Bayesian Mixture Modeling by Monte Carlo Simulation

Radford M. Neal

Technical Report CRG-TR-91-2
Department of Computer Science
University of Toronto

e-mail: radford@ai.toronto.edu

June, 1991

Abstract. It is shown that Bayesian inference from data modeled by a mixture distribution can feasibly be performed via Monte Carlo simulation. This method exhibits the true Bayesian predictive distribution, implicitly integrating over the entire underlying parameter space. An infinite number of mixture components can be accommodated without difficulty, using a prior distribution for mixing proportions that selects a reasonable subset of components to explain any finite training set. The need to decide on a "correct" number of components is thereby avoided. The feasibility of the method is shown empirically for a simple classification task.

Introduction

Mixture distributions [8, 20] are an appropriate tool for modeling processes whose output is thought to be generated by several different underlying mechanisms, or to come from several different populations. One aim of a mixture model analysis may be to identify and characterize these underlying "latent classes" [2, 7], either for some scientific purpose, or as one realization of "unsupervised learning" in artificial intelligence. In other cases, prediction of future observations is the objective. In a "classification" application [6], for example, we are interested in predicting the category attribute of an item on the basis of various indicator attributes.

The standard Bayesian approach to this problem would be to define a prior distribution over the parameter space of the mixture model and combine this with the observed data to give a posterior distribution over this parameter space. The posterior distribution would then be interpreted for latent class analysis, or used to derive a predictive distribution for use in classification. Unfortunately, since the parameter space is extremely large, this approach is computationally difficult. As discussed later, one current approach is to find the single parameter set with maximum posterior probability and use it as the basis for prediction and interpretation. While this procedure often produces good results, it is not the proper Bayesian solution to the problem, and for some prior distributions gives useless answers.

In this paper, I present a technique for exhibiting the true Bayesian predictive distribution, given a conjugate prior for the parameters and a set of training items. In this method, the model parameters are first integrated out analytically, reducing the problem to a summation over all possible assignments of mixture components to data items. This still-formidable problem is then solved by the Monte Carlo simulation technique of "Gibbs Sampling".

It turns out that this procedure extends without any computational difficulty to models in which the number of mixture components is countably infinite. Such infinite mixture models are quite natural in many contexts. Consider, for example, an analysis of plant specimens collected at random from some region. We expect *a priori* that the region will harbour many thousands of plant species, and that furthermore there will be many distinct populations within a species, due to varying soil conditions, etc. It therefore seems reasonable to consider the number of latent classes in this example to be effectively infinite. Of course, any finite sample will contain representatives of only a finite number of these classes. Indeed, we will generally wish to explain the sample using many fewer classes than there are data items. As more data is collected, however, more and more of the infinite number of classes will become apparent, as, for example, we obtain significant numbers of specimens of the rarer species.

The empirical results presented in this paper demonstrate that modeling data as an infinite mixture also works well when there are only a small finite number of components in the true mixture. Infinite mixture models are thus an attractive option whenever the true number of components is unknown, since one thereby avoids the problem of selecting between models with different numbers of components. A similar idea has been applied to the problem of modeling data by polynomials of indefinite degree in [22].

Below, I will define the Bayesian formulation of the mixture modeling problem, and describe how it can be solved by Monte Carlo simulation. Following this, I briefly discuss other mixture modeling approaches, and then present the results of applying the Bayesian mixture modeling technique and some of these other methods to a simple classification task. Finally, I will discuss some applications of the Bayesian method, and its potential extension to more general models.

Bayesian mixture modeling

In formalizing the Bayesian mixture modeling task, I will take prediction to be the primary objective. This is natural in a classification application. In a latent class analysis application, we might not have any real interest in predicting values of unknown attributes, but nevertheless it is by their predictive value that the validity of underlying classes is demonstrated.

The prediction problem. Assume that some process produces data items represented by attribute vectors $\widetilde{v}_i = \langle v_{i1}, \ldots, v_{im} \rangle$, which are seen as realizations of corresponding random variables $\widetilde{V}_i = \langle V_{i1}, \ldots, V_{im} \rangle$. In this paper, the range of attribute V_{ij} is taken to be the set of integers $\{1, \ldots, N_j\}$, but generalizations to real-valued attributes are possible.

We have some knowledge of, or interest in, n of these vectors: $\widetilde{V}_1, \ldots, \widetilde{V}_n$. Specifically, we wish to find the predictive distribution for one or more of the unknown attributes of these vectors, given the values of the known attributes.

In a classification application, V_{i1} might be the category of item i and $V_{i2}, \ldots, V_{im}$ be various indicator attributes. Perhaps we know the indicators and category of the first $n-1$ items (the training set) and have seen the indicators for item n (the test item). We wish to find the consequent probability that item n is in category c:

$$P(V_{n1} = c \mid \widetilde{V}_i = \widetilde{v}_i,\ V_{nj} = v_{nj}\ :\ 1 \le i < n,\ 2 \le j \le m) \tag{1}$$

The mixture model. Assume that the parameters of the process generating the data vectors are stable, and that given knowledge of these parameters, the various data items, $\widetilde{V}_i$, are independent and distributed identically. In the mixture model approach, we believe, or imagine, that the process generates data items by several different mechanisms — which mechanism being a random variable, G_i — and that given knowledge of which mechanism generated a given item, the various attributes are independent. In other words, the distribution for $\widetilde{V}_i$, on the assumption that the process parameters are known, can be expressed as a mixture of component distributions as follows:

$$\begin{aligned} P(\widetilde{V}_i = \widetilde{v}_i) &= \sum_{g=1}^{M} P(G_i = g) \cdot \prod_{j=1}^{m} P(V_{ij} = v_{ij} \mid G_i = g) \\ &= \sum_{g=1}^{M} \phi_g \cdot \prod_{j=1}^{m} \psi_{g,j,v_{ij}} \end{aligned} \tag{2}$$

Here, M is the number of generating mechanisms, ϕ_g the probability of mechanism g being used, and $\psi_{g,j,v}$ the probability that mechanism g will produce value v for attribute j.

At first, the number of mixture components, M, will be a finite constant, but later it will be set to infinity. In a latent class analysis application, we will be interested in the components of the mixture distribution as possible indicators of real underlying mechanisms. In a classification application we might regard the use of a mixture distribution as just a device for expressing correlations among the attributes.

A prior for the mixture model parameters. In a Bayesian treatment of this problem, the parameter vectors $\widetilde{\phi}$ and $\widetilde{\psi}_{g,j}$ are considered to be the values of random variables $\widetilde{\Phi}$ and $\widetilde{\Psi}_{g,j}$ whose distributions reflect our prior knowledge of the process. It is mathematically convenient to express this prior knowledge via independent Dirichlet probability densities for the parameter vectors, since these are the conjugate priors for this problem. Later, I will briefly discuss the situation where the actual prior knowledge is too unspecific to be captured by densities of this form.

This combined Dirichlet prior density can be written as follows:

$$\begin{aligned} & p(\widetilde{\Phi} = \widetilde{\phi},\ \widetilde{\Psi}_{g,j} = \widetilde{\psi}_{g,j}\ :\ 1 \leq g \leq M,\ 1 \leq j \leq m) \\ &\quad = p(\widetilde{\Phi} = \widetilde{\phi}) \cdot \prod_{g,j} p(\widetilde{\Psi}_{g,j} = \widetilde{\psi}_{g,j}) \\ &\quad = \left(\frac{\Gamma(\alpha)}{\Gamma(\alpha/M)^M} \prod_{g} \phi_g^{(\alpha/M)-1} \right) \cdot \prod_{g,j} \left(\frac{\Gamma(\beta_j)}{\Gamma(\beta_j/N_j)^{N_j}} \prod_{v} \psi_{g,j,v}^{(\beta_j/N_j)-1} \right) \end{aligned} \tag{3}$$

where α and the β_j must be greater than zero. Setting $\alpha = M$ and $\beta_j = N_j$ in equation (3) produces a uniform distribution over the parameter space. A smaller value for α produces a prior distribution that favours values for the ϕ_g that are near 0 or 1 — i.e. a situation where a few components are much more probable than the others. A larger value for α favours values of ϕ_g near $1/M$ — i.e. a situation where the components are all about equally probable. Varying β_j has analogous effects on the priors for the distribution of attribute j in the various mixture components.

Integrating out the parameters. The joint distribution of the G_i and the $\widetilde{V}_i$ can be found by integrating their probabilities over the region of valid parameter values, as weighted by prior probability density. This joint distribution will later provide the basis for obtaining the conditional probabilities needed for classification (equation (1)). The $\widetilde{V}_i$ and G_i for different values of i are of course *not* independent — they would be so only if the parameters ϕ_g and $\psi_{g,j,v}$ were known.

