

IMPROVING CLASSIFICATION MODELS WHEN A CLASS HIERARCHY IS
AVAILABLE

by

Babak Shahbaba

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Public Health Sciences
University of Toronto

Copyright © 2007 by Babak Shahbaba

Abstract

Improving classification models when a class hierarchy is available

Babak Shahbaba

Doctor of Philosophy

Graduate Department of Public Health Sciences

University of Toronto

2007

We introduce a new method for modeling hierarchical classes, when we have prior knowledge of how these classes can be arranged in a hierarchy. The application of this approach is discussed for linear models, as well as nonlinear models based on Dirichlet process mixtures. Our method uses a Bayesian form of the multinomial logit (MNL) model, with a prior that introduces correlations between the parameters for classes that are nearby in the hierarchy. Using simulated data, we compare the performance of the new method with the results from the ordinary MNL model, and a hierarchical model based on a set of nested MNL models. We find that when classes have a hierarchical structure, models that acknowledge such existing structure in data can perform better than a model that ignores such information (i.e., MNL). We also show that our model is more robust against misspecification of class structure compared to the alternative hierarchical model. Moreover, we test the new method on page layout analysis and document classification problems, and find that it performs better than the other methods. Our original motivation for conducting this research was classification of gene functions. Here, we investigate whether functional annotation of genes can be improved using the hierarchical structure of functional classes. We also 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. In this approach, 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 extend this method to classification problems where a class hierarchy is available. We use our model to predict protein folding classes, which can be arranged in a hierarchy. We find that our model provides substantial improvement over previous methods, which were based on Neural Networks (NN) and Support Vector Machines (SVM). Together, the results presented in this thesis show that higher predictive accuracy can be obtained using Bayesian models that incorporate suitable prior information.

To my lovely wife Rezy

None of this would be possible without your love and support

Acknowledgements

First and foremost, I would like to thank my supervisor Professor Radford Neal for sharing his wealth of knowledge and wisdom with me, supporting me throughout this journey, and teaching me almost everything I know about Statistics. He also taught me how to think freely and write clearly. I was tremendously fortunate to have him as my supervisor.

I was very privileged to work with a great advisory committee. I thank Drs. Paul Corey and Celia Greenwood for their expertise, availability and commitment to education. I am also grateful to Drs. Steven MacEachern and Andrew Paterson for reading my thesis and providing valuable comments.

I would also like to thank my parents, my wife, my sister, my brother, and my brother-in-law for their love and support. Their constant encouragement helped me throughout the pursuit of this degree.

My sincere gratitude goes to Kevin Laven, Sam Roweis and Scott Leishman for providing the NIPS data, and Lijuan Cai and Thomas Hofmann for providing the processed WIPO-alpha dataset. I would also like to thank Ross King, Andreas Karwath, Amanda Clare and Luc Dehaspe for providing the processed *E. coli* datasets, and Johanna Rommens for helping me with the interpretation of our findings about *E. coli* genome.

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

Contents

1	Overall introduction	1
1.1	Introduction	2
1.2	Outline of thesis	5
2	Classification models with a hierarchy-based prior	8
2.1	Introduction	9
2.2	Hierarchical Classification	11
2.3	Results for Synthetic Datasets	15
2.4	Results for Real Datasets	20
2.4.1	Performance Measures	20
2.4.2	NIPS Dataset	22
2.4.3	WIPO-alpha Dataset	26
2.5	Conclusions and Future Directions	27
3	Gene function classification	30
3.1	Background	31
3.2	Results and Discussion	35
3.3	Conclusions	42
3.4	Methods	44
3.4.1	Implementation	47

4	Nonlinear classification using Dirichlet process mixtures	49
4.1	Introduction	50
4.2	Methodology	54
4.3	Results for synthetic data	57
4.4	Results for protein fold classification	62
4.5	Extension to hierarchical classes	66
4.6	Extension to multiple datasets	68
4.7	Conclusions and future directions	69
5	Overall discussion	72
	Appendix A	76
	Appendix B	79
B.1	The Gibbs sampler	79
B.2	The Metropolis algorithm	80
B.3	Slice sampling	82
	Appendix C	84
	Appendix D	87
	Bibliography	88

List of Tables

2.1	Comparison of models on simulated data created based on Figure 2.2. Average log-probability (log-prob) and accuracy rate (acc) are estimated on the test sets.	18
2.2	Comparison of models on simulated data created based on Figure 2.3. Average log-probability (log-prob) and accuracy rate (acc) are estimated on the test sets.	19
2.3	Performance of models based on NIPS dataset. Here, “acc”, “pacc” and “ Δ -loss” refer to accuracy, parent accuracy and taxonomy-based loss respectively. Larger values are better except for Δ -loss.	25
2.4	Performance of models based on WIPO-alpha dataset, section “D”. Here, “acc”, “pacc” and “ Δ -loss” refer to accuracy, parent accuracy and taxonomy-based loss respectively. The SVM and hierarchical SVM (hSVM) are developed by Cai and Hoffmann (2004). Larger values are better except for Δ -loss.	28
3.1	Comparison of models based on their predictive accuracy (%) using each data source separately.	37
3.2	Comparison of models based on their predictive accuracy (%) for specific coverage (%) provided in parenthesis. The C5 results and the coverage values are from King <i>et al.</i> (2001).	39

3.3	Accuracy (%) of models on the combined dataset with and without separate scale parameters. Results using SIM alone are provided for comparison.	40
3.4	Predictive accuracy (%) for different coverage values (%) of the corMNL model using all three sources with separate scale parameters.	40
4.1	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.	60
4.2	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.	62
4.3	Performance of models based on protein fold classification data. NN and SVM use maximum likelihood estimation, and are developed by Ding and Dubchak (2001).	66
4.4	Comparison of hierarchical models (linear and nonlinear) with non-hierarchical models (linear and nonlinear) based on protein fold classification data.	68
4.5	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.	69
C.1	Comparison of direct functional annotation of several ORFs (whose function was unknown in 2001) with predicted functions using our corMNL model. In this table, we only present genes for which there is a close match between our predictions and the results from direct biological experiments.	85

C.2 Comparison of direct functional annotation of several ORFs (whose function was unknown in 2001) with predicted functions using our corMNL model. In this table, we only present genes for which there is not a close match between our predictions and the results from direct biological experiments.	86
--	----

List of Figures

2.1	A part of a gene annotation hierarchy proposed by Riley (1993) for the <i>E. coli</i> genome.	10
2.2	The simple model used for the simulation study. The coefficient parameters for each classes are presented as a sum of parameters at different level of hierarchy.	17
2.3	A hypothetical hierarchy with a more complex structure.	19
2.4	Hierarchical structure of document labels.	22
2.5	Trace plots of the ARD hyperparameters, σ_l , for two covariates of the corMNL model applied to the NIPS dataset.	24
2.6	Trace plots of two overall scale hyperparameters, τ_m , for two nodes of the corMNL model for two nodes applied to the NIPS dataset.	24
3.1	A simple representation of the corMNL model.	45
4.1	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.	58
4.2	A random sample generated according to Simulation 2 with $a_3 = 0$. The dotted line is the optimal boundary function.	61

4.3	A simple representation of our hierarchical classification model.	67
A.1	The boundary lines obtained by the MNL model for a simulated three-way classification problem. As we can see, class 1 and class 2 are relatively more difficult to distinguish.	78
A.2	The boundary lines obtained by the treeMNL model for a simulated three-way classification problem. The boundaries are not linear between classes 1 and 3, and between classes 2 and 3.	78

Chapter 1

Overall introduction

1.1 Introduction

In this research, we consider classification problems where classes have a hierarchical structure. We discuss both linear and nonlinear (based on Dirichlet process mixtures) hierarchical classification models. The hierarchy reflects our prior opinion with respect to the similarity of classes. Hierarchical classification problems of this sort are abundant in statistics and machine learning fields. One such problem, which is our original motivation for studying hierarchical classification schemes, is prediction of the biological functions of genes. These functions are usually presented in a hierarchical form starting with very general classes (eg, cell processes) and becoming more specific in lower levels of the hierarchy (eg, cell division or chromosome replication).

There are many examples of hierarchical classes in the literature. Fox (1997) discusses one simple hierarchical classification problem regarding Canadian women’s labour-force participation. In this problem, the objective is to classify Canadian women to “working” and “non-working” (outside the home) groups. Working women are further classified to “full-time” and “part-time”. Husband’s income, presence of children in the household, and region of living were used as covariates. Another example, which has become increasingly important due to the exponential growth of information on the internet, is classification of documents (e.g., web pages). A standard benchmark dataset for this problem is one that contains 20000 documents from 20 different on-line newsgroups. The objective is to identify the newsgroup from which an article was taken, using the words appearing in that article. The newsgroups can be grouped into general classes such as “computer”, “science”, and “talk”. A general class such as science can be divided to more specific classes such as “electronics” and “space”. The dataset can be obtained from the UCI KDD repository at <http://kdd.ics.uci.edu>. We discuss some more examples in our papers (Chapters 2, 3, and 4), and evaluate our proposed method based on page region labelling and patent document classification problems.

If we ignore the hierarchical structure of classes, we could use a simple multinomial

logit (MNL) model for classification. This model has the following form:

$$P(y = j|x, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\exp(\alpha_j + x\boldsymbol{\beta}_j)}{\sum_{j'=1}^c \exp(\alpha_{j'} + x\boldsymbol{\beta}_{j'})} \quad (1.1)$$

where c is the number of classes. For each class, j , there is a vector of p unknown parameters $\boldsymbol{\beta}_j$. The entire set of regression coefficients $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_c)$ can be presented as a $p \times c$ matrix. The MNL model treats classes as unrelated entities. This is not, of course, an optimal approach if we know *a priori* that classes can be arranged on a hierarchy.

The importance of using the information regarding the class hierarchy has been emphasized by many authors (e.g., Sattath and Tversky (1977); Fox (1997); Koller and Sahami (1997)). Several methods have been proposed for incorporating this information in classification models. A common approach is using a series of nested models (i.e., nested MNL models), which classify cases sequentially along the hierarchy. That is, for each intermediate node m in the hierarchy, a separate model is used to classify cases that belong to m into the child nodes of m (i.e., nodes that are immediately connected to m). We refer to this method as treeMNL.

To understand why treeMNL may perform better when classes have a hierarchical structure, consider a three-way classification problem in which classes 1 and 2 are completely indistinguishable on the basis of available information, but they both can be easily separated from class 3. This is, of course, an extreme example. Assuming that the hierarchical structure is known to us *a priori*, we can use two nested models such that the first model separates class 3 from classes 1 and 2, and the second model separates only class 1 from class 2. Obviously, only the first model affects the overall performance (assuming the second model has a performance no worse than a random model). The first model is appropriately more parsimonious compared to a non-hierarchical model, which tries to classify objects to all three classes simultaneously (i.e., MNL). The treeMNL model can, therefore, provide better performance than the MNL model.

The extreme example discussed above rarely exists in real life. In most problems,

similar classes can be distinguished to some degree. It is even possible that apparently similar classes are not in fact hard to distinguish. That is, the hierarchy is misspecified. As we will see later, this can be a major problem for the treeMNL model. Fox (1997) suggested using this approach only when there is substantial evidence to believe the hierarchy exists.

In this research, we propose an alternative approach for incorporating the hierarchical structure in classification models. Using simulation studies, we show that unlike treeMNL, our approach is very robust against misspecification of the hierarchy. In our approach, the hierarchy is provided in the form of a prior. We use an MNL model with a prior that introduces correlations between the parameters for classes that are nearby in the hierarchy. For this purpose, we first use a different set of parameters for each branch of the hierarchy. In the final multinomial logit, we sum these parameters over all the branches starting from the root and leading to a specific class. Therefore, the more branches two categories share (i.e., the closer they are on the hierarchy), the larger number of their common parameters will be. This can in turn translate to a higher correlation between the parameters of multinomial logit model. The correlation will be negligible (i.e., classes are independent) if the common parameters are all practically zero. When the hierarchy actually provides additional information regarding the structure of classes, our approach performs better than the non-hierarchical model (i.e., MNL), and the nested hierarchical model (i.e., treeMNL). When the assumed hierarchical structure does not exist, our model has a very small penalty since it can reduce to a non-hierarchical model in such situations. Throughout this thesis, we refer to this model as corMNL.

While MNL and corMNL use different priors, they both use the same model in which the relationship between response variable and covariates is assumed to be linear. If the linearity assumption does not hold, these models may have poor performance compared to alternative nonlinear models. We introduce a new model that is sufficiently flexible to capture nonlinear relationships. In this approach, 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 relationship becomes nonlinear over all components. We first use this method for modeling non-hierarchical classes. For this purpose, we use an MNL model in each component to capture the dependency of y on x . For classification problems where classes have a hierarchical structure, we replace the MNL model with corMNL.

1.2 Outline of thesis

This thesis is presented as a “journal format” dissertation. It is divided into 5 chapters: an overall introduction, three papers, and an overall discussion. Each paper is written to stand on its own, and therefore, includes a separate introduction, a methodology section, a results section, and a discussion section. The papers included in this thesis are: “Improving classification when a class hierarchy is available using a hierarchy-based prior”, which was published by Bayesian Analysis in 2007, “Gene function classification using Bayesian models with hierarchy-based priors”, which was published by BMC Bioinformatics in 2006, and “Nonlinear classification models using Dirichlet process mixtures”, which will be published as a technical report in the department of Statistics.

In the next chapter, we present our first paper, which discusses the problem of hierarchical classification in general. We used simulated data to compare our model to MNL and treeMNL. The results from these simulation studies show that when classes have a hierarchical structure, models that acknowledge such existing structure in data (i.e., treeMNL and corMNL) can perform better than a model that ignores such information (i.e., MNL). We also show that our model is more robust against misspecification of class structure compared to the nested model. In this paper, we also looked at two real examples of hierarchical classification. The first example involves classifying regions of a page in an article, and the second example involves classifying patent documents. The

results based on these two problems confirmed that our model performs better than the other two alternative models.

In Chapter 3, our second paper, we discuss the application of our method to gene function classification. An experimental approach to identifying gene function is based on eliminating or inhibiting expression of a gene and observing any alteration in the phenotype. However, since analysis of all genes for all possible functions is not possible at this current time, statistical models have been employed for this purpose. In this paper, we use our approach to predict the functional class of Open Reading Frames (ORFs) from the *E. coli* genome. An ORF is a part of a genome sequence that could potentially encode proteins. The covariates are based on several characteristics of protein sequences including phylogenic descriptors (SIM), sequence based attributes (SEQ), and predicted secondary structure (STR). Similar to the results in the previous paper, we find that our model has the best performance in terms of prediction accuracy. Moreover, all our models (hierarchical or non-hierarchical) provide substantially better predictions than a previous analysis based on the C5 decision tree algorithm.

In our second paper, we also address the problem of combining different sources of information in gene function classification. Our approach is based on using separate scale parameters for different sources of data in order to adjust their relative weights automatically. This approach provides a higher accuracy rate when compared to models that use each data source (i.e., SIM, SEQ and STR) alone. In previous work by King *et al.* (2001), combining these three different datasets showed no additional benefit compared to using phylogenic descriptors alone.

In the first two papers, we focus only on linear models. We expected that the same approach could be also used for nonlinear models. In Chapter 4, we present our third paper, which introduces a novel nonlinear model for classification based on Dirichlet process mixtures. Simulation studies are used to evaluate the performance of this method. We also apply our model to a real classification problem, where the objective is to identify

protein folds. Protein fold recognition plays an important role in predicting the function of new sequences. Folding classes of protein have a hierarchical structure. We show how our method can be extended to classification problems where a class hierarchy is available.

Finally, Chapter 5 is devoted to a discussion and future directions. In this chapter, we provide an overall review of the three papers presented in the previous chapters, and discuss the connection between these papers. We also provide several suggestions on how to improve the proposed methods in future research. Other possible applications of these methods are also discussed.

Several appendices are also included in the dissertation. Appendix A presents a brief discussion about the difference between the MNL model and the treeMNL model. Using a simple example, we show how the decision boundaries are different between these two models. In Appendix B, we provide an overview of different MCMC algorithms used in this dissertation. We discuss three main sampling methods, namely, the Gibbs sampler, the Metropolis method, and slice sampling. Appendix C presents the comparison between our predicted functions and the results from direct biological experiments for a subset of *E. coli* ORFs. Appendix D includes reprints of the published papers (Chapter 2 and Chapter 3).

Chapter 2

Classification models with a hierarchy-based prior

This chapter appears in *Bayesian Analysis*, 2007, 2, Number 1, pp. 221-238.

