Bayesian Hierarchical Clustering

Katherine A. Heller HELLER@GATSBY.UCL.AC.UK
Zoubin Ghahramani ZOUBIN@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, University College London
17 Queen Square, London, WC1N 3AR, UK

Abstract

We present a novel algorithm for agglomerative hierarchical clustering based on evaluating marginal likelihoods of a probabilistic model. This algorithm has several advantages over traditional distance-based agglomerative clustering algorithms. (1) It defines a probabilistic model of the data which can be used to compute the predictive distribution of a test point and the probability of it belonging to any of the existing clusters in the tree. (2) It uses a model-based criterion to decide on merging clusters rather than an ad-hoc distance metric. (3) Bayesian hypothesis testing is used to decide which merges are advantageous and to output the recommended depth of the tree. (4) 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). It provides a new lower bound on the marginal likelihood of a DPM by summing over exponentially many clusterings of the data in polynomial time. We describe procedures for learning the model hyperparameters, computing the predictive distribution, and extensions to the algorithm. Experimental results on synthetic and real-world data sets demonstrate useful properties of the algorithm.

1. Introduction

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. 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.

The traditional method for hierarchically clustering data as given in (Duda & Hart, 1973) 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 section 6.

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 generative model for our algorithm is a Dirichlet process mixture model (i.e. a countably infinite mixture model), and the algorithm can be viewed as a fast bottom-up agglomerative way of performing approximate inference in a DPM. Instead of giving weight to all possible partitions of the data into clusters, which is intractable and would require the use of sampling methods, the algorithm efficiently computes the weight of exponentially many partitions which are consistent with the tree structure (section 3).

2. 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 $\mathcal{H}_1^k$:

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

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 $\mathcal{D}_k$.

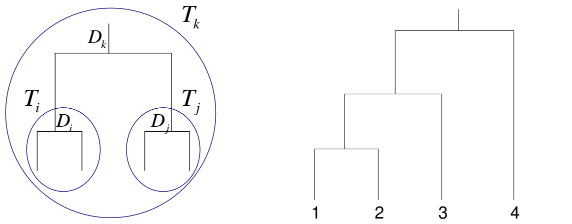


Figure 1. (a) Schematic of a portion of a tree where T_i and T_j are merged into T_k, and the associated data sets $\mathcal{D}_i$ and $\mathcal{D}_j$ are merged into $\mathcal{D}_k$. (b) An example tree with 4 data points. The clusterings (1 2 3)(4) and (1 2)(3)(4) are tree-consistent partitions of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

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 section 3 and 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, $\pi_k \stackrel{\text{def}}{=} p(\mathcal{H}_1^k)$, we obtain the marginal probability of the data in tree T_k:

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

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 section 3 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.[1] 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 $r_k \stackrel{\text{def}}{=} p(\mathcal{H}_1^k|\mathcal{D}_k)$ is obtained using Bayes rule:

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

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 $\mathcal{D} = \{\mathbf{x}^{(1)} \ldots \mathbf{x}^{(n)}\}$, model $p(\mathbf{x}|\theta)$, prior $p(\theta|\beta)$
initialize: number of clusters $c = n$, and $\mathcal{D}_i = \{\mathbf{x}^{(i)}\}$ for $i = 1 \ldots n$
while $c > 1$ **do**
  Find the pair $\mathcal{D}_i$ and $\mathcal{D}_j$ with the highest probability of the merged hypothesis:

$$r_k = \frac{\pi_k p(\mathcal{D}_k|\mathcal{H}_1^k)}{p(\mathcal{D}_k|T_k)}$$

  Merge $\mathcal{D}_k \leftarrow \mathcal{D}_i \cup \mathcal{D}_j$, $\quad T_k \leftarrow (T_i, T_j)$
  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 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.

3. Approximate Inference in a Dirichlet Process Mixture Model

The above algorithm is an approximate inference method for Dirichlet Process mixture models (DPM). Dirichlet Process mixture models consider the limit of infinitely many components of a finite mixture model. Allowing infinitely many components makes it possible to more realistically model the kinds of complicated distributions which we expect in real problems. We briefly review DPMs here, starting from finite mixture models.

Consider a finite mixture model with C components

$$p(\mathbf{x}^{(i)}|\phi) = \sum_{j=1}^{C} p(\mathbf{x}^{(i)}|\theta_j)p(c_i = j|\mathbf{p}) \quad (4)$$

where $c_i \in \{1, \ldots, C\}$ is a cluster indicator variable for data point i, $\mathbf{p}$ are the parameters of a multinomial distribution with $p(c_i = j|\mathbf{p}) = \mathrm{p}_j$, θ_j are the parameters of the jth component, and $\phi = (\theta_1, \ldots, \theta_C, \mathbf{p})$. Let the parameters of each component have conjugate priors $p(\theta|\beta)$ as in section 2, and the multinomial parameters also have a conjugate Dirichlet prior

$$p(\mathbf{p}|\alpha) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/C)^C} \prod_{j=1}^{C} \mathrm{p}_j^{\alpha/C-1}. \quad (5)$$

Given a data set $\mathcal{D} = \{\mathbf{x}^{(1)} \ldots, \mathbf{x}^{(n)}\}$, the marginal likelihood for this mixture model is

$$p(\mathcal{D}|\alpha,\beta) = \int \left[\prod_{i=1}^{n} p(\mathbf{x}^{(i)}|\phi)\right] p(\phi|\alpha,\beta)d\phi \quad (6)$$

where $p(\phi|\alpha,\beta) = p(\mathbf{p}|\alpha)\prod_{j=1}^{C} p(\theta_j|\beta)$. This marginal likelihood can be re-written as

$$p(\mathcal{D}|\alpha,\beta) = \sum_{\mathbf{c}} p(\mathbf{c}|\alpha)p(\mathcal{D}|\mathbf{c},\beta) \quad (7)$$

where $\mathbf{c} = (c_1, \ldots, c_n)$ and $p(\mathbf{c}|\alpha) = \int p(\mathbf{c}|\mathbf{p})p(\mathbf{p}|\alpha)d\mathbf{p}$ is a standard Dirichlet integral. The quantity (7) is well-defined even in the limit $C \to \infty$. Although the number of possible settings of $\mathbf{c}$ grows as C^n and therefore diverges as $C \to \infty$, the number of possible ways of partitioning the n points remains finite (roughly $\mathcal{O}(n^n)$). Using $\mathcal{V}$ to denote the set of all possible partitioning of n data points, we can re-write (7) as:

$$p(\mathcal{D}|\alpha,\beta) = \sum_{v\in\mathcal{V}} p(v|\alpha)p(\mathcal{D}|v,\beta) \quad (8)$$

Rasmussen (2000) provides a thorough analysis of DPMs with Gaussian components, and a Markov chain Monte Carlo (MCMC) algorithm for sampling from the partitionings v, and we have included a small amount of review in appendix A. DPMs have the interesting property that the probability of a new data point belonging to a cluster is propotional to the number of points already in that cluster (Blackwell & MacQueen, 1973), where α controls the probability of the new point creating a new cluster.

