Efficient Bayesian Metho
for Clustering

Katherine Ann Heller

B.S., Computer Science, Applied Mathematics and Statistics,
State University of New York at Stony Brook, USA (2000)

M.S., Computer Science, Columbia University, USA (2003)

Gatsby Computational Neuroscience Unit
University College London
17 Queen Square
London, United Kingdom

THESIS

Submitted for the degree of
Doctor of Philosophy, University of London

2007

I, Katherine Ann Heller, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

One of the most important goals of unsupervised learning is to discover meaningful clusters in data. Clustering algorithms strive to discover groups, or clusters, of data points which belong together because they are in some way similar.

The research presented in this thesis focuses on using Bayesian statistical techniques to cluster data. We take a model-based Bayesian approach to defining a cluster, and evaluate cluster membership in this paradigm. Due to the fact that large data sets are increasingly common in practice, our aim is for the methods in this thesis to be efficient while still retaining the desirable properties which result from a Bayesian paradigm.

We develop a Bayesian Hierarchical Clustering (BHC) algorithm which efficiently addresses many of the drawbacks of traditional hierarchical clustering algorithms. The goal of BHC is to construct a hierarchical representation of the data, incorporating both finer to coarser grained clusters, in such a way that we can also make predictions about new data points, compare different hierarchies in a principled manner, and automatically discover interesting levels of the hierarchy to examine. BHC can also be viewed as a fast way of performing approximate inference in a Dirichlet Process Mixture model (DPM), one of the cornerstones of nonparametric Bayesian Statistics.

We create a new framework for retrieving desired information from large data collections, Bayesian Sets, using Bayesian clustering techniques. Unlike current retrieval methods, Bayesian Sets provides a principled framework which leverages the rich and subtle information provided by queries in the form of a set of examples. Whereas most clustering algorithms are completely unsupervised, here the query provides supervised hints or constraints as to the membership of a particular cluster. We call this "clustering on demand", since it involves forming a cluster once some elements of that cluster have been revealed. We use Bayesian Sets to develop a content-based image retrieval system. We also extend Bayesian Sets to a discriminative setting and use this to perform automated analogical reasoning.

Lastly, we develop extensions of clustering in order to model data with more complex structure than that for which traditional clustering is intended. Clustering models traditionally assume that each data point belongs to one and only one cluster, and although they have proven to be a very powerful class of models, this basic assumption is somewhat limiting. For example, there may be overlapping regions where data points actually belong to multiple clusters, like movies which can each belong to multiple genres. We extend traditional mixture models to create a statistical model for overlapping clustering, the Infinite Overlapping Mixture Model (IOMM), in a nonparametric Bayesian setting, using the Indian Buffet Process (IBP). We also develop a Bayesian Partial Membership model (BPM), which allows data points to have *partial* membership in multiple clusters via a continuous relaxation of a finite mixture model.

Acknowledgments

I would first like to thank my PhD advisor, Zoubin Ghahramani, for all the support he has given me throughout my time as a student; I have been extremely fortunate. I am also very grateful to have been a PhD student at the Gatsby Computational Neuroscience Unit, which always provided me with a lively and intellectually stimulating environment in which to work. I would like to thank all the faculty, Peter Latham, David MacKay, Maneesh Sahani, Yee Whye Teh, and particularly director Peter Dayan.

I would like to thank my fantastic collaborators Ricardo Silva and Edo Airoldi, and our wonderful 4th year student Yang Xu, whose hard work have all gone into this thesis. Also, Alexei Yavlinsky for his advice and assistance regarding content-based image retrieval. I am very grateful to my examiners, David Blei and John Shawe-Taylor, for taking the time to give me such useful feedback.

I would like to thank all the people I have met and worked with during my time at the Gatsby Unit. It is not possible to thank everyone here but I would particularly like to mention Iain Murray, Ed Snelson, and the rest of the 501 crew for their scientific input, discussions, and friendship over the years. I would also like to thank Alex Boss for all of her administrative help, and Deb Ray for submitting my thesis.

I am grateful to everyone who has supported me financially. In particular, the Gatsby Charitable Foundation, UCL, the UK government, and the PASCAL network of excellence.

Lastly, I would like to thank my parents, Mary Noberini and Donald Heller, my sister, Elizabeth, and my friends who have supported me all these years, even from thousands of miles away.

Contents

List of figures

List of tables

List of algorithms

Chapter 1

Introduction

The research presented in this thesis focuses on using Bayesian statistical techniques for clustering, or partitioning, data. Abstractly, clustering is discovering groups of data points that belong together. As an example, if given the task of clustering animals, one might group them together by type (mammals, reptiles, amphibians), or alternatively by size (small or large). Typically, any particular data set does not have a uniquely correct clustering, and the desired clustering may depend on the particular application. Some examples of applications of clustering are: clustering related genes together from gene expression data to help elucidate gene functions (Eisen et al., 1998), clustering news stories by topic to automatically organize online news feeds (Zhang et al., 2004), and clustering images of celestial objects in order to identify different classes of quasars and dwarfs (Pelleg and Moore, 1999).

There has been a great deal of previous work on different methods for clustering data, including hierarchical clustering (Johnson, 1967; Duda and Hart, 1973), spectral clustering (Shi and Malik, 2000; Meila and Shi, 2001; Ng et al., 2002), k-means clustering (Hartigan and Wong, 1979; Duda and Hart, 1973), and mixture modeling (McLachlan and Peel, 2000). While these methods have been widely used in practice, many suffer from some serious limitations. For example, many of these methods cluster data based on pairwise distances, and do not provide a coherent way to predict the probability or cluster assignments of new data points. In this thesis, we work towards developing a new framework for clustering, which addresses many of the limitations of prior work by taking a Bayesian approach.

We also look at ways of extending the traditional domain of clustering methods to new situations, including using clustering methods in new application areas with new demands (e.g. information retrieval problems which need to utilize queries), or problems for which the original formulation of the task of clustering needs to be modified (e.g. overlapping clustering, which requires additional flexibility to allow data points to belong to *multiple* clusters).

Lastly, given the explosion of data available from the web (e.g. digital pictures), from scientific research (e.g. protein databases), and from companies (e.g. credit card transactions), developing effective methods which are also very efficient has become increasingly important. Most of the clustering methods presented in this thesis were developed with efficiency in mind, and with the aim of being accessible and useful to researchers and developers working on a diverse range of applications.

The rest of this thesis proceeds as follows:

- In chapter 2 we give a broad overview of Bayesian methods and the task of clustering. This includes a review of nonparametric Bayesian methods which are relevant to the work contained within this thesis.

- In chapter 3 we present the *Bayesian Hierarchical Clustering* (BHC) algorithm. BHC provides a novel algorithm for agglomerative hierarchical clustering based on evaluating marginal likelihoods of a probabilistic model. It presents several advantages over traditional distance-based agglomerative clustering algorithms, and can be interpreted as a novel fast bottom-up approximate inference method for a Dirichlet process (i.e. countably infinite) mixture model (DPM).

- In chapter 4 we develop a new framework for retrieving desired information from large data collections. Specifically, we consider the problem of retrieving items belonging to a concept or cluster, given a query consisting of a few known items from that cluster. We formulate this as a Bayesian inference problem and describe a very simple algorithm for solving it. We focus on sparse binary data and show that our algorithm can retrieve items using a single sparse matrix multiplication, making it possible to apply our algorithm to very large datasets.

- In chapter 5 we take a Bayesian approach to performing content-based image retrieval (CBIR), based on the framework laid out in chapter 4. Our CBIR system retrieves images, in real-time, from a large unlabeled database using simple color and texture features.

- In chapter 6 we extend the work presented in chapter 4 to create a framework for performing analogical reasoning with relational data. Analogical reasoning is the ability to generalize about relations between objects. Given examples of pairs of objects which share a particular relationship, the method presented in this chapter retrieves other pairs of objects which share the same relationship from a relational database.

- In chapter 7 we introduce a new nonparametric Bayesian method, the *Infinite Overlapping Mixture Model* (IOMM), for modeling overlapping clusters. The IOMM uses exponential family distributions to model each cluster and forms an overlapping mixture by taking products of these distributions. The IOMM allows an unbounded number of clusters, and assignments of points to (multiple) clusters is modeled using an Indian Buffet Process (IBP).

- In chapter 8 we present the *Bayesian Partial Membership Model* (BPM), for modeling partial memberships of data points to clusters. This means that, unlike with a standard finite mixture model, data points can have fractional membership in multiple clusters. The BPM is derived as a continuous relaxation of a finite mixture model, which allows Hybrid Monte Carlo to be used for inference and learning.

- Lastly, in chapter 9, we review the main contributions of this thesis, and discuss future work.

Chapter 2

Bayesian methods and clustering

In this chapter we review some concepts which are fundamental to this thesis and which will appear multiple times in the upcoming chapters. We discuss Bayesian methods in general, followed by a brief overview of clustering techniques and an introduction to nonparametric Bayesian methods.

2.1 Bayesian Methods

Bayesian methods provide a way to reason coherently about the world around us, in the face of uncertainty. Bayesian approaches are based on a mathematical handling of uncertainty, initially proposed by Bayes and Laplace in the 18th century and further developed by statisticians and philosophers in the 20th century. Recently, Bayesian methods have emerged as models of both human cognitive phenomena, in areas such as multisensory integration (Ernst and Banks, 2002), motor learning (Körding and Wolpert, 2004), visual illusions (Knill and Richards, 1996), and neural computation (Hinton et al., 1995; Knill and Pouget, 2004), and as the basis of machine learning systems.

Bayes rule states that:

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)} \tag{2.1}$$

and can be derived from basic probability theory. Here x might be a data point and θ some model parameters. $P(\theta)$ is the probability of θ and is refered to as the *prior*, it represents the prior probability of θ before observing any information about x. $P(x|\theta)$ is the probability of x conditioned on θ and is also refered to as the *likelihood.* $P(\theta|x)$ is the *posterior* probability of θ after observing x, and $P(x)$ is the *normalizing constant.*

Letting $P(x, \theta)$ be the joint probability of x and θ, we can *marginalize* out θ to get:

$$P(x) = \int P(x, \theta) d\theta. \tag{2.2}$$

Thus we can also refer to $P(x)$ as the marginal probability of x. In this thesis we often use Bayes rule to perform model comparisons. For some data set of N data points, $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$, and model m with model parameters θ:

$$P(m|\mathcal{D}) = \frac{P(\mathcal{D}|m)P(m)}{P(\mathcal{D})}. \tag{2.3}$$

We can compute this quantity for many different models m and select the one with the highest posterior probability as the best model for our data. Here:

$$P(\mathcal{D}|m) = \int P(\mathcal{D}|\theta, m)P(\theta|m) d\theta \tag{2.4}$$

is called the *marginal likelihood*, and is necessary to compute equation (2.3). Computing marginal likelihoods (in order to compare different clustering models) is fundamental to this thesis.

We can also try to predict the probability of new data points, x^*, which have not yet been observed :

$$P(x^*|\mathcal{D}, m) = \int P(x^*|\theta)P(\theta|\mathcal{D}, m) d\theta \tag{2.5}$$

where

$$P(\theta|\mathcal{D}, m) = \frac{P(\mathcal{D}|\theta, m)P(\theta|m)}{P(\mathcal{D}|m)}$$

is the posterior probability of model parameters θ conditioned on the data $\mathcal{D}$, and is simply computed using Bayes rule.

Bayesian probability theory can be used to represent degrees of belief in uncertain propositions. In fact, Cox (1946) and Jaynes (2003) show that if one tries to represent beliefs numerically and makes only a few basic assumptions, this numerical representation of beliefs results in the derivation of basic probability theory. There is also a game theoretic result, called the *Dutch Book Theorem*, which states that if one is willing to place bets in accordance with ones beliefs, then unless those beliefs are consistent with probability theory (including Bayes Rule), there is a dutch book of bets that one will be willing to accept that is guaranteed to lose money regardless of the outcome of the bets.

Bayesian methods inherently invoke Occam's razor. Consider two models m_1 and m_2, where m_2 contains m_1 as a special case (for example linear functions m_1 are special cases of higher order polynomials m_2). The marginal likelihood (equation (2.4)) for m_2 will be lower than for m_1 if the data is already being modeled well by m_1 (e.g. a linear function). There are some data sets however (e.g. nonlinear functions) which m_2 will model better than m_1. Therefore, Bayesian methods do not typically suffer from the overfitting problems commonly encountered with other methods.

2.1.1 Exponential Family Models

Much of the work in this thesis revolves around the computation of marginal likelihoods (equation (2.4)). Unfortunately marginal likelihoods are intractable to compute exactly for complicated probabilistic models ($P(\mathcal{D}|\theta, m)$) and priors ($P(\theta|m)$). Therefore we tend to focus on models from the exponential family, for which marginal likelihoods can be computed analytically due to the fact that they have conjugate priors.

An *exponential family distribution* can be written in the form:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \exp\{\boldsymbol{s}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta} + h(\mathbf{x}) + g(\boldsymbol{\theta})\} \tag{2.6}$$

where $\boldsymbol{s}(\mathbf{x})$ is a vector depending on the data known as the ***sufficient statistics***, $\boldsymbol{\theta}$ is a vector of ***natural parameters***, $h(\mathbf{x})$ is a function of the data, and $g(\boldsymbol{\theta})$ is a function of the parameters which ensures that the probability normalizes to one when integrating or summing over $\mathbf{x}$.

A probability distribution $p(\boldsymbol{\theta})$ is said to be ***conjugate*** to the exponential family distribution $p(\mathbf{x}|\boldsymbol{\theta})$ if $p(\boldsymbol{\theta}|\mathbf{x})$ has the same functional form as $p(\boldsymbol{\theta})$. In particular, the conjugate prior to the above exponential family distribution can be written in the form:

$$p(\boldsymbol{\theta}) \propto \exp\{\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\theta} + \nu g(\boldsymbol{\theta})\} \tag{2.7}$$

where $\boldsymbol{\lambda}$ and ν are ***hyperparameters*** of the prior. Given a data set $\mathcal{D} = \{x_1, \ldots, x_N\}$ the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ has the same conjugate form as (2.7) but with parameters:

$$\boldsymbol{\lambda}' = \boldsymbol{\lambda} + \sum_{i=1}^{N} \boldsymbol{s}(x_i)$$

and

$$\nu' = \nu + N$$

Here we give equations for the marginal likelihoods of some simple probability distributions in the exponential family using their conjugate priors:

Bernoulli-Beta

The Bernoulli distribution is a distribution over binary data and has the following form:

$$P(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N}\prod_{d=1}^{D} \theta_d^{x_d^{(i)}}(1-\theta_d)^{(1-x_d^{(i)})} \tag{2.8}$$

where N is the number of data points in the dataset, D is the total number of dimensions, and θ_d gives the probability that the dth dimension is a 1 for any given data point, $x_d^{(i)} = 1$.

The prior distribution which is conjugate to the Bernoulli is the Beta distribution:

$$P(\theta|\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{d=1}^{D} \frac{\Gamma(\alpha_d+\beta_d)}{\Gamma(\alpha_d)\Gamma(\beta_d)}\theta_d^{\alpha_d-1}(1-\theta_d)^{\beta_d-1} \tag{2.9}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are hyperparmeters for the Beta distribution. Here $\Gamma(x)$ is the gamma function generalizing the factorial function to positive reals, $\Gamma(m) = (m-1)!$ for integer m and $\Gamma(x) = (x-1)\Gamma(x-1)$ for $x > 1$.

Plugging these equations into our marginal likelihood expression (2.4) we see that:

$$p(\mathcal{D}|\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{d=1}^{D} \frac{\Gamma(\alpha_d+\beta_d)\Gamma(\alpha_d+m_d)\Gamma(\beta_d+N-m_d)}{\Gamma(\alpha_d)\Gamma(\beta_d)\Gamma(\alpha_d+\beta_d+N)}$$

where $m_d = \sum_{i=1}^{N} x_d^{(i)}$.

Multinomial-Dirichlet

The Multinomial distribution is a distribution over discrete data and has the following form:

$$P(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N} \theta_1^{x_1^{(i)}} \dots \theta_K^{x_K^{(i)}} \tag{2.10}$$

where $\sum_i \theta_i = 1$, K is the total number of discrete bins, $x_k^{(i)}$ is a binary indicator variable indicating that the ith data point takes on the kth discrete value, and N is the number of data points in the dataset.

The prior distribution which is conjugate to the Multinomial is the Dirichlet distribution:

$$P(\theta|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\theta_1^{\alpha_1-1}\dots\theta_K^{\alpha_K-1} \tag{2.11}$$

where $\boldsymbol{\alpha}$ is a hyperparameter vector for the Dirichlet prior.

Plugging these equations into our marginal likelihood expression (2.4) we see that:

$$p(\mathcal{D}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k) \prod_{k=1}^{K} \Gamma(\alpha_k + m_k)}{\Gamma(M + \sum_{k=1}^{K} \alpha_k) \prod_{k=1}^{K} \Gamma(\alpha_k)} \tag{2.12}$$

where $m_k = \sum_{i=1}^{N} x_k^{(i)}$.

Normal-Inverse Wishart-Normal

The Normal distribution is a distribution over real, continuous data and has the following form:

$$P(\mathcal{D}|\Sigma, \boldsymbol{\mu}) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}^{(i)} - \boldsymbol{\mu})^{\top} \Sigma^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu})\right) \tag{2.13}$$

where N is the number of datapoints in $\mathcal{D}$, D is the dimensionality, $\boldsymbol{\mu}$ is the mean and Σ is the covariance matrix.

The prior distribution which is conjugate to the Normal is the Normal-Inverse Wishart distribution:

$$\begin{aligned} P(\Sigma, \boldsymbol{\mu}|\boldsymbol{m}, \boldsymbol{S}, r, v) &= \frac{r^{D/2}}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{\mu} - \boldsymbol{m})^{\top} r \Sigma^{-1} (\boldsymbol{\mu} - \boldsymbol{m})\right) \\ & \quad \frac{|\boldsymbol{S}|^{v/2} |\Sigma|^{-(v+D+1)/2} e^{-\frac{1}{2}\mathrm{trace}(\boldsymbol{S}\Sigma^{-1})}}{2^{vD/2} \pi^{D(D-1)/4} \prod_{d=1}^{D} \Gamma(\frac{v+1-d}{2})} \end{aligned} \tag{2.14}$$

The hyperparameters for this prior include: $\boldsymbol{m}$ which is the prior mean on the mean, $\boldsymbol{S}$ is the prior mean on the precision matrix, r is the scaling factor on the prior precision of the mean, and v is the degrees of freedom.

Plugging these equations into our marginal likelihood expression (2.4) we see that:

$$\begin{aligned} p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{m}, \boldsymbol{S}, r, v) &= (2\pi)^{-\frac{ND}{2}} \left(\frac{r}{N+r}\right)^{D/2} |\mathbf{S}|^{\frac{v}{2}} |\mathbf{S}'|^{-\frac{v'}{2}} \\ & \times \frac{(2)^{\frac{v'D}{2}} \prod_{d=1}^{D} \Gamma\left(\frac{v'+1-d}{2}\right)}{(2)^{\frac{vD}{2}} \prod_{d=1}^{D} \Gamma\left(\frac{v+1-d}{2}\right)} \end{aligned}$$

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:

$$v' = v + N$$

where X is the observed data.

2.2 Clustering

Clustering is a fundamentally important unsupervised learning problem. There are a variety of commonly used clustering methods some of which are reviewed in this section:

K-means clustering (Hartigan and Wong, 1979) and **Mixture Modeling** (McLachlan and Peel, 2000): K-means and Mixture modeling are the most common methods for canonical, flat clustering and are described in detail in the following subsections.

Hierarchical clustering (Johnson, 1967): In hierarchical clustering the goal is not to find a single partitioning of the data, but a hierarchy (generally represented by a tree) of partitionings which may reveal interesting structure in the data at multiple levels of granularity. Hierarchical clustering algorithms may be of two types, *agglomerative* algorithms look at ways of merging data points together to form a hierarchy, while *divisive* methods separate the data repeatedly into finer groups. Methods for agglomerative hierarchical clustering will be discussed in more detail in chapter 3.

Spectral clustering (Shi and Malik, 2000; Meila and Shi, 2001; Ng et al., 2002): In spectral clustering a similarity matrix is computed between all pairs of data points. An eigenvalue decomposition is then performed, data points are projected into a space spanned by a subset of the eigenvectors, and one of the previously mentioned clustering algorithms (typically K-means or hierarchical clustering) is used to cluster the data. Spectral clustering will not be discussed further in this thesis since it does not directly relate to the probabilistic methods presented here.

2.2.1 K-means

K-means (Hartigan and Wong, 1979) is a method for clustering a data set, $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$, of N unlabelled data points into K clusters, where K is specified by the user. The objective of the K-means algorithm is to minimize the following cost function:

$$V = \sum_{i=1}^{K} \sum_{x_j \in C_i} (x_j - \mu_i)^2 \tag{2.15}$$

where the C_i are each of the K clusters and μ_i are their respective cluster centers (means). The algorithm starts by randomly placing each of the K centers and assigning each data point to the cluster with the closest center. It then iteratively recalculates the cluster center based on the new assignments of data points to clusters, and then reassigns each data point, until convergence. This cost function has many local minima and several runs of the algorithm may be needed.

2.2.2 Finite Mixture Model

In a finite mixture model, data is modeled by a mixture of K probability distributions:

$$P(X|\Theta, \boldsymbol{\pi}) = \prod_{i=1}^{N} \sum_{j=1}^{K} \pi_j P(\mathbf{x}_i|\theta_j) \tag{2.16}$$

where π_j is the mixing weight (or mass) of cluster j ($\sum_j \pi_j = 1, \pi_j \geq 0$), $P(\mathbf{x}_i|\theta_j)$ is the probability density for cluster j with parameters θ_j, $\Theta = (\theta_1 \ldots \theta_K)$, and $\mathbf{x}_i$ is data point i in a data set of size N. Although the distribution $P(\mathbf{x}_i|\theta_j)$ may be in the exponential family, the mixture model is not, making inference and learning more difficult. Maximum likelihood learning for mixture models can be performed by computing partial derivatives of equation (2.16) with respect to the unknowns Θ and $\boldsymbol{\pi}$. These partials can then be used for gradient-based learning. More commonly, however, the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) is used for learning mixture models. The EM algorithm iterates between two steps:

1) Evaluate the responsibilities for each data point conditioned on the current parameter values. The responsibilites can be computed as follows:

$$P(s_i = j|\mathbf{x}_i) = \frac{\pi_j P(\mathbf{x}_i|\theta_j)}{\sum_\ell \pi_\ell P(x_i|\theta_\ell)} \tag{2.17}$$

where s_i are latent indicator variables such that $s_i = j$ means that data point i belongs to cluster j.

2) Optimize the parameters Θ given the responsibilities calculated in the previous step.

These two steps are iterated until convergence and Dempster et al. (1977) proves that at each iteration the likelihood is guaranteed to increase.

We can relate a finite mixture of Gaussians to the K-means algorithm from the previous section. In a finite mixture of Gaussians the clusters in our mixture model ($P(\mathbf{x}_i|\theta_j)$) are Gaussians with parameters $\Theta = \{\boldsymbol{\mu}, \Sigma\}$. If we constrain $\pi_j = \frac{1}{K}$ and $\Sigma_k = \sigma^2 I$ in the limit as $\sigma^2 \rightarrow 0$ then the responsibilities for data points are binary with value 1 for the cluster with the closest mean and 0 otherwise. This results in an EM procedure which is equivalent to the K-means algorithm (Roweis and Ghahramani, 1997). These K-means assumptions result in a very restrictive version of EM for mixtures of Gaussians which has limited modeling capability. The unrestricted mixture of Gaussians model is much more flexible, but still has potential problems with overfitting due to the maximum likelihood learning. For example, if a particular Gaussian cluster in the mixture is responsible for only one data point at some iteration then the re-estimated covariance for that Gaussian cluster will be singular, causing the algorithm to crash. In the next section we will describe a Bayesian mixture model which can avoid some of these issues.

2.2.3 Bayesian Mixture Model

A Bayesian Mixture Model is an extension of the finite mixture model described in the previous section. In order to avoid overfitting when estimating the parameters Θ, we put a prior, $P(\Theta)$ on model parameters and integrate them out. Integration makes sense here since Θ is unknown and we should average over all possible values the model parameters can take on. This is also called computing the marginal likelihood, given in section 2.1 (equation (2.4)). Following the same reasoning as with Θ, we also put a Dirichlet prior on $\boldsymbol{\pi}$ and integrate over it. We can thus write out the analogous equation to (2.16) from section 2.2.2 as follows:

$$P(X|\nu,\alpha) = \sum_{S} P(S|\alpha)P(X|S,\nu) \tag{2.18}$$

where the sum is over all possible settings of indicator variables S, $P(S|\alpha)$ integrates over $\boldsymbol{\pi}$:

$$\begin{aligned} P(S|\alpha) &= \int P(S|\boldsymbol{\pi})P(\boldsymbol{\pi}|\alpha)d\boldsymbol{\pi} \\ &= \int \prod_{i=1}^{N} P(s_i|\boldsymbol{\pi})P(\boldsymbol{\pi}|\alpha)d\boldsymbol{\pi} \end{aligned}$$

where $P(\boldsymbol{\pi}|\alpha)$ is the Dirichlet prior over $\boldsymbol{\pi}$ with hyperparameters α. $P(X|S,\nu)$ integrates over Θ:

$$\begin{aligned} P(X|S,\nu) &= \int P(X|\Theta,S)P(\Theta|\nu)d\Theta \\ &= \int \left[\prod_{i=1}^{N} P(\mathbf{x}_i|\boldsymbol{\theta}_{s_i})\right]\left[\prod_{j=1}^{K} P(\boldsymbol{\theta}_j|\nu)\right] d\Theta \end{aligned}$$

and so the probability of the data is now only conditioned on hyperparameters α and ν. The sum over S in (2.18) is intractable, so in practice approximation methods such as Markov Chain Monte Carlo (MCMC) (Neal, 2000) or variational approximations (Ghahramani and Beal, 1999) are used.

In this Bayesian mixture model we still need to set K, the number of clusters, in advance, but we can perform model comparisons. If we compute or approximate $P(X|\nu,\alpha)$ for many different values of K we can select the K with the highest value of this marginal probability of our data. In the next chapter we discuss nonparametric Bayesian methods (specifically Dirichlet Process Mixtures), which allow K to be inferred without explicitly performing expensive model comparisons. There is a very large literature on Bayesian mixture models, see Banfield and Raftery (1993) and Richardson and Green (1997).

2.3 Nonparametric Bayesian Methods

Nonparametric Bayesian methods give very flexible models in a Bayesian setting. Since Bayesian methods are most accurate when the prior adequately encapsulates one's beliefs, nonparametric priors are able to model data better than inflexible models with the number of parameters set a priori (e.g. a mixture of 3 Gaussians). Many nonparametric Bayesian models can be derived by starting with a standard parametric model and taking the limit as the number of parameters goes to infinity (e.g. an infinite mixture of Gaussians). These nonparametric models will automatically infer the correct model size (i.e. number of relevant parameters) from the data, without having to explicitly perform model comparisons (e.g. comparing a mixture of 3 Gaussians to a mixture of 4 Gaussians to determine the correct number).

2.3.1 Dirichlet Process Mixture Models

A classic model from nonparametric Bayesian statistics which will play a central role in this thesis is the Dirichlet Process Mixture (DPM). The DPM defines a mixture model with countably infinitely many components and can be used both for flexible density estimation and for clustering when the number of clusters is a priori unknown. There are many different derivations and constructions resulting in the DPM, and good reviews can be found in Teh et al. (2006) and Blei (2004). Here we focus on the derivation of DPMs starting from finite mixture models (Neal, 2000).

2.3.1.1 Derivation from the Infinite Limit of a Finite Mixture Model

This review of the derivation of a DPM from the infinite limit of a finite mixture model follows Rasmussen (2000). For each of N data points being modelled by a finite mixture, we can associate cluster indicator variables, $s_1...s_N$, which can take on values $s_i \in \{1...K\}$, where K is the number of mixture components, or clusters. The joint distribution of the indicator variables is multinomial with parameter vector $\pi = (\pi_1...\pi_K)$:

$$p(s_1...s_N|\pi) = \prod_{j=1}^{K} \pi_j^{n_j}, \qquad n_j = \sum_{i=1}^{N} \delta(s_i, j)$$

Here n_j is the number of data points assigned to class j and $\delta(k,j) = 1$ if $k = j$ and 0 otherwise. If we place a symmetric Dirichlet prior on π with concentration parameters $\frac{\alpha}{K}$:

$$p(\pi_1...\pi_K|\alpha) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} \prod_{j=1}^{K} \pi_j^{\alpha/K-1}$$

we can integrate π out, obtaining (2.12):

$$\begin{aligned} p(s_1...s_N|\alpha) &= \int p(s_1...s_N|\pi_1...\pi_K)p(\pi_1...\pi_K)d\pi_1...\pi_K \\ &= \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)} \prod_{j=1}^{K} \frac{\Gamma(n_j + \alpha/K)}{\Gamma(\alpha/K)} \end{aligned}$$

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

$$p(s_i = j|\mathbf{s}_{-i}, \alpha) = \frac{n_{-i,j} + \alpha/K}{N - 1 + \alpha}$$

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

$$\begin{aligned} p(s_i = j|\mathbf{s}_{-i}, \alpha) &= \frac{n_{-i,j}}{N - 1 + \alpha}, && (2.19) \\ p(s_i \neq s_{i'}, \forall i' \neq i|\mathbf{s}_{-i}, \alpha) &= \frac{\alpha}{N - 1 + \alpha} && (2.20) \end{aligned}$$

The second term is the prior probability that point i belongs to some new class that no other points belong to. These equations relate the DPM to the distribution over partitions of N objects known as the Chinese Restaurant Process (Aldous, 1983). Equation (2.19) says that the probability of data point i joining cluster j is proportional to the number of other points in cluster j — a sort of "rich get richer" property of these clusterings. Equation (2.20) indicates that with some probability proportional to α, the data point forms a new cluster. Thus the DPM allows new clusters to be represented as new points are observed.

By performing inference in the DPM, for example by Gibbs sampling (Neal, 2000), we automatically infer the total number of clusters represented in the data set and the assignment of data points to them.

2.3.2 Indian Buffet Process

Another nonparametric Bayesian model which we make use of in this thesis is the Indian Buffet Process (IBP) (Griffiths and Ghahramani, 2006). The IBP defines a distribution which can be used as a prior in probabilistic models for the representation of data items which may have a potentially infinite number of hidden features. More specifically, it defines a distribution over infinite binary matrices, which can be derived by starting

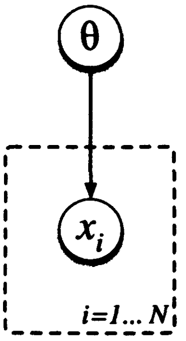


Figure 2.1: Example Directed Graphical Model

with a finite $N \times K$ matrix (where N is the number of data items, and K is the number of features), and taking the limit as K goes to infinity. Exchangeability of the rows is preserved, and the columns are independent. The IBP is a simple generative process which results in this distribution, and Markov Chain Monte Carlo algorithms have been used to do inference in this model (Görür et al., 2006). The IBP has been applied to modeling the presence or absence of objects in images (Griffiths and Ghahramani, 2006), and the underlying diseases of patients with multiple symptoms (Wood et al., 2006). An in-depth description of the IBP is provided in chapter 7.

2.4 Directed Graphical Models

Directed graphical models are occasionally provided in this thesis in order to formalize the independence assumptions between random variables in models. A directed graphical model (Pearl, 1988) represents a factorization of the joint probability distribution over all random variables, where the nodes are the random variables, edges represent potential dependencies between variables, and plates encapsulate nodes and edges which are replicated in accordance with the indexing of the variables on the plate.

An example of a directed graphical model is given in figure 2.1. This graphical model represents a data set $X = \{x_1, x_2, \ldots, x_N\}$ which is generated iid (independently and identically distributed) from a probabilistic model with parameters θ. The exact form of the probabilistic model is not provided by the graphical model. However if our data was generated from a Gaussian distribution, here θ might be the mean and variance parameters of the particular Gaussian which generated X. Some of the variables in the graphical model may be latent (or unobserved), for example we might not know the mean and variance of the Gaussian which generated our data, and we may be interested in infering these values.