Abstract

We introduce a new method for building classification models when we have prior knowledge of how the classes can be arranged in a hierarchy, based on how easily they can be distinguished. The new method uses a Bayesian form of the multinomial logit (MNL, a.k.a. “softmax”) model, with a prior that introduces correlations between the parameters for classes that are nearby in the tree. We compare the performance on simulated data of the new method, the ordinary MNL model, and a model that uses the hierarchy in a different way. We also test the new method on page layout analysis and document classification problems, and find that it performs better than the other methods.

2.1 Introduction

In this paper, we consider classification problems where classes have a hierarchical structure. The hierarchy reflects our prior opinion regarding similarity of classes. Two classes are considered similar if it is difficult to distinguish them from each other on the basis of the features available. The similarity of classes increases as we descend the hierarchy.

Our original motivation for studying hierarchical classification schemes was prediction of the biological functions of genes. Functions are usually presented in a hierarchical form starting with very general classes (eg, cell processes) and becoming more specific in lower levels of the hierarchy (eg, cell division). Figure 2.1 shows a small part of the scheme proposed by Riley (1993) to catalogue the proteins of *Escherichia coli*. We discuss this application of our methods elsewhere (Shahbaba and Neal, 2006). Here, we discuss this problem more generally, and illustrate its use for two other examples of hierarchical classification. We look at the problem of classifying regions of a page in an article, using classes such as “Section Heading”, “Text”, or “Figure Caption”, which can be arranged in a hierarchy based on distinguishability. We also look at the problem of classifying patent documents relating to textiles, where again the classes can be arranged in a hierarchy in

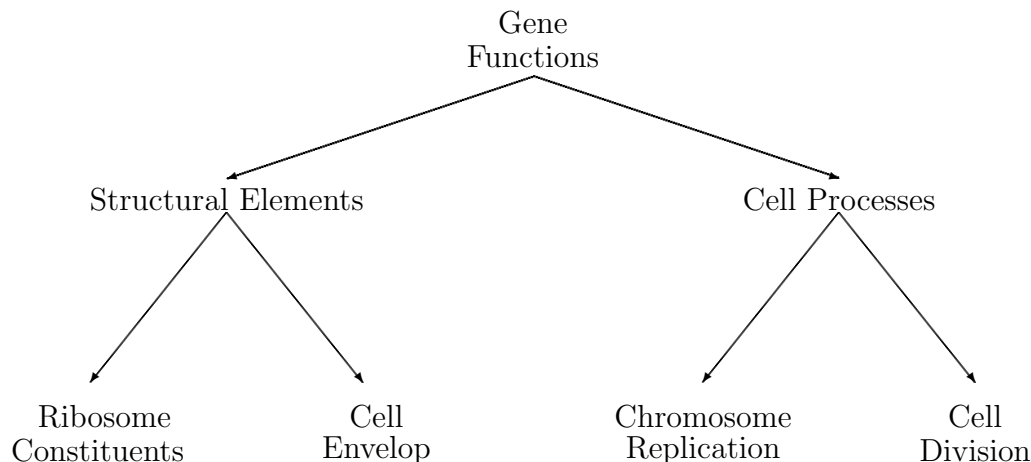


Figure 2.1: A part of a gene annotation hierarchy proposed by Riley (1993) for the *E. coli* genome.

which, for example, a high-level category of “Weaving” contains sub-classes for “Looms” and for “Weavers’ Tools”.

In a Bayesian model, we can incorporate prior knowledge of the class hierarchy using a suitable prior distribution over parameters of the model. In this paper, we introduce a new method of this sort for the multinomial logit (MNL) model, in which the regression coefficients for classes that are nearby in the hierarchy are correlated in the prior.

This paper is organized as follows. In Section 2.2, simple classification models and their extensions for analysing hierarchical classes are discussed. In Section 2.3, using simulated data, we compare the performance of our model, the ordinary MNL model, and an alternative model that uses the hierarchy in a different way. In Section 2.4 we compare the same models on the page region labelling and patent document classification problems. The last section summarizes our findings and presents some ideas for future research.

2.2 Hierarchical Classification

Consider a classification problem in which we have observed data for n cases, $(x^{(1)}, y^{(1)})$, $\dots, (x^{(n)}, y^{(n)})$, where $x^{(i)} = (x_1^{(i)}, \dots, x_p^{(i)})$ is the vector of p covariates (features) for case i , and $y^{(i)}$ is the associated class. Our goal is to classify future cases for which the class membership is unknown but the covariates are available. For binary classification problems, a simple logistic model can be used:

$$P(y = 1|x, \alpha, \boldsymbol{\beta}) = \frac{\exp(\alpha + x\boldsymbol{\beta})}{1 + \exp(\alpha + x\boldsymbol{\beta})} \quad (2.1)$$

Here, α is the intercept, $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown parameters and $x\boldsymbol{\beta}$ is its inner product with the covariate vector.

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):

$$P(y = j|x, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\exp(\alpha_j + x\boldsymbol{\beta}_j)}{\sum_{j'=1}^c \exp(\alpha_{j'} + x\boldsymbol{\beta}_{j'})} \quad (2.2)$$

where c is the number of classes. For each class, j , there is a vector of p unknown parameters $\boldsymbol{\beta}_j$. The entire set of regression coefficients $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_c)$ can be presented as a $p \times c$ matrix. This representation is redundant, since one of the $\boldsymbol{\beta}_j$'s 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. For this model we can use the following priors:

$$\alpha_j | \eta \sim N(0, \eta^2)$$

$$\beta_{jl} | \tau \sim N(0, \tau^2)$$

$$\eta^{-2} \sim \text{Gamma}(v, V)$$

$$\tau^{-2} \sim \text{Gamma}(w, W)$$

where $j = 1, \dots, c$ and $l = 1, \dots, p$.

The MNL model treats classes as unrelated entities without any hierarchical structure. This is not always a realistic assumption. In many classification problems, like those discussed above, one can arrange classes in a hierarchical form analogous to the hierarchy of species arranged in genera, families, etc. If the classes have in fact the assumed structure, one would expect to obtain a higher performance by using this additional information. A special case is when the classes are ordered (e.g., education level). For these problems a more parsimonious model (e.g., cumulative logit model) with improved power can be used (Agresti, 2002).

The importance of using the hierarchy in classification models has been emphasized by many authors (e.g., Sattath and Tversky, 1977; Fox, 1997; Koller and Sahami, 1997). One approach for modelling hierarchical classes is to decompose the classification model into nested models (e.g., logistic or MNL). Nested MNL models are extensively discussed in econometrics (e.g., Sattath and Tversky, 1977; McFadden, 1980) in the context of estimating the probability of a person choosing a specific alternative (i.e., class) from a discrete set of options (e.g., different modes of transportation). These models, known as discrete choice models, aim at forecasting and explaining human decisions through optimizing an assumed utility (preference) function, which is different from our aim of maximizing classification accuracy.

Goodman (2001) showed that using hierarchical classes can significantly reduce the training time of maximum entropy-based language models and results in slightly lower perplexities. He illustrated his approach using a word labelling problem, and recommended that instead of predicting words directly, we first predict the category to which the word belongs, and then predict the word itself. Such a two-level hierarchical model was also used by Weigend *et al.* (1999) for document classification. They evaluated their model on the Reuters-22173 corpus and showed significant improvement, especially for rare classes.

For hierarchical classification problems with simple binary partitions, Fox (1997) sug-

gested using successive logistic models for each binary class. In Figure 2.2 below, for example, these partitions are $\{12, 34\}$, $\{1, 2\}$, and $\{3, 4\}$. The resulting nested binary models are statistically independent, conditioned on the upper levels. The likelihood can therefore be written as the product of the likelihoods for each of the binary models. For example, in Figure 2.2 we have

$$P(y = 1|x) = P(y \in \{1, 2\}|x) \times P(y \in \{1\}|y \in \{1, 2\}, x) \quad (2.3)$$

Restriction to binary models is unnecessary. At each level, classes can be divided into more than two subsets and MNL can be used instead of logistic regression. We refer to methods based on decomposing the tree structure into nested MNL models as treeMNL. Consider a parent node, m , with c_m child nodes, representing sets of classes defined by S_k , for $k = 1, \dots, c_m$. The portion of the nested MNL model for this node has the form:

$$\begin{aligned} P(y \in S_k|x, \boldsymbol{\alpha}_m, \boldsymbol{\beta}_m) &= \frac{\exp(\alpha_{mk} + x\boldsymbol{\beta}_{mk})}{\sum_{k'=1}^{c_m} \exp(\alpha_{mk'} + x\boldsymbol{\beta}_{mk'})} \\ \alpha_{mk}|\eta_m &\sim N(0, \eta_m^2) \\ \beta_{mkl}|\tau_m &\sim N(0, \tau_m^2) \\ \eta_m^{-2} &\sim \text{Gamma}(v_m, V_m) \\ \tau_m^{-2} &\sim \text{Gamma}(w_m, W_m) \end{aligned}$$

where $l = 1, \dots, p$. We calculate the probability of each end node, j , by multiplying the probabilities of all intermediate nodes leading to j .

In contrast to this treeMNL model, Mitchell (1998) showed that the hierarchical naive Bayes classifier is equivalent to the standard non-hierarchical classifier when the probability terms are estimated by maximum likelihood. To improve the hierarchical naive Bayes model, McCallum *et al.* (1998) suggested to smooth parameter estimates of each end node by shrinking its maximum likelihood estimate towards the estimates of all its ancestors in the hierarchy. More recently, new hierarchical classification models based on large margin principals, specifically support vector machine (SVM), have been proposed

(Dumais and Chen, 2000; Dekel *et al.*, 2004; Cai and Hoffmann, 2004; Tsochantaridis *et al.*, 2004; Cesa-Bianchi *et al.*, 2006). Dekel *et al.* (2004) introduced a large margin hierarchical classification model that uses the sum of parameters along the tree for classifying cases to the end nodes. These parameters are estimated based on a set of classifiers that assign cases to the intermediate nodes. Cai and Hoffmann (2004) suggested a similar approach based on the generalization of multiclass SVM. We also use sums of parameters along paths in the tree, but in a rather different way from this past work.

Our new framework for modeling hierarchical classes is illustrated in Figure 2.2, which shows a hierarchical classification problem with four classes. For each branch in the hierarchy, we define a different set of parameters. In Figure 2.2, these parameters are denoted as ϕ_{11} and ϕ_{12} for branches in the first level and ϕ_{21} , ϕ_{22} , ϕ_{23} and ϕ_{24} for branches in the second level. We assign objects to one of the end nodes using an MNL model (Equation 3.1) whose regression coefficients for class j are represented by the sum of parameters on all the branches leading to that class. In Figure 2.2, these coefficients are $\beta_1 = \phi_{11} + \phi_{21}$, $\beta_2 = \phi_{11} + \phi_{22}$, $\beta_3 = \phi_{12} + \phi_{23}$ and $\beta_4 = \phi_{12} + \phi_{24}$ for classes 1, 2, 3 and 4 respectively. Sharing the common terms, ϕ_{11} and ϕ_{12} , introduces prior correlation between the parameters of nearby classes in the hierarchy.

In our model, henceforth called corMNL, ϕ 's are vectors with the same size as β 's. We assume that, conditional on higher level hyperparameters, all the components of the ϕ 's are independent, and have normal prior distributions with zero mean. The variances of these components are regarded as hyperparameters, which control the magnitudes of coefficients. We use one hyperparameter for each non-terminal node. When a part of the hierarchy is irrelevant, we hope the posterior distribution of its corresponding hyperparameter will be concentrated near zero, so that the parameters it controls will also be close to zero.

In detail, the prior we use is as follows:

$$\begin{aligned}\alpha_j|\eta &\sim N(0, \eta^2) \\ \phi_{mkl}|\tau_m &\sim N(0, \tau_m^2) \\ \eta^{-2} &\sim \text{Gamma}(v, V) \\ \tau_m^{-2} &\sim \text{Gamma}(w_m, W_m)\end{aligned}$$

Here, $j = 1, \dots, c$ indexes classes, and ϕ_{mkl} refers to the parameter related to covariate x_l and branch k of node m . The ϕ parameters of all the branches that share the same node are controlled by one hyperparameter, τ_m , which controls the degree to which that portion of the hierarchy is active. In Figure 2.2, for example, when the hyperparameters in the first level are small (compared to the hyperparameters in the second level), the model reduces to simple MNL. In contrast, when these hyperparameters are relatively large, the model reinforces our assumption of hierarchical classes.

By introducing prior correlations between parameters for nearby classes, we can better handle situations in which these classes are hard to distinguish. If the hierarchy actually does provide information about how distinguishable classes are, we expect that performance will be improved. This would be especially true when the training set is small and the prior has relatively more influence on the results. Using an inappropriate hierarchy will likely lead to worse performance than a standard MNL model, but since the hyperparameters can adapt to reduce the prior correlations to near zero, the penalty may not be large.

2.3 Results for Synthetic Datasets

So far, we have discussed three alternative models: MNL, treeMNL, and corMNL. We first compare these models using a synthetic four-way classification problem with two covariates. Data are generated from each of these models in turn, and then fit with each

model in order to test the robustness of the models when applied to data generated from other models.

All regression parameters are given normal priors with mean zero. For the MNL model, the standard deviation for all the intercepts, η , and the standard deviation for the rest of coefficients, τ , have the following priors:

$$\eta^{-2} \sim \text{Gamma}(1, 10) \quad (0.16, 0.38, 1.98)$$

$$\tau^{-2} \sim \text{Gamma}(1, 1) \quad (0.52, 1.20, 6.27)$$

We use the parameterization of the Gamma distribution in which $\text{Gamma}(a, b)$ has density $f(x|a, b) = [b^a \Gamma(a)]^{-1} x^{a-1} e^{-x/b}$, for which the mean is ab and the standard deviation is $a^{1/2}b$. The 2.5, 50 and 97.5 percentiles of τ and η are shown in parenthesis.

For the treeMNL and corMNL models, we assume that classes are arranged in a hierarchical form as shown in Figure 2.2. This hierarchy implies that while it might be easy to distinguish between groups $\{1, 2\}$ and $\{3, 4\}$, further separation of classes might not be as easy. As mentioned above, the treeMNL model for this hierarchy is comprised of three nested logistic models. These models are: $P(y \in \{1, 2\}|\alpha_1, \beta_1, x)$, $P(y = 1|\alpha_2, \beta_2, x, y \in \{1, 2\})$ and $P(y = 3|\alpha_3, \beta_3, x, y \in \{3, 4\})$. The priors for the treeMNL and corMNL models are discussed in section 2. The variances of the regression parameters, β , in treeMNL and of the ϕ 's in corMNL are regarded as hyperparameters. For these two models, one hyperparameter controls all the parameter emerging from the same node. These hyperparameters are given the following prior distributions:

$$\eta^{-2} \sim \text{Gamma}(1, 10) \quad (0.16, 0.38, 1.98)$$

$$\tau_1^{-2} \sim \text{Gamma}(1, 5) \quad (0.23, 0.54, 2.82)$$

$$\tau_2^{-2} \sim \text{Gamma}(1, 20) \quad (0.05, 0.12, 0.63)$$

$$\tau_3^{-2} \sim \text{Gamma}(1, 20) \quad (0.05, 0.12, 0.63)$$

Here, τ_1 , τ_2 and τ_3 correspond to nodes 1, 2, and 3 respectively (Figure 2.2). These parameters have a narrower prior compared to τ in the MNL model. This is to account

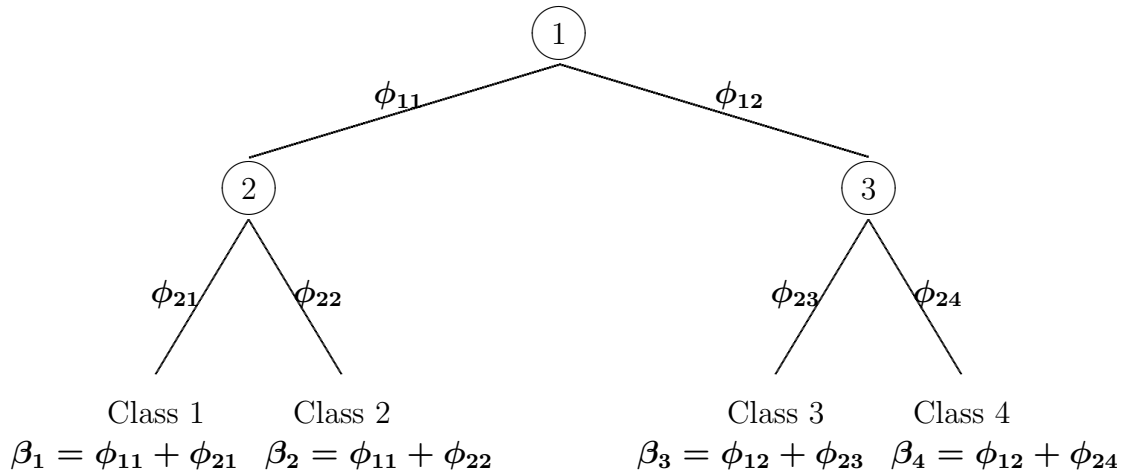


Figure 2.2: The simple model used for the simulation study. The coefficient parameters for each classes are presented as a sum of parameters at different level of hierarchy.

for the fact that the role of β in the MNL model is played by more than one parameter in treeMNL and corMNL. Moreover, the regression parameters in the second level of hierarchy have a relatively smaller standard deviation τ . As a result, these parameters tend to be smaller, making separation of class 1 from 2 and class 3 from 4 more difficult.

