Nonlinear Models Using Dirichlet Process Mixtures

Babak Shahbaba BABAKS@UCI.EDU
Department of Statistics
University of California at Irvine
Irvine, CA, USA

Radford Neal RADFORD@STAT.TORONTO.EDU
Departments of Statistics and Computer Science
University of Toronto
Toronto, ON, Canada

Editor: Zoubin Ghahramani

Abstract

We introduce a new nonlinear model for classification, in which we model the joint distribution of response variable, y, and covariates, x, non-parametrically using Dirichlet process mixtures. We keep the relationship between y and x linear within each component of the mixture. The overall relationship becomes nonlinear if the mixture contains more than one component, with different regression coefficients. We use simulated data to compare the performance of this new approach to alternative methods such as multinomial logit (MNL) models, decision trees, and support vector machines. We also evaluate our approach on two classification problems: identifying the folding class of protein sequences and detecting Parkinson's disease. Our model can sometimes improve predictive accuracy. Moreover, by grouping observations into sub-populations (i.e., mixture components), our model can sometimes provide insight into hidden structure in the data.

Keywords: mixture models, Dirichlet process, classification

1. Introduction

In regression and classification models, estimation of parameters and interpretation of results are easier if we assume that distributions have simple forms (e.g., normal) and that the relationship between a response variable and covariates is linear. However, the performance of such a model depends on the appropriateness of these assumptions. Poor performance may result from assuming wrong distributions, or regarding relationships as linear when they are not. In this paper, we introduce a new model based on a Dirichlet process mixture of simple distributions, which is more flexible in capturing nonlinear relationships.

A Dirichlet process, $\mathcal{D}(G_0, \gamma)$, with baseline distribution G_0 and scale parameter γ, is a distribution over distributions. Ferguson (1973) introduced the Dirichlet process as a class of prior distributions for which the support is large, and the posterior distribution is manageable analytically. Using the Polya urn scheme, Blackwell and MacQueen (1973) showed that the distributions sampled from a Dirichlet process are discrete almost surely.

The idea of using a Dirichlet process as the prior for the mixing proportions of a simple distribution (e.g., Gaussian) was first introduced by Antoniak (1974). In this paper, we will describe the Dirichlet process mixture model as a limit of a finite mixture model (see Neal, 2000, for further description). Suppose exchangeable random values $y_1, \ldots, y_n$ are drawn independently from some unknown distribution. We can model the distribution of y as a mixture of simple distributions, with probability or density function

$$P(y) \quad = \quad \sum_{c=1}^{C} p_c F(y, \phi_c).$$

Here, p_c are the mixing proportions, and $F(y, \phi)$ is the probability or density for y under a distribution, $F(\phi)$, in some simple class with parameters ϕ—for example, a normal in which $\phi = (\mu, \sigma)$. We first assume that the number of mixing components, C, is finite. In this case, a common prior for p_c is a symmetric Dirichlet distribution, with density function

$$P(p_1, \ldots, p_C) \quad = \quad \frac{\Gamma(\gamma)}{\Gamma(\gamma/C)^C} \prod_{c=1}^{C} p_c^{(\gamma/C)-1},$$

where $p_c \geq 0$ and $\sum p_c = 1$. The parameters ϕ_c are independent under the prior, with distribution G_0. We can use mixture identifiers, c_i, and represent this model as follows:

$$\begin{aligned} y_i | c_i, \phi &\sim F(\phi_{c_i}), \\ c_i | p_1, \ldots, p_C &\sim Discrete(p_1, \ldots, p_C), \\ p_1, \ldots, p_C &\sim Dirichlet(\gamma/C, \ldots., \gamma/C), \\ \phi_c &\sim G_0. \end{aligned} \tag{1}$$

By integrating over the Dirichlet prior, we can eliminate the mixing proportions, p_c, and obtain the following conditional distribution for c_i:

$$P(c_i = c | c_1, \ldots, c_{i-1}) \quad = \quad \frac{n_{ic} + \gamma/C}{i - 1 + \gamma}. \tag{2}$$

Here, n_{ic} represents the number of data points previously (i.e., before the i^{th}) assigned to component c. As we can see, the above probability becomes higher as n_{ic} increases.

When C goes to infinity, the conditional probabilities (2) reach the following limits:

$$\begin{aligned} P(c_i = c | c_1, \ldots, c_{i-1}) &\rightarrow \frac{n_{ic}}{i-1+\gamma}, \\ P(c_i \neq c_j \text{ for all } j < i | c_1, \ldots, c_{i-1}) &\rightarrow \frac{\gamma}{i-1+\gamma}. \end{aligned}$$

As a result, the conditional probability for θ_i, where $\theta_i = \phi_{c_i}$, becomes

$$\theta_i | \theta_1, \ldots, \theta_{i-1} \quad \sim \quad \frac{1}{i-1+\gamma} \sum_{j<i} \delta_{\theta_j} + \frac{\gamma}{i-1+\gamma} G_0, \tag{3}$$

where δ_θ is a point mass distribution at θ. Since the observations are assumed to be exchangeable, we can regard any observation, i, as the last observation and write the conditional probability of θ_i given the other θ_j for $j \neq i$ (written θ_{-i}) as follows:

$$\theta_i | \theta_{-i} \quad \sim \quad \frac{1}{n-1+\gamma} \sum_{j \neq i} \delta_{\theta_j} \; + \; \frac{\gamma}{n-1+\gamma} G_0. \tag{4}$$

The above conditional probabilities are equivalent to the conditional probabilities for θ_i according to the Dirichlet process mixture model (as presented by Blackwell and MacQueen, 1973, using the Polya urn scheme), which has the following form:

$$\begin{aligned} y_i|\theta_i &\sim F(\theta_i), \\ \theta_i|G &\sim G, \\ G &\sim \mathcal{D}(G_0,\gamma). \end{aligned} \tag{5}$$

That is, If we let $\theta_i = \phi_{c_i}$, the limit of model (1) as $C \to \infty$ becomes equivalent to the Dirichlet process mixture model (Ferguson, 1983; Neal, 2000). In model (5), G is the distribution over θ's, and has a Dirichlet process prior, $\mathcal{D}$. Phrased this way, each data point, i, has its own parameters, θ_i, drawn independently from a distribution that is drawn from a Dirichlet process prior. But since distributions drawn from a Dirichlet process are discrete (almost surely) as shown by Blackwell and MacQueen (1973), the θ_i for different data points may be the same.

The parameters of the Dirichlet process prior are G_0, a distribution from which θ's are sampled, and γ, a positive scale parameter that controls the number of components of the mixture that will be represented in the sample, such that a larger γ results in a larger number of components. To illustrate the effect of γ on the number of mixture components in a sample of size 200, we generated samples from four different Dirichlet process priors with $\gamma = 0.1, 1, 5, 10$, and the same baseline distribution $G_0 = N_2(0, 10I_2)$ (where I_2 is a 2×2 identity matrix). For a given value of γ, we first sample θ_i, where $i = 1, \ldots, 200$, according to the conditional probabilities (3), and then we sample $y_i|\theta_i \sim N_2(\theta_i, 0.2I_2)$. The data generated according to these priors are shown in Figure 1. As we can see, the (prior) expected number of components in a finite sample increases as γ becomes larger.