Chapter 3

Bayesian Hierarchical Clustering

In this chapter we present a novel algorithm for agglomerative hierarchical clustering based on evaluating marginal likelihoods of a probabilistic model. We show that 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). In section 3.3 we prove that 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. In section 3.4 we describe procedures for learning the model hyperparameters, computing the predictive distribution, and extensions to the algorithm. In section 3.5, experimental results on synthetic and real-world data sets demonstrate useful properties of the algorithm. Lastly, in section 3.6 we propose using randomized algorithms to improve the running time, so our method can be used on very large data sets.

3.1 Traditional Hierarchical Clustering

Hierarchical clustering is one of the most frequently used methods in unsupervised learning. Given a set of data points, hierarchical clustering outputs 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 and Hart (1973) and shown in algorithm 3.1, 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).

Algorithm 3.1 Traditional Hierarchical Clustering

```
input: Data $D = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$, and distance metric $F$
initialise: number of clusters $c = n$, and $D_i = \{\mathbf{x}^{(i)}\}$
while $c > 1$ do
    Find the pair $D_i$ and $D_j$ which minimise:
        $dist_k = F(D_i, D_j)$
    Merge $D_k \leftarrow D_i \cup D_j$, Delete $D_i$ and $D_j$, $c \leftarrow c - 1$
end while
output: A sequence of cluster merges, and the corresponding tree
```

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. In this chapter, we use statistical inference to overcome these limitations. Previous work which uses probabilistic methods to perform hierarchical clustering is discussed in section 3.7.

3.2 The Bayesian Hierarchical Clustering Algorithm

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.

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 3.3).

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 $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$ 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, $\{T_i : i = 1 \dots n\}$ 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 3.1(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 $\mathcal{H}_1^k$:

$$p(\mathcal{D}_k|\mathcal{H}_1^k) = \int p(\mathcal{D}_k|\theta)p(\theta|\beta)d\theta = \int \Big[\prod_{\mathbf{x}^{(i)} \in \mathcal{D}_k} p(\mathbf{x}^{(i)}|\theta) \Big] p(\theta|\beta)d\theta \qquad (3.1)$$

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 chapter 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, (3.1) 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-*

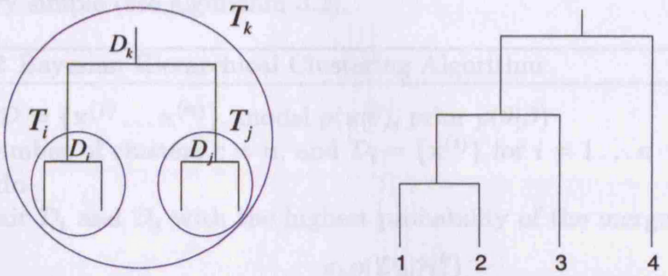


Figure 3.1: (a) Schematic of a portion of a tree where T_i and T_j are merged into T_k, and the associated data sets $\mathcal{D}_i$ and $\mathcal{D}_j$ are merged into $\mathcal{D}_k$. (b) An example tree with 4 data points. The clusterings (1 2 3)(4) and (1 2)(3)(4) are tree-consistent partitions of this data. The clustering (1)(2 3)(4) is not a tree-consistent partition.

consistent partitions in section 3.3 and figure 3.1(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:

$$p(\mathcal{D}_k|T_k) = \pi_k p(\mathcal{D}_k|\mathcal{H}_1^k) + (1-\pi_k)p(\mathcal{D}_i|T_i)p(\mathcal{D}_j|T_j) \tag{3.2}$$

This equation is defined recursively, where 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 clusterings of the data in $\mathcal{D}_k$ which are consistent with the tree structure (see figure 3.1(b)). A clustering of $\mathcal{D}$ is tree consistent with tree T if each cluster consists of data points in some subtree of T. In section 3.3 we show that equation (3.2) 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:

$$r_k = \frac{\pi_k p(\mathcal{D}_k|\mathcal{H}_1^k)}{\pi_k p(\mathcal{D}_k|\mathcal{H}_1^k) + (1-\pi_k)p(\mathcal{D}_i|T_i)p(\mathcal{D}_j|T_j)} \tag{3.3}$$

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

[1] It is important not to confuse the marginal likelihood in equation (3.1), which integrates over the parameters of one cluster, and the marginal likelihood of a DPM, which integrates over all clusterings and their parameters.

algorithm is very simple (see algorithm 3.2).

Algorithm 3.2 Bayesian Hierarchical Clustering Algorithm

```
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 $\mathcal{D}_k \leftarrow \mathcal{D}_i \cup \mathcal{D}_j$,  $T_k \leftarrow (T_i, T_j)$
    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 $r_k < 0.5$
```

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.

3.3 BHC Theory and Dirichlet Process Mixture Models

The above algorithm is an approximate inference method for Dirichlet Process mixture models (DPMs). Dirichlet Process mixture models consider the limit of infinitely many components of a finite mixture model (see chapter 2). 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 K components

$$p(\mathbf{x}^{(i)}|\phi) = \sum_{j=1}^{K} p(\mathbf{x}^{(i)}|\theta_j) p(s_i = j|\mathbf{p}) \tag{3.4}$$

where $s_i \in \{1, \ldots, K\}$ is a cluster indicator variable for data point i, $\mathbf{p}$ are the parameters of a multinomial distribution with $p(s_i = j|\mathbf{p}) = \mathrm{p}_j$, θ_j are the parameters of the jth component, and $\phi = (\theta_1, \ldots, \theta_K, \mathbf{p})$. Let the parameters of each component have conjugate priors $p(\theta|\beta)$ as in section 3.2, and the multinomial parameters also have a conjugate Dirichlet prior

$$p(\mathbf{p}|\alpha) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} \prod_{j=1}^{K} \mathrm{p}_j^{\alpha/K-1}. \tag{3.5}$$

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

$$p(\mathcal{D}|\alpha, \beta) = \int \left[\prod_{i=1}^{n} p(\mathbf{x}^{(i)}|\phi)\right] p(\phi|\alpha, \beta) d\phi \tag{3.6}$$

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

$$p(\mathcal{D}|\alpha, \beta) = \sum_{\boldsymbol{s}} p(\boldsymbol{s}|\alpha) p(\mathcal{D}|\boldsymbol{s}, \beta) \tag{3.7}$$

where $\boldsymbol{s} = (s_1, \ldots, s_n)$ and $p(\boldsymbol{s}|\alpha) = \int p(\boldsymbol{s}|\mathbf{p}) p(\mathbf{p}|\alpha) d\mathbf{p}$ is a standard Dirichlet integral. The quantity (3.7) is well-defined even in the limit $K \to \infty$. Although the number of possible settings of $\boldsymbol{s}$ grows as K^n and therefore diverges as $K \to \infty$, 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 (3.7) as:

$$p(\mathcal{D}|\alpha, \beta) = \sum_{v \in \mathcal{V}} p(v|\alpha) p(\mathcal{D}|v, \beta) \tag{3.8}$$

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. We have also included a more substantial review of DPMs in chapter 2 of this thesis. DPMs have the interesting property that the probability of a new data point belonging to a cluster is proportional to the number of points already in that cluster (Blackwell and MacQueen (1973)), where α controls the probability of the new point creating a new cluster.

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 3.1(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 in $\mathcal{D}_k$ (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 (algorithm 3.3). In algorithm 3.3, 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.

Algorithm 3.3 Algorithm for computing prior on merging for BHC.

```
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
```

We now have the ingredients to relate our algorithm to exact inference in DPMs.

Lemma 3.1 *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, n_ℓ^v is the number of points in cluster ℓ of partitioning v, and $\mathcal{D}_\ell^v$ are the points in cluster ℓ of partitioning v.

This is easily shown since:

$$p(\mathcal{D}_k) = \sum_{v \in \mathcal{V}} p(v)p(\mathcal{D}^v)$$

where:

$$p(v) = \frac{\alpha^{m_v} \prod_{\ell=1}^{m_v} \Gamma(n_\ell^v)}{\left[\frac{\Gamma(n_k+\alpha)}{\Gamma(\alpha)}\right]} \tag{3.9}$$

and:

$$p(\mathcal{D}^v) = \prod_{\ell=1}^{m_v} p(\mathcal{D}_\ell^v)$$

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. Expression (3.9) for $p(v)$ follows from (2.19) and (2.20) applied to a data set of n_k points clustered according to v. Each of the m_v clusters contributes an α by (2.20) and the remaining terms follow from the fact that $\Gamma(x) = (x-1)\Gamma(x-1)$, (2.19) and (2.20).

Theorem 3.1 *The quantity (3.2) computed by the Bayesian Hierarchical Clustering* *algorithm is:*

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

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

Proof Rewriting equation (3.2) using algorithm 3.3 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:

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

where $\mathcal{V}_{\mathrm{NTT}}$ is the set of all non-trivial tree-consistent partionings of $\mathcal{D}_k$. For the trivial single cluster 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 3.1. This completes the proof. Another way to get this result is to expand out $p(\mathcal{D}|T)$ and substitute for π using algorithm 3.3.

Corollary 3.1 *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 This 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 3.1 (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 3.1 *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 $C_iC_j + 1$ 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, $p(\mathcal{D}_k|T_k)$ 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 by algorithm 3.2 is $O(n^2)$, the complexity of computing the marginal likelihood, give a tree, is $O(n)$, and the complexity of computing the predictive distribution (see section 3.4.2) is $O(n)$.

3.4 Learning and Prediction

This section discusses ways of learning the hyperparameters and computing predictive distributions for BHC.

3.4.1 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 and equation (3.2) we can compute gradients for the component model hyperparameters bottom-up as the tree is being built:

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

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 from algorithm 3.3:

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

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.

3.4.2 Predictive Distribution

For any tree, the probability of a new test point $\mathbf{x}$ 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 $p(\mathbf{x}|\mathcal{D}_k) = \int p(\mathbf{x}|\theta)p(\theta|\mathcal{D}_k, \beta)d\theta$ where $\mathcal{D}_k$ is the set of data points in cluster k. The overall predictive distribution sums over all nodes weighted by their posterior probabilities:

$$p(\mathbf{x}|\mathcal{D}) = \sum_{k \in \mathcal{N}} \omega_k \, p(\mathbf{x}|\mathcal{D}_k) \qquad (3.10)$$

where

$$\omega_k \overset{\text{def}}{=} r_k \prod_{i \in \mathcal{N}_k} \left[(1-r_i)\frac{n_{i+}}{n_{i+} + n_{i-}}\right] \qquad (3.11)$$

is the weight on cluster k, $\mathcal{N}$ is the set of all nodes in the tree, and $\mathcal{N}_k$ is the set of nodes on the path from the root node to the parent of node k. Here n_{i+} is the number of data points in the child node of node i on the path to node k, and n_{i-} is the number of data points in the sibling child node.

To account for the possibility that the new test point $\mathbf{x}$ is in its own cluster and does not belong to any of the $|\mathcal{N}|$ clusters represented in the tree, the tree is augmented with a new node placed as a sibling to the root node. This new node is given α fictional data points and predictive distribution $p(\mathbf{x})$. Since it is a sibling of the former root, n_{i-} for the root is therefore α. Including this node adds the required term $\frac{\alpha}{n+\alpha}p(\mathbf{x})$ into the sum (3.10), and downweights the predictions of the rest of the tree by $\frac{n}{n+\alpha}$.

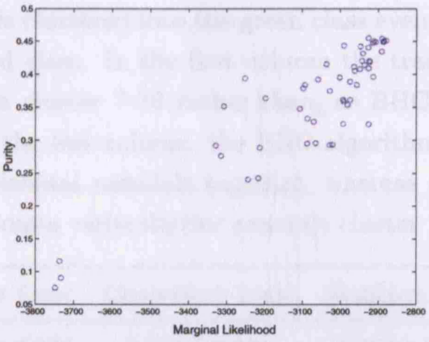


Figure 3.2: Log marginal likelihood (evidence) vs. purity over 50 iterations of hyperparameter optimization on the Newsgroups data set.

This expression (3.10) 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. We can dissect equation (3.11) to gain intuition about the weights. The kth cluster appears in no partitionings represented below node k, so we must consider the path from the root to node k, stopping at k. To reach k, at each node i along this path, the merged hypothesis must not be chosen; this happens with probability $1 - r_i$. At each branch point, the child node leading to k is chosen in proportion to the number of points in that node[2], introducing the terms $\frac{n_{i+}}{n_{i+}+n_{i-}}$. Finally, the process stops at node k without any further splits, with probability r_k. Putting these terms together gives us equation (3.11). It is easy to confirm that the weights ω_k sum to one, and that for a simple tree with two data points expression (3.10) is exact since all partitionings are tree consisent.

For conjugate priors and exponential family components the predictive distribution is simple to compute by summing over the nodes in the tree. For example, Gaussian components with conjugate priors results in a predictive distribution which is a mixture of multivariate t distributions. We show some examples of this in the Results section.

3.5 Results

We compared Bayesian hierarchical clustering to traditional hierarchical clustering using average, single, and complete linkage over 5 datasets (4 real and 1 synthetic). These traditional algorithms are described in Duda et al. (2001) and implemented in the Matlab statistics toolbox. We also compared our algorithm to average linkage hierarchical clustering on several toy 2D problems (figure 3.5). On these toy problems we were able to compare the different hierarchies generated by the two algorithms, and visualize clusterings and predictive distributions. We can see in figure 3.5 that the BHC algorithm behaves structurally more sensibly than the traditional algorithm. For example, point

[2] This is necessarily so given that the model is a Dirichlet Process Mixture.

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).

Data Set	Single Link	Complete Link	Average Link	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 3.1: Purity scores for 3 kinds of traditional agglomerative clustering, and Bayesian hierarchical clustering.

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.*[3] 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 (figure 3.2). 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

[3] 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.

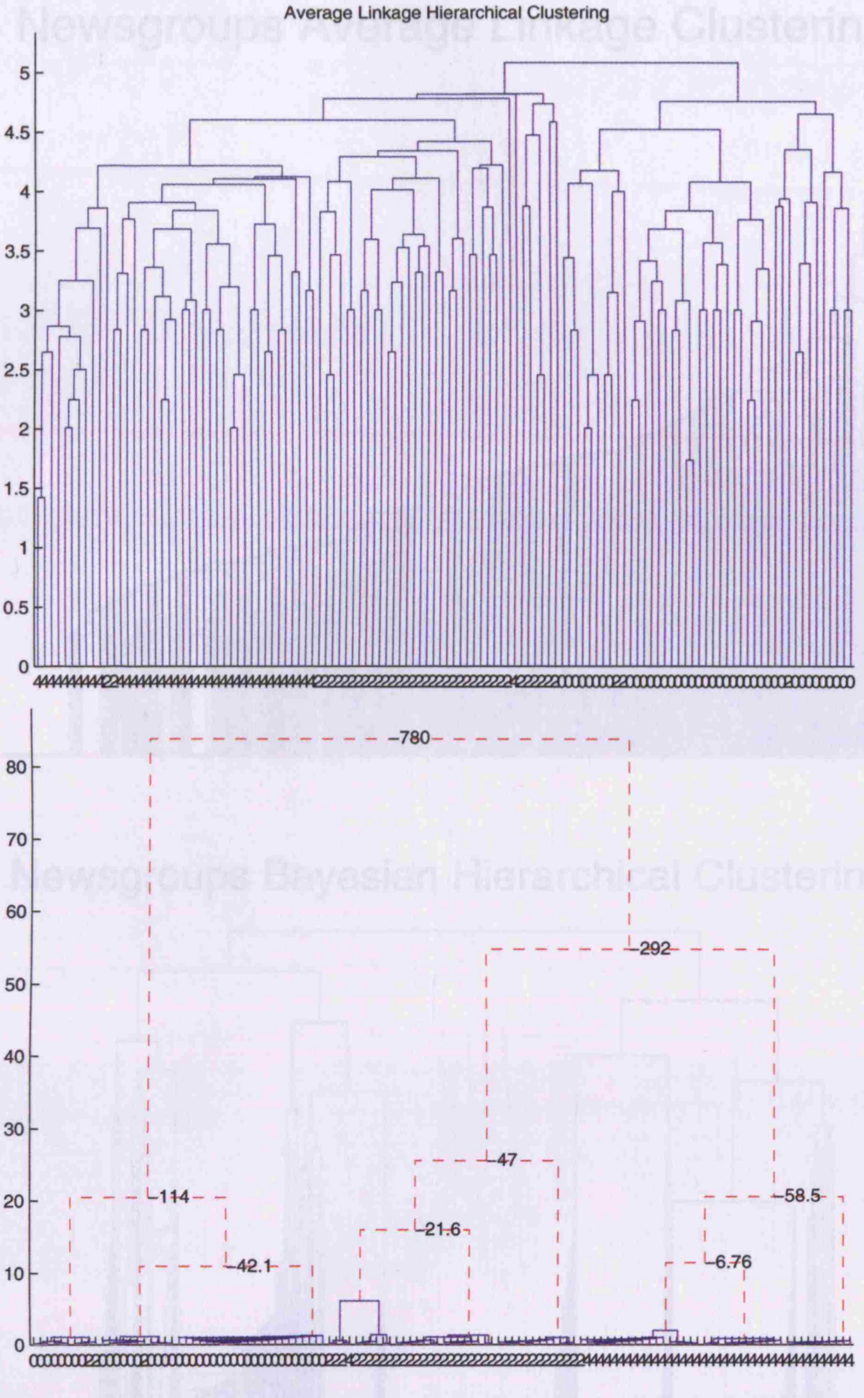
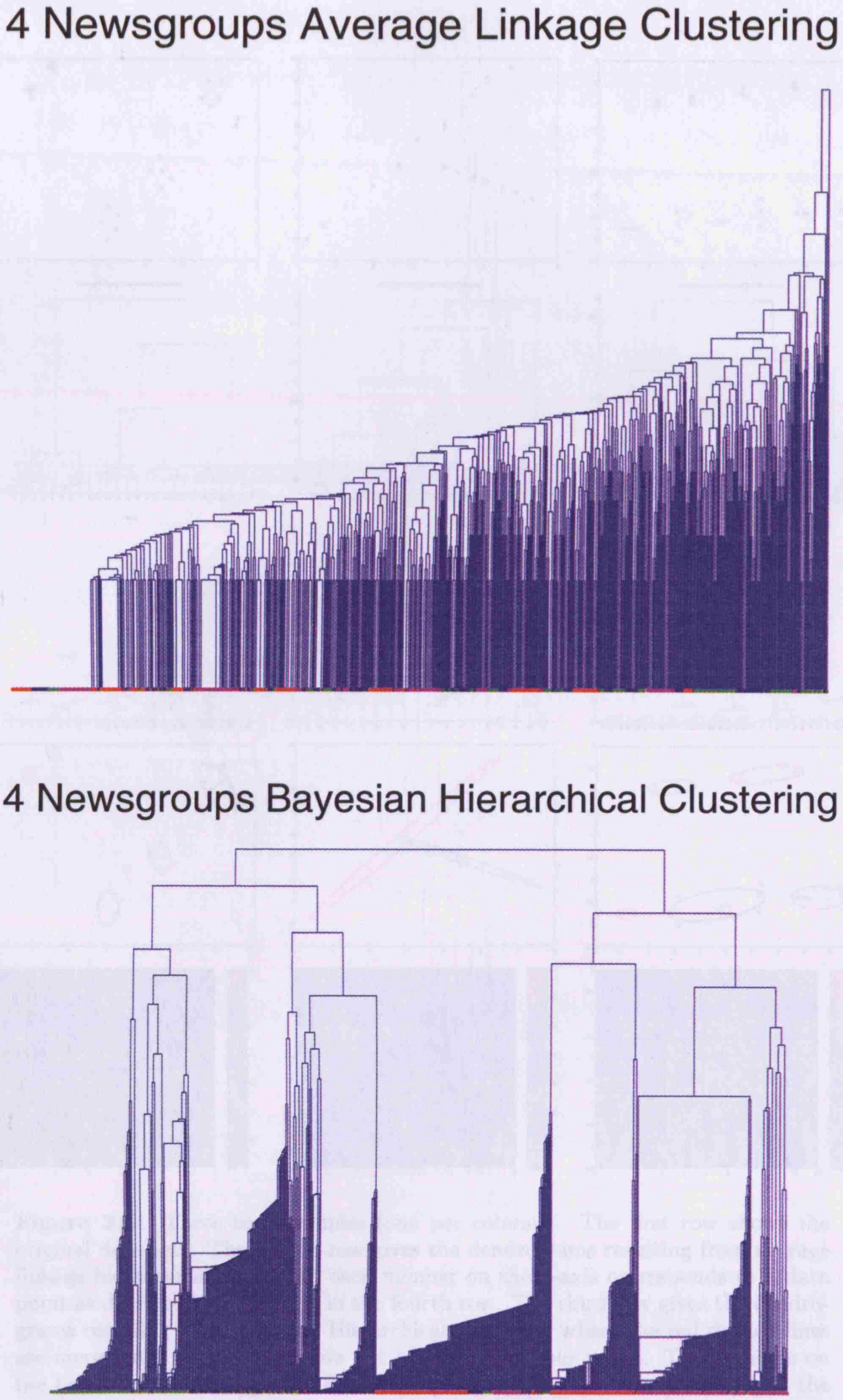


Figure 3.3: (a) The tree given by performing average linkage hierarchical clustering on 120 examples of 3 digits (0,2, and 4), yielding a purity score of 0.870. Numbers on the x-axis correspond to the true class label of each example, while the y-axis corresponds to distance between merged clusters. (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.



4 Newsgroups Average Linkage Clustering

4 Newsgroups Bayesian Hierarchical Clustering

Figure 3.4: 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.

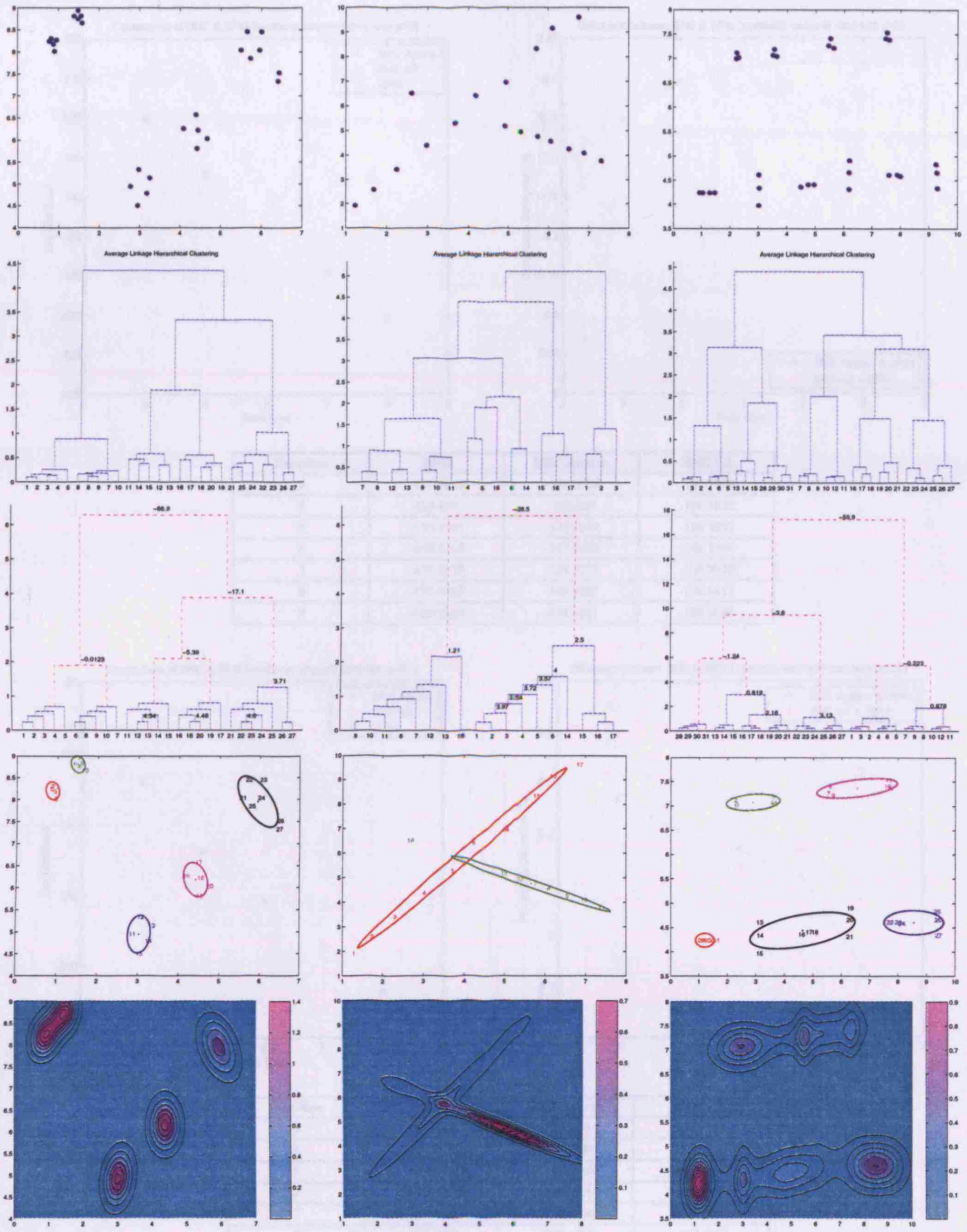
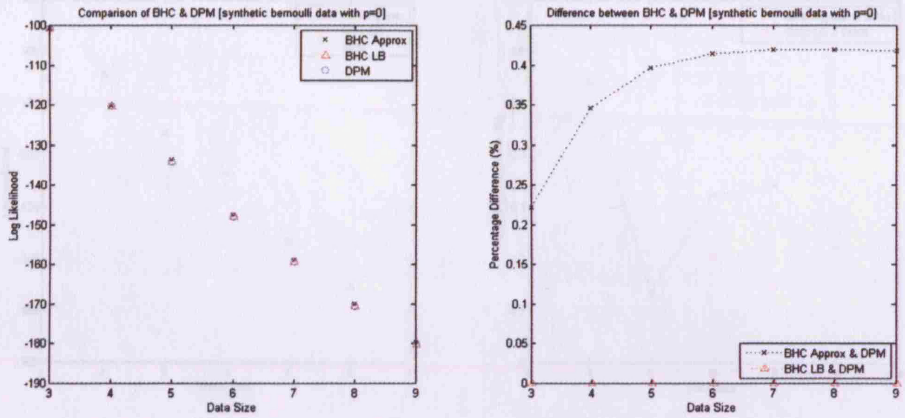
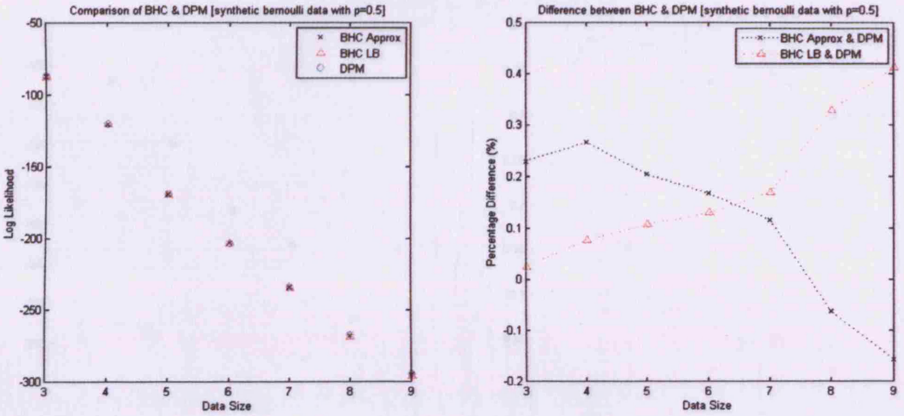


Figure 3.5: Three toy examples (one per column). The first row shows the original data sets. The second row gives the dendrograms resulting from average linkage hierarchical clustering, each number on the x-axis corresponds to a data point as displayed on the plots in the fourth row. The third row gives 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.

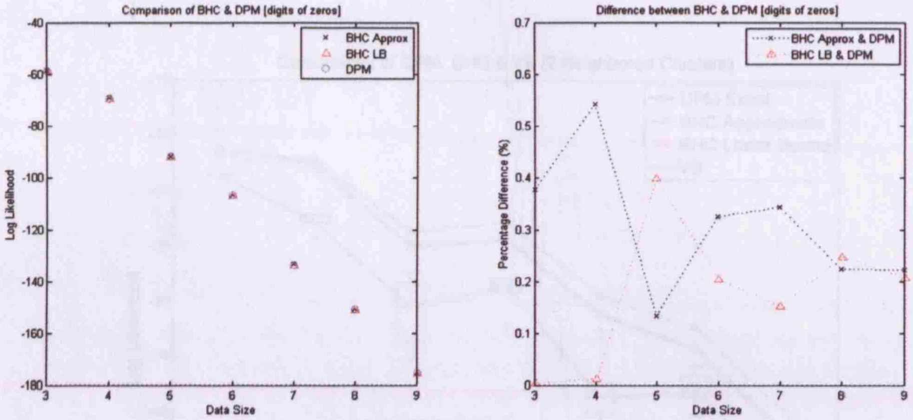


Data Size	DPM	BHC Approx	BHC LB
3	-100.7828	-100.5588	-100.7828
4	-120.4993	-120.083	-120.4993
5	-134.1851	-133.6535	-134.1852
6	-148.1364	-147.5225	-148.1364
7	-159.5938	-158.9237	-159.5938
8	-170.6413	-169.9252	-170.6413
9	-180.3924	-179.6381	-180.3924

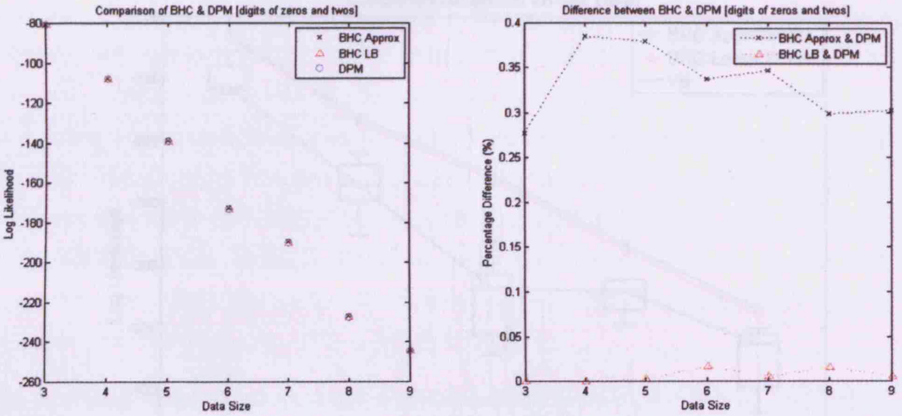


Data Size	DPM	BHC Approx	BHC LB
3	-87.6613	-87.4589	-87.6828
4	-120.8116	-120.4874	-120.9037
5	-169.4232	-169.0739	-169.6056
6	-203.5377	-203.1959	-203.803
7	-234.105	-233.8324	-234.5027
8	-267.4521	-267.6208	-268.3361
9	-294.6543	-295.1163	-295.8698

Figure 3.6: The top plots and table compare the BHC approximation and lower bound to the true log marginal likelihood of a DPM on a synthetic Bernoulli data set where data points are all 0s or all 1s (50 dimensional). The left plot shows the data set size versus log marginal likelihood, while the right plot gives the data set size versus the percentage difference of the approximation methods from the truth. The table lists the computed log marginal likelihoods by method and data set size. The bottom two plots and table are the same but for a synthetic dataset where all features for all data points have a 0.5 probability of being a 1.



Data Size	DPM	BHC Approx	BHC LB
3	-58.6484	-58.4167	-58.6407
4	-69.3149	-68.9381	-69.3226
5	-91.735	-91.6119	-92.1011
6	-106.7813	-106.4346	-106.9996
7	-133.8339	-133.3741	-134.0369
8	-150.7373	-150.4006	-151.1096
9	-175.4661	-175.0759	-175.8292



Data Size	DPM	BHC Approx	BHC LB
3	-80.6996	-80.4756	-80.6996
4	-108.002	-107.5857	-108.002
5	-139.095	-138.5668	-139.0984
6	-173.3023	-172.7176	-173.3315
7	-190.1874	-189.5284	-190.1985
8	-227.4512	-226.7703	-227.4864
9	-244.5553	-243.8142	-244.5685

Figure 3.7: The top plots and table compare the BHC approximation and lower bound to the true log marginal likelihood of a DPM on the digits data set where all data points are the digit 0 (binary, 64 dimensional). The left plot shows the data set size versus log marginal likelihood, while the right plot gives the data set size versus the percentage difference of the approximation methods from the truth. The table lists the computed log marginal likelihoods by method and data set size. The bottom two plots and table are the same but for a digits data set with both the digits 0 and 2.