We do three tests, in which we assume that each of the MNL, treeMNL and corMNL is the correct model. This allows us to see how robust each model is when data actually come from a somewhat different model. For each test, we sample a set of parameters from the prior distribution of the corresponding model. Pairs of data items $(x^{(i)}, y^{(i)})$ are generated by first drawing 10000 independent samples $x_1^{(i)}, x_2^{(i)}$ from the uniform $(-5, 5)$ distribution and then assigning each data item to one of the four possible classes. The assignment is either based on a multinomial model (for data generated from MNL and corMNL) or based on successive logistic models (for data generated from treeMNL).

All three models were trained on the first 100 data items and tested on the remaining 9900 items. The regression coefficients were sampled from their posterior distribution using MCMC methods with single-variable slice sampling (Neal, 2003), using the “stepping out” procedure to find an interval around the current point, and then the “shrinkage”

N=100		Data					
		MNL		treeMNL		corMNL	
		log-prob	acc	log-prob	acc	log-prob	acc
Method	MNL	-0.7958	67.1	-0.8918	58.4	-0.9168	59.4
	treeMNL	-0.8489	65.0	-0.8770	58.7	-0.9113	59.4
	corMNL	-0.7996	67.1	-0.8797	58.6	-0.9075	59.5

Table 2.1: Comparison of models on simulated data created based on Figure 2.2. Average log-probability (log-prob) and accuracy rate (acc) are estimated on the test sets.

procedure to sample from this interval. Since the hyperparameters were given conjugate priors, direct Gibbs sampling could be used for them. For all tests we ran 1000 MCMC iterations to sample from the posterior distributions. We discarded the initial 250 samples and used the rest for prediction. Performance on the test set is measured in terms of average log-probability (based on the estimated probability of the correct class) and accuracy rate (defined as the percentage of the times the correct class is predicted). We make predictions based on the posterior predictive probabilities.

The above procedure was repeated 100 times. Each time, new regression parameters were sampled from priors and new pairs of data items were created based on the assumed models. The average results (over 100 repetitions) are presented in Table 2.1. In this table, each column corresponds to the model used for generating the data and each row corresponds to the model used for building the classifier. As we can see, the diagonal elements have the best performance in each column. That is, the model whose functional form matches the data generation mechanism performs significantly better than the other two models (all p -values based on average log-probability are less than 0.01 using a paired t -test with $n = 100$). Moreover, the results show that when the samples are generated according to the MNL model (i.e., classes are unrelated), corMNL has a significantly (p -value < 0.001) better performance compared to treeMNL. When treeMNL is used to generate data, corMNL performs only slightly worse than treeMNL. The conclusions remain the same when we use different priors and ranges of covariates.

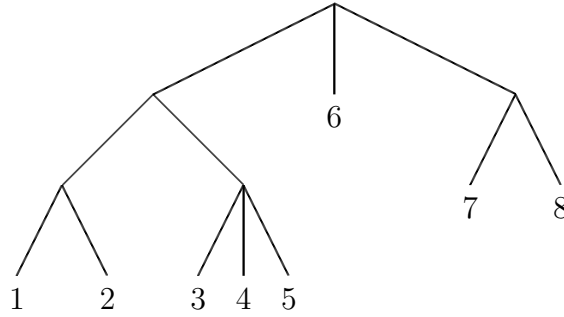


Figure 2.3: A hypothetical hierarchy with a more complex structure.

N=100		Data					
		MNL		treeMNL		corMNL	
		log-prob	acc	log-prob	acc	log-prob	acc
Method	MNL	-0.2539	89.9	-0.3473	87.7	-0.3106	88.7
	treeMNL	-0.6837	76.9	-0.2898	90.3	-0.3614	87.9
	corMNL	-0.2910	89.7	-0.2854	90.1	-0.2841	90.3

Table 2.2: Comparison of models on simulated data created based on Figure 2.3. Average log-probability (log-prob) and accuracy rate (acc) are estimated on the test sets.

While statistically significant, the results presented in Table 2.1 might not be significant for some practical purposes. This is mostly due to the simplicity of the hierarchical structure. We repeated the above tests with a more complex hierarchy, shown in Figure 2.3. For this problem we used four covariates randomly generated from the uniform(0, 1) distribution. In all three models, we used the same prior as before for the intercepts. For the MNL model we set $\tau^{-2} \sim \text{Gamma}(1, 1)$. The hyperparameters of treeMNL and corMNL were given $\text{Gamma}(1, 5)$, $\text{Gamma}(1, 20)$ and $\text{Gamma}(1, 100)$ priors for the first, second and third level of the hierarchy respectively.

Table 2.2 shows the average results over 100 datasets for each test. As we can see, the differences between models are more accentuated. When data are generated by other models, corMNL again performs well, being outperformed only by the true model. When data come from treeMNL, the results from corMNL are very close to those of the true

model (i.e., treeMNL), and are actually better for log-probability, though this must of course be due to chance, and is not statistically significant. In contrast, treeMNL’s performance on data generated by corMNL is substantially worse than corMNL’s performance, and is also worse than that of the non-hierarchical MNL model.

2.4 Results for Real Datasets

In this section, we test our approach on two real classification tasks. The first task is labelling the regions of a page using a predefined set of labels. The dataset used for this problem was collected by Laven *et al.* (2005) and is derived from the page images of 58 articles (460 pages) appearing in the proceedings of the Neural Information Processing Systems (NIPS) conference in 2001 and 2002. The second task is classification of documents into different groups based on their contents. For this task we use patent documents released by World Intellectual Property Organization (WIPO). The MATLAB files for MNL, treeMNL and corMNL, along with the NIPS dataset, are available online at <http://www.utstat.utoronto.ca/babak>.

2.4.1 Performance Measures

To compare the performance of different models, we use average log-probability and accuracy rate as described above. We also employ several other measurements including macroaverage F_1 (van Rijsbergen, 1972), parent accuracy, precision, and taxonomy-based loss (Cai and Hoffmann, 2004). F_1 is a common measurement in document labelling 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. The taxonomy-based loss is equal to half the distance between the predicted and the actual class in the tree. Parent accuracy is accuracy after grouping the end nodes with the same parent. While accuracy measurements are based on the top-ranked (based on the posterior predictive probabilities) category only, precision measures the quality of ranking and is defined as follows:

$$precision = \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{|y : P(y|x^{(i)}) \geq P(y^{(i)}|x^{(i)})|} \right)$$

Here, y ranges over all classes and $y^{(i)}$ is the correct class of case i . The denominator is, therefore, the number of classes with equal or higher rank compared to the correct class.

Except for average log-probability and precision, all these measurements require that each test case be assigned to one specific class. For this purpose, we assigned each test case to the end node with the highest posterior predictive probability, as would be optimal for a simple 0/1 loss function. It is, of course, possible to tailor predictions to different performance measures. For example, to improve the parent accuracy, we can still use the 0/1 loss function but make prediction based on the posterior predictive probability of parent nodes. For the taxonomy-based loss, we can predict the class that minimizes the expected distance when a case is misclassified. We tried these tailored predictions, but since the improvements were negligible, we report only the results based on classifying to the end node with highest predictive probability.

To provide a baseline for interpreting the results, for each task we present the performance of a model that ignores the covariates and whose likelihood is solely based on the observed frequency of classes. For this model, we use a vague Dirichlet prior with parameter 1 for all classes. This is a conjugate prior for multinomial parameters. The posterior distribution is also a Dirichlet distribution with parameter $n_j + 1$ for class j . Here, n_j is the frequency of class j in the training set.

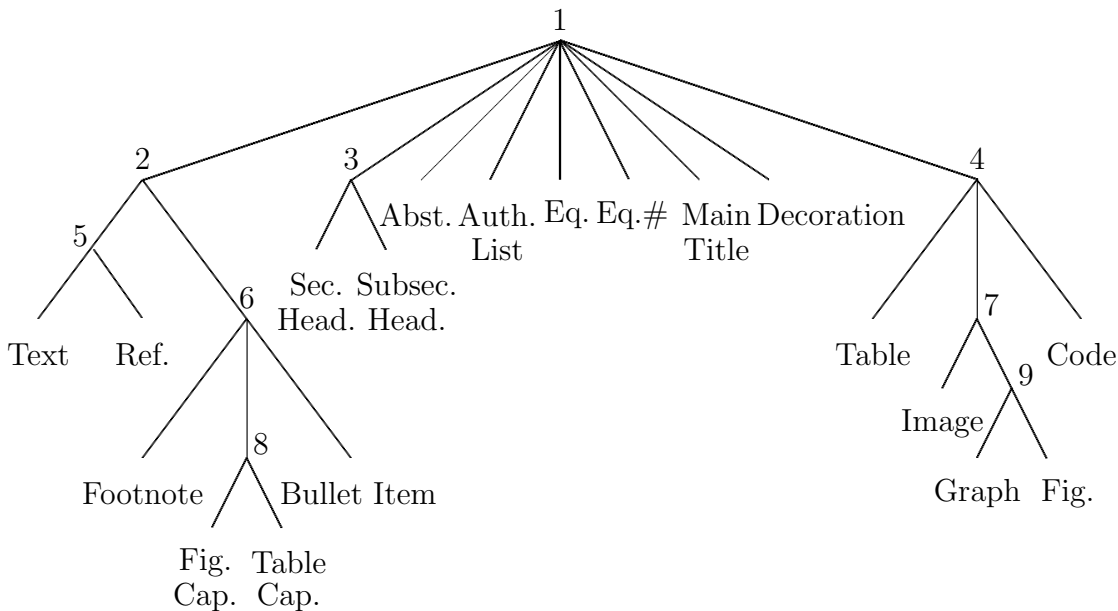


Figure 2.4: Hierarchical structure of document labels.

2.4.2 NIPS Dataset

As mentioned above, the NIPS dataset contains page images of 58 articles. Each page was segmented to several regions, and each region was manually classified to one of 19 possible classes. Figure 2.4 presents these classes in a hierarchical form. The hierarchy is based on our beliefs regarding how difficult it is to separate classes from each other using the available covariates.

The covariates are 59 features such as the location of the region on the page and the density of the ink inside the region. We normalized all features so they have zero mean and standard deviation 1. Laven *et al.* (2005) considered the items from the same article as independent even though they clearly are not. Although this may cause overfitting problems, we follow the same assumption in order to produce comparable results.

Laven *et al.* (2005) divided the dataset into a training set (3445 regions), and a test set (1473 regions). We also trained our three models (MNL, corMNL and treeMNL) on the same training set and evaluated performance on the test set.

The coefficient parameters in the MNL models were given normal priors with mean zero. The variances of these parameters were regarded as hyperparameters. For this problem, since the number of covariates, $p = 59$, is relatively large, we use the Automatic Relevance Determination (ARD) method suggested by Neal (1996). ARD employs a hierarchical prior to determine how relevant each covariate is in classification of objects. In this method, one hyperparameter, σ_l , is used to control the variance of all coefficients, β_{jl} ($j = 1, \dots, c$), for covariate x_l . If a covariate is irrelevant, its hyperparameter will tend to be small, forcing the coefficients for that covariate be near zero. We also use one hyperparameter, τ , to control the overall magnitude of the β 's in the MNL model, so that the the standard deviation of β_{jl} is equal to $\tau\sigma_l$. Therefore, while σ_l controls the relevance of covariate x_l compared to other covariates, τ , controls the overall usefulness of all covariates in separating classes. In detail, the prior for the MNL model was as follows:

$$\begin{aligned} \alpha_j | \eta &\sim N(0, \eta^2) & j = 1, \dots, 19 \\ \beta_{jl} | \tau, \sigma_l &\sim N(0, \tau^2 \sigma_l^2) & l = 1, \dots, 59 \\ \eta^{-2} &\sim \text{Gamma}(0.5, 1) & (0.63, 2.09, 46.31) \\ \tau^{-2} &\sim \text{Gamma}(0.5, 20) & (0.14, 0.47, 10.07) \\ \sigma_l^{-2} &\sim \text{Gamma}(1, 10) & (0.16, 0.38, 1.98) \end{aligned}$$

Similar priors are used for the parameters of treeMNL and corMNL. For these two models, we again used one hyperparameter, $\sigma_l^{-2} \sim \text{Gamma}(1, 10)$ to control all parameters related to covariate x_l . We also used one scale parameter $\tau_m^{-2} \sim \text{Gamma}(0.5, 100)$, with 2.5, 50 and 97.5 percentiles of 0.06, 0.21 and 5.57, for all parameters (β 's in treeMNL, ϕ 's in corMNL) sharing the same node m . The prior for the intercepts was the same as in the MNL model.

We used Hamiltonian dynamics (Neal, 1993) for sampling from the posterior distribution of coefficients. To reduce the random walk aspect of sampling procedure, we use

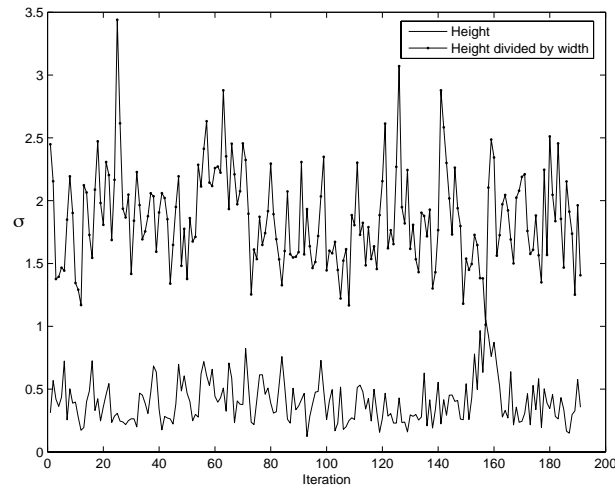


Figure 2.5: Trace plots of the ARD hyperparameters, σ_l , for two covariates of the corMNL model applied to the NIPS dataset.

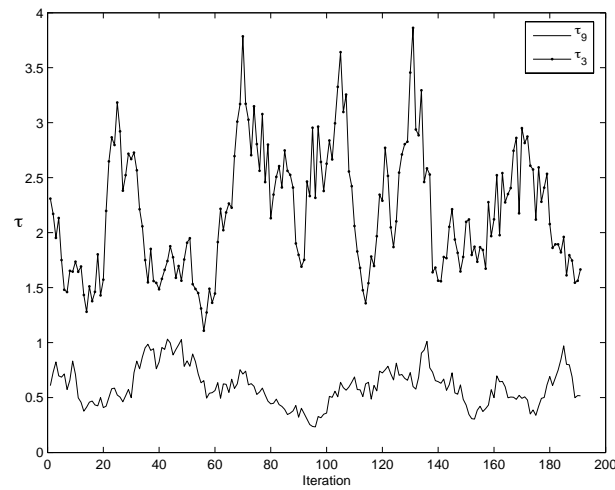


Figure 2.6: Trace plots of two overall scale hyperparameters, τ_m , for two nodes of the corMNL model for two nodes applied to the NIPS dataset.

	log-prob	acc (%)	pacc (%)	precision (%)	F_1 (%)	Δ -loss
Baseline	-2.38	29.4	44.1	47.2	2.4	1.44
LR (ML)	–	88.1	–	–	–	–
MNL	-0.429	88.8	93.0	93.1	74.6	0.22
treeMNL	-0.466	87.2	91.7	92.3	68.9	0.23
corMNL	-0.405	89.5	93.6	93.5	76.0	0.20

Table 2.3: Performance of models based on NIPS dataset. Here, “acc”, “pacc” and “ Δ -loss” refer to accuracy, parent accuracy and taxonomy-based loss respectively. Larger values are better except for Δ -loss.

a reasonably large number of leapfrog steps ($L = 500$). The stepsizes, ϵ 's, are set to 0.02 in order to maintain an acceptance rate of about 90%. In the MNL and corMNL models, new values are proposed for all regression parameters simultaneously. Nested MNL models in treeMNL are updated separately since they are regarded as independent models. The coefficient parameters within each nested model, however, are updated at the same time. Gibbs sampling was used for sampling from the posterior distribution of hyperparameters. Convergence of the Markov chain simulations was assessed by plotting the values of hyperparameters and the average log-likelihood (on training cases). We ran each chain for 2500 iterations, of which the first 500 were discarded. Simulating the Markov chain for 10 iterations took about 5 minutes for MNL, 4 minutes for treeMNL and 9 minutes for corMNL, using a MATLAB implementation on an UltraSPARC III machine.

