Technical Report No. 0707, Department of Statistics, University of Toronto

Nonlinear Models Using Dirichlet Process Mixtures

Babak Shahbaba
Dept. of Public Health Sciences,
Biostatistics
University of Toronto,
Toronto, Ontario, Canada
babak@stat.utoronto.ca

Radford M. Neal
Dept. of Statistics and Dept. of Computer Science,
University of Toronto,
Toronto, Ontario, Canada
radford@stat.utoronto.ca

9 March 2007

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. We use simulated data to compare the performance of this new approach to a simple multinomial logit (MNL) model, an MNL model with quadratic terms, and a decision tree model. We also evaluate our approach on a protein fold classification problem, and find that our model provides substantial improvement over previous methods, which were based on Neural Networks (NN) and Support Vector Machines (SVM). Folding classes of protein have a hierarchical structure. We extend our method to classification problems where a class hierarchy is available. We find that using the prior information regarding the hierarchical structure of protein folds can result in higher predictive accuracy.

1 Introduction

In regression and classification models, estimation of parameters and interpretation of results are easier if we assume a simple distributional form (e.g., normality) and regard the relationship between response variable and covariates as linear. However, the performance of the model obtained 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 to capture 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).

We will describe the Dirichlet process mixture model as a limit of finite mixture model (see Neal (2000) for further description). Suppose $y_1, ..., y_n$ are drawn independently from some unknown distribution. We can model the distribution of y as a mixture of simple distributions such that:

$$P(y) = \sum_{c=1}^{C} p_c f(y|\phi_c)$$

Here, p_c are the mixing proportions, and f is a simple class of distributions, such as normal with $\phi = (\mu, \sigma)$. 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:

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

where $p_c \geq 0$ and $\sum p_c = 1$. Parameters ϕ_c are assumed to be independent under the prior with distribution G_0. We can use mixture identifiers, c_i, and represent the above mixture model as follows (Neal, 2000):

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

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

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

Here, n_{ic} represents the number of data points previously (i.e., before the i^{th}) assigned to component c. The probability of assigning each component to the first data point is $1/C$. As we proceed, this probability becomes higher for components with larger numbers of samples (i.e., larger n_{ic}).

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

$$\begin{aligned} P(c_i = c|c_1, ..., c_{i-1}) &\rightarrow \frac{n_{ic}}{i-1+\gamma} \\ P(c_i \neq c_j \forall j < i|c_1, ..., c_{i-1}) &\rightarrow \frac{\gamma}{i-1+\gamma} \end{aligned} \tag{3}$$

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

$$\theta_i|\theta_1, ..., \theta_{i-1} \sim \frac{1}{i-1+\gamma} \sum_{j<i} \delta(\theta_j) + \frac{\gamma}{i-1+\gamma} G_0 \tag{4}$$

where $\delta(\theta)$ is a point mass distribution at θ. This is equivalent to the conditional probabilities implied by the Dirichlet process mixture model, 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, the limit of the finite mixture model (1) is equivalent to the Dirichlet process mixture model (5) as the number of components goes to infinity. G is the distribution over θ's, and has a Dirichlet process prior, $\mathcal{D}$. 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 in the mixture, such that a larger γ results in a larger number of components. Phrased this way, each data point, i, has its own parameters, θ_i, drawn from a distribution that is drawn from a Dirichlet process prior. But since distributions drawn from a Dirichlet process are discrete (almost surely), the θ_i for different data points may be the same.

Bush and MacEachern (1996), Escobar and West (1995), MacEachern and Müller (1998), and Neal (2000) have used this method for density estimation. Müller *et al.* (1996) used Dirichlet process mixtures 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 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 non-parametrically estimates 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. 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 of each component.

Our focus in this paper is on classification models with a multi-category response. We also show how our method can be extended to classification problems where classes have a hierarchical structure, and to problems with multiple sources of information. The next section describes our methodology. In Section 3, we illustrate our approach and evaluate its performance based 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. Folding classes of protein have a hierarchical structure. In Section 5, we extend our approach to classification problems of this sort where a class hierarchy is available, and evaluate the performance of this new model on the protein fold recognition dataset. Section 6 shows how this approach can be used for multiple sources of information. Finally, Section 7 is devoted to discussion, future directions and limitations of the proposed method.

