Randomized Algorithms for Fast Bayesian Hierarchical Clustering

Katherine A. Heller and Zoubin Ghahramani
Gatsby Computational Neuroscience Unit
University College London, London, WC1N 3AR, UK
{heller,zoubin}@gatsby.ucl.ac.uk

October 18, 2005

Abstract

We present two new algorithms for fast Bayesian Hierarchical Clustering on large data sets. Bayesian Hierarchical Clustering (BHC) [1] is a method for agglomerative hierarchical clustering based on evaluating marginal likelihoods of a probabilistic model. BHC has several advantages over traditional distance-based agglomerative clustering algorithms. It defines a probabilistic model of the data and uses Bayesian hypothesis testing to decide which merges are advantageous and to output the recommended depth of the tree. Moreover, the algorithm can be interpreted as a novel fast bottom-up approximate inference method for a Dirichlet process (i.e. countably infinite) mixture model (DPM). While the original BHC algorithm has $O(n^2)$ computational complexity, the two new randomized algorithms are $O(n \log n)$ and $O(n)$.

1 Introduction

Hierarchical structures are ubiquitous in the natural world. For example, the evolutionary tree of living organisms (and consequently features of these organisms such as the sequences of homologous genes) is a natural hierarchy. Hierarchical structures are also a natural representation for data which was not generated by evolutionary processes. For example, internet newsgroups, emails, or documents from a newswire, can be organized in increasingly broad topic domains. Hierarchical clustering is one of the most frequently used methods in unsupervised learning. Given a set of data points, the output is a binary tree (dendrogram) whose leaves are the data points and whose internal nodes represent nested clusters of various sizes. The tree organizes these clusters hierarchically, where the hope is that this hierarchy agrees with the intuitive organization of real-world data.

The traditional method for hierarchically clustering data as given in [2] is a bottom-up agglomerative algorithm. It starts with each data point assigned to its own cluster and iteratively merges the two closest clusters together until all the data belongs to a single cluster. The nearest pair of clusters is chosen based on a given distance measure (e.g. Euclidean distance between cluster means, or distance between nearest points).

There are several limitations to the traditional hierarchical clustering algorithm. The algorithm provides no guide to choosing the "correct" number of clusters or the level at which to prune the tree. It is often difficult to know which distance metric to choose, especially for structured data such as images or sequences. The traditional algorithm does not define a probabilistic model of the data, so it is hard to ask how "good" a clustering is, to compare to other models, to make predictions and cluster new data into an existing hierarchy. We use statistical inference to overcome these limitations. Previous work which uses probabilistic methods to perform hierarchical clustering is discussed in detail in our technical report [1].

Our Bayesian hierarchical clustering algorithm uses marginal likelihoods to decide which clusters to merge and to avoid overfitting. Basically it asks what the probability is that all the data in a potential merge were generated from the same mixture component, and compares this to exponentially many hypotheses at lower levels of the tree (section 2). The BHC algorithm has $O(n^2)$ complexity, where n is the number of data points. In practice, we would like run the Bayesian hierarchical clustering algorithm on very large data sets.

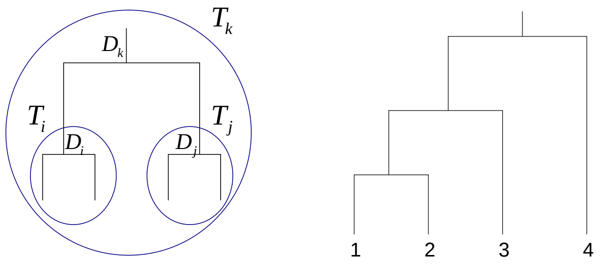

Figure 1: (a) Schematic of a portion of a tree where T_i and T_j are merged into T_k, and the associated data sets $\mathcal{D}_i$ and $\mathcal{D}_j$ are merged into $\mathcal{D}_k$. (b) An example tree with 4 data points. The clusterings (1 2 3)(4) and (1 2)(3)(4) are tree-consistent partitions of this data. The clustering (1)(2 3)(4) is not a tree-consistent partition.

