthermore, there exists an entire class of Bayesian nonparametric methods to infer posterior distributions over hierarchies which best explain the data [1, 2, 3, 4, 5]. An overarching problem in all of these methods, however, is that although these models can flexibly model data, we still often want hierarchies to obey certain constraints for different problems. For example, we may want a phylogenetic tree to match an known evolutionary trend, but if the tree produced by an algorithm does not, it is difficult to resolve the difference.

A popular solution to obtain such constraints is *interaction*, where we are able to ask specific queries about the data to humans. A popular form of interaction used in flat clustering is obtaining pairwise constraints over the data (*cannot-link* and *must-link*) which are then used to constrain the output clustering [6, 7, 8, 9, 10]. Although there has been work extending pairwise constraints to hierarchical clustering [11], this mode of interaction is not conducive to hierarchical clustering. This is because each level of a particular hierarchy is a different flat clustering, so data might link at one level but may not link a level down. However, there has been work integrating other modes of interaction with hierarchies [12, 13].

We consider the *leave-one-out* (LOO) interaction where a human is presented with a triplet of data (a, b, c), and decides which datum does not belong. This form of interaction is natural for hierarchical clustering since it is relatively easy for humans to answer, and qualitatively measures how far data are from each other on a tree. Furthermore, LOO interactions can be efficiently integrated into a class of MCMC samplers for hierarchical Bayesian nonparametric models. In this paper, we specify a sampler over the posterior Dirichlet Diffusion Tree (DDT) [1] that respects a set of LOO interactions, while also showing promising results on a real dataset.

2 Interactive Hierarchical Clustering

In this section, we briefly discuss the specifics of the LOO interaction and another approach to build hierarchies that satisfy a set of LOO constraints. Let T be a binary tree with N leaves, each of which is labeled $1, 2, \ldots, N$. Let mrca(a, b) represent the most recent common ancestor of nodes a and b, and for nodes n and m such that $n \neq m$, $n < m$ if m is an ancestor of n. For a LOO interaction (a, b, c) such that a, b, and c are leaves in T and c is determined the odd one out, T satisfies (a, b, c) if and only if mrca$(a, b) <$ mrca(a, c).

Aho et. al first tackled the problem of constructing a hierarchy given a set of LOO constraints, $C = \{(a_i, b_i, c_i)\}_{i=1}^{K}$ [14]. They present an algorithm which we will call CONSTRUCT-TREE, that minimally guarantees that the tree returned satisfies C, assuming at least one such tree exists. The algorithm first constructs a constraint graph $G = (V, E)$. The set of vertices is just the set of leaves in the tree, $V = \{1, \ldots, N\}$ and the set of the edges $E = \{(a, b)|(a, b, c) \in C\}$. The split at the top level of a tree satisfies C if and only if the split corresponds to a partition of the connected components of G. Thus, the algorithm builds the tree by partitioning the connected components of G, and then recursing on each partition after removing edges corresponding to satisfied LOO constraints. For the purposes of this paper, CONSTRUCT-TREE returns binary trees by only using bipartitions of the constraint graph. Furthermore, we also assume a single "master hierarchy", from which all LOO constraints are chosen from, to avoid the problem of conflicting constraints.

CONSTRUCT-TREE is not sufficient for our purposes, as it only considers the constraints, not features, associated with the data. Although we would like a tree that satisfies a set of LOO constraints, there may exist several candidates, so we also desire the "best" tree among the candidates. An alternate view is that without interactions, our algorithm should return the "best" tree from the space of possible binary trees with N leaves, $\mathcal{T}_N$. After receiving a set of LOO interactions C, our algorithm should return the "best" tree from the constrained space $\mathcal{T}_N^C$, where all trees that violate C are removed, or in a Bayesian setting, sample from a constrained posterior distribution of hierarchies given data.

3 MCMC with Interaction

In this section, we present an algorithm to sample from a constrained posterior distribution over binary trees. To accomplish this, we look to Bayesian nonparametrics. Although there are several Bayesian nonparametric prior distributions over hierarchies, in this work we consider the Dirichlet Diffusion Tree (DDT) [1]. However, many of the algorithms and ideas we propose can generalize to other such models, such as the Tree-Structured Stick Breaking process (TSSB) and the Time-Marginalized Coalescent (TMC) [4, 5].

The DDT is an exchangeable prior distribution over hierarchically structured data. It assumes a binary tree structure along with a continuous-time process responsible for generating data. Let $X = \{x_i\}_{i=1}^{N}$ be a d-dimensional dataset. According to the DDT, the first point, x_1 is generated by a generalized Gaussian diffusion process (Brownian motion) beginning at the origin. The point walks for a fixed amount of time, typically from $t = 0$ to $t = 1$ where if $X_1(t)$ represents the value of x_1 at time t, then $X_1(t + dt) = X_1(t) + \mathcal{N}(0, \sigma^2 I_d dt)$. Neal also includes a Polya urn-inspired path-reinforcement element in the generative process. The i-th point will initially follow the path created by the first $i - 1$ points and will choose to diverge at some time t_i. If the i-th point reaches a divergence location, it selects a branch proportional to the number of previous points that followed paths along each branch. Let m be the number of points that have previously traversed the path that the i-th point is currently following. The probability of diverging within the interval $[t, t + dt]$ of the current time t is $a(t)dt/m$, where $a(t)$ is called the *acquisition function*. For the purpose of this work, we assume the acquisition function is $a(t) = 1/(1 - t)$. When a point diverges, we create a internal node in our binary tree, and associated with the internal node is the divergence time and a d-dimensional latent vector equal to the the value of the i-th point at the time of divergence.

