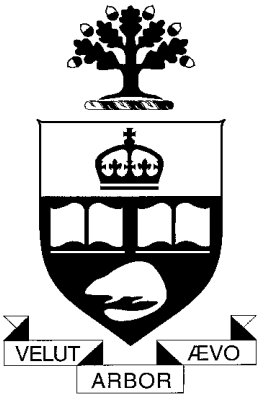

Improving Classification When a Class Hierarchy is Available Using a Hierarchy-Based Prior

by

Babak Shahbaba
Department Public Health Sciences
University of Toronto

and

Radford M. Neal
Department of Statistics
University of Toronto

Technical Report No. 0510 October 18, 2005

TECHNICAL REPORT SERIES

University of Toronto
Department of Statistics

Technical Report No. 0510, Department of Statistics, University of Toronto

Improving Classification When a Class Hierarchy is Available Using a Hierarchy-Based Prior

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

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

18 October 2005

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 different way. We also test the new method on a document labelling problem, and find that it performs better than the other methods, particularly when the amount of training data is small.

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.

One example of a hierarchical classification scheme is the annotations describing the biological function of genes. Functions are usually presented in a hierarchical form starting with very general classes and becoming more specific in lower levels of the hierarchy. There are many different gene annotation schemes. Figure 1 shows a small part of the scheme proposed by Rison [1].

Another example of hierarchical classification is the document labelling problem. This involves classification of document regions to one of many predefined classes. It is possible to arrange these classes in a hierarchy such as that of Figure 4 below. We arranged these classes in a hierarchical form based on our guess as to how easily they can be distinguished. For instance, we considered "Figure Caption" and "Table Caption" similar to each other since they both usually contain small amounts of straight text and both are centre justified.

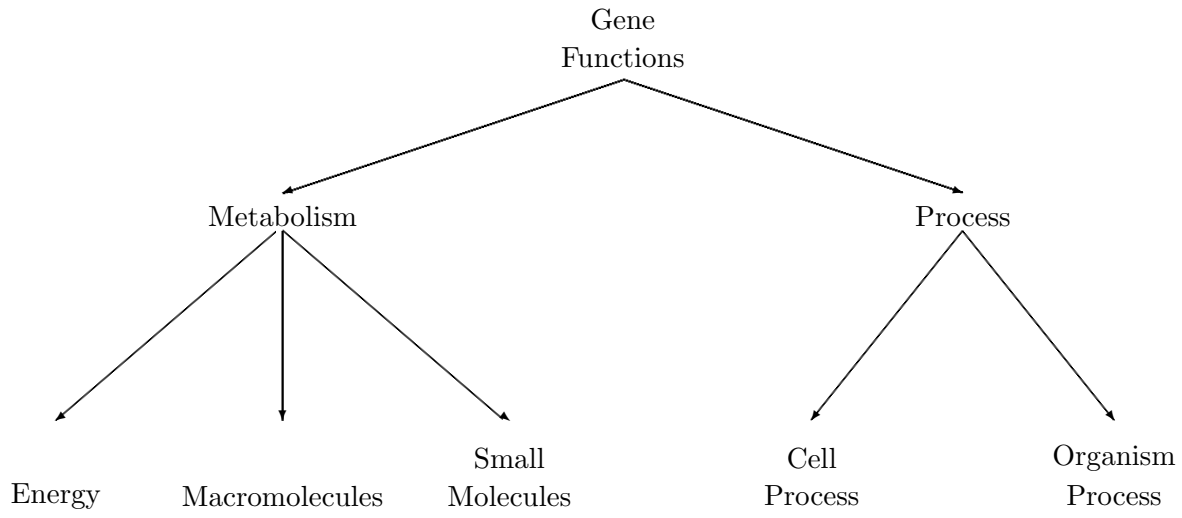

Figure 1: A part of a gene annotation hierarchy.