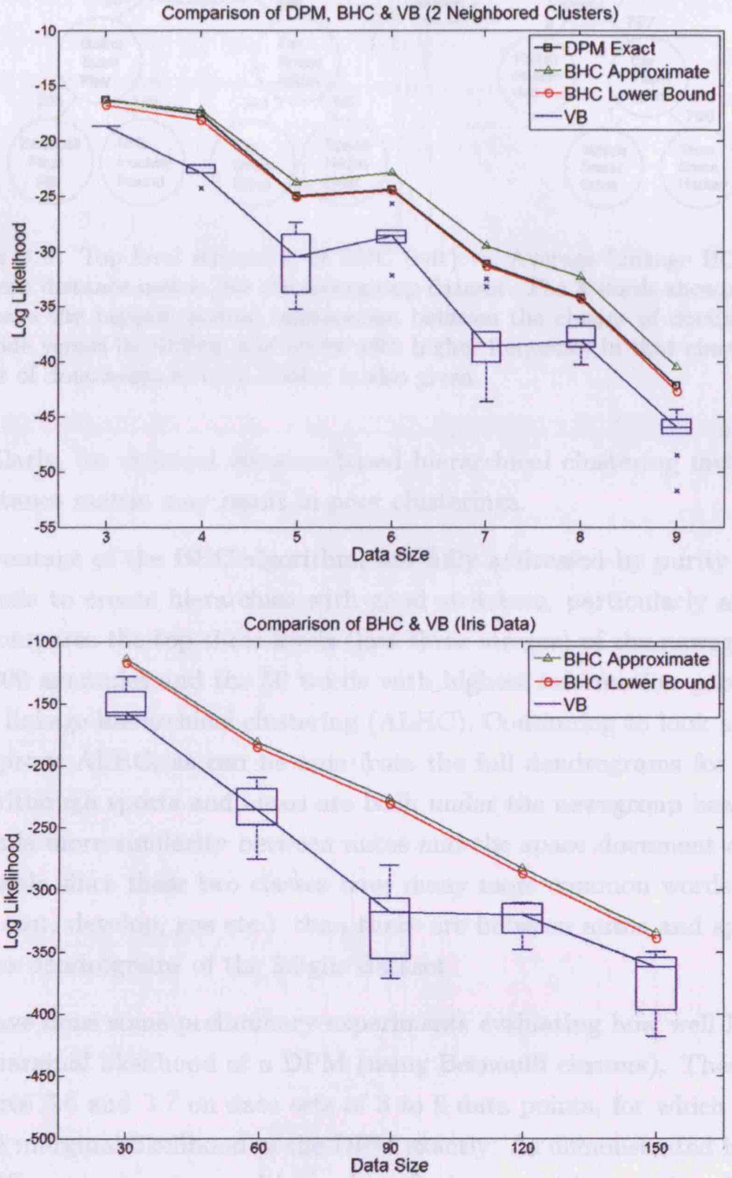


Figure 3.8: The top plot compares the BHC approximation and lower bound to the true log marginal likelihood of a DPM and to the lower bound given by Variational Bayes. The synthetic data in this plot was generated from two close Guassians (and Gaussian clusters are used to model the data). The bottom plot compares the BHC approximation and lower bound to that of Variational Bayes on the Iris UCI dataset. Here the true log marginal likelihood is not known but since both BHC and VB give lower bounds we can still see which is performing better. The VB results are taken over multiple runs and therefore have associated box plots.

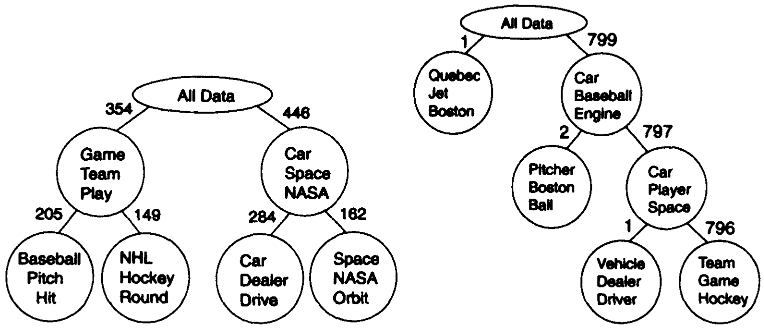


Figure 3.9: Top level structure, of BHC (left) vs. Average Linkage HC with a Euclidean distance metric, for the newsgroup dataset. The 3 words shown at each node have the highest mutual information between the cluster of documents at that node versus its sibling, and occur with higher frequency in that cluster. The number of documents at each cluster is also given.

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 3.9 compares the top three levels (last three merges) of the newsgroups hierarchy (using 800 examples and the 50 words with highest information gain) from BHC and average linkage hierarchical clustering (ALHC). Continuing to look at lower levels does not improve ALHC, as can be seen from the full dendrograms for this dataset, figure 3.4. 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 3.3 also shows dendrograms of the 3digits dataset.

Lastly, we have done some preliminary experiments evaluating how well BHC approximates the marginal likelihood of a DPM (using Bernoulli clusters). These results are given in figures 3.6 and 3.7 on data sets of 3 to 9 data points, for which we can compute the true marginal likelihood of the DPM exactly. As demonstrated by the figures both the BHC approximation and lower bound give promising results. We have also compared the BHC approximation and lower bound with the lower bound given by Variational Bayes. These results are given in figure 3.8.

3.6 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 3.2 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 (Motwani and Raghavan, 1995), to create a faster BHC algorithm. We propose the following randomized algorithm for fast Bayesian Hierarchical Clustering (RBHC):

Algorithm 3.4 Randomized Bayesian Hierarchical Clustering (RBHC) Algorithm

```
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 BHC$(\mathcal{D}^{[m]})$ obtaining a tree $T$
  Filter$(\mathcal{D} \backslash \mathcal{D}^{[m]})$, $\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_R$, through the top level of $T$, obtaining $\mathcal{D}_L$ and $\mathcal{D}_R$
  recurse: run RBHC$(\mathcal{D}_L)$ and RBHC$(\mathcal{D}_R)$
output: Bayesian mixture model where each tree node is a mixture component
```

The algorithm takes in a data set $\mathcal{D}$ and randomly selects a subset $\mathcal{D}^{[m]}$ 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 (algorithm 3.5) takes in the top level partitioning of tree T $(\mathcal{D}_L^{[m]}$ and $\mathcal{D}_R^{[m]})$, along with the priors $(\pi_L$ and $\pi_R)$ 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.

Algorithm 3.5 Filter Algorithm for RBHC

```
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: $\mathcal{D}_L, \mathcal{D}_R$
```

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 of $\mathcal{D}$. 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 $(\gamma n, (1 - \gamma)n)$ 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 3.2 *The RBHC algorithm is $O(nm\log n)$*

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

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

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

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

This gives us $\log n$ terms of $O(nm)$. We can generalize so that γ 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 $O(nmL)$.

We also propose the following alternative randomized algorithm based on EM (algorithm 3.6). 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):

Algorithm 3.6 A randomized BHC algorithm using EM (EMBHC)

```
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
```

Proposition 3.3 *The EMBHC algorithm is $O(nm)$*

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.

3.7 Related Work

The work presented in this chapter is related to and inspired by several previous probabilistic approaches to clustering[4], 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 (2001) 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 (1995) 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 BHC algorithm 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 which is exponentially many other clusterings of the same data (not one vs two clusters at each stage). Lastly, our BHC algorithm does not use any iterative

[4] 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.

method, like EM, or require sampling, like MCMC, and is therefore significantly faster than most of the above algorithms.

3.8 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 this chapter. We have presented prediction and hyperparameter optimization procedures and shown that the algorithm 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. Lastly, we proposed randomized versions of BHC to improve on the computational complexity of the original algorithm which is quadratic in the number of data points.

The limitations of our algorithm include its inherent greediness 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 (3.1). We also plan to extend BHC to systematically incorporate hyperparameter optimization and test the proposed randomized versions of BHC with improved running times. We will compare this novel, fast inference algorithm for DPMs to other inference algorithms such as MCMC (Rasmussen, 2000), EP (Minka and Ghahramani, 2003) and Variational Bayes (Blei and 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. Lastly, we have started to apply BHC to hierarchically cluster gene expression data with some good preliminary results and we hope to continue this work on this application. There are many exciting avenues for further work in this area.

Chapter 4

Information Retrieval using Bayesian Sets

Humans readily learn new concepts after observing a few examples and show extremely good generalization to new instances. In contrast, search tools on the internet exhibit little or no learning and generalization. In this chapter, we present a novel framework for retrieving information based on principles governing how humans learn new concepts and generalize. Given a query consisting of a set of items representing some concept, our method automatically infers which other items are relevant to that concept and retrieves them. Unlike previous such tools, this method evaluates items in terms of set membership, and is based on a Bayesian statistical model of human learning and generalization. Moreover, the underlying computations reduce to an extremely efficient sparse linear equation, making it practical for large scale retrieval problems. We show example applications including searches for scientific articles, proteins, and movies.

4.1 Information Retrieval

The last few decades have seen an explosion in the amount of valuable information available to people. This information has the potential to greatly impact science, society and commerce, but to maximize its use we need advanced tools which meet the challenge of identifying and extracting information that is relevant to us, while filtering out the irrelevant (Salton and McGill, 1983; Schatz, 1997; Brin and Page, 1998; Manning et al., 2007).

Tools such as Google , Amazon , PubMed , and eBay allow a person to type a query representing her information need, and attempt to retrieve items that are relevant to this query. Most successful tools for information retrieval have exploited advances in computer science such as novel data structures, faster computers and vast datasets. However, the problem of information retrieval is also fundamentally an inference problem—

what is the user's intended target given the relatively small amount of data in the query? For example, given the query consisting of the names of two movies: "Gone with the Wind" and "Casablanca," the intended target might be classic romance movies. To answer such queries, we need an understanding of human cognitive patterns of generalization from the point of view of statistical inference.

In the rest of this chapter we demonstrate a novel and efficient approach to information retrieval based on models of human generalization from limited information. The problem of human generalization has been intensely studied in cognitive science and mathematical psychology, and various models have been proposed based on some measure of similarity and feature relevance (Shepard, 1987; Tversky, 1977; Nosofsky, 1986). We focus on a recent framework for human category learning and generalization based on Bayesian inference (Tenenbaum and Griffiths, 2001). While attempts have been made to view information retrieval from a probabilistic framework, (e.g. Robertson and Sparck-Jones (1976); Lafferty and Zhai (2003); Ponte and Croft (1998); Cowans (2006)) none have considered models of human cognition in a fully Bayesian treatment of the problem, nor have they focused on the problem of retrieving sets of items as we now describe.

4.2 Retrieving Sets of Items

Consider a universe of items $\mathcal{D}$. Depending on the application, the set $\mathcal{D}$ may consist of web pages, movies, people, words, proteins, images, or any other object we may wish to form queries on. The user provides a query in the form of a very small subset of items $\mathcal{Q} \subset \mathcal{D}$. The assumption is that the elements in $\mathcal{Q}$ are examples of some concept / class / cluster in the data. The algorithm then has to provide a completion to the set $\mathcal{Q}$—that is, some set $\mathcal{Q}' \subset \mathcal{D}$ which presumably includes all the elements in $\mathcal{Q}$ and other elements in $\mathcal{D}$ which are also in this concept / class / cluster[1].

We can view this problem from several perspectives. First, the query can be interpreted as elements of some unknown cluster, and the output of the algorithm is the completion of that cluster. Whereas most clustering algorithms are completely unsupervised, here the query provides supervised hints or constraints as to the membership of a particular cluster. We call this view *clustering on demand*, since it involves forming a cluster once some elements of that cluster have been revealed. An important advantage of this approach over traditional clustering is that the few elements in the query can give useful information as to the features which are relevant for forming the cluster. For example, the query "Bush", "Nixon", "Reagan" suggests that the features *republican* and *US President* are relevant to the cluster, while the query "Bush", "Putin", "Brown" suggests that *current* and *world leader* are relevant. Given the huge number of features

[1] From here on, we will use the term "cluster" to refer to the target concept.

in many real world data sets, such hints as to feature relevance can produce much more sensible clusters.

Second, we can think of the goal of the algorithm to be to solve a particular information retrieval problem. As in other retrieval problems, the output should be relevant to the query, and it makes sense to limit the output to the top few items ranked by relevance to the query. In our experiments, we take this approach and report items ranked by relevance, where our relevance criterion is closely related to the Bayesian framework for understanding patterns of generalization in human cognition given by Tenenbaum and Griffiths (2001).

4.3 Bayesian Sets Algorithm

Let $\mathcal{D}$ be a data set of items, and $\mathbf{x} \in \mathcal{D}$ be an item from this set. Assume the user provides a query set $\mathcal{Q}$ which is a small subset of $\mathcal{D}$. Our goal is to rank the elements of $\mathcal{D}$ by how well they would "fit into" a set which includes $\mathcal{Q}$. Intuitively, the task is clear: if the set $\mathcal{D}$ is the set of all movies, and the query set consists of two animated Disney movies, we expect other animated Disney movies to be ranked highly.

We use a model-based probabilistic criterion to measure how well items fit into $\mathcal{Q}$. Having observed $\mathcal{Q}$ as belonging to some concept, we want to know how probable it is that $\mathbf{x}$ also belongs with $\mathcal{Q}$. This is measured by $p(\mathbf{x}|\mathcal{Q})$. Ranking items simply by this probability is not sensible since some items may be more probable than others, regardless of $\mathcal{Q}$. For example, under most sensible models, the probability of a string decreases with the number of characters, the probability of an image decreases with the number of pixels, and the probability of any continuous variable decreases with the precision to which it is measured. We want to remove these effects, so we compute the ratio:

$$\text{score}(\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{Q})}{p(\mathbf{x})} \tag{4.1}$$

where the denominator is the prior probability of $\mathbf{x}$ and under most sensible models will scale exactly correctly with number of pixels, characters, discretization level, etc. Using Bayes rule, this score can be re-written as:

$$\text{score}(\mathbf{x}) = \frac{p(\mathbf{x}, \mathcal{Q})}{p(\mathbf{x})\, p(\mathcal{Q})} \tag{4.2}$$

which can be interpreted as the ratio of the joint probability of observing $\mathbf{x}$ *and* $\mathcal{Q}$, to the probability of independently observing $\mathbf{x}$ and $\mathcal{Q}$. Intuitively, this ratio compares the probability that $\mathbf{x}$ and $\mathcal{Q}$ were generated by the same model with the *same*, though unknown, parameters θ, to the probability that $\mathbf{x}$ and $\mathcal{Q}$ came from models with *different* parameters θ and θ' (see figure 4.1 left and right respectively). Finally, up to a multiplicative constant independent of $\mathbf{x}$, the score can be written as: $\text{score}(\mathbf{x}) = p(\mathcal{Q}|\mathbf{x})$,

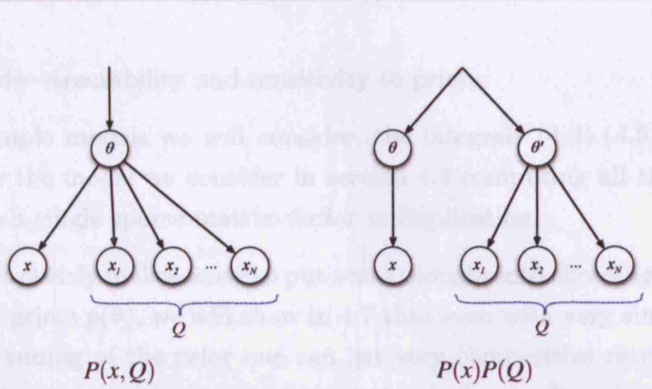


Figure 4.1: Our Bayesian score compares the hypotheses that the data was generated by each of the above graphical models.

which is the probability of observing the query set given $\mathbf{x}$.

From the above discussion, it is still not clear how one would compute quantities such as $p(\mathbf{x}|\mathcal{Q})$ and $p(\mathbf{x})$. A natural model-based way of defining a cluster is to assume that the data points in the cluster all come independently and identically distributed from some simple parameterized statistical model. Assume that the parameterized model is $p(\mathbf{x}|\theta)$ where θ are the parameters. If the data points in $\mathcal{Q}$ all belong to one cluster, then under this definition they were generated from the same setting of the parameters; however, that setting is unknown, so we need to average over possible parameter values weighted by some prior density on parameter values, $p(\theta)$. Using these considerations and the basic rules of probability we arrive at:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\theta)\, p(\theta)\, d\theta \tag{4.3}$$

$$p(\mathcal{Q}) = \int \prod_{\mathbf{x}_i \in \mathcal{Q}} p(\mathbf{x}_i|\theta)\, p(\theta)\, d\theta \tag{4.4}$$

$$p(\mathbf{x}|\mathcal{Q}) = \int p(\mathbf{x}|\theta)\, p(\theta|\mathcal{Q})\, d\theta \tag{4.5}$$

$$p(\theta|\mathcal{Q}) = \frac{p(\mathcal{Q}|\theta)\, p(\theta)}{p(\mathcal{Q})} \tag{4.6}$$

We are now fully equipped to describe the "Bayesian Sets" algorithm:

Algorithm 4.1 Bayesian Sets Algorithm

```
background: a set of items $\mathcal{D}$, a probabilistic model $p(\mathbf{x}|\theta)$ where
    $\mathbf{x} \in \mathcal{D}$, a prior on the model parameters $p(\theta)$
input: a query $\mathcal{Q} = \{\mathbf{x}_i\} \subset \mathcal{D}$
for all $\mathbf{x} \in \mathcal{D}$ do
    compute    $\text{score}(\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{Q})}{p(\mathbf{x})}$
end for
output: return elements of $\mathcal{D}$ sorted by decreasing score
```

We mention two properties of this algorithm to assuage two common worries with

Bayesian methods—tractability and sensitivity to priors:

1. For the simple models we will consider, the integrals (4.3)-(4.5) are analytical. In fact, for the model we consider in section 4.4 computing all the scores can be reduced to a single sparse matrix-vector multiplication.

2. Although it clearly makes sense to put some thought into choosing sensible models $p(\mathbf{x}|\theta)$ and priors $p(\theta)$, we will show in 4.7 that even with very simple models and almost no tuning of the prior one can get very competitive retrieval results. In practice, we use a simple empirical heuristic which sets the prior to be vague but centered on the mean of the data in $\mathcal{D}$ (a scale factor κ is used on the data mean m).

4.4 Bayesian Sets and Sparse Binary Data

We now derive in more detail the application of the Bayesian Sets algorithm to sparse binary data. This type of data is a very natural representation for the large datasets we used in our evaluations (section 4.7). Applications of Bayesian Sets to other forms of data (real-valued, discrete, ordinal, strings) are also possible, and especially practical if the statistical model is a member of the exponential family (section 4.6).

Assume each item $\mathbf{x}_i \in \mathcal{Q}$ is a binary vector $\mathbf{x}_i = (x_{i1}, \ldots, x_{iJ})$ where $x_{ij} \in \{0, 1\}$, and that each element of $\mathbf{x}_i$ has an independent Bernoulli distribution:

$$p(\mathbf{x}_i|\theta) = \prod_{j=1}^{J} \theta_j^{x_{ij}} (1 - \theta_j)^{1-x_{ij}} \tag{4.7}$$

The conjugate prior for the parameters of a Bernoulli distribution is the Beta distribution:

$$p(\theta|\alpha,\beta) = \prod_{j=1}^{J} \frac{\Gamma(\alpha_j + \beta_j)}{\Gamma(\alpha_j)\Gamma(\beta_j)} \theta_j^{\alpha_j - 1} (1 - \theta_j)^{\beta_j - 1} \tag{4.8}$$

where α and β are hyperparameters, and the Gamma function is a generalization of the factorial function. For a query $\mathcal{Q} = \{\mathbf{x}_i\}$ consisting of N vectors it is easy to show that:

$$p(\mathcal{Q}|\alpha,\beta) = \prod_j \frac{\Gamma(\alpha_j + \beta_j)}{\Gamma(\alpha_j)\Gamma(\beta_j)} \frac{\Gamma(\tilde{\alpha}_j)\Gamma(\tilde{\beta}_j)}{\Gamma(\tilde{\alpha}_j + \tilde{\beta}_j)} \tag{4.9}$$

where:

$$\tilde{\alpha}_j = \alpha_j + \sum_{i=1}^{N} x_{ij}$$

and

$$\tilde{\beta}_j = \beta_j + N - \sum_{i=1}^{N} x_{ij}$$

For an item $\mathbf{x} = (x_{\cdot 1} \ldots x_{\cdot J})$ the score, written with the hyperparameters explicit, can be computed as follows:

$$\text{score}(\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{Q}, \alpha, \beta)}{p(\mathbf{x}|\alpha, \beta)} = \prod_j \frac{\frac{\Gamma(\alpha_j+\beta_j+N)}{\Gamma(\alpha_j+\beta_j+N+1)} \frac{\Gamma(\tilde{\alpha}_j+x_{\cdot j})\Gamma(\tilde{\beta}_j+1-x_{\cdot j})}{\Gamma(\tilde{\alpha}_j)\Gamma(\tilde{\beta}_j)}}{\frac{\Gamma(\alpha_j+\beta_j)}{\Gamma(\alpha_j+\beta_j+1)} \frac{\Gamma(\alpha_j+x_{\cdot j})\Gamma(\beta_j+1-x_{\cdot j})}{\Gamma(\alpha_j)\Gamma(\beta_j)}} \tag{4.10}$$

This daunting expression can be dramatically simplified. We use the fact that $\Gamma(x) = (x-1)\,\Gamma(x-1)$ for $x > 1$. For each j we can consider the two cases $x_{\cdot j} = 0$ and $x_{\cdot j} = 1$ separately. For $x_{\cdot j} = 1$ we have a contribution $\frac{\alpha_j+\beta_j}{\alpha_j+\beta_j+N} \frac{\tilde{\alpha}_j}{\alpha_j}$. For $x_{\cdot j} = 0$ we have a contribution $\frac{\alpha_j+\beta_j}{\alpha_j+\beta_j+N} \frac{\tilde{\beta}_j}{\beta_j}$. Putting these together we get:

$$\text{score}(\mathbf{x}) = \prod_j \frac{\alpha_j + \beta_j}{\alpha_j + \beta_j + N} \left(\frac{\tilde{\alpha}_j}{\alpha_j}\right)^{x_{\cdot j}} \left(\frac{\tilde{\beta}_j}{\beta_j}\right)^{1-x_{\cdot j}} \tag{4.11}$$

The log of the score is *linear* in $\mathbf{x}$:

$$\log \text{score}(\mathbf{x}) = c + \sum_j q_j x_{\cdot j} \tag{4.12}$$

where

$$c = \sum_j \log(\alpha_j + \beta_j) - \log(\alpha_j + \beta_j + N) + \log \tilde{\beta}_j - \log \beta_j \tag{4.13}$$

and

$$q_j = \log \tilde{\alpha}_j - \log \alpha_j - \log \tilde{\beta}_j + \log \beta_j \tag{4.14}$$

If we put the entire data set $\mathcal{D}$ into one large matrix $\mathbf{X}$ with J columns, we can compute the vector $\mathbf{s}$ of log scores for all points using a single matrix vector multiplication

$$\mathbf{s} = c + \mathbf{X}\mathbf{q} \tag{4.15}$$

For sparse data sets this linear operation can be implemented very efficiently. Each query $\mathcal{Q}$ corresponds to computing the vector $\mathbf{q}$ and scalar c. This can also be done efficiently if the query is also sparse, since most elements of $\mathbf{q}$ will equal $\log \beta_j - \log(\beta_j + N)$ which is independent of the query.

4.5 Discussion of Implicit Feature Selection

We can analyze the vector $\mathbf{q}$, which is computed using the query set of items, to see that our algorithm implicitly performs feature selection. We can rewrite equation (4.14) as follows:

$$q_j = \log \frac{\tilde{\alpha}_j}{\alpha_j} - \log \frac{\tilde{\beta}_j}{\beta_j} = \log\left(1 + \frac{\sum_i x_{ij}}{\alpha_j}\right) - \log\left(1 + \frac{N - \sum_i x_{ij}}{\beta_j}\right) \tag{4.16}$$

If the data is sparse and α_j and β_j are proportional to the data mean number of ones and zeros respectively, then the first term dominates, and feature j gets weight approximately:

$$q_j \approx \log\left(1 + \text{const}\frac{\text{querymean}_j}{\text{datamean}_j}\right) \tag{4.17}$$

when that feature appears in the query set, and a relatively small negative weight otherwise. Here $\text{querymean}_j = \frac{1}{N}\sum_{i\in\mathcal{Q}} x_{ij}$. A feature which is frequent in the query set but infrequent in the overall data will have high weight. So, a new item which has a feature (value 1) which is frequent in the query set will typically receive a higher score, but having a feature which is infrequent (or not present in) the query set lowers its score.

4.6 Exponential Families

We generalize the result from section 4.4 to models in the exponential family. The distribution for such models can be written in the form $p(\mathbf{x}|\theta) = f(\mathbf{x})g(\theta)\exp\{\theta^\top \boldsymbol{u}(\mathbf{x})\}$, where $\boldsymbol{u}(\mathbf{x})$ is a K-dimensional vector of sufficient statistics, θ are the natural parameters, and f and g are non-negative functions. The conjugate prior is $p(\theta|\eta, \boldsymbol{\nu}) = h(\eta, \boldsymbol{\nu})g(\theta)^\eta \exp\{\theta^\top \boldsymbol{\nu}\}$, where η and $\boldsymbol{\nu}$ are hyperparameters, and h normalizes the distribution.

Given a query $\mathcal{Q} = \{\mathbf{x}_i\}$ with N items, and a candidate $\mathbf{x}$, it is not hard to show that the score for the candidate is:

$$\text{score}(\mathbf{x}) = \frac{h(\eta+1, \boldsymbol{\nu}+\boldsymbol{u}(\mathbf{x}))\, h(\eta+N, \boldsymbol{\nu}+\sum_i \boldsymbol{u}(\mathbf{x}_i))}{h(\eta, \boldsymbol{\nu})\, h(\eta+N+1, \boldsymbol{\nu}+\boldsymbol{u}(\mathbf{x})+\sum_i \boldsymbol{u}(\mathbf{x}_i))} \tag{4.18}$$

This expression helps us understand when the score can be computed efficiently. First of all, the score only depends on the size of the query (N), the sufficient statistics computed from each candidate, and from the whole query. It therefore makes sense to precompute $\mathbf{U}$, a matrix of sufficient statistics corresponding to $\mathbf{X}$. Second, whether the score is a linear operation on $\mathbf{U}$ depends on whether $\log h$ is linear in the second argument. This is the case for the Bernoulli distribution, but not for all exponential family distributions. However, for many distributions, such as diagonal covariance Gaussians, even though the score is nonlinear in $\mathbf{U}$, it can be computed by applying the nonlinearity elementwise to $\mathbf{U}$. For sparse matrices, the score can therefore still be computed in time linear in the number of non-zero elements of $\mathbf{U}$.

4.7 Results

We have built four systems which directly apply the Bayesian retrieval principles described here to diverse problem domains. These applications illustrate the wide range of problems for which our paradigm of retrieval by examples and our methodology provide a powerful new approach to finding relevant information. We show sample results from these five applications. When setting the hyperparameters we found little sensitivity to the value of the scale factor κ but order of magnitude changes across applications did sometimes make a difference.

(1) Retrieving movies from a database of movie preferences.

When deciding what movies to watch, books to read, or music to listen to, we often want to find movies, books or music which are similar to some set of favorites. We used our retrieval paradigm on a database of the movie preferences of 1813 users of an online service. Each of the 1532 movies in this dataset was represented by a binary vector of length equal to the number of users, where a 1 in the vector indicates that that user liked the movie. We can retrieve additional movies by giving a query consisting of a set of movies, which may represent our favorite or least favorite movies, a particular genre or some other unspecified category. For example, when given the query "Casablanca" and "Gone with the Wind", the system retrieves other movies in this classic romantic genre (e.g. "African Queen", Table 4.2). The movie retrieval results are given by our method and by Google Sets , an online set retrieval system provided by Google , which does query-by-example for items represented by words or phrases in list structures on the internet. Another set of results from a query of children's movies is also given in table 4.3. These results were evaluated by human judges, who showed a preference for the Bayesian retrieval results 4.4. Since Google Sets is a publicly available tool which can be used to query sets of movies, we feel that it is useful to make this comparison, but an interpretation of the results should take into account that the two methods are obviously not based on the same data.

(2) Finding sets of researchers who work on similar topics based on their publications.

The space of authors (of literary works, scientific papers, or web pages) can be searched in order to discover groups of people who have written on similar themes, worked in related research areas, or share common interests or hobbies. We used a database of papers presented at the Neural Information Processing Systems (NIPS) conference between 1987 and 1999 to retrieve authors who wrote on similar topics. Each of the 2037 authors is represented by a binary vector corresponding to the distribution of words they used in their NIPS papers. When given the query "Alex Smola" and "Bernhard Schoelkopf", the system retrieves authors who also presented NIPS papers on the topic of Support Vector Machines (Table 4.5). Additional author queries are given in table 4.6

(3) Searching scientific literature for clusters of similar papers.

Our method provides a novel paradigm for searching scientific literature. Instead of providing keywords, one can search by example: a small set of relevant papers can capture the subject matter in a much richer way than a few keywords. We implemented a system for searching the NIPS database in this manner. Each paper is represented by its distribution over words. A query consisting of two papers on Gaussian processes—"Bayesian Model Selection for Support Vector Machines, Gaussian Processes and Other Kernel Classifiers" and " Predictive Approaches for Choosing Hyperparameters in Gaussian Processes"—results in other papers on this topic, including "Computing with Infinite Networks" which is in fact a paper about Gaussian Processes despite the title (Table 4.7). Another query consisting of papers on perceptron learning is given in table 4.8.

(4) Searching a protein database.

Recent efforts in creating large-scale annotated genomic and proteomic databases have resulted in powerful resources for drug discovery, biomedicine, and biology. These databases are typically searched using keywords or via specialized sequence matching algorithms. Using our Bayesian retrieval method, we have created an entirely novel approach to searching UniProt (Universal Protein Resource), the "world's most comprehensive catalog of information on proteins" . Each protein is represented by a feature vector derived from GO (Gene Ontology) annotations, PDB (Protein Data Bank) structural information, keyword annotations, and primary sequence information. The user can query the database using by giving the names of a few proteins, which for example she knows share some biological properties, and our system will return a list of other proteins which, based on their features, are deemed likely to share these biological properties. Since the features include annotations, sequence, and structure information, the matches returned by the system incorporate vastly more information than that of a typical text query, and can therefore tease out more complex relationships. We queried our system with two hypothetical proteins, "METC_YEAST" and "YHR2_YEAST", which are predicted in a separate database (COG; Clusters of Orthologous Groups (Tatusov et al., 1997)) to have the functional category "Cystathionine beta-lyases/cystathionine gamma-synthases." Interestingly, our system retrieves other proteins from UniProt which fit into the same functional category, e.g. "CYS3_YEAST" (Table 4.1 (Left)). The retrieved protein "MET17_YEAST" is also in the same COG category, even though its entry in the UniProt database does not explicitly mention this function; finding such matches using keywords would be difficult. Another example of protein retrieval is given in table 4.1 (Right).

Although the evaluations in this chapter are qualitative, additional quantitative results are presented in chapter 5.

Query: METC_YEAST, HR2_YEAST	
Proteins	Top Features
METC_YEAST	GO:0006878
HR2_YEAST	methionine biosynthesis
METC_ECOLI	pyridoxal phosphate
METC_ARATH	CWP
CYS3_YEAST	ECW
CGL_HUMAN	lyase
MET17_YEAST	WTH
CGL_MOUSE	amino-acid biosynthesis
STR3_YEAST	YAW
METB_ECOLI	YWV

Query: 1433E_DROME 1433F_HUMAN 1433F_MOUSE
Top Members
1433F_HUMAN
1433F_MOUSE
1433E_DROME
1433G_MOUSE
1433G_RAT
1433G_HUMAN
1433G_BOVIN
1433G_SHEEP
1433Z_DROME
1433Z_MOUSE

Table 4.1: Results from UniProt query: (Left) The first column lists the top 10 proteins returned for the given query which consists of two hypothetical proteins belonging to the COG functional category "Cystathionine beta-lyases/cystathionine gamma-synthases". The returned proteins share this function. The second column lists the top 10 features (GO annotation, keywords, and sequence 3-mers) for this entire group of proteins. (Right) Top 10 proteins returned for a query consisting of three proteins from the 14-3-3 family.