With a Dirichlet process prior, we can we find conditional distributions of the posterior distribution of model parameters by combining the conditional prior probability of (4) with the likelihood $F(y_i, \theta_i)$, obtaining

$$\theta_i|\theta_{-i}, y_i \sim \sum_{j \neq i} q_{ij}\delta_{\theta_j} + r_i H_i, \tag{6}$$

where H_i is the posterior distribution of θ based on the prior G_0 and the single data point y_i, and the values of the q_{ij} and r_i are defined as follows:

$$\begin{aligned} q_{ij} &= bF(y_i, \theta_j), \\ r_i &= b\gamma \int F(y_i, \theta) dG_0(\theta). \end{aligned}$$

Here, b is such that $r_i + \sum_{j \neq i} q_{ij} = 1$. These conditional posterior distributions are what are needed when sampling the posterior using MCMC methods, as discussed further in Section 2.

Bush and MacEachern (1996), Escobar and West (1995), MacEachern and Müller (1998), and Neal (2000) have used Dirichlet process mixture models for density estimation. Müller et al. (1996) used this method for curve fitting. They model the joint distribution of data pairs (x_i, y_i) as a Dirichlet process mixture of multivariate normals. The conditional distribution, $P(y|x)$, and the expected value, $E(y|x)$, are estimated based on this distribution for a grid of x's (with interpolation) to obtain a nonparametric curve. The application of this approach (as presented by Müller et al., 1996) is restricted to continuous variables. Moreover, this model is feasible only for problems with a small number of covariates, p. For data with moderate to large dimensionality, estimation of the joint

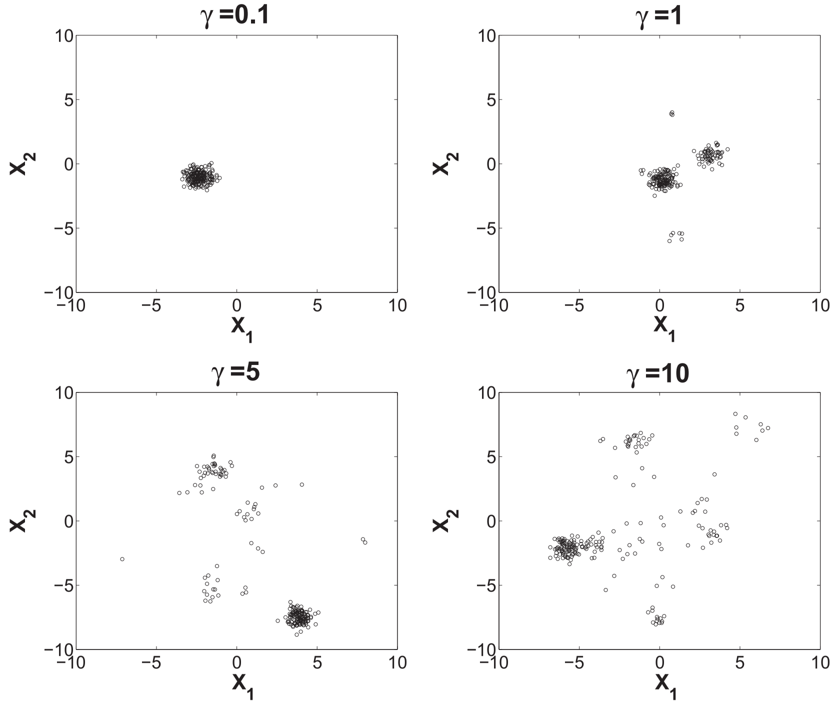


Figure 1: Data sets of size $n = 200$ generated according to four different Dirichlet process mixture priors, each with the same baseline distribution, $G_0 = N_2(0, 10I_2)$, but different scale parameters, γ. As γ increases, the expected number of components present in the sample becomes larger. (Note that, as can be seen above, when γ is large, many of these components have substantial overlap.)

distribution is very difficult both statistically and computationally. This is mostly due to the difficulties that arise when simulating from the posterior distribution of large full covariance matrices. In this approach, if a mixture model has C components, the set of full covariance matrices have $Cp(p+1)/2$ parameters. For large p, the computational burden of estimating these parameters might be overwhelming. Estimating full covariance matrices can also cause statistical difficulties since we need to assure that covariance matrices are positive semidefinite. Conjugate priors based the inverse Wishart distribution satisfy this requirement, but they lack flexibility (Daniels and Kass, 1999). Flat priors may not be suitable either, since they can lead to improper posterior distributions, and they can be unintentionally informative (Daniels and Kass, 1999). A common approach to address these issues is to use decomposition methods in specifying priors for full covariance matrices (see for example, Daniels and Kass, 1999; Cai and Dunson, 2006). Although this approach has demonstrated some computational advantages over direct estimation of full covariance matrices, it is not yet feasible for high-dimensional variables. For example, Cai and Dunson (2006) recommend their approach only for problems with less than 20 covariates.

We introduce a new nonlinear Bayesian model, which also nonparametrically estimates $P(x,y)$, the joint distribution of the response variable y and covariates x, using Dirichlet process mixtures. Within each component, we assume the covariates are independent, and model the dependence between y and x using a linear model. Therefore, unlike the method of Müller et al. (1996), our approach can be used for modeling data with a large number of covariates, since the covariance matrix for one mixture component is highly restricted. Using the Dirichlet process as the prior, our method has a built-in mechanism to avoid overfitting since the complexity of the nonlinear model is controlled. Moreover, this method can be used for categorical as well as continuous response variables by using a generalized linear model instead of the linear model.

The idea of building a nonlinear model based on an ensemble of simple linear models has been explored extensively in the field of machine learning. Jacobs et al. (1991) introduced a supervised learning procedure for models that are comprised of several local models (experts) each handling a subset of data. A gating network decides which expert should be used for a given data point. For inferring the parameters of such models, Waterhouse et al. (1996) provided a Bayesian framework to avoid over-fitting and noise level under-estimation problems associated with traditional maximum likelihood inference. Rasmussen and Ghahramani (2002) generalized mixture of experts models by using infinitely many nonlinear experts. In their approach, each expert is assumed to be a Gaussian process regression model, and the gating network is based on an input-dependent adaptation of Dirichlet process. Meeds and Osindero (2006) followed the same idea, but instead of assuming that the covariates are fixed, they proposed a joint mixture of experts model over covariates and response variable.

Our focus here is on classification models with a multi-category response, in which we have observed data for n cases, $(x_1,y_1),...,(x_n,y_n)$. Here, the class y_i has J possible values, and the covariates x_i can in general be a vector of p covariates. We wish to classify future cases in which only the covariates are observed. For binary ($J=2$) classification problems, a simple logistic model can be used, with class probabilities defined as follows (with the case subscript dropped from x and y):

$$P(y=1|x,\alpha,\beta) \quad = \quad \frac{\exp(\alpha+x^T\beta)}{1+\exp(\alpha+x^T\beta)}.$$

When there are three or more classes, we can use a generalization known as the multinomial logit (MNL) model (called "softmax" in the machine learning literature):

$$P(y=j|x,\alpha,\beta) \quad = \quad \frac{\exp(\alpha_j+x^T\beta_j)}{\sum_{j'=1}^{J}\exp(\alpha_{j'}+x^T\beta_{j'})}.$$

MNL models are *discriminative*, since they model the conditional distribution $P(y|x)$, but not the distribution of covariates, $P(x)$. In contrast, our dpMNL model is *generative*, since it estimates the joint distribution of response and covariates, $P(x,y)$. The joint distribution can be decompose into the product of the marginal distribution $P(x)$ and the conditional distribution $P(y|x)$; that is, $P(x,y)=P(x)P(y|x)$.

