EFFICIENT ONLINE BAYESIAN HIERARCHICAL CLUSTERING

An Honors Thesis

Presented by
Lynn Samson

Completion Date:
May 2018

Approved By:
Andrew McCallum, College Of Information And Computer Sciences

Akshay Krishnamurthy, College Of Information And Computer Sciences
Hierarchical structures are ubiquitous in the world, ranging from the evolutionary and genetic tree of living organisms, to relationships within social networks, to the organization of documents, etc. With the prevalence of such hierarchical structures and data that describes them, it is important to have methods of analyzing and understanding these hierarchies. Bayesian hierarchical clustering is such a model for building a hierarchical representation of such data that can also be interpreted as a clustering of the data points into their respective categories within the hierarchy. However, existing inference algorithms for BHC are either too slow or too inaccurate. We attempt to remedy this by introducing a new inference method for BHC based on the PERCH hierarchical clustering algorithm, which is able to provide a fast, but also non-greedy clustering based on recursive rotations of subtrees.
Title: Efficient Online Bayesian Hierarchical Clustering
Author: Lynn Samson
Thesis/Project Type: Independent Honors Thesis
Approved By: Andrew Mccallum, College Of Information And Computer Sciences
Approved By: Akshay Krishnamurthy, College Of Information And Computer Sciences

Hierarchical structures are ubiquitous in the world, ranging from the evolutionary and genetic tree of living organisms, to relationships within social networks, to the organization of documents, etc. With the prevalence of such hierarchical structures and data that describes them, it is important to have methods of analyzing and understanding these hierarchies. Bayesian hierarchical clustering is such a model for building a hierarchical representation of such data that can also be interpreted as a clustering of the data points into their respective categories within the hierarchy. However, existing inference algorithms for BHC are either too slow or too inaccurate. We attempt to remedy this by introducing a new inference method for BHC based on the PERCH hierarchical clustering algorithm, which is able to provide a fast, but also non-greedy clustering based on recursive rotations of subtrees.
1 Introduction

Bayesian Hierarchical Clustering (BHC) [5] is a model for hierarchical clustering based on evaluating marginal likelihoods of a probabilistic model, as opposed to distance metrics, such as those utilized by traditional clustering algorithms. Probabilistic clustering has several advantages over distance metric-based clustering, such as the ability to capture uncertainty in the distribution of data points, the ability to work with any type of data (e.g. categorical), and a meaningful measure of how likely multiple points are together in a cluster. In particular, BHC has the ability to capture rich hierarchies in data due to the hierarchical representation it builds over the data. BHC is a mixture model over the flat partitions which can be represented by the tree structure. This allows it to represent and express uncertainty about multiple alternative flat clusterings.

Both an agglomerative bottom-up inference method [5], as well as multiple top-down inference methods [4] have been implemented for BHC. The approach most commonly used for BHC, and which is described in the original paper [5] is the agglomerative method, which we will term BUBHC. Although BUBHC implements all of the desirable properties described above, it is held back by the $O(n^3)$ complexity of building the tree, causing it to run very slowly, even on reasonably sized datasets. On the other hand, the two top-down methods described in [4], which are termed RBHC and EMBHC, are much faster, running in $O(n \log n)$ and $O(knm)$ time, respectively. RBHC and EMBHC improve upon the speed of BUBHC by utilizing randomized algorithms. Although randomized algorithms allow for speeding up the clustering significantly, the resulting tree is often
of lower quality than that produced by BUBHC.

The problem that I addressed with my research is the need for a faster algorithm than the agglomerative approach, and which would also create a higher quality clustering than the randomized top-down approaches. The basis for developing this new algorithm is an inference method inspired by the recently developed PERCH algorithm [6]. PERCH is a top-down online algorithm for hierarchical clustering, which uses an inference method based on approximate distance measures and tree rotations in order to achieve very fast and accurate clusterings. In fact, under a separability assumption on the data, PERCH is guaranteed to construct a clustering tree with perfect dendrogram purity. My research investigated how PERCH’s inference method can be adapted to work with BHC’s model.

