# A review of methods for Bayesian hierarchical clustering

Sharad Vikram

June 3, 2016

UCSD

# Background


Background
- Clustering
- Bayesian learning

Bayesian hierarchical clustering
- Coalescent models
- Diffusion models
- Inference

Adding interaction

Conclusion

# Unsupervised learning

In unsupervised learning, we are interested in finding underlying structure in data.

# Unsupervised learning

In unsupervised learning, we are interested in finding underlying structure in data.

Examples of unsupervised learning problems are:

| Dimensionality reduction |
| --- |
| Finding low-dimensional representations of high-dimensional data |

# Unsupervised learning

In unsupervised learning, we are interested in finding underlying structure in data.

Examples of unsupervised learning problems are:

**Dimensionality reduction**

Finding low-dimensional representations of high-dimensional data

**Density estimation**

Estimating an underlying probability distribution given samples from the distribution

# Unsupervised learning

In unsupervised learning, we are interested in finding underlying structure in data.

Examples of unsupervised learning problems are:

**Dimensionality reduction**

Finding low-dimensional representations of high-dimensional data

**Density estimation**

Estimating an underlying probability distribution given samples from the distribution

**Clustering**

Discovering natural groups in data

# Clustering

The most popular methods for clustering produce flat clusterings.

A flat clustering is a partition of a set of data, or an assignment of each data point into one of several disjoint sets.

# Clustering

The most popular methods for clustering produce flat clusterings.

A flat clustering is a partition of a set of data, or an assignment of each data point into one of several disjoint sets.

$$\{1, 2, \ldots, N\} \rightarrow \{\{\ldots\}, \{\ldots\}, \ldots, \{\ldots\}\} \tag{1}$$

# Flat clustering

The most popular algorithm for flat clustering is k-means.

# Flat clustering

The most popular algorithm for flat clustering is k-means.

1. Pick a number of clusters $K$.

# Flat clustering

The most popular algorithm for flat clustering is k-means.

1. Pick a number of clusters $K$.
2. Randomly initialize cluster centers $\{\mu_1, \ldots, \mu_K\}$.

# Flat clustering

The most popular algorithm for flat clustering is k-means.

1. Pick a number of clusters $K$.
2. Randomly initialize cluster centers $\{\mu_1, \ldots, \mu_K\}$.
3. Alternate until convergence:
   - Assign each data point to its closest cluster center.
   - Assign each cluster center to the mean of its assigned data points.

# Flat clustering

The most popular algorithm for flat clustering is k-means.

1. Pick a number of clusters $K$.
2. Randomly initialize cluster centers $\{\mu_1, \ldots, \mu_K\}$.
3. Alternate until convergence:
    - Assign each data point to its closest cluster center.
    - Assign each cluster center to the mean of its assigned data points.

This is equivalent to optimizing the loss function:

$$L(\mu_1, \ldots, \mu_K) = \sum_{i=1}^{N} \min_k \|x_i - \mu_k\|_2^2 \tag{2}$$

# Flat clustering

The most popular algorithm for flat clustering is k-means.

1. Pick a number of clusters $K$.
2. Randomly initialize cluster centers $\{\mu_1, \ldots, \mu_K\}$.
3. Alternate until convergence:
    - Assign each data point to its closest cluster center.
    - Assign each cluster center to the mean of its assigned data points.

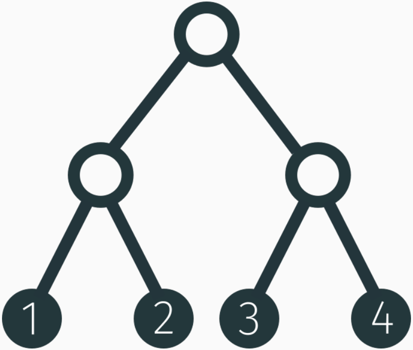
This is equivalent to optimizing the loss function:

$$L(\mu_1, \ldots, \mu_K) = \sum_{i=1}^{N} \min_k \|x_i - \mu_k\|_2^2 \tag{2}$$

How do we pick $K$?

# Hierarchical clustering

In hierarchical clustering, data is recursively partitioned to form a tree (typically binary), also called a hierarchy.



# Traditional hierarchical clustering algorithms

Traditional clustering algorithms can be broadly broken down into two categories:

# Traditional hierarchical clustering algorithms

Traditional clustering algorithms can be broadly broken down into two categories:

**Agglomerative (bottom-up)**

The tree is built by beginning with just singleton clusters (leaves) and iteratively merging them until we have one cluster (root)

# Traditional hierarchical clustering algorithms

Traditional clustering algorithms can be broadly broken down into two categories:

**Agglomerative (bottom-up)**

The tree is built by beginning with just singleton clusters (leaves) and iteratively merging them until we have one cluster (root)

**Divisive (top-down)**

The tree is built by beginning with a single cluster (root) and recursively partitioning it until we have just singleton clusters (leaves)

# Agglomerative clustering

We iteratively merge the two closest clusters until we have one cluster left.


$$\{\{1\},\{2\},\{3\},\{4\}\}$$

# Agglomerative clustering

We iteratively merge the two closest clusters until we have one cluster left.


$$\{\{1,2\},\{3\},\{4\}\}$$

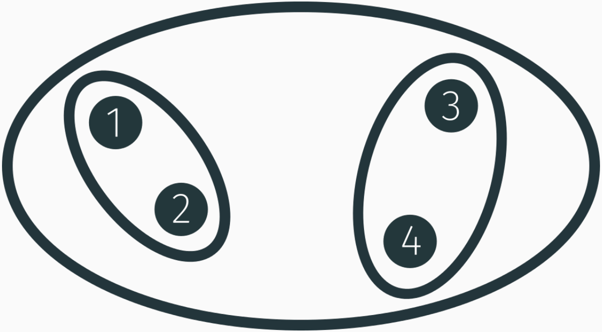
# Agglomerative clustering

We iteratively merge the two closest clusters until we have one cluster left.


$$\{\{1, 2\}, \{3, 4\}\}$$

# Agglomerative clustering

We iteratively merge the two closest clusters until we have one cluster left.




$$\{\{1, 2, 3, 4\}\}$$

# Agglomerative clustering examples

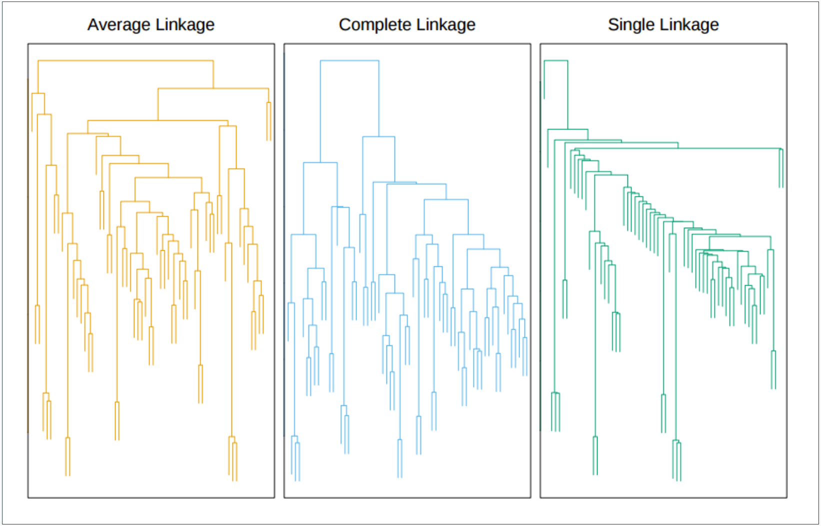



Figure 1: Trees produced by agglomerative clustering algorithms with different linkage criteria. Source: [1]

# Divisive clustering

In divisive clustering, we begin with the root cluster and recursively partition it until we are left with singleton leaf clusters.