The required integration is as follows:

$$\begin{aligned}
&P(G_i = g_i,\ \widetilde{V}_i = \widetilde{v}_i\ :\ 1 \leq i \leq n) \\
&\quad= \int P(G_i = g_i,\ \widetilde{V}_i = \widetilde{v}_i\ :\ 1 \leq i \leq n \mid \widetilde{\Phi} = \widetilde{\phi},\ \widetilde{\Psi}_{g,j} = \widetilde{\psi}_{g,j}\ :\ 1 \leq g \leq M,\ 1 \leq j \leq m) \\
&\qquad\qquad \cdot\, p(\widetilde{\Phi} = \widetilde{\phi},\ \widetilde{\Psi}_{g,j} = \widetilde{\psi}_{g,j}\ :\ 1 \leq g \leq M,\ 1 \leq j \leq m)\ d\langle \widetilde{\phi},\ \widetilde{\psi} \rangle \\
&\quad= C \int \prod_i \phi_{g_i} \cdot \prod_{i,j} \psi_{g_i,j,v_{ij}} \cdot \prod_g \phi_g^{(\alpha/M)-1} \cdot \prod_{g,j} \prod_v \psi_{g,j,v}^{(\beta_j/N_j)-1}\ d\langle \widetilde{\phi},\ \widetilde{\psi} \rangle \\
&\quad= C \int \prod_g \phi_g^{\Sigma_i \delta(g_i,g) + (\alpha/M) - 1} \cdot \prod_{g,j} \prod_v \psi_{g,j,v}^{\Sigma_i \delta(g_i,g)\delta(v_{ij},v) + (\beta_j/N_j) - 1}\ d\langle \widetilde{\phi},\ \widetilde{\psi} \rangle \\
&\quad= C \int_{S_M} \prod_g \phi_g^{\Sigma_i \delta(g_i,g) + (\alpha/M) - 1} d\widetilde{\phi} \cdot \prod_{g,j} \int_{S_{N_j}} \prod_v \psi_{g,j,v}^{\Sigma_i \delta(g_i,g)\delta(v_{ij},v) + (\beta_j/N_j) - 1}\ d\widetilde{\psi}_{g,j}
\end{aligned} \tag{4}$$

where $C = (\Gamma(\alpha)/\Gamma(\alpha/M)^M) \cdot \prod_{g,j} (\Gamma(\beta_j)/\Gamma(\beta_j/N_j)^{N_j})$, and $\delta(x, y)$ is one if $x = y$ and zero otherwise. The integrals in the last formula are taken over the simplexes of valid probability distributions — $S_D = \{\, \langle x_1, \ldots, x_D \rangle\ :\ x_i \geq 0,\ \sum_i x_i = 1 \,\}$.

These integrals are standard for Dirichlet distributions (see [14, section 2.4], for example). Their evaluation gives the following:

$$\begin{aligned}
&P(G_i = g_i,\ \widetilde{V}_i = \widetilde{v}_i\ :\ 1 \leq i \leq n) \\
&\quad= \frac{\Gamma(\alpha)}{\Gamma(n+\alpha)} \prod_g \frac{\Gamma(\Sigma_i \delta(g_i, g) + \alpha/M)}{\Gamma(\alpha/M)} \\
&\qquad\quad \cdot \prod_{g,j} \frac{\Gamma(\beta_j)}{\Gamma(\Sigma_i \delta(g_i, g) + \beta_j)} \prod_v \frac{\Gamma(\Sigma_i \delta(g_i, g)\delta(v_{ij}, v) + \beta_j/N_j)}{\Gamma(\beta_j/N_j)} \\
&\quad= \prod_i \frac{\sum_{k<i} \delta(g_k, g_i) + \alpha/M}{(i-1) + \alpha} \cdot \prod_{i,j} \frac{\sum_{k<i} \delta(g_k, g_i)\delta(v_{kj}, v_{ij}) + \beta_j/N_j}{\sum_{k<i} \delta(g_k, g_i) + \beta_j}
\end{aligned} \tag{5}$$

This formula can be interpreted as a prescription for generating a sequence of values for the G_i and $\widetilde{V}_i$ according to a distribution picked in accord with the prior (equation (3)) — i.e. for generating a sequence from the prior predictive distribution for sequences of G_i and $\widetilde{V}_i$. This generation procedure uses the following incremental conditional probabilities, identified from equation (5):

$$P(G_i = g_i \mid G_k = g_k,\ \widetilde{V}_k = \widetilde{v}_k\ :\ 1 \leq k < i) \;=\; \frac{\sum_{k<i} \delta(g_k, g_i) + \alpha/M}{(i-1) + \alpha} \tag{6}$$

$$P(V_{ij} = V_{ij} \mid G_i = g_i,\ G_k = g_k,\ \widetilde{V}_k = \widetilde{v}_k \ : \ 1 \leq k < i) \ = \ \frac{\sum_{k<i} \delta(g_k, g_i)\delta(v_{kj}, v_{ij}) + \beta_j / N_j}{\sum_{k<i} \delta(g_k, g_i) + \beta_j} \quad (7)$$

To generate a sequence of G_i and $\widetilde{V}_i$, we choose values for the G_i in turn, based on the frequencies with which components were previously chosen, biased by the term α/M. Once a G_i is selected, we chose values for the corresponding V_{ij} based on the frequencies of the various attribute values in vectors previously produced using the same component, biased by the β_j/N_j terms.

Note that by summing over all possible sequences of G_i, we can, via this procedure, define the probability of any sequence of the observable variables, $\widetilde{V}_i$, without any explicit reference to the parameters ϕ_g and $\psi_{g,j,v}$.

Models with an infinite number of components. The above procedure for producing a sequence from the predictive distribution for $\widetilde{V}_i$, remains well-defined even when the number of mixture components, M, is infinite. Values for the underlying G_i are also produced, but are specified only up to identity or non-identity among themselves.

Consider first the choice of a value for G_1. There are an infinite number of candidates for this value, but which of these is chosen has no effect on the values that are then chosen for V_{1j} — equation (7) shows that these are in all cases picked from uniform distributions. Next, consider the choice of a value for G_2. From equation (6), we see that this value should be chosen to be equal to G_1 with probability $(1 + \alpha/M)/(1 + \alpha) = 1/(1 + \alpha)$ (since M is infinite). If this choice is made, values for the V_{2j} will be chosen with probabilities that depend on the corresponding V_{1j}, in accordance with equation (7). Alternatively, a value for G_2 different from G_1 will be picked with probability $\alpha/(1 + \alpha)$. There are an infinite number of such alternatives, but again it makes no difference which is chosen. Proceeding in this fashion, we can pick "values" for the G_i that are defined only in so far as they are specified to be equal or not equal to previously chosen values. This is sufficient to pick values for the corresponding V_{ij}, which are the only observable attributes.

As discussed in the introduction, models with an infinite number of components are natural in many contexts. Nevertheless, one might feel uneasy about estimating an infinite number of model parameters on the basis of a finite sample. In the Bayesian approach, however, we do not attempt to settle on a single set of parameter values. Furthermore, the form of prior distribution chosen here, in which the bias term, α/M, goes to zero as M goes to infinity, ensures that parameter values from the posterior distribution will give some of the infinite number of components significant probability. As a result, overfitted solutions in which each data item is attributed to a separate mixture component do not occur.

Prediction by Monte Carlo simulation

Although $\widetilde{\phi}$ and the $\widetilde{\psi}_{g,j}$ have been integrated away, calculating $P(\widetilde{V}_i = \widetilde{v}_i \ : \ 1 \leq i \leq n)$ would still require summing over all M^n possible combinations of values for the G_i. For a classification application, however, all that is needed is the distribution for the category attribute of a test item, conditional on the known attributes of the test item and on the training data. A sample from this conditional distribution can be obtained by a Monte Carlo simulation procedure.

Exhibiting a distribution via Monte Carlo simulation. Consider the problem of producing a sample from the joint distribution of the random variables $\langle A_1, \ldots, A_n \rangle$ when only the full conditional probabilities, $P(A_i = a_i \mid A_j = a_j \,:\, j \neq i)$ are readily computable. This problem arises in the context of statistical physics [16], image restoration [10], stochastic neural networks [1, 17], belief networks for expert systems [18], and general statistical calculation [12, 9]. In each case, the problem has been solved by a Monte Carlo simulation method, under names such as the "Metropolis algorithm", the "Boltzmann Machine", and the "Gibbs sampler".

The simulation starts with arbitrary values $\langle a_1^0, \ldots, a_n^0 \rangle$. In iteration t, new values $\langle a_1^t, \ldots, a_n^t \rangle$ are stochastically chosen in turn, with a_h^t being picked at position h with probability:

$$P(A_h = a_h^t \mid A_i = a_i^t,\ A_j = a_j^{t-1} \,:\, 1 \leq i < h,\ h < j \leq n)$$

In other words, new values are picked at each position from the conditional distribution for that position given the most-recently picked values at all other positions.

It is easy to see that if the distribution of $\langle a_1^T, \ldots, a_n^T \rangle$ for some T is that of $\langle A_1, \ldots, A_n \rangle$, then the same will be true at all later iterations. Furthermore, the method is in fact guaranteed to converge to this equilibrium distribution in the limit as the number of iterations grows, as long as the conditional probabilities used are bounded away from zero. Accordingly, for some suitably large T, we can treat the values $\langle a_1^T, \ldots, a_n^T \rangle$ as coming from the desired distribution for $\langle A_1, \ldots, A_n \rangle$. Several independent vectors can be obtained by running several simulations, or by taking vectors from a single simulation at sufficiently widely separated times.