2 Methodology

Consider a classification problem with continuous covariates, $x = (x_1, ..., x_p)$, and a categorical response variable, y, with J classes. To model the relationship between y and x, we model the joint distribution of y and x non-parametrically using Dirichlet process mixtures. Within each component of the mixture, the relationship between y and x is assumed to be linear. 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. Our model has the following form:

$$\begin{aligned} y_i, x_{i1}, ..., x_{ip} | \theta_i &\sim F(\theta_i) \\ \theta_i | G &\sim G \\ G &\sim \mathcal{D}(G_0, \gamma) \end{aligned}$$

where $i = 1, ..., n$ indexes the observations, and $l = 1, ..., p$ indexes the covariates. In our model, $\theta = (\mu, \sigma, \alpha, \boldsymbol{\beta})$, and the component distributions, $F(\theta)$, are defined based on $P(y, x) = P(x)P(y|x)$ as follows:

$$\begin{aligned} x_{il} &\sim N(\mu_l, \sigma_l^2) \\ P(y_i = j | x_i, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \frac{\exp(\alpha_j + x_i \boldsymbol{\beta}_j)}{\sum_{j'=1}^{J} \exp(\alpha_{j'} + x_i \boldsymbol{\beta}_{j'})} \end{aligned}$$

Here, the parameters $\mu = (\mu_1, ..., \mu_p)$ and $\sigma = (\sigma_1, ..., \sigma_p)$ are the means and standard deviations of covariates in each component. The component index, c, is omitted for simplicity. Within a component, $\alpha = (\alpha_1, ..., \alpha_J)$, and $\boldsymbol{\beta} = (\boldsymbol{\beta_1, ..., \beta_J})$ are the parameters of the multinomial logit (MNL) model, and J is the number of classes. The entire set of regression coefficients, $\boldsymbol{\beta}$, can be presented as a $p \times J$ matrix. This representation is redundant, since one of the $\boldsymbol{\beta}_j$'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.

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). 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 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. Samples simulated from the posterior distribution are used to estimate posterior predictive probabilities. We predict the response values for new cases based on these probabilities. For a new case with covariates x', the posterior predictive probability of response variable, y', is estimated as follows:

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

where

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

Here, S is the number of post-convergence samples from MCMC, and $\theta^{(s)}$ represents the set of parameters obtained at iteration s.

Neal (2000) presented several possible algorithms for sampling from the posterior distribution of Dirichlet process mixtures. In this research, we use Gibbs sampling with auxiliary parameters