Table 2.3 compares the results from different models. In this table, we present the logistic regression (LR) model (based on maximum likelihood) developed by Laven *et al.* (2005) as the benchmark. As we can see, the corMNL model outperforms all other models. In contrast, treeMNL performs worse than the non-hierarchical MNL model.

To illustrate the effect of hyperparameter σ in identifying relevant features, in Figure 2.5 we show the trace plots of σ_l for two covariates in the corMNL model. These hyperparameters correspond to two features of a region: “height” and “height divided by

width”. The latter is clearly a more relevant feature for this task. To show the effects of the τ_m hyperparameters, in Figure 2.6 we present the trace plots of τ_3 and τ_9 (i.e., scale parameter of node 3 and node 9 in Figure 2.4). As we can see, the scale parameter in node 3 is large compared to the scale parameter in node 9.

2.4.3 WIPO-alpha Dataset

We also evaluated our models on the WIPO-alpha dataset of patent documents (available at <http://www.wipo.int/ibis/datasets>). These documents are classified according to a standard taxonomy known as the International Patent Classification (available at <http://www.wipo.int/classifications/en/>). The classes in this taxonomy are arranged in a four-level tree structure. At the highest level, documents are divided to 8 sections. For our experiment, we use the documents in section “D” (textile; paper), which has 1710 documents and a total of 160 classes. A pre-processed dataset based on these documents was provided by Cai and Hoffmann (2004). This dataset was generated by indexing the title and claim contents. Document parsing, tokenization, and term normalization were performed using the MindServer retrieval engine. The result is a set of word (i.e., token) counts for each document. Overall, there are 18077 unique words, whose counts are used as covariates. We use a square-root transformation for these covariates in order to emphasize more the occurrence of a word rather than its frequency.

The number of covariates is quite large compared to the number of documents. Cai and Hoffmann (2004) devised a variable selection strategy which provides an optimal solution according to their SVM model. Here, we apply Principal Component Analysis (PCA). We first centred the covariates but did not scale them to have variance one. We then projected the covariates on the 300 principal component directions with the highest variance, and used these 300 principal component scores as covariates for our models, rather than the original covariates. For all models, we use the same priors as discussed in section 4.2, with the exception of ARD hyperparameters, σ_l . For these

hyperparameters, we used $\sigma_l^{-2} \sim \text{Gamma}(2, 1)$, where $l = 1, \dots, 300$ (2.5, 50 and 97.5 percentiles of σ_l are 0.45, 0.78 and 1.98). Compared to the priors used in section 4.2, these priors are more concentrated close to 1. We used these priors to minimize the effect of the ARD hyperparameters since the task of variable selection and relevance determination are mainly performed through PCA. As before, Hamiltonian dynamics ($L = 100$ and $\epsilon = 0.005$) and Gibbs sampling were used for sampling from the posterior distribution of coefficients and hyperparameters respectively. We ran each chain for 3000 iterations and discarded the first 500 iterations. For 10 iterations, the Markov chain simulations took about 6 minutes for MNL, 2 minutes for treeMNL and 10 minutes for corMNL.

Table 2.4 compares the performance of different models. Following Cai and Hoffmann (2004), the results are presented based on a three-fold cross-validation where singular classes (i.e., $n_j = 1$) appear only in the training set. As we can see, the corMNL model outperforms both MNL and treeMNL. Although the results reported for the corMNL model are better than the hierarchical SVM (hSVM) developed by Cai and Hoffmann (2004), the difference could be due to several factors, such as randomness of cross-validation, transformation of variables, or the efficiency of variable selection strategy, as well as the different approach to using the hierarchy. For this problem, treeMNL did by most measures improve on the non-hierarchical MNL model, though it was not as good as corMNL.

2.5 Conclusions and Future Directions

In this paper, we have introduced a new approach for modelling hierarchical classes. Our experiments show that when the hierarchy actually does provide information regarding the similarity of classes, our approach outperforms both the simple MNL model and models based on decomposing the hierarchy into nested MNL models.

	log-prob	acc (%)	pacc (%)	precision (%)	F_1 (%)	Δ -loss
Baseline	-4.492	3.6	13.8	12.1	0.04	2.47
SVM	–	41.8	65.4	52.3	–	1.20
hSVM	–	42.8	69.1	54.4	–	1.08
MNL	-2.622	42.0	66.9	55.1	18.4	1.10
treeMNL	-2.408	42.3	68.7	56.0	17.7	1.05
corMNL	-2.397	43.4	69.8	56.9	18.7	1.02

Table 2.4: Performance of models based on WIPO-alpha dataset, section “D”. Here, “acc”, “pacc” and “ Δ -loss” refer to accuracy, parent accuracy and taxonomy-based loss respectively. The SVM and hierarchical SVM (hSVM) are developed by Cai and Hoffmann (2004). Larger values are better except for Δ -loss.

Our method can be applied to many classification problems where there is prior knowledge regarding the structure of classes. One such problem, which was our original motivation, is annotation of gene function. Using a prior based on the class hierarchy, we have been able to predict gene function with a higher accuracy (Shahbaba and Neal, 2006). For this problem, we used a more elaborate prior for hyperparameters. We also introduced a new method for combining different data sources, which is a common issue in gene function classification.

More experiments are needed to compare corMNL with other approaches to utilizing the hierarchy, such as hSVM (Cai and Hoffmann, 2004). One difference in our approach is that it is based on a probability model for hierarchical prior information, not on any particular hierarchy-based loss function. If a good probability model is used, the probability distributions obtained should provide the information needed to obtain good performance with any loss function.

So far, we have focused only on simple tree-like structures. There are other hierarchical structures that are more complex than a tree. For example, one of the most commonly used gene annotation schemes, known as Gene Ontology (GO), is implemented as a directed acyclic graph (DAG). In this structure a node can have more than one parent.

Our method, as it is, cannot be applied to these problems, but it should be possible to extend the idea of summing coefficients along a path to the class in order to allow for multiple paths.

In our approach, we considered only one structure for each hierarchical classification problem. However, we might sometimes be able to think of more than one possible class hierarchy. It is possible to generalize our method to multiple hierarchies. As for the generalization to DAG's, it should be possible to sum coefficients along the multiple paths within different hierarchies. We can further use a set of hyperparameters to discover the relevance of each hierarchy. If we have prior knowledge that leads us to prefer some structures over others, we can incorporate that knowledge into the priors for these hyperparameters.

Finally, although the results presented in this paper are for linear models, we expect that a similar approach can be used in non-linear models such as neural networks.

Chapter 3

Gene function classification

This chapter appears in *BMC Bioinformatics*, 2006, 7:448.

Abstract

We investigate whether annotation of gene function can be improved using a classification scheme that is aware that functional classes are organized in a hierarchy. The classifiers look at phylogenetic descriptors, sequence based attributes, and predicted secondary structure. We discuss three Bayesian models and compare their performance in terms of predictive accuracy. These models are the ordinary multinomial logit (MNL) model, a hierarchical model based on a set of nested MNL models, and an MNL model with a prior that introduces correlations between the parameters for classes that are nearby in the hierarchy. We also provide a new scheme for combining different sources of information. We use these models to predict the functional class of Open Reading Frames (ORFs) from the *E. coli* genome. The results from all three models show substantial improvement over previous methods, which were based on the C5 decision tree algorithm. The MNL model using a prior based on the hierarchy outperforms both the non-hierarchical MNL model and the nested MNL model. In contrast to previous attempts at combining the three sources of information in this dataset, our new approach to combining data sources produces a higher accuracy rate than applying our models to each data source alone. Together, these results show that gene function can be predicted with higher accuracy than previously achieved, using Bayesian models that incorporate suitable prior information.

3.1 Background

Annotating genes with respect to the function of their proteins is essential for understanding the wealth of genomic information now available. A direct approach to identifying gene function is to eliminate or inhibit expression of a gene and observe any alteration in the phenotype. However, analysis of all genes for all possible functions is not feasible at present. Statistical methods have therefore been employed for this purpose. One

statistical approach attempts to predict the functional class of a gene based on similar sequences for which the function is known. The similarity measures used for this task are produced by computer algorithms that compare the sequence of interest against all other sequences with known function. Two commonly used algorithms are BLAST (Altschul *et al.*, 1997) and FASTA (Pearson and Lipman, 1988).

A problem with using such similarity measures is that a gene's function cannot be predicted when no homologous gene of known function exists. To improve the quality and coverage of prediction, other sources of information can be used. For example, King *et al.* (2001) used a variety of protein sequence descriptors, such as residue frequency and the predicted secondary structure (the structure of hydrogen bonding between different residues within a single polypeptide chain). DeRisi *et al.* (1997), Eisen *et al.* (1998) and Brown *et al.* (2000) used gene expression data, on the assumption that similarly expressed genes are likely to have similar function. Marcotte *et al.* (1999) recommended an alternative sequence-based approach that regards two genes as similar if they are together in another genome. Deng *et al.* (2003) predict the function of genes from their network of physical interactions. To address some of the problems associated with similarity-based methods, such as their non-robustness to variable mutation rates Eisen (1998); Rost (2002), annotation of protein sequences using phylogenetic information has been suggested by some authors (e.g., Eisen *et al.*, 1998; Sjölander, 2004; Engelhardt *et al.*, 2005). In this approach, the evolutionary history of a specific protein, captured by a phylogenetic tree, is used for annotating that protein Eisen *et al.* (1998).

The above-mentioned sources of data can be used separately, or as proposed by several authors (e.g., King *et al.*, 2001; Pavlidis and Weston, 2001; Deng *et al.*, 2004), they can be combined within a predictive model. A variety of statistical and machine learning techniques for making such predictions have been used in functional genomics. These include neighbourhood-count methods (Schoikowski *et al.*, 2000), support vector machines (Brown *et al.*, 2000), and Markov random fields (Deng *et al.*, 2003). A common feature of

these models is that they treat classes as unrelated entities without any specific structure.

The assumption of unrelated classes is not always realistic. As argued by Rison *et al.* (2000), in order to understand the overall mechanism of the whole genome, the functional classes of genes need to be organized according to the biological processes they perform. For this purpose, many functional classification schemes have been proposed for gene products. The first such scheme was recommended by Riley (1993) to catalogue the proteins of *Escherichia coli*. Since then, there have been many attempts to provide a standardized functional annotation scheme with terms that are not limited to certain types of proteins or to specific species. These schemes usually have a hierarchical structure, which starts with very general classes and becomes more specific in lower levels of the hierarchy. In some classification hierarchies, such as the Enzyme Commission (EC) scheme (IUBMB, 1992), levels have semantic values (Rison *et al.*, 2000). For example, the first level of the EC scheme represents the major activities of enzyme like “transferases” or “hydrolases”. In some other schemes, like the ones considered here, the levels do not have any uniform meaning. Instead, each division is specific to the parent nodes. For instance, if the parent includes “metabolism” functions, the child nodes could be the metabolism of “large” or “small” molecules. Rison *et al.* (2000) surveyed a number of these structures and compared them with respect to their resolution (total number of function nodes), depth (potential of the scheme for division into subsets) and breadth (number of nodes at the top level).

All these hierarchies provide additional information that can be incorporated into the classification model. The importance of using the hierarchy in classification models has been emphasized by many authors (e.g., Sattath and Tversky, 1977; Fox, 1997; Koller and Sahami, 1997). One approach for modelling hierarchical classes is to decompose the classification model into nested models, one for each node of the hierarchy. Goodman (2001) showed that using nested models can significantly reduce the training time of maximum entropy-based language models and results in slightly lower perplexities. He

illustrated his approach using a word labelling problem, and recommended that instead of predicting words directly, we first predict the class to which the word belongs, and then predict the word itself. Weigend *et al.* (1999) also used a two-level hierarchical model for document classification. They evaluated their model on the Reuters-22173 corpus and showed significant improvement, especially for rare classes. For text classification, McCallum *et al.* (1998) proposed a hierarchical naive Bayes model that smoothes parameter estimates of a child node by shrinking toward its parents in order to obtain more robust parameter estimates. More recently, new hierarchical classification models based on large margin principles, specifically support vector machines (SVM), have been proposed (Dumais and Chen, 2000; Dekel *et al.*, 2004; Cai and Hoffmann, 2004; Tsochantaridis *et al.*, 2004; Cesa-Bianchi *et al.*, 2006). Dekel *et al.* (2004) introduced a large margin hierarchical classification model that uses the sum of parameters along the tree for classifying cases to the end nodes. These parameters are estimated based on a set of classifiers that assign cases to the intermediate nodes. Cai and Hoffmann (2004) suggested a similar approach based on the generalization of multiclass SVM.

Many approaches to using the hierarchy of gene functions have been proposed. Eisner *et al.* (2005) build multiple binary classifiers with training sets modified according to Gene Ontology (GO). For each classifier associated with a node, they regard a gene as a positive example if it belongs to that node, and as a negative example if it does not belong to the node, or to the node's ancestors and descendants. Barutcuoglu *et al.* (2006) also use a set of independent classifiers, whose predictions are combined using a Bayes network defined based on the GO hierarchy. In the methods recommended by both Eisner *et al.* (2005) and Barutcuoglu *et al.* (2006), the individual classifiers are built independently. Although the classifiers are modified to become consistent, it is more natural to model classes simultaneously. Many authors have shown that learning a set of related tasks at the same time will improve the performance of models (e.g., Caruana, 1997; Zhang *et al.*, 2005). King *et al.* (2001) attempted to use the additional

information from the hierarchical structure of gene functional classes by simply using different decision tree models for each level of the hierarchy. Clare and King (2003) investigated a modified decision tree model, in which assignment of a functional class to a node in the decision tree implies membership of all its parent classes. They evaluated this method based on *Saccharomyces cerevisiae* data and found that the modified version is sometimes better than the non-hierarchical model and sometimes worse. Blockeel *et al.* (2002) suggested an alternative modification of decision trees for hierarchical classification models. Their model uses a distance-based measure, where distances are derived from the hierarchy. Struyf *et al.* (2005) followed the same idea but advocated a different distance measure, which is easier to interpret and is guaranteed to be positive. They evaluated their approach based on different datasets available for *Saccharomyces cerevisiae*, and showed that their model has better precision than the hierarchical C4.5 model proposed by Clare and King (2003).

In a previous paper (Shahbaba and Neal, 2007), we introduced an alternative Bayesian framework for modelling hierarchical classes. This method, henceforth called corMNL, uses a Bayesian form of the multinomial logit model (MNL), with a prior that introduces correlations between the parameters for classes that are nearby in the tree. We also discussed an alternative hierarchical model that uses the hierarchy to define a set of nested multinomial logit models, which we refer to as treeMNL. In this paper, we apply these methods (described further below in the methods section) to the gene function classification problem.

3.2 Results and Discussion

We used our Bayesian MNL, treeMNL and corMNL models to predict the functional class of Open Reading Frames (ORFs) from the *E. coli* genome. *E. coli* is a good organism for testing our method since many of its gene functions have been identified through

direct experiments. We used the pre-processed data provided by King *et al.* (2001). This dataset contains 4289 ORFs identified by Blattner *et al.* (1997). Only 2122 of these ORFs, for which the function was known in 2001, are used in our analysis. The functional hierarchy for these proteins is provided by Riley and Labedan (1996). This hierarchy has three levels, with the most general classes at level 1 and the most specific classes at level 3. For example, lipoate-protein ligase A (lplA) belongs to class ‘Macromolecule metabolism’ at level 1, to class ‘Macromolecule synthesis, modification’ at level 2, and to class ‘Lipoprotein’ at level 3. After excluding categories 0 and 7 at level 1, the data we used had 6 level 1 categories, 20 level 2 categories, and 146 level 3 categories.

Since 2001 many additional gene functions have been determined by direct experiment (see King *et al.*, 2004). However, we use the same dataset as King *et al.* (2001), with the same split of data into the training set (1410 ORFs) and test set (712 ORFs), in order to produce comparable results. King *et al.* (2001) further divided the training set into two subsets and used one subset as validation data to select a subset of rules from those produced by the C5 algorithm based on the other part of the training set. Our Bayesian methods do not require a validation set, so we did not subdivide the training set.