Query: Gone with the wind, casablanca	
Google Sets	Bayesian Retrieval
casablanca (1942)	gone with the wind (1939)
gone with the wind (1939)	casablanca (1942)
ernest saves christmas (1988)	the african queen (1951)
citizen kane (1941)	the philadelphia story (1940)
pet detective (1994)	my fair lady (1964)
vacation (1983)	the adventures of robin hood (1938)
wizard of oz (1939)	the maltese falcon (1941)
the godfather (1972)	rebecca (1940)
lawrence of arabia (1962)	singing in the rain (1952)
on the waterfront (1954)	it happened one night (1934)

Table 4.2: Movies found by Google Sets and our Bayesian retrieval method when queried with "Gone with the Wind" and "Casablanca". The top 10 movies retrieved are shown for both methods. Our Bayesian retrieval method returns mainly classic romances, while Google Sets returns some classic movies, but other probably don't qualify (like "Ernest Saves Christmas").

Query: Mary Poppins, Toy Story	
Google Sets	Bayesian Retrieval
toy story	mary poppins
mary poppins	toy story
toy story 2	winnie the pooh
moulin rouge	cinderella
the fast and the furious	the love bug
presque rien	bedknobs and broomsticks
spaced	davy crockett
but i'm a cheerleader	the parent trap
mulan	dumbo
who framed roger rabbit	the sound of music

Table 4.3: Movies found by Google Sets and our Bayesian retrieval method when queries with two children's movies, "Mary Poppins" and "Toy Story". The top 10 retrieved movies are shown for both methods. Our Bayesian retrieval method returns all children's movies, while Google Sets returns some children's movies, but many which are not (like "Moulin Rouge" or "But I'm a Cheerleader").

Query	% Prefer Bayes Retrieval	P-value
Gone with the Wind	86.7	< 0.0001
Mary Poppins	96.7	< 0.0001

Table 4.4: Both movie queries (listed by first query item in the first column) were evaluated by 30 naive human judges. We give the percentage of these judges who preferred the results given by our Bayesian retrieval algorithm as opposed to Google Sets in the second column, and the p-value rejecting the null hypothesis that Google Sets is preferable to our algorithm on that particular movie query.

Query: A.Smola, B.Schoelkopf	
Top Members	Top Words
A.Smola	vector
B.Schoelkopf	support
S.Mika	kernel
G.Ratsch	pages
R.Williamson	machines
K.Muller	quadratic
J.Weston	solve
J.Shawe-Taylor	regularization
V.Vapnik	minimizing
T.Onoda	min

Table 4.5: NIPS authors found by our method based on the query "Alex Smola" and "Bernhard Schoelkopf", who both work on Support Vector Machines. The top 10 returned authors are shown, all of whom have also worked in this area, along with the top 10 words associated with this entire group of authors.

Query: L.Saul, T.Jaakkola		Query: A.Ng, R.Sutton	
Top Members	Top Words	Top Members	Top Words
L.Saul	log	R.Sutton	decision
T.Jaakkola	likelihood	A.Ng	reinforcement
M.Rahim	models	Y.Mansour	actions
M.Jordan	mixture	B.Ravindran	rewards
N.Lawrence	conditional	D.Koller	reward
T.Jebara	probabilistic	D.Precup	start
W.Wiegerinck	expectation	C.Watkins	return
M.Meila	parameters	R.Moll	received
S.Ikeda	distribution	T.Perkins	mdp
D.Haussler	estimation	D.McAllester	selects

Table 4.6: NIPS authors found by our method based on the given queries, where the data consists of papers written between 1987 and 1999. The top 10 retrieved authors are shown for each query along with the top 10 words associated with that cluster of authors. (left) The query consisted of two people who work on probabilistic modelling and the additional authors retrieved and the words associated with the retrieved set correctly indicate this. (right) Similarly, the query consisted of two people who worked on decision making and reinforcement learnining, and the returned results accurately match this description.

4.8 Discussion

We have described an algorithm which takes a query consisting of a small set of items, and returns additional items which belong in this set. Our algorithm computes a score for each item by comparing the posterior probability of that item given the set, to the prior probability of that item. These probabilities are computed with respect to a statistical model for the data, and since the parameters of this model are unknown they are marginalized out.

For exponential family models with conjugate priors, our score can be computed exactly and efficiently. In fact, we show that for sparse binary data, scoring all items in a large data set can be accomplished using a single sparse matrix-vector multiplication. Thus, we get a very fast and practical Bayesian algorithm without needing to resort to approximate inference. For example, a sparse data set with over 2 million nonzero entries can be queried in just over 1 second on a standard laptop in Matlab.

Our method does well when compared to Google Sets in terms of set completions, demonstrating that this Bayesian criterion can be useful in realistic problem domains. One of the problems we have not yet addressed is deciding on the size of the response set. Since the scores have a probabilistic interpretation, it should be possible to find a suitable threshold to these probabilities.

The problem of retrieving sets of items is clearly relevant to many application domains. Our algorithm is very flexible in that it can be combined with a wide variety of types

Query:
• Bayesian Model Selection for Support Vector Machines, Gaussian Processes and Other Kernel Classifiers (Seeger) • Predictive Approaches for Choosing Hyperparameters in Gaussian Processes (Sundararajan and Keerthi)

Title (Authors)
1. Bayesian Model Selection for Support Vector Machines, Gaussian Processes and Other Kernel Classifiers (Seeger)
2. Predictive Approaches for Choosing Hyperparameters in Gaussian Processes (Sundararajan and Keerthi)
3. Gaussian Processes for Regression (Williams and Rasmussen)
4. Gaussian Processes for Bayesian Classification via Hybrid Monte Carlo (Barber and Williams)
5. Probabilistic Methods for Support Vector Machines (Sollich)
6. Efficient Approaches to Gaussian Process Classification (Csato, Fokoue, Opper, Schottky and Winther)
7. Discovering Hidden Features with Gaussian Processes Regression (Vivarelli and Williams)
8. Computing with Infinite Networks (Williams)
9. General Bounds on Bayes Errors for Regression with Gaussian Processes (Opper and Vivarelli)
10. Finite-Dimensional Approximation of Gaussian Processes (Ferrari-Trecate, Williams, and Opper)

Top Words
Bayesian
Covariance
Gaussian
Prior
Process
Processes
Williams
Idea
Posterior
Prediction

Table 4.7: (left) NIPS papers found by our method based on the given query of two papers on the topic of Gaussian Processes. The top 10 retrieved papers are shown, all of which are also on the topic of Gaussian Processes. (right) The top 10 words associated with this entire group of papers.

Query:

- Online Learning from Finite Training Sets: An Analytical Case Study (Sollich and Barber)
- Learning Stochastic Perceptrons Under K-Blocking Distributions (Marchand and Hadjifaradji)

Title (Authors)
1. Online Learning from Finite Training Sets: An Analytical Case Study (Sollich and Barber)
2. Learning Stochastic Perceptrons Under K-Blocking Distributions (Marchand and Hadjifaradji)
3. Online Learning from Finite Training Sets in Nonlinear Networks (Sollich and Barber)
4. Strong Unimodality and Exact Learning of Constant Depth μ-Perceptron Networks (Marchand and Hadjifaradji)
5. Learning from Queries for Maximum Information Gain in Imperfectly Learnable Problems (Sollich and Saad)
6. On Learning μ-Perceptron Networks with Binary Weights (Golea, Marchand and Hancock)
7. The Learning Dynamics of a Universal Approximator (West, Sand and Nabney)
8. Learning in Large Linear Perceptrons and Why the Thermodynamic Limit is Relevant to the Real World (Sollich)
9. Learning with Ensembles: How Overfitting Can Be Useful (Sollich and Krogh)
10. Online Learning of Dichotomies (Barkai, Seung and Somplinksky)

Table 4.8: Results from the literature search application using NIPS conference papers from volumes 0-12 (1987-1999 conferences). The query consisted of two papers related to the topic of perceptron learning. The top 10 returned papers are shown, all of which also relate to perceptron learning, and some of which are by the same authors.

of data (e.g. sequences, images, etc.) and probabilistic models. We plan to explore efficient implementations of some of these extensions. We believe that with even larger datasets the Bayesian Sets algorithm will be a very useful tool for many application areas.

Chapter 5

Content-based Image Retrieval with Bayesian Sets

In this chapter we present a Bayesian framework for content-based image retrieval which models the distribution of color and texture features within sets of related images. Given a user-specified text query (e.g. "penguins") the system first extracts a set of images, from a labelled corpus, corresponding to that query. The distribution over features of these images is used to compute the Bayesian Sets score, discussed in chapter 4, for each image in a large unlabelled corpus. Unlabelled images are then ranked using this score and the top images are returned. Since, in the case of sparse binary data, all images can be scored with a single matrix-vector multiplication, it is extremely efficient to perform image retrieval using this system. We show that our method works surprisingly well despite its simplicity and the fact that no relevance feedback is used. We compare different choices of features, and evaluate our results using human subjects.

5.1 Image Retrieval

As the number and size of image databases grows, accurate and efficient content-based image retrieval (CBIR) systems become increasingly important in business and in the everyday lives of people around the world. Accordingly, there has been a substantial amount of CBIR research, and much recent interest in using probabilistic methods for this purpose (see section 5.4 for a full discussion). Methods which boost retrieval performance by incorporating user provided relevance feedback have also been of interest.

In the remainder of this chapter we describe a novel framework for performing efficient content-based image retrieval using Bayesian statistics. Our method focuses on performing category search, though it could easily be extended to other types of searches, and does not require relevance feedback in order to perform reasonably. It also emphasizes the importance of utilizing information given by sets of images, as opposed to single image queries.

5.2 Bayesian Image Retrieval System

In our Bayesian CBIR system images are represented as binarized vectors of features. We use color and texture features to represent each image, as described in section 5.2.1, and then binarize these features across all images in a preprocessing stage, described in section 5.2.2.

Given a query input by the user, say "penguins", our Bayesian CBIR system finds all images that are annotated "penguins" in a training set. The set of feature vectors which represent these images is then used in a Bayesian retrieval algorithm (section 5.2.3) to find unlabelled images which portray penguins.

5.2.1 Features

We represent images using two types of texture features, 48 Gabor texture features and 27 Tamura texture features, and 165 color histogram features. We compute coarseness, contrast and directionality Tamura features, as in Tamura et al. (1978), for each of 9 (3x3) tiles. We apply 6 scale sensitive and 4 orientation sensitive Gabor filters to each image point and compute the mean and standard deviation of the resulting distribution of filter responses. See Howarth and Rüger (2004) for more details on computing these texture features. For the color features we compute an HSV (Hue Saturation Value) 3D histogram (Heesch et al., 2003) such that there are 8 bins for hue and 5 each for value and saturation. The lowest value bin is not partitioned into hues since they are not easy for people to distinguish.

5.2.2 Preprocessing

After the 240 dimensional feature vector is computed for each image, the feature vectors for all images in the data set are preprocessed together. The purpose of this preprocessing stage is to binarize the data in an informative way. First the skewness of each feature is calculated across the data set. If a specific feature is positively skewed, the images for which the value of that feature is above the 80th percentile assign the value '1' to that feature, the rest assign the value '0'. If the feature is negatively skewed, the images for which the value of that feature is below the 20th percentile assign the value '1', and the rest assign the value '0'. This preprocessing turns the entire image data set into a sparse binary matrix, which focuses on the features which most distinguish each image from the rest of the data set. The one-time cost for this preprocessing is a total of 108.6 seconds for 31,992 images with the 240 features described in the previous section, on a 2GHz Pentium 4 laptop.

5.2.3 Algorithm

Using the preprocessed sparse binary data, our system takes as input a user-specified text query for category search and outputs images ranked as most likely to belong to the category corresponding to the query. The algorithm our system uses to perform this task is an extension of the method for clustering on-demand presented in the previous chapter, Bayesian Sets (Ghahramani and Heller, 2005).

First the algorithm locates all images in the training set with labels that correspond to the query input. Then, using the binary feature vectors which represent the images, the algorithm uses the Bayesian Sets criterion, based on marginal likelihoods, to score each unlabelled image as to how well that unlabelled image fits in with the training images corresponding to the query. This Bayesian criterion, which was presented in chapter 4, can be expressed as follows:

$$\text{score}(\mathbf{x}^*) = \frac{p(\mathbf{x}^*, \mathcal{Q})}{p(\mathbf{x}^*)p(\mathcal{Q})} \tag{5.1}$$

where $\mathcal{Q} = \{\mathbf{x}_1, \dots \mathbf{x}_N\}$ are the training images corresponding to the query, and $\mathbf{x}^*$ is the unlabelled image that we would like to score. We use the symbol $\mathbf{x}_i$ to refer interchangably both to image i, and to the binary feature vector which represents image i. Each of the three terms in equation (5.1) are marginal likelihoods. For more details refer to chapter 4.

Algorithm 5.1 summarizes our Bayesian CBIR framework and a flowchart is given in figure 5.1.

Algorithm 5.1 Bayesian CBIR Algorithm

```
background: a set of labelled images $\mathcal{D}_\ell$, a set of unlabelled images $\mathcal{D}_u$,
  a probabilistic model $p(\mathbf{x}|\theta)$ defined on binary feature vectors representing images,
  a prior on the model parameters $p(\theta)$
compute texture and color features for each image
preprocess: Binarize feature vectors across images
input: a text query, $q$
find images corresponding to $q$, $\mathcal{Q} = \{\mathbf{x}_i\} \subset \mathcal{D}_\ell$
for all $\mathbf{x}^* \in \mathcal{D}_u$ do
  compute    $\text{score}(\mathbf{x}^*) = \frac{p(\mathbf{x}^*, \mathcal{Q})}{p(\mathbf{x}^*)p(\mathcal{Q})}$
end for
output: sorted list of top scoring images in $\mathcal{D}_u$
```

We still have not described the specific model, $p(\mathbf{x}|\theta)$, or addressed the issue of computational efficiency of computing the score. Each image $\mathbf{x}_i \in \mathcal{Q}$ is represented as a binary vector $\mathbf{x}_i = (x_{i1}, \dots, x_{iJ})$ where $x_{ij} \in \{0, 1\}$. As in chapter 4, we define a model in which each element of $\mathbf{x}_i$ has an independent Bernoulli distribution, and use a conjugate Beta prior on the model parameters.

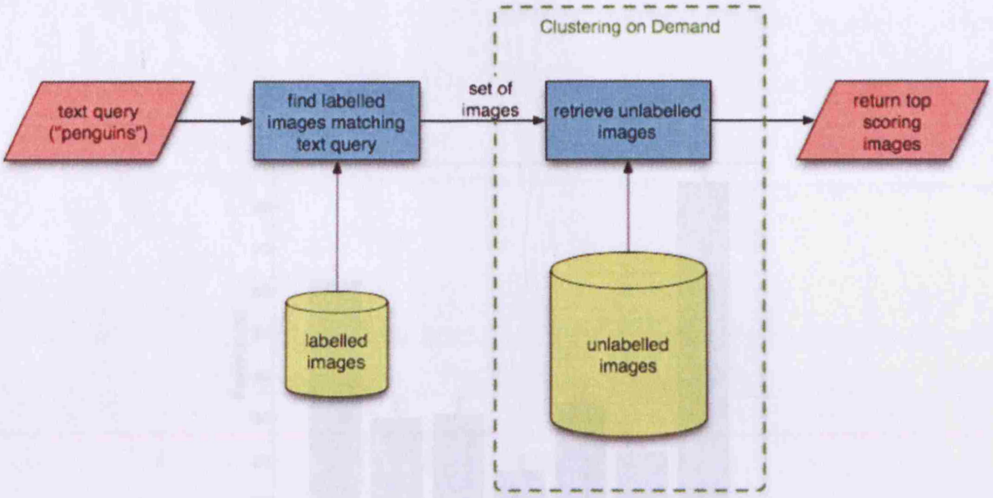

Clustering on Demand
text query ("penguins")
find labelled images matching text query
set of images
retrieve unlabelled images
return top scoring images
labelled images
unlabelled images

Figure 5.1: Flowchart for the Bayesian CBIR system

If we put the entire data set into one large matrix $\mathbf{X}$ with J columns, we can compute the vector $\mathbf{s}$ of log scores for all images using a single matrix vector multiplication

$$\mathbf{s} = c + \mathbf{X}\mathbf{q} \tag{5.2}$$

where $q_j = \log \tilde{\alpha}_j - \log \alpha_j - \log \tilde{\beta}_j + \log \beta_j$ represents the query set of images ($\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the Beta hyperparameters) and c is a query-specific additive constant which has no effect on the ranking. The full derivation of equation (5.2) is given in the previous chapter.

For our sparse binary image data, this linear operation can be implemented very efficiently. Each query $\mathcal{Q}$ corresponds to computing vector $\mathbf{q}$ and scalar c, which can be done very efficiently as well. The total retrieval time for 31,992 images with 240 features and 1.34 million nonzero elements is 0.1 to 0.15 seconds, on a 2GHz Pentium 4 laptop. We can analyze the vector $\mathbf{q}$, to understand which features our algorithm implicitly selecting, as discussed in 4.

5.3 Results

We used our Bayesian CBIR system to retrieve images from a Corel data set of 31,992 images. 10,000 of these images were used with their labels as a training set, $\mathcal{D}_\ell$, while the rest comprised the unlabelled test set, $\mathcal{D}_u$. We tried a total of 50 different queries, corresponding to 50 category searches, and returned the top 9 images retrieved for each query using both texture and color features, texture features alone, and color features alone. We used the given labels for the images in order to select the query set, $\mathcal{Q}$, out of the training set. To evaluate the quality of the labelling in the training data we

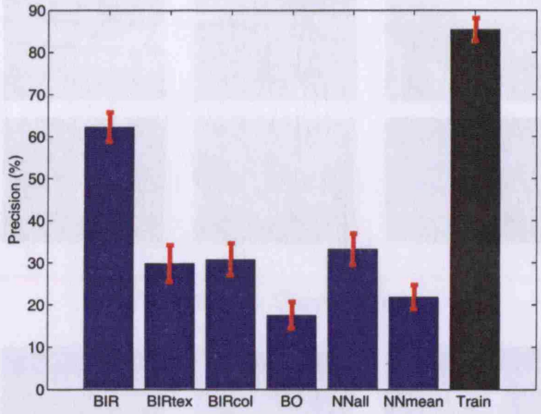


Figure 5.2: mean ± s.e. % correct retrievals over 50 queries

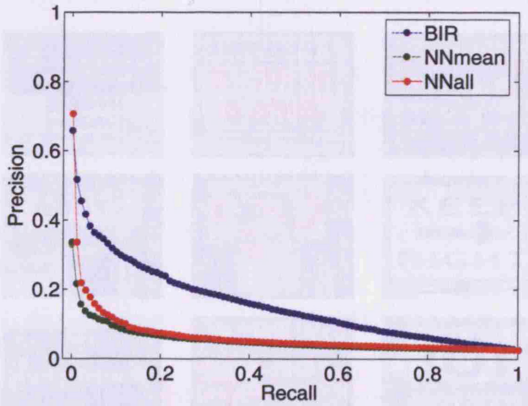


Figure 5.3: Precision-recall curves for our method (blue) and both nearest neighbor comparison methods, averaged over all 50 queries, and using the Corel data labellings

Figure 5.4: Query: desert

Figure 5.5: Query: building

Figure 5.6: Query: sign

also returned a random sample of 9 training images from this query set. In all of our experiments we set $\kappa = 2$.

Figure 5.7: Query: pet

Figure 5.8: Query: penguins

The above process resulted in 1800 images: 50 queries × 9 images × 4 sets (all features, texture features only, color features only, and sample training images). Two uninformed human subjects were then asked to label each of these 1800 images as to whether they thought each image matched the given query. We chose to compute precisions for the top nine images for each query based on factors such as ease of displaying the images, reasonable quantities for human hand labelling, and because when people perform category searches they generally care most about the first few results that are returned. We found the evaluation labellings provided by the two human subjects to be highly correlated, having correlation coefficient 0.94.

We then compared our Bayesian CBIR results on all features with the results from using two different nearest neighbor algorithms to retrieve images given the same image data set, query sets and features. The first nearest neighbor algorithm found the nine images which were closest (euclidean distance) to any individual member of the query set. This algorithm is approximately 200 times slower than our Bayesian approach. More analagous to our algorithm, the second nearest neighbor algorithm found the nine images which were closest to the mean of the query set. Lastly we compared to the Behold Image Search online . Behold Image Search online runs on a more difficult 1.2 million image dataset. We compare to the Behold system because it is a currently available online CBIR system which is fast and handles query words, and also because is was part of the inspiration for our own Bayesian CBIR system. The results given by these three algorithms were similarly evaluated by human subjects.

The results from these experiments are given in table 5.1. The first column gives the query being searched for, the second column is the number of images out of the nine images returned by our algorithm which were labelled by the human subjects as being relevant to the query (precision × 9). The third and fourth columns give the same kind of score for our system, but restricting the features used to texture only and color only, respectively. The fifth column shows the results of the Behold online system, where N/A entries correspond to queries which were not in the Behold vocabulary. The sixth and seventh columns give the results for the nearest neighbor algorithms using all members of the query set and the mean of the query set respectively. The eighth column gives the number of images out of the 9 randomly displayed training images that were labelled by our subjects as being relevant to the query. This gives an indication of the quality of the labelling in the Corel data. The last column shows the number of training images, n, which comprise the query set (i.e. they were labelled with the query word in the labellings which come with the Corel data).

Looking at the table we can notice that our algorithm using all features (BIR) performs better than either the texture features (BIRtex) or color features alone (BIRcol); the p-values for a Binomial test for texture or color features alone performing better than all features are less than 0.0001 in both cases. In fact our algorithm can do reasonably well even when there are no correct retrievals using either color or texture features alone (see, for example, the query "eiffel"). Our algorithm also substantially outperforms all three of the comparison algorithms (BO, NNmean, NNall). It tends to perform better on examples where there is more training data, although it does not always need a large amount of training data to get good retrieval results; in part this may result from the particular features we are using. Also, there are queries (for example, "desert") for which the results of our algorithm are judged by our two human subjects to be better than a selection of the images it is training on. This suggests both that the original labels for these images could be improved, and that our algorithm is quite robust to outliers and poor image examples. Lastly, our algorithm finds at least 1, and generally many more, appropriate images, in the nine retrieved images, on all of the 50 queries.

The average number of images returned, across all 50 queries, which were labelled by our subjects as belonging to that query category, are given in figure 5.2. The error bars show the standard error about the mean. Some sample images retrieved by our algorithm are shown in figures 5.5-5.8, where the queries are specified in the figure captions.

Query	BIR	BIRtex	BIRcol	BO	NNmean	NNall	Train	# Train
abstract	8	4	8	5.5	2	1	5	391
aerial	4	0.5	2	0	2	3	8	201
animal	8	5	6	1	3	9	9	1097
ape	4	1	0	0	2	7	8.5	27
boat	1	0	1	0.5	1	1	7	61
building	7.5	9	2.5	4	6	5.5	8	1207
butterfly	5	4	1	1	2	0	9	112
castle	3.5	2	2	1	0	3	8	229
cavern	5.5	1	2.5	0.5	2	1	9	34
cell	6	0	5	9	5	4	8	29
church	3.5	1	2	0	5	0	6	173
clouds	5	5.5	1.5	0	3	5	5.5	604
coast	7	3	2	1	2	2	9	299
desert	4.5	0	1	1	0	1.5	2	168
door	8.5	8	1	0	2	0	5.5	92
drawing	4	0	0	2	7	3	9	69
eiffel	6	0	0	N/A	0	0	8.5	15
fireworks	9	9	3	0	1	3	9	76
flower	9	1	7.5	2	3	1	9	331
fractal	3	0	5.5	0.5	0	2	8.5	43
fruit	5.5	0.5	6.5	0	0	1	8	199
house	6	8	0	1.5	1	2	8	184
kitchen	6	1	2	N/A	5	3	9	32
lights	6.5	3	1.5	N/A	1	0	7	203
model	5	4	0	N/A	3	4	9	102
mountain	6	1	2.5	1	2	3	8	280
mountains	7	2	8	N/A	1	3	8.5	368
penguins	6	1	5	N/A	0	0	8.5	34
people	6	2	0	1.5	4	5	8.5	239
person	4	0.5	1.5	1.5	4	5	7.5	114
pet	3	2	2	0.5	0	4.5	8.5	138
reptile	3	1	1	1	0	1	9	99
river	4.5	1.5	4.5	1.5	2	4	7	211
sea	7.5	6	3	0.5	2	3	6	90
sign	9	9	1	8	1	0	9	53
snow	6	0	4	1	2	3	9	259
stairs	3	3.5	2	0	1	2	8	53
sunset	9	7.5	4	2.5	3	2.5	8.5	187
textures	7	8	1	N/A	0	8	3	615
tool	4	1	4	1	1	5	9	28
tower	7.5	3.5	0.5	2.5	3	2.5	6	138
trees	9	1	8	N/A	6	8	8.5	1225
turtle	2	0	1	N/A	0	0	9	13
urban	7.5	4.5	2	N/A	3	3	9	133
volcano	2	0	3	0	0	0	3	54
water	9	3	5.5	0	5	9	5.5	1863
waterfall	2	0	2	1	0	3	9	103
white	9	3	9	4.5	1	6.5	7.5	240
woman	4	2	0	3	2	3	8.5	181
zebra	2	0	0	N/A	0	2	8	21

Table 5.1: Results table over 50 queries

By looking at these examples we can see where the algorithm performs well, and what the algorithm mistakenly assigns to a particular query when it does not do well. For example, when looking for "building" the algorithm occasionally finds a large vertical outdoor structure which is not a building. This gives us a sense of what features the algorithm is paying attention to, and how we might be able to improve performance through better features, more training examples, and better labelling of the training data. We also find that images which are *prototypical* of a particular query category tend to get high scores (for example, the query "sign" returns very prototypical sign images).

We also compute precision-recall curves for our algorithm and both nearest neighbor variants that we compared to (figure 5.3). For the precision-recall curves we use the labellings which come with the Corel data. Both nearest neighbor algorithms perform significantly worse than our method. NNall has a higher precision than our algorithm at the lowest level of recall. This is because there is often at least one image in the Corel test set which is basically identical to one of the training images (a common criticism of this particular data set). The precision of NNall immediately falls because there are few identical images for any one query, and generalization is poor. Our algorithm does not preferentially return these identical images (nor does NNmean), and they are usually not present in the top 9 retrieved.

Four sets of retrieved images (all features, texture only, color only, and training) for all 50 queries can be found in additional materials [1], which we encourage the reader to have a look through.

5.4 Related Work

There is a great deal of literature on content-based image retrieval. An oft cited early system developed by IBM was "Query by Image Content" (QBIC Flickner et al. (1995)). A thorough review of the state of the art until 2000 can be found in Smeulders et al. (2000).

We limit our discussion of related work to (1) CBIR methods that make use of an explicitly probabilistic or Bayesian approach, (2) CBIR methods that use sets of images in the context of relevance feedback, and (3) CBIR methods that are based on queries consisting of sets of images.

Vasconcelos and Lippman have a significant body of work developing a probabilistic approach to content-based image retrieval (e.g. Vasconcelos and Lippman (1998)). They approach the problem from the framework of classification, and use a probabilistic model of the features in each class to find the maximum a posteriori class label. In Vasconcelos (2004) the feature distribution in each class is modelled using a Gaussian

[1] `http://www.gatsby.ucl.ac.uk/~heller/BIRadd.pdf`

mixture projected down to a low dimensional space to avoid dimensionality problems. The model parameters are fit using EM for maximum likelihood estimation. Our approach differs in several respects. Firstly, we employ a fully Bayesian approach which involves treating parameters as unknown and marginalizing them out. Second, we use a simpler binarized feature model where this integral is analytic and no iterative fitting is required. Moreover, we represent each image by a single feature vector, rather than a set of query vectors. Finally, we solve a different problem in that our system starts with a text query and retrieves images from an unlabelled data set—the fact that the training images are given a large number of non-mutually exclusive annotations suggests that the classification paradigm is not appropriate for our problem.

PicHunter (Cox et al., 2000) is a Bayesian approach for handling relevance feedback in content based image retrieval. It models the uncertainty in the users' goal as a probability distribution over goals and uses this to optimally select the next set of images for presentation.

PicHunter uses a weighted pairwise distance measure to model the similarity between images, with weights chosen by maximum likelihood. This is quite different from our approach which models the joint distribution of sets of images averaging over model parameters.

Rui et al. (1997) explore using the tf-idf[2] representation from document information retrieval in the context of image retrieval. They combine this representation with a relevance feedback method which reweights the terms based on the feedback and report results on a dataset of textures. It is possible to relate tf-idf to the feature weightings obtained from probablistic models but this relation is not strong.

Yavlinsky et al. (2005) describe a system for both retrieval and annotation of images. This system is based on modeling $p(\mathbf{x}|w)$ where $\mathbf{x}$ are image features and w is some word from the annotation vocabulary. This density is modeled using a non-parameteric kernel density estimator, where the kernel uses the Earth Mover's Distance (EMD). Bayes rule is used to get $p(w|\mathbf{x})$ for annotation.

Gosselin and Cord (2004) investigate active learning approaches to efficient relevance feedback using binary classifiers to distinguish relevant and irrelevant classes. Among other methods, they compare a "Bayes classifier" which uses Parzen density estimators with a fixed-width Gaussian kernel to model $P(\mathbf{x}|\text{relevant})$ and $P(\mathbf{x}|\text{irrelevant})$ where $\mathbf{x}$ are image features. Our approach differs in several respects. First, we model the probablity of the target $\mathbf{x}$ belonging to a cluster while integrating out all parameters of the cluster, and compare this to the prior $p(\mathbf{x})$. Strictly speaking, Parzen density estimators are not Bayesian in that they do not define a prior model of the data, but rather can be thought of as frequentist smoothers for the empirical distribution of the data.

[2] term-frequency inverse-document-frequency

The combination of labeled and unlabeled data and the sequential nature of relevance feedback, mean that active learning approaches are very natural for CBIR systems. Hoi and Lyu (2005) adapt the semi-supervised active learning framework of Zhu et al. (2003) as a way of incorporating relevance feedback in image retrieval.

In Assfalg et al. (2000), the user manually specifies a query consisting of a set of positive and negative example images. The system then finds images which minimize the distance in color histogram space to the positive examples, while maximizing distance to the negative examples. While our method is not directly based on querying by examples, since it uses text input to extract images from a labelled set, it implicitly also uses a *set* of images as the query. However, in our system the set only contains positive examples, the user only has to type in some text to index this set, and the subsequent retrieval is based on different principles.

5.5 Conclusions and Future Work

We have described a new Bayesian framework for content-based image retrieval. We show the advantages of using a set of images to perform retrieval instead of a single image or plain text. We obtain good results from using the Bayesian Sets criterion, based on marginal likelihoods, to find images most likely to belong to a query category. We also show that using this criterion can be very efficient when image feature vectors are sparse and binary.

In all of our experiments, the two free parameters, the preprocessing percentile threshold for binarizing the feature vectors and the scale factor on the means for setting the hyperparameters (κ), are set to 20 and 2 respectively. In our experience, this initial choice of values seemed to work well, but it would be interesting to see how performance varies as we adjust the values of these two parameters.

In the future there are many extensions which we would like to explore. We plan to extend the system to incorporate multiple word queries where the query sets from all words in the query are combined by either taking the union or the intersection. We would also like to look into incorporating relevance feedback, developing revised query sets, in our Bayesian CBIR system. By combining with relevance feedback, the principles used here can also be applied to other types of seaches, such as searching for a specific target image. Lastly, we would like to explore using our Bayesian CBIR framework to perform automatic image annotation as well as retrieval.