Associated with a DDT is a probability density over divergence time and location, which when sampled yields a particular branch of the DDT and a time. The exchangeability of the DDT allows us to obtain samples with a simple but efficient procedure. We start with a particle at the root of the tree and have it follow the generative process described earlier, returning the location and time when the particle diverges from the tree. This algorithm lends itself to a procedure for sampling the posterior

DDT given data, based on the subtree-prune and regraft (SPR) move used in phylogenetics [1, 15]. An SPR move first involves selecting a subtree uniformly at random and pruning it from the tree. A branch on the tree is selected according to some distribution and the subtree is grafted back onto the tree. Our choice of regraft distribution is simply the probability density on divergence locations and times in a DDT. In addition to sampling the tree structure, Neal suggests either interleaving Gibbs sampling moves for the latent vectors of each internal node or integrating the latent vectors out of the model entirely [1]. The state of the sampler is thus the current tree structure, the times associated with each node on the tree, and possibly the latent divergence location vectors at each internal node.

We use a *constrained-SPR move* as the basis of incorporating interactions into posterior DDT. Let C be a set of LOO constraints. To sample from the posterior distribution over DDTs that satisfy C, we restrict the regraft locations of an ordinary SPR move. Neal's algorithm for sampling a regraft location for the DDT involves selecting a path from the root to a particular divergence location. We use an SPR move that assigns zero probability to any divergence locations that would violate constraints in C. We accomplish this by generating the path from the root in the same manner as the DDT diffusion process, but avoid branches that would resulted in violated constraints. For details on the constrained-SPR move, see Appendix A and for a visualization see Figure 2. We also interleave Gibbs sampling moves for the latent vectors at each internal node. This constrained-SPR move results in a Markov chain that indeed converges to a stationary distribution. We prove this by showing the chain's aperiodicity and irreducibility assuming the first tree in the chain satisfies C.

Theorem 3.1. *A constrained-SPR Markov chain is aperiodic.*

Proof. See Appendix B.1. □

Theorem 3.2. *A constrained-SPR Markov chain is irreducible.*

Proof. See Appendix B.2. □

We now consider two interaction scenarios: offline and online. In the offline interaction scenario, we have already obtained a set of LOO interactions C and wish to sample from the constrained DDT posterior given data. We adopt an inductive approach by instantiating our sampler with an initial tree produced by CONSTRUCT-TREE [14], where divergence times and latent vectors are chosen deterministically. Every future tree from the sampler will thus satisfy C. The online interactive scenario occurs when we are currently sampling from a posterior distribution that satisfies C, but are presented with a new constraint c and now wish to sample from posterior that satisfies $C + \{c\}$. For an interaction (a, b, c) we find the subtree whose root is $\text{mrca}(a, \text{mrca}(b, c))$ and replace it with a subtree produced by CONSTRUCT-TREE using the constraint set $C + \{c\}$. We then continue sampling as normal using $C + \{c\}$.

4 Experiments

We first obtained the *zoo* dataset from the UCI repository [16]. It consists of 101 animals and 15 binary features associated with each animal. We defined the "master hierarchy" by finding the induced subtree for the associated animals on the Open Tree of Life (OTOL) [17]. This required us to use the subset of 97 animals which were defined in the OTOL database. Using the master hierarchy, we obtained the set of all possible LOO constraints (n = 129,766). This set was randomly subsetted into a training set (n = 200) and a test set (n = 10,000).

We first performed single-linkage clustering on the dataset, the result of which is pictured in Figure 5. A notable error in this result is that seals, dolphins, and porpoises are in the fish branch. In addition, we also ran an SPR-move sampler for 300,000 iterations whose MAP hierarchy makes the same mistake (see Figure 6). We finally ran a constrained-SPR move sampler using the training set in offline-mode for 300,000 iterations, with the resulting MAP tree pictured in Figure 7. This tree did not make the same mistake.

We further evaluated the offline interaction scenario (Figure 1a and b) by running several samplers for 100,000 iterations, each instantiated with a different number of constraints from the training set. We benchmarked each sampler with a *constraint score*, or the percentage of LOO constraints from the test set that were satisfied. Using just 10 constraints shows little to no benefit, but using