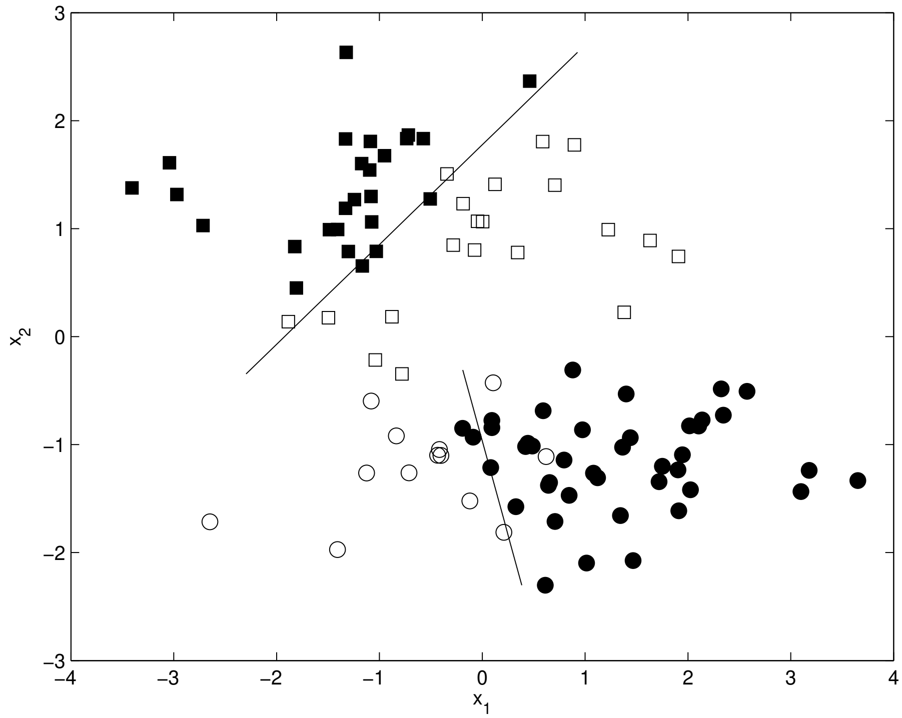

Figure 1: An illustration of our model for a binary (black and white) classification problem with two covariates. Here, the mixture has two components, which are shown with circles and squares. In each component, an MNL model separates the two classes into "black" or "white" with a linear decision boundary.

(Neal's algorithm 8). This approach is similar to the algorithm proposed by MacEachern and Müller (1998), with a difference that the auxiliary parameters exist only temporarily. To improve the MCMC sampling, after each update using auxiliary variables, we update the component parameters using their corresponding data points. For a complete description of this method, see the paper by Neal (2000). All our models are coded in MATLAB and are available online at `http://www.utstat.utoronto.ca/~babak`.

In Figure 1, we show a state from an MCMC simulation for our model in which there are two covariates and the response variable is binary. In this iteration, our model has identified two components (circles and squares). Within a component, two classes (stars and crosses) are separated using an MNL model. Note, the decision boundaries shown are component specific. The overall decision boundary, which is a smooth function, is not shown in this figure. In our approach, division of the data into components and fitting of MNL models are performed simultaneously.

3 Results for synthetic data

In this section, we illustrate our approach, henceforth called dpMNL, using synthetic data. We compare our model to a simple MNL model, an MNL model with quadratic terms (i.e., $x_l x_k$, where

$l = 1, ..., p$ and $k = 1, ..., p$), referred to as qMNL, and a decision tree model (Breiman *et al.*, 1993) that uses 10-fold cross-validation for pruning. For the simple MNL model, we use both Bayesian and maximum likelihood estimation. 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 and is 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.

For the first test, we compare the models using a synthetic four-way classification problem with 5 covariates. Data are generated according to our model with G_0 being the following prior:

$$\begin{aligned} \mu_l &\sim N(0, 1) \\ \log(\sigma_l^2) &\sim N(0, 2^2) \\ \log(\tau^2) &\sim N(0, 0.1^2) \\ \log(\nu^2) &\sim N(0, 2^2) \end{aligned}$$

Note that $\alpha_j|\tau \sim N(0, \tau^2)$, and $\beta_{jl}|\nu \sim N(0, \nu^2)$, where $l = 1, ..., 5$ and $j = 1, ..., 4$. From the above baseline prior, we sample two components, θ_1 and θ_2, where $\theta = (\mu, \sigma, \eta, \nu, \alpha, \boldsymbol{\beta})$. For each θ, we generate 5000 data points by first drawing $x_{il} \sim N(\mu_l, \sigma_l)$ and then sampling y using the following MNL model:

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

The overall sample size is 10000. We randomly split the data to the training set, with 100 data points, and 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:

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

To fit the decision tree models (Breiman *et al.*, 1993), we used the available functions in MATLAB. These functions are treefit, treetest (for cross-validation) and treeprune.

Model	Accuracy (%)	F_1 (%)
Baseline	45.57	15.48
MNL (Maximum Likelihood)	77.30	66.65
MNL	78.39	66.52
qMNL	83.60	74.16
Tree (Cross Validation)	70.87	55.82
dpMNL	**89.21**	**81.00**

Table 1: Simulation 1: the average performance of models based on 50 simulated datasets. The Baseline model assigns test cases to the class with the highest frequency in the training set.