## Approaches

- Use $k$-means recursively until only singleton clusters
- Define a similarity graph $G$ in terms of similarity function $s(x, x')$ and perform partitions by finding a minimum cut on $G$.

# Ambiguous data

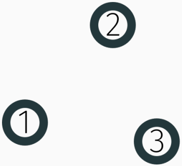
Consider the two following scenarios where circles represent tight clusters of data.

Consider the two following scenarios where circles represent tight clusters of data.


A single binary tree is not sufficient to describe either of these configurations.

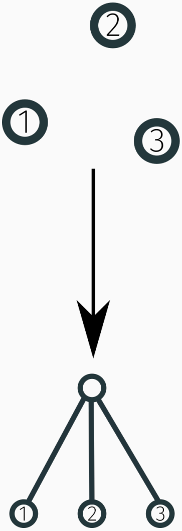
# Multifurcating trees

Perhaps extending our binary trees to $k$-ary trees will help.



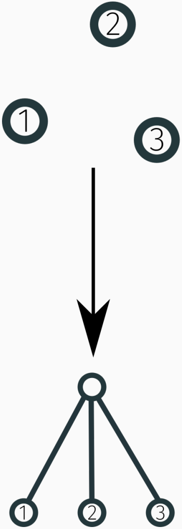

# Multifurcating trees

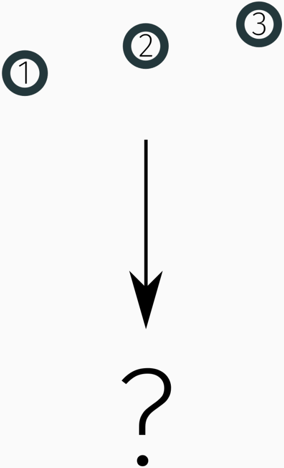
Perhaps extending our binary trees to $k$-ary trees will help.




# Multifurcating trees

Perhaps extending our binary trees to $k$-ary trees will help.



# Uncertainty in hierarchies

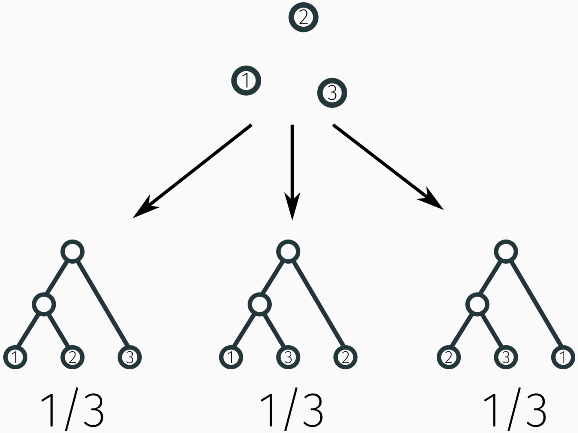
Furthermore, small perturbations in our data can have an effect on the output hierarchy.

Furthermore, small perturbations in our data can have an effect on the output hierarchy.

# Probabilistic reasoning

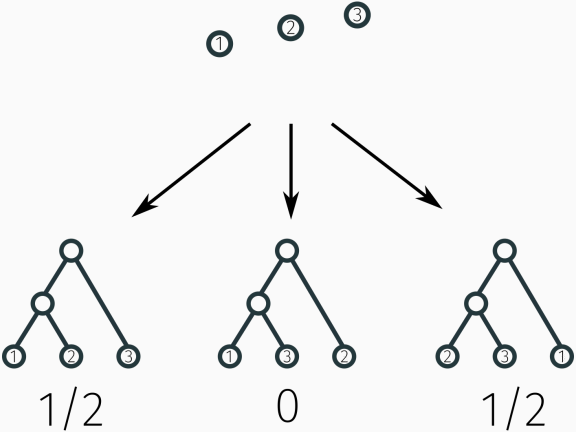
One approach is to model ambiguity and uncertainty with probability. We output a probability distribution over all possible trees.

One approach is to model ambiguity and uncertainty with **probability**. We output a probability distribution over all possible trees.



# Probabilistic reasoning

One approach is to model ambiguity and uncertainty with **probability**. We output a probability distribution over all possible trees.



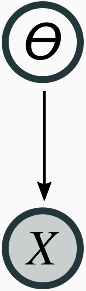
# Latent variable models

We take a quick detour to review Bayesian learning.

# Latent variable models

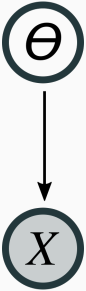
We take a quick detour to review Bayesian learning.

We take a quick detour to review Bayesian learning.



# Latent variable models

We take a quick detour to review Bayesian learning.




Our data $X$ is generated conditionally given a set of unobserved latent variables $\theta$.

# Distributions of interest

We are interested in the posterior distribution of latent variables given data, $P(\theta|X)$, calculated via Bayes rule.

# Distributions of interest

We are interested in the posterior distribution of latent variables given data, $P(\theta|X)$, calculated via Bayes rule.

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)} = \frac{P(X|\theta)P(\theta)}{\int_{\Theta} P(X|\theta)P(\theta)\,d\theta} \tag{3}$$

# Distributions of interest

We are interested in the posterior distribution of latent variables given data, $P(\theta|X)$, calculated via Bayes rule.

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)} = \frac{P(X|\theta)P(\theta)}{\int_\Theta P(X|\theta)P(\theta)\,d\theta} \tag{3}$$

$P(\theta)$ is called the prior distribution and $P(X|\theta)$ is called the likelihood model. They are specified beforehand.

# Bayesian inference

Typically the hardest part of computing the the posterior distribution is calculating $P(X) = \int_{\Theta} P(X|\theta)P(\theta)\,d\theta$, the marginal distribution.

# Bayesian inference

Typically the hardest part of computing the the posterior distribution is calculating $P(X) = \int_\Theta P(X|\theta)P(\theta)\,d\theta$, the marginal distribution.

When the prior $P(\theta)$ and likelihood $P(X|\theta)$ are conjugate, the posterior is analytically computable.

# Bayesian inference

Typically the hardest part of computing the the posterior distribution is calculating $P(X) = \int_\Theta P(X|\theta)P(\theta)\,d\theta$, the marginal distribution.

When the prior $P(\theta)$ and likelihood $P(X|\theta)$ are conjugate, the posterior is analytically computable.

Otherwise, we usually approximate it with methods such as:

- Markov chain Monte Carlo (MCMC)
- Variational inference
- MAP estimation via EM

# Bayesian hierarchical clustering

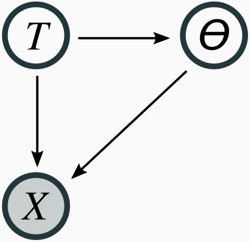

- Background
    - Clustering
    - Bayesian learning
- Bayesian hierarchical clustering
    - Coalescent models
    - Diffusion models
    - Inference
- Adding interaction
- Conclusion

# BHC as a latent variable model

Bayesian hierarchical clustering is a statistical approach, specifically a latent variable model.