The number of iterations required for this method to give a reasonable approximation can be difficult to determine, however, so empirical tests of the practicality of the method in any particular application are necessary. The situation is analogous to the problem of local maxima with deterministic optimization procedures, except that the stochastic aspect ensures that the simulation will escape the local maximum eventually. The time required for this can be reduced by the method of "simulated annealing", as is done in [1], but such elaborations are not considered in this paper.

Simulating the Bayesian predictive distribution. This Monte Carlo simulation technique can be used to solve the Bayesian prediction problem for mixture models. Given that we know certain of the V_{ij}, we would like to obtain a sample from the joint distribution for the G_i and the V_{ij} whose values we do not know. This can be done by a simulation in which the values of the V_{ij} that are known are left fixed, while new values for each G_i and each unknown V_{ij} are chosen repeatedly from their distributions conditional on the current values of all the other variables.

The conditional distributions for the G_i and the unknown V_{ij} are readily obtained from the joint distribution of the G_i and V_{ij} (equation (5)):

$$\begin{aligned} & P(V_{ij} = v \mid G_k = g_k,\ \widetilde{V}_k = \widetilde{v}_k,\ G_i = g_i,\ V_{il} = v_{il} \,:\, 1 \leq k \leq n,\ k \neq i,\ 1 \leq l \leq m,\ l \neq j) \\ & \quad = \frac{\sum_{k \neq i} \delta(g_k, g_i)\delta(v_{kj}, v) + \beta_j / N_j}{\sum_{k \neq i} \delta(g_k, g_i) + \beta_j} \end{aligned} \tag{8}$$

$$\begin{aligned} & P(G_i = g \mid G_k = g_k,\ \widetilde{V}_k = \widetilde{v}_k,\ \widetilde{V}_i = \widetilde{v}_i \,:\, 1 \leq k \leq n,\ k \neq i) \\ & \quad = \frac{1}{Z} \cdot \left(\sum_{k \neq i} \delta(g_k, g) + \alpha / M \right) \cdot \prod_j \frac{\sum_{k \neq i} \delta(g_k, g)\delta(v_{kj}, v_{ij}) + \beta_j / N_j}{\sum_{k \neq i} \delta(g_k, g) + \beta_j} \end{aligned} \tag{9}$$

Here, Z is a factor independent of g that normalizes the distribution to sum to one.

If the number of components is infinite, equation (9) is adapted to allow a new "value" for G_i to be selected that is defined in so far as it is equal to the value for some other G_k, or different from all others, according to the following formulas:

$$\begin{aligned} & P(G_i = g \mid G_k = g_k,\ \widetilde{V}_k = \widetilde{v}_k,\ \widetilde{V}_i = \widetilde{v}_i\ :\ 1 \le k \le n,\ k \ne i) \\ & \quad = \ \frac{1}{Z} \cdot \left(\textstyle\sum_{k \ne i} \delta(g_k, g) \right) \cdot \prod_j \frac{\sum_{k \ne i} \delta(g_k, g)\delta(v_{kj}, v_{ij}) + \beta_j / N_j}{\sum_{k \ne i} \delta(g_k, g) + \beta_j} \end{aligned} \tag{10}$$

$$\begin{aligned} & P(G_i \ne g_k\ :\ 1 \le k \le n,\ k \ne i \mid G_k = g_k,\ \widetilde{V}_k = \widetilde{v}_k,\ \widetilde{V}_i = \widetilde{v}_i\ :\ 1 \le k \le n,\ k \ne i) \\ & \quad = \ \frac{1}{Z} \cdot \alpha \cdot \prod_j \frac{1}{N_j} \end{aligned} \tag{11}$$

where again Z is a normalizing factor, the same for both the above formulas. The second formula gives the probability for setting G_i to one of the infinite number of components that are not currently assigned to any of the other G_k. These components do not need to be distinguished, since they all have equivalent effects on the rest of the simulation.

Implementation of the method. Figure 1 shows an implementation of the simulation procedure for finite M. This procedure takes as inputs the values of attributes for training items — v_{ij}, for $1 \le i < n$ and $1 \le j \le m$ — along with the values of the indicator attributes for a test item — v_{nj}, for $2 \le j \le m$. It outputs an estimate of the distribution for the category attribute of the test item, V_{n1}:

$$q_c \ \approx \ P(V_{n1} = c \mid \widetilde{V}_i = \widetilde{v}_i,\ V_{nj} = v_{nj},\ :\ 1 \le i < n,\ 2 \le j \le m) \tag{12}$$

The number of mixture components, M, the number of simulation iterations, T, and the parameters of the prior distributions, α and the β_j, are additional inputs to the procedure.

This implementation incrementally maintains the required occurrence counts for mixture components and for item attributes associated with mixture components. These are kept in the arrays C and A, with

$$C[g] \ = \ \sum_{i=1}^{n} \delta(g_i, g), \quad A[g, j, v] \ = \ \sum_{i=1}^{n} \delta(g_i, g)\delta(v_{ij}, v) \tag{13}$$

This method results in a time complexity for the procedure of $O((nmM + N_1)T + MN)$, with $N = \sum_j N_j$. On the reasonable assumption that $N_1 < nmM$ and $N < nm$, this reduces to $O(nmMT)$.

If no limit is placed on the number of mixture components, the simulation can be done in time $O(nm\overline{M}T)$, where $\overline{M}$ is the average number of distinct mixture components actually in use (this is never more than n). Achieving this time bound requires maintenance of an additional array listing mixture components currently in use.

In practice, several modifications to the procedure shown are desirable. Rather than initialize all the g_i to 1, it may be better to set them to sequential or random values. It will also generally be better to let the simulation run for some number of iterations, allowing it to reach equilibrium, before starting to observe which values show up for the category attribute of the

Initialize the mixture component associated with each training item, and the unknown v_{n1}.

for $i \leftarrow 1..n$ **do** $g_i \leftarrow 1$ **od**
$v_{n1} \leftarrow 1$

Initialize the occurrence counts for components and for component/attribute combinations.

for $g \leftarrow 1..M,\ j \leftarrow 1..m,\ v \leftarrow 1..N_j$ **do** $C[g] \leftarrow 0,\ A[g,j,v] \leftarrow 0$ **od**

for $i \leftarrow 1..n$ **do**
    $C[g_i] \leftarrow C[g_i] + 1$
    for $j \leftarrow 1..m$ **do** $A[g_i, j, v_{ij}] \leftarrow A[g_i, j, v_{ij}] + 1$ **od**
od

Initialize the frequency counts for the unknown attribute, v_{n1}.

for $c \leftarrow 1..N_1$ **do** $F[c] \leftarrow 0$ **od**

Conduct the simulation for T iterations.

for $t \leftarrow 1..T$ **do**

    Select a new mixture component to go with each item, while updating counts.

    for $i \leftarrow 1..n$ **do**

        Remove the old value for g_i from the occurrence counts.

        $C[g_i] \leftarrow C[g_i] - 1$
        for $j \leftarrow 1..m$ **do** $A[g_i, j, v_{ij}] \leftarrow A[g_i, j, v_{ij}] - 1$ **od**

        Calculate the distribution for the new value of g_i and pick a value from it.

        $Z \leftarrow 0$
        for $g \leftarrow 1..M$ **do**
            $X[g] \leftarrow C[g] + \alpha/M$
            for $j \leftarrow 1..m$ **do** $X[g] \leftarrow X[g] \cdot (A[g, j, v_{ij}] + \beta_j/N_j) \,/\, (C[g] + \beta_j)$ **od**
            $Z \leftarrow Z + X[g]$
        od

        for $g \leftarrow 1..M$ **do** $X[g] \leftarrow X[g] \,/\, Z$ **od**
        $g_i \leftarrow$ Value in the range $1..M$ picked according to the distribution X

        Update the occurrence counts to reflect the new value for g_i.

        $C[g_i] \leftarrow C[g_i] + 1$
        for $j \leftarrow 1..m$ **do** $A[g_i, j, v_{ij}] \leftarrow A[g_i, j, v_{ij}] + 1$ **od**

    od

    Select a new value for v_{n1}, updating the occurrence and frequency counts accordingly.

    $A[g_n, 1, v_{n1}] \leftarrow A[g_n, 1, v_{n1}] - 1$

    for $v \leftarrow 1..N_1$ **do** $Y[v] \leftarrow (A[g_n, 1, v] + \beta_1/N_1) \,/\, (C[g_n] + \beta_1)$ **od**
    $v_{n1} \leftarrow$ Value in the range $1..N_1$ picked according to the distribution Y

    $A[g_n, 1, v_{n1}] \leftarrow A[g_n, 1, v_{n1}] + 1$
    $F[v_{n1}] \leftarrow F[v_{n1}] + 1$

od

Calculate the observed distribution for the category attribute of the test item, v_{n1}.

for $c \leftarrow 1..N_1$ **do** $q_c \leftarrow F[c] \,/\, T$ **od**

Figure 1: The simulation procedure for finite M.

test item.

The procedure shown assumes that all the attributes of the training items and all the indicator attributes of the test item are known. If this is not so, values for the unknown V_{ij} could be selected during the simulation, just as is done for V_{n1}. However, if these values are not of interest, the factors that correspond to them in the calculation of the distribution $X[g]$ can instead be simply omitted. (The validity of this procedure can be seen by formulating the joint distribution of the G_i and the V_{ij} of interest, and from that deriving the appropriate conditional distributions.)