Two fast versions of BHC using randomized algorithms, with $O(n \log n)$ and $O(n)$ complexity, are presented in section 3.

2 The BHC Algorithm

Our Bayesian hierarchical clustering algorithm is similar to traditional agglomerative clustering in that it is a one-pass, bottom-up method which initializes each data point in its own cluster and iteratively merges pairs of clusters. As we will see, the main difference is that our algorithm uses a statistical hypothesis test to choose which clusters to merge.

Let $\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ denote the entire data set, and $\mathcal{D}_i \subset \mathcal{D}$ the set of data points at the leaves of the subtree T_i. The algorithm is initialized with n trivial trees, $\{T_i : i = 1 \ldots n\}$ each containing a single data point $\mathcal{D}_i = \{\mathbf{x}^{(i)}\}$. At each stage the algorithm considers merging all pairs of existing trees. For example, if T_i and T_j are merged into some new tree T_k then the associated set of data is $\mathcal{D}_k = \mathcal{D}_i \cup \mathcal{D}_j$ (see figure 1(a)).

In considering each merge, two hypotheses are compared. The first hypothesis, which we will denote $\mathcal{H}_1^k$ is that all the data in $\mathcal{D}_k$ were in fact generated independently and identically from the *same probabilistic model*, $p(\mathbf{x}|\theta)$ with unknown parameters θ. Let us imagine that this probabilistic model is a multivariate Gaussian, with parameters $\theta = (\mu, \Sigma)$, although it is crucial to emphasize that for different types of data, different probabilistic models may be appropriate. To evaluate the probability of the data under this hypothesis we need to specify some prior over the parameters of the model, $p(\theta|\beta)$ with hyperparameters β. We now have the ingredients to compute the probability of the data $\mathcal{D}_k$ under $\mathcal{H}_1^k$:

$$p(\mathcal{D}_k|\mathcal{H}_1^k) = \int p(\mathcal{D}_k|\theta)p(\theta|\beta)d\theta = \int \Big[\prod_{\mathbf{x}^{(i)} \in \mathcal{D}_k} p(\mathbf{x}^{(i)}|\theta) \Big] p(\theta|\beta)d\theta \tag{1}$$

This calculates the probability that all the data in $\mathcal{D}_k$ were generated from the same parameter values assuming a model of the form $p(\mathbf{x}|\theta)$. This is a natural model-based criterion for measuring how well the data fit into one cluster. If we choose models with conjugate priors (e.g. Normal-Inverse-Wishart priors for Normal continuous data or Dirichlet priors for Multinomial discrete data) this integral is tractable. Throughout this paper we use such conjugate priors so the integrals are simple functions of sufficient statistics of $\mathcal{D}_k$. For example, in the case of Gaussians, (1) is a function of the sample mean and covariance of the data in $\mathcal{D}_k$.

The alternative hypothesis to $\mathcal{H}_1^k$ would be that the data in $\mathcal{D}_k$ has two or more clusters in it. Summing over the exponentially many possible ways of dividing $\mathcal{D}_k$ into two or more clusters is intractable. However, if we restrict ourselves to clusterings that partition the data in a manner that is consistent with the subtrees T_i and T_j, we can compute the sum efficiently using recursion. (We elaborate on the notion of *tree-consistent partitions* in figure 1(b)). The probability of the data under this restricted alternative hypothesis, $\mathcal{H}_2^k$, is simply a product over the subtrees $p(\mathcal{D}_k|\mathcal{H}_2^k) = p(\mathcal{D}_i|T_i)p(\mathcal{D}_j|T_j)$ where the probability of a data set under a tree (e.g. $p(\mathcal{D}_i|T_i)$) is defined below.