The covariates are based on three different sources of information: phylogenetic descriptors, sequence based attributes, and predicted secondary structure. Following King *et al.* (2001), we refer to these three sources of data as SIM, SEQ and STR respectively. Attributes in SEQ are largely based on composition of residues (i.e., the number of residues of type R) and of pairs of residues (i.e., the number of residue pairs of types R and S) in a sequence. There are 933 such attributes (see Table 1 in King *et al.* (2001)). Information in SIM (see Table 2 in King *et al.* (2001)) and STR (see Table 3 in King *et al.* (2001)) is derived based on a PSI-BLAST (position-specific iterative BLAST) search with parameters $e = 10$, $h = 0.0005$, $j = 20$ from NRProt 05/10/99 database. King *et al.* (2001) used the Inductive Logic Programming (ILP) algorithm known as Warmr (Dehaspe *et al.*, 1998) to produce binary attributes based on the iden-

Accuracy (%)	SEQ			STR			SIM		
	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
Baseline	42.56	21.21	8.15	42.56	21.21	8.15	42.56	21.21	8.15
MNL	60.25	33.99	20.93	50.98	25.14	15.87	69.10	45.79	30.76
treeMNL	59.27	34.13	18.26	52.67	27.39	16.29	67.70	45.93	30.34
corMNL	61.10	35.96	21.21	52.81	27.95	16.71	70.51	47.19	30.90

Table 3.1: Comparison of models based on their predictive accuracy (%) using each data source separately.

tified frequent patterns (1 if the pattern is present and 0 otherwise) in SIM and STR data. The rules created by Warmr and their corresponding attributes can be found at <http://www.aber.ac.uk/~dcswww/Research/bio/ProteinFunction>. There are 13799 such attributes generated for SIM and 18342 attributes for STR. As described below in the methods section, we reduced the dimensionality for each dataset using Principal Component Analysis (PCA). We used 100 components for SEQ, 100 components for STR, and 150 components for SIM.

Table 3.1 compares the three models with respect to their accuracy of prediction at each level of the hierarchy. In this table, level 1 corresponds to the top level of the hierarchy, while level 3 refers to the most detailed classes (i.e., the end nodes). For level 3, we use a simple 0/1 loss function and minimize the expected loss by assigning each test case to the end node with the highest posterior predictive probability. We could use the same predictions for measuring the accuracy at levels 1 and 2, but to improve accuracy, we instead make predictions based on the total posterior predictive probability of nodes at level 1 and level 2.

To provide a baseline for interpreting the results, for each task we present the performance of a model that ignores the covariates and simply assigns genes to the most common category at the given level in the training set.

As we can see in Table 3.1, corMNL outperforms all other models. For the SEQ dataset, MNL performs better than treeMNL. Compared to MNL, the corMNL model

achieves a slightly better accuracy at level 3 and more marked improvements at level 1 and level 2. For the STR dataset, both hierarchical models (i.e., treeMNL and corMNL) outperform the non-hierarchical MNL. For this dataset, corMNL has a slightly better performance than treeMNL. For the SIM dataset, the advantage of using the corMNL model is more apparent in the first and second levels.

For analysing these datasets, King *et al.* (2001) used a decision tree model based on the C5 algorithm. They selected sets of rules that had an accuracy of at least 50% with the coverage of at least two correct examples in the validation set. In Table 3.2, we compare the accuracy of our models to those of King *et al.* (2001). In order to make the results comparable, we used the same coverage values as they used. Coverage is defined as the percentage of test cases for which we make a confident prediction. In a decision tree model, these test cases can be chosen by selecting rules that lead to a specific class with high confidence. For our models, we base confidence on posterior predictive probability, which is defined as the expected probability of each class with regard to the posterior distribution of model parameters. We assign each test case to a class with the maximum posterior predictive probability. The higher this probability, the more confident we are in classifying the case. We rank the test cases based on how high the highest probability is, and for a coverage of g , we classify only the top g percent of genes. In Table 3.2, the coverage values are given in parenthesis. All three of our models discussed here substantially outperform the decision tree model. Overall, corMNL has better performance than MNL and treeMNL.

In an attempt to improve predictive accuracy, King *et al.* (2001) combined the three datasets (SEQ, STR and SIM). Although one would expect to obtain better predictions by combining several sources of information, their results showed no additional benefit compared to using the SIM dataset alone. We also tried combining datasets in order to obtain better results. Initially, we used the principal components which we found individually for each dataset, and kept the number of covariates contributed from each

Accuracy (%)	SEQ			STR			SIM		
	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
	(20)	(18)	(4)	(10)	(1)	(5)	(29)	(26)	(16)
C5	64	63	41	59	44	17	75	74	69
MNL	81	79	88	83	100	67	96	90	84
treeMNL	81	76	70	70	86	69	95	87	84
corMNL	84	82	89	83	100	73	97	90	82

Table 3.2: Comparison of models based on their predictive accuracy (%) for specific coverage (%) provided in parenthesis. The C5 results and the coverage values are from King *et al.* (2001).

data source the same as before. Principal components from each dataset were scaled so that the standard deviation of the first principal component was 1. We did this to make the scale of variables from different data sources comparable while preserving the relative importance of principal components within a dataset.

Using the combined dataset, all our models provided better predictions, although the improvement was only marginal for some predictions. We speculated that some of the covariates may become redundant after combining the data (i.e., are providing the same information). One may often obtain better results by removing redundancy and reducing the number of covariates. To examine this idea, we kept the number of principal components from SIM as before (i.e., 150) but only used the first 25 principal components from SEQ and STR. The total number of covariates was therefore 200. Reducing the number of covariates from SEQ and STR may also prevent them from overwhelming the covariates from SIM, which is the most useful single source. This strategy led to even higher accuracy rates compared to when we used the SIM dataset alone. The results are shown in Table 3.3 (middle section). It is worth noting that when SEQ and STR are used alone, using 25 principal components (rather than 100 before) results in lower accuracy (results not shown).

To improve the models even further, we tried an alternative strategy in which different

Accuracy (%)	SIM only			Combined dataset single scale parameter			Combined dataset separate scale parameters		
	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
MNL	69.10	45.79	30.76	69.66	48.88	32.02	70.65	49.16	33.71
treeMNL	67.70	45.93	30.34	68.26	46.63	30.34	68.82	46.63	31.74
corMNL	70.51	47.19	30.90	71.49	49.30	32.87	72.75	49.16	34.41

Table 3.3: Accuracy (%) of models on the combined dataset with and without separate scale parameters. Results using SIM alone are provided for comparison.

Accuracy (%)	Coverage (%)					
	5	10	20	50	90	100
Level 1	100	98	96	92	76	73
Level 2	100	98	96	71	53	49
Level 3	100	97	80	52	36	34

Table 3.4: Predictive accuracy (%) for different coverage values (%) of the corMNL model using all three sources with separate scale parameters.

sources of data are combined such that their relative weights are automatically adjusted. As we can see in Table 3.3 (right section), this strategy, which is described in more detail in the methodology section, resulted in further improvements in the performance of the models. We also examined this approach with larger numbers of covariates. We found that when we increased the number of principal components for SEQ and STR back to the original 100, the accuracy of predictions mostly remained the same, though a few dropped slightly.

In practice, we might be most interested in genes whose function can be predicted with high confidence. There is a trade-off between predictive accuracy and the percentage of the genes we select for prediction (i.e., coverage). Table 3.4 shows this trade-off for results on the test set from the corMNL model applied to the combined dataset. In this table, the accuracy rates for different coverage values are provided. As we can see, our model can almost perfectly classify 10% of the genes in the test set.

Finally, we trained the corMNL model on all ORFs with known function to annotate the function of unknown ORFs. Many of these ORFs, whose function was previously unknown, have been recently annotated using direct biological experiments. However, since the functional ontology of *E. coli* genome (provided by the Riley group) has changed over time, it is not possible to compare our results directly. King *et al.* (2004) also faced this problem. They evaluated their predictions manually for a subset of ORFs. We present our predictions for the same set of ORFs (see Appendix C). We use the MultiFun Classification System (<http://genprotoc.mbl.edu/>) to obtain the function(s) associated with each ORF through direct experiments.

For many of these ORFs, our prediction is closely related to the confirmed function. For example, we classified *yojH* (b2210) hierarchically as “Metabolism of small molecules” at the first level, “Degradation of small molecules” at the second level, and “Carbon compounds” at the third level. Through a direct experiment, the function of this gene was classified as “Metabolism”, “Energy metabolism (carbon)”, and “Tricarboxylic acid cycle”, at levels 1, 2, and 3 respectively. In some cases, such as *ybhO* (b0789), there is an exact match between our prediction (Macromolecule metabolism : Macromolecule synthesis : Phospholipids) and the function provided by MultiFun (Metabolism : Macromolecule (cellular constituent) biosynthesis : Phospholipid). For some other cases, although our prediction does not exactly match the functions provided by MultiFun Classification System, the results are comparable up to the first or second level of the hierarchy. For example, we predicted that *ydeD* (b1533) is “Transport/binding proteins” and belongs to “ABC superfamily”. Direct experiment also show that in fact this gene does belongs to the “Transport” group, however, it is more specifically in the “Major Facilitator Superfamily” (MFS) class instead of ABC. The comparison of our predictions with MultiFun Classification System is not always as straightforward as the examples provided above. For instance, we predicted *bfd* (b3337) to be in the “Metabolism of small molecules : Energy metabolism, carbon : Anaerobic respiration” categories. The

hierarchical function provided by MultiFun is “Cell processes : Adaptation to stress : Fe acquisition”. At the first glance, these two seem to be unrelated. However, it is known that oxygen induces stress and results in enormous changes in *E. coli* (Spiro and Guest, 1991; Guest *et al.*, 1996). *E. coli* adapts to this environmental change by switching from aerobic respiration (which is its preferred metabolic mode) to anaerobic respiration. More detailed examination of these predictions will be needed to definitively evaluate performance.

Our predictions for ORFs of unknown function (in 2001) are available online at <http://www.utstat.utoronto.ca/~babak>. In this website, we also provide the combined dataset for *E. coli*, and the MATLAB programs for MNL, treeMNL and corMNL along with their respective outputs for the test set.

3.3 Conclusions

In this paper, we investigated the use of hierarchical classification schemes to perform functional annotation of genes. If the hierarchy provides any information regarding the structure of gene function, we would expect this additional information to lead to better prediction of classes. To examine this idea, we compared three Bayesian models: a non-hierarchical MNL model, a hierarchical model based on nested MNL, referred to as treeMNL, and our new corMNL model, which is a form of the multinomial logit model with a prior that introduces correlations between the parameters of nearby classes. We found corMNL provided better predictions in most cases. Moreover, we introduced a new approach for combining different sources of data. In this method, we use separate scale parameters for each data source in order to allow their corresponding coefficients have appropriately different variances. This approach provided better predictions compared to other methods.

While our emphasis in this paper was on the importance of using hierarchical schemes

in gene classification, we also showed that even the non-hierarchical Bayesian MNL model outperforms previous methods that used the C5 algorithm. Overall, our results are encouraging for the prospect of accurate gene function annotation, and also illustrate the utility of a Bayesian approach with problem-specific priors. For our experiments, we used the pre-processed datasets provided by King *et al.* (2001), who used the Warmr (Dehaspe *et al.*, 1998) algorithm to generate binary attributes. It is conceivable that the accuracy of predictions can be further improved by using other data processing methods. Similarly, it is possible that a method other than our use of PCA might be better for reducing dimensionality before doing classification.

In the *E. coli* dataset we used here, each ORF was assigned to only one function. In the more recent classification system provided by Riley’s group (<http://genprotec.mbl.edu/>), ORFs may belong to more than one class. For such problems, one can modify the likelihood part of the models described here so that if a training case belongs to several classes, its contribution to the likelihood is calculated based on the sum of probabilities of those classes.

The functional hierarchies considered here are simple tree-like structures. There are other hierarchical structures that are more complex than a tree. For example, one of the most commonly used gene annotation schemes, known as Gene Ontology (GO), is implemented as a directed acyclic graph (DAG). In this structure a node can have more than one parent. Our method, as it is, cannot be applied to these problems, but it should be possible to extend the idea of summing coefficients along the path to the class in order to allow for multiple paths.

Our approach can also be generalized to problems where the relationship among classes can be described by more than one hierarchical structure. For these problems, different hyperparameters can be used for each hierarchy and predictions can be made by summing the parameters in branches from all these hierarchies.

3.4 Methods

In this section, we first explain our models using a simple hierarchy for illustration. Consider Figure 3.1, which shows a hierarchical classification problem with four classes. By ignoring the hierarchy, a simple multinomial logit (MNL) can be used for classifying cases to one of the four classes. If the class for a case is denoted by y , and the covariates for this case are x , then the MNL model is

$$P(y = j|x, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\exp(\alpha_j + x\boldsymbol{\beta}_j)}{\sum_{j'=1}^4 \exp(\alpha_{j'} + x\boldsymbol{\beta}_{j'})}$$

For each class j (for $j = 1, \dots, 4$), there is an intercept α_j and a vector of p unknown parameters $\boldsymbol{\beta}_j$, where p is the number of covariates in x . The inner product of these parameters with the covariate vector is shown as $x\boldsymbol{\beta}_j$.

Alternatively, we can use the hierarchy to decompose the classification model into nested models (e.g., MNL). For example, in Figure 3.1, class 1 can be modeled as the product of two independent MNL models:

$$P(y = 1|x) = P(y \in \{1, 2\}|x) \times P(y \in \{1\}|y \in \{1, 2\}, x)$$

We refer to models in which the tree structure is used to define a set of nested MNL models as treeMNL.

For modelling hierarchical classes, we propose a Bayesian MNL with a prior that introduces correlations between the parameters of nearby classes. Our model, called corMNL, includes a vector of parameters, $\boldsymbol{\phi}$, for each branch in the hierarchy (Figure 3.1). We assign 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. This way, we can better handle situations in which these classes are hard to distinguish. In Figure 3.1, parameter vectors denoted as $\boldsymbol{\phi}_{11}$ and $\boldsymbol{\phi}_{12}$ are associated with branches in the first

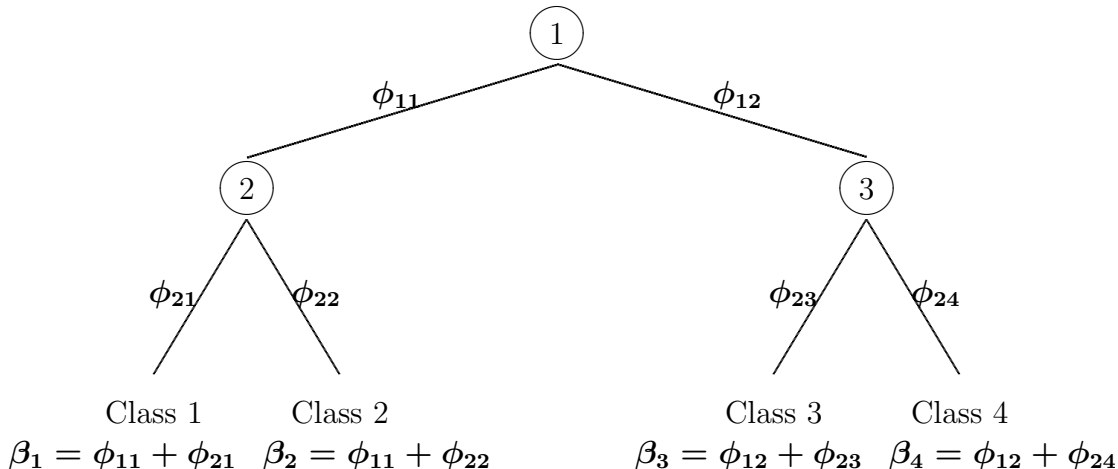


Figure 3.1: A simple representation of the corMNL model.

level, and ϕ_{21} , ϕ_{22} , ϕ_{31} and ϕ_{32} with branches in the second level. We assign objects to one of the end nodes using an MNL model with regression coefficients $\beta_1 = \phi_{11} + \phi_{21}$, $\beta_2 = \phi_{11} + \phi_{22}$, $\beta_3 = \phi_{12} + \phi_{31}$ and $\beta_4 = \phi_{12} + \phi_{32}$ for classes 1, 2, 3 and 4 respectively. Note that the intercept parameters, α_j , are not treated hierarchically.