The procedure could easily be modified to observe distributions for the category attributes of several test items as the simulation is run. However, the results would not necessarily be the same as would be obtained by running separate simulations for each test item, since correlations among the indicator attributes of the test items could affect the rest of the simulation.

This effect is probably beneficial. It would, however, make comparison with other methods difficult, since the usual criterion for evaluating a classification method is expected performance on a single test item, though naturally many test items are classified in order to obtain significant statistics.

Accordingly, the empirical tests reported below used an implementation in which only the training items, $\widetilde{V}_1, \ldots, \widetilde{V}_n$, participate in the simulation. The following quantities were calculated with respect to a test item, $\widetilde{V}_*$, as the simulation was run:

$$\begin{aligned} p_c &= \frac{1}{T} \sum_{t=1}^{T} \sum_{g=1}^{M} \frac{C[g] + \alpha/M}{n + \alpha} \cdot \frac{A[g,1,c] + \beta_1/N_1}{C[g] + \beta_1} \cdot \prod_{j=2}^{m} \frac{A[g,j,v_{*j}] + \beta_j/N_j}{C[g] + \beta_j} \\ &\approx P(V_{*1} = c,\ V_{*j} = v_{*j}\ :\ 2 \leq j \leq m \mid \widetilde{V}_i = \widetilde{v}_i\ :\ 1 \leq i \leq n) \end{aligned} \tag{14}$$

One can then obtain the desired category probabilities:

$$P(V_{*1} = c \mid \widetilde{V}_i = \widetilde{v}_i,\ V_{*j} = v_{*j}\ :\ 1 \leq i \leq n,\ 2 \leq j \leq m) \ \approx\ p_c \,/\, \textstyle\sum_c p_c \tag{15}$$

In this way, many test items can be classified simultaneously without any interaction between them. The computation also yields the probability of the complete test item — this is just $p_{v_{*1}}$, where v_{*1} is the true category of the test item.

By saving the contents of the A and C arrays at selected points during a simulation run, equations (14) and (15) can also be used to classify many test items presented sequentially, without re-running the simulation for each item. Compared to the direct method of Figure 1, this approach also reduces sampling error in the estimates of category probabilities.

Other mixture modeling methods

In this section, I will review several other methods for modeling data as a mixture, and discuss how they relate to the Bayesian method presented above.

Maximum likelihood estimation. Maximum likelihood is a standard non-Bayesian approach to fitting a model, in which the model parameters that maximize the probability of the observed data (the "likelihood" of the parameters) are used as the basis for interpretation and prediction. Computation of the maximum likelihood parameters for a mixture model can conveniently be carried out by the EM algorithm [5, 8, 20]. In this iterative procedure, expectations for which unobserved mixture component underlies each data item are first computed using the current estimate of parameters (the E step), after which a new parameter estimate is computed using these expectations for the unobserved data (the M step). With repeated application of these steps, the method is guaranteed to converge to a set of parameters that at least locally maximizes the likelihood. By running the algorithm from several starting points, one may be able to find parameter values with likelihood close to that of the global maximum.

This method can be applied directly only if the number of mixture components, M, is considered fixed. Trying to estimate M itself by maximum likelihood leads to overfitting, with a separate mixture component assigned to each data item. In practice, setting M to a value at the high end of prior expectations sometimes gives an acceptable result, with excess components having low probability, or innocuously duplicating other components of the mixture. With some difficulty, classical hypothesis testing criteria can be used to decide between models with different numbers of components [8, 20]. The method of cross-validation [19] should also be applicable.

Maximum a posteriori probability (MAP) estimation. Rather than use the parameters which maximize the likelihood, one can instead use the parameters with maximum posterior probability density — the product of likelihood and prior probability density. From a Bayesian viewpoint, this is theoretically justified only when the mode of the posterior distribution can be shown to approximate the optimal parameter estimate under some relevant loss function, though the method is commonly applied without such a demonstration. MAP estimation is algorithmically identical to the non-Bayesian "penalized likelihood" method.

This method has been applied to latent class analysis using mixture models in the early versions of the AutoClass system of [4]. In this system, a conjugate prior is used for the model parameters (as in equation (3) here), allowing the MAP estimate to be found using the EM algorithm. If the prior distribution used has β_j significantly greater than N_j, the problem of overfitting encountered in maximum likelihood estimation is much reduced — there is a pressure for a single mixture component to underly many data items, since that allows its posterior attribute probabilities to overcome the prior bias and accurately fit the observed frequencies.

However, significant deviations from the true Bayesian result would still be expected if the user has a prior with $\beta_j \leq N_j$ — for example, using $\alpha = M$ and $\beta_j = N_j$ gives estimates identical to those of maximum likelihood. Furthermore, if several local maxima are found, selecting between them on the basis of their posterior probability density will not, in general, give the result which is most likely to be close to the best value.

Local approximations to the Bayesian solution. In later versions of the AutoClass system [11], the EM algorithm is run from several starting points, with the number of mixture components set to various values. The local maxima found in this way are then evaluated by computing an approximation to the total posterior probability of the region of parameter space in their vicinity. (In this calculation, the probability is adjusted for the number of permutations of mixture component identities that give equivalent results, and a prior for the number of components is introduced.) The local maximum with highest total probability is then considered the best (or they are all used, with weights given by their probabilities).

The quality of the solutions found with this procedure depends on whether the posterior parameter probability is in fact concentrated in a set of discrete peaks, whether all the significant peaks are found, and whether the approximations used to compute the total probability around each peak are good. In comparison, the quality of the predictions made by the Bayesian method presented in this paper depends principally on whether the simulation approaches equilibrium in the time allotted. Which method works best in practice will have to be discovered by experience.

Both the method of this paper and the later versions of AutoClass can handle the case where the number of mixture components is unknown, but they do so in rather different ways. The relative merits of these two approaches will again have to be evaluated empirically.

Stochastic forms of the EM algorithm. In the E step of the EM algorithm, rather than calculate exact expectations for the unobserved data, one can instead generate a sample from the distribution for these variables conditional on the observed data and the current estimate of the model parameters [21]. In some situations, this procedure will be computationally more efficient than the standard E step (though this is not so for the mixture models considered in this paper).

In the limit as the size of the sample taken grows, this stochastic form of the EM algorithm will give the same results as the exact form. At the other extreme, one might generate only a single value for each unobserved variable. This is the procedure used in [3] in the context of maximum likelihood estimation for a mixture of Gaussians. Various benefits are claimed for this method, but it would appear that the algorithm is guaranteed to settle to a state where only a single mixture component is used if it is allowed to run long enough, since once a component is eliminated due to random fluctuations it is never re-introduced.

This defect would not be present in the analogous algorithm for finding an approximation to the MAP estimate, provided the prior has $\alpha > M$ and $\beta_j > N_j$. Such an algorithm would in fact be quite close to the Monte Carlo simulation method for full Bayesian inference presented here, with the differences lying in the timing of model parameter updates, and in the interpretation of the bias terms as priors.

Performance on a simple classification task

An empirical test of the Bayesian mixture modeling technique was undertaken in order to determine whether the Monte Carlo simulations involved reach approximate equilibrium in a reasonable time, and to examine the effect of using a model with an infinite number of components. Some preliminary comparisons with maximum likelihood and MAP estimation using the EM algorithm and with nearest-neighbor classification were also done.

The classification task. The Bayesian and other methods were evaluated on the task of modeling data synthetically generated from the mixture distribution shown in Table 1. This mixture is composed of four equally-probable components, each of which produces a distribution over nine two-valued attributes in which each attribute is independent of the others, given knowledge of the mixture component.

The first of the attributes is considered to be the category of the item. The classification task is to guess this attribute on the basis of the values for the other eight attributes. With knowledge of the real distribution, the optimal error rate on this task is 18.6%. Note that none of the indicator attributes provide any information about the category attribute when taken alone, so good performance on this task requires that the procedure uncover the two latent classes underlying each category.

g	$P(G_i = g)$	$P(V_{ij} = 1 \mid G_i = g),\ j = 1..9$
1	0.25	1.0 0.8 0.8 0.8 0.8 0.2 0.2 0.2 0.2
2	0.25	1.0 0.2 0.2 0.2 0.2 0.8 0.8 0.8 0.8
3	0.25	0.0 0.8 0.8 0.2 0.2 0.8 0.8 0.2 0.2
4	0.25	0.0 0.2 0.2 0.8 0.8 0.2 0.2 0.8 0.8

Table 1: The true mixture distribution.

The methods were also evaluated on their ability to predict the entirety of a test item, $\widetilde{V}_*$, whose true value is $\widetilde{v}_*$, with a loss of $-\log_2(\widehat{P}(\widetilde{V}_* = \widetilde{v}_*))$, where $\widehat{P}$ is the estimated probability. This would be the appropriate loss function for a data compression application. Performance on this task with knowledge of the real distribution is its entropy, which is 7.67 bits.

Each method was applied to three training sets consisting of 12 data items each (S1, S2, and S3), and to three training sets consisting of 48 data items each (L1, L2, L3). These sets were randomly generated from the true distribution. A training set of size 12 is just large enough for meaningful inference to be possible, but is also just small enough that for small M the exact Bayesian solution can be computed in not completely unreasonable amounts of time. This allowed the accuracy of the Monte Carlo simulations to be judged.