Bayesian models provide a framework that allows us to incorporate prior knowledge of this sort. The prior distribution in Bayesian models represents our belief (as well as our uncertainty) regarding likely values of parameters. For modelling hierarchical classes, we introduce a new method which is based on a Bayesian form of the multinomial logit (MNL) model. We use a prior that introduces correlations between the parameters of nearby classes.

This paper is organized as folllows. In section 2, simple classification models and their extensions for analysing hierarchical classes are discussed. In section 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 4 we compare the same models on a document labelling problem. The last section summarizes our findings and presents some ideas for future research.

2 Hierarchical classification

Consider a classification problem in which we have observed data for n cases, $(x^{(1)}, y^{(1)})$, ...,$(x^{(n)}, y^{(n)})$, where $x^{(i)} = x_1^{(i)}, ..., x_p^{(i)}$ is the vector of p covariates (features) for case i, and $y^{(i)}$ is the associated class. Our goal is to develop a model that assigns each data item to its correct class based on the observed covariates. The resulting model will be used 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:

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

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:

$$P(y = j|x, \boldsymbol{\alpha}, \boldsymbol{\beta}) \quad = \quad \frac{\exp(\alpha_j + x\boldsymbol{\beta}_j)}{\sum_{j=1}^{c} \exp(\alpha_j + x\boldsymbol{\beta}_j)} \tag{2}$$

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 $\boldsymbol{\beta} = \boldsymbol{\beta_1, ..., \beta_c}$ can be presented as a $p \times c$ matrix. This representation is redundant, since one of the β_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:

$$\begin{aligned} \alpha_j|\tau_0 &\sim N(0, \tau_0^2) \\ \beta_{jl}|\tau &\sim N(0, \tau^2) \\ \tau_0^{-2} &\sim Gamma(a_0, b_0) \\ \tau^{-2} &\sim Gamma(a, b) \end{aligned}$$

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

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 [2].

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., [3, 4]) 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 [5] 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 can first predict the class the word belongs to, and then predict the word itself. This approach can be applied to the other learning techniques.

Fox [6] suggested using successive binary partitions (i.e., dichotomies) of classes and fitting a logistic model to each partition. In other words, we recursively split a set of classes into two mutually exclusive subsets where each subset contains similar classes. In Figure 2, these partitions are {12, 34}, {1, 2}, and {3, 4}. The resulting nested binary models are statistically independent. The likelihood can therefore be written as the product of the likelihoods for each of the binary models. For example, in Figure 2 we have

$$P(y = 1|x) \quad = \quad P(y \in \{1, 2\}|x) \times P(y \in \{1\}|y \in \{1, 2\}, x) \tag{3}$$

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 children nodes. The portion of the nested MNL model for this node has the following form:

$$\begin{aligned} P(y=k|x,\boldsymbol{\alpha}_m,\boldsymbol{\beta}_m) &= \frac{\exp(\alpha_{mk}+x\boldsymbol{\beta}_{mk})}{\sum_{k=1}^{c_m}\exp(\alpha_{mk}+x\boldsymbol{\beta}_{mk})} \\ \alpha_{mk}|\tau_0 &\sim N(0,\tau_0^2) \\ \beta_{mkl}|\tau_m &\sim N(0,\tau_m^2) \\ \tau_0^{-2} &\sim Gamma(a_0,b_0) \\ \tau_m^{-2} &\sim Gamma(a_m,b_m) \end{aligned}$$