Generative models have several advantages over discriminative models (see for example, Ulusoy and Bishop, 2005). They provide a natural framework for handling missing data or partially labeled data. They can also augment small quantities of expensive labeled data with large quantities of cheap unlabeled data. This is especially useful in applications such as document labeling and image analysis, where it may provide better predictions for new feature patterns not present in the data at

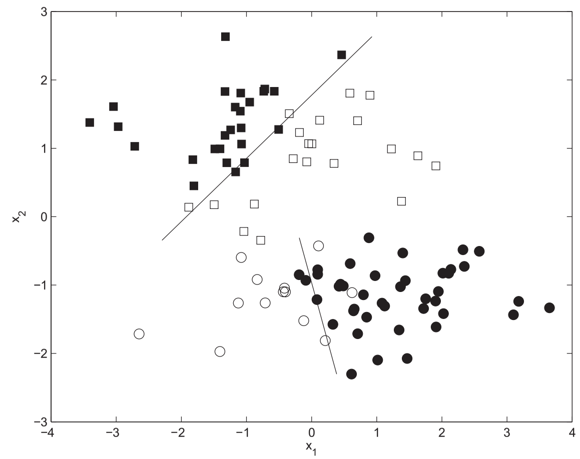


Figure 2: An illustration of our model for a binary (black and white) classification problem with two covariates. Here, the mixture has two components, which are shown with circles and squares. In each component, an MNL model separates the two classes into "black" or "white" with a linear decision boundary. The overall decision boundary, which is a smooth function, is not shown in this figure.

the time of training. For example, the Latent Dirichlet Allocation (LDA) model proposed by Blei et al. (2003) is a well-defined generative model that performs well in classifying documents with previously unknown patterns.

While generative models are quite successful in many problems, they can be computationally intensive. Moreover, finding a good (but not perfect) estimate for the joint distribution of all variables (i.e., x and y) does not in general guarantee a good estimate of decision boundaries. By contrast, discriminative models are often computationally fast and are preferred when the covariates are in fact non-random (e.g., they are fixed by an experimental design).

Using a generative model in our proposed method provides an additional benefit. Modeling the distribution of covariates jointly with y allows us to implicitly model the dependency of covariates on each other through clustering (i.e., assigning data points to different components), which could provide insight into hidden structure in the data. To illustrate this concept, consider Figure 2 where the objective is to classify cases into black or white. To improve predictive accuracy, our model has divided the data into two components, shown as squares and circles. These components are distinguished primarily by the value of the second covariate, x_2, which is usually positive for squares and negative for circles. For cases in the squares group, the response variable strongly depends on both x_1 and x_2 (the linear separator is almost diagonal), whereas, for cases in the circles group, the model mainly depends on x_1 alone (the linear model is almost vertical). Therefore, by grouping the data into sub-populations (e.g., circles and squares in this example), our model not only improves classification accuracy, but also discovers hidden structure in the data (i.e., by clustering covariate observations). This concept is briefly discussed in Section 5, where we use our model to predict Parkinson's disease. A more detailed discussion on using our method to detect hidden structure in data is provided elsewhere (Shahbaba, 2009), where a Dirichlet process mixture of autoregressive models us used to analyze time-series processes that are subject to regime changes, with no specific economic theory about the structure of the model.

The next section describes our methodology. In Section 3, we illustrate our approach and evaluate its performance on simulated data. In Section 4, we present the results of applying our model to an actual classification problem, which attempts to identify the folding class of a protein sequence based on the composition of its amino acids. Section 5 discusses another real classification problem, where the objective is to detect Parkinson's disease. This example is provided to show how our method can be used not only for improving prediction accuracy, but also for identifying hidden structure in the data. Finally, Section 6 is devoted to discussion and future directions.

2. Methodology

We now describe our classification model, which we call dpMNL, in detail. We assume that for each case we observe a vector of continuous covariates, x, of dimension p. The response variable, y, is categorical, with J classes. To model the relationship between y and x, we non-parametrically model the joint distribution of y and x, in the form $P(x,y) = P(x)P(y|x)$, using a Dirichlet process mixture. Within each component of the mixture, the relationship between y and x (i.e., $P(y|x)$) is expressed using a linear function. The overall relationship becomes nonlinear if the mixture contains more than one component. This way, while we relax the assumption of linearity, the flexibility of the relationship is controlled.

Each component in the mixture model has parameters $\theta = (\mu, \sigma^2, \alpha, \beta)$. The distribution of x within a component is multivariate normal, with mean vector μ and diagonal covariance, with the vector σ^2 on the diagonal. The distribution of y given x within a component is given by a multinomial logit (MNL) model—for $j = 1, \ldots, J$,

$$P(y = j|x, \alpha, \beta) \quad = \quad \frac{\exp(\alpha_j + x^T \beta_j)}{\sum_{j'=1}^{J} \exp(\alpha_{j'} + x^T \beta_{j'})}.$$

The parameter α_j is scalar, and β_j is a vector of length p. Note that given x, the distribution of y does not depend on μ and σ. This representation of the MNL model is redundant, since one of the β_j's (where $j = 1, ..., J$) can be set to zero without changing the set of relationships expressible with the model, but removing this redundancy would make it difficult to specify a prior that treats all classes symmetrically. In this parameterization, what matters is the difference between the parameters of different classes.

In addition to the mixture view, which follows Equation (1) with $C \to \infty$, one can also view each observation, i, as having its own parameter, θ_i, drawn independently from a distribution drawn from a Dirichlet process, as in Equation (5):

$$\begin{aligned} \theta_i|G &\sim G, \ \text{ for } i = 1, \ldots, n, \\ G &\sim \mathcal{D}(G_0, \gamma). \end{aligned}$$

Since G will be discrete, many groups of observations will have identical θ_i, corresponding to components in the mixture view.

Although the covariates in each component are assumed to be independent with normal priors, this independence of covariates exists only locally (within a component). Their global (over all components) dependency is modeled by assigning data to different components (i.e., clustering). The relationship between y and x within a component is captured using an MNL model. Therefore, the relationship is linear locally, but nonlinear globally.

We could assume that y and x are independent within components, and capture the dependence between the response and the covariates by clustering too. However, this may lead to poor performance (e.g., when predicting the response for new observations) if the dependence of y on x is difficult to capture using clustering alone. Alternatively, we could also assume that the covariates are dependent within a component. For continuous response variables, this becomes equivalent to the model proposed by Müller et al. (1996). If the covariates are in fact dependent, using full covariance matrices (as suggested by Müller et al., 1996) could result in a more parsimonious model since the number of mixture component would be smaller. However, as we discussed above, this approach may be practically infeasible for problems with a moderate to large number of covariates. We believe that our method is an appropriate compromise between these two alternatives.

We define G_0, which is a distribution over $\theta = (\mu, \sigma^2, \alpha, \beta)$, as follows:

$$\begin{aligned} \mu_l|\mu_0, \sigma_0 &\sim N(\mu_0, \sigma_0^2), \\ \log(\sigma_l^2)|M_\sigma, V_\sigma &\sim N(M_\sigma, V_\sigma^2), \\ \alpha_j|\tau &\sim N(0, \tau^2), \\ \beta_{jl}|\nu &\sim N(0, \nu^2). \end{aligned}$$

The parameters of G_0 may in turn depend on higher level hyperparameters. For example, we can regard the variances of coefficients as hyperparameters with the following priors:

$$\begin{aligned} \log(\tau^2)|M_\tau, V_\tau &\sim N(M_\tau, V_\tau^2), \\ \log(\nu^2)|M_\nu, V_\nu &\sim N(M_\nu, V_\nu^2). \end{aligned}$$

We use MCMC algorithms for posterior sampling. We could use Gibbs sampling if G_0 is the conjugate prior for the likelihood given by F. That is, we would repeatedly draw samples from $\theta_i|\theta_{-i}, y_i$ (where $i = 1, ..., n$) using the conditional distribution (6). Neal (2000) presented several algorithms for sampling from the posterior distribution of Dirichlet process mixtures when non-conjugate priors are used. Throughout this paper, we use Gibbs sampling with auxiliary parameters (Neal's algorithm 8).