[1] It is important not to confuse the marginal likelihood in equation 1, which integrates over the parameters of one cluster, and the marginal likelihood of a DPM, which integrates over all clusterings and their parameters.

For an n point data set, each possible clustering is a different partition of the data, which we can denote by placing brackets around data point indices: e.g. (1 2)(3)(4). Each individual cluster, e.g. (1 2), is a nonempty subset of data, yielding $2^n - 1$ possible clusters, which can be combined in many ways to form clusterings (i.e. partitions) of the whole data set. We can organize a subset of these clusters into a tree. Combining these clusters one can obtain all *tree-consistent partitions* of the data (see figure 1(b)). Rather than summing over all possible partitions of the data using MCMC, our algorithm computes the sum over all exponentially many tree-consistent partitions for a particular tree built greedily bottom-up. This can be seen as a fast and deterministic alternative to MCMC approximations.

Returning to our algorithm, since a DPM with concentration hyperparameter α defines a prior on all partitions of the n_k data points n $\mathcal{D}_k$ (the value of α is directly related to the expected number of clusters), the prior on the merged hypothesis is the relative mass of all n_k points belonging to one cluster versus all the other partitions of those n_k data points consistent with the tree structure. This can be computed bottom-up as the tree is being built (figure 3).

initialize each leaf i to have $d_i = \alpha$, $\pi_i = 1$
for each internal node k **do**
  $d_k = \alpha\Gamma(n_k) + d_{\text{left}_k}\, d_{\text{right}_k}$
  $\pi_k = \frac{\alpha\Gamma(n_k)}{d_k}$
end for

Figure 3. Algorithm for computing prior on merging, where right_k (left_k) indexes the right (left) subtree of T_k and d_{right_k}(d_{left_k}) is the value of d computed for the right (left) child of internal node k.

Lemma 1 *The marginal likelihood of a DPM is:*

$$p(\mathcal{D}_k) = \sum_{v \in \mathcal{V}} \frac{\alpha^{m_v} \prod_{\ell=1}^{m_v} \Gamma(n_\ell^v)}{\left[\frac{\Gamma(n_k+\alpha)}{\Gamma(\alpha)}\right]} \prod_{\ell=1}^{m_v} p(\mathcal{D}_\ell^v)$$

where $\mathcal{V}$ is the set of all possible partitionings of $\mathcal{D}_k$, m_v is the number of clusters in partitioning v, and n_ℓ^v is the number of points in cluster ℓ of partitioning v.

This is easily shown since:

$$p(\mathcal{D}_k) = \sum_v p(v) p(\mathcal{D}^v)$$

where:

$$p(v) = \frac{\alpha^{m_v} \prod_{l=1}^{m_v} \Gamma(n_l^v)}{\left[\frac{\Gamma(n_k+\alpha)}{\Gamma(\alpha)}\right]}$$

and:

$$p(\mathcal{D}^v) = \prod_{l=1}^{m_v} p(\mathcal{D}_l^v)$$

Here $p(\mathcal{D}_k)$ is a sum over all partitionings, v, where the first (fractional) term in the sum is the prior on partitioning v and the second (product) term is the likelihood of partioning v under the data.

Theorem 1 *The quantity (2) computed by the Bayesian Hierarchical Clustering algorithm is:*

$$p(\mathcal{D}_k|T_k) = \sum_{v \in \mathcal{V}_T} \frac{\alpha^{m_v} \prod_{\ell=1}^{m_v} \Gamma(n_\ell^v)}{d_k} \prod_{\ell=1}^{m_v} p(\mathcal{D}_\ell^v)$$

where $\mathcal{V}_T$ is the set of all tree-consistent partitionings of $\mathcal{D}_k$.

Proof Rewriting equation (2) using algorithm 3 to substitute in for π_k we obtain:

$$p(\mathcal{D}_k|T_k) = p(\mathcal{D}_k|\mathcal{H}_1^k)\frac{\alpha\Gamma(n_k)}{d_k} + p(\mathcal{D}_i|T_i)p(\mathcal{D}_j|T_j)\frac{d_i d_j}{d_k}$$

We will proceed to give a proof by induction. In the base case, at a leaf node, the second term in this equation drops out since there are no subtrees. $\Gamma(n_k = 1) = 1$ and $d_k = \alpha$ yielding $p(\mathcal{D}_k|T_k) = p(\mathcal{D}_k|\mathcal{H}_1^k)$ as we should expect at a leaf node.

For the inductive step, we note that the first term is always just the trivial partition with all n_k points into a single cluster. According to our inductive hypothesis:

$$p(\mathcal{D}_i|T_i) = \sum_{v' \in \mathcal{V}_{T_i}} \frac{\alpha^{m_{v'}} \prod_{\ell'=1}^{m_{v'}} \Gamma(n_{\ell'}^{v'})}{d_i} \prod_{\ell'=1}^{m_{v'}} p(\mathcal{D}_{\ell'}^{v'})$$

and similarly for $p(\mathcal{D}_j|T_j)$, where $\mathcal{V}_{T_i}$ ($\mathcal{V}_{T_j}$) is the set of all tree-consistent partitionings of $\mathcal{D}_i$ ($\mathcal{D}_j$). Combining terms we obtain:

$$\begin{aligned}
p(\mathcal{D}_i|T_i)&p(\mathcal{D}_j|T_j)\frac{d_i d_j}{d_k} \\
&= \frac{1}{d_k}\left(\sum_{v' \in \mathcal{V}_{T_i}} \alpha^{m_{v'}} \prod_{\ell'=1}^{m_{v'}} \Gamma(n_{\ell'}^{v'}) \prod_{\ell'=1}^{m_{v'}} p(\mathcal{D}_{\ell'}^{v'})\right) \\
&\quad \times \left(\sum_{v'' \in \mathcal{V}_{T_j}} \alpha^{m_{v''}} \prod_{\ell''=1}^{m_{v''}} \Gamma(n_{\ell''}^{v''}) \prod_{\ell''=1}^{m_{v''}} p(\mathcal{D}_{\ell''}^{v''})\right) \\
&= \sum_{v \in \mathcal{V}_{\text{NTT}}} \frac{\alpha^{m_v} \prod_{\ell=1}^{m_v} \Gamma(n_\ell^v)}{d_k} \prod_{\ell=1}^{m_v} p(\mathcal{D}_\ell^v)
\end{aligned}$$

where $\mathcal{V}_{\text{NTT}}$ is the set of all non-trivial tree-consistent partionings of $\mathcal{D}_k$. For the trivial partition, $m_v = 1$

and $n_1^v = n_k$. By combining the trivial and non-trivial terms we get a sum over *all* tree-consistent partitions yielding the result in Theorem 1. This completes the proof. Another way to get this result is to expand out $p(\mathcal{D}|T)$ and substitute for π using algorithm 3.

Corollary 1 *For any binary tree T_k with the data points in D_k at its leaves, the following is a lower bound on the marginal likelihood of a DPM:*

$$\frac{d_k \Gamma(\alpha)}{\Gamma(n_k + \alpha)} p(\mathcal{D}_k|T_k) \leq p(\mathcal{D}_k)$$

Proof Proof of Corollary 1 follows trivially by multiplying $p(\mathcal{D}_k|T_k)$ by a ratio of its denominator and the denominator from $p(\mathcal{D}_k)$ from lemma 1 (i.e. $\frac{d_k\Gamma(\alpha)}{\Gamma(n_k+\alpha)}$), and from the fact that tree-consistent partitions are a subset of all partitions of the data.

Proposition 1 *The number of tree-consistent partitions is exponential in the number of data points for balanced binary trees.*

Proof If T_i has C_i tree-consistent partitions of $\mathcal{D}_i$ and T_j has C_j tree-consistent partitions of $\mathcal{D}_j$, then $T_k = (T_i, T_j)$ merging the two has $C_i C_j + 1$ tree-consistent partitions of $\mathcal{D}_k = \mathcal{D}_i \cup \mathcal{D}_j$, obtained by combining all partitions and adding the partition where all data in $\mathcal{D}_k$ are in one cluster. At the leaves $C_i = 1$. Therefore, for a balanced binary tree of depth ℓ the number of tree-consistent partitions grows as $\mathcal{O}(2^{2^\ell})$ whereas the number of data points n grows as $\mathcal{O}(2^\ell)$

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 $O(n \log n)$, and the complexity of computing the predictive distribution (see section 4) is $O(n)$.

4. Learning and Prediction

Learning Hyperparameters. For any given setting of the hyperparameters, the root node of the tree approximates the probability of the data given those particular hyperparameters. In our model the hyperparamters are the concentration parameter α from the DPM, and the hyperparameters β of the probabilistic model defining each component of the mixture. We can use the root node marginal likelihood $p(\mathcal{D}|T)$ to do model comparison between different settings of the hyperparameters. For a fixed tree we can optimize over the hyperparameters by taking gradients. In the case of all data belonging to a single cluster:

$$p(\mathcal{D}|\mathcal{H}_1) = \int p(\mathcal{D}|\theta) p(\theta|\beta) d\theta$$

we can compute $\frac{\partial p(\mathcal{D}|\mathcal{H}_1)}{\partial \beta}$. Using this we can compute gradients for the component model hyperparamters bottom-up as the tree is being built:

$$\begin{aligned} \frac{\partial p(\mathcal{D}_k|T_k)}{\partial \beta} &= \pi_k \frac{\partial p(\mathcal{D}|\mathcal{H}_1)}{\partial \beta} \\ &+ (1-\pi_k) \frac{\partial p(\mathcal{D}_i|T_i)}{\partial \beta} p(\mathcal{D}_j|T_j) \\ &+ (1-\pi_k) p(\mathcal{D}_i|T_i) \frac{\partial p(\mathcal{D}_j|T_j)}{\partial \beta} \end{aligned}$$

Similarly, we can compute

$$\begin{aligned} \frac{\partial p(\mathcal{D}_k|T_k)}{\partial \alpha} &= \frac{\partial \pi_k}{\partial \alpha} p(\mathcal{D}_k|\mathcal{H}_1) \\ &- \frac{\partial \pi_k}{\partial \alpha} p(\mathcal{D}_i|T_i) p(\mathcal{D}_j|T_j) \\ &+ (1-\pi_k) \frac{\partial p(\mathcal{D}_i|T_i)}{\partial \alpha} p(\mathcal{D}_j|T_j) \\ &+ (1-\pi_k) p(\mathcal{D}_i|T_i) \frac{\partial p(\mathcal{D}_j|T_j)}{\partial \alpha} \end{aligned}$$

where:

$$\frac{\partial \pi_k}{\partial \alpha} = \frac{\pi_k}{\alpha} - \frac{\pi_k}{d_k} \left(\frac{\partial d_k}{\partial \alpha} \right)$$

and:

$$\frac{\partial d_k}{\partial \alpha} = \Gamma(n_k) + \left(\frac{\partial d_{left_k}}{\partial \alpha} \right) d_{right_k} + d_{left_k} \left(\frac{\partial d_{right_k}}{\partial \alpha} \right)$$

These gradients can be computed bottom-up by additionally propogating $\frac{\partial d}{\partial \alpha}$. This allows us to construct an EM-like algorithm where we find the best tree structure in the (Viterbi-like) E step and then optimize over the hyperparameters in the M step. In our experiments we have only optimized one of the hyperparameters with a simple line search for Gaussian components. A simple empirical approach is to set the hyperparameters β by fitting a single model to the whole data set.