For each method and each training set, all $2^9 = 512$ possible test items were used to evaluate performance on classification and whole-item prediction, with each item weighted by its probability under the true distribution. There is thus no sampling error from this source in the evaluations.

An illustrative run. A typical simulation run on training set S1 with the number of mixture components, M, fixed at 4 is shown in Figure 2. The columns of the figure show which of the mixture components, represented by the symbols $\clubsuit$, $\diamondsuit$, $\heartsuit$, and $\spadesuit$, is associated with each training item at successive iterations of the simulation, starting with an initial state in which all items are assigned the same component ($\clubsuit$).

By about the sixth iteration, the simulation reaches an apparent equilibrium situation in which 11 of the 12 items belong to one of four stable or nearly stable groups. Items 8, 11, and 12 form a group associated with component $\diamondsuit$. Items 1, 7, and 10, items 2, 4, and 5, and items 3 and 9 also form groups, though the mixture components associate with these groups change with time. Item 6 spends part of its time with the 8, 11, 12 group and part of its time with the 2, 4, 5 group.

These groupings, including the ambivalent behaviour of item 6, are in accord with what one would expect from manual examination of the data and/or knowledge of the true distribution.

Convergence of the simulations. A more objective picture of convergence to equilibrium can be obtained by observing the predictive performance of the simulations as the number of iterations increases. Such tests were performed for $M = 4$, with $\alpha = 1$ and all the $\beta_j = 1$. Simulations using larger values for α or the β_j would presumably reach equilibrium more quickly, since the conditional probabilities that arise are bounded further away from zero. A larger value for M would also be expected to improve convergence to equilibrium, by eliminating some local maxima.

#	*Data item*	*Assignments of components to items at successive iterations*
1	⟨1 12 11 21 22⟩	♣♣♣♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♣♣♣♣♣♣♣♣♣
2	⟨1 22 22 21 11⟩	♣♣♣♣♣♣♣♣♣♣♣♣♢♢♢♢♠♠♠♠♠♠♠♠♠♠♠♠♠♢♢♣♣♢♠♠♠♣♢♢♢♢♡♡
3	⟨2 11 22 12 22⟩	♣♣♡♠♠♠♠♠♠♠♠♠♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♠♣♣♠♠♡♣♣♡♡♡♠♠♠♠♠
4	⟨1 22 22 12 11⟩	♣♣♣♣♣♣♣♣♣♣♣♣♣♢♣♣♠♠♠♠♠♠♠♠♠♠♣♠♠♢♣♣♠♠♠♢♢♢♢♢♢♢♡♡
5	⟨1 12 22 11 11⟩	♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♠♠♠♠♠♠♠♠♣♠♣♠♠♠♣♣♣♠♠♠♠♠♣♢♢♢♡♡♡
6	⟨2 22 21 21 11⟩	♣♣♣♣♢♣♢♢♢♣♢♢♢♢♢♢♢♠♠♠♢♢♠♠♢♠♢♢♢♢♢♣♢♢♠♢♣♢♢♢♢♢♢♢
7	⟨1 11 11 21 12⟩	♣♣♣♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♡♣♣♣♣♣♣♣♣♣
8	⟨2 22 11 22 11⟩	♣♣♣♣♣♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢
9	⟨2 12 22 12 21⟩	♣♣♠♠♠♠♠♠♠♠♠♣♣♣♣♣♣♣♣♣♠♣♣♣♣♣♣♠♠♣♣♠♠♠♣♢♡♢♠♠♠♠♠♠
10	⟨1 11 21 22 22⟩	♣♣♡♣♡♠♡♡♡♡♡♡♡♡♡♡♣♡♡♣♡♡♡♡♡♡♡♡♡♡♠♡♡♡♣♣♢♡♣♣♣♣♣♣
11	⟨2 22 12 22 11⟩	♣♣♣♣♣♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢♢
12	⟨2 21 11 22 11⟩	♣♣♣♣♢♢

Figure 2: A run of the simulation procedure for training set S1, with $M = 4$, $\alpha = 1$, and $\beta_j = 1$. The four mixture components are represented by the symbols ♣, ♢, ♡, and ♠.

Figure 3 plots the change in performance for classification and whole-item prediction for three simulation runs on each of the training sets of size 12. For comparison, the performance using the exact Bayesian predictive distribution for the 512 possible test items is also shown (the dotted line). This was calculated using equation (5) by a summation over all $4^{13} = 67\,108\,864$ possible assignments of mixture components to the training data and a test item (a computation taking sixteen hours for each training set). Performance using the training sets of size 48 is shown in Figure 4. In this case, the exact Bayesian computation is out of the question.

In these two Figures, classification performance (in percent errors) is shown on the left, and performance at whole item prediction with loss of $-\log_2(\widehat{P}(\widetilde{V}_* = \widetilde{v}_*))$ (in bits) is shown on the right. Both vertical scales extend from the optimum level given knowledge of the true distribution up to the level achieved with a uniform distribution. Performance is plotted for every iteration up to 10, then every tenth iteration up to 100, and finally every hundredth iteration up to 1000. Each run began with all items assigned the same mixture component. After each iteration, classification and whole-item prediction for each possible test item were done using equations (14) and (15), giving the results shown.

Note that these performance curves are the combined result of two effects — the approach of the simulation state to an equilibrium distribution, and the accumulation of a sample of sufficient size to give good estimates of the probabilities.

For the small training sets, convergence to the exact solution is evident within about 200 iterations, except for one run whose classification performance still differed somewhat from that of the exact solution after 1000 iterations. This one anomaly may be due to the fact that classification performance has a discrete set of possible values. Very small differences in estimated probabilities can therefore have a significant effect on performance.

If anything, the simulations for the larger training sets appear to converge more quickly than those for the smaller training sets. One might expect that equilibrium would be *harder* to reach

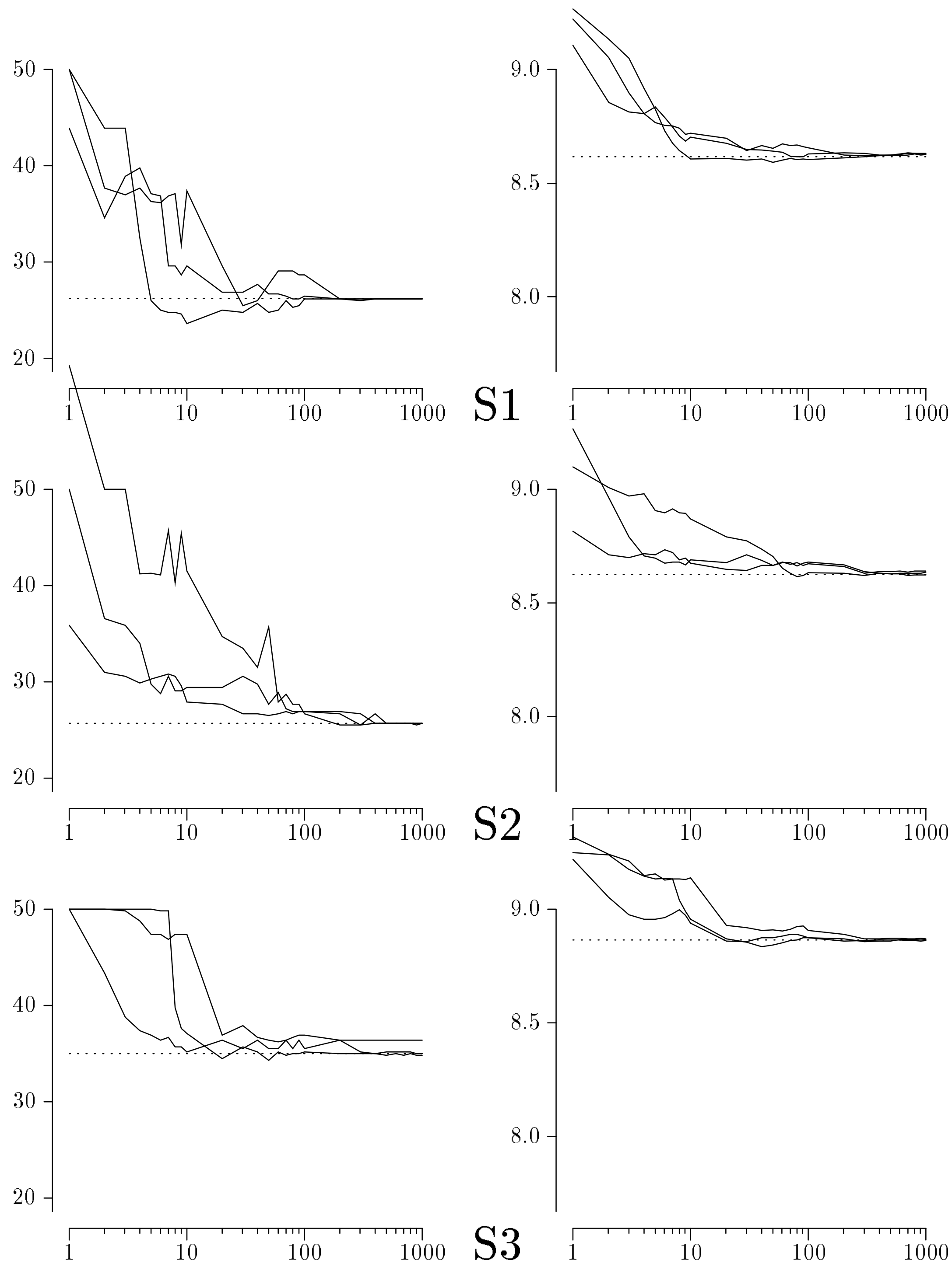

Figure 3: Convergence of simulation runs on the training sets of size 12. Three runs taken to 1000 iterations are shown for each set. Plots on the left show classification performance (percent errors); those on the right show performance on whole-item prediction (in bits). The dotted lines show the performance of the exact Bayesian solutions.