Chapter 6

Analogical Reasoning with Bayesian Sets

Analogical reasoning is the ability to learn and generalize about relations between objects. Performing analogical reasoning in an automated manner is very challenging since there are a large number of ways in which objects can be related. In this chapter we develop an approach to performing automated analogical reasoning, which, given a set of pairs of objects $\mathbf{S} = \{A^1{:}B^1, A^2{:}B^2 \ldots, A^N{:}B^N\}$ where each pair in S shares the same relationship, R, measures how well other pairs $A : B$ fit in with the pairs in set $\mathbf{S}$, thereby determining how similar (or analagous) the shared relation of $A{:}B$ is to those of the pairs in set S (relation R). In this problem the degree of similarity between individual objects in the pairs is not necessarily relevant to discovering the correct analogy. For example, the analogy between an electron orbiting around the nucleus of an atom and a planet orbiting around the Sun cannot be detected by measuring the non-relational similarity between an electron and a planet or a nucleus and the Sun. We take a Bayesian approach to solving the problem of automated analogical reasoning by developing a generative model for predicting the existance of relations, extending the framework presented in chapter 4. The utility of our method is demonstrated though practical applications in exploratory data analysis[1].

6.1 Analogical Reasoning

In order to discover analogies we need to be able to define a measure of similarity between the structures of related objects, where typically we are not interested in how similar individual objects in a candidate pair are to objects in the "query" pair(s). To illustrate this we consider an analogical reasoning question from an SAT exam, where for the given "query" pair of words *water:river* the examinee must choose the one of

[1] The work in this chapter was done in collaboration with Ricardo Silva and Edo Airoldi. Some of this work has been published at the AISTATS conference (Silva et al., 2007b).

five possible pairs which is analogous (i.e. its relation best matches) to the query pair. In this case the word pair *car:traffic* would be a better choice than *soda:ocean* since the relationship they share is the same (cars flow through traffic in the same way water does through a river). This is the case regardless of the fact that water is more similar to soda than to car, illustrating that the similarity of individual objects is not necessarily meaningful. Therefore, in analogical reasoning features of objects are only themselves meaningful in the context of predicting relations.

The specific type of problem which we address in this chapter can be illustrated through the following two examples from Silva et al. (2007b):

Example 1 A researcher has a large collection of papers. She plans to use such a database in order to write an comprehensive article about the evolution of her field through the past two decades. In particular, she has a collection organized as pairs of papers where one cites the other. There are several reasons why a paper might cite another: one of them is a big bibliographic survey, or one paper was written by the advisor of the author of the second paper, or the cited paper was given a best paper award, or the authors were geographically close, or a combination of several features of such papers with different likelihoods. Such combinations define a (potentially very large) variety of ***subpopulations*** of pairs of papers. While the features that imply such subpopulations might be assumed to be recorded in the database, the way such groups are defined is never explicitly indicated in the data. Yet the researcher is not completely in the dark: she already has an idea of important subgroups of pairs of papers which are representative of the most interesting subpopulations, although it might be difficult to characterize any such set with a simple description. She, however, would like to know which other pairs of papers might belong to such subgroups. Instead of worrying about writing some simple query rules that explain the common properties of such subgroups, she would rather have an intelligent information retrieval system that is able to identify which other pairs in the database are linked in an analogous way to those pairs in her selected sets. □

Example 2 A scientist is investigating a population of proteins, within which some pairs are known to interact, while the remaining pairs are known not to interact. It is known that recorded gene expression profiles of the respective genes can be used as a reasonable predictor of the existence or not of an interaction. The current state of knowledge is still limited regarding which subpopulations (i.e., classes) of interactions exist, although a partial hierarchy of such classes for some proteins is available. Given a selected set of interacting proteins that are believed to belong to a particular level of the class hierachy, the researcher would like to query her database to discover other plausible pairs of proteins whose mechanism of linkage is of the same nature as in the selected set, i.e., to query for analogous relations. Ideally, she would like to do it without being required to write down query rules that explicitly describe the selected set. □

These are practical analogical reasoning problems which require performing information retrieval for exploratory data analysis. As in chapter 4, the number of classes of interest is not known a priori and may be quite large, and specifying representative negative examples by hand is generally impractical.

6.2 Related Work

There is a large literature on analogical reasoning in artificial intelligence and psychology which is discussed in this section, from Silva et al. (2007b). We refer to French (2002) for a recent survey, as well as to some recent work in the machine learning literature (Marx et al., 2002; Turney and Littman, 2005). Other authors have benefited from the idea of exploring similarity of relations for other problems such as dimensionality reduction (Memisevic and Hinton, 2005). Here we will use a Bayesian framework for inferring similarity of relations. Given a set of relations, our goal will be to score other relations as relevant or not. The score is based on a Bayesian model comparison generalizing the "Bayesian sets" score (Ghahramani and Heller, 2005) to discriminative models over pairs of objects.

The graphical model formulation of Getoor et al. (2002) incorporates models of link existence in relational databases, an idea used explicitly in Section 6.3 as the first step of our problem formulation. In the clustering literature, the probabilistic approach of Kemp et al. (2006) is motivated by principles similar to those in our formulation: the idea is that there is an infinite mixture of subpopulations that generates the observed relations. Our problem, however, is to retrieve other elements of a subpopulation described by elements of a query set, a goal that is also closer to the classical paradigm of analogical reasoning. A more detailed comparison of block models and our formulation is presented in the next section.

Our focus here is not on predicting the presence or absence of relational links, as in, e.g., (Popescul and Ungar, 2003) – a very hard task, due to the large discrepancy of the number of positive and negative examples – but rather on retrieving similar relational links from among those already known to exist in the relational database. Neither is our focus to provide a fully unsupervised clustering of the whole database of pairs (as in, e.g., Kemp et al., 2006), or to use relational information to improve classification of other attributes (as in, e.g., Getoor et al., 2002).

6.3 Automated Analogical Reasoning using Discriminative Bayesian Sets

In order to determine if a relation A:B is analagous to set $\mathbf{S} = \{A^1{:}B^1, A^2{:}B^2 \ldots A^N{:}B^N\}$ we need to define a measure of similarity between a pair $(A : B)$ and a set of pairs $(\mathbf{S})$. The meausre of similarity that we are interested in does not directly compare the information contained in the distributions of the objects $\{A^i\} \subset \mathcal{A}, \{B^i\} \subset \mathcal{B}$, where $\mathcal{A}$ and $\mathcal{B}$ are object classes, but rather compares the relations, or *mappings* which classify the pairs as being linked (i.e. the mappings which discriminate the particular relationship shared by each of the pairs).

In order to formulate this concept more formally, we can consider a space of latent functions in $\mathcal{A} \times \mathcal{B} \to \{0, 1\}$. We assume that A and B are two objects which are classified as linked by some unknown function $f(A, B)$ (i.e. $f(A, B) = 1$). Our goal is to measure the similarity between the function $f(A, B)$ and the function $g(\cdot, \cdot)$, where $g(\cdot, \cdot)$ classifies all pairs $(A^i, B^i) \in \mathbf{S}$ as being linked (i.e. $g(A^i, B^i) = 1$). The similarity measure we employ is a function of our observations $(\mathbf{S}, A, B)$ and prior distributions over $f(\cdot, \cdot)$ and $g(\cdot, \cdot)$.

We define our similarity measure as computing a Bayes factor, which involves integrating over the function space of latent functions $f(\cdot)$ and $g(\cdot)$. In the remainder of this chapter we restrict ourselves to the case where our family of latent functions is parameterized by a finite-dimensional vector, specifically the logistic regression function with multivariate Gaussian priors for the parameters. This is not a necessary restriction, this framework holds for arbitrary functions, but one done for convenience.

Given a pair $(A^i \in \mathcal{A}, B^j \in \mathcal{B})$, we define $X^{ij} = [\phi_1(A^i, B^j)\phi_2(A^i, B^j) \dots \phi_k(A^i, B^j)]^\top$ to be a point in feature space given by the mapping $\phi : (\mathcal{A} \times \mathcal{B}) \to \mathcal{R}^K$. We also define $C^{ij} \in \{0, 1\}$ to be an indicator of a link (or relation) between A^i and B^j. $\Theta = [\theta_1, \dots, \theta_K]^\top$ is the parameter vector for the logistic regression model:

$$p(C^{ij} = 1 | X^{ij}, \Theta) = \text{logistic}(\Theta^T X^{ij}) \tag{6.1}$$

where $\text{logistic}(x) = (1 + e^{-x})^{-1}$. The similarity measure we use to score a pair (A^i, B^j) with respect to a query set $\mathbf{S}$ is the Bayes factor given in chapter 4:

$$score(A^i, B^j) = \frac{p(C^{ij} = 1 | X^{ij}, \mathbf{S}, \mathbf{C}^{\mathbf{S}} = 1)}{p(C^{ij} = 1 | X^{ij})} \tag{6.2}$$

where $\mathbf{C}^{\mathbf{S}}$ is the vector of link indicators for $\mathbf{S}$ (figure 6.2). We compute:

$$p(C^{ij} = 1 | X^{ij}, \mathbf{S}, \mathbf{C}^{\mathbf{S}} = 1) = \int P(C^{ij} = 1 | X^{ij}, \Theta) P(\Theta | \mathbf{S}, \mathbf{C}^{\mathbf{S}} = 1) \, d\Theta \tag{6.3}$$

in order to calculate the score for each pair. $p(C^{ij} = 1 | X^{ij})$ is computed analogously by integrating over the model parameters Θ. Since these integrals do not have a closed form solution, we use the Bayesian variational approximation by Jaakkola and Jordan (2000) in practice. An intuition behind this score is that we are comparing the hyposthesis that the query set $\mathbf{S}$ and the pair being scored, are classified as linked by the same logistic

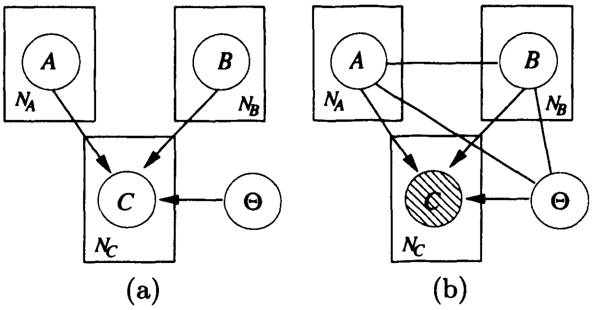


Figure 6.1: (a) Graphical model for the relational Bayesian logistic regression, where N_A, N_B and N_C are the number of objects of each class. (b) Extra dependencies induced by further conditioning on C are represented by undirected edges.

regression function, with the same set of parameters, to the hypothesis that they are classified as linked by different logistic regression functions. If pairs are classified as linked by the same logistic regression function, then they share the same relationship.

In the work we present in this chapter we assume that we are given a relational database $(\mathbf{D_A}, \mathbf{D_B}, \mathbf{R_{AB}})$, where $\mathbf{D_A}$ and $\mathbf{D_B}$ are objects sampled from the object classes $\mathcal{A}$ and $\mathcal{B}$ respectively, and $\mathbf{R_{AB}}$ is a binary matrix which represents the existence of a relationship (or link) between objects A and B. This binary matrix is assumed to have been generated from a logistic regression model for relation existence.

The algorithm we use for retrieving relations is as follows:

Algorithm 6.1 Analogical Reasoning Algorithm

```
background: sets of items $\mathbf{D_A}$ and $\mathbf{D_B}$ and a binary relationship matrix $\mathbf{R_{AB}}$, the logistic regression model $p(C^{ij}|X^{ij}, \Theta)$, and a Gaussian prior on the model parameters $p(\Theta)$
input: a query set of pairs S which share some relationship
for all $(A^i, B^j)$ where $R^{ij} = 1$ do
    compute    $score(A^i, B^j)$
end for
output: return pairs $(A^i, B^j)$ sorted by decreasing score
```

The graphical model for the relational Bayesian logistic regression is given in figure 6.1(a). The latent parameter vector Θ and objects A and B are parents of the relation indicator variable C. When we condition on $C = 1$ we introduce dependencies between the elements of vector Θ and the objects $\{A, B\}$ (shown in figure 6.1(b)). Due to this information about a pair (A, B) can be passed on through Θ and used to evaluate other pairs. The method presented here has a computational complexity of $O(K^3)$ due to a matrix inversion necessary for the variational Bayesian logistic regression (Jaakkola and Jordan, 2000).

Our framework assumes that feature space ϕ encapsulates enough information to allow for a reasonable classifier to predict the existence of relations. It can also be used for

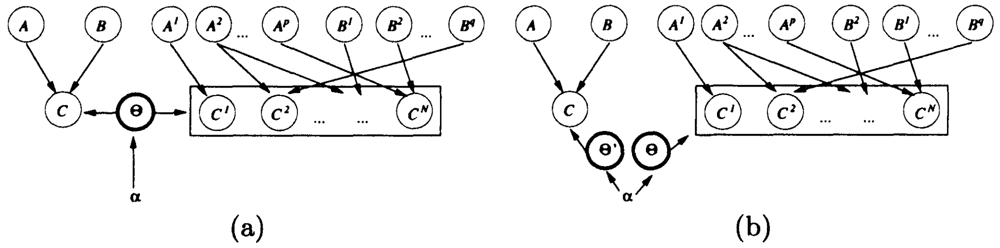


Figure 6.2: The score of a new data point $\{A, B, C\}$ is given by the Bayes factor that compares models (a) and (b). α are the hyperparameters for Θ. In (a), the generative model is the same for both the point being scored and the query set (which is represented in the rectangle). Notice that our set $\mathbf{S}$ of pairs in $\{A^i\} \times \{B^j\}$ might contain repeated instances of the same object (some A^i or B^j might appear multiple times in different relations). In (b), the point being scored and the query set do not share the same parameters.

solving non-relational problems with a variety of classifiers, instead of modeling the presence and absence of interactions between objects as we have presented here. There are algorithms for automatically selecting useful predictive features which can be used with this method (Popescul and Ungar, 2003; Džeroski and Lavrač, 2001). Jensen and Neville (2002) also discusses some shortcomings of these automated feature selection methods for relational classification.

We also assume that all subpopulations of relations of interest can be measured on the same feature space. Although a very general formulation would not require this, in order for the problem to be well-defined feature spaces must relate to each other somehow. Experimenting with using a hierarchical Bayesian formulation to link different feature spaces is an area for future research. Also, this method can easily be extended to deal with *continuous* measures of relationship (instead of using a binary indicator variable C).

We set our priors in an analogous way to Bayesian Sets (chapter 4) where the prior is set empirically from the observed data. We use the prior $P(\Theta) = \mathcal{N}(\widehat{\Theta}, (c\widehat{\mathbf{T}})^{-1})$, where $\mathcal{N}(\mathbf{m}, \mathbf{V})$ is a Normal distribution with mean $\mathbf{m}$ and variance $\mathbf{V}$, and $\widehat{\Theta}$ is the maximum likelihood estimator of Θ computed on a subset of both positive and negative examples. Matrix $\widehat{\mathbf{T}}$ is the empirical second moments matrix of the linked object features in $\mathbf{X}$, a measure of their variability. Constant c is a smoothing parameter set by the user. In our experiments, we selected it to be twice the total number of relations.

6.4 Results

We now describe two experiments on analogical retrieval using the proposed model, and taken from Silva et al. (2007b). Evaluation of the significance of retrieved items often relies on subjective assessments (Ghahramani and Heller, 2005). To simplify our study, we will focus on particular setups where objective measures of success can be

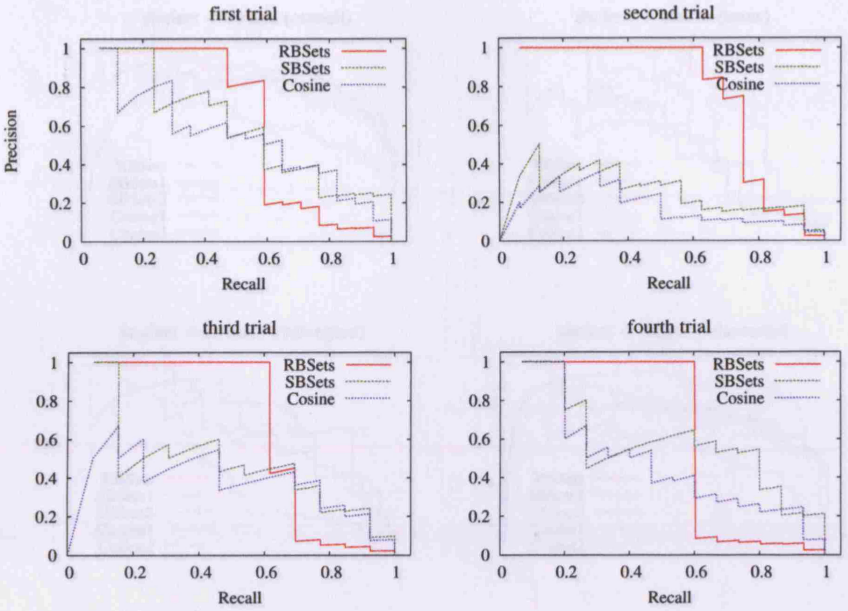


Figure 6.3: Precision/recall curves for four different random queries of size 10 for the three algorithms: relational Bayesian sets (RBSETS), regular Bayesian sets with Bernoulli model (SBSETS-B) and cosine distance.

derived.

Our main standard of comparison will be a "flattened Bayesian sets" algorithm (which we will call "standard Bayesian sets," SBSETS, in constrast to the relational model, RBSETS). Using a multivariate independent Bernoulli model as in chapter 4, we join linked pairs into single rows, and then apply the original algorithm directly on this joined data. This algorithm serves the purpose of both measuring the loss of not treating relational data as such, as well as the limitations of treating similarity of pairs through the generative models of $\mathcal{A}$ and $\mathcal{B}$ instead of the generative model for the latent predictive function $g(\cdot, \cdot)$.

In both experiments, objects are of the same type, and therefore, dimensionality. The feature vector X^{ij} for each pair of objects $\{A^i, B^j\}$ consists of the V features for object A^i, the V features of object B^j, and measures $\{Z_1, \dots, Z_V\}$, where $Z_v = (A^i_v \times B^j_v)/(\|A^i\| \times \|B^j\|)$, $\|A^i\|$ being the Euclidean norm of the V-dimensional representation of A^i. We also use a constant value (1) as part of the feature set as an intercept term for the logistic regression. The $\mathbf{Z}$ features are exactly the ones used in the cosine distance measure, a common and practical measure widely used in information retrieval (Manning et al., 2007). They also have the important advantage of scaling well with the number of variables in the database. Moreover, adopting such features will make our comparisons in the next sections more fair, since we evaluate how well cosine distance performs in our task. Notice X^{ij} represents asymmetric relationships as required in our applications. For symmetric relationships, features such as $|A^i_v - B^j_v|$ could be used

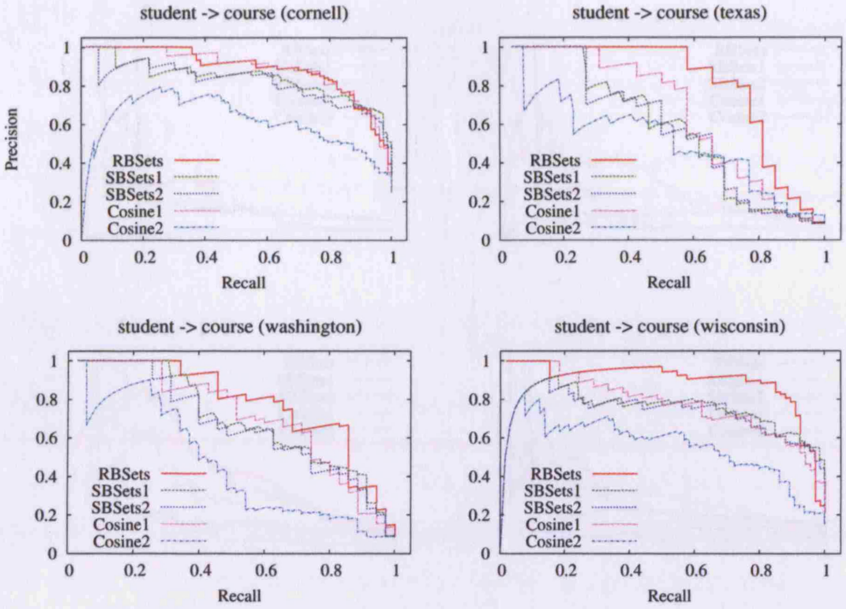


Figure 6.4: Results for *student* → *course* relationships.

instead.

6.4.1 Synthetic experiment

We first discuss a synthetic experiment where there is a known ground truth. We generate data from a simulated model with six classes of relations represented by six different instatiations of Θ, $\{\Theta_0, \Theta_1, \dots, \Theta_5\}$. This simplified setup defines a multiclass logistic softmax classifier that outputs a class label out of $\{0, 1, \dots, 5\}$. Object spaces $\mathcal{A}$ and $\mathcal{B}$ are the same, and defined by a multivariate Bernoulli distribution of 20 dimensions, where each attribute has independently a probability 1/2 of being 1. We generate 500 objects, and considered all 500^2 pairs to generate $250,000$ feature vectors X. For each X we evaluate our logistic classifier to generate a class label. If this class is zero, we label the corresponding pair as "unlinked." Otherwise, we label it as "linked." The intercept parameter for parameter vector Θ_0 was set manually to make class 0 appear in at least 99% the data[2], thus corresponding to the usual sparse matrices found in relational data.

The algorithms we evaluate do not know which of the 5 classes the linked pairs originally corresponded to. However, since the labels are known through simulation, we are able to tell how well ranked are points of a particular class given a query of pairs from the same class. Our evaluation is as follows. We generate precision/recall curves for three algorithms: our relational Bayesian sets RBSets, "flattened" standard Bayesian sets

[2] Values for vectors $\Theta_1, \Theta_2, \dots, \Theta_5$ were otherwise generated by independent multivariate Gaussian distributions with zero mean and standard deviation of 10

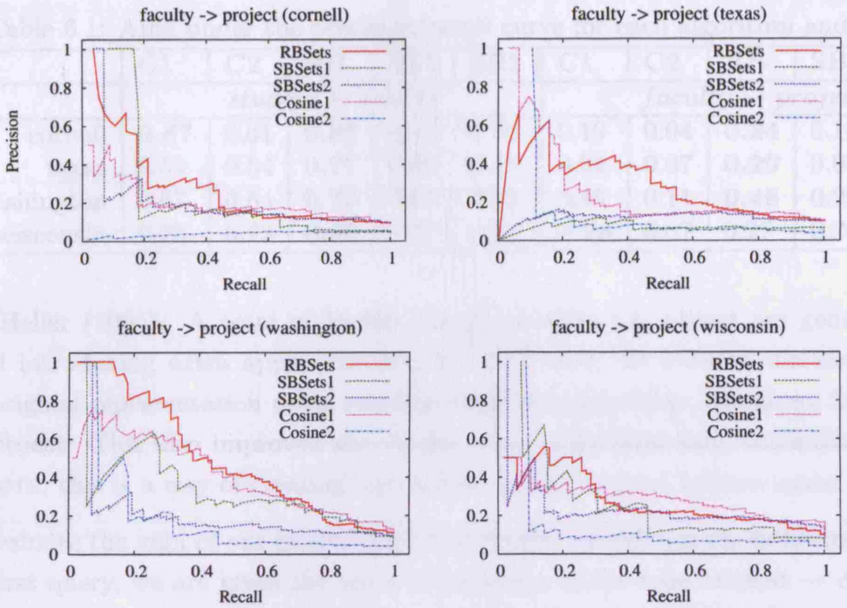


Figure 6.5: Results for *faculty* → *project* relationships.

with Bernoulli model (SBSets) and cosine distance (summing over all elements in the query). For each query, we randomly sampled 10 elements out of the pool of elements of the least frequent class (about 1% of the total number of links), and ranked the remaining 2320 linked pairs. We counted an element as a hit if it was originally from the selected class.

RBSets gives consistently better results for the top 50% retrievals. As an illustration, we depicted four random queries of 10 items in Figure 6.3. Notice that sometimes SBSets can do reasonably, often achieving better precision at the bottom 40% recalls: by the virtue of having few objects in the space of elements of this class, a few of them will appear in pairs both in the query and outside of it, facilitating matching by object similarity since half of the pair is already given as input. We conjecture this explains the seemingly strong results of feature-based approaches on the bottom 40%. However, when this does not happen the problem can get much harder, making SBSets much more sensitive to the query than RBSets, as illustrated in some of the runs in Figure 6.3.

6.4.2 The WebKB experiment

The WebKB data is a collection of webpages from several universities, where relations are directed and given by hyperlinks (Craven et al., 1998). Webpages are classified as being of type *course*, *department*, *faculty*, *project*, *staff*, *student* and *other*. Documents from four universities (*cornell*, *texas*, *washington* and *wisconsin*) are also labeled as such. Binary data was generated from this database using the same methods of Ghahramani

Table 6.1: Area under the precision/recall curve for each algorithm and query.

	C1	C2	RB	SB1	SB2	C1	C2	RB	SB1	SB2
	student → *course*					*faculty* → *project*				
cornell	**0.87**	0.61	**0.87**	0.84	0.80	0.19	0.04	**0.24**	0.18	0.18
texas	0.55	0.54	**0.77**	0.62	0.48	0.24	0.07	**0.29**	0.07	0.12
washington	0.67	0.64	**0.76**	0.69	0.44	0.40	0.11	**0.48**	0.29	0.18
wisconsin	0.75	0.73	**0.88**	0.77	0.55	**0.28**	0.07	0.27	0.20	0.21

and Heller (2005). A total of 19,450 binary variables per object are generated. To avoid introducing extra approximations into RBSets, we reduced dimensionality in the original representation using singular value decomposition, obtaining 25 measures per object. This also improved the results of our algorithm and cosine distance. For SBSets, this is a way of creating correlations in the original feature space.

To evaluate the gain of our model over competitors, we will use the following setup. In the first query, we are given the pairs of webpages of the type *student* → *course* from three of the labeled universities, and evaluate how relations are ranked in the fourth university. Because we know class labels (while the algorithm does not), we can use the class of the returned pairs to label a hit as being "relevant" or "irrelevant." We label a pair (A^i, B^j) as relevant if and only if A^i is of type *student* and B^j is of type *course*, and A^i links into B^j.

This is a very stringent criterion, since other types of relations could also be valid (e.g., *staff* → *course* appears to be a reasonable match). However, this facilitates objective comparisons of algorithms. Also, the *other* class contains many types of pages, which allows for possibilities such as a *student* → *"hobby"* pair. Such pairs might be hard to evaluate (e.g., is that particular hobby incrementally demanding in a way that coursework is? Is it as fun as taking a machine learning course?) As a compromise, we omit all pages from the category *other* in order to better clarify differences between algorithms[3].

Precision/recall curves for the *student* → *course* queries are shown in Figure 6.4. There are four queries, each corresponding to a search over a specific university given all valid *student* → *course* pairs from the other three. There are four algorithms on each evaluation: the standard Bayesian sets with the original 19,450 binary variables for each object, plus another 19,450 binary variables, each corresponding to the product of the respective variables in the original pair of objects (SBSets1); the standard Bayesian sets with the original binary variables only (SBSets2); a standard cosine distance measure over the 25-dimensional representation (Cosine 1); a cosine distance measure using the 19,450-dimensional text data with TF-IDF weights (Cosine 2); our approach, RBSets.

[3] As an extreme example, querying *student* → *course* pairs from the *wisconsin* university returned *student* → *other* pairs at the top four. However, these *other* pages were for some reason course pages - such as `http://www.cs.wisc.edu/~markhill/cs752.html`

In Figure 6.4, RBSets demonstrates consistently superior or equal precision-recall. Although SBSets performs well when asked to retrieve only *student* items or only *course* items, it falls short of detecting what features of *student* and *course* are relevant to predict a link. The discriminative model within RBSets conveys this information through the parameters.

We also did an experiment with a query of type *faculty* → *project*, shown in Figure 6.5. This time results between algorithms were closer. To make differences more evident, we adopt a slightly different measure of success: we count as a 1 hit if the pair retrieved is a *faculty* → *project* pair, and count as a 0.5 hit for pairs of type *student* → *project* and *staff* → *project*. Notice this is a much harder query. For instance, the structure of the *project* webpages in the *texas* group was quite distinct from the other universities: they are mostly very short, basically containing links for members of the project and other project webpages.

Although the precision/recall curves convey a global picture of performance for each algorithm, they might not be completely clear way of ranking approaches for cases where curves intersect on several points. In order to summarize individual performances with a single statistic, we computed the area under each precision/recall curve (with linear interpolation between points). Results are given in Table 6.2. Numbers in bold indicate the algorithm with the highest area. The dominance of RBSets should be clear.

Silva et al. (2007a) describe another application of RBSets, in this case for symmetric protein-protein interactions. In this application, there are no individual object features on which Cosine and SBSets can rely (every X^{ij} measures a pairwise feature), and RBSets performs substantially better.

6.4.3 Biological Application

In this section we consider the problem of automatically discovering analogies between pairs of proteins that are known to interact (Silva et al., 2007a). The goal is to find new subclasses of interactions that might be relevant for further study: e.g., a pair of proteins P_1:P_2 might belong to a class of interactions that is not yet fully formalized, and scientists exploring the interaction between P_1:P_2 might want to find other interactions which behave in an analogous way. This can lead to novel ways of categorizing proteins based on functional similarity.

In the molecular biology experiments presented here, we use data that indicates if two pairs of proteins interact or not. This reduces to a binary class problem with two classes: *interaction exists* and *interaction does not exist*. The type of interaction will depend on the data. For instance, the MIPS data (Mewes and Amid, 2004)indicate co-complex protein pairs. It is hand curated data and does not include information from high-throughput datasets. Since the population of protein-protein interactions depend on the experimental conditions under which they were measured, the subpopulations that our method is meant to find also depend on such conditions.

The approach cannot succeed if the underlying model is not a good classifier of protein-protein interactions. Results from Qi et al. (2006) indicate that logistic regression can provide reasonable predictions for the data we use in our experiments. It is always important to check this condition before using our method, since it assumes from the outset that the data can be reasonably separated into the interaction/no interaction classes.

We evaluate our approach on data collected from yeast cells. Our gold standard for protein-protein interactions is the Munich Information Center for Protein Sequences (MIPS) catalog (Mewes and Amid, 2004). Moreover, we make use of the MIPS classification system for proteins in the evaluation criteria described shortly. We also describe competing approaches against which we compare our algorithm.

Evaluation is not a straightforward process, and is prone to subjective assessments and extended discussions, as typically happens in the development of a new taxonomy. In our studies, we propose an objective measure of evaluation that is used to rank different algorithms. We use the classes lying on the bottom of the MIPS classification system, for instance, $M_1 = 67.04.01.02$ (*other cation transporters (Na, K, Ca , NH4, etc.)*) and $M_2 = 67.5$ (*transport mechanism*) to generate a precision-recall curve and calculate the area under that curve (AUC). The retrieval algorithm that generates the ranked list of protein pairs does not receive any information concerning the MIPS taxonomy.

Notice that in the experiments that follow, we want to focus on very specific MIPS subclasses. Our criterion is rather stringent, in the sense it requires a perfect match of each protein pair with the MIPS categorization, which is not an unique gold standard for analogical similarity. AUC scores will be lower than in typical information retrieval applications.

We compare our approach against two widely used similarity metrics for information retrieval, and one state-of-the-art method:

- the nearest neighbor measure (NN) with Euclidean distances: for a given query set S and a given candidate point R^I, the distance between the point and the set is given by the minimum distance between R^I and any individual point in S;
- the cosine distance metric (COS): the distance between any two vectors is taken as the inner product of the normalized vectors, where the normalized vector is obtained by dividing it by its Euclidean norm. To measure the distance between a point and a set, we take the average of the distances;
- the Gaussian Bayesian sets metric (GBSETS): Bayesian sets (Ghahramani and Heller, 2005) give state-of-the-art performance for tasks such as retrieval of word concepts and images. Since our features are continuous, we used a variation based

on Gaussian models.

Because our variation of Bayesian sets is motivated by relational data, we call our approach the relational Bayesian sets method (RBSETS), to contrast it with the Gaussian Bayesian set (GBSETS) described above.

None of these other approaches can be interpreted as measures of analogical similarity, since they do not take into account how the protein pair features (gene expression, in our case) contribute to their interaction. We are not aware of any other measure which does. It is true that a direct measure of analogical similarity is not theoretically required to perform well according to our evaluation metric. However, we will see that in this task our measure can often perform an order of magnitude better than other approaches by reasoning analogically.