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

We introduce an alternative Bayesian framework for modelling hierarchical classes. Consider Figure 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, these parameters are denoted as $\boldsymbol{\phi_{11}}$ and $\boldsymbol{\phi_{12}}$ for branches in the first level and $\boldsymbol{\phi_{21}}$, $\boldsymbol{\phi_{22}}$, $\boldsymbol{\phi_{23}}$ and $\boldsymbol{\phi_{24}}$ for branches in the second level. We assign objects to one of the end nodes using a MNL model (equation 2) whose regression coefficients for class j are represented by the sum of parameters on all the branches leading to that class. In Figure 2, these coefficients are $\boldsymbol{\beta_1}=\boldsymbol{\phi_{11}}+\boldsymbol{\phi_{21}}$, $\boldsymbol{\beta_2}=\boldsymbol{\phi_{11}}+\boldsymbol{\phi_{22}}$, $\boldsymbol{\beta_3}=\boldsymbol{\phi_{12}}+\boldsymbol{\phi_{31}}$ and $\boldsymbol{\beta_4}=\boldsymbol{\phi_{12}}+\boldsymbol{\phi_{32}}$ for classes $1,2,3$ and 4 respectively. Sharing the common terms, $\boldsymbol{\phi_{11}}$ and $\boldsymbol{\phi_{12}}$, introduces prior correlation between the parameters of nearby classes in the hierarchy.

In our model, henceforth called corMNL, $\boldsymbol{\phi}$'s are vectors with the same size as $\boldsymbol{\beta}$'s. We assume all the components of the $\boldsymbol{\phi}$'s are independent, and have normal prior distributions with zero mean. The variance of the ϕ's is regarded as a hyperparameter, which controls the magnitude of coefficient parameters. When a part of the hierarchy is irrelevant, we hope the posterior distribution of its corresponding hyperparameter will be concentrated near zero, which results in the related parameters becoming close to zero. In figure 2, for example, when the hyperparameters in the first level become small (compare 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. For this hierarchy we use the following priors:

$$\begin{aligned} \alpha_j|\tau_0 &\sim N(0,\tau_0^2) \qquad j=1,...,4 \\ \phi_{mkl}|\tau_m &\sim N(0,\tau_m^2) \\ \tau_0^{-2} &\sim Gamma(a_0,b_0) \\ \tau_m^{-2} &\sim Gamma(a_m,b_m) \end{aligned}$$

Here, ϕ_{mbl} refers to the parameter related to covariate x_l and branch k of node m. As we can see, all the branches that share the same node are controlled by one hyperparameter.