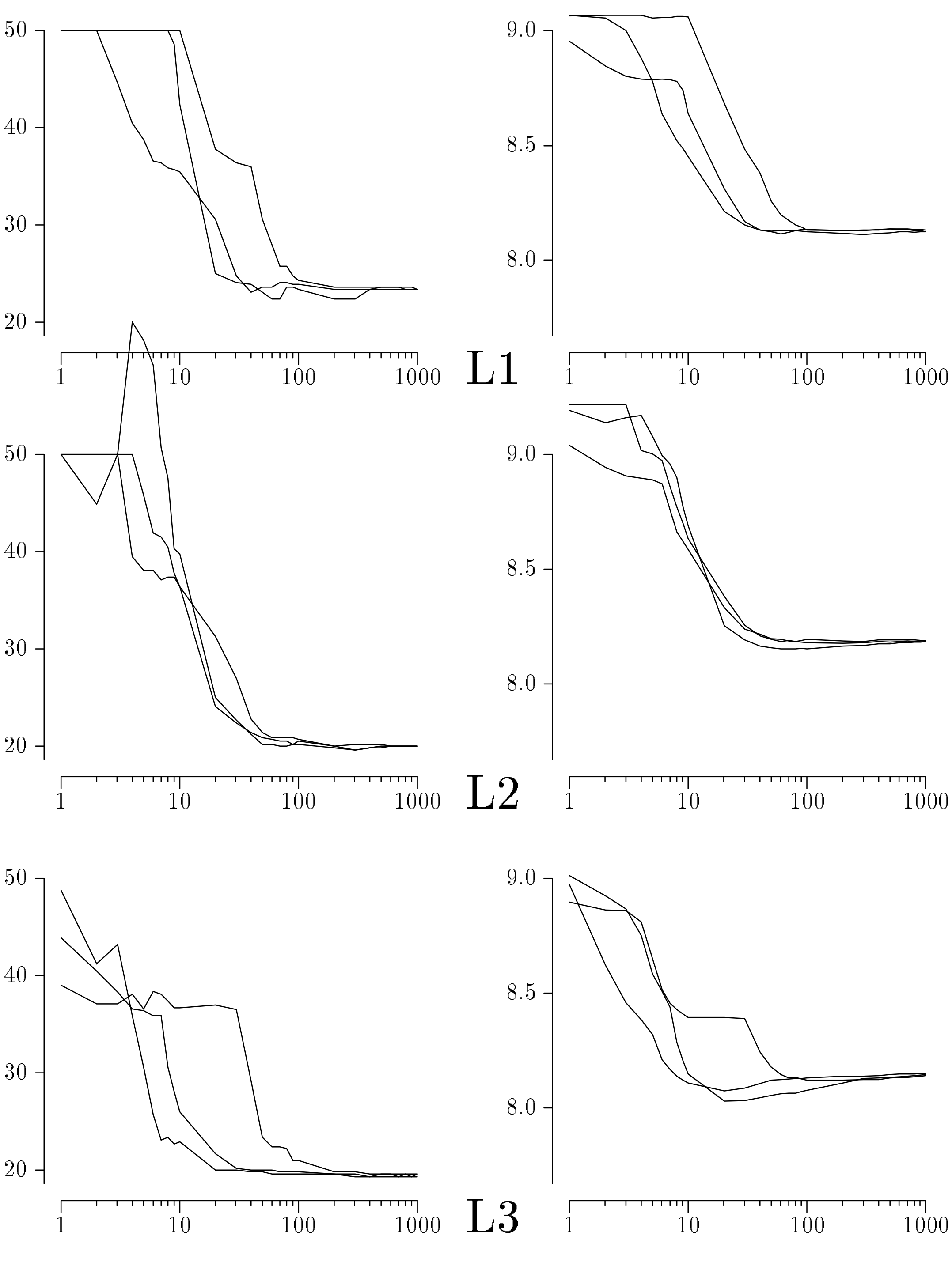

Figure 4: Convergence of simulation runs for the training sets of size 48. Three runs taken to 1000 iterations are shown for each set. Plots on the left show classification performance (percent errors); those on the right show performance on whole-item prediction (in bits).

when more data items are involved, but if so, it appears that this is out-weighed by the more rapid accumulation of a significant sample.

Performance of the Bayesian method. The Bayesian mixture modeling method was evaluated on the three large and three small training sets, with the number of mixture components (M) set to the true number (4), and to infinity. Various values of α were used, each with a range of values for the β_j (all equal) from $\frac{1}{2}$ to 4 in geometric steps of $\sqrt{2}$.

The resulting performance at classification and whole-item prediction is shown in Figure 5. In the simulations on which these results are based, mixture components were assigned sequentially to data items (repeating, for $M < \infty$), and the simulation was then run for 100 iterations so as to reach equilibrium (as seen in the previous section, this is probably sufficient). Data was then collected for 400 iterations for use in classification and prediction of the 512 possible test items.

As expected, performance on the larger training sets was generally better than that on the smaller sets. Performance on training set S3 was particularly bad, a fact explained by the absence from this set of any good exemplar of one of the mixture components.

The results for both $M = 4$ and $M = \infty$ are quite insensitive to the value of α, within the range explored. Results for the largest α were slightly better. For $M = 4$ at least, this is as expected, since in the true distribution, the four components are all equally likely, a situation favoured by a high α.

Sensitivity to the value of the β_j is much greater. Since the attribute probabilities in the true distribution are all near 0 or 1, one would expect a values of β_j somewhat less than $N_j = 2$ to be preferred. This is indeed the case for the small training sets. For the large training sets, a value near 2 appears best, perhaps because these sets have enough data to force most of the probability to the region near the true parameter values without assistance from a prior with $\beta_j < N_j$ — a prior which would also have the undesirable effect of increasing the probability of parameter values more extreme than the true ones.

Perhaps surprisingly, performance with the number of mixture components set to infinity is as good as, even better than, that with the number of components fixed to its true value. This may seem paradoxical, but is explicable if one remembers that the value of α used is not optimal — setting α to 100000 with $M = 4$ in fact gives better results than any shown. The results certainly confirm the feasibility of using a model with $M = \infty$ when the number of mixture components is unknown or indefinite.

Performance of estimation with the EM algorithm. A parallel evaluation was done of MAP estimation using the EM algorithm. For $M = 4$, Figure 6 shows results for the same range of α and the β_j as were used with the Bayesian method. In an attempt to mimic the capability of the Bayesian method to give good results with $M = \infty$, runs with M set to the relatively large values of 10 and 100 are also shown.

In all cases, the EM algorithm was run for 100 iterations, which was ample to reach convergence, starting from (the same) three initial sets of parameter values. These starting points were obtained by setting the ϕ_g and $\psi_{g,j,v}$ to values from a uniform distribution over the interval $[1.0, 1.1]$ and then normalizing so that the parameter vectors summed to one as required. The parameters from the run that converged to the point with highest posterior probability density were then selected for use on the test items. When $\alpha < M$ and/or $\beta_j < N_j$, the posterior

