2005

A Dynamic Bayesian Network Model for Hierarchial Classification and its Application in Predicting Yeast Genes Functions

Xutao Deng
University of Nebraska at Omaha, xdeng@mail.unomaha.edu

Huimin Geng
University of Nebraska, huimingeng@unmc.edu

Hesham H. Ali
University of Nebraska at Omaha, hali@mail.unomaha.edu

A Dynamic Bayesian Network Model for Hierarchical Classification and its Application in Predicting Yeast Genes Functions

Xutao Deng
College of Information Science and Technology
University of Nebraska at Omaha
xdeng@mail.unomaha.edu

Huimin Geng
Department of Pathology and Microbiology
University of Nebraska Medical Center
huimingeng@unmc.edu

Hesham H. Ali
Department of Computer Science
University of Nebraska at Omaha
hali@mail.unomaha.edu

ABSTRACT

In this paper, we propose a Dynamic Naive Bayesian (DNB) network model for classifying data sets with hierarchical labels. The DNB model is built upon a Naive Bayesian (NB) network, a successful classifier for data with flattened (nonhierarchical) class labels. The problems using flattened class labels for hierarchical classification are addressed in this paper. The DNB has a top-down structure with each level of the class hierarchy modeled as a random variable. We defined augmenting operations to transform class hierarchy into a form that satisfies the probability law. We present algorithms for efficient learning and inference with the DNB model. The learning algorithm can be used to estimate the parameters of the network. The inference algorithm is designed to find the optimal classification path in the class hierarchy. The methods are tested on yeast gene expression data sets, and the classification accuracy with DNB classifier is significantly higher than it is with previous approaches– flattened classification using NB classifier.

Keywords

Bayesian network, naive Bayesian classifier, dynamic Bayesian network, hierarchical classification.

INTRODUCTION

Classification is an essential operation in data mining and pattern recognition. It often involves a two-step learning-prediction procedure. Learning is to induce a classifier from observations with known class labels, that is, to construct a function assigning class labels for new observations. Prediction is to apply the induced classifier to assigning class labels for new observations. Many methods for classification exist including support vector machines, decision trees, artificial neural networks, and Bayesian networks.

Most classification methods assume that the class labels are mutually exclusive and non-overlapping so that all class labels are at the same level. Recently there is interest in generalizing traditional methods to deal with class labels in a hierarchy. Hierarchical class labels are very common in applications such as World Wide Web mining, market planning, medical diagnosis, and bioinformatics. Figure 1 shows an example of graphical representation for class hierarchy. The class hierarchy has a tree structure if we don't consider the case of multiple inheritances (in that case, the class hierarchy is a DAG (Directed Acyclic Graph)). Another observation is that any child-class label automatically inherits its ancestors' class labels.

There exist numerous approaches dealing with hierarchical classification. The obvious choice is the so-called "flattened" approach (Koller and Sahami, 1997) which ignores the class hierarchy and flattens all class labels to the same level so that standard classification methods can be directly applied. These kinds of classifiers are forced to compare hypotheses at different levels, i.e., "Is she a student or a graduate student?" In a probability classifier such as a Bayesian network, the classifier will favor root-level classes because the likelihood for parental classes is generally higher than for offspring classes. Another category of approaches (Koller and Sahami, 1997; Chakrabarti *et al.*, 1997) focuses on making a series of classifications in a top-down direction along the class hierarchy. For example, if we predict a person is a faculty member in the first level, we then try to predict whether she or he is tenured or nontenured. A major problem with this approach is that it

is not very fault-tolerant; that is, if we make a false prediction on a top level, it will never reach the correct label in lower levels. To overcome this problem, Cheng *et al.* (2001) developed a Bayesian model with error-control mechanisms. Recently, Gyftodimos and Flach (2003) developed a Bayesian network model for hierarchical classification. Their Bayesian network has the same structure as the class hierarchy with each class label modeled as a random variable. Their approach is to find the path with the highest likelihood, and the class labels on this path would be predicted labels for a new instance. The main problem of this method, again, is comparing hypotheses at different levels of the class hierarchy. We believe that comparing hypotheses in different levels will result in inconsistent classification and poor classification accuracy. The technical reason is that comparing higher-level hypotheses with lower-level hypotheses will violate the probability law. In addition, higher-level classes usually have higher likelihood and shorter description length and thus are more favorable in classification than are lower-level hypotheses.