# BHC as a latent variable model

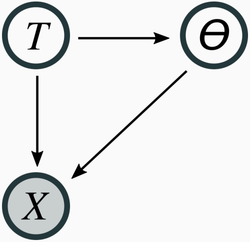
Bayesian hierarchical clustering is a statistical approach, specifically a latent variable model.




It is composed of data $X$ and latent variables $T$ and $\theta$.

# BHC as a latent variable model

Bayesian hierarchical clustering is a statistical approach, specifically a latent variable model.

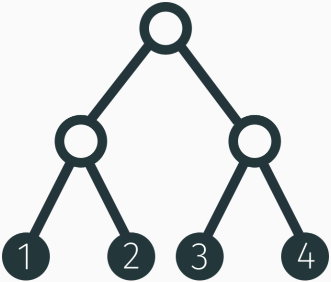



It is composed of data $X$ and latent variables $T$ and $\theta$.

- $T$ is a tree structure, sampled from a tree prior distribution $P(T)$.
- $\theta$ is a set of parameters, generated in a tree likelihood model $P(\theta | T)$.

# Tree priors

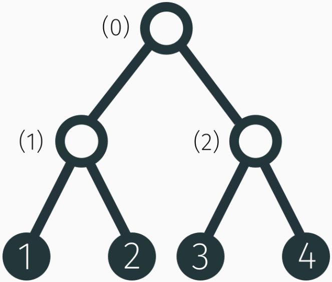
Most often we are interested in rooted binary trees with labeled leaves, also called cladograms.




Very often, cladograms will contain additional information.

# Tree priors

Most often we are interested in rooted binary trees with labeled leaves, also called cladograms.

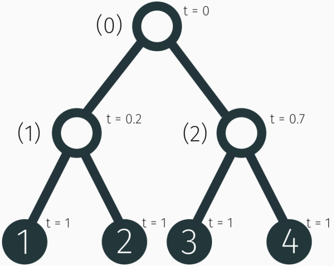



Very often, cladograms will contain additional information.

- An ordering on the internal nodes

# Tree priors

Most often we are interested in rooted binary trees with labeled leaves, also called cladograms.



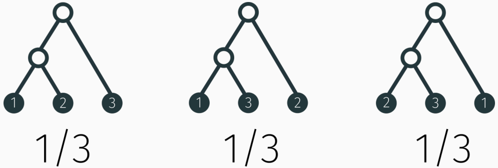

Very often, cladograms will contain additional information.

- An ordering on the internal nodes
- Times associated with each node

# Tree priors

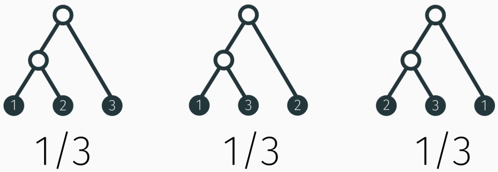
The simplest tree prior is the uniform distribution over cladograms.

The simplest tree prior is the uniform distribution over cladograms.



# Tree priors

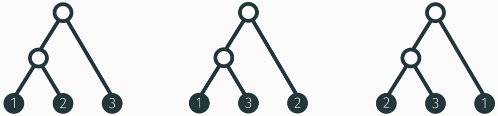
The simplest tree prior is the uniform distribution over cladograms.




Tree priors mostly fall into two categories:

# Tree priors

The simplest tree prior is the uniform distribution over cladograms.




1/3 1/3 1/3

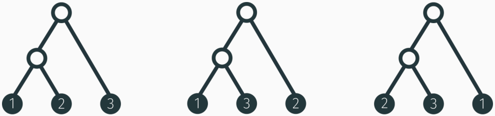
Tree priors mostly fall into two categories:

**Coalescent models**

Trees are modeled in an agglomerative fashion, merging clusters until one is left

# Tree priors

The simplest tree prior is the uniform distribution over cladograms.




1/3 1/3 1/3

Tree priors mostly fall into two categories:

**Coalescent models**

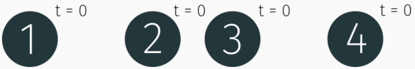
Trees are modeled in an agglomerative fashion, merging clusters until one is left

**Diffusion models**

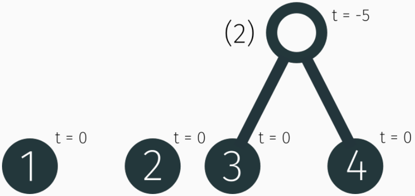
Trees are modeled inductively, starting with a tree of size $1$ and growing it to a tree of size $N$.

# Coalescent models

Coalescent models begin with a set of $N$ data, or "individuals", at $t = 0$.



Coalescent models begin with a set of $N$ data, or "individuals", at $t = 0$.

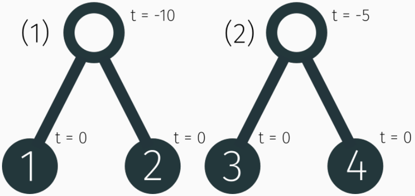



Every iteration, we:

- Pick two distinct nodes $a$ and $b$ uniformly at random.
- Sample a coalesce time $t$.
- Create an internal node whose children are $a$ and $b$ with time $t$.

# Coalescent models

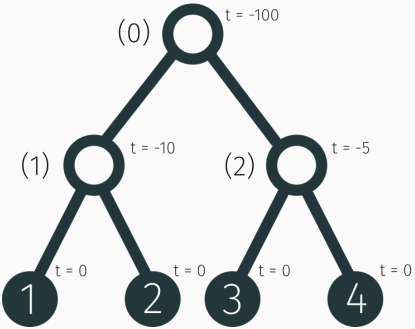
Coalescent models begin with a set of $N$ data, or "individuals", at $t = 0$.




Every iteration, we:

- Pick two distinct nodes $a$ and $b$ uniformly at random.
- Sample a coalesce time $t$.
- Create an internal node whose children are $a$ and $b$ with time $t$.

Coalescent models begin with a set of $N$ data, or "individuals", at $t = 0$.




Every iteration, we:

- Pick two distinct nodes $a$ and $b$ uniformly at random.
- Sample a coalesce time $t$.
- Create an internal node whose children are $a$ and $b$ with time $t$.

# Kingman's coalescent

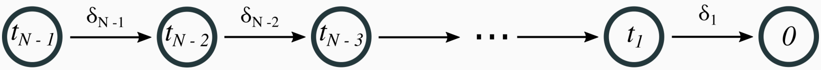
The canonical coalescent model is Kingman's coalescent [2].

There are a total of $N-1$ events happening at time $t_{N-1} < t_{N-2} < \cdots < t_1$, which happen at a constant coalesce rate.

# Kingman's coalescent

The canonical coalescent model is Kingman's coalescent [2].

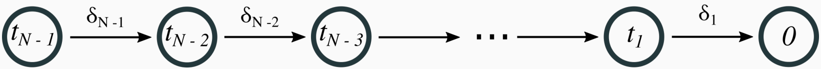
There are a total of $N-1$ events happening at time $t_{N-1} < t_{N-2} < \cdots < t_1$, which happen at a constant coalesce rate.