Predictive Distribution. 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:

$$p(\mathbf{x}|\mathcal{D}) = \sum_{k \in \mathcal{N}} \omega_k \, p(\mathbf{x}|\mathcal{D}_k) \tag{9}$$

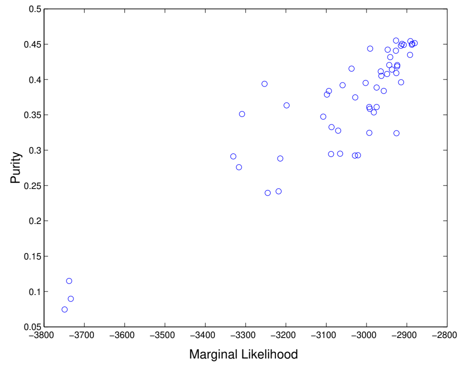


Figure 4. Log marginal likelihood (evidence) vs. purity over 50 iterations of hyperparameter optimization

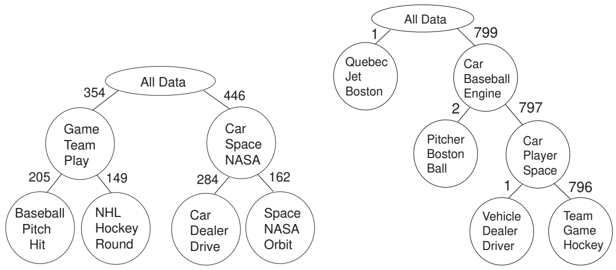


Figure 5. Top level structure, of BHC (left) vs. Average Linkage HC, for the newsgroup dataset. The 3 words shown at each node have the highest mutual information between the cluster of documents at that node versus its sibling, and occur with higher frequency in that cluster. The number of documents at each cluster is also given.

where $\mathcal{N}$ is the set of all nodes in the tree, $\omega_k \stackrel{\text{def}}{=} r_k \prod_{i \in \mathcal{N}_k}(1 - r_i)$ 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. This expression can be derived by rearranging the sum over all tree-consistent partitionings into a sum over all clusters in the tree (noting that a cluster can appear in many partitionings).

For Gaussian components with conjugate priors, this results in a predictive distribution which is a mixture of multivariate t distributions. We show some examples of this in the Results section.

5. Results

We compared Bayesian hierarchical clustering to traditional hierarchical clustering using average, single, and complete linkage over 5 datasets (4 real and 1 synthetic). We also compared our algorithm to average linkage hierarchical clustering on several toy 2D problems (figure 8). On these problems we were able to compare the different hierarchies generated by the two algorithms, and visualize clusterings and predictive distributions.

The 4 real datasets we used are the spambase (100 random examples from each class, 2 classes, 57 attributes) and glass (214 examples, 7 classes, 9 attributes) datasets from the UCI repository the CEDAR Buffalo digits (20 random examples from each class, 10 classes, 64 attributes), and the CMU 20Newsgroups dataset (120 examples, 4 classes - rec.sport.baseball, rec.sport.hockey, rec.autos, and sci.space, 500 attributes). We also used synthetic data generated from a mixture of Gaussians (200 examples, 4 classes, 2 attributes). The synthetic, glass and toy datasets were modeled using Gaussians, while the digits, spambase, and newsgroup datasets were binarized and modeled using Bernoullis. We binarized the digits dataset by thresholding at a greyscale value of 128 out of 0 through 255, and the spambase dataset by whether each attribute value was zero or non-zero. We ran the algorithms on 3 digits (0,2,4), and all 10 digits. The newsgroup dataset was constucted using Rainbow (McCallum, 1996), 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. For these classification datasets, where labels for the data points are known, we computed a measure between 0 and 1 of how well a dendrogram clusters the known labels called the *dendrogram purity*.[2] We found that the marginal likelihood of the tree structure for the data was highly correlated with the purity. Over 50 iterations with different hyperparameters this correlation was 0.888 (see figure 4). Table 1 shows the results on these datasets.

On all datasets except Glass BHC found the highest purity trees. For Glass, the Gaussian assumption may have been poor. This highlights the importance of model choice. Similarly, for classical distance-based hierarchical clustering methods, a poor choice of distance metric may result in poor clusterings.

Another advantage of the BHC algorithm, not fully addressed by purity scores alone, is that it tends to create hierarchies with good structure, particularly at high levels. Figure 5 compares the top three levels (last three merges) of the newsgroups hierarchy (using

[2]Let T be a tree with leaves $1, \ldots, n$ and $c_1, \ldots, c_n$ be the known discrete class labels for the data points at the leaves. Pick a leaf ℓ uniformly at random; pick another leaf j uniformly in the same class, i.e. $c_\ell = c_j$. Find the smallest subtree containing ℓ and j. Measure the fraction of leaves in that subtree which are in the same class (c_ℓ). The expected value of this fraction is the dendrogram purity, and can be computed exactly in a bottom up recursion on the dendrogram. The purity is 1 iff all leaves in each class are contained in some pure subtree.

Data Set	Single Linkage	Complete Linkage	Average Linkage	BHC
Synthetic	0.599 ± 0.033	0.634 ± 0.024	0.668 ± 0.040	$\mathbf{0.828 \pm 0.025}$
Newsgroups	0.275 ± 0.001	0.315 ± 0.008	0.282 ± 0.002	$\mathbf{0.465 \pm 0.016}$
Spambase	0.598 ± 0.017	0.699 ± 0.017	0.668 ± 0.019	$\mathbf{0.728 \pm 0.029}$
3Digits	0.545 ± 0.015	0.654 ± 0.013	0.742 ± 0.018	$\mathbf{0.807 \pm 0.022}$
10Digits	0.224 ± 0.004	0.299 ± 0.006	0.342 ± 0.005	$\mathbf{0.393 \pm 0.015}$
Glass	0.478 ± 0.009	0.476 ± 0.009	$\mathbf{0.491 \pm 0.009}$	0.467 ± 0.011

Table 1. Purity scores for 3 kinds of traditional agglomerative clustering, and Bayesian hierarchical clustering.

800 examples and the 50 words with highest information gain) from BHC and ALHC. Continuing to look at lower levels does not improve ALHC, as can be seen from the full dendrograms for this dataset, figure 7. Although sports and autos are both under the newsgroup heading rec, our algorithm finds more similarity between autos and the space document class. This is quite reasonable since these two classes have many more common words (such as engine, speed, cost, develop, gas etc.) than there are between autos and sports. Figure 6 also shows dendrograms of the 3digits dataset.

6. Related Work

The work in this paper is related to and inspired by several previous probabilistic approaches to clustering[3], which we *briefly* review here. Stolcke and Omohundro (1993) described an algorithm for agglomerative model merging based on marginal likelihoods in the context of hidden Markov model structure induction . Williams (2000) and Neal (2003) describe Gaussian and diffusion-based hierarchical generative models, respectively, for which inference can be done using MCMC methods. Similarly, Kemp et al. (2004) present a hierarchical generative model for data based on a mutation process. Ward uses a hierarchical fusion of contexts based on marginal likelihoods in a Dirichlet language model.