My approach to this problem initially consisted of attempting to compute the full BHC marginal tree probability model by computing the terms bottom-up using PERCH’s inference method. However, while estimating the likelihood terms lower down in the tree, numerical instability issues that rendered this impractical. However, we were able to approximate the model to a satisfactory degree by only using the likelihood term to perform rotation using PERCH’s inference method. The results show that our method is much faster than the original bottom-up inference for BHC, and more accurate than the existing fast randomized inference methods.
2 Significance

Hierarchical structures are ever so present in the world, as well as in numerous active areas of research. They range from the evolutionary and genetic tree of living organisms, to relationships within social networks, to organization of documents in newsgroups, emails, and academic journals. With such a strong presence of hierarchies and the constant stream of new data describing them, it is more crucial than ever to have methods to analyze and understand hierarchical structures well. Developing a method to obtain fast, high quality hierarchical clusterings would be highly beneficial to studying these types of structures more effectively and efficiently. While BUBHC is able to provide high quality clusterings, it is much too slow for most real-world datasets. On the other hand, we want a method that is guaranteed to build higher quality clusterings and which is more consistent than the existing randomized inference methods RBHC and EMBHC. Our method is able to deliver both, with an inference procedure inspired by PERCH that is fast, yet non-greedy.

3 Background

3.1 Bayesian Learning

The basis of Bayesian learning is the maximum a posteriori (MAP) estimate, which is defined as follows:

\[ \hat{\theta}_{\text{MAP}}(x) = \arg \max_{\theta} p(\theta | x) = \arg \max_{\theta} \frac{p(x | \theta) p(\theta)}{\int_v p(x | v) p(v) dv} = \arg \max_{\theta} p(x | \theta) p(\theta) \]

The MAP estimate essentially maximizes the posterior probability of the target distribution’s
parameters $\theta$, given a particular data point $x$. The MAP estimate assumes a prior distribution over $\theta$. A special case of MAP known as the maximum likelihood estimate (MLE) treats $\theta$ as a uniform value, and thus, solely maximizes the likelihood function $p(x \mid \theta)$. The MLE of $\theta$ is given by:

$$\hat{\theta}_{\text{MLE}}(x) = \arg\max_\theta p(x \mid \theta)$$

One particularly important aspect of Bayesian learning is that of conjugate priors. Suppose we have a likelihood function $p(D \mid \theta)$ depending on a parameter $\theta$. Then, if the posterior distribution $p(\theta \mid D)$ is in the same parametrized family of distributions as the prior distribution $p(\theta)$, we say that the prior distribution is a conjugate prior for the likelihood. Conjugate priors allow us to get a closed form representation of the posterior distribution. Another very important concept in Bayesian learning is that of proportionality. A function $f$ is proportional to another function $g$, denoted $f \propto g$, if $g(x) = cf(x)$ for all $x$ and nonzero constants $c$. Then, it holds that if $f$ is a probability density function and $f \propto g$, then $f$ is uniquely determined by $g$ and $f(x) = \frac{g(x)}{\int g(x)dx}$, and thus, the constant can be ignored altogether. To verify, $f(x) = \frac{g(x)}{\int g(x)dx} = \frac{cf(x)}{\int cf(x)dx} = \frac{f(x)}{\int f(x)dx} = \frac{f(x)}{1} = f(x)$. The conjugacy of certain probability distributions as well as our ability to ignore constants due to proportionality allow us to easily calculate the posterior distributions of useful distributions such as the binomial or multinomial distributions, as we will do in the next sections.