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

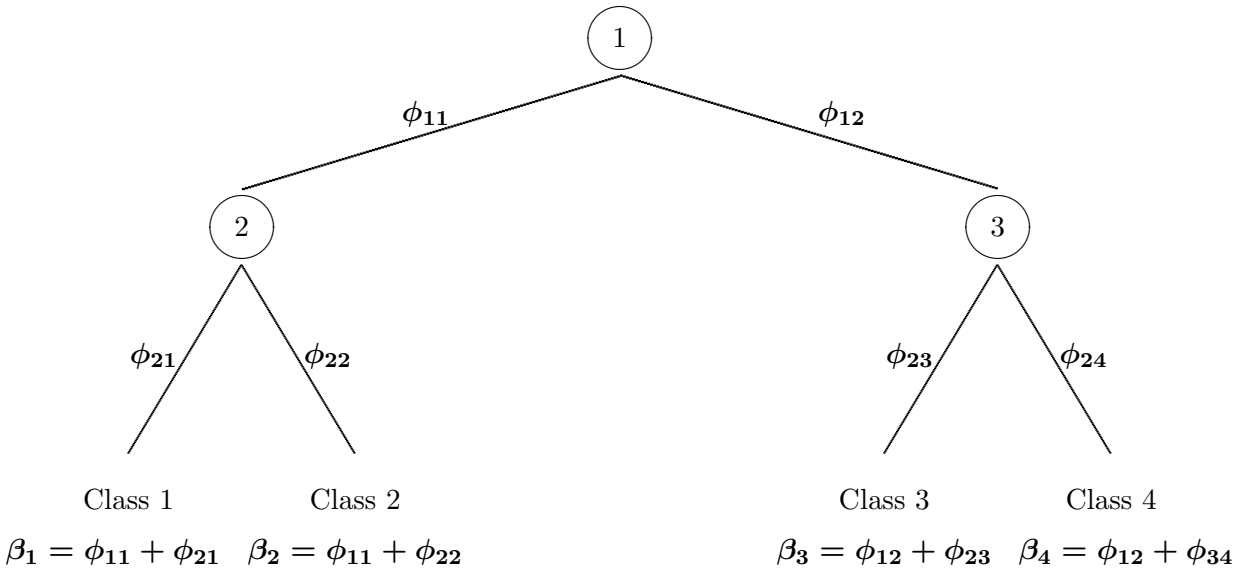

Figure 2: A simple representation of our model. The coefficient parameters for each classes are presented as a sum of parameters at different level of hierarchy

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.

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, τ_0, and the standard deviation for the rest of coefficients, τ, have the following priors:

$$\begin{aligned} \tau_0^{-2} &\sim Gamma(1, 10) \qquad (0.16, 0.38, 1.98) \\ \tau^{-2} &\sim Gamma(1, 1) \qquad (0.52, 1.20, 6.27) \end{aligned}$$

We use the parameterization of the Gamma distribution in which 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 $a^{1/2}b$. We gave the 2.5, 50 and 97.5 percentiles of τ in parenthesis.

For treeMNL and corMNL models, we assume that classes are arranged in a hierarchical form as shown in Figure 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: $P(y \in \{1,2\}|\alpha_1, \boldsymbol{\beta_1}, x)$, $P(y = 1|\alpha_2, \boldsymbol{\beta_2}, x, y \in \{1,2\})$ and $P(y = 3|\alpha_3, \boldsymbol{\beta_3}, x, y \in \{3,4\})$. For corMNL model, we use the priors discussed in section 2. The variance of regression parameters β's in treeMNL and 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:

$$\begin{aligned}
\tau_0^{-2} &\sim Gamma(1, 10) && (0.16, 0.38, 1.98) \\
\tau_1^{-2} &\sim Gamma(1, 5) && (0.23, 0.54, 2.82) \\
\tau_2^{-2} &\sim Gamma(1, 20) && (0.05, 0.12, 0.63) \\
\tau_3^{-2} &\sim Gamma(1, 20) && (0.05, 0.12, 0.63)
\end{aligned}$$

Here, τ_1, τ_2 and τ_3 correspond to nodes 1, 2, and 3 respectively (Figure 2). These parameters have a narrower prior compared to τ in the MNL model. This is to account 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 allow 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 the multinomial model (for data generated from the MNL and corMNL) or based on successive logistic models (for data generated from the treeMNL).

All three models are 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 [7]. At each iteration, we used the "stepping out" procedure to find the interval around the current point and used the "shrinkage" procedure for sampling from the interval. Since the hyperparameters were given conjugate priors, direct Gibbs sampling could be used for them. For all tests we drew 1000 samples from the posterior distributions. We discarded the initial 250 samples and used the rest for prediction. Performance is measured in terms of average log-probability and error rate on the test set.

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 iterations) are presented in Table 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

N=100	**Data from MNL**		**Data from treeMNL**		**Data from corMNL**	
	AvgLogProb	Error %	AvgLogProb	Error %	AvgLogProb	Error %
MNL method	**-0.7958**	**32.9**	-0.8918	41.6	-0.9168	40.6
treeMNL method	-0.8489	35.0	**-0.8770**	**41.3**	-0.9113	40.6
corMNL method	-0.7996	32.9	-0.8797	41.4	**-0.9075**	**40.5**

N= 50	**Data from MNL**		**Data from treeMNL**		**Data from corMNL**	
	AvgLogProb	Error %	AvgLogProb	Error %	AvgLogProb	Error %
MNL method	**-0.8041**	**33.6**	-0.8915	42.4	-0.9248	41.4
treeMNL method	-0.8506	35.1	**-0.8779**	**42.0**	-0.9227	41.5
corMNL method	-0.8086	33.7	-0.8794	42.1	**-0.9167**	**41.3**

Table 1: Comparison of models on simulated data based on average log-probability using training sets of size $N = 100$ and $N = 50$.