We chose 4 different protein-protein combinations for our benchmark. They were chosen according to the MIPS categorization and shown below, along with the percentage of interacting pairs they represent after we remove the query elements:

- **Query 1:** 67.04.01.02 × 67.5 (i.e., *other cation transporters (Na, K, Ca , NH4, etc.)* × *transport mechanism*), 1% of the interacting pairs;
- **Query 2:** 40.03 × 06.13.01 (i.e., *cytoplasm* × *cytoplasmic and nuclear degradation*), 2.5% of the interacting pairs;
- **Query 3:** 04.03.03 × 04.01.04 (i.e., *tRNA processing* × *rRNA processing*), 0.3% of the interacting pairs;
- **Query 4:** 8.04 × 8.04 (i.e., *mitochondrial transport* × *mitochondrial transport*), 0.7% of the interacting pairs;

For each query evaluation, we randomly choose 15 elements of the given class of pairs and run the 4 algorithms with the selected input. This is repeated 20 times. Figure 6.6 shows the average precision-recall curves for each query, with the coordinates of each point in the curve being the average of the 20 query results.

As expected, such curves are lower than typical precision-recall curves for the binary classification problem of predicting protein-protein interactions, such as the ones depicted in Qi et al. (2006). A direct comparison between the classification curves , as in Qi et al. (2006), and the retrieval curves of Figure 6.6 is not appropriate, since the classification curves have a well-defined loss function (0/1, for wrong and correct predictions)[4].

We can see how much better RBSETS performs when compared against different approaches. Table 6.2 summarizes the difference in the area under curve between our

[4] It is also clear that we are dealing with many fewer "positive examples" (i.e., the selected query size) than in a common binary classification setup. In Qi et al.'s setup, there are thousands of "positive examples" against 15 of ours.

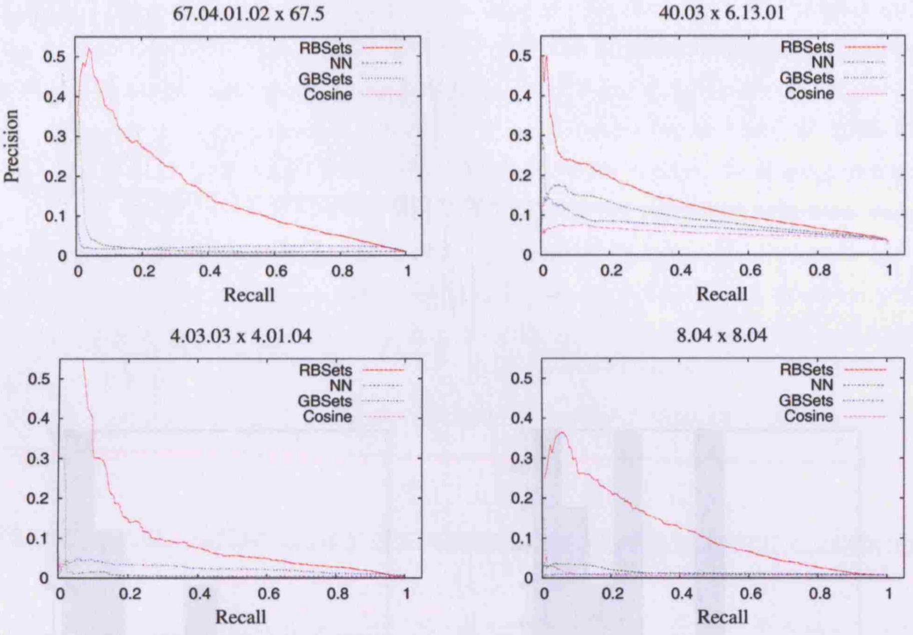


Figure 6.6: Average precision/recall curves for four different types of queries of size 15. Each curve is an average over 20 random trials.

Query	RBSets - NN	RBSets - GBSets	RBSets - Cosine
1	0.14 (0.05)	0.14 (0.05)	0.16 (0.05)
2	0.04 (0.03)	0.06 (0.02)	0.08 (0.02)
3	0.10 (0.03)	0.08 (0.03)	0.10 (0.04)
4	0.11 (0.04)	0.12 (0.04)	0.12 (0.04)

Table 6.2: Differences in the area under the curve between our algorithm and each of the other three algorithms.Each entry contains the average (over 20 trials) and the respective standard deviations in parenthesis. The areas under the curve for our algorithm are $(0.17, 0.14, 0.10, 0.13)$, with respective standard deviations of $(0.05, 0.02, 0.03, 0.04)$.

approach and the others. All differences are significant under a sign test at a 0.01 level (the differences are all positive).

The limited performance of Gaussian Bayesian sets can be attributed not only to the relational nature of the data, which this model was not designed for, but also to the multimodality of the distribution of a few variables. However, removing these variables did not alter significantly the behavior the algorithm, which now might be due to the loss of information given by these variables.

It is also interesting to visualize the distribution of the ratios of the performance of different algorithms 6.7. For the nearest neighbor algorithm, we computed the ratio between the area under curve of RBSets and the area for NN in each of the 20 trials. Gains over an order of magnitude are common.

There are some explanations for the rapid degradation of precision in Query 2 (40.03 ×

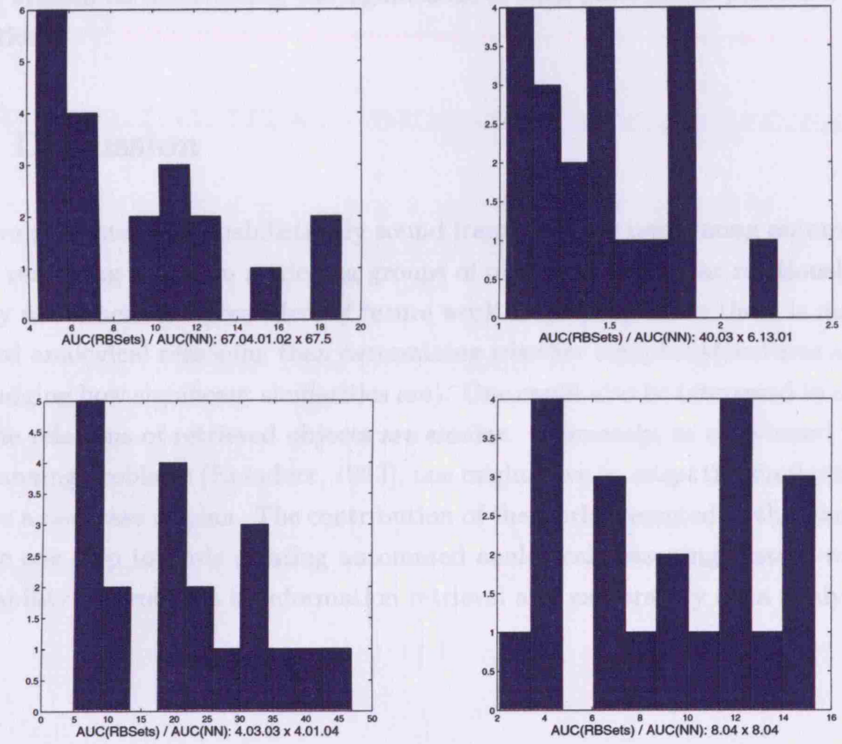


Figure 6.7: Histogram of the ratio AUC(RBSETS) / AUC(NN) for the four different types of queries.

06.13.01). The particular difficulty in this case is the fact that every protein in all queries is also of MIPS types (5.01, 40.03) (with the ribosome biogenesis type (5.01) being one of the most numerous categories in the MIPS data). This is a good illustration of the strictness of our evaluation criterion, since in some sense pairs of type 40.03 × 06.13.01 are also of type 40.03 × 5.01, 40.03 × 4.03, 5.01 × 5.01. Had we counted pairs of type (5.01, 40.03) × (5.01, 40.03) as valid hits, we would have achieved very high precision-recall curves (e.g., close to 100% precision in the top 50 hits), but the query would then be uninteresting due to the high concentration of pairs of this subpopulation. The restriction of perfect matching makes results appear less dramatic, but implicitly it might be ranking relevant subpopulations within (5.01, 40.03) × (5.01, 40.03). Only further studies on determining the significance of such pairs might provide a more fair evaluation.

6.5 Discussion

We have presented a probabilistically sound framework for performing automated analogical reasoning based on retrieving groups of objects with similar relationships given a query set. There is a great deal of future work to be done, since there is more to automated analogical reasoning than determining whether complex structures are similar (like judging how significant similarities are). One could also be interested in explaining why the relations of retrieved objects are similar. Ultimately, in case-based reasoning and planning problems (Kolodner, 1993), one might have to *adapt* the similar structures to solve a new case or plan. The contribution of the work presented in this chapter is to provide one step towards creating automated analogical reasoning system with direct applicability to problems in information retrieval and exploratory data analysis.

Chapter 7

Infinite Overlapping Mixture Model

Although clustering data into mutually exclusive partitions has been an extremely successful approach to unsupervised learning, there are many situations in which a richer model is needed to fully represent the data. This is the case in problems where data points actually simultaneously belong to multiple, overlapping clusters. For example a particular gene may have several functions, therefore belonging to several distinct clusters of genes, and a biologist may want to discover these through unsupervised modeling of gene expression data. In this chapter, we present a new nonparametric Bayesian method, the Infinite Overlapping Mixture Model (IOMM), for modeling overlapping clusters. The IOMM uses exponential family distributions to model each cluster and forms an overlapping mixture by taking products of such distributions, much like products of experts (Hinton, 2002). The IOMM allows an unbounded number of clusters, and assignments of points to (multiple) clusters is modeled using an Indian Buffet Process (IBP), (Griffiths and Ghahramani, 2006). The IOMM has the desirable properties of being able to focus in on overlapping regions while maintaining the ability to model a potentially infinite number of clusters which may overlap. We derive MCMC inference algorithms for the IOMM and show that these can be used to cluster movies into multiple genres.

7.1 Overlapping Clustering

The problem of clustering data has led to many pivotal methods and models in pattern recognition and machine learning which are widely used across many fields. Unfortunately, while clustering methods are wonderful tools for many applications, they are actually quite limited. Clustering models traditionally assume that each data point belongs to one and only one cluster; that is, there are K exhaustive and mutually exclusive clusters explaining the data. In many situations the data being modeled can have a much richer and more complex hidden representation than this single, discrete hidden variable (the cluster or partition assignment) which clustering strives to discover. For example, there may be overlapping regions where data points actually belong to multiple clusters (e.g. the movie "Scream" could belong to both the "horror" movie cluster and the "comedy" cluster of movies). Also, in collaborative filtering one might be interested in predicting which movies someone will like based on previous movies they have liked, and the patterns of movie preferences of others. A common approach is to cluster people; clusters could characterize gender, age, ethnicity, or simply movie taste (e.g. people who like horror movies). However, any particular person can clearly belong to multiple such clusters at the same time, e.g. a female in her 20s who likes horror movies.

In this paper, we develop a new model for overlapping clusters based on a principled statistical framework. Consider the traditional mixture model (Bishop, 2006) for clustering, which can be written

$$p(\mathbf{x}_i|\Theta) = \sum_{j=1}^{K} \pi_j p_j(\mathbf{x}_i|\boldsymbol{\theta}_j)$$

where π_j represents the mixing weight (or mass) of cluster j, $p_j(\mathbf{x}_i|\boldsymbol{\theta}_j)$ is the density for cluster j with parameters $\boldsymbol{\theta}_j$, and $\mathbf{x}_i$ represents data point i. This mixture model can be rewritten

$$p(\mathbf{x}_i|\Theta) = \sum_{\mathbf{z}_i} p(\mathbf{z}_i) \prod_{j=1}^{K} p_j(\mathbf{x}_i|\boldsymbol{\theta}_j)^{z_{ij}} \tag{7.1}$$

where $\mathbf{z}_i = [z_{i1}, \ldots, z_{iK}]$ is a binary vector of length K, $z_{ij} \in \{0,1\} \forall ij$, $\sum_j z_{ij} = 1$, and $P(z_{i1} = 0, \ldots, z_{i,j-1} = 0, z_{ij} = 1, z_{i,j+1} = 0, \ldots, z_{iK} = 0) = \pi_j$. The setting $z_{ij} = 1$ means data point i belongs to cluster j.

To create a model for *overlapping* clusters, two modifications can be made to this representation. First of all, removing the restriction $\sum_j z_{ij} = 1$ allows binary vectors with multiple ones in each row. In other words, instead of K possible binary $\mathbf{z}$ vectors allowed in the mixture model, this allows 2^K possible assignments to overlapping clusters. Removing this restriction will also introduce a normalization constant for the product which for exponential family densities $p_j(x)$ will be easy to compute. Secondly, the number of such overlapping clusters K can be taken to infinity by making use of the Beta-Binomial model underlying the Indian Buffet Process (IBP), (Griffiths and Ghahramani, 2006). This infinite limit means that the model is not restricted a priori to having a fixed number of clusters; and it allows the data to determine how many clusters are required. In the case where the $p_j(\cdot)$ are Gaussian densities, this model will define overlapping clusters in terms of the region where the mass of all Gaussians j, such that $z_{ij} = 1$, overlaps; this region itself will define a Gaussian since the product of Gaussians is Gaussian. Other exponential family models (e.g. multinomials for text data) will work analogously. In sections 7.2 and 7.3 we describe this Infinite Overlapping Mixture Model in detail, and in section 7.4 we outline how to perform inference in the model.

This model for overlapping clusters can be seen as a modern nonparametric generalization of the multiple cause factorial models (Saund, 1994; Ghahramani, 1995). Moreover, it can also be seen as an infinite nonparametric generalization of the influential products-of-experts model (Hinton, 2002). These relationships will be discussed further in section 7.5. Lastly, we give experimental results for our model in section 8.3.

7.2 Overlapping Mixture Models

We are interested in clustering data such that each data point is allowed to belong to multiple clusters, instead of being constrained to a single cluster. In order to do this we need a sensible way of modeling individual data points that belong to many clusters, and which derives from the broad models for each individual cluster. We modify a traditional finite mixture model (7.1) to achieve this. First we remove the restriction that the binary assignment vector, $\mathbf{z}$, for each data point must sum to 1, and secondly, as we will discuss in the next section, we use a prior that allows a potentially infinite number of clusters, where the actual required number of clusters is inferred automatically from the data. Removing the restiction that $\mathbf{z}$ sums to one in (7.1), results in a model in which, if a data point belongs simultaneously to several clusters, the distribution of that point is given by the product of component distributions:

$$p(\mathbf{x}_i|\mathbf{z}_i, \Theta) = \frac{1}{c}\prod_k p_k(\mathbf{x}_i|\boldsymbol{\theta}_k)^{z_{ik}} \tag{7.2}$$

Here $\mathbf{z}_i = (z_{i1} \dots z_{iK})$ is a binary vector of cluster assignments for data point i, $\boldsymbol{\theta}_k$ are the parameters of cluster k, and c is the normalizing constant which depends on $\mathbf{z}_i$ and Θ, and which is needed to ensure that the density integrates to one. Multiplying distributions is a very natural and general way of encoding the idea of overlapping clusters—each cluster provides a soft constraint on the probable region for observing a data point, and overlapping clusters correspond to a conjunction of these constaints.

If the models we are using, $p(\mathbf{x}_i|\boldsymbol{\theta}_k)$, are in the exponential family then:

$$p(\mathbf{x}_i|\boldsymbol{\theta}_k) = g(\mathbf{x}_i)f(\boldsymbol{\theta}_k)e^{s(\mathbf{x}_i)^\top \phi(\boldsymbol{\theta}_k)} \tag{7.3}$$

where $s(\mathbf{x}_i)$ are the sufficient statistics, $\phi(\boldsymbol{\theta}_k)$ are the natural parameters, and f and g are non-negative functions. Substituting into equation (7.2) we get:

$$p(\mathbf{x}_i|\mathbf{z}_i, \Theta) = \frac{g(\mathbf{x}_i)^{\sum_k z_{ik}}}{c}\left[\prod_k f(\boldsymbol{\theta}_k)\right] e^{s(\mathbf{x}_i)^\top (\sum_k z_{ik}\phi(\boldsymbol{\theta}_k))} \tag{7.4}$$

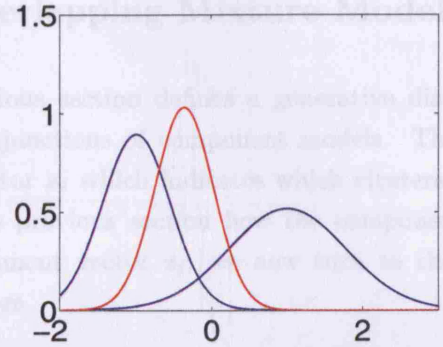


Figure 7.1: Product of two Gaussians. Here the product of the two blue Gaussians ($\mu_1 = -1$, $\mu_2 = 1$ and $\sigma_1^2 = 0.2$, $\sigma_2^2 = 0.6$) is the red Gaussian.

From this we see that, conditioned on $\mathbf{z}_i$, the product of exponential family distributions results in a distribution in the same family (7.3), but with new natural parameters $\tilde{\phi} = \sum_k z_{ik}\phi(\boldsymbol{\theta}_k)$. It follows that normalization constants like c are not problematic when the component densities are in the exponential family.

In the case of Gaussian clusters:

$$p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\mu}, \Sigma) = \frac{1}{c}\exp\left\{-\frac{1}{2}[\mathbf{x}^\top(\sum_k z_{ik}\Sigma_k^{-1})\mathbf{x} \right. \\ \left. -2\mathbf{x}^\top(\sum_k z_{ik}\Sigma_k^{-1}\boldsymbol{\mu}_k) + \sum_k z_{ik}\boldsymbol{\mu}_k^\top\Sigma_k^{-1}\boldsymbol{\mu}_k]\right\} \quad (7.5)$$

Letting $S^{-1} = \sum_k z_{ik}\Sigma_k^{-1}$ and $\mathbf{m} = \sum_k z_{ik}\Sigma_k^{-1}\boldsymbol{\mu}_k$ from within equation (7.5), we can see that the new parameters for the Gaussian product model are $\tilde{\Sigma} = S$ and $\tilde{\boldsymbol{\mu}} = S\mathbf{m}$.

In the case of binary data and multivariate Bernoulli clusters:

$$p(\mathbf{x}_i|\mathbf{z}_i, \Theta) = \frac{1}{c}\exp\left\{\sum_{k,d} z_{ik}x_{id}\log(\frac{\theta_{kd}}{1-\theta_{kd}})\right\} \quad (7.6)$$

where d indexes the dimensions of $\mathbf{x}_i$. Using equation (7.6) we can derive that the new parameters for the Bernoulli product model are:

$$\tilde{\Theta}_d = \frac{\prod_k \theta_{kd}^{z_{ik}}}{\prod_k(1-\theta_{kd})^{z_{ik}} + \prod_k \theta_{kd}^{z_{ik}}}. \quad (7.7)$$

These product models have the desirable property that multiple cluster assignments will help focus the model on a particular overlapping region. See Figure 7.1 for a simple illustration. The two blue Gaussians each model independent Gaussian clusters ($\mathbf{z}_1 = [1\ 0]$ and $\mathbf{z}_2 = [0\ 1]$); the red Gaussian models the overlap of the two blue Gaussians clusters and defines the overlapping cluster $\mathbf{z}_3 = [1\ 1]$.

7.3 Infinite Overlapping Mixture Models via the IBP

The model in the previous section defines a generative distribution for overlapping clusters by forming conjunctions of component models. The key component in this model is the binary vector $\mathbf{z}_i$ which indicates which clusters data point i belongs to. We have defined in the previous section how the component models are combined, given the binary assignment vector $\mathbf{z}_i$; we now turn to the distribution over these binary assignment vectors.

A very simple model assigns each element z_{ik} an independent Bernoulli distribution

$$z_{ik}|\pi_k \sim \text{Bernoulli}(\pi_k) \tag{7.8}$$

where π_k is the mixing proportion, or probability of belonging to cluster k. Note that the π_k need not sum to 1 over k, since belonging to one cluster does not exclude belonging to others. We give each π_k a Beta distribution

$$\pi_k|\alpha \sim \text{Beta}(\frac{\alpha}{K}, 1) \tag{7.9}$$

which is conjugate to the Bernoulli, where α controls the expected number of clusters a data point will belong to.

A classical problem in clustering, which also occurs in our overlapping clustering model, is how to choose the number of clusters K. While it is possible to perform model comparison for varying K, this is both computationally costly and statistically hard to justify (Neal, 2000). A more elegant solution is to define a nonparametric model which allows an unbounded number of clusters, K.

In order to derive the nonparametric model, we have defined the prior over π_k in (7.9) to scale so that as K grows larger, the prior probability of each data point belonging to cluster k decreases. Using this scaling it is possible to take the limit $K \to \infty$, integrate out all the mixing proportions π, and still obtain a well-defined distribution over the binary assignment vectors $\mathbf{z}$. This distribution over the assignment vectors results in a process known as the Indian Buffet Process (IBP), (Griffiths and Ghahramani, 2006).

The IBP defines a distribution which can be used to represent a potentially infinite number of hidden features, or in this case cluster assignments, associated with data points. More specifically, it defines a distribution over infinite binary matrices, Z, which can be derived by starting with a distribution over finite $N \times K$ matrices given by (7.8) and (7.9), where N is the number of data items, K is the number of features, and the ith row of Z is $\mathbf{z}_i$, and taking the limit as K goes to infinity. Exchangeability of the rows is preserved, and the columns are independent.

The IBP is a simple generative process which results from this distribution, with an analogy to customers eating from Indian Buffets. N customers line up on one side

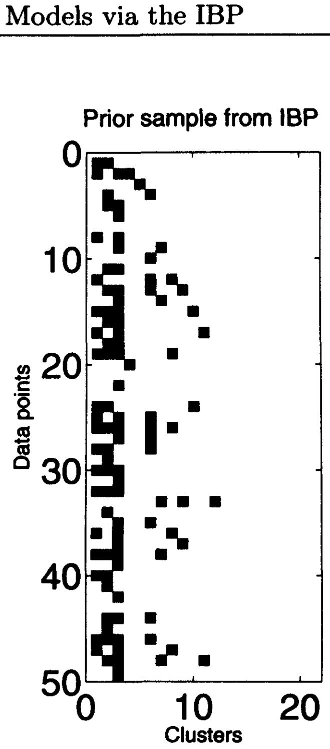


Figure 7.2: The first 50 rows of the IBP sample (Z matrix) which was used to assign data points to clusters in Figure 7.4b.

of an Indian Buffet with infinitely many dishes. The first customer serves himself from Poisson(α) dishes (at which point his plate is full). The next customers serve themselves dishes in proportion to their popularity, such that customer i serves herself dish k with probability $\frac{m_k}{i}$, where m_k is the number of previous customer which have served themselves that dish. After passing all previously sampled dishes, customer i then proceeds to try Poisson($\frac{\alpha}{i}$) new dishes. In terms of binary matrices, each of the N customers is a row in the matrix, each dish is a column, and each binary value in the matrix, z_{ik}, indicates whether customer i helped themselves to dish k. A sample of such a matrix is shown in Figure 7.2.

Markov Chain Monte Carlo algorithms have been used to do inference in this model (Griffiths and Ghahramani, 2005; Görür et al., 2006). These algorithms need to compute the full conditional distribution of the assignment variables:

$$P(z_{ik} = 1|Z_{-(ik)}, X) \propto P(X|Z)P(z_{ik} = 1|Z_{-(ik)}) \tag{7.10}$$

where X is the complete data matrix and Z is the full binary feature matrix, and $Z_{-(ik)}$ is the binary matrix excluding element z_{ik}. In order to compute the last term in equation (7.10), we can generalize from the finite binary matrix case. Starting from

(7.8) and (7.9) and integrating out π_k gives:

$$\begin{aligned} P(z_{ik}=1|\mathbf{z}_{-i,k}) &= \int_0^1 P(z_{ik}|\pi_k)P(\pi_k|\mathbf{z}_{-i,k})d\pi_k \\ &= \frac{m_{-i,k}+\frac{\alpha}{K}}{N+\frac{\alpha}{K}} \end{aligned} \tag{7.11}$$

where $m_{-i,k} = \sum_{j\neq i} z_{jk}$ and $\mathbf{z}_{-i,k}$ is $\mathbf{z}_i$ excluding. Taking the limit as $K \to \infty$ results in:

$$P(z_{ik}=1|\mathbf{z}_{-i,k}) = \frac{m_{-i,k}}{N} \tag{7.12}$$

for any k in which $m_{-i,k} > 0$. The number of new features associated with i should be drawn from a Poisson($\frac{\alpha}{N}$) distribution. The IBP is described in full detail in Griffiths and Ghahramani (2005).

Incorporating this IBP prior over the assignment vectors into the OMM defined in section 7.2 results in an Infinite Overlapping Mixture Model (IOMM), with all the components required to do inference and learning.

7.4 IOMM Learning

We use Markov Chain Monte Carlo (MCMC) to do inference in our Infinite Overlapping Mixture Model (IOMM). The MCMC algorithm that we implemented is based on Algorithm 7.1, where k_+ is the number of clusters that data points, excluding i, belong to. Since the product model is non-conjugate we use Metropolis-Hastings (MH) to resample the model parameters, Θ.

Algorithm 7.1 MCMC for the IOMM.

```
Initialize $\Theta$
for $j = 1$ to NumIters do
  for $i = 1$ to $N$ do
    for $k = 1$ to $k_+$ do
      $z_{ik} \sim z_{ik}|z_{-i,k}, \mathbf{x}_i, \Theta$
    end for
    Propose adding new clusters
    Accept or reject proposal
  end for
  Resample $\Theta|Z, X$ using MH proposal
end for
```

At each iteration we resample the binary matrix, Z, using Gibbs sampling for existing clusters k (i.e. those clusters which have data points other than i as members), where:

$$p(z_{ik}=1|\mathbf{z}_{-i,k}, \mathbf{x}_i, \Theta) \propto \frac{m_{-i,k}}{N} p(\mathbf{x}_i|\Theta, z_{ik}=1, \mathbf{z}_{-i,k})$$

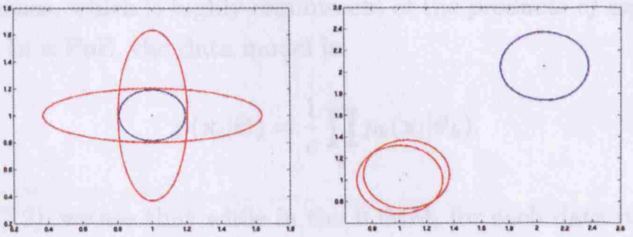


Figure 7.3: Left: IOMM where the cluster component densities are Gaussian (contours at 1 s.d.). Right: Factorial Model. In each figure, the original Gaussian clusters are shown in red, while the Gaussian cluster modeling membership in both of the original clusters is shown in blue. The IOMM is able to focus in on the area where the original two Gaussians overlap, taking their (unrestricted) covariances into account. The factorial model yields a Gaussian whose mean is the sum of the means of the original two Gaussians, and the (typically axis-aligned) covariance is restricted to be the same for all clusters, since it results from the same additive noise.

and

$$p(z_{ik}\!=\!0|\mathbf{z}_{-i,k},\mathbf{x}_i,\Theta) \propto \frac{N\!-\!m_{-i,k}}{N} p(\mathbf{x}_i|\Theta, z_{ik}\!=\!0, \mathbf{z}_{-i,k})$$

After resampling the existing cluster assignments for a data point i, we then propose adding assignments of i to new clusters using Metropolis-Hastings and following Meeds et al. (2007). Here the number of new clusters and their parameters are proposed jointly, where the number of new clusters is drawn from a Poisson($\frac{\alpha}{N}$) and the new parameters for those clusters are drawn from the prior.

After resampling the entire Z matrix we resample each θ'_{kd} drawing from the proposal distribution centered around the current value of θ_{kd}. The acceptance ratio for θ'_{kd} is:

$$a = \frac{p(\mathbf{x}_d|Z, \boldsymbol{\theta}'_d)p(\boldsymbol{\theta}'_d)}{p(\mathbf{x}_d|Z, \boldsymbol{\theta}_d)p(\boldsymbol{\theta}_d)} \frac{T(\theta_{kd}|\theta'_{kd}, \omega)}{T(\theta'_{kd}|\theta_{kd}, \omega)} \tag{7.13}$$

where $\boldsymbol{\theta}'_d$ is $\boldsymbol{\theta}_d$ substituting θ'_{kd} for θ_{kd}, T is the transition probability between different values of θ_{kd}, and ω controls the width of this transition proposal distribution. For example, for binary data we can use multivariate Bernoulli clusters (7.6), (7.7). A sensible proposal for θ_{kd} might be $\theta'_{kd} \sim \text{Beta}(\omega\theta_{kd}, \omega(1-\theta_{kd}))$.

7.5 Related Work

The infinite overlapping mixture model has many interesting relationships to other statistical models. In this section we review some of these relationships, highlighting similarities and differences.

The likelihood function in equation (7.2) is a product of likelihoods from different component densities, which is highly reminiscent of the *products of experts* (PoE) model (Hinton, 2002). In a PoE, the data model is:

$$p(\mathbf{x}_i|\Theta) = \frac{1}{c}\prod_k p_k(\mathbf{x}_i|\boldsymbol{\theta}_k).$$

Comparing to (7.2), we see that while in the IOMM, for each data point, $\mathbf{x}_i$ a product of a *subset* of the experts is taken depending on the setting of $\mathbf{z}_i$, in the PoE, each data point is assumed to be generated by the product of *all* experts. This would appear to be a large difference; however we will now show that it is not. Consider the special case of a PoE where each expert is a mixture of a uniform and a Gaussian distribution (a "unigauss" distribution), described in Section 4 of Hinton (2002).[1] For this model, using $\mathbf{1}(x) = 1$, $\forall x$, to denote the unnormalized uniform distribution (where normalization is subsumed in c above):

$$\begin{aligned} p_k(\mathbf{x}_i|\boldsymbol{\theta}_k) &= (1-\pi_k)\,\mathbf{1}(\mathbf{x}_i) + \pi_k\mathcal{N}(\mathbf{x}_i|\mu_k, \Sigma_k) \\ &= \sum_{z_{ik}\in\{0,1\}} p(\mathbf{x}_i|z_{ik}, \boldsymbol{\theta}_k)p(z_{ik}|\boldsymbol{\theta}_k) \end{aligned} \tag{7.14}$$

where $p(z_{ik} = 1|\boldsymbol{\theta}_k) = \pi_k$ and $p(\mathbf{x}_i|z_{ik}, \boldsymbol{\theta}_k) = \mathcal{N}(\mathbf{x}_i|\mu_k, \Sigma_k)^{z_{ik}}$. Conditioning on $\mathbf{z}_i$ we now see that

$$p(\mathbf{x}_i|\mathbf{z}_i, \Theta) \propto \prod_k \mathcal{N}(\mathbf{x}_i|\mu_k, \Sigma_k)^{z_{ik}}$$

which is of the same form as in the IOMM (7.2). More generally, we can therefore view our IOMM as an infinite nonparametric Bayesian Product of Experts, under the assumption that each expert is a mixture of a uniform and an exponential family distribution.

Another line of thought relates the IOMM to multiple cause or factorial models (Saund, 1994; Hinton and Zemel, 1994; Ghahramani, 1995; Sahami et al., 1996). Factorial models are closely related to factor analysis. Each data point $\mathbf{x}_i$ is represented by a latent vector $\mathbf{z}_i = (z_{i1}, \ldots, z_{iK})$. In factor analysis, $\mathbf{z}_i$ is assumed to be multivariate Gaussian, and $\mathbf{x}_i$ and $\mathbf{z}_i$ are assumed to be linearly related. A factorial model can be obtained by letting each z_{ik} be discrete, the binary case $z_{ik} \in \{0, 1\}$ corresponds to data points having 2^K possible feature vectors. The corresponding distributions over data for each possible feature vector are formed by somehow combining parameters associated with the individual features. In Hinton and Zemel (1994) and Ghahramani (1995), the parameters of the individual features are simply added to form the mean of the distribution of $\mathbf{x}_i$ given $\mathbf{z}_i$, with subsequent Gaussian noise added. That is, $E[\mathbf{x}_i] = A\mathbf{z}_i$, where A is some $D \times K$ matrix whose columns are means for the individual features, and the binary vector $\mathbf{z}_i$ picks out which columns to include in the sum for each point. This idea was used to model gene expression data by Lu et al. (2004); it was also

[1] Strictly speaking a "uniform" on the reals is improper, but this can be approximated by a Gaussian with very large variance.