### 3.2 Beta-Binomial

The beta distribution is the conjugate prior of the binomial distribution. That is, if the likelihood is binomial and the prior is a beta distribution, then the posterior is also a beta distribution. In order to
see this, recall that when we have an experiment consisting of \( n \) Bernoulli trials with a probability parameter \( p \), the number of successful trials, denoted by the random variable \( X \) has a binomial distribution with parameters \( n \) and \( p \), where \( n \) is the total number of trials and \( p \) is the probability of success for each trial. We write this as \( X \sim \text{Bin}(n,p) \) and the density function of \( X \) can be explicitly written as \( p(X = k) = \binom{n}{k}p^k(1-p)^{n-k} \) for \( k \in \{0, 1, 2, \ldots, n\} \). Now, we assume that the prior distribution of the probability parameter \( p \) is a beta distribution with hyperparameters \( \alpha \) and \( \beta \). We write this as \( p \sim \text{Beta}(\alpha, \beta) \) and the density function of \( p \) can be explicitly written as \( p(P = p) = \frac{1}{B(\alpha, \beta)}p^{\alpha-1}(1-p)^{\beta-1} \) for \( p \in (0, 1) \). Note that this assumption makes intuitive sense, since the support of a beta random variable is between 0 and 1, just as it is for a probability.

Assuming that the likelihood is binomial with parameters \( n \) and \( p \), the observed data \( k \) will be an integer value between 0 and \( n \). That is, the likelihood looks like \( \binom{n}{k}p^k(1-p)^{n-k} \) and the beta prior looks like \( \frac{1}{B(\alpha, \beta)}p^{\alpha-1}(1-p)^{\beta-1} \). Multiplying the two, we get the posterior distribution \( p(\theta \mid D) = \frac{\binom{n}{k}p^{\alpha+k-1}(1-p)^{\beta+n-k-1}}{B(\alpha, \beta)} \). Then, by ignoring the normalizing constants, we can see that

\[
p(\theta \mid D) \propto p^{\alpha+k-1}(1-p)^{\beta+n-k-1} = \text{Beta}(\alpha + k, \beta + n - k)
\]

In other words, the posterior distribution is proportional to a beta distribution with updated parameters according to the observed data \( k \), and is itself a beta distribution with parameters \( \alpha + k \) and \( \beta + n - k \).
3.3 Dirichlet-Multinomial

The multinomial distribution is the multivariate analogue of the binomial distribution. Hence, there are $m$ categories $X = \{1, 2, \ldots, m\}$ rather than just the two categories of success and failure, as in the case of the binomial distribution. The probabilities for these $m$ categories is given by the probability vector $\theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$, with $\sum_{i=1}^{m} \theta_i = 1$. All such $\theta$’s make up what is known as the $m$-dimensional probability simplex. If $X$ is a multinomial random variable, we write $X \sim \text{Multi}(n, \theta)$ and the density can be explicitly written as

$$p(X = k_1, \ldots, k_m) = \frac{n!}{k_1! \ldots k_m!} \prod_{i=1}^{m} \theta_i^{k_i}$$

where $k_i$ is the count of the number of times that category $i$ appears in the sample. The Dirichlet distribution is the multivariate analogue of the beta distribution, and as such, can be specified as a prior distribution over this $m$-dimensional simplex. Thus, for any probability vector $\theta$ over the $m$ categories, we write $\theta \sim \text{Dir}(\alpha)$, where $\alpha$ is an $m$-dimensional vector of hyperparameters $\alpha_i$, all of which are strictly positive. We write the density of $\theta$ as $p(\theta = \theta_1, \ldots, \theta_m) = \frac{1}{B(\alpha)} \prod_{i=1}^{m} \theta_i^{\alpha_i-1}$. Now we want to show the fact that the Dirichlet distribution is the conjugate prior of the multinomial distribution, as the beta distribution is the conjugate prior of the binomial distribution. Then, as in the case of the beta and binomial distributions, we assume that the likelihood is multinomial, in which case it looks like $\frac{n!}{k_1! \ldots k_m!} \prod_{i=1}^{m} \theta_i^{k_i}$, and the Dirichlet prior looks like $\frac{1}{B(\alpha)} \prod_{i=1}^{m} \theta_i^{\alpha_i-1}$. Then, we get the posterior distribution $p(\theta \mid D) = \frac{n!}{k_1! \ldots k_m!B(\alpha)} \prod_{i=1}^{m} \theta_i^{k_i+\alpha_i-1}$. Multiplying both, we get