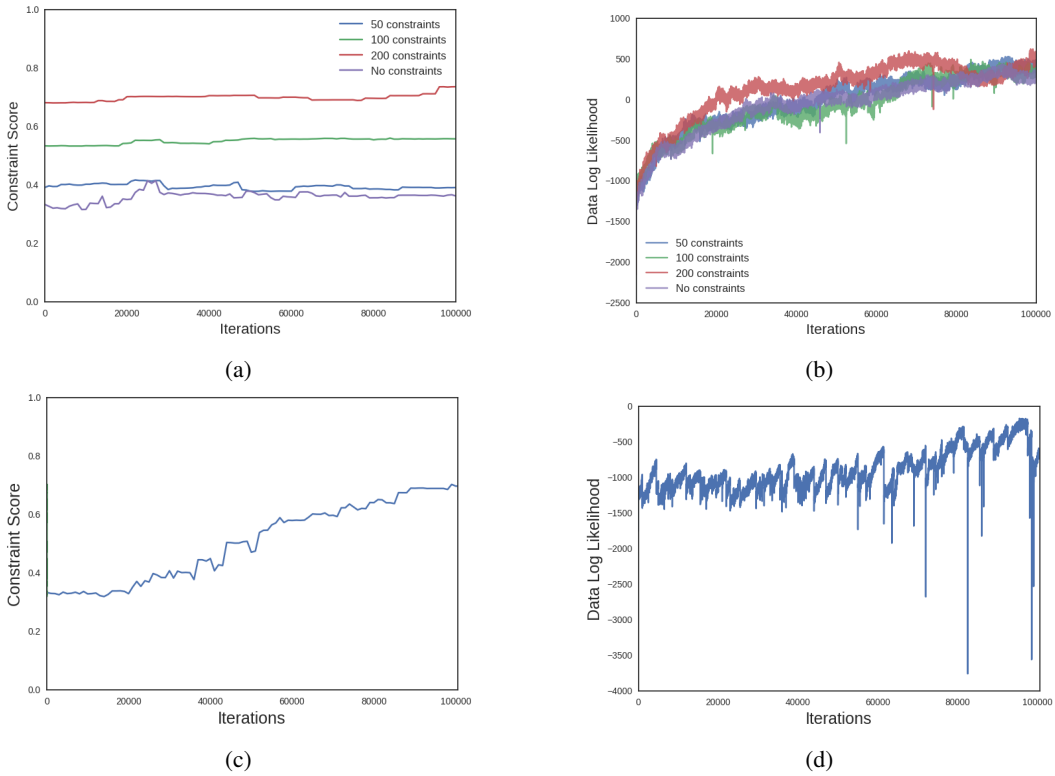


Figure 1: (a) and (b) are the constraint scores and likelihoods for a set of four offline samplers, with 0, 50, 100, and 200 constraints respectively. (c) and (d) are the constraint scores and likelihoods for an online sampler, where a constraint was added every 500 iterations.

100 constraints increases the constraint score significantly. Interestingly, data likelihood has little correlation with the constraint score, indicating that human-defined correctness does not correlate with maximum-likelihood explanations of the data. Multiple runs of the samplers and different training subsets resulted in similar constraint scores. We finally tested the online interaction scenario (Figure 1c and d) by using a single sampler, but adding one constraint every 500 iterations for a total of 100,500 iterations. Downward spikes in the likelihood graph for the online sampler correspond to subtrees being reconfigured by CONSTRUCT-TREE. However, constraint score still improved on average.

5 Future Work

We would like to pursue an active model of interaction, where constraints are queried intelligently to speed up the convergence of the sampler. For example, queries that over highly variable regions of a tree are likely to be more informative than queries over unchanging regions. However, random queries will help avoid being stuck in local minima, so we aim to try a mixture of the two approaches. Another aspect of improvement would be handling conflicting constraints gracefully, as we currently assume constraints are chosen from a single master hierarchy.

We would also like to generalize to other Bayesian nonparametric models. Constrained-SPR moves easily extend to to the posterior TMC [5], but aren't directly applicable to the TSSB model. However, the TSSB Gibbs sampler chooses a path down the tree to an internal node in a similar manner to the DDT, so it is straightforward to limit these paths using LOO constraints [4]. We also look to propose a general probabilistic model over hierarchies with LOO constraints.

Acknowledgements

Sharad Vikram and Sanjoy Dasgupta are supported by NSF grant CNS-1446912.

References

[1] Radford M. Neal. Density modeling and clustering using Dirichlet diffusion trees. *Bayesian Statistics*, 2003.

[2] Yee Whye Teh, Hal Daumé, and Daniel Roy. Bayesian Agglomerative Clustering with Coalescents. jul 2009.

[3] David M. Blei, Thomas L. Griffiths, and Michael I. Jordan. The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):1–30, jan 2010.

[4] Ryan Prescott Adams, Zoubin Ghahramani, and Michael I. Jordan. Tree-Structured Stick Breaking Processes for Hierarchical Data. *Advances in Neural . . .* , page 16, 2010.

[5] Levi Boyles and Max Welling. The time-marginalized coalescent prior for hierarchical clustering. *Advances in Neural Information Processing . . .* , 2012.

[6] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. *ICML*, 2001.

[7] Zhengdong Lu and Todd K. Leen. Semi-supervised learning with penalized probabilistic clustering. *Advances in neural information . . .* , 2004.

[8] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Active Semi-Supervision for Pairwise Constrained Clustering. *SDM*, 2004.

[9] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. *Advances in neural . . .* , 2004.

[10] Zhengdong Lu and Todd K. Leen. Penalized probabilistic clustering. *Neural Computation*, 2007.

[11] Ian Davidson and S.S. Ravi. Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. *Data mining and knowledge discovery*, 2009.

[12] Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. *arXiv preprint arXiv:1312.6724*, 2013.

[13] Yuyin Sun, Adish Singla, Dieter Fox, and Andreas Krause. Building Hierarchies of Concepts via Crowdsourcing. *arXiv preprint arXiv:1504.07302*, 2015.

[14] A. V. Aho, Y. Sagiv, T. G. Szymanski, and J. D. Ullman. Inferring a Tree from Lowest Common Ancestors with an Application to the Optimization of Relational Expressions. *SIAM Journal on Computing*, 10(3):405–421, aug 1981.

[15] Steven N. Evans and Anita Winter. Subtree prune and regraft: A reversible real tree-valued Markov process. *The Annals of Probability*, 34(3):918–961, may 2006.

[16] M. Lichman. UCI machine learning repository, 2013.