Banfield and Raftery (1993) present an approximate method based on the likelihood ratio test statistic to compute the marginal likelihood for c and $c-1$ clusters and use this in an agglomerative algorithm. Vaithyanathan and Dom (2000) perform hierarchical clustering of multinomial data consisting of a vector of features. The clusters are specified in terms of which subset of features have common distributions. Their clusters can have different parameters for some features and the same parameters to model other features and their method is based on finding merges that maximize marginal likelihood under a Dirichlet-Multinomial model.

Iwayama and Tokunaga define a Bayesian hierarchical clustering algorithm which attempts to agglomeratively find the maximum posterior probability clustering but it makes strong independence assumptions and does not use the marginal likelihood.

Segal et al. (2002) present probabilistic abstraction hierarchies (PAH) which learn a hierarchical model in which each node contains a probabilistic model and the hierarchy favors placing similar models at neighboring nodes in the tree (as measured by a distance function between probabilstic models). The training data is assigned to leaves of this tree. Ramoni et al. (2002) present an agglomerative algorithm for merging time series based on greedily maximizing marginal likelihood. Friedman (2003) has also recently proposed a greedy agglomerative algorithm based on marginal likelihood which simultaneously clusters rows and columns of gene expression data.

The algorithm in our paper is different from the above algorithms in several ways. First, unlike (Williams, 2000; Neal, 2003; Kemp et al., 2004) it is not in fact a hierarchical generative model of the data, but rather a hierarchical way of organizing nested clusters. Second our algorithm is derived from Dirichlet process mixtures. Third the hypothesis test at the core of our algorithm tests between a single merged hypothesis and the alternative is exponentially many other clusterings of the same data (not one vs two clusters at each stage). Lastly, our algorithm does not use any iterative method, like EM, or require sampling, like MCMC, and is therefore significantly faster than most of the above algorithms.

7. Discussion

We have presented a novel algorithm for Bayesian hierarchical clustering based on Dirichlet process mixtures. This algorithm has several advantages over traditional approaches, which we have highlighted throughout the paper. We have presented prediction and hyperparameter optimization procedures and shown that the al-

[3] There has also been a considerable amount of decision tree based work on Bayesian tree structures for classification and regression (Chipman et al., 1998; Denison et al., 2002), but this is not closely related to work presented here.

gorithm provides competitive clusterings of real-world data as measured by purity with respect to known labels. The algorithm can also be seen as an extremely fast alternative to MCMC inference in DPMs.

The limitations of our algorithm include its inherent greediness, a computational complexity which is quadratic in the number of data points, and the lack of any incorporation of tree uncertainty.

In future work, we plan to try BHC on more complex component models for other realistic data—this is likely to require approximations of the component marginal likelihoods (1). We also plan to extend BHC to systematically incorporate hyperparameter optimization and improve the running time to $O(n \log n)$ by exploiting a randomized version of the algorithm. We need to compare this novel, fast inference algorithm for DPMs to other inference algorithms such as MCMC (Rasmussen, 2000), EP (Minka & Ghahramani, 2003) and Variational Bayes (Blei & Jordan, 2004). We also hope to explore the idea of computing several alternative tree structures in order to create a manipulable tradeoff between computation time and tightness of our lower bound. There are many exciting avenues for further work in this area.

A. Review of Dirichlet Process Mixtures

This section follows Rasmussen (2000). For each of n data points being modelled by a finite mixture, we can associate cluster indicator variables, $c_1 ... c_n$, which can take on values $c_i \in \{1...C\}$, where C is the number of mixture components. The joint distribution of the indicator variables is multinomial with parameter vector $\mathbf{p} = (\mathrm{p}_1 ... \mathrm{p}_C)$:

$$p(c_1 ... c_n | \mathbf{p}) = \prod_{j=1}^{C} \mathrm{p}_j^{n_j}, \qquad n_j = \sum_{i=1}^{n} \delta(c_i, j)$$

Here n_j is the number of data points assigned to class j. If we place a symmetric Dirichlet prior on $\mathbf{p}$ with concentration parameters $\frac{\alpha}{C}$:

$$p(\mathrm{p}_1 ... \mathrm{p}_C | \alpha) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/C)^C} \prod_{j=1}^{C} \mathrm{p}_j^{\alpha/C - 1}$$

we can integrate $\mathbf{p}$ out, obtaining:

$$\begin{aligned} p(c_1 ... c_n | \alpha) &= \int p(c_1 ... c_n | \mathrm{p}_1 ... \mathrm{p}_C) p(\mathrm{p}_1 ... \mathrm{p}_C) d\mathrm{p}_1 ... \mathrm{p}_C \\ &= \frac{\Gamma(\alpha)}{\Gamma(n + \alpha)} \prod_{j=1}^{C} \frac{\Gamma(n_j + \alpha/C)}{\Gamma(\alpha/C)} \end{aligned}$$

From this we can compute the conditional probability of a single indicator:

$$p(c_i = j | \mathbf{c}_{-i}, \alpha) = \frac{n_{-i,j} + \alpha/C}{n - 1 + \alpha}$$

where $n_{-i,j}$ is the number of data points, excluding point i, assigned to class j. Finally, taking the infinite limit ($C \to \infty$), we obtain:

$$\begin{aligned} p(c_i = j | \mathbf{c}_{-i}, \alpha) &= \frac{n_{-i,j}}{n - 1 + \alpha}, \\ p(c_i \neq c_{i'}, \forall i' \neq i | \mathbf{c}_{-i}, \alpha) &= \frac{\alpha}{n - 1 + \alpha} \end{aligned}$$

The second term is the prior probability that point i belongs to some new class that no other points belong to.

B. Marginal Likelihoods for Single Components

Here we give equations for the marginal likelihoods of single components using simple distributions and their conjugate priors. For each component model we compute:

$$p(\mathcal{D} | \mathcal{H}_1) = \int p(\mathcal{D} | \theta) p(\theta | \beta) d\theta$$

Bernoulli-Beta

$$p(\mathcal{D} | \mathcal{H}_1) = \prod_{d=1}^{k} \frac{\Gamma(\alpha_d + \beta_d) \Gamma(\alpha_d + m_d) \Gamma(\beta_d + N - m_d)}{\Gamma(\alpha_d) \Gamma(\beta_d) \Gamma(\alpha_d + \beta_d + N)}$$

where k is the total number of dimensions, N is the number of data points in the dataset, $m_d = \sum_{i=1}^{N} x_d^{(i)}$, $\boldsymbol{X}$ is the dataset, and $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are hyperparmeters for the Beta distribution.