Combining the probability of the data under hypotheses $\mathcal{H}_1^k$ and $\mathcal{H}_2^k$, weighted by the prior that all points in $\mathcal{D}_k$ belong to one cluster, $\pi_k \overset{\text{def}}{=} p(\mathcal{H}_1^k)$, we obtain the marginal probability of the data in tree T_k:

$$p(\mathcal{D}_k|T_k) = \pi_k p(\mathcal{D}_k|\mathcal{H}_1^k) + (1-\pi_k)p(\mathcal{D}_i|T_i)p(\mathcal{D}_j|T_j) \tag{2}$$

This equation is defined recursively, there the first term considers the hypothesis that there is a single cluster in $\mathcal{D}_k$ and the second term efficiently sums over all other other clusterings of the data in $\mathcal{D}_k$ which are consistent with the tree structure (see figure 1(a)). In our tech report [1] we show that equation 2 can be used to derive an approximation to the marginal likelihood of a Dirichlet Process mixture model, and in fact provides a new lower bound on this marginal likelihood. We also show that the prior for the merged hypothesis, π_k, can be computed bottom-up in a DPM. The posterior probability of the merged hypothesis $r_k \overset{\text{def}}{=} p(\mathcal{H}_1^k|\mathcal{D}_k)$ is obtained using Bayes rule:

$$r_k = \frac{\pi_k p(\mathcal{D}_k|\mathcal{H}_1^k)}{\pi_k p(\mathcal{D}_k|\mathcal{H}_1^k) + (1-\pi_k)p(\mathcal{D}_i|T_i)p(\mathcal{D}_j|T_j)} \tag{3}$$

This quantity is used to decide greedily which two trees to merge, and is also used to determine which merges in the final hierarchy structure were justified. The general algorithm is very simple (see figure 2).

```
**input:** data $\mathcal{D} = \{\mathbf{x}^{(1)} \dots \mathbf{x}^{(n)}\}$, model $p(\mathbf{x}|\theta)$, prior $p(\theta|\beta)$
**initialize:** number of clusters $c = n$, and $\mathcal{D}_i = \{\mathbf{x}^{(i)}\}$ for $i = 1 \dots n$
**while** $c > 1$ **do**
  Find the pair $\mathcal{D}_i$ and $\mathcal{D}_j$ with the highest probability of the merged hypothesis, $r_k$ (eq. 3)
  Merge $\mathcal{D}_k \leftarrow \mathcal{D}_i \cup \mathcal{D}_j$,   $T_k \leftarrow (T_i, T_j)$
  Delete $D_i$ and $D_j$, $c \leftarrow c - 1$
**end while**
**output:** Bayesian mixture model where each tree node is a mixture component
The tree can be cut at points where $r_k < 0.5$
```

Figure 2: Bayesian Hierarchical Clustering (BHC) Algorithm

Our Bayesian hierarchical clustering algorithm has many desirable properties which are absent in traditional hierarchical clustering. For example, it allows us to define predictive distributions for new data points, it decides which merges are advantageous and suggests natural places to cut the tree using a statistical model comparison criterion (via r_k), and it can be customized to different kinds of data by choosing appropriate models for the mixture components.

For any tree, the probability of a new test point given the data can be computed by recursing through the tree starting at the root node. Each node k represents a cluster, with an associated predictive distribution $p(\mathbf{x}|\mathcal{D}_k) = \int p(\mathbf{x}|\theta)p(\theta|\mathcal{D}_k, \beta)d\theta$. The overall predictive distribution sums over all nodes weighted by their posterior probabilities:

$$p(\mathbf{x}|\mathcal{D}) = \sum_{k \in \mathcal{N}} \omega_k \, p(\mathbf{x}|\mathcal{D}_k) \tag{4}$$

where $\mathcal{N}$ is the set of all nodes in the tree, $\omega_k \overset{\text{def}}{=} r_k \prod_{i \in \mathcal{N}_k}(1 - r_i)$ is the weight on cluster k, and $\mathcal{N}_k$ is the set of nodes on the path from the root node to the parent of node k.

For Gaussian components, for example, with conjugate priors, this results in a predictive distribution which is a mixture of multivariate t distributions. We give a more detailed explanation and show some examples of this in our tech report [1].

In summary, $p(\mathcal{D}|T)$ sums the probabilities for all tree-consistent partitions, weighted by the prior mass assigned to each partition by the DPM. The computational complexity of constructing the tree is $O(n^2)$, the complexity of computing the marginal likelihood is $O(n)$, and the complexity of computing the predictive distribution is $O(n)$.