This algorithm uses a Markov chain whose state consists of $c_1, ..., c_n$ and $\phi = (\phi_c : c \in \{c_1, ..., c_n\})$, so that $\theta_i = \phi_{c_i}$. In order to allow creation of new clusters, the algorithm temporarily supplements the ϕ_c parameters of existing clusters with m (or $m-1$) additional parameter values drawn from the prior, where m a postive integer that can be adjusted to give good performance. Each iteration of the Markov chain simulation operates as follows:

- For $i = 1, ..., n$: Let k^- be the number of distinct c_j for $j \neq i$ and let $h = k^- + m$. Label these c_j with values in $\{1, ..., k^-\}$. If $c_i = c_j$ for some $j \neq i$, draw values independently from G_0 for those ϕ_c for which $k^- < c \leq h$. If $c_i \neq c_j$ for all $j \neq i$, let c_i have the label $k^- + 1$, and draw values independently from G_0 for those ϕ_c where $k^- + 1 < c \leq h$. Draw a new value for c_i from $\{1, ..., h\}$ using the following probabilities:

$$P(c_i = c|c_{-i}, y_i, \phi_1, ..., \phi_h) \quad = \quad \begin{cases} b \dfrac{n_{-i,c}}{n-1+\gamma} F(y_i, \phi_c) & \text{for } 1 \leq c \leq k^-, \\[2ex] b \dfrac{\gamma/m}{n-1+\gamma} F(y_i, \phi_c) & \text{for } k^- < c \leq h, \end{cases}$$

where $n_{-i,c}$ is the number of c_j for $j \neq i$ that are equal to c, and b is the appropriate normalizing constant. Change the state to contain only those ϕ_c that are now associated with at least to one observation.

- For all $c \in \{c_1, \ldots, c_n\}$ draw a new value from the distribution $\phi_c \mid \{y_i$ such that $c_i = c\}$, or perform some update that leaves this distribution invariant.

Throughout this paper, we set $m = 5$. This algorithm resembles one proposed by MacEachern and Müller (1998), with the difference that the auxiliary parameters exist only temporarily, which avoids an inefficiency in MacEachern and Müller's algorithm.

Samples simulated from the posterior distribution are used to estimate posterior predictive probabilities. For a new case with covariates x', the posterior predictive probability of the response variable, y', is estimated as follows:

$$P(y' = j|x') = \frac{P(y' = j, x')}{P(x')},$$

where

$$P(y' = j, x') = \frac{1}{S}\sum_{s=1}^{S} P(y' = j, x'|G_0, \theta^{(s)}),$$
$$P(x') = \frac{1}{S}\sum_{s=1}^{S} P(x'|G_0, \theta^{(s)}).$$

Here, S is the number of post-convergence samples from MCMC, and $\theta^{(s)}$ represents the set of parameters obtained at iteration s. Alternatively, we could predict new cases using $P(y' = j, x') = \frac{1}{S}\sum_{s=1}^{S} P(y' = j|x', G_0, \theta^{(s)})$. While this would be computationally faster, the above approach allows us to learn from the covariates of test cases when predicting their response values. Note also that the above predictive probabilities include the possibility that the test case is from a new cluster.

We use these posterior predictive probabilities to make predictions for test cases, by assigning each test case to the class with the highest posterior predictive probability. This is the optimal strategy for a simple 0/1 loss function. In general, we could use more problem-specific loss functions and modify our prediction strategy accordingly.

Implementations for all our models were coded in MATLAB, and are available online at `http://www.ics.uci.edu/~babaks/codes`.

3. Results for Simulated Data

In this section, we illustrate our dpMNL model using synthetic data, and compare it with other models as follows:

- A simple MNL model, fitted by maximum likelihood or Bayesian methods.
- A Bayesian MNL model with quadratic terms (i.e., $x_l x_k$, where $l = 1, \ldots, p$ and $k = 1, \ldots, p$), referred to as qMNL.
- A decision tree model (Breiman et al., 1993) that uses 10-fold cross-validation for pruning, as implemented by the MATLAB "treefit", "treetest" (for cross-validation) and "treeprune" functions.
- Support Vector Machines (SVMs) (Vapnik, 1995), implemented with the MATLAB "svmtrain" and "svmclassify" functions from the Bioinformatics toolbox. Both a linear SVM (LSVM) and a nonlinear SVM with radial basis function kernel (RBF-SVM) were tried.

When the number of classes in a classification problem was bigger than two, LSVM and RBF-SVM used the all-vs-all scheme as suggested by Allwein et al. (2000), Fürnkranz (2002), and Hsu and Lin (2002). In this scheme, $\binom{J}{2}$ binary classifiers are trained where each classifier separates a pair of classes. The predicted class for each test case is decided by using a majority voting scheme where the class with the highest number of votes among all binary classes wins. For RBF-SVM, the scaling parameter, λ, in the RBF kernel, $k(x, x') = \exp(-||x - x'||/2\lambda)$, was optimized based on a validation set comprised of 20% of training samples.

The models are compared with respect to their accuracy rate and the F_1 measure. Accuracy rate is defined as the percentage of the times the correct class is predicted. F_1 is a common measurement in machine learning defined as:

$$F_1 = \frac{1}{J}\sum_{j=1}^{J} \frac{2A_j}{2A_j + B_j + C_j},$$

where A_j is the number of cases which are correctly assigned to class j, B_j is the number cases incorrectly assigned to class j, and C_j is the number of cases which belong to the class j but are assigned to other classes.

We do two tests. In the first test, we generate data according to the dpMNL model. Our objective is to evaluate the performance of our model when the distribution of data is comprised of multiple components. In the second test, we generate data using a smooth nonlinear function. Our goal is to evaluate the robustness of our model when data actually come from a different model.

3.1 Simulation 1

The first test was on a synthetic four-way classification problem with five covariates. Data are generated according to our dpMNL model, except the number of components was fixed at two. Two hyperparameters defining G_0 were given the following priors:

$$\log(\tau^2) \sim N(0, 0.1^2),$$
$$\log(\nu^2) \sim N(0, 2^2).$$

The prior for component parameters $\theta = (\mu, \sigma^2, \alpha, \beta)$ defined by this G_0 was

$$\mu_l \sim N(0, 1),$$
$$\log(\sigma_l^2) \sim N(0, 2^2),$$
$$\alpha_j|\tau \sim N(0, \tau^2),$$
$$\beta_{jl}|\nu \sim N(0, \nu^2),$$

where $l = 1, \ldots, 5$ and $j = 1, \ldots, 4$. We randomly draw parameters θ_1 and θ_2 for two components as described from this prior. For each component, we then generate 5000 data points by first drawing $x_{il} \sim N(\mu_l, \sigma_l)$ and then sampling y using the following MNL model:

$$P(y = j|x, \alpha, \beta) = \frac{\exp(\alpha_j + x\beta_j)}{\sum_{j'=1}^{J} \exp(\alpha_{j'} + x\beta_{j'})}.$$

Model	Accuracy (%)	F_1 (%)
Baseline	45.57 (1.47)	15.48 (1.77)
MNL (Maximum Likelihood)	77.30 (1.23)	66.65 (1.41)
MNL	78.39 (1.32)	66.52 (1.72)
qMNL	83.60 (0.99)	74.16 (1.30)
Tree (Cross Validation)	70.87 (1.40)	55.82 (1.69)
LSVM	78.61 (1.17)	67.03 (1.51)
RBF-SVM	79.09 (0.99)	63.65 (1.44)
dpMNL	89.21 (0.65)	81.00 (1.23)

Table 1: Simulation 1: the average performance of models based on 50 simulated data sets. The Baseline model assigns test cases to the class with the highest frequency in the training set. Standard errors of estimates (based on 50 repetitions) are provided in parentheses.