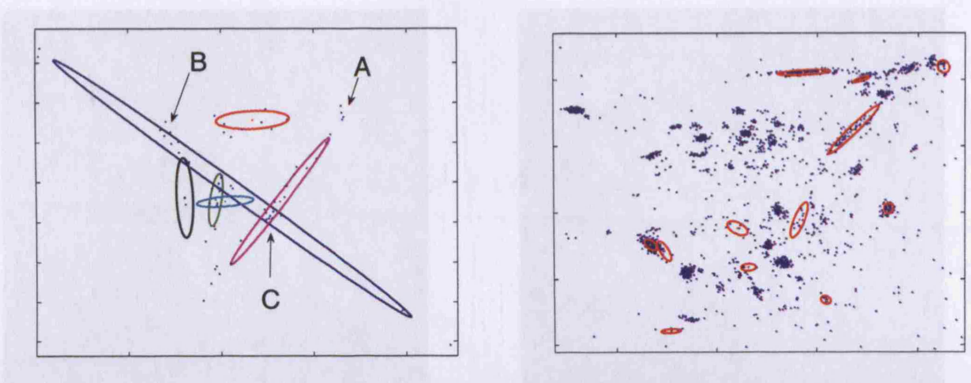


Figure 7.4: Two draws from the IOMM with Gaussian cluster models. a) left: A draw with 6 independent Gaussian clusters. Label A shows data points which belong to both the red and magenta clusters, label B shows data points which belong to both the red and blue clusters, and label C shows datapoints which belong to both the magenta and blue clusters. b) right: A larger draw from the IOMM with more independent clusters. Part of the IBP sample (Z matrix) used for assignments of data points to clusters is shown in Figure 7.2

independently re-invented by Segal et al. (2003) and Battle et al. (2005) and also used to discover multiple overlapping processes in gene expression data. Recently, the model of Segal et al. (2003) was extended by Banerjee et al. (2005) from Gaussians to other exponential family distributions.

While all the models we have reviewed in the previous paragraph are clearly useful and share with the IOMM the idea of using a binary latent vector $\mathbf{z}_i$ to model presence or absence of a hidden feature (which could be seen as indicating membership in a particular "cluster"), they do not make reasonable models for *overlapping* clusters. The additive combination rule of the factorial models $E[\mathbf{x}_i] = A\mathbf{z}_i$ does not capture the intuition of overlapping clusters, but rather of multiple processes that add together (Figure 7.3). For example, if the first and second columns of A are identical ($\mathbf{a}_1 = \mathbf{a}_2$), then one would expect that data points that simultaneously belong to both the first and second cluster ($z_{i1} = 1 = z_{i2}$) should have the same mean as the first cluster ($\mathbf{a}_1$). While this is the case for the IOMM due to the overlapping model we define ((7.2)), this is not the case in any of the factorial models described in the previous paragraph.

7.6 Experiments

Since the IOMM is a generative model, we first tried generating from the model using full covariance Gaussians (7.5). Figure 7.4 shows two illustrative datasets that were generated in 2D along with the Gaussians which represent each independent cluster in the model. The IBP sample (or Z matrix) from which Figure 7.4b was generated is given in Figure 7.2. The parameters for each independent cluster were drawn from the prior (Normal-Inverse Wishart), and the data points were drawn from the product of Gaussians which corresponded to their cluster assignments from the Z matrix. We can

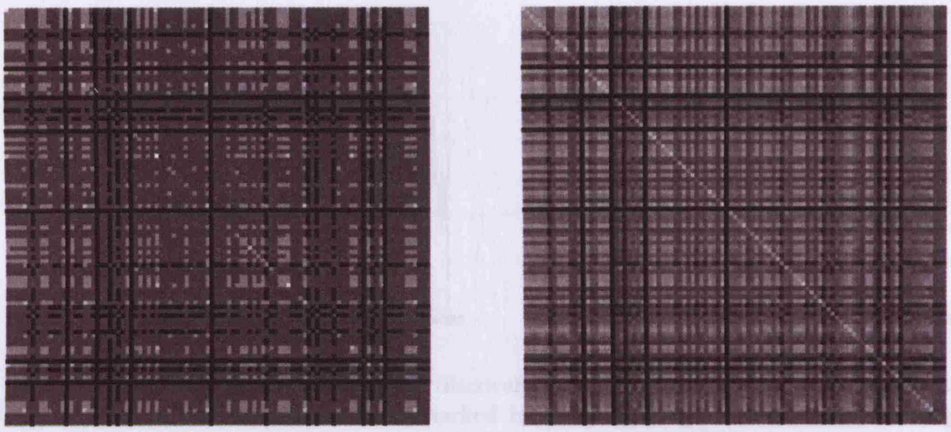


Figure 7.5: The U^* (left) and learned $\hat{U}$ (right) matrices showing the number of shared clusters for each pair of data points in the synthetic data set.

see from these figures that even with a small number of components the IOMM can generate richly structured data sets.

We also generated data from the IOMM with Bernoulli clusters, and then used this synthetic data to test IOMM learning. This synthetic data consisted of $N = 100$ data points in $D = 32$ dimensions, and had $K = 11$ underlying independent clusters. We ran our MCMC sampler for 4000 iterations, burning in the first 1000. Because the clusters that specific columns in the Z matrix correspond to can be permuted, we cannot directly compare the learned Z matrix to the true Z matrix which generated the data. Instead we compute the matrix $U = ZZ^{\top}$, which is invariant to column permutations. This $N \times N$ matrix computes the number of shared clusters between each pair of data points in the data set, and is therefore a good column-invariant way of determining how well the underlying cluster assignment structure is being discovered. Since we have many MCMC samples from which to compute the learned U matrix (which we will call $\hat{U}$), we average all the U matrices together to get $\hat{U}$. The true U matrix, U^*, is constructed from the true Z matrix. Both U^* and $\hat{U}$ are shown in Figure 7.5. Since U^* is a single sample and $\hat{U}$ is averaged over many samples, $\hat{U}$ is a little lighter (it is reasonable to expect that a few samples will assign a data point to even improbable clusters) and smoother, but the structure of $\hat{U}$ is extremely similar to that of U^*. We then rounded the values in $\hat{U}$ to the nearest integer and compared with U^*. Table 7.1 provides summary statistics for $\hat{U}$ in terms of the percentage of pairs of data points in $\hat{U}$ which share the exact same number of clusters as the same pair in U^*, differ by at most 1 cluster, and differ by at most 2 clusters. Figure 7.6 (right) is a box plot showing the distribution of the number of inferred overlaps in $\hat{U}$ for each true number of overlaps in U^*. We can see that the model gives reasonable estimates of the number of overlaps, but is less able to estimate the rare cases of large numbers of overlaps. Lastly, Figure 7.6 (left) plots the inferred number of clusters at each MCMC iteration, and suggests reasonable mixing of the sampler.

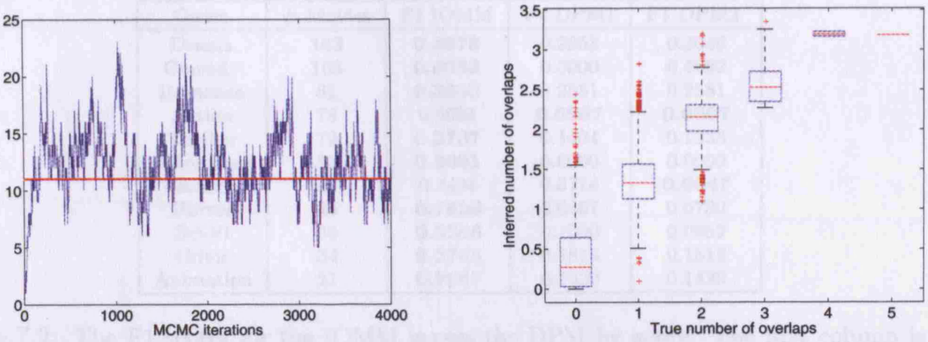


Figure 7.6: Left: The number of discovered clusters, K, across MCMC iterations. The true number of clusters is marked in red (11). Right: A box plot showing the distribution of inferred number of shared clusters in $\hat{U}$ for each true number of shared cluster in U^*, for every data point pair.

Statistic	Percent
$\lvert(\hat{U} - U^*)\rvert \leq 0$	69.96
$\lvert(\hat{U} - U^*)\rvert \leq 1$	99.12
$\lvert(\hat{U} - U^*)\rvert \leq 2$	100.00

Table 7.1: Summary statistics for learned $\hat{U}$. Reports the percentage of pairs in $\hat{U}$ which have the same number of shared clusters as the same pair in U^*, or are off by at most 1 or 2 shared clusters.

Lastly, we used the IOMM to cluster movies by genre using the MovieLens data set of people rating movies. We normalized the data over movies such that the ratings for each movie summed to 1 and then binarized the matrix so that a (movie,user) entry was given a value 1 if the new rating value was greater than the mean of the values of all movies that user rated. We then removed users with less than 20 movies given value 1, and movies which less than 10 users assigned a value 1 to. This resulted in a binary matrix of 797 movies by 426 users from which we selected 500 movies at random. These 500 movies belonged to 18 different genres. Unfortunately, an unsupervised learning algorithm does not know what a genre is, and would be very unlikely to cluster movies in accordance with them unless we specify them in advance. In particular people's movie preferences are not simply correlated with genres, and there are many other latent factors which can determine preference (e.g. actors, budget, recency, script, etc.) Instead, we took a semi-supervised approach, randomly selecting 200 movies, fixing the Z matrix for those data points to their correct genres, and trying to learn the remaining 300 movies using the cluster information given by the fixed 200. We ran our IOMM sampler for 3000 iterations, burning in the first 1000 samples. If a movie was assigned to a genre in over half the sampled Z matrices, we said that the movie was assigned to that genre by the IOMM. We compared these IOMM results to two sets of results obtained by using a Dirichlet Process Mixture model (DPM), which can only assign each movie to a single genre. DPM inference was run semi-supervised on the same data set by replicating each of the 200 fixed movies m_i times, once for each of the m_i

Genre	# Movies	F1 IOMM	F1 DPM1	F1 DPM2
Drama	183	**0.4978**	0.2953	0.3046
Comedy	168	**0.6032**	0.5000	0.4962
Romance	81	**0.3030**	0.2581	0.2581
Action	78	0.5696	**0.6667**	**0.6667**
Thriller	72	**0.2737**	0.1404	0.1333
Adventure	50	**0.3091**	0.0000	0.0000
Children	46	0.3434	0.5714	**0.6047**
Horror	45	**0.7826**	0.6667	0.6780
Sci-Fi	38	**0.3256**	0.1000	0.0952
Crime	34	**0.2745**	0.1818	0.1818
Animation	21	**0.2667**	0.1429	0.1429

Table 7.2: The F1 scores for the IOMM versus the DPM by genre. The first column is the genre name, the second column is the number of movies in the data set which belong to that genre, the third column is the IOMM F1 score, the fourth column is the DPM1 F1 score, and the last column is the DPM2 F1 score for that genre.

genres they belong to. We compared the IOMM results to the DPM results using an F1 score ($F1 = \frac{2pr}{p+r}$, where p is precision and r is recall), which takes into account both precision and recall, and which can be computed from the true MovieLens assignments of movies to genres. The difference between the two sets of DPM results is that in DPM1 genre membership is decided in the same way as in the IOMM, thus allowing movies to belong to only one genre. In DPM2, we allow movies to belong to multiple genres by saying that a movie belongs to a genre if the movie was assigned to that genre in at least $M/(K+1)$ samples, where M is the total number of samples and K is the known *true* number of genres that movie actually belongs to. These results are presented in table 7.2, on the 11 genres with at least 10 movie members in the fixed set.

We can see that the IOMM has a better F1 score on 9 of the 11 genres, illustrating that the flexibility of assigning movies to multiple genres leads to better performance even when evaluating single genre membership. It is worth noting that the DPM in this case is fully conjugate and that we took care to integrate out all parameters, resulting in a sampler with much faster mixing. Despite this, the DPM was not able to capture the genre assignments as well as the IOMM.

7.7 Discussion

We presented a new nonparametric Bayesian method, the Infinite Overlapping Mixture Model, for modeling overlapping clusters. The IOMM extends traditional mixture models to allow data points membership in an unrestricted number of clusters, where the total number of clusters is itself unbounded. The IOMM uses products of models in the exponential family to model overlaps, allowing it to focus in on overlapping regions. We derived MCMC inference algorithms for the IOMM and applied it to the problem of clustering movies into genres, where we showed that its performance is superior to that of Dirichlet Process Mixtures, which restrict movies to a single genre. Our novel approach to discovering overlapping clusters should be applicable to data modeling problems in a wide range of fields.

Chapter 8

Bayesian Partial Membership Model

In this chapter we present a principled Bayesian framework for modeling *partial memberships* of data points to clusters. Unlike a standard mixture model which assumes that each data point belongs to one and only one mixture component, or cluster, a partial membership model allows data points to have fractional membership in multiple clusters. Algorithms which assign data points partial memberships to clusters can be useful for tasks such as clustering genes based on microarray data (Gasch and Eisen, 2002) and global positioning and orbit determination (Soto et al., 2007). Our Bayesian Partial Membership Model (BPM) uses exponential family distributions to model each cluster, and a product of these distibtutions, with weighted parameters, to model each datapoint. Here the weights correspond to the degree to which the datapoint belongs to each cluster. All parameters in the BPM are continuous, so we can use Hybrid Monte Carlo to perform inference and learning. We discuss relationships between the BPM and Latent Dirichlet Allocation, Mixed Membership models, Exponential Family PCA, and fuzzy clustering. Lastly, we show some experimental results and discuss nonparametric extensions to the model.

8.1 Partial Membership

Partial membership is the cornerstone of fuzzy set theory. While in traditional set theory, items either belong to a set or they don't, fuzzy set theory equips sets with a membership function $\mu_k(x)$ where $0 \leq \mu_k(x) \leq 1$ denotes the degree to which x partially belongs to set k.

The idea of partial membership is in fact quite intuitive and practically useful. Consider, for example, an individual with a mixed ethnic background, say, partly asian and partly white. It seems sensible to represent that individual as partly belonging to two different classes or sets. Such a partial membership representation may be relevant to predicting that individual's phenotype, or their food preferences. We clearly need models that can coherently represent partial membership.

Note that partial membership is conceptually very different from uncertain membership. Being certain that a person is partly asian and partly white, is very different that being uncertain about a person's ethnic background. More information about the person, such as DNA tests, could resolve uncertainty, but cannot make the person change his partial membership.

The notion that probabilistic models are unable to handle partial membership has been used to argue that probability is a subtheory of fuzzy logic, or that fuzzy logic is different in character from probability (Zadeh, 1965; Kosko, 1992). While it might be easy for many researchers in machine learning and statistics to dismiss fuzzy logic, fuzzy set theory, fuzzy clustering, and their myriad fuzzy derivatives, it is undeniable that the ability of this framework to represent partial membership has captured the imagination of many researchers. A literature search using Google Scholar reveals over 45,000 papers mentioning fuzzy clustering, and the classic papers in this field have been cited as many times as the most cited papers on topics usually covered in the NIPS community, such as Support Vector Machines, Bayesian methods, and neural networks.

In this chapter we describe a fully probabilistic approach to data modelling with partial membership. Our approach makes use of a simple way of representing partial membership using continuous latent variables. We define a model which can cluster data but which fundamentally assumes that data points can have partial membership in the clusters.

8.2 A Partial Membership Model

We can derive our method for modeling partial memberships from a standard finite mixture model. In a finite mixture model the probability of a data point, $\mathbf{x}_n$ given Θ, which contains the parameters for each of the K mixture components (clusters) is:

$$p(\mathbf{x}_n|\Theta) = \sum_{k=1}^{K} \rho_k p_k(\mathbf{x}_n|\boldsymbol{\theta}_k) \tag{8.1}$$

where p_k is the probability distribution of mixture component k, and ρ_k is the mixing proportion (fraction of data points belonging to) component k [1].

Equation (8.1) can be rewritten using indicator variables $\boldsymbol{\pi}_n = [\pi_{n1}\pi_{n2}\dots\pi_{nK}]$ as follows:

[1] This notation differs slightly from standard notation for mixture models.

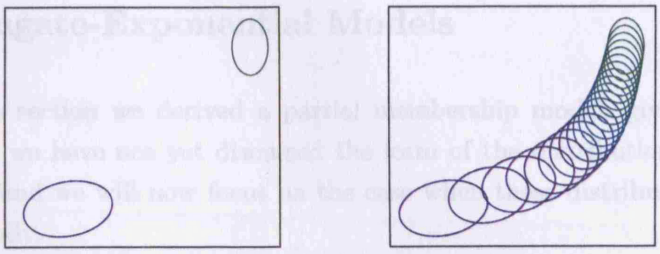


Figure 8.1: Left: A mixture model with two Gaussian mixture components, or clusters, can generate data from the two distributions shown. Right: Partial membership model with the same two clusters can generate data from all the distributions shown (there are actually infinitely many), which lie between the two original clusters.

$$p(\mathbf{x}_n|\Theta) = \sum_{\pi_n} p(\pi_n) \prod_{k=1}^{K} p_k(\mathbf{x}_n|\theta_k)^{\pi_{nk}} \tag{8.2}$$

where $\pi_{nk} \in 0, 1$ and $\sum_k \pi_{nk} = 1$. Here we can notice that if $\pi_{nk} = 1$ this means that data point n belongs to cluster k (and also $p(\pi_n) = \rho_k$). Therefore the π_{nk} denote *memberships* of data points to clusters.

In order to obtain a model for *partial memberships* we can relax the constraint $\pi_{nk} \in \{0, 1\}$ to now allow π_{nk} to take any continuous value in the range $[0, 1]$. However, in order to compute the probability of the data under this continuous relaxation of a finite mixture model, we need to modify equation (8.2) as follows:

$$p(\mathbf{x}_n|\Theta) = \int_{\pi_n} p(\pi_n) \frac{1}{c} \prod_{k=1}^{K} p_k(\mathbf{x}_n|\theta_k)^{\pi_{nk}} d\pi_n \tag{8.3}$$

The modifications include integrating over all values of $\boldsymbol{\pi}_n$ instead of summing, and since the product over clusters K from equation (8.2) no longer normalizes we put in a normalizing constant c, which is a function of $\boldsymbol{\pi}_n$ and Θ. Equation (8.3) now gives us a model for partial membership.

We illustrate the difference between our partial membership model and a standard mixture model in figure 8.1. Here we can see contours of the Gaussian distributions which can generate data in the mixture model (left) and the partial membership model (right), where both models are using the same two Gaussian clusters. As an example, if one of these clusters represents the ethnicity "White British" and the other cluster represents the ethnicity "Pakistani", then the figure illustrates that the partial membership model will be able to capture someone of mixed ethnicity, whose features may lie in between those of either ethic group (for example skin color or nose size), better than the mixture model.

8.3 Conjugate-Exponential Models

In the previous section we derived a partial membership model, given by equation (8.3). However we have not yet discussed the form of the distribution for each cluster, $p_k(\mathbf{x}_n|\boldsymbol{\theta}_k)$, and we will now focus on the case when these distributions are in the exponential family.

As described in chapter 2, an *exponential family distribution* can be written in the form:

$$p_k(\mathbf{x}_n|\boldsymbol{\theta}_k) = \exp\{\boldsymbol{s}(\mathbf{x}_n)^\top \boldsymbol{\theta}_k + h(\mathbf{x}_n) + g(\boldsymbol{\theta}_k)\} \tag{8.4}$$

where $\boldsymbol{s}(\mathbf{x}_n)$ is a vector depending on the data known as the *sufficient statistics*, $\boldsymbol{\theta}_k$ is a vector of *natural parameters*, $h(\mathbf{x}_n)$ is a function of the data, and $g(\boldsymbol{\theta}_k)$ is a function of the parameters which ensures that the probability normalizes to one when integrating or summing over $\mathbf{x}_n$. We will use the short-hand $\mathbf{x}_n \sim \text{Expon}(\boldsymbol{\theta}_k)$ to denote that $\mathbf{x}_n$ is drawn from an exponential family distribution with natural parameters $\boldsymbol{\theta}_k$.

If we plug the exponential family distribution (equation (8.4)) into our partial membership model (equation (8.3)) it follows that:

$$\mathbf{x}_n|\boldsymbol{\pi}_n, \Theta \sim \text{Expon}(\sum_k \pi_{nk}\boldsymbol{\theta}_k) \tag{8.5}$$

where $\mathbf{x}_n$ comes from the *same* exponential family distribution as the original clusters p_k, but with *new* natural parameters which are a convex combination of the natural parameters of the original clusters, $\boldsymbol{\theta}_k$, weighted by π_{nk}, the partial membership values for data point $\mathbf{x}_n$. Computation of the normalizing constant c is therefore always tractable when p_k is in the exponential family.

We will use the short-hand, $\boldsymbol{\theta} \sim \text{Conj}(\boldsymbol{\lambda}, \nu)$ to refer to a probability distribution which is *conjugate* to the exponential family distribution $p(\mathbf{x}_n|\boldsymbol{\theta}_k)$ and has the form:

$$p(\boldsymbol{\theta}) \propto \exp\{\boldsymbol{\lambda}^\top \boldsymbol{\theta} + \nu g(\boldsymbol{\theta})\} \tag{8.6}$$

where $\boldsymbol{\lambda}$ and ν are *hyperparameters* of the prior (see chapter 2 for more details).

Given the above conjugacy, it is easy to show that $p(\boldsymbol{\theta}|\mathbf{x}) = \text{Conj}(\boldsymbol{\lambda} + \boldsymbol{s}(\mathbf{x}), \nu + 1)$. In general, for a data set $\mathcal{D} = \{\mathbf{x}_n : n = 1 \ldots N\}$, we have $p(\boldsymbol{\theta}|\mathcal{D}) = \text{Conj}(\boldsymbol{\lambda} + \sum_n \boldsymbol{s}(\mathbf{x}_n), \nu + N)$. We now have the tools to define our Bayesian partial membership model.

8.4 Bayesian Partial Membership Models

Consider a model with K clusters, and a data set $\mathcal{D} = \{\mathbf{x}_n : n = 1 \ldots N\}$. Let $\boldsymbol{\alpha}$ be a K-dimensional vector of positive hyperparameters. We start by drawing mixture

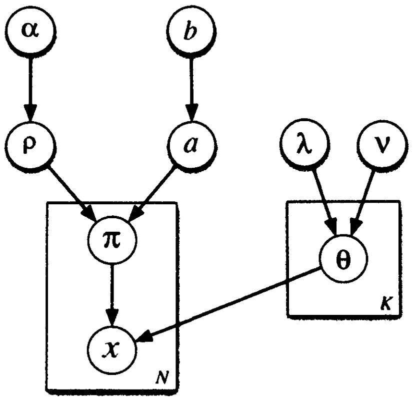


Figure 8.2: Graphical model for the BPM

weights from a Dirichlet distribution:

$$\rho \sim \text{Dir}(\alpha) \tag{8.7}$$

Here $\rho \sim \text{Dir}(\alpha)$ is shorthand for $p(\rho|\alpha) = c\prod_{k=1}^{K}\rho_k^{\alpha_k-1}$ where $c = \Gamma(\sum_k \alpha_k)/\prod_k \Gamma(\alpha_k)$ is a normalization constant which can be expressed in terms of the Gamma function[2]. For each data point, n we draw a partial membership vector π_n which represents how much that data point belongs to each of the K clusters:

$$\pi_n \sim \text{Dir}(a\rho). \tag{8.8}$$

The parameter a is a positive scaling constant drawn, for example, from an exponential distribution $p(a) = be^{-ba}$, where $b > 0$ is a constant. We assume that each cluster k is characterized by an exponential family distribution with natural parameters θ_k and that

$$\theta_k \sim \text{Conj}(\lambda, \nu). \tag{8.9}$$

Given all these latent variables, each data point is drawn from

$$\mathbf{x}_n \sim \text{Expon}(\sum_k \pi_{nk}\theta_k) \tag{8.10}$$

In order to get an intuition for what the functions of the parameters we have just defined are, we return to the ethnicity example. Here, each cluster k is an ethnicity (for example, "White British" and "Pakistani") and the parameters θ_k define a distribution over features for each of the k ethnic groups (for example, how likely it is that someone from that ethnic group likes pizza or marmite or bindi bhaji). The parameter ρ gives the ethnic composition of the population (for example, 75% "White British" and 25% "Pakistani"), while a controls how similar to the population an individual is expected

[2] The Gamma function generalizes the factorial to positive reals: $\Gamma(x) = (x-1)\Gamma(x-1)$, $\Gamma(n) = (n-1)!$ for integer n

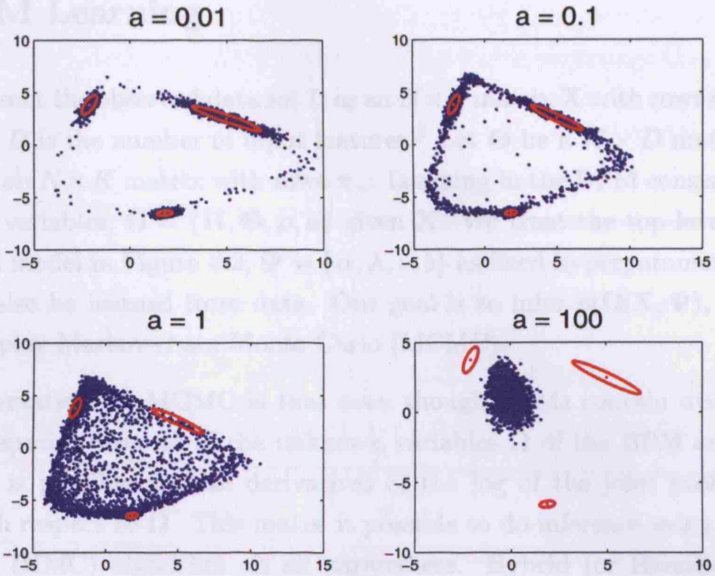


Figure 8.3: 3000 BPM generated data points with partial assignments to 3 Gaussian clusters shown in red, as parameter a varies.

to be (Are 100% of the people themselves 75% "White British" and 25% "Pakistani"? Or are 75% of the people 100% "White British" and the rest are 100% "Pakistani"? Or somewhere in between?). For each person n, π_n gives their individual ethnic composition, and finally $\mathbf{x}_n$ gives their individual feature values (e.g. how much they like marmite). The graphical model representing this generative process is drawn in Figure 8.2.

Since the Bayesian Partial Membership Model is a generative model, we tried generating data from it using full-covariance Gaussian clusters. Figure 8.3 shows the results of generating 3000 data points from our model with $K = 3$ clusters as the value of parameter a changes. We can see that as the value of a increases data points tend to have partial membership in more clusters. In fact we can prove the following lemmas:

Lemma 8.1 *In the limit that $a \to 0$ the exponential family BPM is a standard mixture model with K components and mixing proportions $\boldsymbol{\rho}$.*

Lemma 8.2 *In the limit that $a \to \infty$ the exponential family BPM model has a single component with natural parameters $\sum_k \rho_k \boldsymbol{\theta}_k$.*

Proofs of these lemmas follow simply from taking the limits of equation (8.8) as a goes to 0 and ∞ respectively.

8.5 BPM Learning

We can represent the observed data set $\mathcal{D}$ as an $N \times D$ matrix $\mathbf{X}$ with rows corresponding to $\mathbf{x}_n$, where D is the number of input features.[3] Let $\boldsymbol{\Theta}$ be a $K \times D$ matrix with rows $\boldsymbol{\theta}_k$ and $\boldsymbol{\Pi}$ be an $N \times K$ matrix with rows $\boldsymbol{\pi}_n$. Learning in the BPM consists of inferring all unknown variables, $\boldsymbol{\Omega} = \{\boldsymbol{\Pi}, \boldsymbol{\Theta}, \boldsymbol{\rho}, a\}$ given $\mathbf{X}$. We treat the top level variables in the graphical model in Figure 8.2, $\boldsymbol{\Psi} = \{\boldsymbol{\alpha}, \boldsymbol{\lambda}, \nu, b\}$ as fixed hyperparameters, although these could also be learned from data. Our goal is to infer $p(\boldsymbol{\Omega}|\mathbf{X}, \boldsymbol{\Psi})$, for which we decide to employ Markov chain Monte Carlo (MCMC).

Our key observation for MCMC is that even though BPMs contain discrete mixture models as a special case, *all* of the unknown variables $\boldsymbol{\Omega}$ of the BPM are continuous. Moreover, it is possible to take derivatives of the log of the joint probability of all variables with respect to $\boldsymbol{\Omega}$. This makes it possible to do inference using a full Hybrid Monte Carlo (HMC) algorithm on all parameters. Hybrid (or Hamiltonian) Monte Carlo is an MCMC procedure which overcomes the random walk behaviour of more traditional Metropolis or Gibbs sampling algorithms by making use of the derivatives of the log probability (Neal, 1993; MacKay, 2003). In high dimensions, this derivative information can lead to a dramatically faster mixing of the Markov chain, analogous to how optimization using derivatives is often much faster than using greedy random search.

We start by writing the probability of all parameters and variables[4] in our model:

$$p(\mathbf{X}, \boldsymbol{\Omega}|\boldsymbol{\Psi}) = p(\mathbf{X}|\boldsymbol{\Pi}, \boldsymbol{\Theta})p(\boldsymbol{\Theta}|\boldsymbol{\lambda}, \nu)p(\boldsymbol{\Pi}|a, \boldsymbol{\rho})p(a|b)p(\boldsymbol{\rho}|\boldsymbol{\alpha}) \tag{8.11}$$

We assume that the hyperparameter $\nu = 1$, and omit it from our derivation. Since the forms of all distributions on the right side of equation (8.11) are given in section 8.4 we can easily see that:

$$\begin{aligned}
\log p(\mathbf{X}, \boldsymbol{\Omega}|\boldsymbol{\Psi}) &= \log \Gamma(\sum_k \alpha_k) - \sum_k \log \Gamma(\alpha_k) + \sum_k (\alpha_k - 1) \log \rho_k \\
&\quad + \log b - ba + N \log \Gamma\left(\sum_k a\rho_k\right) - N \sum_k \log \Gamma(a\rho_k) \\
&\quad + \sum_n \sum_k (a\rho_k - 1) \log \pi_{nk} + \sum_k \left[\boldsymbol{\theta}_k^{\top} \boldsymbol{\lambda} + \mathrm{g}(\boldsymbol{\theta}_k) + \mathrm{f}(\boldsymbol{\lambda})\right] \\
&\quad + \sum_n \left[(\sum_k \pi_{nk} \boldsymbol{\theta}_k)^{\top} \mathbf{x}_n + \mathrm{h}(\mathbf{x}_n) + \mathrm{g}\left(\sum_k \pi_{nk} \boldsymbol{\theta}_k\right)\right]
\end{aligned} \tag{8.12}$$

The Hybrid Monte Carlo algorithm simulates dynamics of a system with continuous

[3] We assume that the data is represented in its natural representation for the exponential family likelihood, so that $s(\mathbf{x}_n) = \mathbf{x}_n$.

[4] A formal distinction between hidden variables, e.g. the $\{\boldsymbol{\pi}_n\}$, and unknown parameters is not necessary as they are both unknowns.

state $\mathbf{\Omega}$ on an energy function $\mathcal{E}(\mathbf{\Omega}) = -\log p(\mathbf{X}, \mathbf{\Omega}|\mathbf{\Psi})$. The derivatives of the energy function $\frac{\partial \mathcal{E}(\mathbf{\Omega})}{\partial \mathbf{\Omega})}$ provide forces on the state variables which encourage the system to find high probability regions, while maintaining detailed balance to ensure that the correct equilibrium distribution over states is achieved (Neal, 1993). Since $\mathbf{\Omega}$ has constraints, e.g. $a > 0$ and $\sum_k \rho_k = 1$, we use a tranformation of variables so that the new state variables are unconstrained, and we perform dynamics in this unconstrained space. Specifically, we use $a = e^{\eta}$, $\rho_k = \frac{e^{r_k}}{\sum_{k'} e^{r_{k'}}}$, and $\pi_{nk} = \frac{e^{p_{nk}}}{\sum_{k'} e^{p_{nk'}}}$. For HMC to be valid in this new space, the chain rule needs to be applied to the derivatives of $\mathcal{E}$, and the prior needs to be transformed through the Jacobian of the change of variables. For example, $p(a)da = p(\eta)d\eta$ implies $p(\eta) = p(a)(da/d\eta) = ap(a)$. We also extended the HMC procedure to handle missing inputs in a principled manner, by analytically integrating them out, as this was required for some of our applications. More details and general pseudocode for HMC can be found in MacKay (2003).