# Kingman's coalescent

The canonical coalescent model is Kingman's coalescent [2].

There are a total of $N-1$ events happening at time $t_{N-1} < t_{N-2} < \cdots < t_1$, which happen at a constant coalesce rate.




Let $\delta_i$ represent the elapsed time between coalescent event $i-1$ and $i$. We let

$$\delta_i \sim \mathrm{Exp}\left(\binom{N-i+1}{2}\right) \tag{4}$$

# Kingman's coalescent

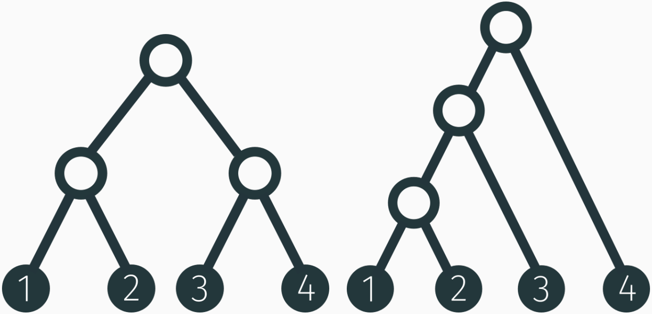
- Kingman's coalescent corresponds to the uniform distribution over *ordered cladograms*.

# Kingman's coalescent

- Kingman's coalescent corresponds to the uniform distribution over *ordered cladograms*.
- Its induced distribution over unordered cladograms is the time-marginalized coalescent (TMC) [3].

# Kingman's coalescent

- Kingman's coalescent corresponds to the uniform distribution over *ordered cladograms*.
- Its induced distribution over unordered cladograms is the time-marginalized coalescent (TMC) [3].
- What sort of trees does Kingman's coalescent favor?



# Diffusion models

Basic idea: begin with a tree over one data, then for $N$ iterations:

- Sample a branch and time at random.
- Attach a leaf to the branch, creating a new internal node with the sampled time.

# Diffusion models

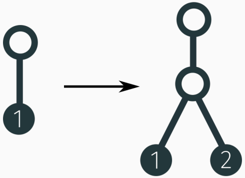
Basic idea: begin with a tree over one data, then for $N$ iterations:

- Sample a branch and time at random.
- Attach a leaf to the branch, creating a new internal node with the sampled time.

# Diffusion models

Basic idea: begin with a tree over one data, then for $N$ iterations:

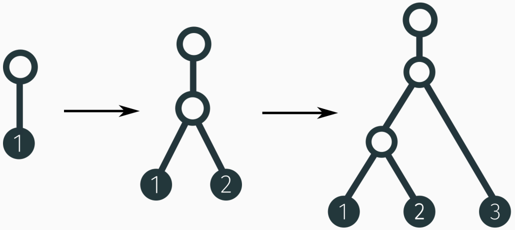
- Sample a branch and time at random.
- Attach a leaf to the branch, creating a new internal node with the sampled time.



# Diffusion models

Basic idea: begin with a tree over one data, then for $N$ iterations:

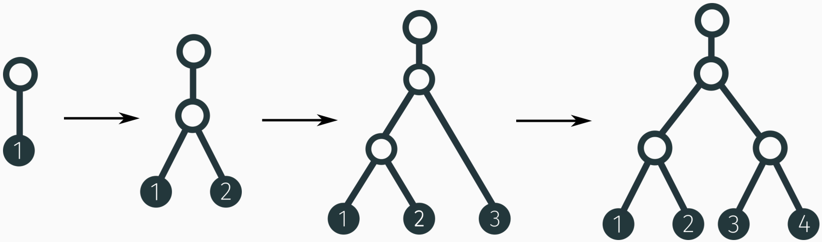
- Sample a branch and time at random.
- Attach a leaf to the branch, creating a new internal node with the sampled time.



# Diffusion models

Basic idea: begin with a tree over one data, then for $N$ iterations:

- Sample a branch and time at random.
- Attach a leaf to the branch, creating a new internal node with the sampled time.



# Dirichlet diffusion tree

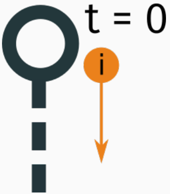
The simplest model is the Dirichlet diffusion tree (DDT), which produces trees with both ordering and times associated with internal nodes [4].

The model is defined inductively via a continuous time process. Assume we have a tree of size $i-1$.

# Dirichlet diffusion tree

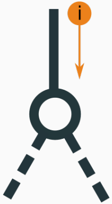
The simplest model is the Dirichlet diffusion tree (DDT), which produces trees with both ordering and times associated with internal nodes [4].

The model is defined inductively via a continuous time process. Assume we have a tree of size $i - 1$.



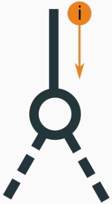

On each iteration, a particle labeled $i$ begins at the root and travels downwards.

**Case 1:** The particle reaches an internal node.



# Dirichlet diffusion tree

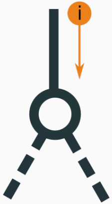
**Case 1:** The particle reaches an internal node.



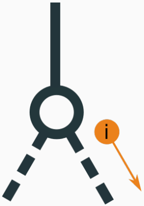

It picks one of the branches proportional to the past number of particles that have picked either branch.

# Dirichlet diffusion tree

**Case 1:** The particle reaches an internal node.

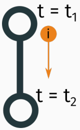



It picks one of the branches proportional to the past number of particles that have picked either branch.

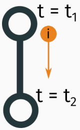


# Dirichlet diffusion tree

**Case 2:** The particle *diverges* on its current branch, creating an internal node and a leaf according to an acquisition function, $a(t)$.

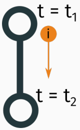


**Case 2:** The particle *diverges* on its current branch, creating an internal node and a leaf according to an acquisition function, $a(t)$.

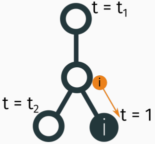



Let $m$ be the number of past particles that have traversed the current branch. The probability of diverging at time $dt$ is $a(t)dt/m$.

**Case 2:** The particle *diverges* on its current branch, creating an internal node and a leaf according to an acquisition function, $a(t)$.

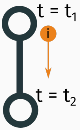



Let $m$ be the number of past particles that have traversed the current branch. The probability of diverging at time $dt$ is $a(t)dt/m$.

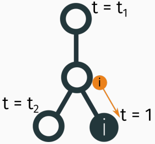


# Dirichlet diffusion tree

**Case 2:** The particle *diverges* on its current branch, creating an internal node and a leaf according to an acquisition function, $a(t)$.




Let $m$ be the number of past particles that have traversed the current branch. The probability of diverging at time $dt$ is $a(t)dt/m$.

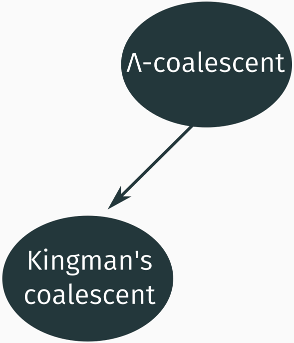



If $a(1) = \infty$, each particle is guaranteed to diverge before $t = 1$.