The overall sample size is 10000. We randomly split the data into a training set, with 100 data points, and a test set, with 9900 data points. We use the training set to fit the models, and use the independent test set to evaluate their performance. The regression parameters of the Bayesian MNL model with Bayesian estimation and the qMNL model have the following priors:

$$\begin{aligned} \alpha_j|\tau &\sim N(0,\tau^2), \\ \beta_{jl}|\nu &\sim N(0,\nu^2), \\ \log(\tau) &\sim N(0,1^2), \\ \log(\nu) &\sim N(0,2^2). \end{aligned}$$

The above procedure was repeated 50 times. Each time, new hyperparameters, τ^2 and ν^2, and new component parameters, θ_1 and θ_2, were sampled, and a new data set was created based on these θ's.

We used Hamiltonian dynamics (Neal, 1993) for updating the regression parameters (the α's and β's). For all other parameters, we used single-variable slice sampling (Neal, 2003) with the "stepping out" procedure to find an interval around the current point, and then the "shrinkage" procedure to sample from this interval. We also used slice sampling for updating the concentration parameter γ, We used $\log(\gamma) \sim N(-3,2^2)$ as the prior, which, encourages smaller values of γ, and hence a smaller number of components. Note that the likelihood for γ depends only on C, the number of unique components (Neal, 2000; Escobar and West, 1995). For all models we ran 5000 MCMC iterations to sample from the posterior distributions. We discarded the initial 500 samples and used the rest for prediction.

Our dpMNL model has the highest computational cost compared to all other methods. Simulating the Markov chain took about 0.15 seconds per iteration using a MATLAB implementation on an UltraSPARC III machine (approximately 12.5 minutes for each simulated data set). Each MCMC iteration for the Bayesian MNL model took about 0.1 second (approximately 8 minutes for each data set). Training the RBF-SVM model (with optimization of the scale parameter) took approximately 1 second for each data set. Therefore, SVM models have a substantial advantage over our approach in terms of computational cost.

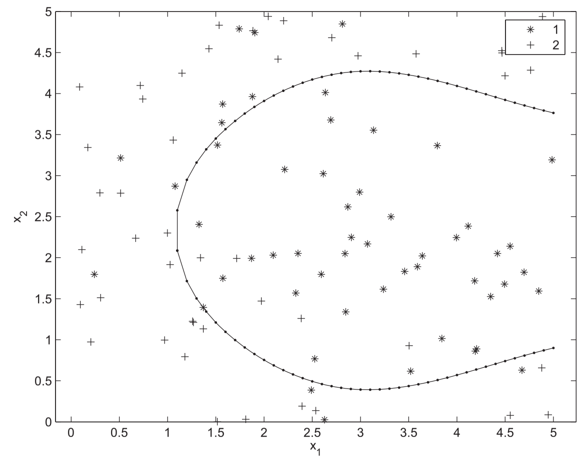


Figure 3: A random sample generated according to Simulation 2, with $a_3 = 0$. The dotted line is the optimal boundary function.

The average results (over 50 repetitions) are presented in Table 1. As we can see, our dpMNL model provides better results compared to all other models. The improvements are statistically significant (p-values < 0.001) for comparisons of accuracy rates using a paired t-test with $n = 50$.

3.2 Simulation 2

In the above simulation, since the data were generated according to the dpMNL model, it is not surprising that this model had the best performance compared to other models. In fact, as we increase the number of components, the amount of improvement using our model becomes more and more substantial (results not shown). To evaluate the robustness of the dpMNL model, we performed another test. This time, we generated x_{i1}, x_{i2}, x_{i3} (where $i = 1, \ldots, 10000$) from the $Uniform(0,5)$ distribution, and generated a binary response variable, y_i, according the following model:

$$P(y=1|x) = \frac{1}{1+\exp[a_1\sin(x_1^{1.04}+1.2)+x_1\cos(a_2x_2+0.7)+a_3x_3-2]},$$

where a_1, a_2 and a_3 are randomly sampled from $N(1,0.5^2)$. The function used to generate y is a smooth nonlinear function of covariates. The covariates are not clustered, so the generated data do not conform with the assumptions of our model. Moreover, this function includes a completely arbitrary set of constants to ensure the results are generalizable. Figure 3 shows a random sample from this model except that a_3 is fixed at zero (so x_3 is ignored). In this figure, the dotted line is the optimal decision boundary.

Table 2 shows the results for this simulation, which are averages over 50 data sets. For each data set, we generated 10000 cases by sampling new values for a_1, a_2, and a_3, new covariates, x, for each case, and new values for the response variable, y, in each case. As before, models were trained on 100 cases, and tested on the remaining 9900. As before, the dpMNL model provides significantly

Model	Accuracy (%)	F_1 (%)
Baseline	61.96 (1.53)	37.99 (0.57)
MNL (Maximum Likelihood)	73.58 (0.96)	68.33 (1.17)
MNL	73.58 (0.97)	67.92 (1.41)
qMNL	75.60 (0.98)	70.12 (1.36)
Tree (Cross Validation)	73.47 (0.95)	66.94 (1.43)
LSVM Linear	73.09 (0.99)	64.95 (1.71)
RBF-SVM	76.06 (0.94)	68.46 (1.77)
dpMNL	77.80 (0.86)	73.13 (1.26)

Table 2: Simulation 2: the average performance of models based on 50 simulated data sets. The Baseline model assigns test cases to the class with the highest frequency in the training set. Standard errors of estimates (based on 50 repetitions) are provided in parentheses.

(all p-values are smaller than 0.001) better performance compared to all other models. This time, however, the performances of qMNL and RBF-SVM are closer to the performance of the dpMNL model.

4. Results on Real Classification Problems

In this section, we first apply our model the problem of predicting a protein's 3D structure (i.e., folding class) based on its sequence. We then use our model to identify patients with Parkinson's disease (PD) based on their speech signals.

4.1 Protein Fold Classification

When predicting a protein's 3D structure, it is common to presume that the number of possible folds is fixed, and use a classification model to assign a protein to one of these folding classes. There are more than 600 folding patterns identified in the SCOP (Structural Classification of Proteins) database (Lo Conte et al., 2000). In this database, proteins are considered to have the same folding class if they have the same major secondary structure in the same arrangement with the same topological connections.

We apply our model to a protein fold recognition data set provided by Ding and Dubchak (2001). The proteins in this data set are obtained from the PDB_select database (Hobohm et al., 1992; Hobohm and Sander, 1994) such that two proteins have no more than 35% of the sequence identity for aligned subsequences larger than 80 residues. Originally, the resulting data set included 128 unique folds. However, Ding and Dubchak (2001) selected only the 27 most populated folds (311 proteins) for their analysis. They evaluated their models based on an independent sample (i.e., test set) obtained from PDB-40D (Lo Conte et al., 2000). PDB-40D contains the SCOP sequences with less than 40% identity with each other. Ding and Dubchak (2001) selected 383 representatives of the same 27 folds in the training set with no more than 35% identity to the training sequences. The training and test data sets are available online at `http://crd.lbl.gov/~cding/protein/`.

The covariates in these data sets are the length of the protein sequence, and the percentage composition of the 20 amino acids. While there might exist more informative covariates to predict protein folds, we use these so that we can compare the results of our model to that of Ding and Dubchak (2001), who trained several Support Vector Machines (SVM) with nonlinear kernel functions.