Multinomial-Dirichlet

$$\begin{aligned} p(\mathcal{D} | \mathcal{H}_1) &= \prod_{i=1}^{N} \binom{M_i}{x_1^{(i)} \ldots x_k^{(i)}} \\ &\times \frac{\Gamma(\sum_{d=1}^{k} \alpha_d) \prod_{d=1}^{k} \Gamma(\alpha_d + m_d)}{\Gamma(M + \sum_{d=1}^{k} \alpha_d) \prod_{d=1}^{k} \Gamma(\alpha_d)} \end{aligned}$$

where k is the total number of dimensions, N is the number of data points in the dataset, $M = \sum_{i=1}^{N} M_i$, $M_i = \sum_{j=1}^{k} x_j^{(i)}$, $m_d = \sum_{i=1}^{N} x_d^{(i)}$, $\boldsymbol{X}$ is the dataset, and $\boldsymbol{\alpha}$ is a hyperparameter for the Dirichlet prior.

Normal-Inverse Wishart-Normal

$$
\begin{aligned}
p(\mathcal{D}|\mathcal{H}_1) &= (2\pi)^{-\frac{Nk}{2}} \left(\frac{r}{N+r}\right)^{k/2} |\mathbf{S}|^{\frac{v}{2}} |\mathbf{S}'|^{-\frac{v'}{2}} \\
&\times \frac{(2)^{\frac{v'k}{2}} \prod_{d=1}^{k} \Gamma\left(\frac{v'+1-d}{2}\right)}{(2)^{\frac{vk}{2}} \prod_{d=1}^{k} \Gamma\left(\frac{v+1-d}{2}\right)}
\end{aligned}
$$

where:

$$
\begin{aligned}
\mathbf{S}' &= \mathbf{S} + \mathbf{X}\mathbf{X}^T + \frac{rN}{N+r}\mathbf{m}\mathbf{m}^T \\
&+ \frac{1}{N+r}\left(\sum_{i=1}^{N} \mathbf{x}^{(\mathbf{i})}\right)\left(\sum_{i=1}^{N} (\mathbf{x}^{(\mathbf{i})})^T\right) \\
&- \frac{r}{N+r}\left(\mathbf{m}\left(\sum_{i=1}^{N} (\mathbf{x}^{(\mathbf{i})})^T\right) + \left(\sum_{i=1}^{N} \mathbf{x}^{(\mathbf{i})}\right)\mathbf{m}^T\right)
\end{aligned}
$$

and:

$$
v' = v + N
$$

Here $\boldsymbol{X}$ is the dataset, k is the number of dimensions, and N is the number of datapoints in $\boldsymbol{X}$. The hyperparameters for the Inverse Wishart-Normal prior include: $\boldsymbol{m}$ which is the prior on the mean, $\boldsymbol{S}$ is the prior on the precision matrix, r is scaling factor on the prior precision of the mean, and v is the degree of freedom.

References

Banfield, J. D., & Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics*, *49*, 803–821.

Blackwell, & MacQueen (1973). Ferguson distributions via Polya urn schemes. *Ann. Stats.*

Blei, D., & Jordan, M. (2004). *Variational methods for dirichlet process mixture models* (Technical Report 674). UC Berkeley.

Chipman, H., George, E., & McCulloch, R. (1998). Bayesian cart model search (with discussion). *Journal of the American Statistical Association*, *93*, 935–960.

Denison, D., Holmes, C., Mallick, B., & Smith, A. (2002). *Bayesian methods for nonlinear classification and regression.* Wiley.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* Wiley.

Friedman, N. (2003). *Pcluster: Probabilistic agglomerative clustering of gene expression profiles* (Technical Report 2003-80). Herbew University.

Iwayama, M., & Tokunaga, T. (1995). Hierarchical Bayesian clustering for automatic text classification. *Proceedings of IJCAI-95, 14th International Joint Conference on Artificial Intelligence* (pp. 1322–1327). Montreal, CA: Morgan Kaufmann Publishers, San Francisco, US.

Kemp, C., Griffiths, T. L., Stromsten, S., & Tenenbaum, J. B. (2004). Semi-supervised learning with trees. *NIPS 16.*

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

Minka, T., & Ghahramani, Z. (2003). Expectation propagation for infinite mixtures. *NIPS Workshop on Nonparametric Bayesian Methods and Infinite Models.*

Neal, R. M. (2003). Density modeling and clustering using dirichlet diffusion trees. *Bayesian Statistics 7* (pp. 619–629).

Ramoni, M. F., Sebastiani, P., & Kohane, I. S. (2002). Cluster analysis of gene expression dynamics. *Proc Nat Acad Sci*, *99*, 9121–9126.

Rasmussen, C. E. (2000). The infinite Gaussian mixture model. *NIPS 12* (pp. 554–560).

Segal, E., Koller, D., & Ormoneit, D. (2002). Probabilistic abstraction hierarchies. *NIPS 14.*

Stolcke, A., & Omohundro, S. (1993). Hidden Markov Model induction by bayesian model merging. *NIPS 5* (pp. 11–18).

Vaithyanathan, S., & Dom, B. (2000). Model-based hierarchical clustering. *UAI.*

Ward, D. J. (2001). *Adaptive computer interfaces.* Doctoral dissertation, University of Cambridge.