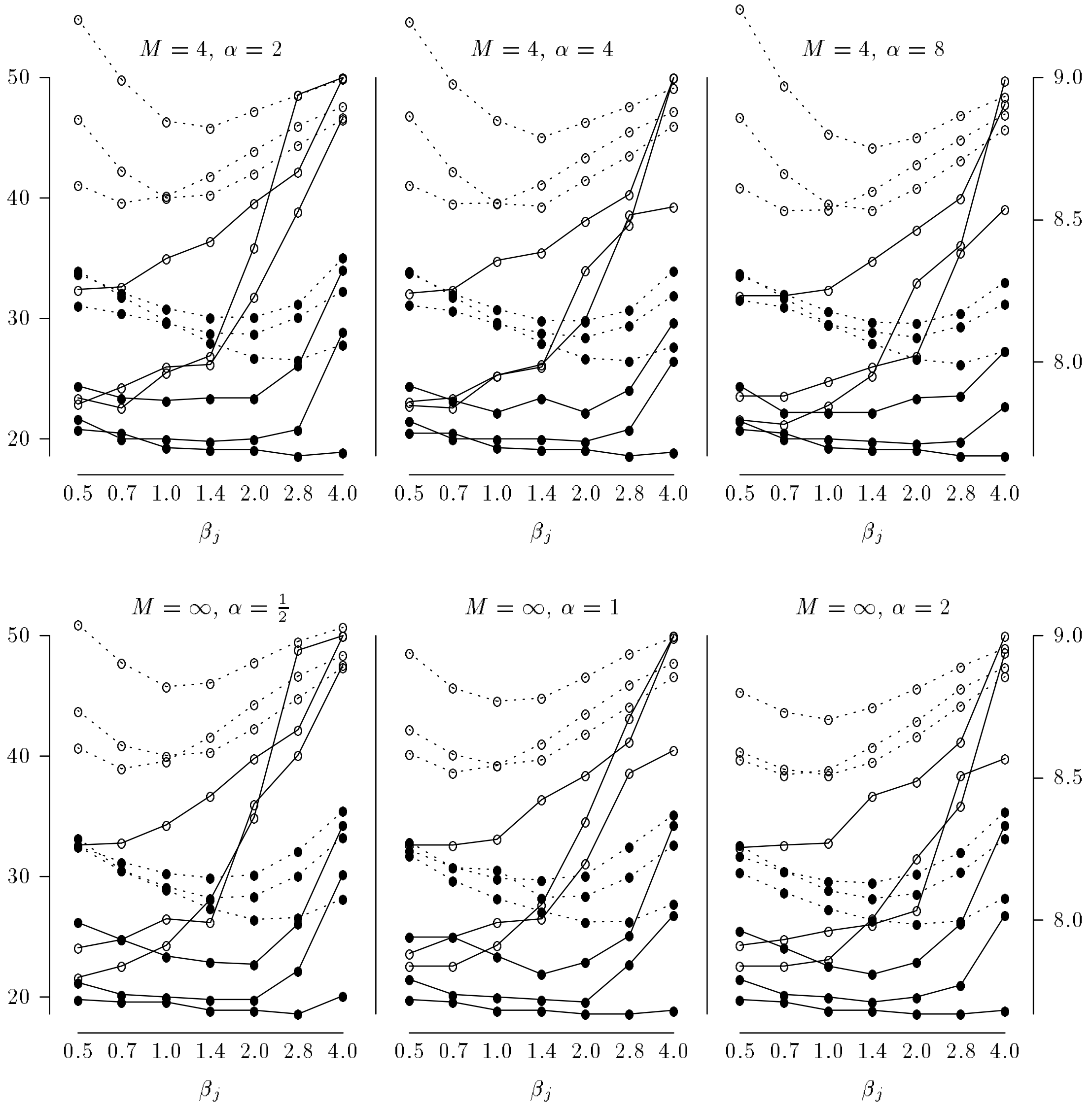

Figure 5: Performance of the Bayesian mixture modeling method. Results shown in the top row are with the number of mixture components (M) fixed at its true value; those in the bottom row with M set to infinity. In each case, graphs for various values of α are shown, with β varying along the horizontal scale. Classification performance is plotted with solid lines, using the scale at the left (percent errors). Performance at whole-item prediction is plotted with dotted lines, using the scale at the right (bits). Results on the small training sets are plotted with "$\circ$"; those on the large training sets with "$\bullet$".

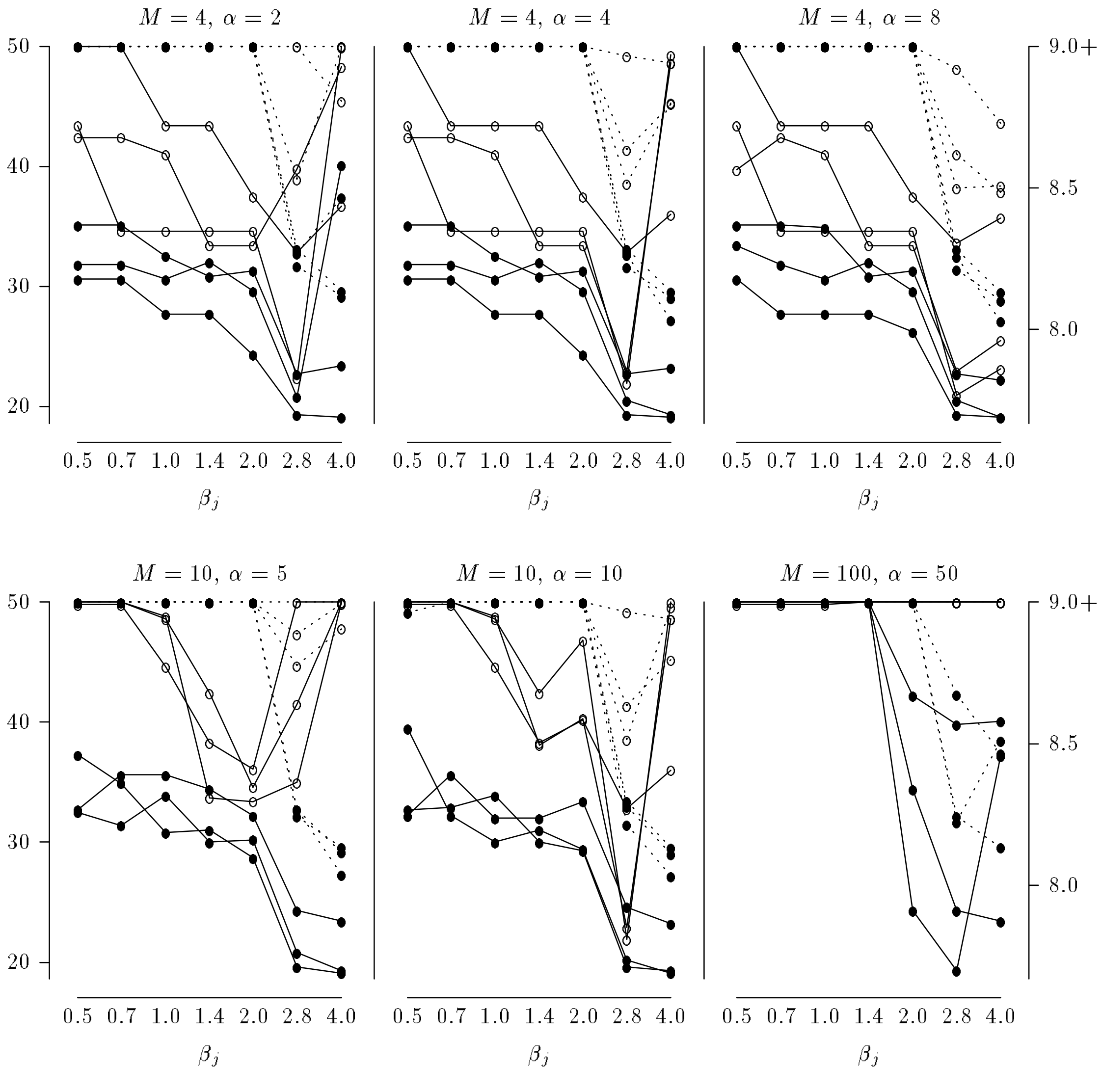

Figure 6: Performance of mixture modeling using the EM algorithm. Results shown in the top row are with the number of mixture components (M) fixed at its true value; those in the bottom row with M set to 10 and 100. Results for various values of α are shown, with β varying along the horizontal scale. Classification performance is plotted with solid lines, using the scale at the left (percent errors). Performance at whole-item prediction is plotted with dotted lines, using the scale at the right (bits), with performance worse than 9.0 shown as that value. Results on the small training sets are plotted with "$\circ$"; those on the large training sets with "$\bullet$".

density function has singularities at points where one or more parameter values are zero. In these cases, the densities at different points were compared as if all such zero parameters were set to the same infinitesimal value.

For $M = 4$, these results are qualitatively different from those obtained with the Bayesian method. Classification performance with $\beta_j \leq N_j = 2$ is relatively poor, especially with the small training sets — the reverse of the results with the Bayesian method. Whole-item prediction performance with such β_j is very poor (in fact, often infinitely poor). This is not surprising, since with $\beta_j \leq N_j$, the singularities in the prior distribution at zero values of the $\psi_{g,j,v}$ will encourage extreme overfitting. Results with $\alpha = 8$ and β_j equal to 2.8 and 4.0, values for which there are no singularities in the prior, show differences in both magnitude and trend from the corresponding Bayesian results as well.

Clearly, the result of MAP estimation with given α and β_j cannot be regarded as an approximation to the true Bayesian solution. A better correspondence appears if one regards the results of MAP estimation with $\beta_j > N_j$ as an approximation to the Bayesian result with $\beta'_j = \beta_j - N_j$. Such a correspondence might be expected by analogy with the situation with Dirichlet priors for a simple discrete distribution, though in the mixture case there appears to be no exact equivalence. Alternatively, one might simply regard the EM algorithm as performing maximum penalized likelihood estimation, abandoning any attempt to interpret it in a Bayesian framework.

Nevertheless, for $M = 4$, $\alpha = 8$, and $\beta_j = 4.0$, the results shown for MAP estimation are quite good. The attempt at mimicing the success of the Bayesian method when the number of components is regarded as unknown was somewhat less successful, and for $M = 100$ the results shown are also fragile — apparently minor changes to M and/or α can give much worse results.

Comparison of the methods. For both the Bayesian and the MAP estimation methods, the overall best values for α and the β_j with the number of components set to the true value and to infinity (or a reasonable facsimile) were selected based on performance on items from the true distribution, as shown in Figures 5 and 6. This is, of course, cheating. A proper Bayesian treatment of the situation where α and the β_j are not fixed by prior knowledge would involve treating them as hyperparameters, as is discussed below. For MAP estimation, cross-validation might be used to estimate these parameters from the training data. Such procedures lie outside the scope of this paper, however, so for what it is worth, the performance of the two methods with parameters selected by this not-quite-legitimate method is displayed for comparison in Figure 7.