[17] Cody E. Hinchliff, Stephen A. Smith, James F. Allman, J. Gordon Burleigh, Ruchi Chaudhary, Lyndon M. Coghill, Keith A. Crandall, Jiabin Deng, Bryan T. Drew, Romina Gazis, Karl Gude, David S. Hibbett, Laura A. Katz, H. Dail Laughinghouse, Emily Jane McTavish, Peter E. Midford, Christopher L. Owen, Richard H. Ree, Jonathan A. Rees, Douglas E. Soltis, Tiffani Williams, and Karen A. Cranston. Synthesis of phylogeny and taxonomy into a comprehensive tree of life. *Proceedings of the National Academy of Sciences*, page 201423041, sep 2015.

Appendix A The constrained-SPR sampler

Description of SPR Sampler for the DDT

To help elucidate the details of the constrained-SPR sampler, we first explain the unconstrained SPR sampler for the Dirichlet Diffusion Tree.

An SPR move consists of a *prune* and a *regraft*. The pruning procedure is as follows. Suppose T is a binary tree with n leaves. Let s be a non-root node in T selected uniformly at random and S be its corresponding subtree. To prune S from T, we remove s's parent p from T, and replace p with s's sibling.

The regraft procedure, for a general binary tree, involves selecting a branch in the tree to attach S to. For the DDT, both a branch and a time are selected according to the DDT density over divergence locations. We can sample from this distribution by instantiating a particle at the root, and sampling the particle's path down the tree.

The sampling procedure is a recursive algorithm. A particle at a particular node will do the following: 1) pick a particular branch to enter, then 2) either diverge on the branch or recursively sample the subtree along the branch. If we choose to diverge, we sample a divergence time according to the acquisition function $a(t)$. Note that each of these choices has a probability associated with it, according to the properties of the tree (details can be found in [1]). Also, the particle will always diverge at a branch connected to a leaf as the acquisition function tends to infinity at $t = 1$.

This procedure returns the location at which the particle diverges. Let (u, v) be the chosen branch. S is then attached at that location by creating a node p with children s and v and parent u. An example SPR move is visualized in Figure 2a.

Description of constrained-SPR sampler

The pruning procedure for the constrained-SPR sampler is identical to the unconstrained pruning procedure. Sampling a valid regraft location, however, involves eliminating all possible locations that would result in violated constraints. For the DDT, this can be accomplished by modifying the generative process used to sample divergence locations.

Recall that in the sampling procedure for divergence locations, a particle at a node picks a branch, and either diverges or recursively samples the node's child along that branch. We define four properties that will determine which decision our particle must make, all conditioned on a current set of LOO constraints. Let s be the root of the subtree we are currently grafting back onto tree T, let C be our current constraint set, and let leaves(u) denote the data points that are in leaves of the subtree rooted at u.

1. A node u is *path-banned* if for any constraint $(a, b, c) \in C$, $(a \in \text{leaves}(s) \wedge b \notin \text{leaves}(u) \wedge c \in \text{leaves}(u)) \vee (b \in \text{leaves}(s) \wedge a \notin \text{leaves}(u) \wedge c \in \text{leaves}(u))$. Intuitively, if u is *path-banned*, we cannot enter the branch connected to u, as we would automatically violate a constraint. This is because there is some constraint (a, b, c) or (b, a, c) in C such that c is in u and a and b are split up over our subtree and a node that's not u.

2. A node u is *sample-banned* if for any constraint $(a, b, c) \in C$, $(c \in \text{leaves}(s)) \wedge ((a \in \text{leaves}(v) \wedge b \in \text{leaves}(w)) \vee (a \in \text{leaves}(w) \wedge b \in \text{leaves}(v))$ where v and w are the children of u. Intuitively, this means that there exist some a and b in our constraint set that are split over u's children. Attaching a subtree containing c below u would automatically violate that constraint.

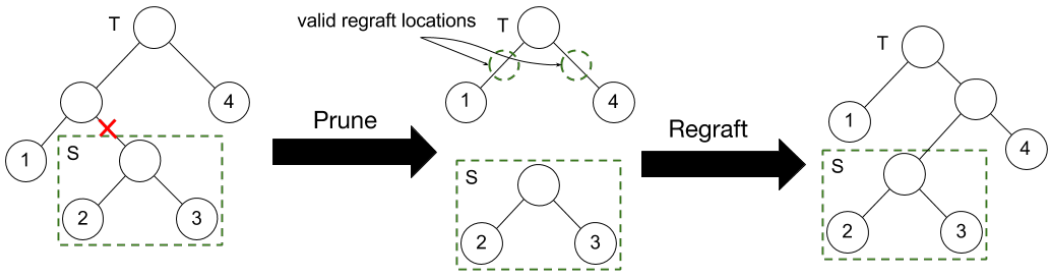
3. A node u is *path-required* if for any constraint $(a, b, c) \in C$, $(a \in \text{leaves}(s) \wedge b \in \text{leaves}(u)) \vee (b \in \text{leaves}(s) \wedge a \in \text{leaves}(u))$. If u is *path-required*, the particle must enter its branch, as for some constraint (a, b, c), if a is in our pruned subtree, then b is in u and c is in u's sibling. Choosing the branch to u's sibling would violate this constraint.