N= 100	**Data from MNL**		**Data from treeMNL**		**Data from corMNL**	
	AvgLogProb	Error %	AvgLogProb	Error %	AvgLogProb	Error %
MNL method	**-0.2539**	**10.1**	-0.3473	12.3	-0.3106	11.3
treeMNL method	-0.6837	23.1	-0.2898	**9.7**	-0.3614	12.1
corMNL method	-0.2910	10.3	**-0.2854**	9.9	**-0.2841**	**9.7**

Table 2: Comparison of models using a more complex hierarchy.

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. Reducing the size of training sets, N, to 50 increases the gap between the average performance of models but provides the same conclusions. The conclusions also remain the same when we use different priors and ranges of covariates.

While statistically significant, the results presented in Table 1 might not be significant for some practical purposes. This is mostly due to the simplicity of the hierarchical structure. Next, we repeated the above tests with a more complex hierarchy, as shown in Figure 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 $\tau^{-2} \sim Gamma(1, 1)$. The hyperparameters of treeMNL and corMNL were given $Gamma(1, 5)$, $Gamma(1, 20)$ and $Gamma(1, 100)$ priors for the first, second and third level of the hierarchy respectively.

Table 2 shows the average results over 100 datasets for each test. As we can see, the differences between models are more accentuated. Nevertheless, corMNL is still performing well even when the data are generated by other models, where corMNL is outperformed only by the true model. When data come from treeMNL, the results from corMNL are very close to those of the actual model (i.e., treeMNL).

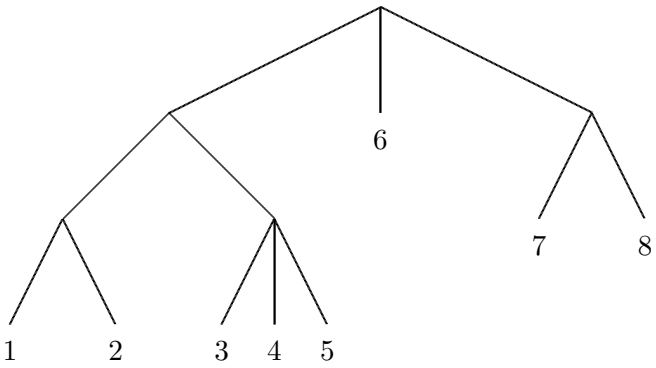

Figure 3: A hypothetical hierarchy with a relatively more complex structure.

4 Results on a document labelling problem

We next test our approach using a dataset of the page images of 15 articles (472 pages) from the Journal of Machine Learning Research (JMLR) collected by Laven [8]. Each page was segmented to several regions, and each region was manually classified to one of 24 possible classes. Figure 4 presents these classes in a hierarchical form. The hierarchy is based on our belief regarding how difficult it is to separate classes from each other using the available features.

The covariates are 59 different 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 divided the dataset into a training set (including 10 articles with 3521 regions), and a