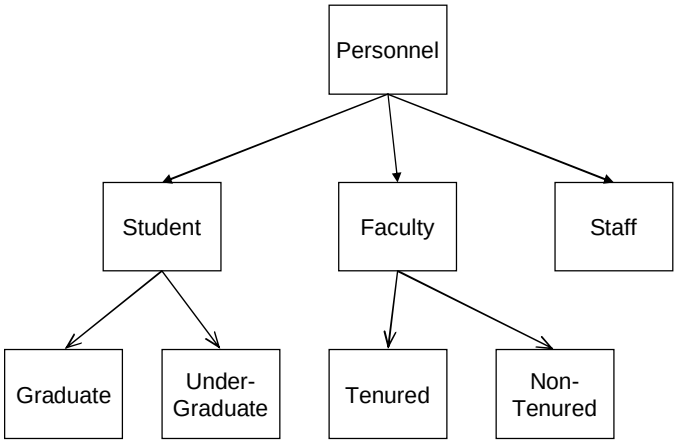

Figure 1. A Typical Class Hierarchy of School Personnel

In this paper, we present a Dynamic Naive Bayesian (DNB) network model to capture the class hierarchy. We defined two graph manipulations to augment the original class hierarchy in order to avoid comparing hypotheses in different levels. This paper is organized as follows. The next section provides background information regarding Bayesian networks and Naive Bayesian (NB) classifiers (Duda and Hart, 1973). In section 3, we describe how to manipulate hierarchy to satisfy the probability law. In section 4, we present the DBN model structure and the learning and inference algorithms. Experimental results on classifying yeast genes are presented in section 5.

PRELIMINARIES

Bayesian Networks and Dynamic Bayesian Networks

A Bayesian network (Pearl, 1988; Heckman *et al.*, 1995; Buntine, 1996) is a graphical representation of the joint probability distribution of a set of random variables $X_1, X_2, \mathsf{K}, X_n$. Formally, a Bayesian network can be represented as a pair $B = \langle G, \theta \rangle$. The first component, *G*, is a directed acyclic graph whose nodes correspond to the random variables $X_1, X_2, \mathsf{K}, X_n$, and whose edges represent the direct dependencies between the variables. We use PA(X_i) to denote the parent nodes of the node X_i. The second component, θ, represents the set of parameters that quantifies the network. It contains the value $\theta_{x_i|\mathrm{PA}(x_i)} = P(x_i \mid \mathrm{PA}(x_i))$ for each of the possible values x_i of X_i and PA(x_i) of PA(X_i). The parameter set θ is usually represented as CPT (Conditional Probability Table) for discrete distributions. A key feature of a Bayesian network is the *Markov condition*, which says each variable X_i is independent of its nondescendants given the value of its parents in *G*:

$$P(X_1,..., X_n) = \prod_{i=1}^{n} P(X_i \mid \mathrm{PA}(X_i)) = \prod_{i=1}^{n} \theta_{X_i|PA(X_i)} \ . \qquad (1)$$

A dynamic Bayesian network is a special type of Bayesian network. Dynamic Bayesian networks are usually used to describe sequential variables in which time (or another factor) represents a natural dependency relationship between variables. In our proposed classifier, the natural dependency is the class hierarchy. Examples of dynamic Bayesian networks include Markov models, hidden Markov models and Kalman filters.

Naive Bayesian Classifier

Bayesian networks are not only representation models but also computing tools which are widely applied in diagnosis, classification and forecasting. One of the most successful applications is the NB classifier. A NB classifier has a single variable C with all class labels as its possible values (states). The node C is the parent of all other attributes (nodes) A_1, A_2, …, A_n, where n is the number of attributes for the instances. A NB classifier has an umbrella shape shown in Figure 2.

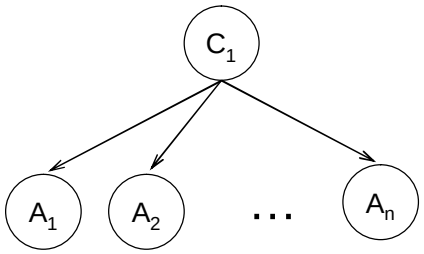

Figure 2. Network Structure of Naive Bayesian Classifier

The value of the parameter θ (CPT) can be estimated from the training instances. The prediction for a new instance can be performed by calculating its posterior distribution