We centered the covariates so they have mean zero, and used the following priors for the MNL model and qMNL model (with no interactions, only x_i and x_i^2 as covariates):

$$\begin{aligned}
\alpha_j|\eta &\sim N(0,\eta^2),\\
\log(\eta^2) &\sim N(0,2^2),\\
\beta_{jl}|\xi,\sigma_l &\sim N(0,\xi^2\sigma_l^2),\\
\log(\xi^2) &\sim N(0,1),\\
\log(\sigma_l^2) &\sim N(-3,4^2).
\end{aligned}$$

Here, the hyperparameters for the variances of regression parameters are more elaborate than in the previous section. One hyperparameter, σ_l, is used to control the variance of all coefficients, β_{jl} (where $j = 1,...,J$), for covariate x_l. If a covariate is irrelevant, its hyperparameter will tend to be small, forcing the coefficients for that covariate to be near zero. This method, termed Automatic Relevance Determination (ARD), has previously been applied to neural network models by Neal (1996). We also used another hyperparameter, ξ, to control the overall magnitude of all β's. This way, σ_l controls the relevance of covariate x_l compared to other covariates, and ξ controls the overall usefulness of all covariates in separating all classes. The standard deviation of β_{jl} is therefore equal to $\xi\sigma_l$.

The above scheme was also used for the dpMNL model. Note that in this model, one σ_l controls all β_{jlc}, where $j = 1,...,J$ indexes classes, and $c = 1,...,C$ indexes the unique components in the mixture. Therefore, the standard deviation of β_{jlc} is $\xi\sigma_l\nu_c$. Here, ν_c is specific to each component c, and controls the overall effect of coefficients in that component. That is, while σ and ξ are global hyperparameters common between all components, ν_c is a local hyperparameter within a component. Similarly, the standard deviation of intercepts, α_{jc} in component c is $\eta\tau_c$. We used $N(0,1)$ as the prior for ν_c and τ_c.

We also needed to specify priors for μ_l and σ_l, the mean and standard deviation of covariate x_l, where $l = 1,..., p$. For these parameters, we used the following priors:

$$\begin{aligned}
\mu_{lc}|\mu_{0,l},\sigma_{0,l} &\sim N(\mu_{0,l},\sigma_{0,l}^2),\\
\mu_{0,l} &\sim N(0,5^2),\\
\log(\sigma_{0,l}^2) &\sim N(0,2^2),\\
\log(\sigma_{lc}^2)|M_{\sigma,l},V_{\sigma,l} &\sim N(M_{\sigma,l},V_{\sigma,l}^2),\\
M_{\sigma,l} &\sim N(0,1^2),\\
\log(V_{\sigma,l}^2) &\sim N(0,2^2).
\end{aligned}$$

These priors make use of higher level hyperparameters to provide flexibility. For example, if the components are not different with respect to covariate x_l, the corresponding variance, $\sigma_{0,l}^2$, becomes small, forcing μ_{lc} close to their overall mean, $\mu_{0,l}$.

The MCMC chains for MNL, qMNL, and dpMNL ran for 10000 iterations. Simulating the Markov chain took about 2.1 seconds per iteration (5.8 hours in total) for dpMNL and 0.5 seconds per iteration (1.4 hours in total) for MNL using a MATLAB implementation on an UltraSPARC III machine. Training the RBF-SVM model took about 2.5 minutes.

Model	Accuracy (%)	F_1 (%)
MNL	50.0	41.2
qMNL	50.5	42.1
SVM (Ding and Dubchak, 2001)	49.4	-
LSVM	50.5	47.3
RBF-SVM	53.1	49.5
dpMNL	58.6	53.0

Table 3: Performance of models based on protein fold classification data.

The results for MNL, qMNL, LSVM, RBF-SVM, and dpMNL are presented in Table 3, along with the results for the best SVM model developed by Ding and Dubchak (2001) on the exact same data set. As we can see, the nonlinear RBF-SVM model that we fit has better accuracy than the linear models. Our dpMNL model provides an additional improvement over the RBF-SVM model. This shows that there is in fact a nonlinear relationship between folding classes and the composition of amino acids, and our nonlinear model could successfully identify this relationship.

4.2 Detecting Parkinson's Disease

The above example shows that our method can potentially improve prediction accuracy, though of course other classifiers, such as SVM and neural networks, may do better on some problems. However, we believe the application of our method is not limited to simply improving prediction accuracy—it can also be used to discover hidden structure in data by identifying subgroups (i.e., mixture components) in the population. This section provides an example to illustrate this concept.

Neurological disorders such as Parkinson's disease (PD) have profound consequences for patients, their families, and society. Although there is no cure for PD at this time, it is possible to alleviate its symptoms significantly, especially at the early stages of the disease (Singh et al., 2007). Since approximately 90% of patients exhibit some form of vocal impairment (Ho et al., 1998), and research has shown that vocal impairment could be one of the earliest indicators of onset of the illness (Duffy, 2005), voice measurement has been proposed as a reliable tool to detect and monitor PD (Sapir et al., 2007; Rahn et al., 2007; Little et al., 2008). For example, patients with PD commonly display a symptom known as *dysphonia*, which is an impairment in the normal production of vocal sounds.

In a recent paper, Little et al. (2008) show that by detecting *dysphonia*, we could identify patients with PD. Their study used data on sustained vowel phonations from 31 subjects, of whom 23 were diagnosed with PD. The 22 covariates used include traditional variables, such as measures of vocal fundamental frequency and measures of variation in amplitude of signals, as well as a novel measurement referred to as *pitch period entropy* (PPE). See Little et al. (2008) for a detailed description of these variables. This data set is publicly available at UCI Machine Learning Repository (`http://archive.ics.uci.edu/ml/datasets/Parkinsons`).

Little et al. (2008) use an SVM classifier with Gaussian radial basis kernel functions to identify patients with PD, chosing the SVM penalty value and kernel bandwidth by an exhaustive search over a range of values. They also perform an exhaustive search to select the optimal subset of features (10 features were selected). Their best model provides a 91.4% ($\pm$4.4) accuracy rate based on a bootstrap algorithm. This of course does not reflect the true prediction accuracy rate of the model for future observations since the model is trained and evaluated on the same sample. Here, we use

Model	Accuracy (%)	F_1 (%)
MNL	85.6 (2.2)	79.1 (2.8)
qMNL	86.1 (1.5)	79.7 (2.1)
LSVM	87.2 (2.3)	80.6 (2.8)
RBF-SVM	87.2 (2.7)	79.9 (3.2)
dpMNL	87.7 (3.3)	82.6 (2.5)

Table 4: Performance of models based on detecting Parkinson's disease. Standard errors of estimates (based on 5 cross-validation folds) are provided in parentheses.

Group	Frequency	Age Average	Male Proportion
1	107	66 (0.7)	0.86 (0.03)
2	12	72 (1.1)	0.83 (0.11)
3	36	63 (1.8)	0.08 (0.04)
4	40	65 (2.2)	0.40 (0.08)
Population	195	66 (0.7)	0.60 (0.03)

Table 5: The age average and male proportion for each cluster (i.e., mixture component) identified by our model. Standard errors of estimates are provided in parentheses.

a 5-fold cross validation scheme instead in order to obtain a more accurate estimate of prediction accuracy rate and avoid inflating model performance due to overfitting. As a result, our models cannot be directly compared to that of Little et al. (2008).