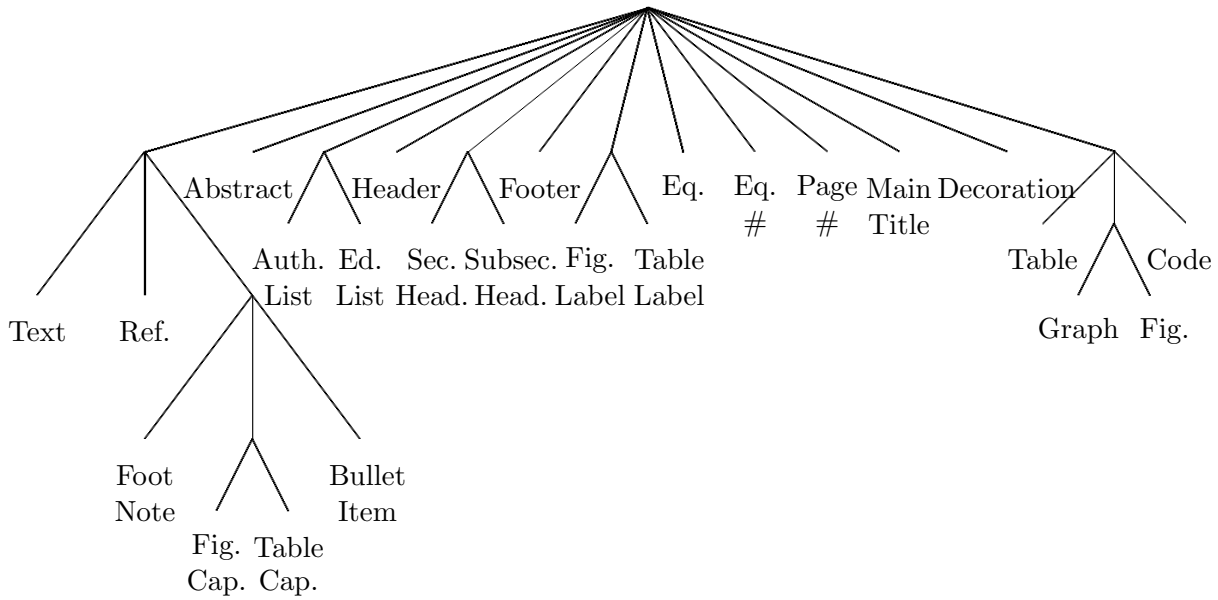

Figure 4: Hierarchical structure of document labels.

test set (including 5 articles with 2035 regions). The items from the same article are considered as independent even though they clearly are not. Although this may cause overfitting problems, we follow the same assumption for our initial test in order to make our results comparable to Laven's.

We trained the three models (MNL, corMNL and treeMNL) on the training set and evaluated their performance on the test set. The coefficient parameters in the MNL models were given normal prior 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 [9]. 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} $(j = 1, ..., c)$, 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. As before, we also use one hyperparameter, τ, to control all β's in the MNL model. We set the standard deviation of β_{jl} equal to $\tau\sigma_l$. Therefore, while σ_l control the relevance of covariate x_l compared to other covariates, the scale parameter τ, controls the overall usefulness of all covariates in separating classes. For the MNL model we used the following priors:

$$
\begin{array}{rcll}
\alpha_j|\quad \sigma_0 & \sim & N(0, \tau_0^2) & j = 1, ..., 24 \\
\beta_{jl}|\tau, \sigma_l & \sim & N(0, \tau^2\sigma_l^2) & l = 1, ..., 59 \\
\tau_0^{-2} & \sim & Gamma(0.5, 1) & (0.63, 2.09, 46.31) \\
\tau^{-2} & \sim & Gamma(0.5, 20) & (0.14, 0.47, 10.07) \\
\sigma_l^{-2} & \sim & Gamma(1, 10) & (0.16, 0.38, 1.98)
\end{array}
$$

Similar priors are used for the parameters of treeMNL and corMNL. For these two models, we used one hyperparamter, $\sigma_l^{-2} \sim Gamma(1, 10)$ to control all parameters related to covariate x_l. We also used one scale parameter $\tau_m^{-2} \sim Gamma(0.5, 100)$ 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 [10] for sampling from the posterior distribution of coefficients parameters. To reduce the random walk aspect of sampling procedure, we use a reasonably large number of leapfrog steps ($L = 500$). The stepsizes is set to 0.02 in order to maintain an acceptance rate of about 90%. In the MNL and corMNL models, the new updates 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 used single-variable slice sampling [7], as described above, for hyperparameters. The convergence of the Markov chain simulations were assessed by plotting the values of hyperparameters and average log-likelihood (on training cases). We ran each chain for 2500 iterations, of which the first 500 were discarded.