$$p(\theta \mid D) = \frac{n!}{k_1! \ldots k_m!B(\alpha)} \prod_{i=1}^{m} \theta_i^{k_i+\alpha_i-1},$$

where $k$ is the vector of counts for each category and $\alpha$ is the vector of hyperparameters for the Dirichlet prior. As in the case of the beta and the binomial, we
can ignore the normalizing constants due to proportionality, and we can see that

\[ p(\theta \mid D) \propto \prod_{i=1}^{m} \theta_k^{k_i+\alpha_i-1} = Dir(k + \alpha) \]

### 3.4 Finite Mixture Models

Mixture models are probabilistic methods for clustering where it is believed that the data points are generated by multiple distributions that can be interpreted as subpopulations of an overall population. The likelihood of a mixture model with \( N \) data points and \( K \) clusters can be expressed as follows:

\[
\prod_{i=1}^{N} \sum_{j=1}^{K} \phi_j P(x_i \mid \theta_j)
\]

Here, \( \phi_j \) expresses the prior probability of component \( j \); that is, the prior belief that the data point was generated from the distribution corresponding to cluster \( j \), and \( P(x_i \mid \theta_j) \) expresses the likelihood of data point \( i \) under the distribution at cluster \( j \), which is described by the set of parameters \( \theta_j \). The sum over all clusters \( j \) for a particular data point \( x_i \) is its overall likelihood in the mixture model, where the contribution from the distribution at cluster \( j \) is proportional to the prior \( \phi_j \). In this case, the standard method of maximum likelihood estimation by differentiating the log likelihood and solving for 0 is analytically intractable due to the sum over all data points. Hence, the most common alternative algorithm used for learning mixture models is known as the Expectation-Maximization (EM) algorithm [2]. The EM algorithm works because of the fact that it is easy to compute values for the parameters \( \theta_j \) and \( \phi_j \) if the cluster assignment \( C_j \) for each \( x_i \) is known, and vice versa, it is easy to compute \( P(C_j \mid x_i) \) if we have values for the parameters \( \theta_j \) and
\( \phi_j \). The former is known as the "maximization" step and the latter is known as the "expectation" step. We start out with initial random values for the parameters, and alternate between the expectation and maximization steps. We iterate this process until convergence to the maximum likelihood estimate of the model parameters. Since a Gaussian distribution is not an unreasonable assumption for the generative source of most real-world data, the most common type of mixture model is the Gaussian mixture model (GMM). Similar to how a unimodal Gaussian distribution has a peak around the mean (or centre) of the data, a GMM (which is multimodal) has several peaks corresponding to the means of the subpopulations within the overall population.

3.5 Dirichlet Process Mixture Models

When working with finite mixture models, the number of clusters \( K \) has to be guessed based on a-priori beliefs about the data. In practice, several models fitted with different values of the parameter \( K \) have to be compared in order to settle on a satisfactory value. By taking the limit as the number of model parameters go to infinity, we can derive a nonparametric or infinite mixture model [7]. This infinite mixture model will be able to automatically infer the number of clusters based on the data, rather than an arbitrary a-priori guess. This type of mixture model with infinitely many components is known as a Dirichlet process mixture model.

3.6 Bayesian Hierarchical Clustering

Bayesian hierarchical clustering [5] (BHC) is a model for clustering data based on marginal likelihoods of a probabilistic model. In the original paper [5], a bottom up inference procedure
(BUBHC) is given. It is a batch procedure that starts with all $N$ data points in their own trivial cluster, such that for any point $x_i$, there is a cluster containing the data point $D_i = \{x_i\}$. Like traditional agglomerative clustering, at each step of the algorithm, all pairs of clusters are considered for a merge. When two clusters $D_i$ and $D_j$ are considered for a merge, their hypothetical merged cluster is represented by $D_k$, and the resulting hypothetical subtree tree is represented by $T_k$. Then, the following value is computed for each pair of clusters:

$$r_k = \frac{\pi_k P(D_k \mid H_k^1)}{P(D_k \mid T_k)}$$

where $P(D_k \mid H_k^1)$ is the probability that the hypothetical merged cluster of data should, in fact, be under one cluster. The normalizing value $P(D_k \mid T_k)$ is the marginal probability of the tree $T_k$, which is defined as $P(D_k \mid T_k) = \pi_k P(D_k \mid H_k^1) + (1 - \pi_k) P(D_i \mid T_i) P(D_j \mid T_j)$. In addition to the hypothesis that the data at $D_k$ should be under one cluster, the marginal probability of tree $T_k$ also considers the alternative hypothesis that the data at $D_k$ should be under two or more different tree-consistent clusters, represented by $P(D_i \mid T_i) P(D_j \mid T_j)$. The prior probability $\pi_k$ is the Dirichlet prior value that $D_k$ is a new clustering, and conversely, $1 - \pi_k$ is the prior value that $D_k$ can be represented by some existing (tree-consistent) clustering. At each step, the $r_k$ probability of a merge is maximized over all possible pairs of clusters, and the pair of clusters with the maximum $r_k$ probability are merged. In the end, the algorithm results in a hierarchical mixture model, where each node of the tree can be considered a component of the mixture. Because a probabilistic measure is used to merge clusters at each step, a flat clustering of the hierarchical tree can be obtained by cutting the tree at points where the merged hypothesis $r_k$ is less than 0.5. The pairwise comparisons
of clusters is repeated for \( N \) data points and so the BUBHC algorithm has a runtime efficiency of \( O(n^3) \).

### 3.7 Randomized Algorithms for Bayesian Hierarchical Clustering

To overcome BUBHC’s slow runtime of \( O(n^3) \), the authors introduced two faster inference algorithms for the BHC model, termed RBHC and EMBHC [4]. These algorithms are able to achieve much faster runtimes than BUBHC due to the fact that they are top-down and randomized. RBHC works in the following way: a small subset \( M \) of the total number of data points \( N \) is sampled and used to create a BHC tree \( T \) using the BUBHC algorithm. Then, the remaining points that were not sampled are filtered through the resulting tree \( T \). The filter algorithm takes the two top-level partitions \( D_L \) and \( D_R \) of tree \( T \), as well as their Dirichlet priors \( \pi_L \) and \( \pi_R \). Then, for each remaining data point \( x_i \), its probability of belonging to each subtree is computed, and it is added to the subtree with the highest probability of containing it. Then, RBHC recurses on the subtrees associated with partitions \( D_L \) and \( D_R \) of tree \( T \), resulting in a hierarchical mixture model. RBHC runs in \( O(n\log n) \) time, and is therefore, much faster than BUBHC. However, unlike BUBHC, RBHC makes two prior assumptions. The first assumption is that the random sample of size \( m \) taken at each recursive step of the algorithm is representative of the true top-level clustering of the data, as it will be in the BUBHC algorithm. The second assumption is that the RBHC algorithm will produce roughly balanced subtrees at each step of the recursion. In other words, it is assumed that at each step, roughly the same number of data points will be routed to the left and right subtrees. The first assumption is required in order for RBHC to build an accurate hierarchical clustering, and the
second assumption is required in order for RBHC to guarantee the \( O(n \log n) \) runtime. However, in practice, these assumptions may not always hold. The alternative fast inference algorithm, EMBHC, works in the following way: a small subset \( M \) of the total number of data points \( N \) is sampled and used to create \( M \) clusters centred at the respective data points. The remaining data points are then filtered into its most probable cluster using the filter algorithm, and then \( k \) steps of the EM algorithm is ran in order to refine the clusters. Then, EMBHC recurses on each of the \( M \) clusters, resulting in a hierarchical mixture model. EMBHC runs in \( O(knm) \) time, due to running the EM algorithm \( k \) times on the \( M \) clusters at each step of the recursion. EMBHC also makes the same assumptions as RBHC, and is thus, susceptible to the assumptions not holding in practice.