3 Randomized BHC Algorithms

Our goal is to run Bayesian Hierarchical Clustering on very large datasets, but to accomplish this we need a BHC algorithm which has very small computational complexity. The BHC algorithm as given in section 2 is $O(n^2)$, and computation is dominated by pairwise comparisons of data points at the lowest levels. This seems wasteful and limits its use for large data sets. We aim to capitalize on this inefficiency, combined with the powerful resource of randomized algorithms [3], to create a faster BHC algorithm. We propose the following randomized algorithm for fast Bayesian Hierarchical Clustering (RBHC):

```
input: data $\mathcal{D} = \{\mathbf{x}^{(1)} \ldots \mathbf{x}^{(n)}\}$
  pick $m \ll n$ points randomly from $\mathcal{D}$, so $\mathcal{D}^{[m]} \subset \mathcal{D}$
  run $\mathbf{BHC}\big(\mathcal{D}^{[m]}\big)$ obtaining a tree $T$
  $\mathbf{Filter}\big(\mathcal{D}\backslash\mathcal{D}^{[m]}\big)$ through the top level of tree $T$, obtaining $\mathcal{D}_L$ and $\mathcal{D}_R$ where $\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_R$
  recurse: run $\mathbf{RBHC}(\mathcal{D}_L)$ and $\mathbf{RBHC}(\mathcal{D}_R)$
output: Bayesian mixture model where each tree node is a mixture component
```

Figure 3: Randomized Bayesian Hierarchical Clustering (RBHC) Algorithm

The algorithm takes in a data set $\mathcal{D}$ and randomly selects a subset $\mathcal{D}^{[n]}$ of m data points from the data set. The original BHC algorithm is run on that subset of m data points, obtaining a tree T. The remaining $(n-m)$ data points $(\mathcal{D}\backslash\mathcal{D}^{[m]})$ are then filtered through the top level (last merge) of tree T. The filter algorithm (figure 4) takes in the top level partitioning of tree T ($\mathcal{D}_L^{[m]}$ and $\mathcal{D}_R^{[m]}$), along with the priors (π_L and π_R) computed in the BHC algorithm, and all remaining data points. It then takes each remaining data point (x_i) and computes the probabilities that x_i belongs to the left subtree and right subtree (in cluster $\mathcal{D}_L^{[m]}$ or $\mathcal{D}_R^{[m]}$). The data point is then added to the highest probability cluster (subtree). The assignments of all n data points to the left and right subtrees are returned to the RBHC algorithm, which then runs itself separately on each subtree. The constant m may be reduced as the data set becomes smaller.

```
input: $\mathcal{D}_L^{[m]}, \mathcal{D}_R^{[m]}, \pi_L, \pi_R, \mathcal{D}\backslash\mathcal{D}^{[m]}$
initialize: $\mathcal{D}_L = \mathcal{D}_L^{[m]}, \mathcal{D}_R = \mathcal{D}_R^{[m]}$
  foreach $\mathbf{x}^{(i)} \in \mathcal{D}\backslash\mathcal{D}^{[m]}$
    compute $p(\mathbf{x}^{(i)}|\mathcal{D}_L^{[m]})$ and $p(\mathbf{x}^{(i)}|\mathcal{D}_R^{[m]})$
    if $\pi_L p(\mathbf{x}^{(i)}|\mathcal{D}_L^{[m]}) > \pi_R p(\mathbf{x}^{(i)}|\mathcal{D}_R^{[m]})$
      then $\mathcal{D}_L \leftarrow \mathcal{D}_L \cup \{\mathbf{x}^{(i)}\}$
    else $\mathcal{D}_R \leftarrow \mathcal{D}_R \cup \{\mathbf{x}^{(i)}\}$
output: $\mathcal{D}_L, \mathcal{D}_R$
```

Figure 4: Filter Algorithm

The RBHC algorithm rests on two assumptions. The first assumption is that the top level clustering, built from a subset of m data points ($\mathcal{D}^{[m]}$), will be a good approximation to the top level clustering ($\mathcal{D}$). This means that the assignments of data points into $\mathcal{D}_L$ and $\mathcal{D}_R$ will be similar for the subsample and filter based RBHC as compared to running the full BHC algorithm. The second assumption is that the BHC algorithm tends to produce roughly balanced trees with $(\alpha n, (1-\alpha)n)$ points in each of the top level clusters (i.e. $O(n)$ points per branch rather than $O(1)$ points). This is necessary for RBHC to maintain its smaller running time.