Figure 7 also shows the performance of maximum likelihood estimation of the mixture distribution parameters, which is equivalent to MAP estimation with $\alpha = M$ and $\beta_j = N_j$. For additional perspective, the performance of a nearest neighbor classifier is shown as well. This classifier attributes a test item to the category of the training item that is closest to it in Hamming distance. If several training items are equally close, they vote on the category. If the vote is tied, it is assumed that a random choice is made; the performance shown is the average over such choices.

If the true number of mixture components is assumed known, little difference is apparent between the performance of the Bayesian method and that of MAP estimation. If the number of components is not known, but for MAP estimation is assumed to be no greater than 10, the Bayesian method gives significantly better performance at whole-item prediction, and perhaps

Modeling method	M	α	β_j
Bayesian	4	8	1.4
	∞	2	1.0
MAP estimation	4	8	4.0
	10	10	2.8
	100	50	2.8
Maximum likelihood	4	4	2
	10	10	2
Nearest neighbor	–	–	–

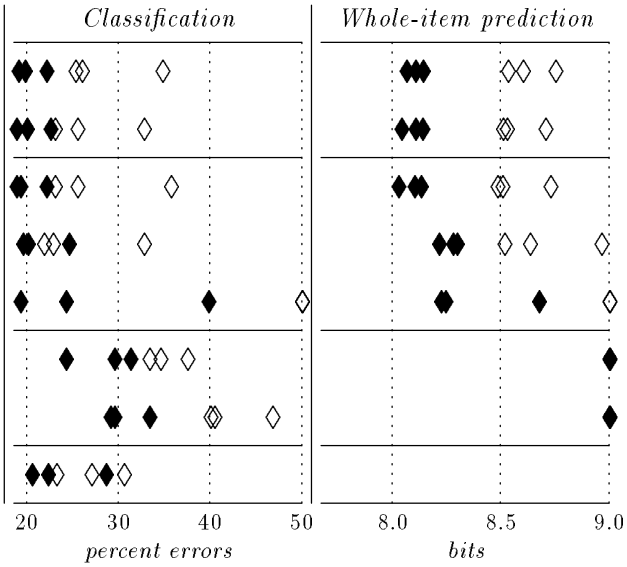

Figure 7: Comparative performance of Bayesian prediction, MAP estimation, maximum likelihood estimation, and nearest neighbor classification. Results on the small training sets are plotted with "◊"; those on the large training sets with "♦".

slightly better classification performance with the larger training sets. Performance of the MAP method deteriorates further if as many as 100 components are allowed.

Maximum likelihood estimation is not competitive with either the full Bayesian or the MAP methods in any context. Nearest neighbor classification does surprising well, however, especially for the small training sets. This is perhaps an indication that the problem is not tremendously difficult (though in some respects, such as the equal importance of all attributes, the problem is well-suited to this classifier).

All the methods compared required less than one minute of computation time on a 25 MIPS machine to classify the 512 possible test items on the basis of one of the large training sets. For small values of M, the EM algorithm is somewhat faster than the Bayesian method. The EM algorithm has a relative advantage if a single training set is used to classify many test items, since it processes the training data once, with the results then applied to any number of test items, while, at least in a straightforward implementation, the Bayesian method examines all test items throughout the simulation.

Clearly, results such as these on synthetic data can be suggestive only. A true picture of the worth of the Bayesian method can be obtained only by applying it to significant real data sets.

Conclusion

I have presented a method for performing Bayesian prediction from data modeled by a mixture distribution. The practical benefits of this method include simplicity of implementation, avoidance of overfitting, and some protection from local maxima in the computation. Of both practical and theoretical significance is the new approach to handling problems where the number of mixture components is either unknown or indefinite. This approach is made possible by the ability of the method to cope easily with countably infinite mixtures.

As is proper for a Bayesian solution to the problem, the method presented here does not produce a single "best" set of model parameters. This might be seen as undesirable for latent class analysis applications, but it would be an advantage if some convenient way of interpreting the full output of the simulation could be found. For the related task of finding clusters in a set of data items, one possible approach would be to accumulate a matrix recording how often each pair of items was assigned the same mixture component in the course of the simulation. The items could then be clustered on the basis of this similarity matrix by any of several methods [13].

In this paper, I have assumed that the user's prior knowledge can be adequately captured by a choice of values for α and the β_j in the prior distribution (equation (3)). Often, the real prior will not be so specific. In this case, α and the β_j can be treated as hyperparameters with their own prior distributions. At moderate computational cost, these hyperparameters can be included in the Monte Carlo simulation, allowing Bayesian inference for this hierachical model to be performed. Modifications to allow Dirichlet distributions that are not necessarily symmetrical with respect to the discrete attribute values are also possible.

Preliminary experiments with an algorithm encorporating these extensions have been performed. They show that one benefit of a hierarchical model is an increased ability to identify "noise" attributes that do not discriminate between any of the mixture components. The distributions for such attributes can be modeled at the hyperparameter level more economically than at the level of parameters for individual components, thereby eliminating the detrimental effects that spurious correlations among such noise attributes can have.

Although the development in this paper is confined to discrete data, the technique should also be applicable when items have real-valued attributes modeled by independent Gaussian distributions if an appropriate conjugate prior is used. The method can also be extended to the Hidden Markov Models used in speech recognition [15], and to the "noisy-OR" form of the stochastic neural networks described in [17].

Acknowledgements

I thank David MacKay, Geoff Hinton, and the members of the Connectionist Research Group at the University of Toronto for helpful comments on earlier versions of this paper. This work was supported by the Natural Sciences and Engineering Research Council of Canada, and by the Ontario Information Technology Research Centre.

References

[1] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985) A learning algorithm for Boltzmann machines, *Cognitive Science*, vol. 9, pp. 147-169.

[2] Andersen, E. B. (1982) Latent structure analysis: A survey, *Scandinavian Journal of Statistics*, vol. 9, pp. 1-12.

[3] Celeux, G. and Cedex, C. (1986) Validity tests in cluster analysis using a probabilistic teacher algorithm, *COMPSTAT 1986: Proceedings in Computational Statistics*, Heidelberg: Physica-Verlag.

[4] Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. (1988) AutoClass: A Bayesian classification system, *Proceedings of the Fifth International Conference on Machine Learning.*

[5] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977) Maximum likelihood from incomplete data via the EM algorithm (with discussion), *Journal of the Royal Statistical Society B*, vol. 39, pp. 1-38.

[6] Duda, R. O. and Hart, P. E. (1973) *Pattern Classification and Scene Analysis*, New York: John Wiley.

[7] Everitt, B. S. (1984) *An Introduction to Latent Variable Models*, London: Chapman and Hall.

[8] Everitt, B. S. and Hand, D. J. (1981) *Finite Mixture Distributions*, London: Chapman and Hall.

[9] Gelfand, A. E. and Smith, A. F. M. (1990) Sampling-based approaches to calculating marginal densities, *Journal of the American Statistical Association*, vol. 85, pp. 398-409.

[10] Geman, S. and Geman, D. (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721-741.

[11] Hanson, R., Stutz, J., and Cheeseman, P. (1991) Bayesian classification with correlation and inheritance, to be presented at the 12th International Joint Conference on Artificial Intelligence, Sydney, Australia, August 1991.

[12] Hastings, W. K. (1970) Monte Carlo sampling methods using Markov chains and their applications, *Biometrika*, vol. 57, no. 1, pp. 97-109.

[13] Jain, A. K. (1988) *Algorithms for Clustering Data*, Englewood Cliffs, New Jersey: Prentice Hall.

[14] Kai-Tai, F. and Yao-Ting, Z. (1990) *Generalized Multivariate Analysis*, Berlin: Springer-Verlag.

[15] Levinson, S. E., Rabiner, L. R., and Sondhi, M. M. (1983) An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition, *Bell System Technical Journal*, vol. 62, no. 4.

[16] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953) Equation of state calculations by fast computing machines, *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087-1092.

[17] Neal, R. M. (1990) Learning stochastic feedforward networks, University of Toronto, Department of Computer Science, Connectionist Research Group, Technical Report CRG-TR-90-7.

[18] Pearl, J. (1987) Evidential reasoning using stochastic simulation of causal models, *Artificial*

Intelligence, vol. 32, no. 2, pp. 245-257.

[19] Stone, M. (1974) Cross-validatory choice and assessment of statistical predictions (with discussion), *Journal of the Royal Statistical Society B*, vol. 36, pp. 111-147.

[20] Titterington, D. M., Smith, A. F. M., and Makov, U. E. (1985) *Statistical Analysis of Finite Mixture Distributions*, Chichester, New York: Wiley.

[21] Wei, G. C. G. and Tanner, M. A. (1990) A Monte Carlo implementation of the EM algorithm and the Poor Man's Data Augmentation algorithms, *Journal of the American Statistical Association*, vol. 85, pp. 699-704.

[22] Young, A. S. (1977) A Bayesian approach to prediction using polynomials, *Biometrika*, vol. 64, no. 2, pp. 309-317.