$$\begin{aligned}\Pr(C \mid A_1, A_2, \mathrm{K}, A_n) &= \frac{\Pr(C)\cdot \Pr(A_1, A_2, \mathrm{K}, A_n \mid C)}{\Pr(A_1, A_2, \mathrm{K}, A_n)} \\ &= \alpha \cdot \Pr(C)\cdot \Pr(A_1, A_2, \mathrm{K}, A_n \mid C) \\ &= \alpha \cdot \Pr(C)\cdot \prod_{i=1}^{n} \Pr(A_i \mid C)\end{aligned}$$

where the first line follows from Bayes' theorem; the third line is valid for NB classifiers where the conditional independence assumption holds, i.e., given the value of class label nodes, the attribute variables are independent. This assumption, however, enables us to decompose the likelihood function $\Pr(A_1,A_2,\ldots,A_n|C)$ so that both learning and inference can be performed in a timely and efficient fashion. Following the above equation, the classification using NB classifiers can be performed according to

$$\hat{c} = \arg\max_{C_j} \Pr(C_j \mid A_1, A_2, \mathrm{K}, A_n) = \arg\max_{C_j} \Pr(C_j)\cdot \prod_{i=1}^{n} \Pr(A_i \mid C_j) \quad j = 1, 2, \ldots, m, \tag{2}$$

where $\hat{c}$ is the predicted class label for an instance; m is the number of possible classes for all instances. In order to simplify notation, we use $A = \langle A_1, A_2, \mathrm{K}, A_n \rangle$ to denote both the attribute and the value of the attribute. The NB classifier works by comparing the posterior probability of competing hypotheses (class labels) given the particular instance of A. The most likely class label is assigned as the label for the new instance.

The assumption of conditional independence generally does not hold in general applications, but the results from NB classifiers are surprisingly good. Domingos and Pazzani (1997) gave a good explanation addressing this issue. For more information regarding the Bayesian network and NB classifier, refer to the text by Friedman *et al*. (1997).

PROBABILITY ARGUMENTS AND CLASS HIERARCHY AUGMENTATION

The problem of comparing contradictory hypotheses is rooted in the violation of the probability law, which says the probability of all mutually exclusive events in the entire sample space should add up to 1. In order to make the class hierarchy comply with the probability law, we require the classifier to satisfy that the probability of all competing hypotheses (class labels) in any level must sum to 1.

Formally, we denote

$$\begin{aligned} S_{ij} &= \{\text{all instances belong to class } j \text{ on the } i\text{th level}\} \\ P(\bigcup_{j=1}^{m_i} S_{ij}) &= \sum_{j=1}^{m_i} P(S_{ij}) = 1, \quad i = 1, 2, \mathrm{K}, k \end{aligned} \tag{3}$$

where k is the length of the class hierarchy (the length of the longest path from the imaginary root to a leaf); m_i is the number of classes in level i. This condition implies that all possible instances (events) cover the entire sample space in any level.

We define two types of augmentation operations to transform the class hierarchy which violates the probability law into a form satisfying the above condition. Figure 3 show examples of the augmentations.

Type I augmentation*:* Let k be the greatest length of all the paths from the root to every leaf. Type I augmentation is an operation that extends the leaves to make all paths from the root to any leaves the same length k. Figure 3-*b* shows a demo of type I augmentation in which class $c_{1,2}$ is augmented by creating a new class node $c_{2,3}$. Label $c_{2,3}$ can be understood as the only subclass of $c_{1,2}$.

Type II augmentation*:* Type II augmentation is needed only when the instances are not represented in any leaf node in the original class hierarchy. This happens only to the non-leaf nodes. For example, some instances are of class c_{11} but not of c_{21} or c_{22}. We create an imaginary path to represent those instances in the hierarchy. This path (see Figure 3-*c*) should reach the leaf level. It can be viewed as an "unknown" subclass label.