Proposition 1 *The RBHC algorithm is* $O(n \log n)$

The number of operations required to run RBHC can be expressed recursively as:

$$Ops(\text{RBHC}(n)) = m^2 + n + Ops(\text{RBHC}(\alpha n)) + Ops(\text{RBHC}((1-\alpha)n))$$

Here the m^2 term comes from running BHC on m data points and the n term comes from the Filter algorithm. Expanding this expression out to L levels of recursion and letting $\alpha = \frac{1}{2}$ we get:

$$Ops(\text{RBHC}(n)) = m^2 + n + 2(m^2 + \frac{n}{2}) + 4(m^2 + \frac{n}{4}) + ... + 2^L(m^2 + \frac{n}{2^L})$$

This gives us $\log n$ terms of $O(n)$. We can generalize so that α takes on other values, and this yields the same result, merely adjusting the base of the log. In practice, when a level where m is comparable to $n/2^L$ is reached the algorithm can simply call BHC on the $n/2^L$ points. Since we are generally interested in the top few levels of the tree, it may make sense to truncate the algorithm after the first several, say eight or so, levels, and avoid running down to the levels where individual points are in their own clusters. This truncated algorithm is $O(nL)$.

We also propose the following alternative randomized algorithm based on EM. This algorithm takes advantage of the fact that BHC can be run on clusters of points rather than individual points (i.e. it can cluster clusters):

```
input: data $\mathcal{D} = \{\mathbf{x}^{(1)} \dots \mathbf{x}^{(n)}\}$
  subsample $m$ points randomly from $\mathcal{D}$, so $\mathcal{D}^{[m]} \subset \mathcal{D}$
  foreach point $\mathbf{x}^{(i)} \in \mathcal{D}^{[m]}$ create a cluster $\mathcal{D}_i^{[m]} = \{\mathbf{x}^{(i)}\}$
  Filter$(\mathcal{D} \backslash \mathcal{D}^{[m]})$ into these $m$ clusters
  refine the clusters by running $k$ steps of hard EM:
    for each point in $\mathcal{D}$ reassign to most probable cluster
  run BHC on the $m$ clusters output by EM
output: Bayesian mixture model where each tree node is a mixture component
```

Figure 5: A randomized algorithm using EM (EMBHC)

Proposition 2 *The EMBHC algorithm is $O(n)$*

The number of operations for this alternate algorithm is $nm - m^2 + knm + m^2 = O(knm)$, where the $nm - m^2$ term comes from the filtering step, which filters $n - m$ points into m clusters, the knm term from running EM, and m^2 from the BHC step. So for small k and m this algorithm is linear in n.

4 Results

We have compared Bayesian Hierarchical Clustering to traditional hierarchical clustering methods. Extensive results are available in our tech report, showing significant improvements in tree structure, clustering, and classification ability [1]. We reproduce two small examples comparing BHC and average linkage hierarchical clustering (ALHC) below.

Using the CEDAR Buffalo digits data set (8x8 images of digits, 64 attributes), we ran the algorithms on 20 examples each of 3 digits (0,2,4) (figure 6). If we compare the structure of the trees generated by average linkage hierarchical clustering versus Bayesian hierarchical clustering, we can see that BHC tree structure is much more desirable, particularly at the highest levels of the tree, where it immediately finds 3 distinct clusters (one for each of the three digits), unlike ALHC.

We also compared the BHC and ALHC algorithms on 200 examples each of 4 newsgroups (rec.sport.baseball, rec.sport.hockey, rec.autos, and sci.space) from the CMU 20newsgroups dataset. Our dataset was constucted using Rainbow [4], where a stop list was used and words appearing fewer than 5 times were ignored. The dataset was then binarized based on word presence/absence in a document.