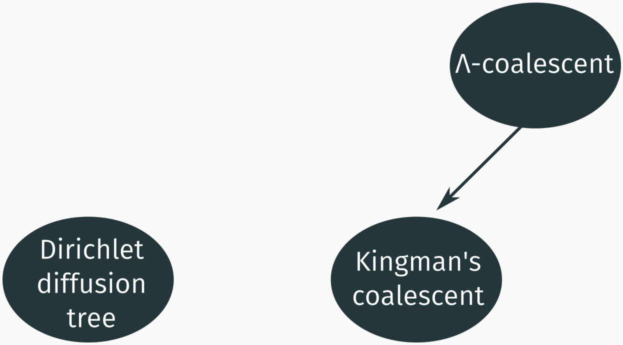
# Generalizations

# Generalizations

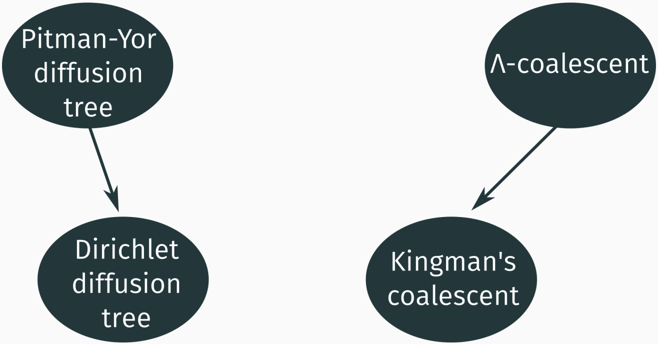
# Generalizations



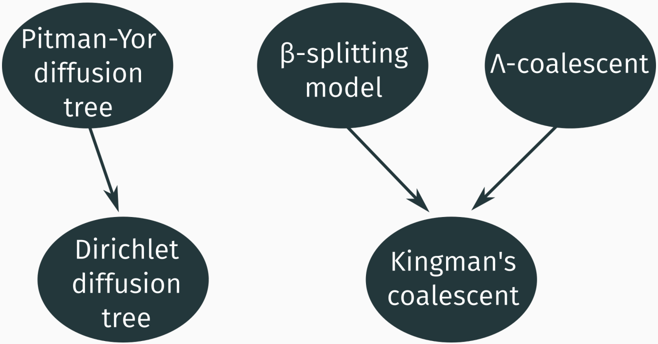
# Generalizations



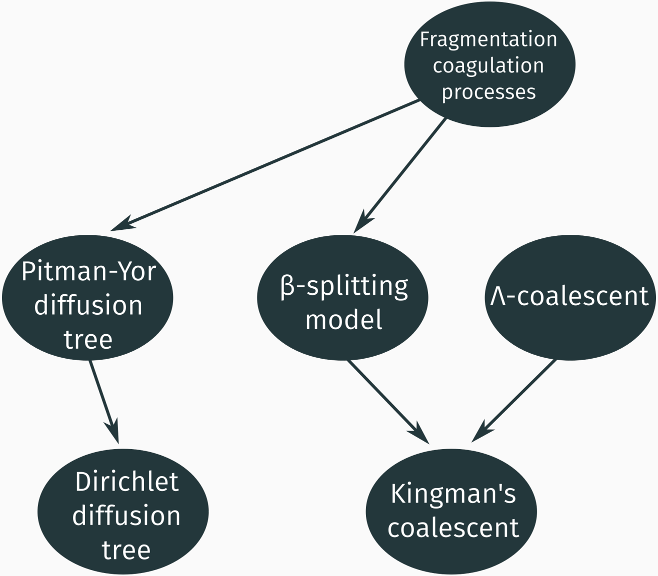
## Generalizations



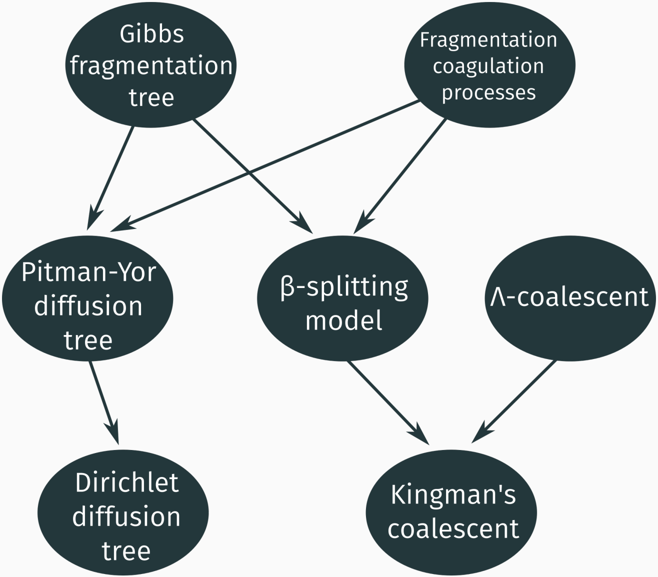
## Generalizations



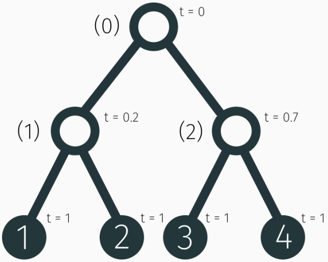
# Generalizations



# Generalizations

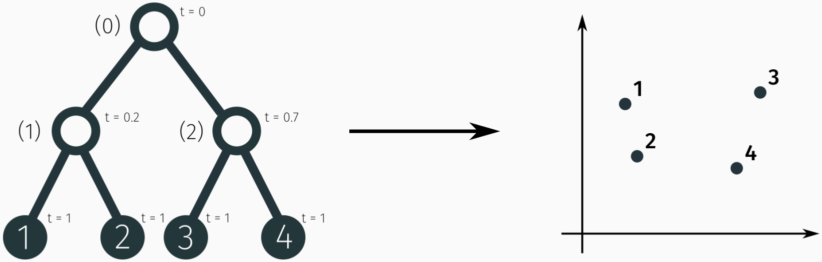


Now, given a sample from a tree prior (assume a cladogram with ordering and times), we need to generate a dataset.



# Tree likelihood models

Now, given a sample from a tree prior (assume a cladogram with ordering and times), we need to generate a dataset.

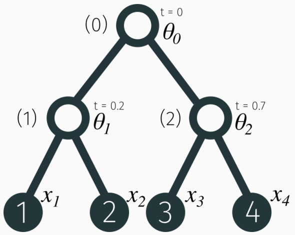


# Latent tree parameters

For every internal node in the tree $n$ we associate a latent parameter $\theta_n$ which is in the same space as the data.

# Latent tree parameters

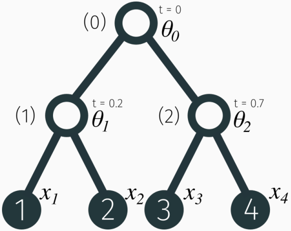
For every internal node in the tree $n$ we associate a latent parameter $\theta_n$ which is in the same space as the data.




We then define:

# Latent tree parameters

For every internal node in the tree $n$ we associate a latent parameter $\theta_n$ which is in the same space as the data.

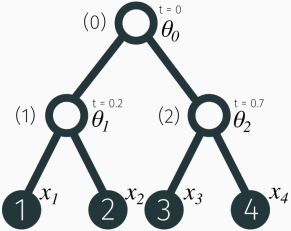



We then define:

- A prior distribution $P_0(\theta)$

# Latent tree parameters

For every internal node in the tree $n$ we associate a latent parameter $\theta_n$ which is in the same space as the data.