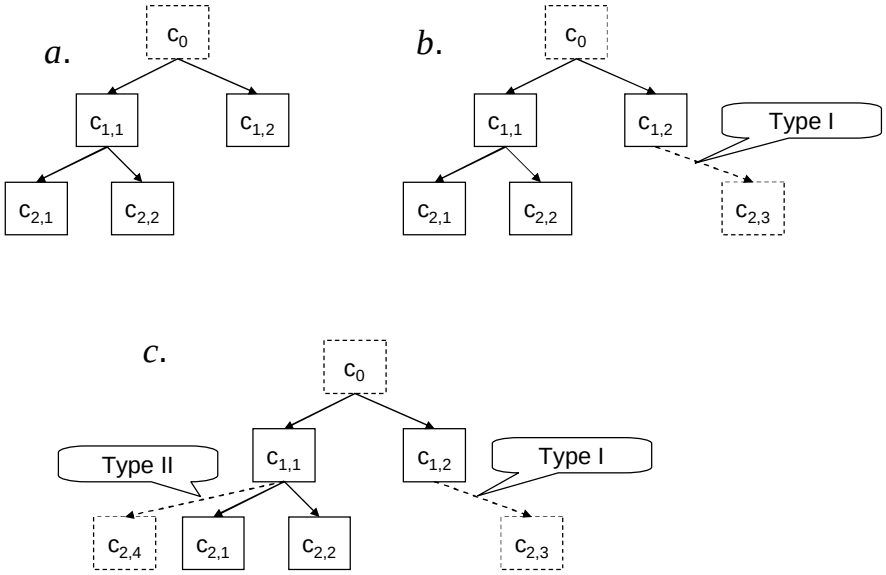

Figure 3. Augmentation Operations on a Class Hierarchy. *a.* The original hierarchy; *b.* The hierarchy after type I augmentation; *c.* The hierarchy after both types of augmentation

Augmentations result in a new class hierarchy that satisfies the above conditions and allows classification in any class level. The nodes and the edges added for balancing and filling are treated the same way as the others (in terms of representation and computing). Balancing and extending allow comparing hypotheses (class labels) in any level in the hierarchy without violating the two conditions. They can be performed either manually or automatically by analyzing the class hierarchy.

DYNAMIC NAIVE BAYESIAN CLASSIFIER

Model Structure

The structure of our Bayesian classifier is based on the NB classifier. The standard NB classifier is designed to handle data with flattened class labels which are modeled as the states (possible values) of a random variable representing the class. Figure 4 shows the structure of our DNB model. We use a series of class variables C_1, C_2, …, C_k, one for each class layer, to represent the class hierarchy, where k is the number of layers in the hierarchy. The natural dependency between class variables follows the hierarchical structure from the top-down. The "dynamic" in our DNB refers to this sequential class hierarchy, not to the time factor. The attributes variables A_1, A_2, …, A_n are handled similarly as those in the NB classifier, except that each attribute variable is pointed to by all class variables. The class hierarchy is captured by the backbone chain in the model. Note that we use circles to represent variables and squares to represent their states. C_i is a random variable which has states $c_{i,1}, c_{i,2}, \mathrm{K}\, c_{i,m_i}$, where $i = 1, 2, \mathrm{K}, k$ and m_i is the number of states in the ith level.

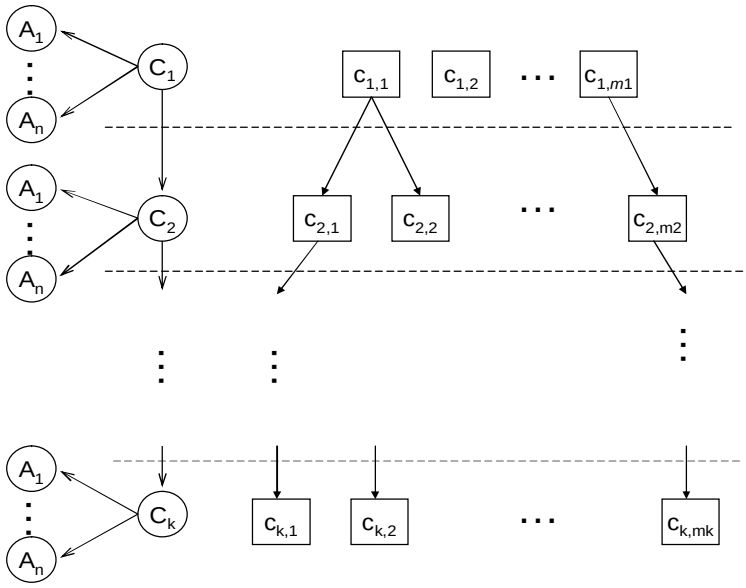

Figure 4. The Structure of DNB Model and Its Corresponding Class Hierarchy

DNB Learning

Since the graph structure is specified, the learning is to find the CPT $\theta_{x_i|\mathrm{PA}(x_i)} = P(x_i \mid \mathrm{PA}(x_i))$ for each of the possible values x_i of X_i and PA(x_i) of PA(X_i). When all random variables are discrete, we can estimate parameters from a set of training instances by its maximum likelihood estimator. It can be shown that the maximum likelihood estimator for each parameter $\theta_{x_i|\mathrm{PA}(x_i)}$ is the relative frequency of training data,