4. A node u is *sample-required* if for any constraint $(a, b, c) \in C$, $(a \in \text{leaves}(s) \wedge b \in \text{leaves}(u) \wedge c \in \text{leaves}(u)) \vee (b \in \text{leaves}(s) \wedge a \in \text{leaves}(u) \wedge c \in \text{leaves}(u))$. where v and w are the children of u. If u being *sample-required* means that there exists some a and c or

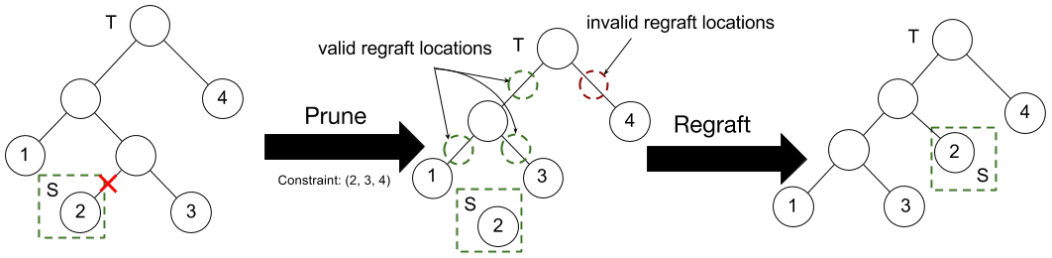
b and c in our constraint set that are split over the children of u. This means that we must sample u instead of diverging.

Using these four properties, we can define our final sampling procedure. Suppose we have a particle at node u. If a child of u is *path-banned*, we immediately pick the other branch. If a child of u is *path-required*, we immediately select its branch. Otherwise, we sample a branch according to the unconstrained procedure. Suppose branch (u, v) is selected, where v is a child of u. If v is *sample-required*, sample v with probability one. If v is *sample-banned*, we diverge with probability one. Otherwise, we choose to diverge or recursively sample according to the unconstrained procedure. Finally, if we choose to sample v, we remove constraints from our current set C that are now satisfied, and continue recursively.

This defines a procedure by which we can sample divergence locations that do not violate constraints.



(a)



(b)

Figure 2: (a) and (b) are visualizations of SPR and constrained-SPR moves respectively. In both examples, a subtree of T, S is selected uniformly at random and is then pruned from the tree. Next, a regraft location is selected from the valid regraft locations, and S is re-attached at that location. In (b), we are limiting regraft locations by the constraint (2, 3, 4), so we cannot re-attach S to the branch with 4 attached to it.

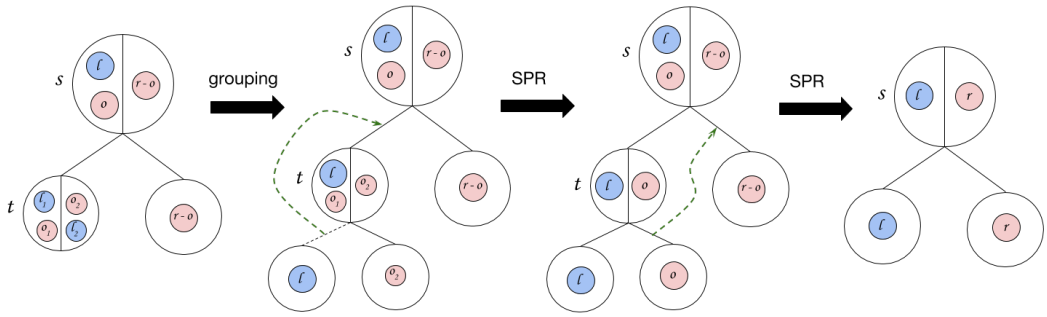


Figure 3: The process of converting s into canonical form. We first group nodes from l into their own pure subtree, then perform two constrained-SPR moves to put s into canonical form.

Appendix B Proof of convergence for constrained-SPR Markov chain

B.1 Proof of Theorem 3.1

Theorem 3.1. *A constrained-SPR Markov chain is aperiodic.*

Proof. A sufficient condition for the aperiodicity of a Markov chain is the existent of a "self-loop" in the transition matrix, or a non-zero probability of a state transitioning to itself. We assume that T, the current state of the Markov chain, satisfies constraint set C. In the posterior DDT, every edge has a non-zero probability of being a new divergence location, so when sampling a regraft location, it is possible to select the original detach location. In a constrained-SPR move, our regraft locations are limited to those that satisfy constraints. Since we know that T satisfies C, the original detach location has a non-zero probability of being selected. We thus have an aperiodic Markov chain. □

B.2 Proof of Theorem 3.2

Theorem 3.2. *A constrained-SPR Markov chain is irreducible.*

Proof. To prove irreducibility, we show that there is a non-zero probability of moving from state T to T', both of which satisfy C. We accomplish this by first defining a *canonical tree* T_C given a constraint set C and showing that we can reach T_C from T using constrained-SPR moves. We then show that a constrained-SPR move is reversible, proving that from T_C we can reach T', since there exists a path from T' to T_C.

A binary tree T can be entirely defined by the bipartitions made over the data at each node. For a given node, the constraint graph at that node is defined as follows. Let D_n be the data present at a node n and $G_n = (V_n, E_n)$ be the constraint graph for a given node n for a constraint set C. The vertices are $V_n = D_n$ and the edges are $E_n = \{(a, b)|(a, b, c) \in C, a \in D_n, b \in D_n, c \in D_n\}$. For a constrained binary tree, the bipartition at each node n must be a bipartition of the connected components of G_n. We define a particular node to be in *canonical form* if either a) it is a leaf, or b) the bipartition over G_n at that node can be written as (l, r), where l exactly matches a single, particular connected component of G_n, and r is the rest of the connected components. The particular component l is the connected component in G_n with the minimum data index inside it. Note that we treat the children of nodes as unordered. A canonical tree T_C is one such that every node in the tree is in canonical form.