We then define:

- A prior distribution $P_0(\theta)$
- A transition kernel $T(\theta'|\theta)$

# Example likelihood models

The most common likelihood model is Brownian motion, also called Gaussian diffusion, which is

$$P_0(\theta) = \mathcal{N}(0, \sigma_0^2 I) \tag{5}$$

$$T(\theta'|\theta) = \mathcal{N}(\theta, \sigma^2(t - t')), \tag{6}$$

where $t$ and $t'$ are the times associated with nodes $\theta$ and $\theta'$ and $\sigma_0^2$ and $\sigma^2$ are hyperparameters.

# Example likelihood models

The most common likelihood model is Brownian motion, also called Gaussian diffusion, which is

$$P_0(\theta) = \mathcal{N}(0, \sigma_0^2 I) \tag{5}$$

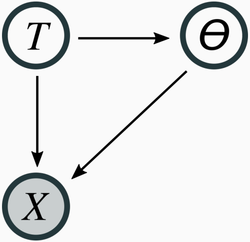
$$T(\theta'|\theta) = \mathcal{N}(\theta, \sigma^2(t - t')), \tag{6}$$

where $t$ and $t'$ are the times associated with nodes $\theta$ and $\theta'$ and $\sigma_0^2$ and $\sigma^2$ are hyperparameters.

Other options are:

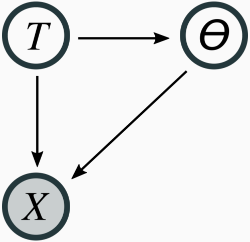
- Multinomial-Dirichlet diffusion: useful for categorical data
- Multinomial diffusion: useful for counts (such as bag-of-words)

Recall the latent variable model.



# Inference

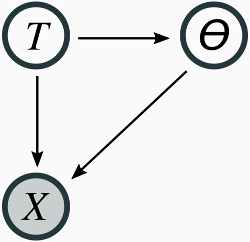
Recall the latent variable model.




We are interested in computing the posterior distribution $P(T, \theta | X)$ and the posterior marginal distribution $P(T|X)$.

# Inference

Recall the latent variable model.




We are interested in computing the posterior distribution $P(T, \theta | X)$ and the posterior marginal distribution $P(T|X)$.

We generally use approximate methods. In this report, we'll focus on a particular sampling method.

# Metropolis-Hastings

The Metropolis-Hastings (MH) algorithm is an MCMC method that uses samples from a distribution $p(x)$ to approximate it.

# Metropolis-Hastings

The Metropolis-Hastings (MH) algorithm is an MCMC method that uses samples from a distribution $p(x)$ to approximate it.

## Metropolis-Hastings algorithm

- Given initial distribution $p_0(x)$ and proposal distribution $q(x'|x)$
- Instantiate $x_0$ by sampling $p_0(x)$.
- Repeat for $t = 1 \dots T$
    - Sample $x'$ from $q(x'|x_{t-1})$.
    - Calculate acceptance ratio

$$\alpha = \frac{p(x')q(x_t|x')}{p(x_t)q(x'|x_t)} \tag{7}$$

    - If $\alpha > 1$, accept the sample, setting $x_t = x'$
    - If $\alpha \leq 1$, accept $x'$ with probability $\alpha$ and reject otherwise, setting $x_t = x_{t-1}$.

# Sampling latent variables

We are interested in sampling $P(T, \theta | X)$.

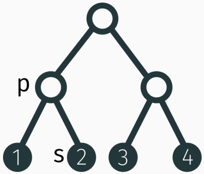
- Define tree proposal $q_{\mathcal{T}}(T' | T)$
- Define parameter proposal $q_{\Theta}(\theta' | T', \theta)$.

# Sampling latent variables

We are interested in sampling $P(T, \theta | X)$.

- Define tree proposal $q_{\mathcal{T}}(T' | T)$
- Define parameter proposal $q_{\Theta}(\theta' | T', \theta)$.

**Tree proposal:** subtree-prune and regraft (SPR) move

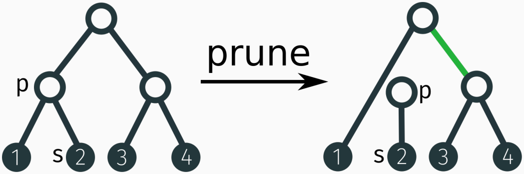


# Sampling latent variables

We are interested in sampling $P(T, \theta | X)$.

- Define tree proposal $q_{\mathcal{T}}(T' | T)$
- Define parameter proposal $q_{\Theta}(\theta' | T', \theta)$.

**Tree proposal:** subtree-prune and regraft (SPR) move

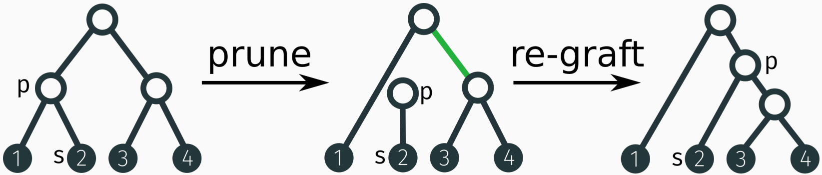


# Sampling latent variables

We are interested in sampling $P(T, \theta | X)$.

- Define tree proposal $q_{\mathcal{T}}(T' | T)$
- Define parameter proposal $q_{\Theta}(\theta' | T', \theta)$.

**Tree proposal:** subtree-prune and regraft (SPR) move

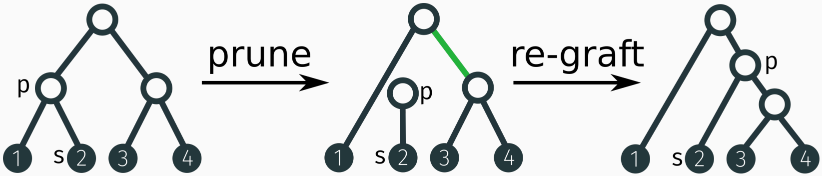


# Sampling latent variables

We are interested in sampling $P(T, \theta | X)$.

- Define tree proposal $q_{\mathcal{T}}(T' | T)$
- Define parameter proposal $q_{\Theta}(\theta' | T', \theta)$.

**Tree proposal:** subtree-prune and regraft (SPR) move




**Parameter proposal:** Gibbs sampling or marginalization

# Adding interaction

Background
- Clustering
- Bayesian learning

Bayesian hierarchical clustering
- Coalescent models
- Diffusion models
- Inference

Adding interaction

Conclusion

## Why interaction?

Recall the motivating example for Bayesian hierarchical clustering.

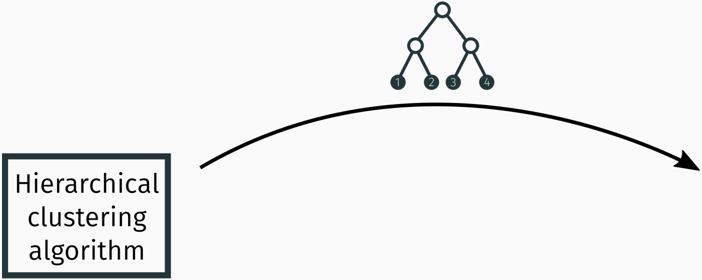

Could a user provide feedback and help decide which hierarchy makes the most sense?