$$\hat{\theta}_{x_i|\mathrm{PA}(x_i)} = \frac{count(x_i \mid \mathrm{PA}(x_i))}{\sum_{x_i' \in Val(X_i)} count(x_i' \mid \mathrm{PA}(x_i))} = \frac{count(x_i \mid \mathrm{PA}(x_i))}{count(\mathrm{PA}(x_i))}, \quad (4)$$

where $count(x_i \mid \mathrm{PA}(x_i))$ is the number of instances having a value of $x_i \mid \mathrm{PA}(x_i)$. For small training data sets, the maximum likelihood estimates are likely to be unreliable because there may be very few instances for estimating certain parameters. This situation is the so-called "overfitting" problem. We apply a smoothing technique (Friedman *et al.* 1997) using *Dirchlet priors* to improve the maximum likelihood estimators. This method is also called the *pseudo-count* method (Durbin *et al.* 1998) where the pseudo-count is an arbitrary number which reflects the prior belief of the distribution of $\theta_{x_i|\mathrm{PA}(x_i)}$. The smoothed estimators have the form

$$\hat{\theta}_{x_i|\mathrm{PA}(x_i)} = \frac{count(x_i \mid \mathrm{PA}(x_i)) + peudo\_count(x_i \mid \mathrm{PA}(x_i))}{count(\mathrm{PA}(x_i)) + peudo\_count(\mathrm{PA}(x_i))}. \quad (5)$$

DNB Inferences

With all the parameters determined from learning, we have two options for classifying new instances. For a new instance, we can determine the most likely class label (node) it belongs to in any class level by computing probabilities based on Bayesian theorem. Alternatively, we can compute the most likely path from the imaginary root node to any leaf node.

Node Inference: For an instance with the value $(a_1, a_2, \mathrm{K}, a_n)$, its most likely class label in the *i*th level of the class hierarchy, $\hat{c}_i$, can be determined in the following equations by plugging $(a_1, a_2, \mathrm{K}, a_n)$:

$$\begin{aligned}\hat{c}_i &= \arg\max_{Val(C_i)} \Pr(C_i \mid A_1, A_2, \mathsf{K}, A_n) \\ &= \arg\max_{Val(C_i)} \left(\Pr(C_i) \cdot \prod_{l=1}^{n} \Pr(A_l \mid C_i) \right) \\ &= \arg\max_{Val(C_i)} \left(\prod_{p=1}^{i} \theta_{C_p \mid C_{p-1}} \cdot \prod_{l=1}^{n} \theta_{A_l \mid C_i} \right) \quad i = 1, 2, \ldots, k\end{aligned}$$

The above calculation can be performed in $O(nkm_i)$ time since the likelihood term can be decomposed in our DNB model.

Path Inference: Denote the most likely state path $\hat{\pi}_i$, which starts from the root and ends at a node in level *i*, $1 \le i \le k$. We have the following theorem:

Theorem*: When the augmented class hierarchy is a tree, we have the $\hat{c}_i \in \hat{\pi}_i$ using our DNB model.*

Proof:

$$\begin{aligned}\hat{\pi}_k &= \arg\max_{\pi=(c_1, c_2, \ldots, c_k)} \Pr(C_1, C_2, \mathsf{K}, C_k \mid A) \\ &= \arg\max_{\pi=(c_1, c_2, \ldots, c_k)} \prod_{p=1}^{k} \Pr(C_p \mid C_{p-1}, A) \\ &= \arg\max_{\pi=(c_1, c_2, \ldots, c_k)} \prod_{p=1}^{k} \frac{\Pr(C_p, C_{p-1}, A)}{\Pr(C_{p-1}, A)} \\ &= \arg\max_{\pi=(c_1, c_2, \ldots, c_k)} \left(\frac{\Pr(C_k, C_{k-1}, A)}{\Pr(C_{k-1}, A)} \cdot \frac{\Pr(C_{k-1}, C_{k-2}, A)}{\Pr(C_{k-2}, A)} \cdot \mathsf{L} \cdot \Pr(C_1) \right) \\ &= \arg\max_{\pi=(c_1, c_2, \ldots, c_k)} \left(\frac{\Pr(C_k, A) \cdot \Pr(C_{k-1} \mid C_k, A)}{\Pr(C_{k-1}, A)} \cdot \frac{\Pr(C_{k-1}, A) \cdot \Pr(C_{k-2} \mid C_{k-1}, A)}{\Pr(C_{k-2}, A)} \cdot \mathsf{L} \cdot \Pr(C_1) \right) \\ &= \arg\max_{\pi=(c_1, c_2, \ldots, c_k)} \Pr(C_k, A) \\ &= \arg\max_{\pi=(c_1, c_2, \ldots, c_k)} \Pr(C_k \mid A)\end{aligned}$$

when the augmented class hierarchy has a tree structure, each node has only one parent so that we have $\Pr(C_{k-1} \mid C_k, A) = \begin{cases} 1, & \text{when } C_{k-1} = \mathrm{PA}(C_k) \\ 0, & \text{Otherwise} \end{cases}$.