To convert an arbitrary tree T that satisfies C into T_C, we first convert the root node of T into canonical form using constrained-SPR moves.

Let s be the root of T and let l be the set of points that ought to be in their own partition according to G_s. In order for s not to be in canonical form, l must be in a partition with data from other connected components in G_s, which we will call o. The bipartition if s were in canonical form would be (l, r) and the current non-canonical bipartition can thus be written as $(l + o, r - o)$.

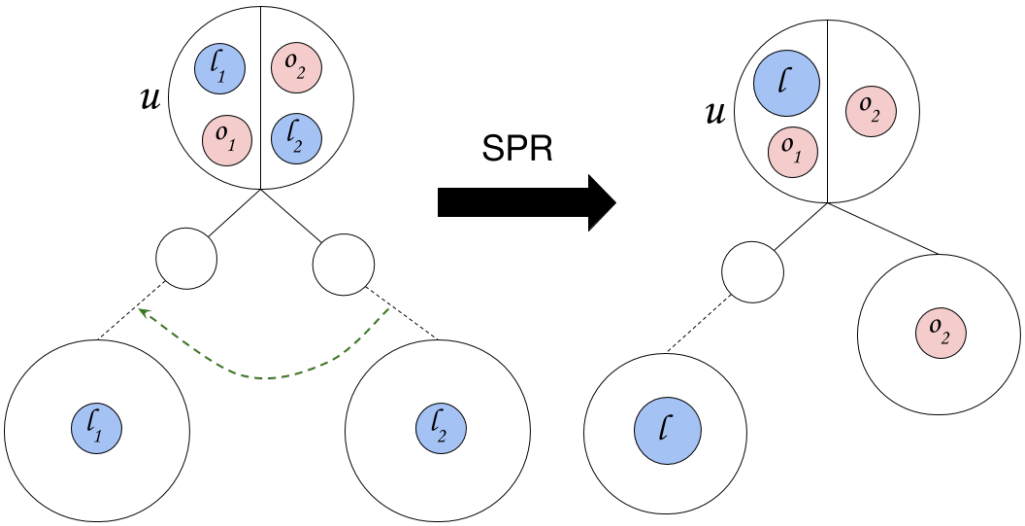


Figure 4: The process of grouping the data in u that belong to l into their own pure subtree. u is the lowest node of t (see Figure 3) that has data from l in both of its children.

We first examine t, the child of the root that contains $l + o$. In general, the data from l and the data from o could be split over the children of t, so the current partition can be written as (l_1+o_1, l_2+o_2) where $l = l_1 + l_2$ and $o = o_1 + o_2$. This can be seen in the first tree of Figure 3. We first group the data from l into their own "pure" subtree of t. Let u be the root of the lowest non-pure subtree of t that has data from l in both of its children. There exist two subtrees that are descendants of u that contain data from l (one on the left and one on the right). Those two subtrees must be pure, and furthermore, they are both free to move within u via constrained-SPR moves because they are in different connected components in G_u. Thus, we can perform a constrained-SPR move to merge these two pure subtrees together into a larger pure subtree. We can repeat this process for t until all nodes from l are in their own pure subtree of t. The partition of t can thus be written as $(l + o_1, o_2)$, since the pure subtree may be several levels down from t. This grouping process is visualized in Figure 4 and the results can be seen in the second tree in Figure 3.

We now perform a constrained-SPR move to detach pure subtree of l and regraft it to the edge between s and t. This is a permissible move since l is its own connected component in G_s. We now have the third tree in Figure 3. We now perform a final constrained-SPR move, moving the subtree of o to the opposite side of s, creating the proper canonical partition of (l, r). To entirely convert T into T_C, we need to recurse and convert every node in T into canonical form.

Every constrained-SPR move is reversible because undoing a particular regraft in the next constrained-SPR move just involves selecting the particular subtree that was regrafted, then re-attaching it to its previous location. We know that this regraft has non-zero probability because the previous tree did not violate constraints. Thus, since any arbitrary T can be converted into T_C and by reversibility, T_C can be converted into an arbitrary tree T', we have a non-zero probability path to convert T into T'. □