### 3.8 PERCH

Purity Enhancing Rotations for Cluster Hierarchies (PERCH) [6] is a top-down incremental hierarchical clustering algorithm that scales well to a large number of data points \( N \), as well as a large number of clusters \( K \), a problem setting termed "extreme clustering." The incremental nature of PERCH means that data points come in one at a time, rather than as a batch, and are routed down to their appropriate subtree (cluster). Each internal node in the tree maintains statistics summarizing the bounds of an approximate "box" structure (aptly termed "bounding box") that contains the points at the leaves of that internal node. This approximation structure allows for fast routing down the tree, as a new data point only has to perform comparisons with the bounds of the structure, rather than each data point at the leaf nodes. In addition, a recursive rotation condition is used to maintain a high dendrogram purity and balancedness of the tree, which is a crucial factor for the fast clustering
4 Methodology

In Section 3.7, we discuss two randomized inference algorithms for BHC: RBHC and EMBHC [4]. Although they are fast approximations to the BHC model, both algorithms are based on assumptions that may not be true in practice. The first of these assumptions is that the data sample of size $M$ taken at each recursive step of the algorithms is representative of the top-level clustering at that step. The second assumption is that data points will be routed in roughly equal proportions to each subtree corresponding to a split of the data.

4.1 Using PERCH-like Inference for BHC

This is our motivation to develop an alternative inference method for BHC based on something like PERCH. A PERCH-like inference would cluster data one point at a time by routing it down the tree to its nearest neighbor using A* search and applying a bottom-up recursive rotation procedure to correct the mistakes made by the A* algorithm, which is a greedy search. This rotation procedure essentially makes PERCH a non-greedy algorithm. It is easy to change the measure that PERCH uses to determine whether to rotate, which will be our main modification to the algorithm. In addition to its non-greedy nature, PERCH makes no assumptions about top-level splits based on arbitrary samples of the data, like RBHC and EMBHC do. Rather, the top-level splits might be constantly changing upon seeing new data due to PERCH’s rotation procedure. PERCH is not as
fast as RBHC and EMBHC, as it has a runtime of $O(n^2)$. However, in practice, it has been shown that PERCH runs in $O(n \text{polylog}(n))$ time.

### 4.2 Our Initial Approach

Our initial approach to create a new inference method for BHC using ideas borrowed from PERCH involved consuming points in a streaming fashion and routing each one to its nearest neighbor using A*. In place of PERCH’s standard distance-based rotations we planned to employ rotations based on the BHC model. Under the BHC model [5], the marginal tree probability of any subtree in the BHC tree is:

$$P(D_k | T_k) = \pi_k P(D_k | H^k_1) + (1 - \pi_k) P(D_i | T_i)p(D_j | T_j)$$

All parameters required to compute the marginal tree probability can be computed incrementally and bottom-up along the leaf-to-root path using PERCH, since PERCH already does bottom-up updates for its bounding box approximations after the addition of a new data point. In computing the likelihood values $P(D_k | H^k_1)$ bottom-up, we use maximum likelihood estimation (MLE) rather than conjugate prior-based estimation. This is because it is difficult to specify a sufficiently accurate prior for the data when the number of data points is small. In particular, we experimented with a Normal-Inverse Wishart (NIW) prior on the Gaussian density function and tested several different methods for estimating the parameters of the NIW distribution. For example, we used parameters estimated from the overall data set, parameters estimated based on the data points at each node, etc. Ultimately, we chose MLE over a prior based estimation because in doing so we are able to perfectly
cluster separated data (which PERCH is also able to do) when rotating using just the likelihood term $P(D_k \mid H^k_1)$. On the other hand, we are not able to cluster separated data perfectly when employing a prior based estimation.