The above procedure was repeated 50 times. Each time, new θ_1 and θ_2 were sampled from the prior, and a new dataset was created based on these θ's. We used Hamiltonian dynamics (Neal, 1993) for updating the regression parameters, α's and $\boldsymbol{\beta}$'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, where $\log(\gamma) \sim N(-3, 2^2)$. This prior encourages smaller values of γ, which results in 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.

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 based accuracy rates) using a paired t-test with $n = 50$.

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 $i = 1, ..., 10000$) from the $Uniform(0, 5)$ distribution, and generated a binary response variable, y_i, according the following model:

$$P(y = 1|x) \quad = \quad \frac{1}{1 + \exp[a_1 \sin(x_1^{1.04} + 1.2) + x_1 \cos(a_2 x_2 + 0.7) + a_3 x_3 - 2]}$$

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 2 shows a random sample from this model for $a_3 = 0$. In this figure, the dotted line is the optimal decision boundary.

We generated 50 datasets ($n = 10000$) using the above model. Each time, we sampled new covariates, x, new constant values, a_1, a_2, a_3, and new response variable, y. As before, models were trained on 100 data points, and tested on the remaining samples. The average results over 50 datasets are presented in Table 2. As before, the dpMNL model provides significantly (all p-values are smaller than 0.001) better performance compared to all other models. This time, however, the performance of the qMNL model is closer to the results from the dpMNL model.

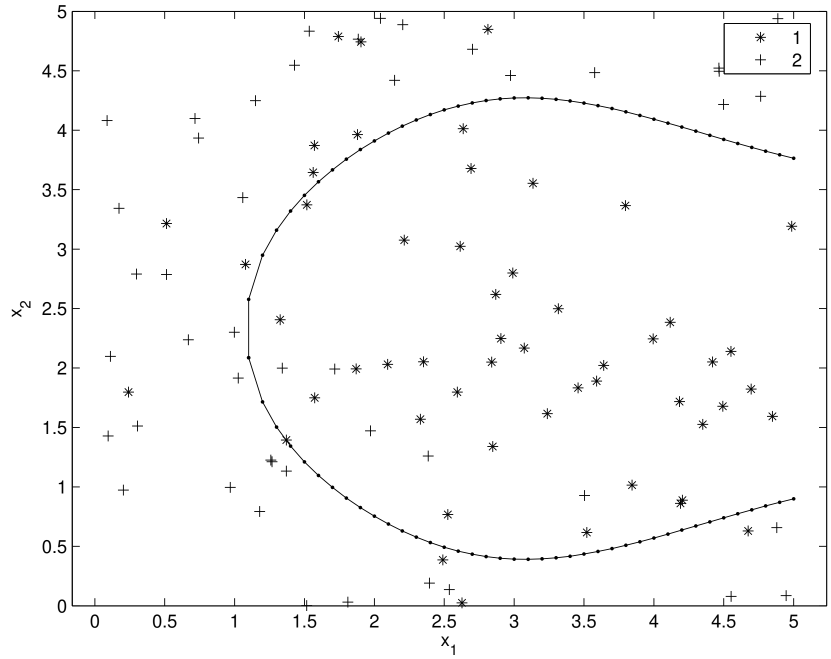

Figure 2: A random sample generated according to Simulation 2 with $a_3 = 0$. The dotted line is the optimal boundary function.

Model	Accuracy (%)	F_1 (%)
Baseline	61.96	37.99
MNL (Maximum Likelihood)	73.58	68.33
MNL	73.58	67.92
qMNL	75.60	70.12
Tree (Cross Validation)	73.47	66.94
dpMNL	**77.80**	**73.13**

Table 2: Simulation 2: the average performance of models based on 50 simulated datasets. The Baseline model assigns test cases to the class with the highest frequency in the training set.

4 Results for protein fold classification

In this section, we consider the problem of predicting a protein's 3D structure (i.e., folding class) based on its sequence. For this problem, 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 the 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 dataset provided by Ding and Dubchak (2001). The proteins in this dataset 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 dataset included 128 unique folds. However, Ding and Dubchak (2001) selected only 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 datasets are available online at `http://crd.lbl.gov/~cding/protein/`. These datasets include the length of protein sequences, and 20 other covariates based on the percentage composition of different amino acids. For a detail description of data, see Dubchak *et al.* (1995).