We apply our dpMNL model to the same data set, along with MNL and qMNL (no interactions, only x_i and x_i^2 as covariates). Although the observations from the same subject are not independent, we assume they are, as done by Little et al. (2008). Instead of selecting an optimum subset of features, we used PCA and chose the first 10 principal components. The MCMC algorithm for MNL, qMNL, and dpMNL ran for 3000 iterations (the first 500 iterations were discarded). Simulating the Markov chain took about 0.7 second per iteration (35 minutes per data set) for dpMNL and 0.1 second per iteration (5 minutes per data set) for MNL using a MATLAB implementation on an UltraSPARC III machine. Training the RBF-SVM model took 38 seconds for each data set.

Using the dpMNL model, the most probable number of components in the posterior is four (note that might change from one iteration to another). Table 4 shows the average and standard errors (based on 5-fold cross validation) of the accuracy rate and the F_1 measure for MNL, LSVM, RBF-SVM, and dpMNL. (But note that the standard errors assume independence of cross-validation folds, which is not really correct.)

While dpMNL provides slightly better results, the improvement is not statistically significant. However, examining the clusters (i.e., mixture components) identified by dpMNL reveals some information about the underlying structure in the data. Table 5 shows the average age of subjects and male proportion for the four clusters (based on the most probable number of components in the posterior) identified by our model. Note that age and gender are not available from the UCI Machine Learning Repository, and they are not included in our model. They are, however, available from Table 1 in Little et al. (2008). The first two groups include substantially higher percentages of male subjects than female subjects. The average age in the second of these groups is higher compared to the first group. Most of the subjects in the third group are female (only 8% are male).

The fourth group also includes more female subjects than male subjects, but the disproportionality is not as high as for the third group.

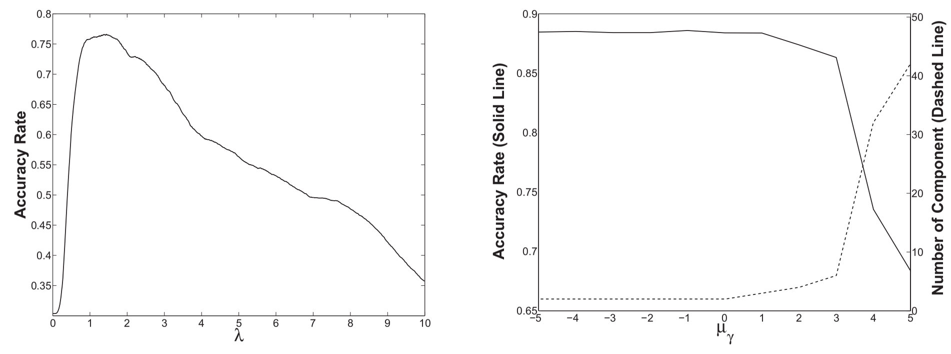
When identifying Parkinson's disease by detecting dysphonia, it has been shown that gender has a confounding effect (Cnockaert et al., 2008). By grouping the data into clusters, our model has identified (to some extent) the heterogeneity of subjects due to age and gender, even though these covariates were not available to the model. Moreover, by fitting a separate linear model to each component (i.e., conditioning on mixture identifiers), our model approximates the confounding effect of age and gender. For this example, we could have simply taken the age and gender of subjects from Table 1 in Little et al. (2008) and included them in our model. In many situations, however, not all the relevant factors are measured. This could result in unobservable changes in the structure of data. We discuss this concept in more detail elsewhere (Shahbaba, 2009).

5. Discussion and Future Directions

We introduced a new nonlinear classification model, which uses Dirichlet process mixtures to model the joint distribution of the response variable, y, and the covariates, x, non-parametrically. We compared our model to several linear and nonlinear alternative methods using both simulated and real data. We found that when the relationship between y and x is nonlinear, our approach provides substantial improvement over alternative methods. One advantage of this approach is that if the relationship is in fact linear, the model can easily reduce to a linear model by using only one component in the mixture. This way, it avoids overfitting, which is a common challenge in many nonlinear models.

We believe our model can provide more interpretable results. In many real problems, the identified components may correspond to a meaningful segmentation of data. Since the relationship between y and x remains linear in each segment, the results of our model can be expressed as a set of linear patterns for different data segments.

Hyperparameters such as λ in RBF-SVM and γ in dpMNL can substantially influence the performance of the models. Therefore, it is essential to choose these parameters appropriately. For RBF-SVM, we optimized λ using a validation set that includes 20% of the training data. Figure 4 (a) shows the effect of λ on prediction accuracy for one data set. The value of λ with the highest accuracy rate based on the validation set was used to train the RBF-SVM model. The hyperparameters in our dpMNL model are not fixed at some "optimum" values. Instead, we use hyperpriors that reflect our opinion regarding the possible values of these parameters before observing the data, with the posterior for these parameters reflecting both this prior opinion and the data. Hyperpriors for regression parameters, β, facilitate their shrinkage towards zero if they are not relevant to the classification task. The hyperprior for the scale parameter γ affects how many mixture components are present in the data. Instead of setting γ to some constant number, we allow the model to decide the appropriate value of γ, using a hyperprior that encourages a small number of components, but which is not very restrictive, and hence allows γ to become large in the posterior if required to fit the data. Choosing unreasonably restrictive priors could have a negative effect on model performance and MCMC convergence. Figure 4 (b) illustrates the negative effect of unreasonable priors for γ. For this data set, the correct number of components is two. We gradually increase μ_γ, where $\log(\gamma) \sim N(\mu_\gamma, 2)$, in order to put higher probability on larger values of γ and lower probability on smaller values. As we can see, setting $\mu_\gamma \geq 4$, which makes the hyperprior very restrictive, results in



(a) (b)

Figure 4: Effects of scale parameters when fitting a data set generated according to Simulation 1. (a) The effect of λ in the RBF-SVM model. (b) The effect of the prior on the scale parameter, $\gamma \sim$ log-N$(\mu_\gamma, 2^2)$, as μ_γ changes in the dpMNL model.

a substantial decline in accuracy rate (solid line) due to overfitting with a large number of mixture components (dashed line).

The computational cost for our model is substantially higher compared to other methods such as MNL and SVM. This could be a preventive factor in applying our model to some problems. The computational cost of our model could be reduced by using more efficient methods, such as the "split-merge" approach introduced by Jain and Neal (2007). This method uses a Metropolis-Hastings procedure that resamples clusters of observations simultaneously rather than incrementally assigning one observation at a time to mixture components. Alternatively, it might be possible to reduce the computational cost by using a variational inference algorithm similar to the one proposed by Blei and Jordan (2005). In this approach, the posterior distribution P is approximated by a tractable variational distribution Q, whose free variational parameters are adjusted until a reasonable approximation to P is achieved.

We expect our model to outperform other nonlinear models such as neural networks and SVM (with nonlinear kernel functions) when the population is comprised of subgroups each with their own distinct pattern of relationship between covariates and response variable. We also believe that our model could perform well if the true function relating covariates to response variable contains sharp changes.

The performance of our model could be negatively affected if the covariates are highly correlated with each other. In such situations, the assumption of diagonal covariance matrix for x adopted by our model could be very restrictive. To capture the interdependencies between covariates, our model would attempt to increase the number of mixture components (i.e., clusters). This however is not very efficient. To address this issue, we could use mixtures of factor analyzers, where the covariance structure of high dimensional data is model using a small number of latent variables (see for example, Rubin and Thayer, 2007; Ghahramani and Hinton, 1996).

In this paper, we considered only continuous covariates. Our approach can be easily extended to situations where the covariate are categorical. For these problems, we need to replace the normal distribution in the baseline, G_0, with a more appropriate distribution. For example, when the covariate x is binary, we can assume $x \sim Bernoulli(\mu)$, and specify an appropriate prior distribution (e.g., *Beta* distribution) for μ. Alternatively, we can use a continuous latent variable, z, such that $\mu = \exp(z)/\{1+\exp(z)\}$. This way, we can still model the distribution of z as a mixture of normals. For categorical covariates, we can either use a Dirichlet prior for the probabilities of the K categories, or use K continuous latent variables, $z_1,...,z_K$, and let the probability of category j be $\exp(z_j)/\sum_{j'}^{K}\exp(z_{j'})$.