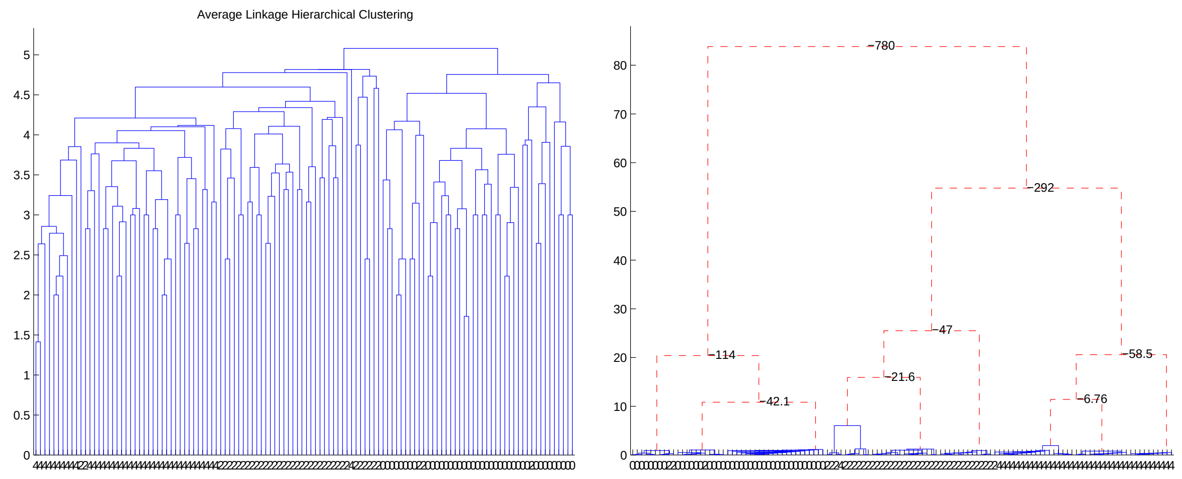

Figure 6: (a) The tree given by performing average linkage hierarchical clustering on 120 examples of 3 digits (0,2, and 4). Numbers on the x-axis correspond to the true class label of each example, while the y-axis corresponds to distance between merged clusters. (b) The tree given by performing Bayesian hierarchical clustering on the same dataset. Here higher values on the y-axis also correspond to higher levels of the hierarchy (later merges), though the exact numbers are irrelevant. Red dashed lines correspond to merges with negative posterior log probabilities and are labeled with these values.

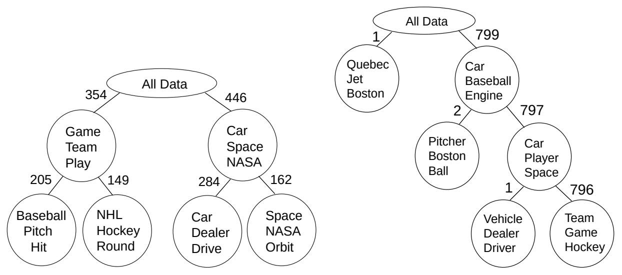

Figure 7: Top level structure, of BHC (left) vs. Average Linkage HC, for the newsgroup dataset. The 3 words shown at each node have the highest mutual information between the cluster of documents at that node versus its sibling, and occur with higher frequency in that cluster. The number of documents at each cluster is also given.

Figure 7 compares the top three levels (last three merges) of the newsgroups hierarchy (using all 800 examples and the 50 word attributes with highest information gain) from BHC and ALHC. Continuing to look at lower levels does not improve ALHC. Other traditional hierarchical clustering algorithms perform similarly to ALHC. Full dendrograms for this dataset, plus many more results are availabe in our tech report [1].

We are currently working on implementions of RBHC and EMBHC so as to apply the algorithm to much larger datasets.

References

[1] K. A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. Technical Report 2005-002, Gatsby Computational Neuroscience Unit, 2005.

[2] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* Wiley, 1973.

[3] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[4] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.