4.3 Numerical Instability Issues

Although we are able to cluster some small datasets perfectly when estimating parameters via MLE and rotating by the likelihood $P(D_k \mid H^k_1)$, we encountered numerical issues when implementing the full marginal tree probability $P(D_k \mid T_k)$. The MLE computation at the leaf nodes overfits to the small number of data points, and essentially believes that the general parameters of the entire dataset look very close to those small number of data points seen at the leaves. Correspondingly, likelihoods at the leaf nodes are very large, and thus dominate the marginal tree probability when it is recursively computed up the tree in PERCH’s leaf-to-root update path. Due to numerical representation issues with very large numbers, the marginal tree likelihood values that are compared for rotation look the same. This results in subtrees not being rotated when they should be, and thus, critically affects the accuracy of the PERCH-like method.

4.4 Approximating the BHC Model

Because of the numerical instability issues, it is not viable to perform rotations based on the full marginal tree probability. Thus, we approximate the full marginal tree probability with a lower bound value. We observe that of the values for the individual terms in the full tree probability computation, most of the probability density that differentiates the subtrees are in the likelihood
term $P(D_k | H^k_1)$. This term corresponds to the probability that all the data in that subtree belongs to one cluster, rather than multiple tree-consistent clusters. Intuitively, we want to rotate based on this probability because it compares whether the data points resulting from a combination of one pair of subtrees is more likely to form a cluster than the data points resulting from a combination of the alternate pair of subtrees.

5 Results

5.1 Datasets

In order to evaluate our proposed approach against existing methods, we used three datasets. The first dataset we used is the Glass data set [3], in which each data point describes the features of a particular type of glass. We used two versions of this dataset, one with the full 10 dimensions and one with 2 dimensions, obtained by principal components analysis (PCA). The second dataset I used is the 20 Newsgroups data set, which contains data points corresponding to articles from 20 different newsgroups. The raw text data was featurized using tf-idf and the dimensions were reduced using latent semantic analysis (LSA)[1]. Finally, We used separated data, which was generated synthetically. In separated data, the maximum within-cluster distance is less than the minimum inter-cluster distance. Full descriptions of the datasets can be seen in table 1.
<table>
<thead>
<tr>
<th>Name</th>
<th>Data Points</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>214</td>
<td>9</td>
</tr>
<tr>
<td>PCA Glass</td>
<td>214</td>
<td>2</td>
</tr>
<tr>
<td>20 Newsgroups</td>
<td>120</td>
<td>50</td>
</tr>
<tr>
<td>Separated</td>
<td>1250</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 1: Datasets

5.2 Metrics

We evaluate each method using dendrogram purity (DP). Kobren et. al. define dendrogram purity as follows in their paper on the PERCH algorithm [6]. The dendrogram purity of a tree $T$ with a true clustering $C^*$ is

$$DP(T) = \sum_{k=1}^{K} \sum_{x_i, x_j \in C_k^* \text{pur}(lvs(LCA(x_i, x_j)), C_k^*)} \left| \{(x_i, x_j) \mid C^*(x_i) = C^*(x_j)\} \right|$$

where $\text{pur}(D_1, D_2) = \frac{|D_1 \cap D_2|}{|D_1|}$, $lvs(z)$ is the set of leaves for any internal node $z$ in $T$, and $LCA(x_i, x_j)$ is the least common ancestor of any two points $x_i$ and $x_j$ in $T$. This measure is adapted from the BUBHC paper [5], in which Heller et. al. empirically show that there is a positive correlation between dendrogram purity and marginal tree probabilities. Since we are not able to compute marginal tree probabilities due to numerical issues, this gives us confidence in using dendrogram purity as a universal metric across all the methods we compare.
### Table 2: DP vs. Time results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Glass</th>
<th>2D Glass (PCA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DP</td>
<td>Time</td>
</tr>
<tr>
<td>PERCH</td>
<td>0.4753 ± 0.01730</td>
<td>2.16254 ± 0.03366</td>
</tr>
<tr>
<td>BUBHC</td>
<td>0.47147 ± 0.00000</td>
<td>645.00457 ± 2.20783</td>
</tr>
<tr>
<td>RBHC</td>
<td>0.42591 ± 0.03692</td>
<td>0.73941 ± 0.08614</td>
</tr>
<tr>
<td>PERCH-BHC</td>
<td>0.46566 ± 0.02143</td>
<td>12.87056 ± 0.64254</td>
</tr>
</tbody>
</table>