8.6 Related Work

The BPM model has interesting relations to several models that have been proposed in machine learning, statistics and pattern recognition. We describe these relationships here.

Latent Dirichlet Allocation: Using the notation introduced above, the BPM model and LDA (Blei et al., 2003) both incorporate a K-dimensional Dirichlet distributed $\boldsymbol{\pi}$ variable. In LDA, $\boldsymbol{\pi}_n$ are the mixing proportions of the topic mixture for each document n. Each word in document n can then be seen as having been generated by topic k, with probability π_{nk}, where the word distribution for topic k, is given by a multinomial distribution with some parameters, $\boldsymbol{\theta}_k$. The BPM also combines π_{nk} with some exponential family parameters $\boldsymbol{\theta}_k$, but here the way in which they are combined does not result in a mixture model from which another variable (e.g. a word) is assumed to be generated. In contrast, the data points are indexed by n directly, and therefore exist at the document level of LDA. Each data point is assumed to have come from an exponential family distribution parameterized by a weighted sum of natural parameters $\boldsymbol{\theta}$, where the weights are given by π_n for data point n. In LDA, data is organized at two levels (e.g. documents and words). More generally, mixed membership (MM) models (Erosheva et al., 2004), or admixture models, assume that each data attribute (e.g. words) of the data point (e.g. document) is drawn independently from a mixture distribution given the membership vector for the data point, $x_{nd} \sim \sum_k \pi_{nk} P(x|\theta_{kd})$. LDA and mixed membership models do not average natural parameters of exponential family distributions like the BPM. LDA or MM models could not generate the continuous densities in Fig 8.3 from full-covariance Gaussians. The analagous generative process for MM models is given in figure 8.4. Since data attributes are drawn

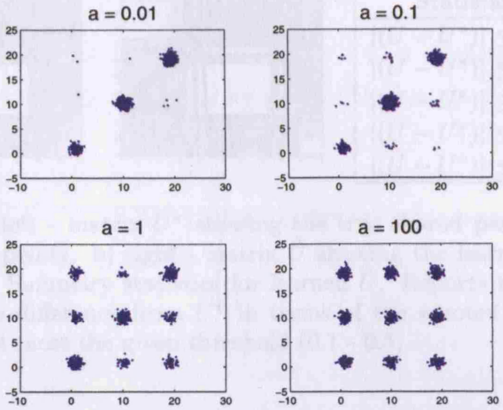
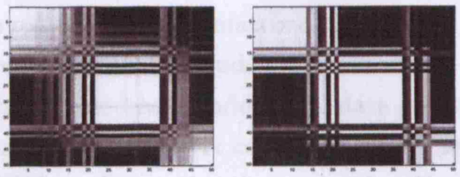


Figure 8.4: Generative plot for MM model with 3 Gaussian clusters

independently, the original clusters (not explicity shown) are one dimensional and have means at 0, 10 and 20 for both attribute dimensions. We can notice from the plot that this model always generates a mixture of 9 Gaussians, which is a very different behavior than the BPM, and clearly not as suitable for the general modeling of partial memberships. LDA only makes sense when the objects (e.g. documents) being modelled constitute bags of exchangeable sub-objects (e.g. words). Our model makes no such assumption. Moreover, in LDA and MM models there is a discrete latent variable for every sub-object corresponding to which mixture component that sub-object was drawn from. This large number of discrete latent variables makes MCMC sampling in LDA potentially much more expensive than in BPM models.

Exponential Family PCA: Our model bears an interesting relationship to Exponential Family PCA (Collins et al., 2002). EPCA was originally formulated as the solution to an optimization problem based on Bregman divergences, while our model is a fully probabilistic model in which all parameters can be integrated out via MCMC. However, it is possible to think of EPCA as the likelihood function of a probabilistic model, which coupled with a prior on the parameters, would make it possible to do Bayesian inference in EPCA and would render it closer to our model. However, our model was entirely motivated by the idea of partial membership in clusters, which is enforced by forming convex combinations of the natural parameters of exponential family models, while EPCA is based on *linear* combinations of the parameters. This has several consequences: EPCA does not naturally reduce to clustering, none of the variables can be interpreted as partial memberships, and the coefficients define a plane rather than a convex region in parameter space.

The recent work of Buntine and Jakulin (2006) focusing on the analysis of discrete data is also closely related to the BPM model. The framework of Buntine and Jakulin (2006) section III B expresses a model for discrete data in terms of linear mixtures of dual (or mean) exponential family parameters where MAP inference is performed. Section V B



Statistic	Percent
$\|(\hat{U} - U^*)\| \leq 0.1$	60.40
$\|(\hat{U} - U^*)\| \leq 0.2$	84.28
$\|(\hat{U} - U^*)\| \leq 0.3$	95.48
$\|(\hat{U} - U^*)\| \leq 0.4$	99.68
$\|(\hat{U} - U^*)\| \leq 0.5$	100.00

Figure 8.5: a) left - matrix U^* showing the true shared partial memberships for pairs of data points. b) right - matrix $\hat{U}$ showing the learned shared partial memberships. c) Summary statistics for learned $\hat{U}$. Reports the percentag e of pairs in $\hat{U}$ whose difference from U^* in terms of the amount of shared partial memberships is at most the given threshold (0.1 - 0.5).

also provides insights on differences between using dual and natural parameters.

Fuzzy Clustering: Fuzzy K-means clustering (Bezdek, 1981) iteratively minimizes the following objective: $J = \sum_{n=1}^{N} \sum_{k=1}^{K} \pi_{nk}^{\gamma} d^2(\mathbf{x}_n, \mathbf{c}_k)$, where $\gamma > 1$ is an exponent parameter, π_{nk} represents the degree of membership of data point n in cluster k ($\sum_k \pi_{nk} = 1$), and $d^2(\mathbf{x}_n, \mathbf{c}_k)$ is a measure of squared distance between data point $\mathbf{x}_n$ and cluster center $\mathbf{c}_k$. By varying γ it is possible to attain different amounts of partial membership, where the limiting case $\gamma = 1$ is K-means with no partial membership. Although the π parameters represent partial membership, none of the variables have probabilistic interpretations.

8.7 Experiments

We generated a synthetic binary data set from the BPM, and used this to test BPM learning. The synthetic data set had 50 data points which each have 32 dimensions and can hold partial memberships in 3 clusters. We ran our Hybrid Monte Carlo sampler for 4000 iterations, burning in the first half. In order to compare our learned partial membership assignments for data points (Π_L) to the true ones (Π_T) for this synthetic data set, we compute ($\hat{U} = \Pi_L \Pi_L^\top$) and ($U^* = \Pi_T \Pi_T^\top$), which basically give the total amount of cluster membership shared between each pair of data points, and is invariant to permutations of cluster labels. Both of these matrices can be seen in figure 8.5. One can see that the structure of these two matrices is quite similar, and that the BPM is learning the synthetic data reasonably. For a more quantitative measure table 8.5c gives statistics on the number of pairs of data points whose learned shared membership differs from the true shared membership by more than a given threshold (the range of this statistic is [0,1]).

We also used the BPM to model two "real-world" data sets. The first is senate roll call data from the 107th US congress (2001-2002) (Jakulin, 2004), and the second is a data set of images of sunsets and towers.

The senate roll call data is a matrix of 99 senators (one senator died in 2002 and neither he or his replacement is included) by 633 votes. It also includes the outcome of each vote, which is treated as an additional data point (like a senator who always voted the actual outcome). The matrix contained binary features for yea and nay votes, and we used the BPM to cluster this data set using $K = 2$ clusters. There are missing values in this dataset but this can easily be dealt with in the HMC log probability calculations by explicitly representing both 0 and 1 binary values and leaving out missing values. The results are given in figure 8.6. The line in figure 8.6 represents the amount of membership of each senator in one of the clusters (we used the "Democrat" cluster, where senators on the far left have partial memberships very close to 0, and those on the far right have partial memberships extremely close to 1). Since there are two clusters, and the amount of membership always sums to 1 across clusters, the figure looks the same regardless of whether we are looking at the "Democrat" or "Republican" cluster. We can see that most Republicans and Democrats are tightly clustered at the ends of the line (and have partial memberships very close to 0 and 1), but that there is a fraction of senators (around 20%) which lies somewhere reasonably in between the extreme partial memberships of 0 or 1. Interesting properties of this figure include the location of Senator Jeffords who left the Republican party in 2001 to become an independent who caucused with the Democrats. Also Senator Chafee who is known as a moderate Republican and who often voted with the Democrats (for example, he was the only Republican to vote against authorizing the use of force in Iraq), and Senator Miller a conservative Democrat who supported George Bush over John Kerry in the 2004 US Presidential elections. Lastly, it is interesting to note the location of the Outcome data point, which is very much in the middle. This makes sense since the 107th congress was split 50-50 (with Republican Dick Cheney breaking ties), until Senator Jeffords became an Independent at which point the Democrats had a one seat majority.

We also tried running both fuzzy k-means clustering and Dirichlet Process Mixture models (DPMs) on this data set. While fuzzy k-means found roughly similar rankings of the senators in terms of membership to the "Democrat" cluster, the exact ranking and, in particular, the amount of partial membership (π_n) each senator had in the cluster was *very* sensitive to the fuzzy exponent parameter, which is typically set by hand. Figure 8.7a plots the amount of membership for the Outcome data point in black, as well as the most extreme Republican, Senator Ensign, in red, and the most extreme Democrat, Senator Schumer, in blue, as a function of the fuzzy exponent parameter. We can see in this plot that as the assignment of the Outcome data point begins to reach a value even reasonably close to 0.5, the most extreme Republican already has 20% membership in the "Democrat" cluster. This reduction in range does not make sense semantically, and presents a trade-off between finding reasonable values for π_n in the middle of the range, versus at the extremes. This kind of sensitivity to parameters does not exist in our BPM model, which models both extreme and middle range values

Algorithm	Mean	Median	Min	Max	"Outcome"
BPM	**187**	**168**	**93**	422	**224**
DPM	196	178	112	**412**	245

Table 8.1: Comparison between the BPM and a DPM in terms of negative log predictive probability (in bits) across senators.

well.

We tried using a DPM to model this data set where we ran the DPM for 1000 iterations of Gibbs sampling, sampling both assignments and concentration parameter. The DPM confidently finds 4 clusters: one cluster consists solely of Democrats, one consists solely of Republicans, the third cluster has 9 of the most moderate Democrats and Republicans plus the "vote outcome" variable, and the last cluster has just one member, Hollings (D-SC). Figure 8.7b is a 100x100 matrix showing the overlap of cluster assignments for pairs of senators, averaged over 500 samples (there are no changes in relative assignments, the DPM is completely confident). The interpretation of the data provided by the DPM is very different from the BPM model's. The DPM does *not* use uncertainty in cluster membership to model Senators with intermediate views. Rather, it creates an entirely new cluster to model these Senators. This makes sense for the data as viewed by the DPM: there is ample data in the roll calls that these Senators are moderate — it is not the case that there is uncertainty about whether they fall in line with hard- core Democrats or Republicans. This highlights the fact that the responsibilities in a mixture model (such as the DPM) cannot and should not be interpreted as partial membership, they are representations of *uncertainty* in full membership. The BPM model, however, explicitly models the partial membership, and can, for example, represent the fact that a Senator might be best characterized as moderate (and quantify how moderate they are). In order to quantify this comparison we calculated the negative log predictive probability (in bits) across senators for the BPM and the DPM (Table 8.1). We look at a number of different measures: the mean, median, minimum and maximum number of bits required to encode a senator's votes. We also look at the number of bits needed to encode the "Outcome" in particular. On all of these measures except for maximum, the BPM performs better than the DPM, showing that the BPM is a superior model for this data set.

Lastly, we used the BPM to model images of sunsets and towers. The dataset consisted of 329 images of sunsets or towers, each of which was represented by 240 binary simple texture and color features (see chapter 5). Partial assignments to $K = 2$ clusters were learned, and figure 8.8 provides an illustrative result. The top row of the figure has the three images with the most membership in the "sunset" cluster, the bottom row contains the three images with the most membership in the "tower" cluster, and the middle row shows the 3 images which have closest to 50/50 membership in each cluster ($\pi_{nk} \approx 0.5$). In this dataset, as well as all the datasets described in this section, our

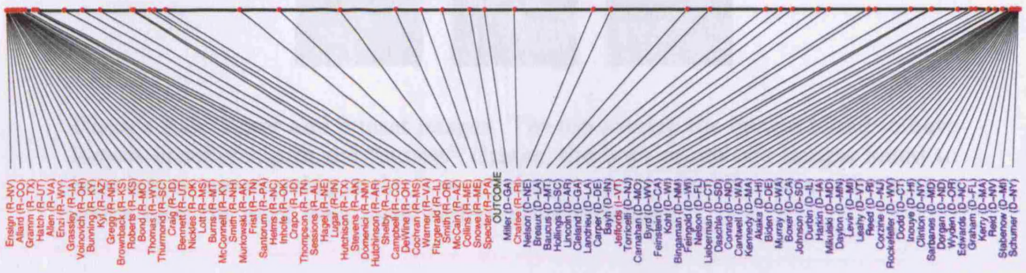


Figure 8.6: Analysis of the partial membership results on the Senate roll call data from 2001-2002. The line shows amount of membership in the "Democrat" cluster with the left of the line being the lowest and the right the highest.

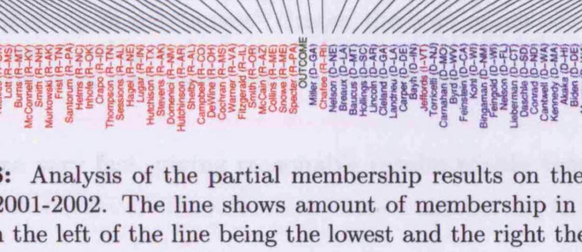


Figure 8.7: a) left - fuzzy k-means: plot of the partial membership values for the Outcome data point (in black) and the most extreme Republican (in red) and Democrat (in blue) as a function of the fuzzy exponent parameter. b) right - DPMs: an ordered 100x100 matrix showing the fraction of times each pair of senators was assigned to the same cluster, averaged over 500 Gibbs sampling iterations.

Figure 8.8: Tower and Sunset images. The top row are the images found to have largest membership in the "sunset" cluster, the bottom row are images found to have largest membership in the "tower" cluster, and the middle row are the images which have the most even membership in both clusters.

HMC sampler was very fast, giving reasonable results within tens of seconds.

8.8 Conclusions and Future Work

In summary, we have described a fully probabilistic approach to data modelling with partial membership using continuous latent variables, which can be seen as a relaxation of clustering with finite mixture models. We employed a full Hybrid Monte Carlo algorithm for inference, and our experience with HMC has been very positive. Despite the general reputation of MCMC methods for being slow, our model using HMC seems to discover sensible partial membership structure after surprisingly few samples.

In the future we would like to develop a nonparametric version of this model. The most obvious way to try to generalize this model would be with a Hierarchical Dirichlet Process (Teh et al., 2006). However, this would involve averaging over infinitely many potential clusters, which is both computationally infeasible, and also undesirable from the point of view that each data point should have non-zero partial membership in only a few (certainly finite) number of clusters. An more promising alternative is to use an Indian Buffet Process (Griffiths and Ghahramani, 2005), where each 1 in a row in an IBP sample matrix would represent a cluster in which the data point corresponding to that row has non-zero partial membership, and then draw the continuous values for those partial memberships conditioned on that IBP matrix.

Chapter 9

Summary and future work

This thesis has presented several Bayesian methods for clustering data and extensions of the clustering paradigm. We have strived to make these methods as efficient as possible so that they may be run on larger scale data sets. Chapter 3 presents a novel Bayesian method for performing hierarchical cluster with the same efficiency as traditional hierarchical clustering algorithms. We also show that this method can be used as a fast approximation for doing inference in Dirichlet Process Mixture models. In chapter 4 we describe an algorithm for retrieving information based on queries in the form of sets of examples. Bayesian "clustering on demand" can be used to retrieve items, often times by merely computing a sparse matrix-vector product. Chapter 5 applies the ideas presented in chapter 4 to performing content-based image retrieval. Chapter 6 extends this work to discriminative learning, addressing the problem of automated analogical reasoning. We present a nonparametric Bayesian method for performing overlapping clustering in chapter 7, which allows an unbounded number of clusters and assignments to clusters. Lastly, chapter 8 describes a Bayesian partial membership model for modeling the partial memberships of data points to clusters. This model can be derived from a continuous relaxation of a standard mixture model, which allows Hybrid Monte Carlo to be used to perform more efficient inference.

There are many exciting directions for future research. It would be interesting to further explore nonparametric Bayesian methods, particularly as they pertain to developing richer, more complex methods for unsupervised learning at data modeling. There appears to be a growing interest in the further incorporation of probabilistic inference into information retrieval, which often demands efficient algorithms, and this would also be an interesting area for future work. In chapter 3 we presented a combinatorial lower bound on the marginal likelihood of a DPM, and in future research I would be interested in further exploring these type of combinatorial lower bounds, possibly for other nonparametric Bayesian methods, or possibly to improve the current bound for DPMs.

Bibliography

Amazon. http://www.amazon.com.

Behold image search online. http://photo.beholdsearch.com/.

eBay. http://www.ebay.com.

Google. http://www.google.com.

Google sets. http://labs.google.com/sets.

PubMed. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=pubmed.

UniProt. http://www.pir.uniprot.org/.

D. Aldous. Exchangeability and related topics. In *l'cole d't de probabilits de Saint-Flour, XIII*, 1983. (page 24)

J. Assfalg, A. D. Bimbo, and P. Pala. Using multiple examples for content-based image retrieval. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2000)*, 2000. (page 75)

A. Banerjee, C. Krumpelman, S. Basu, R. Mooney, and J. Ghosh. Model based overlapping clustering. In *International Conference on Knowledge Discovery and Data Mining KDD*, 2005. (page 101)

J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, 1993. (pages 23 and 47)

A. Battle, E. Segal, and D. Koller. Probabilistic discovery of overlapping cellular processes and their regulation. *Journal of Computational Biology*, 12(7):909–927, 2005. (page 101)

J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* kluwer, 1981. (page 115)

C. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006. (page 93)

Blackwell and MacQueen. Ferguson distributions via Polya urn schemes. *Ann. Stats.*, 1973. (page 31)

D. Blei. Probabilistic models of text and images. *PhD. Thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley*, 2004. (page 23)

D. Blei and M. Jordan. Variational methods for Dirichlet process mixture models. Technical Report 674, UC Berkeley, 2004. (page 48)

D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *JMLR*, 2003. (page 113)

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *WWW7/Computer Networks*, 30(1–7):107–117, 1998. (page 49)

W. Buntine and A. Jakulin. Discrete component analysis. *LNCS*, 3940, 2006. (page 114)

H. Chipman, E. George, and R. McCulloch. Bayesian CART model search (with discussion). *Journal of the American Statistical Association*, 93:935–960, 1998. (page 47)

M. Collins, S. Dasgupta, and R. Schapire. A generalization of principal components analysis to the exponential family. In *NIPS*, 2002. (page 114)

P. Cowans. *Probabilistic Document Modeling.* PhD thesis, University of Cambridge, 2006. (page 50)

I. Cox, M. Miller, T. Minka, T. Papathornas, and P. Yianilos. The Bayesian image retrieval system, pichunter: Theory, implementation, and psychophysical experiments. *IEEE Tran. On Image Processing*, 9:20–37, 2000. URL `citeseer.ist.psu.edu/article/cox00bayesian.html`. (page 74)

R. Cox. Probability, frequency, and reasonable expectation. *Am. Jour. Phys.*, 14:1–13, 1946. (page 16)

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. *Proceedings of AAAI'98*, pages 509–516, 1998. (page 84)

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977. (page 21)

D. Denison, C. Holmes, B. Mallick, and A. Smith. *Bayesian Methods for Nonlinear Classification and Regression.* Wiley, 2002. (page 47)

R. Duda, P. Hart, and D. Stork. *Pattern Classification.* Wiley, 2001. (page 36)

R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* Wiley, 1973. (pages 12 and 27)

S. Džeroski and N. Lavrač. *Relational Data Mining.* Springer, 2001. (page 81)

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci*, 95:14863–8, 1998. (page 12)

M. Ernst and M. Banks. Humans integrate visual and haptic information in a statistically optimal way. *Nature*, 415:429–433, 2002. (page 15)

E. Erosheva, S. Fienberg, and J. Lafferty. Mixed membership models of scientific publications. *PNAS*, 101, 2004. (page 113)

M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P.Yanker. Query by image and video content: the qbic system. *IEEE Computer*, 28:23–32, 1995. (page 73)

R. French. The computational modeling of analogy-making. *Trends in Cognitive Sciences*, 6, 2002. (page 78)

N. Friedman. Pcluster: Probabilistic agglomerative clustering of gene expression profiles. Technical Report 2003-80, Herbew University, 2003. (page 47)

A. Gasch and M. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol.*, 3(11), 2002. (page 106)

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *JMLR*, 3:679–707, 2002. (page 78)

Z. Ghahramani. Factorial learning and the EM algorithm. In *Adv. in Neural Information Processing Systems NIPS*, 1995. (pages 94 and 100)

Z. Ghahramani and M. Beal. Variational inference for Bayesian mixture of factor analysers. *Advances in Neural Information Processing Systems*, 12, 1999. (page 22)

Z. Ghahramani and K. Heller. Bayesian sets. *18th NIPS*, 2005. (pages 66, 78, 81, 84, and 88)

D. Görür, Jäkel, and C. Rasmussen. A choice model with inifinitely many latent features. In *International Conference on Machine Learning ICML*, 2006. (pages 25 and 97)

P. H. Gosselin and M. Cord. A comparison of active classification methods for content-based image retrieval. In *First International Workshop on Computer Vision meets Databases (CVDB 2004)*, 2004. URL `citeseer.ist.psu.edu/730059.html`. (page 74)

T. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. Technical report, Gatsby Computational Neuroscience Unit, 2005. (pages 97, 98, and 119)

T. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. In *Adv. in Neural Information Processing Systems NIPS*, 2006. (pages 24, 25, 92, 93, and 96)

J. Hartigan and M. Wong. A k-means clustering algorithm. *Applied Statistics*, 28(1): 100–108, 1979. (pages 12 and 20)

D. Heesch, M. Pickering, S. Rüger, and A. Yavlinsky. Video retrieval with a browsing framework using key frames. In *Proceedings of TRECVID*, 2003. (page 65)

K. Heller and Z. Ghahramani. A nonparametric bayesian approach to modeling overlapping clusters. In *AISTATS*, 2007.

K. A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. In *ICML*, 2005a.

K. A. Heller and Z. Ghahramani. Randomized algorithms for fast Bayesian hierarchical clustering. In *PASCAL Workshop on Statistics and Optimization of Clustering*, 2005b.

K. A. Heller and Z. Ghahramani. A simple Bayesian framework for content-based image retrieval. In *CVPR*, 2006.

G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14, 2002. (pages 92, 94, and 100)

G. Hinton, P. Dayan, B. Frey, and R. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995. (page 15)

G. E. Hinton and R. Zemel. Autoencoders, minimum description length, and helmholtz free energy. In *Adv. in Neural Info. Proc. Systems NIPS*, 1994. (page 100)

S. C. H. Hoi and M. R. Lyu. A semi-supervised active learning framework for image retrieval. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, 2005. (page 75)

P. Howarth and S. Rüger. Evaluation of texture features for content-based image retrieval. In *International Conference on Image and Video Retrieval (CIVR)*, 2004. (page 65)

M. Iwayama and T. Tokunaga. Hierarchical Bayesian clustering for automatic text classification. In C. E. Mellish, editor, *Proceedings of IJCAI-95, 14th International Joint Conference on Artificial Intelligence*, pages 1322–1327, Montreal, CA, 1995. Morgan Kaufmann Publishers, San Francisco, US. (page 47)

T. Jaakkola and M. Jordan. Bayesian parameter estimation via variational bounds. *Statistics and Computing*, 10:25–37, 2000. (pages 79 and 80)

A. Jakulin, 2004. URL `http://www.ailab.si/aleks/politics/`. (page 115)

E. Jaynes. *Probability Theory: The Logic of Science.* Cambridge University Press, 2003. (page 16)

D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. *Proceedings of ICML*, 2002. (page 81)

S. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2:241–254, 1967. (pages 12 and 20)

C. Kemp, T. L. Griffiths, S. Stromsten, and J. B. Tenenbaum. Semi-supervised learning with trees. In *NIPS 16*, 2004. (page 47)

C. Kemp, J. Tenenbaum, T. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. *Proceedings of AAAI'06*, 2006. (page 78)

D. Knill and A. Pouget. The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neuroscience*, 27(12):712–719, 2004. (page 15)

D. Knill and W. Richards, editors. *Perception as Bayesian Inference.* Cambridge University Press, 1996. (page 15)

J. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann, 1993. (page 91)

K. Körding and D. Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427:244–247, 2004. (page 15)

B. Kosko. *Neural Networks and Fuzzy Systems.* Prentice-Hall, 1992. (page 107)

J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. In *Language Modeling for Information Retrieval*, volume 13 of *Kluwer International Series on Information Retrieval.* Kluwer, 2003. (page 50)

X. Lu, M. Hauskrecht, and R. Day. Modeling cellular processes with variational bayesian cooperative vector quantizer. In *Proceedings of the Pacific Symposium on Biocomputing PSB*, 2004. (page 100)

D. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003. (pages 112 and 113)

C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* In press, 2007. (pages 49 and 82)

Z. Marx, I. Dagan, J. Buhmann, and E. Shamir. Couple clustering: a method for detecting structural correspondence. *JMLR*, 3:747–780, 2002. (page 78)

A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/~mccallum/bow, 1996. (page 37)

G. McLachlan and D. A. Peel. *Finite Mixture Models.* Wiley, 2000. (pages 12 and 20)

E. Meeds, Z. Ghahramani, S. Roweis, and R. Neal. Modeling dyadic data with binary latent factors. In *Adv. in Neural Information Processing Systems NIPS*, 2007. (page 99)

M. Meila and J. Shi. Learning segmentation with random walk. In *Neural Information Processing Systems*, 2001. (pages 12 and 20)

R. Memisevic and G. Hinton. Multiple relational embedding. *18th NIPS*, 2005. (page 78)

H. Mewes and C. Amid. Mips: analysis and annotation of proteins from whole genome. *Nucleic Acids Research*, 2004. (page 87)

T. Minka and Z. Ghahramani. Expectation propagation for infinite mixtures. In *NIPS Workshop on Nonparametric Bayesian Methods and Infinite Models*, 2003. (page 48)

R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995. (page 45)

R. Neal. Probabilistic inference using markov chain monte carlo methods. Technical report, University of Toronto, 1993. (pages 112 and 113)

R. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000. (pages 22, 23, 24, and 96)

R. M. Neal. Density modeling and clustering using dirichlet diffusion trees. In *Bayesian Statistics 7*, pages 619–629, 2003. (page 47)

A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002. (pages 12 and 20)

R. Nosofsky. Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, 115(1):39–57, 1986. (page 50)

J. Pearl. *Probabilistic Reasoning in Expert Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988. (page 25)

D. Pelleg and A. Moore. Accelerating exact k -means algorithms with geometric reasoning. In *KDD*, 1999. (page 12)

J. Ponte and W. Croft. A language modeling approach to information retrieval. In *SIGIR*, 1998. (page 50)

A. Popescul and L. H. Ungar. Structural logistic regression for link analysis. *Multi-Relational Data Mining Workshop at KDD-2003*, 2003. (pages 78 and 81)

Y. Qi, Z. Bar-Joseph, and J. Klein-Seetharaman. Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Bioinformatics*, 63:490–500, 2006. (pages 87 and 88)

M. F. Ramoni, P. Sebastiani, and I. S. Kohane. Cluster analysis of gene expression dynamics. *Proc Nat Acad Sci*, 99(14):9121–9126, 2002. (page 47)

C. E. Rasmussen. The infinite Gaussian mixture model. In *NIPS 12*, pages 554–560, 2000. (pages 23, 31, and 48)

S. Richardson and P. Green. On bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society*, 1997. (page 23)

S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *J. Amer. Soc. Information Science*, 27(3):129–146, 1976. (page 50)

S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 1997. (page 21)

Y. Rui, T. Huang, and S. Mehrotra. Content-Based image retrieval with relevance feedback in MARS. In *Proceedings of IEEE International Conference on Image Processing*, pages 815–818, 1997. URL `citeseer.ist.psu.edu/rui97contentbased.html`. (page 74)

M. Sahami, M. A. Hearst, and E. Saund. Applying the multiple cause mixture model to text categorization. In *International Conference on Machine Learning ICML*, 1996. (page 100)

G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983. (page 49)

E. Saund. Unsupervised learning of mixtures of multiple causes in binary data. In *Adv. in Neural Info. Proc. Systems NIPS*, 1994. (pages 94 and 100)

B. Schatz. Information retrieval in digital libraries: Bringing search to the net. *Science*, 275:327–334, 1997. (page 49)

E. Segal, D. Koller, and D. Ormoneit. Probabilistic abstraction hierarchies. In *NIPS 14*, 2002. (page 47)

E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. In *Proceedings of the Pacific Symposium on Biocomputing*, 2003. (page 101)

R. N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987. (page 50)

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. (pages 12 and 20)

R. Silva, E. Airoldi, and K. Heller. Small sets of interacting proteins suggest latent linkage mechanisms through analogical reasoning. *Gatsby Technical Report, GCNU TR 2007-001*, 2007a. (page 86)

R. Silva, K. Heller, and Z. Ghahramani. Analogical reasoning with relational bayesian sets. In *AISTATS*, 2007b. (pages 76, 77, 78, and 81)

A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12), 2000. (page 73)

J. Soto, M. Aguiar, and A. Flores-Sintas. A fuzzy clustering application to precise orbit determination. *Journal of Computational and Applied Mathematics*, 204, 2007. (page 106)

A. Stolcke and S. Omohundro. Hidden Markov Model induction by bayesian model merging. In *NIPS 5*, pages 11–18, 1993. (page 47)

H. Tamura, S. Mori, and T. Yamawaki. Textual features corresponding to visual perception. *IEEE Trans on Systems, Man and Cybernetics*, 8:460–472, 1978. (page 65)

R. Tatusov, E. Koonin, and D. Lipman. A genomic perspective on protein families. *Science*, 278(5338):631–637, 1997. `http://www.ncbi.nlm.nih.gov/COG/`. (page 57)

Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *JASA*, 101 (476), 2006. (pages 23 and 119)

J. Tenenbaum and T. Griffiths. Generalization, similarity, and Bayesian inference. *Behavioral and Brain Sciences*, 24:629–640, 2001. (pages 50 and 51)

P. Turney and M. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning Journal*, 60:251–278, 2005. (page 78)

A. Tversky. Features of similarity. *Psychological Review*, 89:123–154, 1977. (page 50)

S. Vaithyanathan and B. Dom. Model-based hierarchical clustering. In *UAI*, 2000. (page 47)

N. Vasconcelos. Minimum probability of error image retrieval. *IEEE Transactions on Signal Processing*, 52(8), 2004. (page 73)

N. Vasconcelos and A. Lippman. A bayesian framework for content-based indexing and retrieval. In *Proceedings of IEEE Data Compression Conference*, 1998. (page 73)

D. J. Ward. *Adaptive Computer Interfaces.* PhD thesis, University of Cambridge, 2001. (page 47)

C. Williams. A MCMC approach to hierarchical mixture modelling. In *NIPS 12*, 2000. (page 47)

F. Wood, T. Griffiths, and Z. Ghahramani. A non-parametric Bayesian method for inferring hidden causes. In *UAI*, 2006. (page 25)

A. Yavlinsky, E. Schofield, and S. Rüger. Automated image annotation using global features and robust nonparametric density estimation. In *Proceedings of the International Conference on Image and Video Retrieval*, 2005. (page 74)

L. Zadeh. Fuzzy sets. *Information and Control*, 8, 1965. (page 107)

J. Zhang, Z. Ghahramani, and Y. Yang. A probabilistic model for online document clustering with application to novelty detection. In *NIPS*, 2004. (page 12)

X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data*, 2003. URL `citeseer.ist.psu.edu/zhu03combining.html`. (page 75)