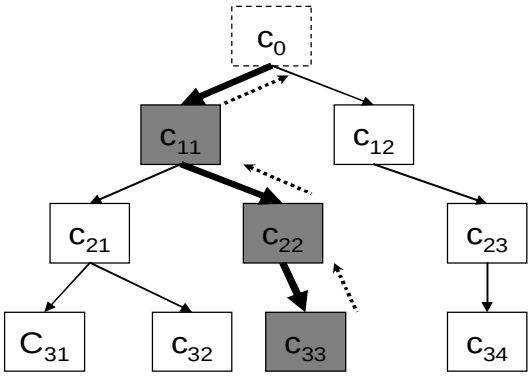

Figure 5. A Path Inference Can Be Transformed into a Node Inference Problem

This proposition shows that a path inference can be transformed into a node inference problem when the class structure is a tree. The example in Figure 5 shows the most likely state path is $\hat{\pi}_3 = \langle c_0, c_{1,1}, c_{2,2}, c_{3,3} \rangle$ when $\hat{c}_3 = c_{3,3}$.

EXPERIMENTAL RESULTS

Yeast is an excellent model organism which has a reasonably simple genome structure, well-characterized gene functions, and huge expression data sets. The proposed methods are applied to classifying yeast gene functions based on their expression data. The data set has been classified using support vector machines, K-nearest-neighbors (Brown *et al.*, 2000; Pavlidis *et al.*, 2001; Kuramochi and Karypis, 2001), and Hidden Markov Models (Deng and Ali, 2004; Deng *et al.*, 2005). However, none of these approaches have addressed the specific problems of classification with hierarchical data sets. The expression data set is obtained from http://rana.lbl.gov/EisenData.htm (Eisen *et al.*, 1998). The complete data set contains 2467 genes with each gene having 79 experimental measurements recorded. Among the 2467 genes, 2432 have at least one function annotation at MIPS, http://mips.gsf.de/ (Mewes *et al.*, 2002). For the purpose of training, we only include 23 function classes which have generally more than 100 ORFs in MIPS.

The augmented class hierarchy is labeled as a two-level suffix tree shown in Figure 6. We use the equal frequency approach to discretize the original numerical data into 10 symbols. Together with the augmented nodes (labeled in the dashed square), there are 18 class paths in level 2. In the hierarchical classification using our DNB model, the prediction is made on 18 classes in level 2 to find the most probable class paths for each instance. In the flattened classification using a NB model which ignores the class hierarchy, 23 classes at all levels (labeled in the solid square) are included in the training and prediction. When comparing the results using the hierarchical classification to those using the flattened classification, in addition to the number of true positives and the number of false positives, we use *precision* to measure the classification accuracy. Precision is defined as:

$$Precision = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Positives}. \quad (6)$$

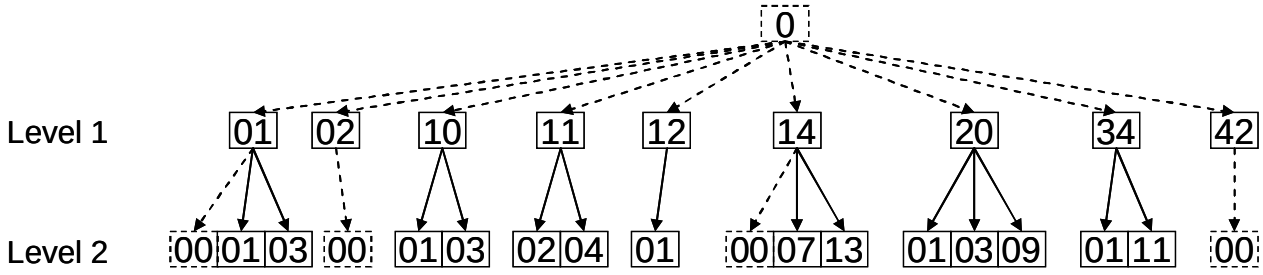

Figure 6. The Class Hierarchy of the Yeast Gene Function Data Set

Table 1 shows the total number of true positives and false positives and precisions of the two methods. In the flattened classification, we consider it is a false if some instance is labeled as a class in level 1 but in fact it belongs to a class in level 2. In the hierarchical classification, since all comparisons are based on level 2, it is clear that a misclassified instance is a false. We see the overall precision is improved by 11.7% using the DNB model. This result proved our hypothesis that the classification accuracy would increase if we consider the hierarchical class structure.