- Interactive Bayesian Hierarchical Clustering - Vikram and Dasgupta (2016) [5]

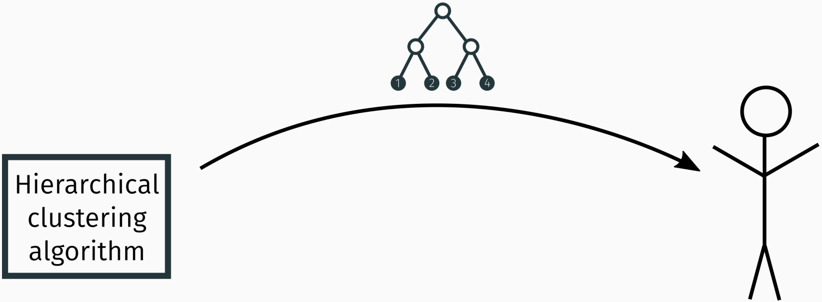
# Interactive hierarchical clustering

Hierarchical clustering algorithm

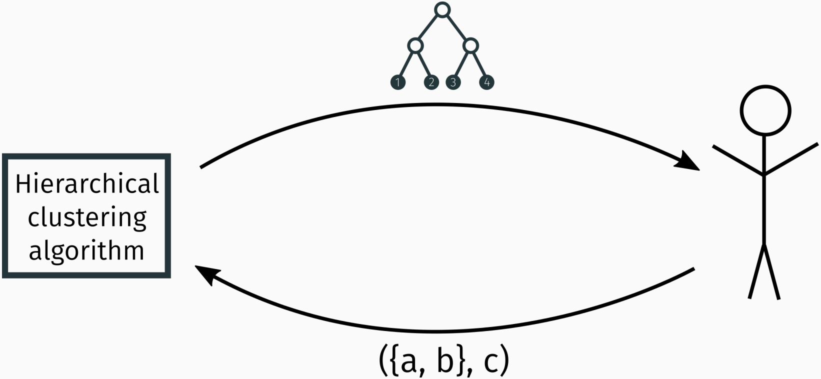
# Interactive hierarchical clustering



# Interactive hierarchical clustering



# Interactive hierarchical clustering



# Subtree queries

Two main ideas:

# Subtree queries

Two main ideas:

**Enforcing constraints**

To enforce constraints, we use a modified SPR move that avoids regraft branches that violate constraints.

# Subtree queries

Two main ideas:

**Enforcing constraints**

To enforce constraints, we use a modified SPR move that avoids regraft branches that violate constraints.

**Intelligent subset queries**

We pick subsets of data that have high variance, which we can measure using samples.

# Results

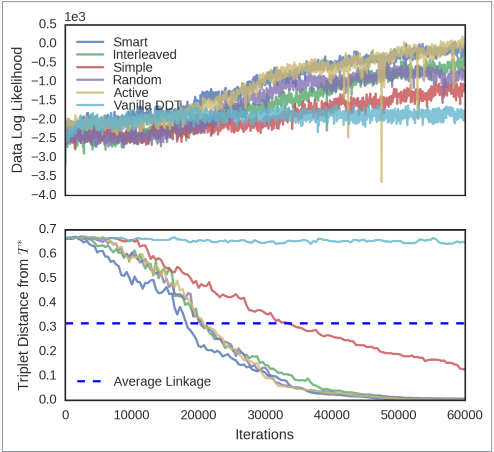



**Figure 2:** The results of interactive Bayesian hierarchical clustering on a dataset of zoo animals. Pictured are the data log likelihood and percentage of triplets satisfied for several subset querying methods. Source: [5]

# Conclusion

Background
- Clustering
- Bayesian learning

Bayesian hierarchical clustering
- Coalescent models
- Diffusion models
- Inference

Adding interaction

Conclusion

# Summary

- Bayesian hierarchical clustering (BHC) is a general framework in unsupervised learning that naturally handles ambiguity and uncertainty in data.

# Summary

- Bayesian hierarchical clustering (BHC) is a general framework in unsupervised learning that naturally handles ambiguity and uncertainty in data.
- BHC models can be decomposed into prior distributions on trees, and likelihood models. Tree priors can further be decomposed into coalescent and diffusion models.

# Summary

- Bayesian hierarchical clustering (BHC) is a general framework in unsupervised learning that naturally handles ambiguity and uncertainty in data.
- BHC models can be decomposed into prior distributions on trees, and likelihood models. Tree priors can further be decomposed into coalescent and diffusion models.
- Inference in BHC can be performed with MCMC methods like Metropolis-Hastings using the subtree-prune and regraft move.

# Summary

- Bayesian hierarchical clustering (BHC) is a general framework in unsupervised learning that naturally handles ambiguity and uncertainty in data.
- BHC models can be decomposed into prior distributions on trees, and likelihood models. Tree priors can further be decomposed into coalescent and diffusion models.
- Inference in BHC can be performed with MCMC methods like Metropolis-Hastings using the subtree-prune and regraft move.
- BHC enables incorporating user interaction into hierarchical clustering.

# Future work

- How can we improve on interactive Bayesian hierarchical clustering? Robust constraints, more tree priors, other measures of variance

# Future work

- **How can we improve on interactive Bayesian hierarchical clustering?** Robust constraints, more tree priors, other measures of variance
- **What is the effect of interaction in Bayesian problems?** constrained posterior distributions, complexity of inference

# Future work

- **How can we improve on interactive Bayesian hierarchical clustering?** Robust constraints, more tree priors, other measures of variance
- **What is the effect of interaction in Bayesian problems?** constrained posterior distributions, complexity of inference
- **How can we extend interactive methods to other domains?** metric learning, deep learning, embeddings

Questions?

# References I


- Trevor Hastie, Robert Tibshirani, and Jerome Friedman.
  *The Elements of Statistical Learning*.
  2009.
- J.F.C. Kingman.
  The coalescent.
  *Stochastic Processes and their Applications*, 1982.
- Levi Boyles and Max Welling.
  The time-marginalized coalescent prior for hierarchical clustering.
  *Advances in Neural Information Processing ...*, 2012.
- Radford M. Neal.
  Density modeling and clustering using Dirichlet diffusion trees.
  *Bayesian Statistics*, 2003.

Sharad Vikram and Sanjoy Dasgupta.
**Interactive Bayesian Hierarchical Clustering.**
2016.

Jim Pitman.
**Coalescents With Multiple Collisions.**
*The Annals of Probability*, 27(4):1870--1902, oct 1999.

David A. Knowles and Zoubin Ghahramani.
**Pitman Yor Diffusion Trees for Bayesian Hierarchical Clustering.**
*IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

## Time-marginalized coalescent

- Kingman's coalescent produces cladograms with ordering and times.

# Time-marginalized coalescent

- Kingman's coalescent produces cladograms with ordering and times.
- Marginalizing out the time is easy, since it's generated independently. Node ordering is not so easy.