Ding and Dubchak (2001) trained several Support Vector Machines (SVM) with nonlinear kernel functions, and Neural Networks (NN) with different architecture on this dataset. They also tried different classification schemes, namely, one versus others (OvO), unique one versus others (uOvO), and all versus all (AvA). The details for these methods can be found in their paper. The performance of these models on the test set is presented in Table 3.

We first centered the covariates so they have mean 0. We trained our MNL and dpMNL on the training set, and evaluated their performance on the test set. For these models, we used similar priors as the ones used in the previous section. However, the hyperparameters for the variances of regression parameters are more elaborate. We used the following priors for the MNL model:

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

Here, 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 is called Automatic Relevance Determination (ARD), and was suggested 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 $\xi\sigma_l$.

We used the same scheme for the MNL models in dpMNL. 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,

Model	Accuracy (%)	F_1 (%)
NN-OvO	20.5	-
SVM-OvO	43.5	-
SVM-uOvO	49.4	-
SVM-AvA	44.9	-
MNL	50.0	41.2
dpMNL	**58.6**	**53.0**

Table 3: Performance of models based on protein fold classification data. NN and SVM use maximum likelihood estimation, and are developed by Ding and Dubchak (2001).

where $l = 1, ..., p$. For these parameters, we used the following priors:

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

As we can see, the priors depend on higher level hyperparameters. This provides a more flexible scheme. If, for example, the components are not different with respect to covariate x_l, the corresponding variance, $\sigma^2_{0,l}$, becomes small, forcing μ_{lc} close to their overall mean, $\mu_{0,l}$.

For each of our Bayesian models discussed in this section (and also in the following sections), we performed four simultaneous MCMC simulations each of size 10000. The chains have different starting values. We discarded the first 1000 samples from each chain and used the remaining samples for predictions. For this problem, running multiple chains results in faster and more efficient sampling. Simulating the Markov chain for 10 iterations took about half a minute for MNL, and about 3 minutes for dpMNL, using a MATLAB implementation on an UltraSPARC III machine.

The results for MNL and dpMNL models are presented in Table 3. As a benchmark, we also present the results for the SVM and NN models developed by Ding and Dubchak (2001) on the exact same dataset. As we can see, our linear MNL model provides better accuracy rate compared to the SVM and NN models developed by Ding and Dubchak (2001). Our dpMNL model provides an additional improvement over the MNL 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.

It is worth noting the performance of the NN models is influenced by many design choices, and by model assumptions. We found that Bayesian neural networks model (Neal, 1996) had better performance than the NN model of Ding and Dubchak (2001). Our NN model performs very similarly to the performance of the dpMNL model.

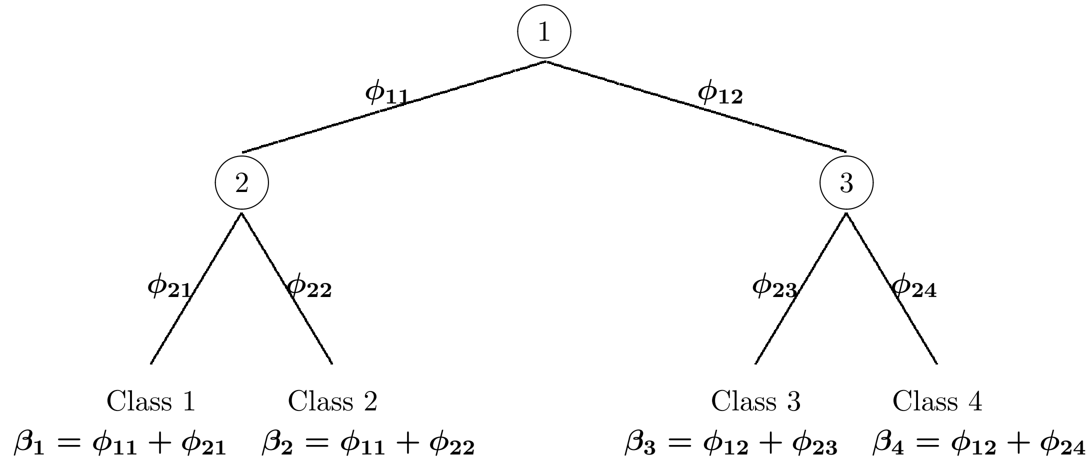

Figure 3: A simple representation of our hierarchical classification model.