	# True Positives	# False Positives	Precision (%)
Hierarchical Classification	1270	1039	55.0
Flattened Classification	1001	1308	43.3

Table 1. 10-Fold Cross-Validation of the Hierarchical Classification and Flattened Classification

Figure 7 shows the detailed comparison of the number of true positives between the two methods in all 18 class paths in level 2. We see that the number of true positives is higher in the hierarchical classification than in the flattened classification in the 17 out of a total of 18 class paths. Again, this result shows that classification in the same level is fair and would improve the classification accuracy. Figure 8 shows the number of predictions made by the two methods for all 23 class labels in both levels. We can see that the number of predictions made for each class on level 2 is generally higher in hierarchical classification than its correspondent in flattened classification. However, for the five classes (10, 11, 12, 20 and 34) on level 1, the hierarchical classification never made a prediction because we separated the classes for disparate levels. This tells us that in flattened classification, the competitive parental classes on level 1 have taken away many predictions which cause the

low number of total predictions on level 2. Therefore, the number of true positives and overall precision decrease in the flattened classification using NB. The overall precision for 23 classes with flattened classification is poor (Table 1) because it is very possible that an instance is predicted in level 1 while in fact it belongs to a subclass in level 2. Again, we must avoid overlapping classification on disparate levels.

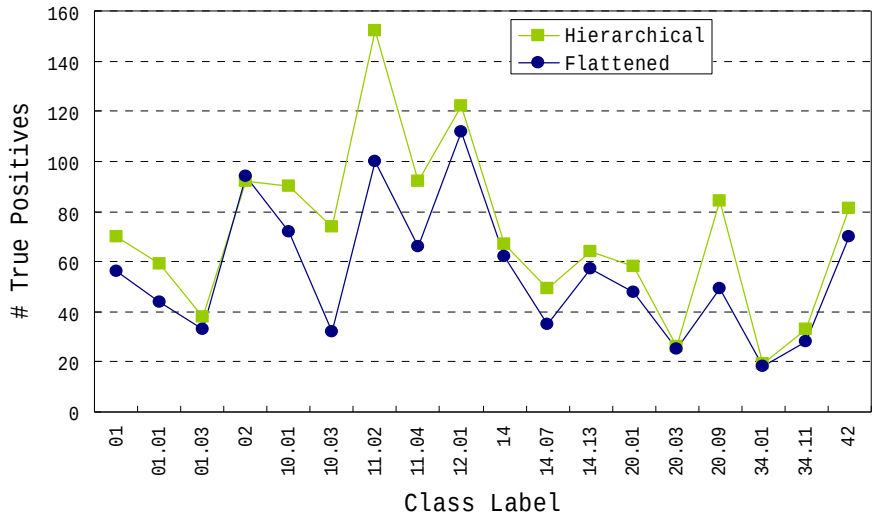

Figure 7. Number of True Positives Obtained by Both Methods

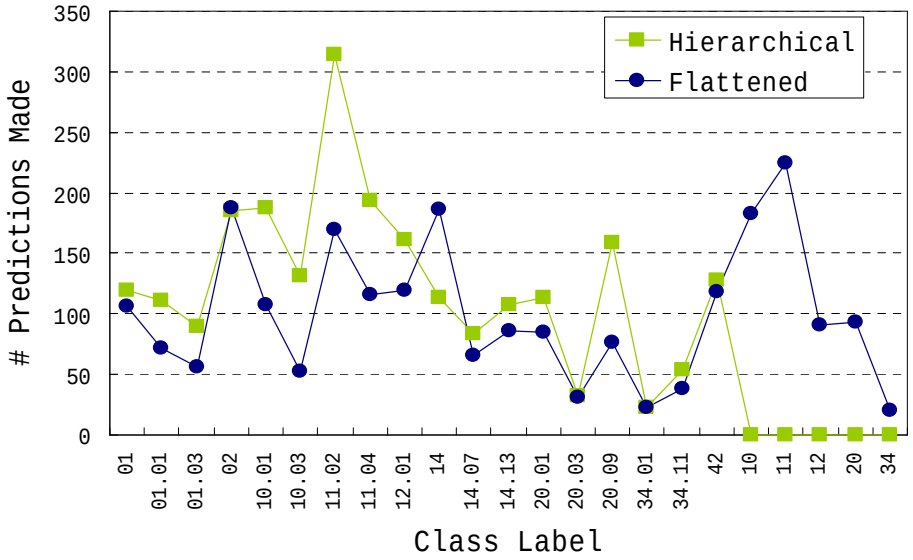

Figure 8. Number of Predictions Obtained by Both Methods Over 23 Classes in Both Levels