We first used these models (i.e., MNL, treeMNL and corMNL) to predict gene function using each data source (SIM, STR and SEQ) separately. Since the numbers of covariates in these datasets are large, we applied Principal Component Analysis (PCA). Prior to applying PCA, the variables were centred to have mean zero, but they were not rescaled to have variance one. We selected the first p components with the highest eigenvalues. The cut-off, p , was set based on the plot of eigenvalues against PCs (i.e., the scree plot). Since there was not a clear cut-off point at which the magnitude of eigenvalues drops sharply, the plots could only help us to narrow down the appropriate values for p . We decided to choose a value at the upper end of the range suggested by the scree plot. We selected 100 components from SEQ, 100 components from STR, and 150 components from SIM.

Principal components are derived solely based on the input space and do not necessarily provide the best set of variables for predicting the response variable. In order to find

the relevant variables (among the principal components) for the classification task, we use the Automatic Relevance Determination (ARD) method suggested by Neal (1996). ARD employs a hierarchical prior to determine how relevant each covariate is to classification. In the MNL model, for example, one hyperparameter, σ_l , is used to control the variance of all coefficients, β_{jl} ($j = 1, \dots, c$), 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. We also use a set of hyperparameters, τ_j , to control the magnitude of the β 's for each class. We use a third 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, τ_j controls the usefulness of covariates in identifying class j , and ξ controls the overall usefulness of all covariates in separating all classes. The standard deviation of β_{jl} is therefore equal to $\xi\tau_j\sigma_l$.

For the MNL model we used the following priors:

$$\begin{aligned}\alpha_j|\eta &\sim N(0, \eta^2) \\ \beta_{jl}|\xi, \sigma_l, \tau_j &\sim N(0, \xi^2\tau_j^2\sigma_l^2) \\ \log(\eta) &\sim N(0, 1) \\ \log(\xi) &\sim N(-3, 2^2) \\ \log(\tau_j) &\sim N(-1, 0.5^2) \\ \log(\sigma_l) &\sim N(0, 0.3^2)\end{aligned}$$

Since the task of variable selection is mainly performed through PCA, the ARD hyperparameters, σ 's, are given priors with fairly small standard deviation. The priors for τ 's are set such that both small values (i.e., close to zero) and large values (i.e., close to 1) are possible. The overall scale of these hyperparameters is controlled by ξ , which has a broader prior. Note that since these hyperparameters are used only in the combination $\xi\tau_j\sigma_l$, only the sum of the means for $\log(\xi)$, $\log(\tau_j)$, and $\log(\sigma_l)$ really matters.

Similar priors are used for the parameters of treeMNL and corMNL. For these two

models, we again used one hyperparameter, σ_l , to control all parameters (β 's in treeMNL, ϕ 's in corMNL) related to covariate x_l . We also used one scale parameter τ_k for all parameters related to branch k of the hierarchy. The overall scale of all parameters is controlled by one hyperparameter ξ . When we combine different sources of information, we sometimes used separate scale parameters, ξ , for each data source. This allows the coefficients from different sources of data to have appropriately different variances in the model. This is additional to what ARD hyperparameters provide.

The setting of priors described in this paper is different from what we used in a previous paper (Shahbaba and Neal, 2007), where we used one hyperparameter to control all the coefficients (regardless of their corresponding class) in the MNL model, and we used one hyperparameter to control the parameters of all the branches that share the same node in treeMNL and corMNL. The scheme used in this paper provides an additional flexibility to control β 's. In this paper, the hyperparameters are given log-normal distributions instead of the gamma distributions used in Shahbaba and Neal (2007). Using gamma priors has the advantage of conjugacy and, therefore, easier MCMC sampling. However, we prefer log-normal distributions since they are more convenient for formalizing our prior beliefs.

3.4.1 Implementation

These models are implemented using Markov chain Monte Carlo (Neal, 1993). We use Hamiltonian dynamics (Neal, 1993) for sampling from the posterior distribution of coefficients (with hyperparameters temporarily fixed). The number of leapfrog steps was set to 50. The stepsizes were set dynamically at each iteration, based on the current values of the hyperparameters (Neal, 1996). In the MNL and corMNL models, new values are proposed for all regression parameters simultaneously. Nested MNL models in treeMNL are updated separately since they are regarded as independent models. The coefficient parameters within each nested model, however, are updated at the same time.

We use single-variable slice sampling (Neal, 2003) to sample from the posterior distribution of hyperparameters. At each iteration, we use the “stepping out” procedure to find the interval around the current point and the “shrinkage” procedure for sampling from the interval. The initial values of the ARD hyperparameters, σ 's, were set to the inverse of the standard deviation of their corresponding covariates. The initial values of τ 's and ξ were set to 1.

Convergence of the Markov chain simulations was assessed from trace plots of hyperparameters. We ran each chain for 5000 iterations, of which the first 1000 were discarded. Simulating the Markov chain for 10 iterations took about 2 minutes for MNL, 1 minute for treeMNL, and 3 minutes for corMNL, using a MATLAB implementation on an UltraSPARC III machine.

Chapter 4

Nonlinear classification using Dirichlet process mixtures

This chapter will be published as a technical report in the department of Statistics,
University of Toronto.

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.

4.1 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, \dots, 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:

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

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):

$$\begin{aligned} y_i | c_i, \phi &\sim F(\phi_{c_i}) \\ c_i | p_1, \dots, p_C &\sim \text{Discrete}(p_1, \dots, p_C) \\ p_1, \dots, p_C &\sim \text{Dirichlet}(\gamma/C, \dots, \gamma/C) \\ \phi_c &\sim G_0 \end{aligned} \tag{4.1}$$

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

$$P(c_i = c | c_1, \dots, c_{i-1}) = \frac{n_{ic} + \gamma/C}{i - 1 + \gamma} \tag{4.2}$$

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 (4.2) reach the following limits:

$$\begin{aligned} P(c_i = c | c_1, \dots, c_{i-1}) &\rightarrow \frac{n_{ic}}{i-1+\gamma} \\ P(c_i \neq c_j \forall j < i | c_1, \dots, c_{i-1}) &\rightarrow \frac{\gamma}{i-1+\gamma} \end{aligned} \quad (4.3)$$

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

$$\theta_i | \theta_1, \dots, \theta_{i-1} \sim \frac{1}{i-1+\gamma} \sum_{j < i} \delta(\theta_j) + \frac{\gamma}{i-1+\gamma} G_0 \quad (4.4)$$

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:

$$\begin{aligned} y_i | \theta_i &\sim F(\theta_i) \\ \theta_i | G &\sim G \\ G &\sim \mathcal{D}(G_0, \gamma) \end{aligned} \quad (4.5)$$

That is, the limit of the finite mixture model (4.1) is equivalent to the Dirichlet process mixture model (4.5) 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 4.3, we illustrate our approach and evaluate its performance based on simulated data. In Section 4.4, 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 4.5, 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 4.6 shows how this approach can be used for multiple sources of information. Finally, Section 4.7 is devoted to discussion, future directions and limitations of the proposed method.

4.2 Methodology

Consider a classification problem with continuous covariates, $x = (x_1, \dots, x_p)$, 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:

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

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

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

Here, the parameters $\mu = (\mu_1, \dots, \mu_p)$ and $\sigma = (\sigma_1, \dots, \sigma_p)$ are the means and standard deviations of covariates in each component. The component index, c , is omitted for simplicity. Within a component, $\alpha = (\alpha_1, \dots, \alpha_J)$, and $\beta = (\beta_1, \dots, \beta_J)$ are the parameters of the multinomial logit (MNL) model, and J is the number of classes. The entire set of regression coefficients, β , can be presented as a $p \times J$ matrix. This representation is redundant, since one of the β_j 's (where $j = 1, \dots, J$) 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:

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

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 (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 4.1, 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.

4.3 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 $l = 1, \dots, p$ and $k = 1, \dots, p$), 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

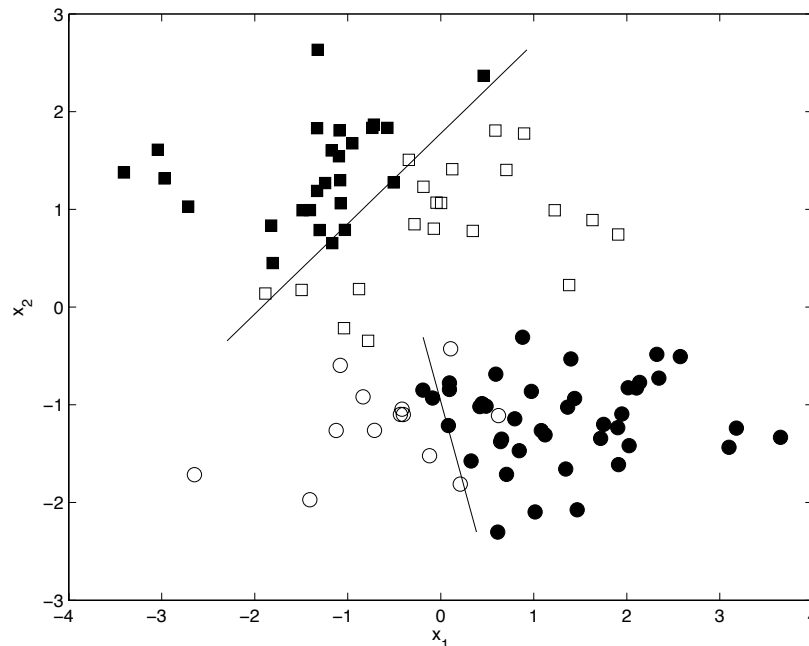


Figure 4.1: An illustration of our model for a binary (black and white) classification problem with two covariates. Here, the mixture has two components, which are shown with circles and squares. In each component, an MNL model separates the two classes into “black” or “white” with a linear decision boundary.

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 $l = 1, \dots, 5$ and $j = 1, \dots, 4$. From the above baseline prior, we sample two components, θ_1 and θ_2 , where $\theta = (\mu, \sigma, \eta, \nu, \alpha, \beta)$. 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:

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

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 4.1: 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 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 , 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 4.1. 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

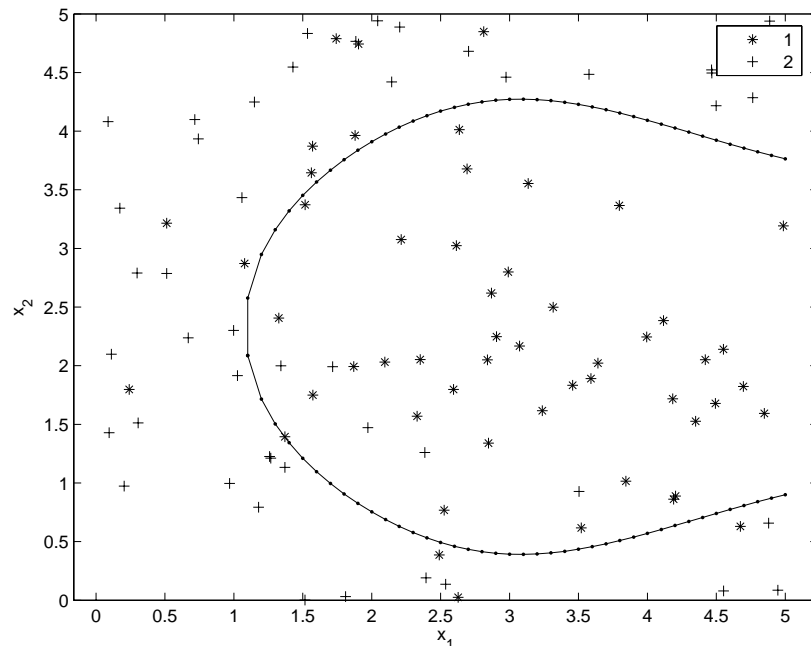


Figure 4.2: A random sample generated according to Simulation 2 with $a_3 = 0$. The dotted line is the optimal boundary function.

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

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

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

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 4.2: 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.

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 4.2. 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.

4.4 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 non-linear 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 4.3.

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 $j = 1, \dots, J$), 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 $j = 1, \dots, J$ indexes classes, and $c = 1, \dots, C$ 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 $l = 1, \dots, p$. For these parameters, we used the following priors:

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

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, $\sigma_{0,l}^2$, 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 4.3. 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.

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

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.

4.5 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 4.3 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

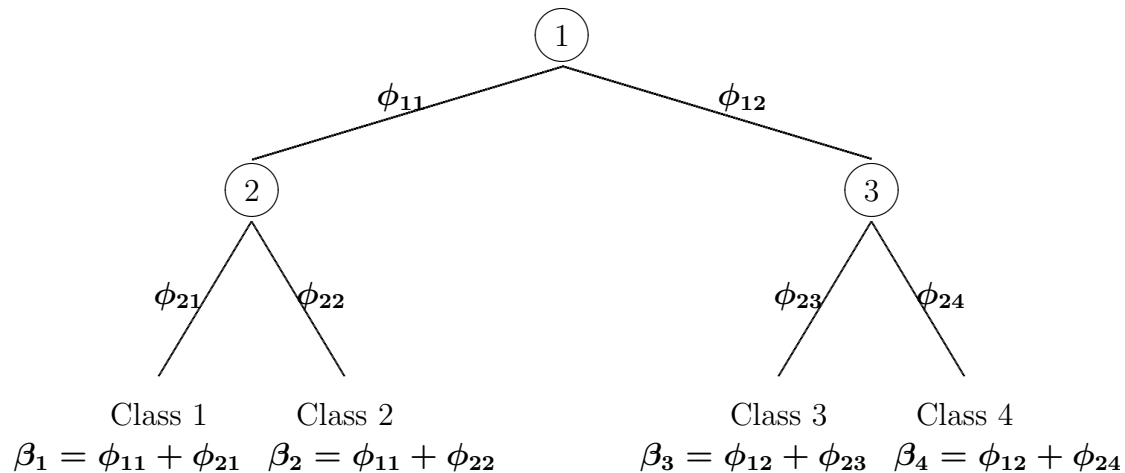


Figure 4.3: A simple representation of our hierarchical classification model.

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 result is a nonlinear model which takes the hierarchical structure of classes into account. We refer to these models as dpCorMNL.

Table 4.4 presents the results for the two linear models (with and without hierarchy-based 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 4.4: Comparison of hierarchical models (linear and nonlinear) with non-hierarchical models (linear and nonlinear) based on protein fold classification data.

4.6 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 van 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

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 4.5: 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.

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 4.5. 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.

4.7 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 alterna-

tive 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 $x \sim \text{Bernoulli}(\mu)$, 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 (μ_1, \dots, μ_K) , where K is the number of categories in x , or use K continuous latent variables, z_1, \dots, 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 multi-

nomial. 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.

Chapter 5

Overall discussion

In this thesis, we showed how classification models can be improved using Bayesian methods that incorporate suitable prior information. Mainly, we discussed classification models where classes have a hierarchical structure. This hierarchy reflects our prior opinion regarding the similarity of classes. Using several simulation studies and real datasets, we showed that if such information is used properly, it can result in a higher prediction accuracy.

One specific problem, namely, gene function classification, was of main interest. We used our approach to predict the cellular function of genes in the *E. coli* genome. Our model provided a substantially higher predictive accuracy compared to previous methods based on C5 decision trees models. We also presented a new approach for combining different sources of information for gene function classification.

Having useful prior information can help to improve the results. However, it is also important to use appropriate models. Some of the common assumptions in statistics, such as normality of distributions and linearity of dependencies, may result in a poor performance (i.e., when predicting the response value for new cases). In this thesis, we proposed a more flexible model based on Dirichlet process mixtures. Using simulation studies we showed that our model can provide better results compared to some alternative linear models and several nonlinear ones. We also discussed the application of our model for protein fold identification, which is an important step in predicting cellular function of genes, and found that our model provided better predictive accuracy compared to previous models based on Neural Networks and Support Vector Machines.

Our methods can be applied to a variety of problems, of which only few were discussed here. Another possible application is cancer prediction using microarray gene expression data. Our dpMNL model could be used to improve predictive accuracies. Cancer types can sometimes be arranged on a hierarchy starting with general types and dividing each type to its sub-groups. Our corMNL and dpCorMNL models could therefore be used for these problems. Moreover, the information from microarray analysis can be combined

with other sources of information. For such situations, our approach for combining multiple data sources could be used.

The methods discussed in this thesis can be extended in several ways. For example, we only discussed simple tree-like hierarchies. Our approach for summing parameters along the hierarchy, presented in Chapter 2, can be easily extended to more complex structures such as Directed Acyclic Graphs (DAGs). The idea of summing coefficients along a path to the class can be generalized in order to allow for multiple paths. Similarly, we can also generalize our method to multiple hierarchies. That is, we can sum coefficients along the paths (single or multiple) within different hierarchies.