5 Extension to hierarchical classes

In the previous section, we modeled the folding classes 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, with a prior that introduces correlations between the parameters for classes that are nearby in the hierarchy.

Figure 3 illustrates this approach using a simple hierarchical structure. For each branch in the hierarchy, we define a different set of parameters, ϕ. Our model classifies objects to one of the end nodes using an MNL model whose regression coefficients for class j are represented by the sum of the parameters for all the branches leading to that class. Sharing of common parameters (from common branches) introduces prior correlations between the parameters of nearby classes in the hierarchy. We refer to this model as corMNL.

In this section, we extend our nonlinear model to classification problems where classes have a hierarchical structure. For this purpose, 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. We refer to this models as dpCorMNL.

Table 4 presents the results for the two linear models (with and without hierarchy-base priors), and two nonlinear models (with and without hierarchy-based priors). In this table, "parent accuracy" refers to the accuracy of models based on the four major structural classes, namely α, β, α/β. When comparing the hierarchical models to their non-hierarchical counterparts, the advantage of using the hierarchy is apparent only for some measures (i.e., parent accuracy rate for corMNL, and the F_1 measure for dpCorMNL). As we can see, however, the dpCorMNL model provides a substantial improvement over corMNL.

Model	Accuracy (%)	Parent accuracy (%)	F_1 (%)
MNL	50.0	76.5	41.2
corMNL	49.5	77.9	41.4
dpMNL	58.6	**79.9**	53.0
dpCorMNL	**59.1**	79.4	**55.2**

Table 4: Comparison of hierarchical models (linear and nonlinear) with non-hierarchical models (linear and nonlinear) based on protein fold classification data.

Model	Accuracy (%)	Parent accuracy (%)	F_1 (%)
NN-OvO	41.4	-	-
SVM-OvO	43.2	-	-
SVM-uOvO	49.4	-	-
SVM-AvA	56.5	-	-
MNL	56.5	80.4	51.4
corMNL	59.6	83.3	54.6
dpMNL	60.4	82.0	55.9
dpCorMNL	**61.4**	**83.8**	**57.8**

Table 5: Comparison of hierarchical models (linear and nonlinear) with non-hierarchical models (linear and nonlinear) based on protein fold classification data. The covariates are obtained from four different feature sets: composition of amino acids, predicted secondary structure, hydrophobicity, and normalized van der Waals volume.

6 Extension to multiple datasets

In order to improve the prediction of folding classes for proteins, Ding and Dubchak (2001) combined the feature set based on amino acid compositions with 5 other feature sets, which were independently extracted based on various physico-chemical and structural properties of amino acids in the sequence. The additional features predicted secondary structure, hydrophobicity, normalized varn der Waals volume, polarity, and polarizability. Each data source has 21 covariates. For a detailed description of these features, see Dubchak *et al.* (1995). Ding and Dubchak (2001) added the above 5 datasets sequentially to the amino acid composition dataset. For prediction, they used a majority voting system, in which the votes obtained from models based on different features sets are combined, and the class with the most votes is regarded as predicted fold. Their results show that adding additional feature sets can improve the performance in some cases and can result in lower performance in some other cases. One main issue with this method is that it gives equal weights to votes based on different data sources. The underlying assumption, therefore, is that the quality of predictions is the same for all sources of information. This is, of course, not a realistic assumption for many real problems. In our previous paper (Shahbaba and Neal, 2006), we provided a new scheme for combining different sources of information. In this approach, we use separate scale parameters, ξ, for each data source in order to adjust their relative weights automatically. This allows the coefficients from different sources of data to have appropriately different variances in the model.