CONCLUSIONS

In this paper, we present a DNB model for hierarchical classification. Two augmenting operations are defined on the class hierarchy. Learning and Inference algorithms are provided for the DNB model. Experimental results using yeast gene expression data proved our hypotheses of the new method. The DNB model structure shares similarity with HMMs in which

the states' space are the same for each hidden variable (class). Future developments include classification without training labels which can be estimated using the EM algorithm. We also plan to generalize the tree class hierarchy to DAGs for class structure with multiple inheritances.

ACKNOWLEDGEMENTS

This work was supported by the NIH grant number P20 RR16469 from the INBRE program of National Center for Research Resource.

REFERENCES

1. Buntine, W. L. (1996) A Guide to the Literature on Learning Probabilistic Networks from Data, *IEEE Trans. Knowl. Data Eng.*, 8(2), 195-210.
2. Chakrabarti, S., Dom, B., Agrawal, R. and Raghavan, P. (1997) Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases, *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB)*, 446-455.
3. Cheng, C. H., Tang, J., Wai-chee, A. and King, I. (2001) Hierarchical Classification of Documents with Error Control, *Proceedings of the 5th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD 2001)*, 433-443.
4. Deng, X. and Ali, H. (2004) A Hidden Markov Model for Gene Function Prediction from Sequential Expression Data**,** *Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference (CSB 2004)*, 670-671.
5. Deng, X., Geng, H. and Ali, H. (2005) Predicting Yeast Gene Function Based on Hidden Markov Models, *Proceedings of the 20th International Conference on Computers and Their Applications (CATA 2005)*, 196-201.
6. Domingos, P. and Pazzani, M. (1997) On the optimality of the simple Bayesian classifier under zero-one loss, *Machine Learning*, 29, 103–130.
7. Duda, R. and Hart, P. (1973) Pattern Classification and Scene Analysis. New York John Wiley and Sons.
8. Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. (1998) Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press.
9. Eisen, M., Spellman, P., Brown, P.O. and Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns, *Proceedings of the National Academy of Sciences of the United States of America*, 95, 14863–14868.
10. Friedman, N., Geiger, D. and Goldszmidt**,** M. (1997) Bayesian networks classifiers, *Machine Learning*, 29, 131-163.
11. Friedman, N., Linial, M., Nachman, I. and Pe'er, D. (2000) Using Bayesian networks to analyze expression data, *Journal of Computational Biology*, 7, 3/4, 601--620.
12. Gyftodimos, E and Flach, P. (2003) Hierarchical Bayesian Networks: an Approach to Classification and Learning for Structured Data, *Proceedings of the ECML/PKDD - 2003 Workshop on Probabilistic Graphical Models for Classification*, 25-36.
13. Heckerman, D., Geiger, D. and Chickering, D. (1995) Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning*, 20, 197-243.
14. Koller, D. and Sahami, M. (1997) Hierarchically classifying documents using very few words, *Proceedings of the 14th International Conference on Machine Learning (ML)*, 170-178.
15. Kuramochi, M. and Karypis, G. (2001) Gene Classification Using Expression Profiles: A Feasibility Study, *Proceedings of The 2nd IEEE International Symposium on Bioinformatics & Bioengineering (BIBE 2001)*, 191.
16. Mewes, H. W., Frishman, D., Güldener, U., Mannhaupt, G., Mayer, K., Mokrejs, M., Morgenstern, B., Münsterkoetter, M., Rudd, S. and Weil, B. (2002) MIPS: a database for genomes and protein sequences, *Nucleic Acids Research*, 30(1), 31-34.
17. Pavlidis, P., Weston, J., Cai, J. and Grundy, W. N. (2001) Gene function classification from heterogeneous data, *Proceedings of the Fifth International Conference on Computational Molecular Biology*, 242-248.
18. Pearl, J. (1988) Probabilistic reasoning for intelligent systems. Morgan Kaufmann, San Francisco.