The hierarchical classification model introduced in this thesis was mainly intended to improve prediction in classification models. We believe our corMNL model could also be used for inference regarding our choice of hierarchy. The posterior distribution obtained from the corMNL model might provide insight on whether the assumed hierarchy was appropriate. For example, consider the simple hierarchy presented in Figure 2.2. If the posterior distribution of ϕ_{11} is tightly concentrated around zero (i.e., the associate variance hyperparameter is small), the posterior correlation between the parameters of Class 1 and Class 2 will be small, which in turn means that the similarity of these two classes assumed in prior might not exist in reality.

We extended our nonlinear model (based on Dirichlet process mixtures) to classification problems where a class hierarchy exists. Alternatively, we could modify other nonlinear models (e.g., Neural Networks, Support Vector Machines, and Gaussian process models) to relax the linearity assumption in our corMNL model. For example, consider a Gaussian process model for non-hierarchical classes. As suggested by Neal (1998), such model can be defined using J “latent values”, z_{i1}, \dots, z_{iJ} , each associated with one class. Class probabilities for this model can be defined as follows:

$$P(y_i = j) = \frac{\exp(z_{ij})}{\sum_{j'=1}^J \exp(z_{ij'})}$$

To extend this model to hierarchical classes, we can use a different latent value for each

branch of the hierarchy, and represent the final latent value for each class by the sum of the latent values for all the branches leading to that class.

The nonlinear method presented here 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. For example, while we might identify a large number of genes from an organism's genome, the biological function might be known only for a small subset of those genes. Supervised learning methods use only those genes for which the function is known. Semi-supervised methods, on the other hand, use all genes. This way, the unlabeled part of the data might provide additional information, which could be useful for the classification task.

Although, we focused on classification problems with hierarchical classes, some of the proposed methods can be used beyond hierarchical classification models. For example, our approach for combining multiple datasets can be used for regression models, where the response variable is continuous. Moreover, the nonlinear model introduced in Chapter 4 can be easily extended to all generalized linear models. To do this, we need to replace MNL with a more appropriate model for the response variable. For example, if the response variable measures counts, we can use a Poisson model to capture the dependency between the response and the covariates within each component.

Appendix A

In Chapters 2 and 3, we discussed an alternative hierarchical classification model, referred to as treeMNL, in which the tree structure is decomposed into nested MNL models. The resulting nested models are statistically independent, conditioned on the upper levels. The likelihood can therefore be written as the product of the likelihoods for all models. Here, we discuss this model in more detail, and show how it differs from the ordinary MNL model.

Consider a three-way classification problem in which classes 1 and 2 are hard to distinguish on the basis of available information, but they both can be easily separated from class 3. We can model these classes using two nested MNL models. The first model combines class 1 and 2, and classifies cases to class $\{1, 2\}$ or class 3. Since this is a binary split, we can use logistic regression, which is a special form MNL. The second model attempts to separate class 1 from class 2, knowing that cases belong to either 1 or 2. These two models are therefore as follows:

$$\begin{aligned} P(y = 3|x, \alpha_1, \beta_1) &= \frac{\exp(\alpha_1 + x\beta_1)}{1 + \exp(\alpha_1 + x\beta_1)} \\ P(y = 2|y \in \{1, 2\}, x, \alpha_2, \beta_2) &= \frac{\exp(\alpha_2 + x\beta_2)}{1 + \exp(\alpha_2 + x\beta_2)} \end{aligned}$$

From the first model, we have

$$P(y \in \{1, 2\}|x, \alpha_1, \beta_1) = \frac{1}{1 + \exp(\alpha_1 + x\beta_1)}$$

Therefore, the final probabilities for class 1 and class 2 are as follows:

$$P(y = 1|x, \alpha_1, \beta_1, \alpha_2, \beta_2) = \frac{1}{1 + \exp(\alpha_1 + x\beta_1)} \times \frac{1}{1 + \exp(\alpha_2 + x\beta_2)}$$

$$P(y = 2|x, \alpha_1, \beta_1, \alpha_2, \beta_2) = \frac{1}{1 + \exp(\alpha_1 + x\beta_1)} \times \frac{\exp(\alpha_2 + x\beta_2)}{1 + \exp(\alpha_2 + x\beta_2)}$$

In the ordinary MNL model, the boundary lines between classes are linear. Figure A.1 shows these boundaries for a simulated sample. In the treeMNL model above, the boundary line between class 1 and class 2 is also linear, and is defined by $\alpha_2 + x\beta_2 = 0$. However, the boundary lines that separate class 3 from class 1 and class 2 will not be linear in general. These lines are defined according to the following equations:

$$d_{13} : \alpha_1 + x\beta_1 + \log\{1 + \exp(\alpha_2 + x\beta_2)\} = 0$$

$$d_{23} : (\alpha_1 - \alpha_2) + x(\beta_1 - \beta_2) + \log\{1 + \exp(\alpha_2 + x\beta_2)\} = 0$$

where d_{13} represents the boundary line between class 1 and class 3, and d_{23} represents the boundary line between class 2 and class 3. Figure A.2 shows these boundaries for the the same simulated data as in Figure A.1.

Note that the difference between the corMNL model introduced in this thesis and the ordinary MNL is only in their priors. That is, the corMNL model has the same form as the the MNL model. Therefore, similar to the ordinary MNL model, the decision boundaries for the corMNL model are linear.

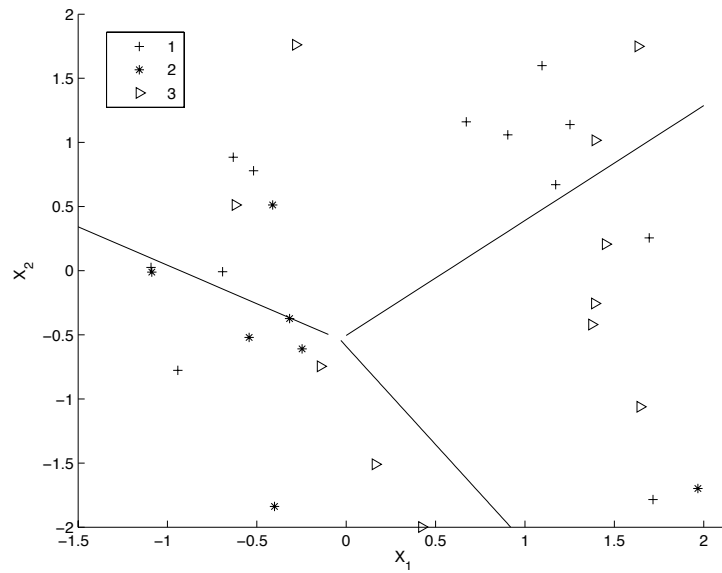


Figure A.1: The boundary lines obtained by the MNL model for a simulated three-way classification problem. As we can see, class 1 and class 2 are relatively more difficult to distinguish.

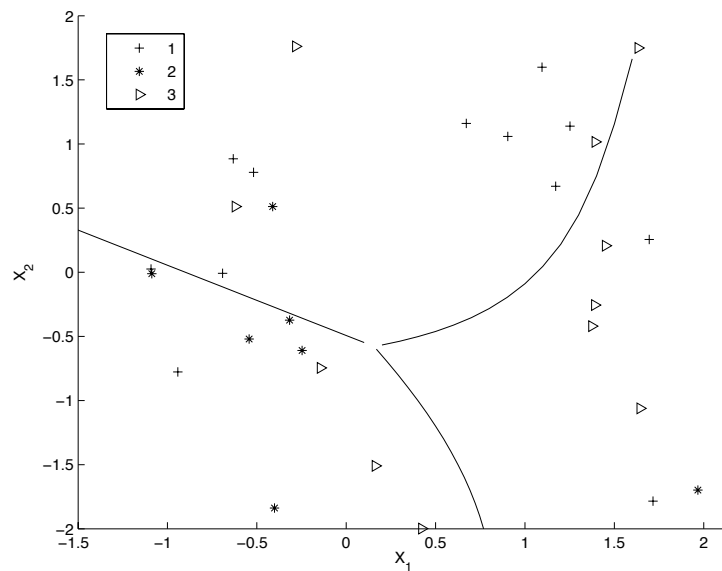


Figure A.2: The boundary lines obtained by the treeMNL model for a simulated three-way classification problem. The boundaries are not linear between classes 1 and 3, and between classes 2 and 3.

Appendix B

Throughout this thesis, we used a variety of computational techniques in order to obtain posterior distributions. For simple distributions, where a conjugate prior is available, posterior distributions have a closed form, and can be used directly for inference. However, for more complex problems, such as those discussed in this paper, it may not be possible to obtain a closed form for posterior distributions. A general approach for such problems is to use a *Markov chain* to obtain samples which come (approximately) from the posterior distribution, π , and use the *Monte Carlo* method for estimation. The Markov chain must have π as its invariant distribution, and must converge (asymptotically) to π . To ensure convergence, the chain needs to be ergodic and converge quickly. This requires the use of appropriate transition probabilities (or densities), $T(\theta'|\theta)$ (i.e., from θ to θ'), for the Markov chain. Here, we describe several Markov chain Monte Carlo (MCMC) algorithms that are used in this thesis, and explain our strategies for choosing appropriate transition probabilities.

B.1 The Gibbs sampler

Although it might not be possible to use a conjugate prior for a multidimensional posterior distribution, it is sometimes possible to assume a conjugate prior for a subset of parameters conditional on the remaining parameters. The Gibbs sampler (Geman and Geman, 1984) is a very useful technique for such problems. We first need to

divide the parameter vector, θ , into smaller components, $\theta = (\theta_1, \dots, \theta_d)$. At each iteration, s , the Gibbs sampler cycles through these components, and samples from $\pi(\theta_j | y, \theta_1^s, \dots, \theta_{j-1}^s, \theta_{j+1}^{s-1}, \dots, \theta_d^{s-1})$, the conditional distribution of θ_j given the current value of θ_k for $k \neq j$ (where $j = 1, \dots, d$). Here, y is the observed data.

We use normal priors for the regression parameters in all models discussed in this thesis. The variances of these normals are regarded as hyperparameters. In Chapter 2, we assumed inverse-Gamma distributions as the prior for these hyperparameters. Inverse-Gamma in this case is a conjugate prior (conditional on other parameters), so the conditional posterior distribution has itself an inverse-Gamma distribution. For example, if $\beta_i \sim N(0, \sigma^2)$, where $i = 1, \dots, n$, and $\sigma^{-2} \sim \text{Gamma}(a, b)$, we have

$$\begin{aligned} \sigma^{-2} | a, b, \beta_1, \dots, \beta_n &\sim \text{Gamma}(a_n, b_n) \\ a_n &= a + \frac{n}{2} \\ b_n &= \frac{1}{1/b + \sum_{i=1}^n \beta_i^2 / 2} \end{aligned}$$

Note that throughout this thesis we use the parameterization of the Gamma distribution in which $\text{Gamma}(a, b)$ has density $f(x|a, b) = [b^a \Gamma(a)]^{-1} x^{a-1} e^{-x/b}$, for which the mean is ab and the standard deviation is $a^{1/2}b$

B.2 The Metropolis algorithm

The Metropolis algorithm (Metropolis *et al.*, 1953) provides a general method for Markov chain sampling. In this approach, given the current state of the chain, θ , a new candidate, θ^* , is proposed according to some probabilities $S(\theta^*|\theta)$ such that the symmetry condition $S(\theta|\theta^*) = S(\theta^*|\theta)$ is satisfied. We accept this new candidate as the next state (i.e., $\theta' = \theta^*$) with probability

$$\min [1, \pi(\theta^*)/\pi(\theta)]$$

If we do not accept θ^* , we set $\theta' = \theta$.

A useful generalization of this approach, known as the Metropolis-Hastings (MH) algorithm (Hastings, 1970), relaxes the symmetry requirement for the transitions. The acceptance probability is instead as follows:

$$\min \left[1, \frac{\pi(\theta^*)/S(\theta^*|\theta)}{\pi(\theta)/S(\theta|\theta^*)} \right]$$

A common transition probability is $N(\theta, \delta^2)$. δ is selected such that a good convergence rate is achieved. Large values for δ may result in low probabilities of acceptance, whereas small values can slow down the convergence rate. Neal (1993) has a thorough discussion on how to improve the convergence rate of Metropolis methods. In this thesis, we use one specific variation of the MH algorithm based on the Hamiltonian dynamics, which avoids the random walk behavior with simple $N(\theta, \delta^2)$ proposals. To use this method, we need to evaluate the gradient of the target posterior distribution with respect to parameters of interest. The new proposal is obtained by taking a series of steps in directions indicated by the gradient. This way, the Markov chain finds states with higher probability more rapidly.

In physics, the Hamiltonian, H , represents the overall energy of a physical system as the sum of potential and kinetic energy. Similarly, we define the Hamiltonian for a parameter θ as follows:

$$H(\theta, p) = E(\theta) + K(p)$$

where $E(\theta)$ is the negative log-probability (or differs from it up to a constant), and $K(p)$ is the kinetic energy, which is a function of a momentum variable p . The momentum variable is independent of θ with standard normal distribution, and is included to augment the state space θ . Note that θ and p can be vectors in general.

The algorithm involves two main steps. First, an initial momentum is drawn randomly from the standard normal distribution. Second, the state of the Markov chain, θ , is changed according to this momentum (i.e., $\Delta\theta = \epsilon p$, where ϵ is a small “stepsize” parameter), and the gradient of $E(x)$ determines how the momentum p will change (i.e.,

$\Delta p = \epsilon \frac{-\partial E(\theta)}{\partial \theta}$). The algorithm alternates between these two steps for a predefined number of iterations until it reaches a new state, (θ^*, p^*) . This state is accepted or rejected according to the Metropolis acceptance probability. We discard the momentum variables at the end of each iteration, and keep only the new states for θ . For a more detail description of this approach see Neal (1993) or MacKay (2003).

We used Hamiltonian dynamics for sampling from the posterior distribution of regression coefficients in all Bayesian models discussed in this thesis. The gradient of the posterior distribution with respect to regression parameters can be easily obtained. We used a sufficiently large numbers of steps to propose new states that are far from the current state. However, it is essential to use an appropriate step size so that such proposed states are accepted with high probability, but also do not take too much time to compute. Although the step size can be the same for all parameters. high efficiency can be obtained if the step size is considered a function of the current variance for each coefficient (Neal, 1993). For example, if $\beta_j \sim N(0, \sigma_j^2)$, and $[\sigma_j^{(s)}]^2$ is the current value of the variance for the j^{th} coefficient at iteration s , we set:

$$\epsilon_j = \frac{\epsilon_0}{\sqrt{1/[\sigma_j^{(s)}]^2 + n/4}}$$

Here, ϵ_0 is a small positive number which remains constant for all coefficients. To find an appropriate value for this parameter, we perform several preliminary runs with different values for ϵ_0 , and choose the value that provides around 90% acceptance rate.

B.3 Slice sampling

Slice sampling, introduced by Neal (2003), is another approach to improve the efficiency of the Metropolis algorithms. This method is based on the idea in order to sample from a distribution, we can sample uniformly from the region under the corresponding density function, $f(\theta)$. For this purpose, Neal (2003) suggested to alternate between two steps. Given the current state of the Markov chain, θ , we uniformly sample a new point, z , from

the interval $(0, f(\theta))$. Next, given the current value of z , we uniformly sample from the region $S = \{\theta : z < f(\theta)\}$, which is referred to as the “slice” defined by z . Since sampling an independent point uniformly from S might be difficult, we can substitute this step by any update that leaves the uniform distribution over S invariant. Neal (2003) proposed several methods to perform this task. Here, we use the “stepping-out” procedure for finding an interval around the current point, and use the “shrinkage” procedure for sampling from the interval obtained. For a detail description of these methods see Neal (2003).

In this thesis, we used the slice sampling method for the simulation studies in Chapter 2, since the number of parameters was small. We also used slice sampling to obtain samples from the posterior distribution of hyperparameters, when non-conjugate priors (e.g., log-Normal distribution for variances of regression parameters) were used. To use slice sampling efficiently, we need to choose an appropriate step size, w , in the stepping-out procedure, and the maximum number of steps, m . The size of a slice is limited to mw . However, unlike the Metropolis method, the choice of step size is not very critical to the rate of progress, since in slice sampling the step size is self-tuning MacKay (2003). We chose w and m such that the amount of time the algorithm spends on stepping out and shrinkage to the right interval size is small.