Table 3 compares the results from different models. As we can see, our MNL model has exactly the same performance as the likelihood-based model developed by Laven. The corMNL and treeMNL models have a slightly better error rate and a higher average log-probability compared to the MNL model.

We speculated that the assumed structure would have a more profound effect when a smaller

	AvgLogProb	Error rate %
Laven's model (ML)	–	9.4
MNL	-0.3998	9.4
treeMNL	-0.3602	9.3
corMNL	**-0.3598**	**9.2**

Table 3: Performance of models for the document labelling problem on the test set.

sample is available. To test this idea, we reduced the size of the training set to 200, and used 10 non-overlapping training sets in order to obtain a confidence interval for the performance of each model. To avoid problems from incorrectly assuming independence, we first combined the training and the test sets and then randomly sampled (without replacement) 10 training sets each with 200 cases and used the remaining 3556 cases as the test set. Sampling was performed without regard to which document each region comes from, which artificially makes the independence assumption be true. We trained the models on each training set and measured the performance on the test set. We discarded a burn-in period of 500 initial updates and retained the next 3500 updates for prediction. The results are shown in Table 4. Since the test set is large, the observed variation in the results is dominated by the difference between training sets, and we can use paired t-tests ($n = 10$) for comparing the results. Based on the average log-probability results, the corMNL model outperforms both MNL (p-value = 0.002) and treeMNL (p-value = 0.004) models. The improvement on error rate is only slightly significant compared to MNL (p-value = 0.09) and treeMNL (p-value = 0.06). The differences in performance of MNL and treeMNL are not statistically significant.

	AvgLogProb	Error rate %
MNL	-0.5441	15.4
treeMNL	-0.5381	15.5
corMNL	**-0.5160**	**14.9**

Table 4: Performance of models for the document labelling problem on the test set using 10 non-overlapping training sets of size 200.

5 Conclusions and future directions

In this paper, we have introduced a new approach for modelling hierarchical classes. Our analysis shows that when the hierarchy actually does provide information regarding the similarity of classes, our approach outperforms simple MNL model and models based on decomposing the hierarchy into nested MNL models. Our method can be applied to many classification problems where there is prior knowledge regarding the structure of classes. One such problem, which we hope to investigate soon, is annotation of gene functions.

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. We might then consider all possible hierarchies as equally likely and form an ensemble model based on this assumption (as done in [11]), but this might not be easy when the number of classes is large. Also, in most cases we have prior knowledge that leads us to prefer some structures over others a priori. 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.

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.

Acknowledgements

We thank Kevin Laven for providing the document labelling dataset. This research was supported by the Natural Sciences and Engineering Research Council of Canada. Radford Neal holds a Canada Research Chair in Statistics and Machine Learning.

References

[1] Rison S. Comparison of functional annotation schemes for genomes. *Functional and Intergrative Genomics*, 1:56–69, 2000.

[2] Agresti A. *Categorical Data Analysis*. John Willey and Son, Hoboken, New Jersy, 2002.

[3] Sattath S. and Tversky A. Additive similarity trees. *Psychometrika*, 42:319–345, 1977.

[4] McFadden D. Econometric Models for Probabilistic Choice Among Products. *Journal of Business* 53:13-36, 1980.

[5] Goodman J. Classes for fast maximum entropy training. *Proceedings of the IEEE International Conference on Acoustics, Speach and Signal Processing (ICASSP-2001), IEEE press*, 2001.

[6] Fox J. *Applied Regression Analysis, Linear Models and Related Methods*. Sage, 1997.

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

[8] Laven K. *Application of Statistical Pattern Recognition to Document Segmentation and Labelling*. Master's Thesis, University of Toronto, Department of Computer Science, 2005.

[9] Neal R.M. *Bayesian Learning for Neural Networks*. Springer Verlag, New York, 1996.

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

[11] Eibe F. and Kramer S. Ensembles of nested dichotomies for multi-class problems. *Proceedings of the 21st International conference of Machine Learning (ICML-2004)*, 2004.