### Table 3: DP vs. Time results (cont.)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Newsgroups</th>
<th>Separated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DP</td>
<td>Time</td>
</tr>
<tr>
<td>PERCH</td>
<td>0.30253 ± 0.02301</td>
<td>1.26086 ± 0.02325</td>
</tr>
<tr>
<td>BUBHC</td>
<td>0.20359 ± 0.00000</td>
<td>1050.550256</td>
</tr>
<tr>
<td>RBHC</td>
<td>0.14234 ± 0.01810</td>
<td>2.81994 ± 0.69718</td>
</tr>
<tr>
<td>PERCH-BHC</td>
<td>0.30225 ± 0.02470</td>
<td>5.96747 ± 0.49773</td>
</tr>
</tbody>
</table>
5.3 Experiments

The methods that we evaluated are PERCH, BUBHC, RBHC, and our PERCH-BHC method. The results of running these methods on the datasets described above are summarized in tables 2 and 3. They are also plotted in Figures 1-4. It is clear from the experiments that RBHC tends to construct trees with lower dendrogram purity than BUBHC, PERCH, and PERCH-BHC. The exception is the PCA Glass data set. Despite the shortcoming in DP, RBHC is a fast method, matching PERCH’s speed on every dataset. Unsurprisingly, BUBHC is slow, reflecting its runtime of $O(n^3)$. In fact, we were not able to run BUBHC to completion on the separated data. Within 7 hours, it was only able to finish 150 of the 1250 iterations. PERCH, BUBHC, and PERCH-BHC tend to achieve similar DP across all the datasets. This can be explained by positive correlation between DP and marginal tree probabilities reported in Heller et. al. [5]. Also, PERCH-likelihood is a much faster inference method that BUBHC, and also more accurate than RBHC. We were not able to test the performance of EMBHC, as we did not have an available implementation at hand and there wasn’t time to implement it ourselves.

We were able to find interesting hierarchies in the trees constructed by PERCH-BHC from the 20 Newsgroups dataset. Figure 5 shows hierarchies that relate the topics of "Windows" and "graphics", which are similar as graphics-related discussion is usually in the context of the Windows operating system. In the same part of the discovered hierarchy, there is also mention of "hardware," which is intuitive, as GPU hardware is the basis of computer graphics. Figure 6 shows hierarchies that
relate the sports hockey and baseball. There are a few outliers in the hierarchy, namely politics and space. However, none of the algorithms were able to get very good dendrogram purity with the Newsgroups dataset, so this kind of inaccuracy makes sense.

6 Conclusions

In terms of overall results, we were able to show that using a PERCH-like inference method for the BHC model would result in clusterings that can be produced much faster than with BUBHC, and also clusterings that are much more pure (or correct) than with existing fast randomized methods such as RBHC and EMBHC. The fact that we only approximate the BHC model with the lower bound likelihood term makes us that much more confident in the BHC model.

There remains a lot of work for the future in terms of improving this new inference method for BHC,
which we did not have the time to explore. We could place a non-informative prior on the MLE estimate in order to introduce higher variance when observing a small number of data points. The conjugate prior method could also be explored more, to see if the structure of the cluster tree can be used to obtain good estimates on parameters. This would make our method truly hierarchical. In addition, we could also explore a probabilistic method for routing data points rather than using the A* search algorithm, which would involve computing the likelihood of a new data point with all existing leaf node points and adding the new point as a sibling of the leaf node it is most likely with. This method would avoid any need to route the data points down the tree. Finally, we could definitely run our new method on more data sets and also data sets of different types. For example, the BHC method might work much better on categorical data or data with rich hierarchies.
Figure 3: 20 Newsgroups DP vs. Time

Figure 4: Separated Data DP vs. Time
Figure 5: 20 Newsgroups Hierarchy 1

Figure 6: 20 Newsgroups Hierarchy 2
References