For models developed by Ding and Dubchak (2001), the highest accuracy rate, 56.5, was achieved only when they combined the covariates based on the composition of amino acids, secondary structure, hydrophobicity, and polarity. We also used these four datasets, and applied our models to the combined data. We used a different scale parameters, ξ, for each dataset. The results from our models are presented in Table 5. For comparison, we also present the results obtained by Ding and Dubchak (2001) based on the same datasets. As we can see, this time, using the hierarchy results in more substantial improvements. Moreover, nonlinear models provided better performance compared to their corresponding linear models.

7 Conclusions 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 segments of data.

As mentioned above, for sampling from the posterior distribution, we used multiple chains which appeared to be sampling different regions of the posterior space. Ideally, we prefer to have one chain that can efficiently sample from the whole posterior distribution. In future, we intend to improve our MCMC sampling. For this purpose, we can use more efficient methods, such as the "split-merge" approach introduced by Jain and Neal (2007) and the short-cut Metropolis method introduced by Neal (2005).

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 covariates with multinomial distribution, we can either extend the Bernoulli distribution by using $(\mu_1, ..., \mu_K)$, where K is the number of categories in x, or use K continuous latent variables, $z_1, ..., z_K$, and set $\theta_j = \exp(z_j)/\sum_{j'}^{K} \exp(z'_j)$.

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. The distribution of y can be assumed normal within a component. We 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.

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. In future, we will examine the application of our approach for these problems.

References

Antoniak CE (1974). "Mixture of Dirichlet process with applications to Bayesian nonparamteric problems." *Annals of Statistics*, **273(5281)**, 1152–1174.

Blackwell D, MacQueen JB (1973). "Ferguson distributions via Polya urn scheme." *Annals of Statistics*, **1**, 353–355.

Breiman L, Friedman J, Olshen RA, Stone CJ (1993). *Classification and Regression Trees*. Chapman and Hall, Boca Raton.

Bush CA, MacEachern SN (1996). "A semi-parametric Bayesian model for randomized block designs." *Biometrika*, **83**, 275–286.

Cai B, Dunson DB (2006). "Bayesian covariance selection in generalized linear mixed models." *Biometrics*, **62**, 446–457.

Daniels MJ, Kass RE (1999). "Nonconjugate Bayesian estimation of covariance matrices and its use in hierarchical models." *Journal of the American Statistical Association*, **94**(448), 1254–1263.

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

Dubchak I, Muchnik J, Holbrook S, Kim S (1995). "Prediction of protein folding class using global description of amino acid sequence." *Proceedings of the National Academy of Sciences (USA)*, **92**, 8700–8704.

Escobar MD, West M (1995). "Bayesian density estimation and inference usinng mixtures." *Journal of American Statistical Society*, **90**, 577–588.

Ferguson TS (1973). "A Bayesian analysis of some nonparamteric problems." *Annas of Statistics*, **1**, 209–230.

Hobohm U, Sander C (1994). "Enlarged representative set of proteins." *Protein Science*, **3**, 522–524.

Hobohm U, Scharf M, Schneider R, Sander C (1992). "Selection of a representative set of structure from the Brookhaven Protein Bank." *Protein Science*, **1**, 409–417.

Jain S, Neal RM (2007). "Splitting and merging components of a nonconjugate Dirichlet process mixture model." *to appear in Bayesian Analysis*.

Lo Conte L, Ailey B, Hubbard T, Brenner SE, Murzing A, Chothia C (2000). "SCOP: a structural classification of protein database." *Nucleic Acids Research*, **28**, 257–259.

MacEachern SN, Müller P (1998). "Estimating mixture of Dirichlet process models." *Journal of Computational and Graphcal Statistics*, **7**, 223–238.

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

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

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

Neal RM (2000). "Markov chain sampling methods for Dirichlet process mixture models." *Journal of Computational and Graphical Statistics*, **9**, 249–265.

Neal RM (2003). "Slice sampling." *Annals of Statistics*, **31**(3), 705–767.

Neal RM (2005). "The short-cut Metropolis method." *Technical Report 0506*, Department of Statistics, University of Toronto.

Shahbaba B, Neal RM (2006). "Gene function classification using Bayesian models with hierarchy-based priors." *BMC Bioinformatics*, **7:448**.

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