Appendix C

In this appendix, we compare the gene annotation results based on our corMNL model and the results from direct biological experiments for a subset of ORFs. This subset was selected by King *et al.* (2001). For each ORF, we provide its Blattner number, predicted hierarchical class (based on the older hierarchy of *E. coli*), and the corresponding class labels in the first line. The subsequent lines show the recent annotation of each ORF based on direct experiment. Here, SE = “Some Evidence” and NE = “No Evidence”.

Evidence	bNumber	Class	Class description
SE	b0805	4.1.3	Structural elements > Cell envelop > Outer membrane constituents Cell structure > Membrane Location of gene products > Outer membrane
SE	b1519	3.2.8	Metabolism of small molecules > Biosynthesis of cofactors > Menaquinone, ubiquinone Metabolism > Central intermediary metabolism > Unassigned reversible reactions
SE	b1533	1.5.2	Cell processes > Transport/binding proteins > ABC superfamily (membrane) Transport > Electrochemical potential driven transporters > Porters > MFS Cell structure > Membrane
SE	b1981	1.5.21	Cell processes > Transport/bindingproteins > MFS Transport > Electrochemical potential driven transporters > Porters > MFS
SE	b2210	3.4.3	Metabolism of small molecules > Degradation of small molecules > Carbon compounds Metabolism > Energy metabolism (carbon) > Tricarboxylic acid cycle
SE	b3839	1.5.2	Cell processes > Transport/binding proteins > ABC superfamily (membrane) Transport > Cell Substrate transported > Protein Cell structure > Membrane
SE	b1822	2.2.1	Macromolecule metabolism > Macromolecule synthesis, modification > Amino acyl tRNA syn Information transfer > RNA related > Modification
SE	b3223	3.4.3	Metabolism of small molecules > Degradation of small molecules > Carbon compounds Metabolism > Central intermediary metabolism > Amino sugar conversions
SE	b3337	3.5.2	Metabolism of small molecules > Energy metabolism, carbon > Anaerobic respiration Cell processes > Adaptation to stress > Fe aquisition
SE	b3569	3.4.3	Metabolism of small molecules > Degradation of small molecules > Carbon compounds Metabolism > Carbon compound utilization > Carbohydrate degradation
SE	b3955	4.1.3	Structural elements > Cell envelop > Outer membrane constituents Cell structure > Membrane
SE	b3222	3.4.3	Metabolism of small molecules > Degradation of small molecules > Carbon compounds Metabolism > Central intermediary metabolism > Amino sugar conversions
SE	b0570	6.1.1	Global functions > Global regulatory functions Regulation > Type of regulation > Transcriptional level
SE	b0619	6.1.1	Global functions > Global regulatory functions Regulation > Type of regulation > Transcriptional level
SE	b2219	6.1.1	Global functions > Global regulatory functions Regulation > Type of regulation > Transcriptional level
SE	b0505	3.3.15	Metabolism of small molecules > Central metabolism > Conversions of intermed. met-m Metabolism > Central intermediary metabolism > Allantoin assimilation
SE	b0508	3.4.3	Metabolism of small molecules > Degradation of small molecules > Carbon compounds Metabolism > Central intermediary metabolism
SE	b0662	3.5.2	Metabolism of small molecules > Energy metabolism, carbon > Anaerobic respiration Metabolism > Energy metabolism (carbon) > Aerobic respiration
SE	b0789	2.2.7	Macromolecule metabolism > Macromolecule synthesis > Phospholipids Metabolism > Macromolecule (cellular constituent) biosynthesis > Phospholipid
SE	b2924	4.1.2	Structural elements > Cell envelop > Murein sacculus, peptidoglycan Cell structure > Membrane
SE	b2052	3.3.18	Metabolism of small molecules > Central metabolism > Sugar-nucleotide biosynthesis Metabolism > Macromolecule biosynthesis > Colanic acid (M antigen)
SE	b2889	2.2.3	Macromolecule metabolism > Macromolecule synthesis > DNA - replication, repair Metabolism > Building block biosynthesis > Cofactor, small molecule carrier biosynthesis

Table C.1: Comparison of direct functional annotation of several ORFs (whose function was unknown in 2001) with predicted functions using our corMNL model. In this table, we only present genes for which there is a close match between our predictions and the results from direct biological experiments.

Evidence	bNumber	Class	Class description
NE	b2392	3.5.2	Metabolism of small molecules > Energy metabolism, carbon > Anaerobic respiration Transport > Substrate transported > Mn ⁺ /H
NE	b0103	1.5.1	Cell processes > Transport/binding proteins > ABC superfamily (atp-bind) Metabolism > Building block biosynthesis > Small molecule carrier > Coenzyme A
NE	b2530	3.3.15	Metabolism of small molecules > Central metabolism > Conversions of intermed. met-m Information transfer > Protein related > Posttranslational modification
NE	b0162	3.5.2	Metabolism of small molecules > Energy metabolism, carbon > Anaerobic respiration Regulation > Genetic unit regulated > Regulon
NE	b0613	3.4.3	Metabolism of small molecules > Degradation of small molecules > Carbon compounds Information transfer -> Protein related
NE	b2972	3.4.3	Metabolism of small molecules > Degradation of small molecules > Carbon compounds Information transfer > Protein related > Export, signal peptide cleavage
NE	b0053	2.1.1	Macromolecule metabolism > Macromolecule degradation > Degradation of DNA Information transfer > Protein related > Chaperone, folding
NE	b0441	1.7.1	Cell processes > Cell division Information transfer > Protein related > Chaperone, folding
NE	b1199	1.5.23	Cell processes > Transport/binding proteins > Mechanism not stated Metabolism > Central intermediary metabolism > Unassigned reversible reactions
NE	b3836	4.2.2	Structural elements > Ribosome constituents > Ribosomal proteins - synthesis, modification Cell structure > Membrane Location of gene products > Inner membrane
NE	b3838	5.1.2	Extrachromosomal > Laterally acquired elements > Phage-related functions and prophages Cell structure > Membrane Location of gene products > Inner membrane

Table C.2: Comparison of direct functional annotation of several ORFs (whose function was unknown in 2001) with predicted functions using our corMNL model. In this table, we only present genes for which there is not a close match between our predictions and the results from direct biological experiments.

Appendix D

Chapter 2 of this dissertation is published in Bayesian Analysis (<http://ba.stat.cmu.edu>), and Chapter 3 is published in BMC Bioinformatics (<http://www.biomedcentral.com/bmcbioinformatics>). According to the copyright and license rules of both these journals, the authors are the copyright holders, and have the right to use, reproduce or disseminate their article. In this appendix, we provide the published version of our papers.

Bibliography

- Agresti A (2002). *Categorical Data Analysis*. John Willey and Son, Hoboken, New Jersey.
- Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ (1997). “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.” *Nucleic Acids Research*, **25**, 3389–3402.
- Antoniak CE (1974). “Mixture of Dirichlet process with applications to Bayesian non-parametric problems.” *Annals of Statistics*, **273(5281)**, 1152–1174.
- Barutcuoglu Z, Schapire RE, Troyanskaya OG (2006). “Hierarchical multi-label prediction of gene function.” *Bioinformatics*, **22**, 830–836.
- Blackwell D, MacQueen JB (1973). “Ferguson distributions via Polya urn scheme.” *Annals of Statistics*, **1**, 353–355.
- Blattner FR, Plunkett Gr, Bloch CA, Perna NT, Burland V, Riley M, Collado-Vides J, Glasner JD, Rode CK, Mayhew GF, Gregor J, Davis NW, Kirkpatrick HA, Goeden M, Rose DJ, Mau B, Shao Y (1997). “The complete genome sequence of *Escherichia coli* K-12.” *Science*, **277**, 1453–1474.
- Blockeel H, Bruynooghe M, Dzeroski S, Ramon J, Struyf J (2002). “Hierarchical multi-classification with predictive clustering trees in functional genomics.” In “Proceedings of the ACM SIGKDD 2002 Workshop on Multi-Relational Data Mining (MRDM 2002),” pp. 21–35.

- Breiman L, Friedman J, Olshen RA, Stone CJ (1993). *Classification and Regression Trees*. Chapman and Hall, Boca Raton.
- Brown M, Nobel GW, Lin D, Cristianini N, Walsh SC, Furey T, Ares MJ, Haussler D (2000). “Knowledge-based analysis of microarray gene expression data by using support vector machines.” *Proceedings of the National Academy of Sciences (USA)*, **97**(1), 262–267.
- 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.
- Cai L, Hoffmann T (2004). “Hierarchical document categorization with Support Vector Machines.” *ACM 13th Conference on Information and Knowledge Management*.
- Caruana R (1997). “Multitask Learning.” *Machine Learning*, **28**, 41–75.
- Cesa-Bianchi N, Gentile C, Zaniboni L (2006). “Incremental Algorithms for Hierarchical Classification.” *Journal of Machine Learning Research*, **7**, 31–54.
- Clare A, King RD (2003). “Predicting gene function in *Saccharomyces cerevisiae*.” In “Proceedings of the European Conference on Computational Biology (ECCB 2003), September 27-30, 2003, Paris, France,” pp. 42–49.
- 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.
- Dehaspe L, Toivonen H, King RD (1998). “Finding frequent substructures in chemical compounds.” In R Agrawl, P Stolorez, G Piatetsky-Shapiro (eds.), “Proceedings of

- the Fourth International Conference on Knowledge Discovery and Data Mining,” pp. 30–36. AAAI Press, Menlo Park, CA.
- Dekel O, Keshet J, Singer Y (2004). “Large margin hierarchical classification.” In “Proceedings of the 21st International Conference on Machine Learning (ICML),” .
- Deng M, Chen T, Sun F (2004). “An integrated probabilistic model for functional prediction of proteins.” *Journal of Computational Biology*, **11**(2-3), 463–475.
- Deng M, Zhang K, Mehta S, Chen T, Sun F (2003). “Prediction of protein function using protein-protein interaction data.” *Journal of Computational Biology*, **10**(6), 947–960.
- DeRisi J, Iyer V, Brown P (1997). “Exploring the metabolic and genetic control of gene expression on a genomic scale.” *Science*, **278**, 680–686.
- Ding C, Dubchak I (2001). “Multi-class protein fold recognition 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.
- Dumais ST, Chen H (2000). “Hierarchical classification of Web content.” In “Proceedings of the 23rd ACM International Conference on Research and Development in Information Retrieval (SIGIR),” pp. 256–263.
- Eisen JA (1998). “Phylogenomics: Improving functional prediction for uncharacterized genes by evolutionary analysis.” *Genome Research*, **8**, 163–167.
- Eisen M, Spellman P, Brown P, Botstein D (1998). “Cluster analysis and display of genome-wide expression patterns.” *Proceedings of the National Academy of Sciences (USA)*, **95**, 14863–14868.

- Eisner R, Poulin B, Szafron D, Lu P, R G (2005). "Improving Protein Function Prediction using the Hierarchical Structure of the Gene Ontology." *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*.
- Engelhardt BE, Jordan MI, Muratore KE, Brenner SE (2005). "Protein molecular function prediction by Bayesian phylogenomics." *PLoS Computational Biology*, **1**(5), 432–445.
- Escobar MD, West M (1995). "Bayesian density estimation and inference using mixtures." *Journal of American Statistical Society*, **90**, 577–588.
- Ferguson TS (1973). "A Bayesian analysis of some nonparametric problems." *Annals of Statistics*, **1**, 209–230.
- Fox J (1997). *Applied Regression Analysis, Linear Models and Related Methods*. Sage.
- Geman S, Geman D (1984). "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721–741.
- Goodman J (2001). "Classes for fast maximum entropy training." *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *IEEE press*.
- Guest JR, Green J, Irvine AS, Spiro S (1996). *The FNR modulon and FNR-regulated gene expression*, pp. 317–342. in Regulation of gene expression in Escherichia coli (Lin, E.C.C. and Lynch, A.S. Eds) R.G. Landes Co., Austin, Texas.
- Hastings WK (1970). "Monte Carlo sampling methods using Markov chains and their applications." *Biometrika*, **57**(1), 97–109.
- 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.
- IUBMB (1992). *Enzyme Nomenclature: Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology*. Academic Press, New York.
- Jain S, Neal RM (2007). “Splitting and merging components of a nonconjugate Dirichlet process mixture model.” *to appear in Bayesian Analysis*.
- King RD, Karwath A, Clare A, Dehaspe L (2001). “The utility of different representations of protein sequence for predicting functional class.” *Bioinformatics*, **17**(5), 445–454.
- King RD, Wise PH, Clare A (2004). “Confirmation of data mining based predictions of protein function.” *Bioinformatics*, **20**, 1110–1118.
- Koller D, Sahami M (1997). “Hierarchically classifying documents using very few words.” In “Proceedings of the 14th International Conference on Machine Learning,” .
- Laven K, Leishman S, Roweis S (2005). “A statistical learning approach to document image analysis.” *Conference on Document Analysis and Recognition (ICDAR), Seoul, South Korea*.
- 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 Graphical Statistics*, **7**, 223–238.
- MacKay DJC (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

- Marcotte EM, Pellegrini M, Ng HL, Rice DW, Yeates TO, Eisenberg D (1999). “Detecting protein function and protein-protein interactions from genome sequences.” *Science*, **285**, 751–753.
- McCallum A, Rosenfeld R, Mitchell T, A N (1998). “Improving text classification by shrinkage in a hierarchy of classes.” In “Proceedings of the International Conference on Machine Learning (ICML),” pp. 359–360.
- McFadden D (1980). “Econometric models for probabilistic choice among products.” *Journal of Business*, **53**, 13–36.
- Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953). “Equation of state calculations by fast computing machines.” *Journal of Chemical Physics*, **21**, 1087–1092.
- Mitchell TM (1998). “Conditions for the equivalence of hierarchical and flat Bayesian classifiers.” URL <http://www.cs.cmu.edu/~tom/hierproof.ps>.
- Müller P, Erkanli A, West M (1996). “Bayesian curve fitting using multivariate normal 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 (1998). “Regression and classification using Gaussian process priors.” *Bayesian Statistics*, **6**, 471–501.
- 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.
- Pavlidis P, Weston J (2001). "Gene functional classification from heterogeneous data." *Proceedings of the 5th International Conference on Computational Molecular Biology (RECOMB)*, pp. 249–255.
- Pearson WR, Lipman DJ (1988). "Improved tools for biological sequence comparison." *Proceedings of the National Academy of Sciences (USA)*, **85**, 2444–2448.
- Riley M (1993). "Functions of the gene products of *Escherichia coli*." *Microbiology Review*, **57**, 862–952.
- Riley M, Labedan B (1996). "E. coli gene products: physiological functions and common ancestries." In FN Neidhardt, RI Curtiss, ECC Lin, JL Ingraham, KB Low, B Magasanik, W Reznikoff, M Riley, M Schaechter, E Umbarger (eds.), "Escherichia coli and Salmonella: cellular and molecular biology, 2nd edition," ASM Press, Washington, DC.
- Rison S, Hodgman TC, Thornton JM (2000). "Comparison of functional annotation schemes for genomes." *Functional and Integrative Genomics*, **1**, 56–69.
- Rost B (2002). "Enzyme function less conserved than anticipated." *Journal of Molecular Biology*, **318**, 595–608.
- Sattath S, Tversky A (1977). "Additive similarity trees." *Psychometrika*, **42**, 319–345.
- Schoikowski B, Uetz P, Fields S (2000). "A network of protein-protein interaction in yeast." *Nature Biotechnology*, **18**, 1257–1261.
- 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.
- Sjölander K (2004). “Phylogenomics inference of protein molecular function: Advances and challenges.” *Bioinformatics*, **20**, 170–179.
- Spiro S, Guest JR (1991). “Adaptive responses to oxygen limitation in *Escherichia coli*.” *Trends in Biochemical Sciences*, **16**(8), 310–314.
- Struyf J, Dzeroski S, Blockeel H, Clare A (2005). “Hierarchical multi-classification with predictive clustering trees in functional genomics.” In “Proceedings (Bento, C. and Cardoso, A. and Dias, G., eds.), vol 3808, Lecture Notes in Computer Science,” pp. 272–283.
- Tsochantaridis I, Hoffmann T, Joachims T, Altum Y (2004). “Support Vector Machine learning for independent and structured output spaces.” *Proceedings of the 21st International Conference on Machine Learning (ICML)*.
- van Rijsbergen CJ (1972). *Automatic Information Structuring and Retrieval*. Ph.D. thesis, King’s College, Cambridge.
- Weigend AS, Wiener ED, Pedersen JO (1999). “Exploiting hierarchy in text categorization.” *Information Retrieval*, **1**(3), 193–216.
- Zhang J, Ghahramani Z, Yang Y (2005). “Learning Multiple Related Tasks using Latent Independent Component Analysis.” In “Proceedings of NIPS 2005, Vancouver, Canada,” .