Appendix C Additional Experiment Figures

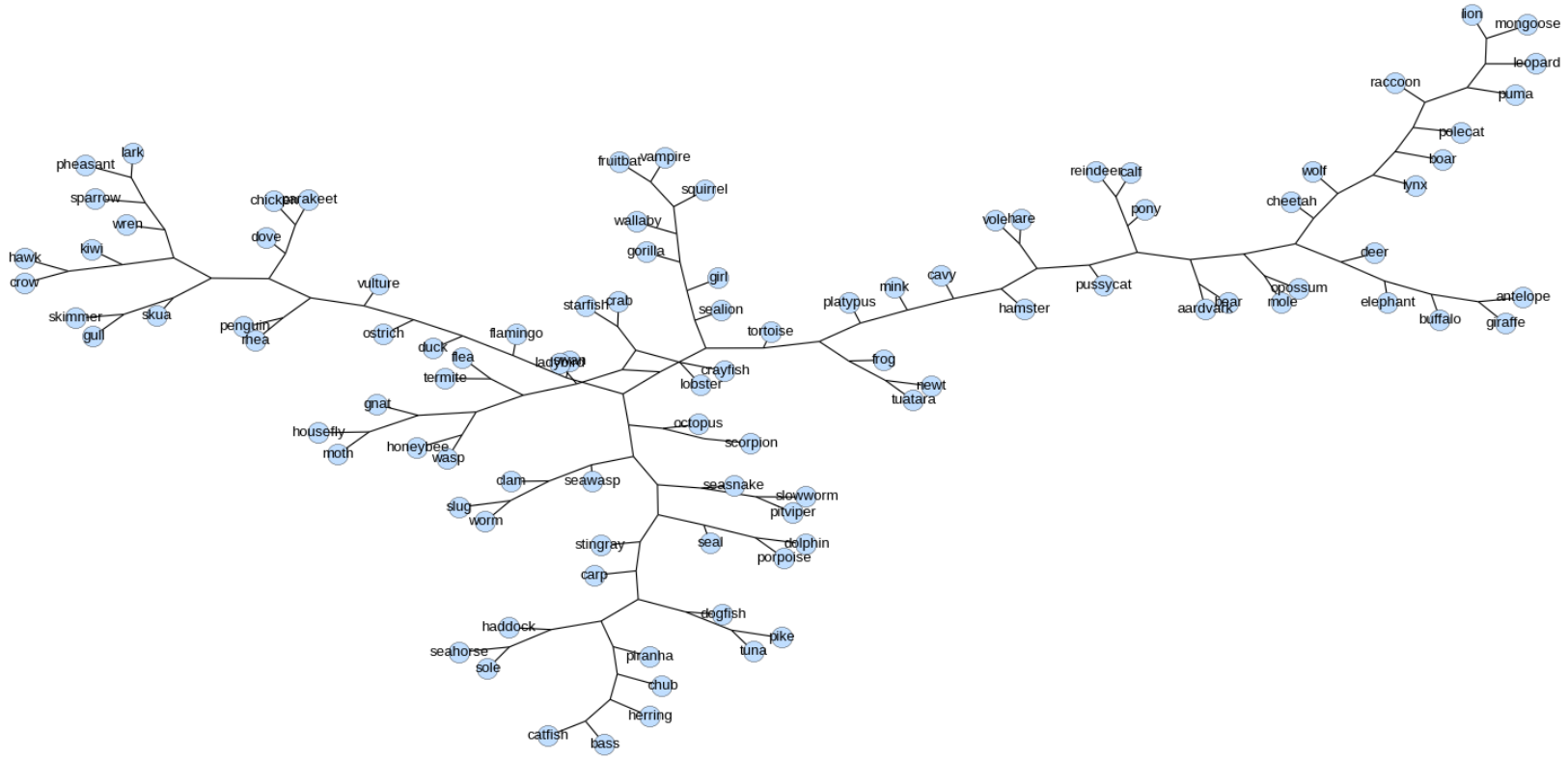


Figure 5: A visualization of the hierarchy produced by single-linkage clustering. A notable error in this tree is that the seal, dolphin, and porpoise leaves are in the subtree for fish.