Throughout this paper, we assumed that the relationship between y and x is linear within each component of the mixture. It is possible of course to relax this assumption in order to obtain more flexibility. For example, we can include some nonlinear transformation of the original variables (e.g., quadratic terms) in the model.

Our model can also be extended to problems where the response variable is not multinomial. For example, we can use this approach for regression problems with continuous response, y, which could be assumed normal within a component. We would model the mean of this normal distribution as a linear function of covariates for cases that belong to that component. Other types of response variables (i.e., with Poisson distribution) can be handled in a similar way.

In the protein fold prediction problem discussed in this paper, classes were regarded as a set of unrelated entities. However, these classes are not completely unrelated, and can be grouped into four major structural classes known as α, β, α/β, and $\alpha+\beta$. Ding and Dubchak (2001) show the corresponding hierarchical scheme (Table 1 in their paper). We have previously introduced a new approach for modeling hierarchical classes (Shahbaba and Neal, 2006, 2007). In this approach, we use a Bayesian form of the multinomial logit model, called corMNL, with a prior that introduces correlations between the parameters for classes that are nearby in the hierarchy. Our dpMNL model can be extend to classification problems where classes have a hierarchical structure (Shahbaba, 2007). For this purposse, we use a corMNL model, instead of MNL, to capture the relationship between the covariates, x, and the response variable, y, within each component. The results is a nonlinear model which takes the hierarchical structure of classes into account.

Finally, our approach provides a convenient framework for semi-supervised learning, in which both labeled and unlabeled data are used in the learning process. In our approach, unlabeled data can contribute to modeling the distribution of covariates, x, while only labeled data are used to identify the dependence between y and x. This is a quite useful approach for problems where the response variable is known for a limited number of cases, but a large amount of unlabeled data can be generated. One such problem is classification of web documents.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada. Radford Neal holds a Canada Research Chair in Statistics and Machine Learning.

References

E. L. Allwein, R. E. Schapire, Y. Singer, and P. Kaelbling. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

C. E. Antoniak. Mixture of Dirichlet process with applications to Bayesian nonparametric problems. *Annals of Statistics*, 273(5281):1152–1174, 1974.

D. Blackwell and J. B. MacQueen. Ferguson distributions via polya urn scheme. *Annals of Statistics*, 1:353–355, 1973.

D. M. Blei and M. I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1:121–144, 2005.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall, Boca Raton, 1993.

C. A. Bush and S. N. MacEachern. A semi-parametric Bayesian model for randomized block designs. *Biometrika*, 83:275–286, 1996.

B. Cai and D. B. Dunson. Bayesian covariance selection in generalized linear mixed models. *Biometrics*, 62:446–457, 2006.

L. Cnockaert, J. Schoentgen, P. Auzou, C. Ozsancak, L. Defebvre, and F. Grenez. Low-frequency vocal modulations in vowels produced by parkinsonian subjects. *Speech Communication*, 50(4): 288–300, 2008.

M. J. Daniels and R. E. Kass. Nonconjugate Bayesian estimation of covariance matrices and its use in hierarchical models. *Journal of the American Statistical Association*, 94(448):1254–1263, 1999.

C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognision using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

J. R. Duffy. *Motor Speech Disorders: Substrates, Differential Diagnosis and Management*. Elsevier Mosby, St. Louis, Mo., 2nd edition, 2005.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of American Statistical Society*, 90:577–588, 1995.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1: 209–230, 1973.

T. S. Ferguson. *Recent Advances in Statistics, Ed: Rizvi, H. and Rustagi, J.*, chapter Bayesian density estimation by mixtures of normal distributions, pages 287–302. Academic Press, New York, 1983.

J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002. ISSN 1533-7928.

Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto, 1996.

A. K. Ho, R. Iansek, C. Marigliani, J. L. Bradshaw, and S. Gates. Speech impairment in a large sample of patients with Parkinson's disease. *Behavioural Neurology*, 11:131–137, 1998.

U. Hobohm and C. Sander. Enlarged representative set of proteins. *Protein Science*, 3:522–524, 1994.

U. Hobohm, M. Scharf, R. Schneider, and C. Sander. Selection of a representative set of structure from the brookhaven protein bank. *Protein Science*, 1:409–417, 1992.

C. W. Hsu and C. J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

S. Jain and R. M. Neal. Splitting and merging components of a nonconjugate Dirichlet process mixture model (with discussion). *Bayesian Analysis*, 2:445–472, 2007.

M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Transactions on Biomedical Engineering*, (in press), 2008.

L. Lo Conte, B. Ailey, T.J.P Hubbard, S. E. Brenner, A.G. Murzing, and C. Chothia. Scop: a structural classification of protein database. *Nucleic Acids Research*, 28:257–259, 2000.

S. N. MacEachern and P. Müller. Estimating mixture of Dirichlet process models. *Journal of Computational and Graphcal Statistics*, 7:223–238, 1998.

E. Meeds and S. Osindero. An alternative infinite mixture of Gaussian process experts. *Advances in Neural Information Processing Systems*, 18:883, 2006.

P. Müller, A. Erkanli, and M. West. Bayesian curve fitting using multivariate mormal mixtures. *Biometrika*, 83(1):67–79, 1996.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.

R. M. Neal. Slice sampling. *Annals of Statistics*, 31(3):705–767, 2003.

R. M. Neal. *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

R. M. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics No. 118, New York: Springer-Verlag, 1996.

D. A. Rahn, M. Chou, J. J. Jiang, and Y. Zhang. Phonatory impairment in Parkinson's disease: evidence from nonlinear dynamic analysis and perturbation analysis.(Clinical report). *Journal of Voice*, 21:64–71, 2007.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems 14*, page 881. MIT Press, 2002.

D. Rubin and D. Thayer. EM algorithms for ML factor analysis. *Pshychometrika*, 47(1):69–76, 2007.

S. Sapir, J. L. Spielman, L. O. Ramig, B. H. Story, and C. Fox. Effects of intensive voice treatment (the Lee Silverman Voice Treatment [lsvt]) on vowel articulation in dysarthric individuals with idiopathic Parkinson disease: Acoustic and perceptual findings. *Journal of Speech, Language, and Hearing Research*, 50(4):899–912, 2007.

B. Shahbaba. Discovering hidden structures using mixture models: Application to nonlinear time series processes. *Studies in Nonlinear Dynamics & Econometrics*, 13(2):Article 5, 2009.

B. Shahbaba. *Improving Classification Models When a Class Hierarchy is Available*. PhD thesis, Biostatistics, Public Health Sciences, University of Toronto, 2007.

B. Shahbaba and R. M. Neal. Improving classification when a class hierarchy is available using a hierarchy-based prior. *Bayesian Analysis*, 2(1):221–238, 2007.

B. Shahbaba and R. M. Neal. Gene function classification using Bayesian models with hierarchy-based priors. *BMC Bioinformatics*, 7:448, 2006.

N. Singh, V. Pillay, and Y. E. Choonara. Advances in the treatment of Parkinson's disease. *Progress in Neurobiology*, 81:29–44, 2007.

I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 258–265, Washington, DC, USA, 2005. IEEE Computer Society.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

S. Waterhouse, D. MacKay, and T. Robinson. Bayesian methods for mixtures of experts. In *Advances in Neural Information Processing Systems 8*, page 351. MIT Press, 1996.