Williams, C. (2000). A MCMC approach to hierarchical mixture modelling. *NIPS 12.*

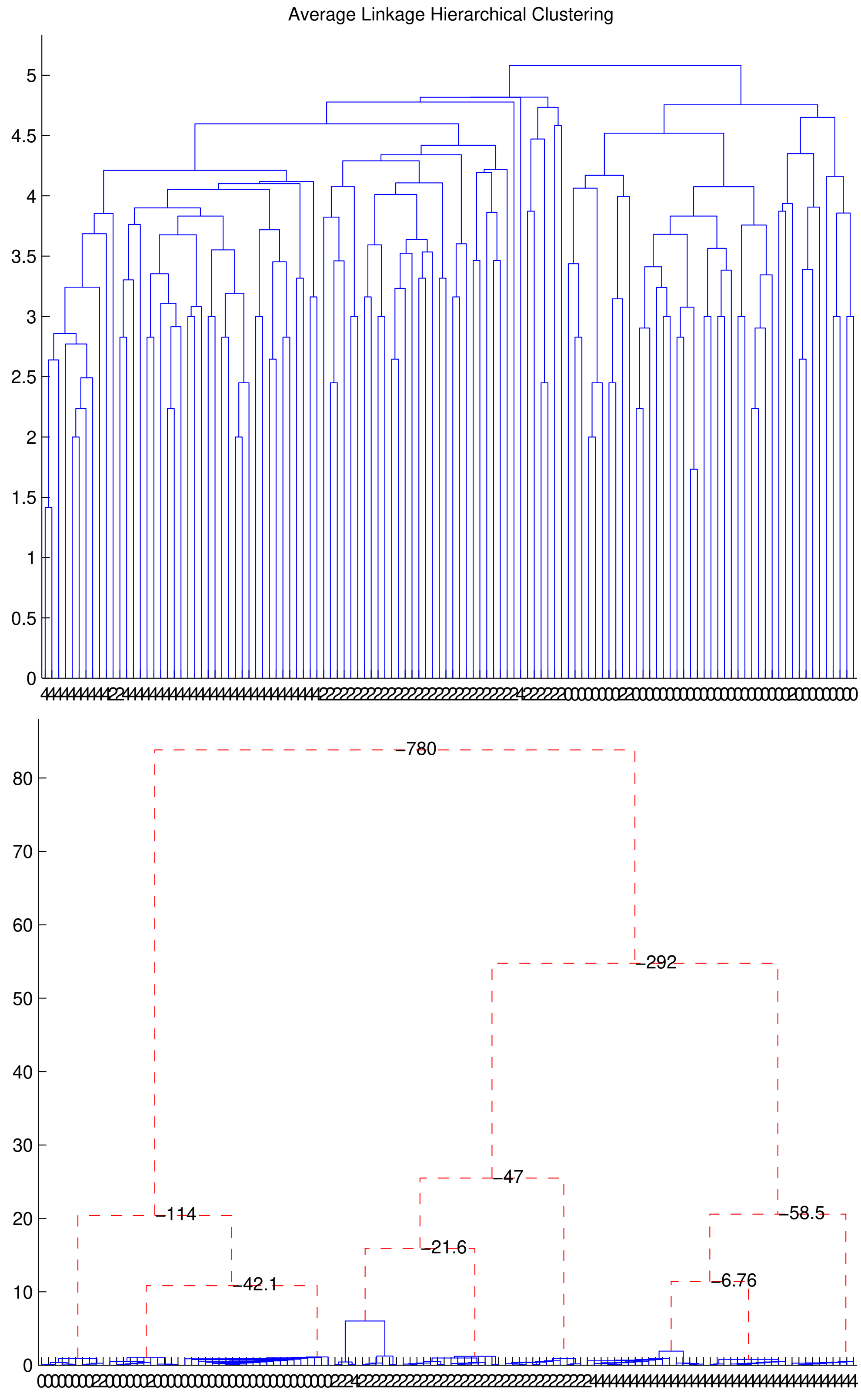
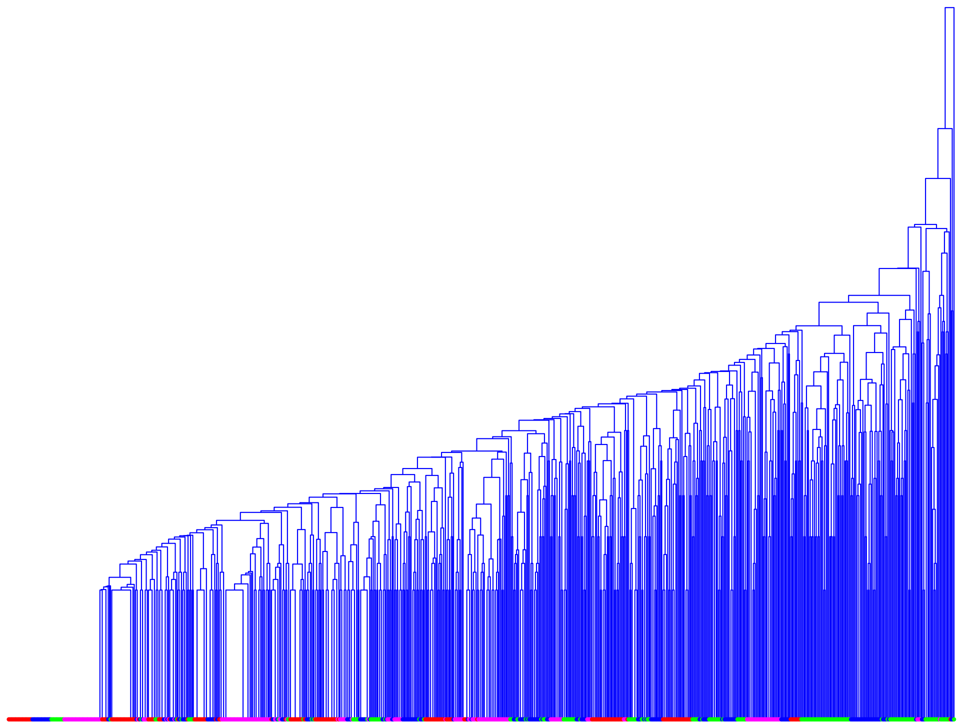


Figure 6. (a) The tree given by performing average linkage hierarchical clustering on 120 examples of 3 digits (0,2, and 4), yielding a purity score of 0.870. Numbers on the x-axis correspond to the true class label of each example, while the y-axis corresponds to distance between merged clusters. (b) The tree given by performing Bayesian hierarchical clustering on the same dataset, yielding a purity score of 0.924. 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.