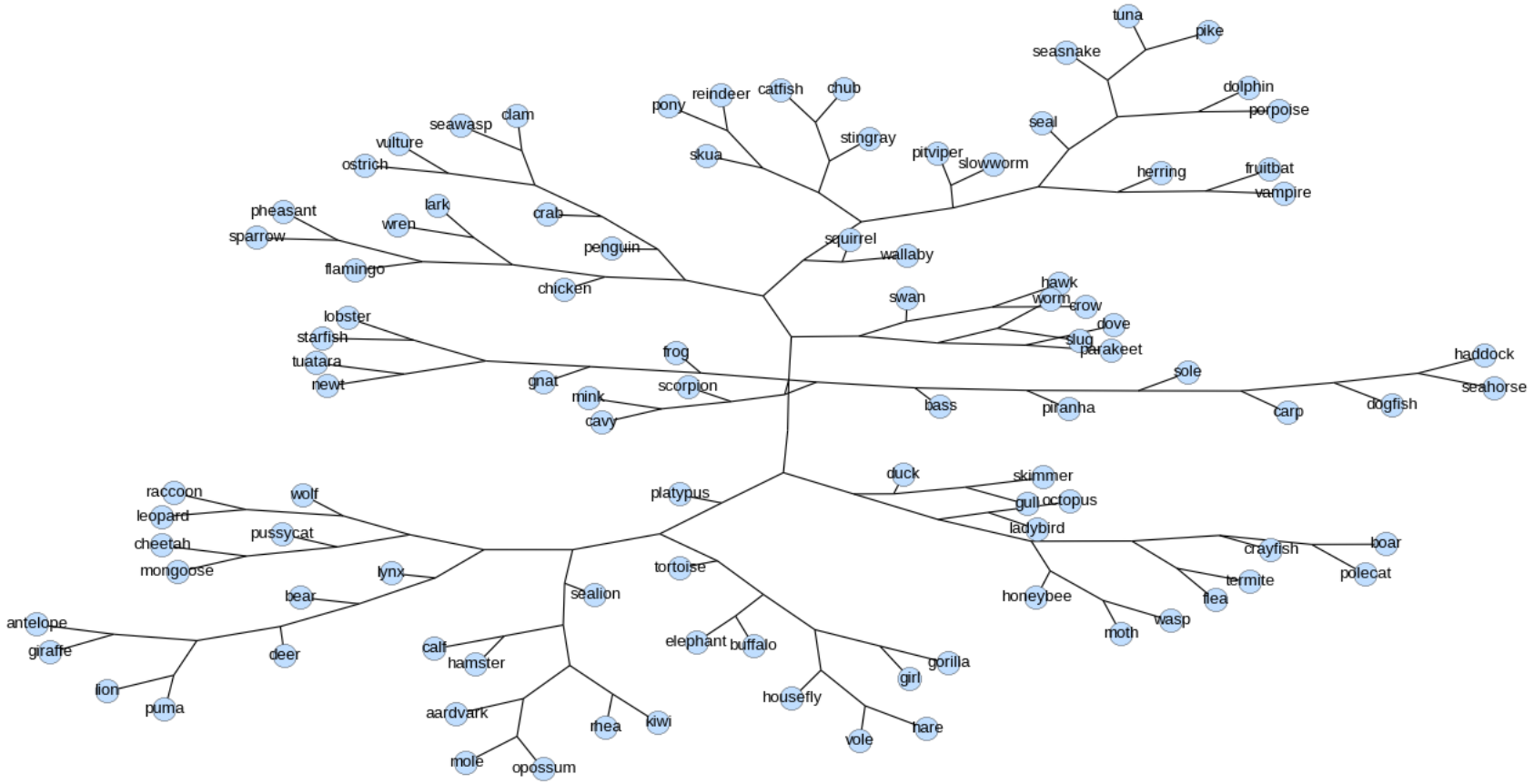


Figure 6: A visualization of the unconstrained *maximum a posteriori* DDT. Along with other errors, this tree contains the same mistake made by the single linkage tree in Figure 5.

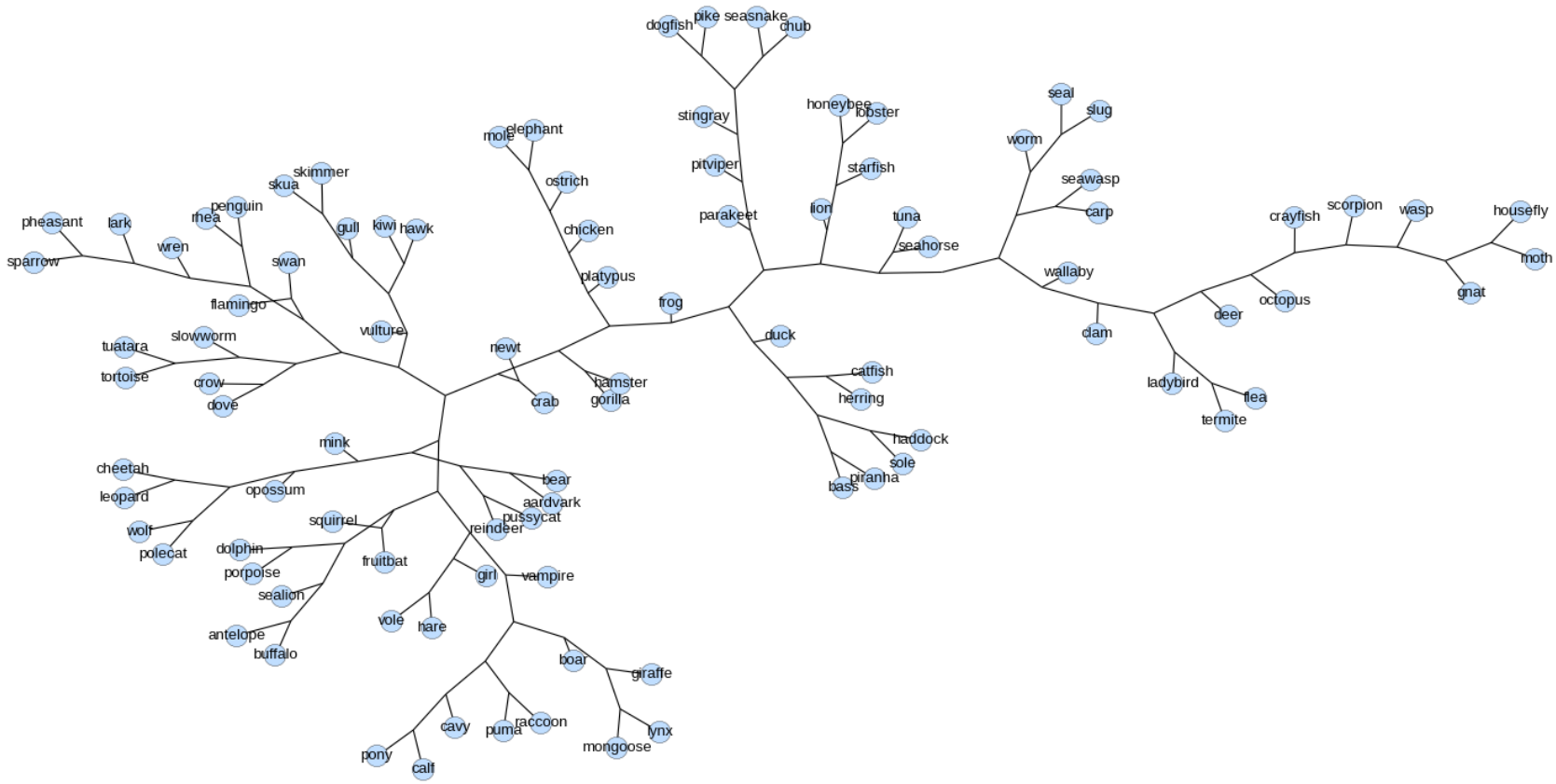


Figure 7: A visualization of the constrained *maximum a posteriori* DDT (200 constraints). This tree does not make the same mistake as the one in Figure 6 and Figure 5.