# Time-marginalized coalescent

- Kingman's coalescent produces cladograms with ordering and times.
- Marginalizing out the time is easy, since it's generated independently. Node ordering is not so easy.
- The number of possible node orderings for a given unordered binary tree with $N$ leaves is

$$\frac{(N-1)!}{\prod_{i=1}^{N-1} m_i} \tag{8}$$

where $m_i$ is the number of internal nodes in the subtree indexed by $i$.

# TMC vs. Kingman's coalescent

Consider an ordered cladogram $\phi$ and an unordered cladogram $\psi$, both with $N$ leaves.

- Kingman's coalescent induces the uniform distribution over ordered cladograms.

$$P(\phi) = \left( \prod_{i=1}^{N} \binom{i}{2} \right)^{-1}$$

# TMC vs. Kingman's coalescent

Consider an ordered cladogram $\phi$ and an unordered cladogram $\psi$, both with $N$ leaves.

- Kingman's coalescent induces the uniform distribution over ordered cladograms.

$$P(\phi) = \left( \prod_{i=1}^{N} \binom{i}{2} \right)^{-1}$$

- The TMC is the induced distribution over unordered cladograms.

$$P(\psi) = \frac{(N-1)!}{\prod_{i=1}^{N-1} m_i} \left( \prod_{i=1}^{N} \binom{i}{2} \right)^{-1}$$

# TMC vs. Kingman's coalescent

Consider an ordered cladogram $\phi$ and an unordered cladogram $\psi$, both with $N$ leaves.

- Kingman's coalescent induces the uniform distribution over ordered cladograms.

$$P(\phi) = \left( \prod_{i=1}^{N} \binom{i}{2} \right)^{-1}$$

- The TMC is the induced distribution over unordered cladograms.

$$P(\psi) = \frac{(N-1)!}{\prod_{i=1}^{N-1} m_i} \left( \prod_{i=1}^{N} \binom{i}{2} \right)^{-1}$$

- Kingman's coalescent and TMC favor balanced trees!

# The $\Lambda$-coalescent

- The $\Lambda$-coalescent generalizes Kingman's coalescent to both multifurcating trees and more complex distributions over times [6].

# The $\Lambda$-coalescent

- The $\Lambda$-coalescent generalizes Kingman's coalescent to both multifurcating trees and more complex distributions over times [6].
- Rather than $\delta_i \sim \mathrm{Exp}\left(\binom{N-i+1}{2}\right)$, we have

$$\lambda_i^k = \int_0^1 \gamma^{k-2}(1-\gamma)^{(i-k)}\Lambda(d\gamma) \tag{9}$$

$$\delta_i \sim \mathrm{Exp}\left(\lambda_i^k\right), \tag{10}$$

for finite measure $\Lambda$ and max branching factor $k$.

# The $\Lambda$-coalescent

- The $\Lambda$-coalescent generalizes Kingman's coalescent to both multifurcating trees and more complex distributions over times [6].
- Rather than $\delta_i \sim \mathrm{Exp}\left(\binom{N-i+1}{2}\right)$, we have

$$\lambda_i^k = \int_0^1 \gamma^{k-2}(1-\gamma)^{(i-k)}\Lambda(d\gamma) \tag{9}$$

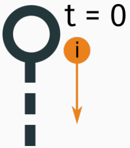
$$\delta_i \sim \mathrm{Exp}\left(\lambda_i^k\right), \tag{10}$$

  for finite measure $\Lambda$ and max branching factor $k$.
- When $\Lambda$ is the Dirac delta and $k = 2$, we have Kingman's coalescent.

# Pitman-Yor diffusion tree

The DDT has been extended to multifurcating trees with the Pitman-Yor diffusion tree (PYDT) [7].

The $i$-th particle behaves slightly differently, but starts in the same way.



# Pitman-Yor diffusion tree

**Case 1:** The particle reaches an internal node that has $b$ branches already.

# Pitman-Yor diffusion tree

**Case 1:** The particle reaches an internal node that has $b$ branches already.


It picks the $k$-th branch with probability

$$\frac{m_k - \beta}{m + \alpha} \tag{11}$$

where $m_k$ is the number of past particles that have traversed the $k$-th branch, $m$ is the total number of particles that have traversed this subtree, and $\alpha$ and $\beta$ are hyperparameters.

# Pitman-Yor diffusion tree

**Case 2:** The particle reaches an internal node that has $b$ branches already.

# Pitman-Yor diffusion tree

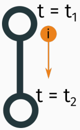
**Case 2:** The particle reaches an internal node that has $b$ branches already.


It creates a new branch with probability

$$\frac{\alpha + \beta b}{m + \alpha} \tag{12}$$

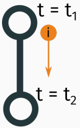
# Pitman-Yor diffusion tree

**Case 3:** The particle *diverges* on its current branch, creating an internal node and a leaf. The probability of divergence is calculated from an acquisition function, $a(t)$.



# Pitman-Yor diffusion tree

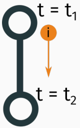
**Case 3:** The particle *diverges* on its current branch, creating an internal node and a leaf. The probability of divergence is calculated from an acquisition function, $a(t)$.




Let $m$ be the number of past particles that have traversed the current branch. The probability of diverging at time $dt$ is

# Pitman-Yor diffusion tree

**Case 3:** The particle *diverges* on its current branch, creating an internal node and a leaf. The probability of divergence is calculated from an acquisition function, $a(t)$.




Let $m$ be the number of past particles that have traversed the current branch. The probability of diverging at time $dt$ is

$$\frac{a(t)\Gamma(m-\beta)dt}{\Gamma(m+1+\alpha)} \tag{13}$$

# Enforcing triplet constraints

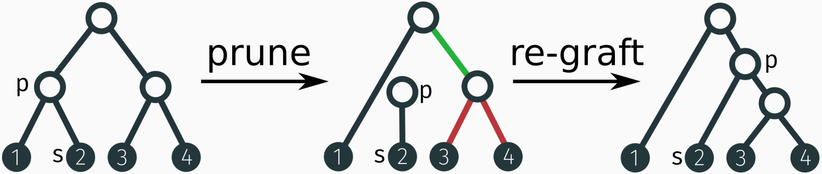
We can easily enforce triplet constraints in Bayesian hierarchical clustering.

During inference, we use MH with a modified SPR proposal called the constrained-SPR proposal which will only select regraft branches that don't violate a set of triplet constraints.

# Enforcing triplet constraints

We can easily enforce triplet constraints in Bayesian hierarchical clustering.

During inference, we use MH with a modified SPR proposal called the constrained-SPR proposal which will only select regraft branches that don't violate a set of triplet constraints.



# Intelligent subset queries

The simplest way to select a subset of data to show the user is random selection, we can use the Bayesian framework to choose better subsets.

In BHC, we compute a posterior distribution over trees given data. Using the posterior, we can estimate the variance of various regions of the tree and show the user the region with the most variance.

**Definition: tree distance variance (TDV)**

Given a subset of data $S$ and tree samples $\mathcal{T} = T_1, \ldots, T_N$,

$$\mathrm{TDV}(S, \mathcal{T}) = \max_{i,j \in S} \mathrm{Var}_{T \in \mathcal{T}} \left[ \texttt{tree-dist}_{T|S}(i, j) \right] \tag{14}$$

where $\texttt{tree-dist}_T$ is the number of edges needed to get from leaf $i$ to leaf $j$ in tree $T$.

We now instantiate several random subsets and show the user the one with the highest TDV.