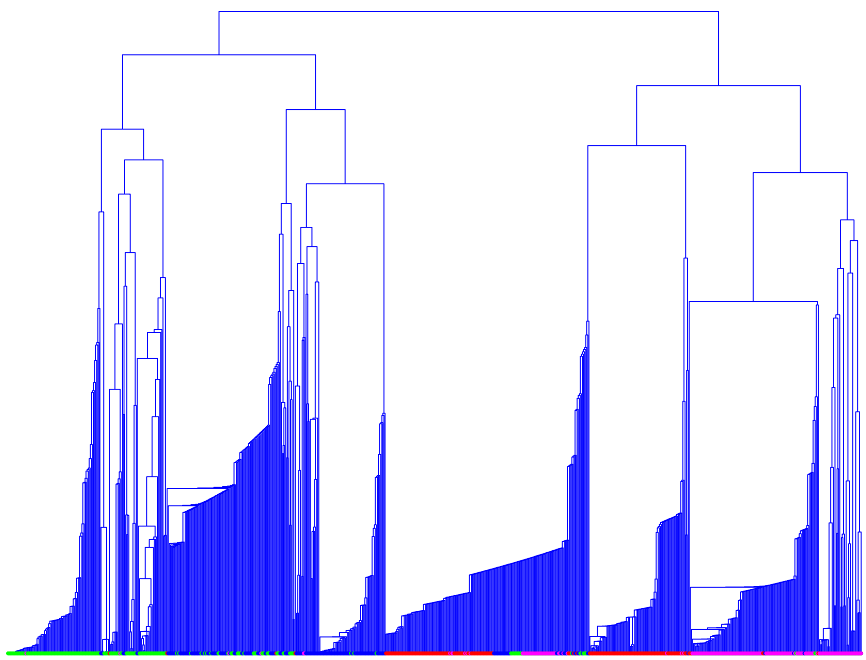


Figure 7. Full dendrograms for average linkage hierarchical clustering and BHC on 800 newsgroup documents. Each of the leaves in the document is labeled with a color, according to which newsgroup that document came from. Red is rec.autos, blue is rec.sport.baseball, green is rec.sport.hockey, and magenta is sci.space.

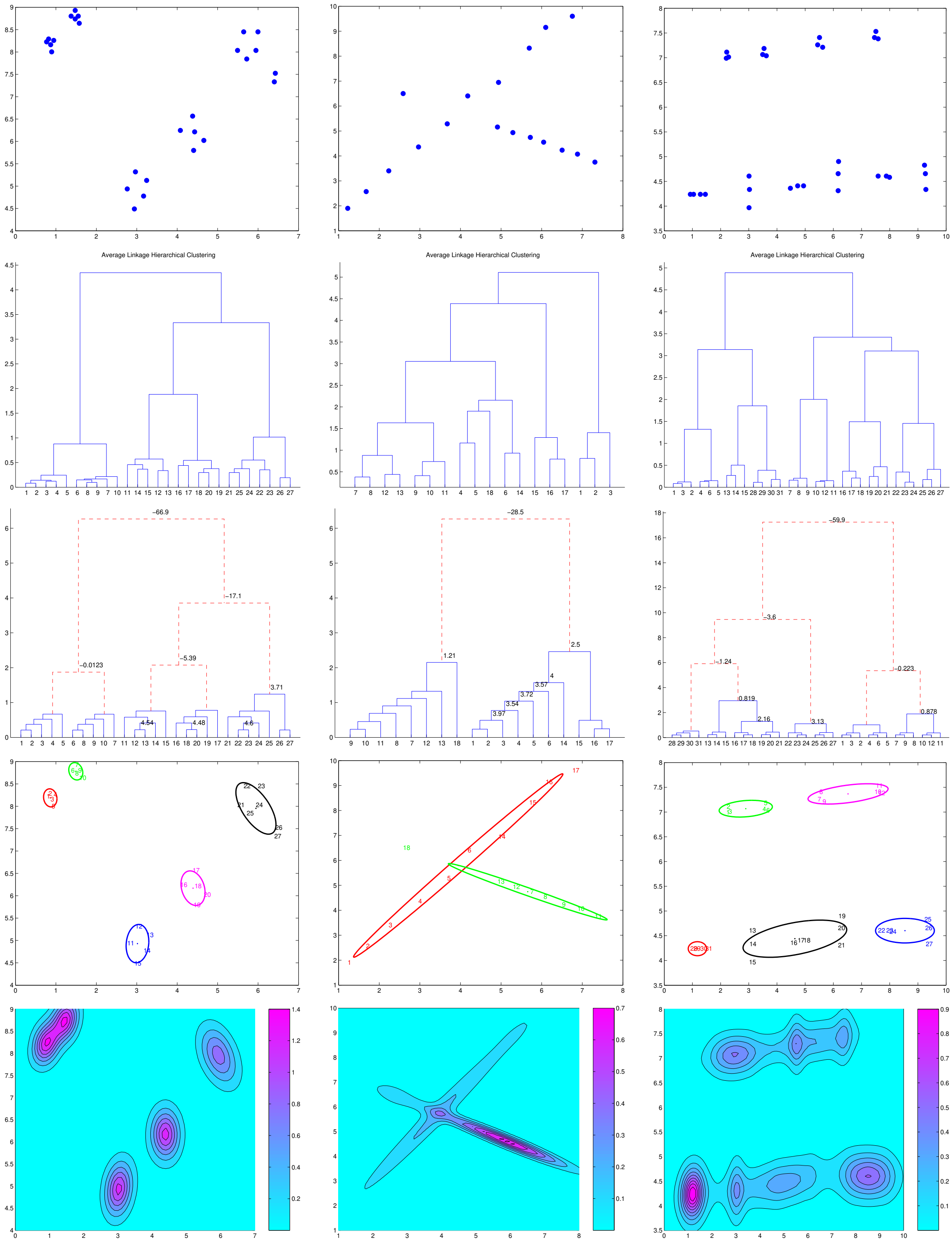


Figure 8. Three toy examples (one per column). The first row shows the original data sets. The second row gives the dendrograms resulting from average linkage hierarchical clustering, each number on the x-axis corresponds to a data point as displayed on the plots in the fourth row. The third row gives the dendrograms resulting from Bayesian Hierarchical clustering where the red dashed lines are merges our algorithm prefers not to make given our priors. The numbers on the branches are the log odds for merging ($\log \frac{r_k}{1-r_k}$). The fourth row shows the clusterings found by our algorithm using Gaussian components, when the tree is cut at red dashed branches ($r_k < 0.5$). The last row shows the predictive densities resulting from our algorithm. The BHC algorithm behaves structurally more sensibly than the traditional algorithm. Note that point 18 in the middle column is clustered into the green class even though it is substantially closer to points in the red class. In the first column the traditional algorithm prefers to merge cluster 1-6 with cluster 7-10 rather than, as BHC does, cluster 21-25 with cluster 26-27. Finally, in the last column, the BHC algorithm clusters all points along the upper and lower horizontal parallels together, whereas the traditional algorithm merges some subsets of points veritcally(for example cluster 7-12 with cluster 16-27).