Learning Hierarchical Classifiers with Class Taxonomies

Feihong Wu, Jun Zhang, and Vasant Honavar

Artificial Intelligence Research Laboratory
Department of Computer Science
Iowa State University
Ames, Iowa 50011-1040, USA
{wuflyh, jzhang, honavar}@cs.iastate.edu

Abstract. As more and more data with class taxonomies emerge in diverse fields, such as pattern recognition, text classification and gene function prediction, we need to extend traditional machine learning methods to solve classification problem in such data sets, which presents more challenges over common pattern classification problems. In this paper, we define structured label classification problem and investigate two learning approaches that can learn classifier in such data sets. We also develop distance metrics with label mapping strategy to evaluate the results. We present experimental results that demonstrate the promise of the proposed approaches.

1 Introduction

Pattern classification is an important topic in machine learning and data mining research, and many state-of-the-art pattern classification algorithms have been developed. However, most of such algorithms are targeted to solve classification problem with single class label, which assumes all class labels are mutually exclusive. In many real world problems, it is quite common to have more complex class labels, such as multiple topic categories for text documents and multiple functional classes for biological data. The main characteristics of this problem are: (1) Class labels are naturally organized as a taxonomy structure (Class Taxonomy) which defines an abstraction over class labels; (2) Because of the large possible class combinations within a class taxonomies and relatively sparse data for combinatorial class labels, it is a hard problem to many standard classifier learning algorithms; (3) Standard evaluation method for classifiers targeting single label problem might not be suitable to evaluate classifiers for solving complex class label problems, new evaluation approaches are needed.

Although such problems have been explored to some extent, they are not fully formalized, and we still lack of a general strategy to solve the problems. In this paper, we formalize the structured label learning problem and analyze their distinct features. We propose and implement two approaches, with general strategies of applying any up-to-date classification learning methods. Because of the restrictions of the standard evaluation criterion, we propose a new evaluation

criterion called "label mapping", as an extension to standard distance metrics and general measurements to the learned classifiers. Our experimental results show that our methods work well.

The paper is organized as follows. In section 2 we formalize the single label, multi label and the structured label classification problem. In section 3, we describe binarization approach and split-based approach. In section 4 we describe how to use the distance metrics with label mapping to evaluate the learning result. In section 5, we apply the two methods to artificial data, Reuters-21578[13] data and genotype data[5, 6] and analyze the results. Finally we conclude with a summary, related work and possible future directions.

2 Preliminaries

In this section, we formally define single label and multi label classification problems and extend the definitions to structured label classification problems with regard to a pre-specified class taxonomy.

2.1 Single label Classification

Many standard classifier learning algorithms normally make the basic assumption of single label instances. That is, each instance that is represented by an ordered set of attributes $\mathbf{A} = \{A_1, A_2, ..., A_N\}$ can belong to one and only one class from a set of classes $\mathbf{C} = \{c_1, c_2, ..., c_M\}$. Therefore, class labels in $\mathbf{C}$ are mutually exclusive.

2.2 Multi label Classification

In many real world applications, it is quite common to encounter instances that have more than one class label.

In multi label classification settings, class labels are not mutually exclusive. Each instance can be labelled using a subset of labels $c_s \subset \mathbf{C}$, where $\mathbf{C} = \{c_1, c_2, ..., c_M\}$ is a finite set of possible classes. Considering the possibility that any multi label for an instance can be an arbitrary subset of $\mathbf{C}$, the total number of possible multi label combinations in $\mathbf{C}$ is 2^M.

2.3 Structured label Classification

An even more complex classification problem is that instances to be classified have structured labels with respect to a class taxonomy. Here, we define class taxonomy first and then formalize the structured label problem.

Definition 1 (Class Taxonomy). *A Class Taxonomy $\mathcal{CT}$ is a tree structured regular concept hierarchy defined over a partially order set $(\boldsymbol{C}_T, \prec)$, where $\boldsymbol{C}_T$ is a finite set that enumerates all class concepts in the application domain, and relation $\prec$ equivalently represents is-a relationship that is both anti-reflective and transitive:*

– *The only one greatest element "ANY" is the root of the tree.*
– $\forall c_i \in \boldsymbol{C}$, $c_i \prec c_i$ *is false.*
– $\forall c_i, c_j, c_k \in \boldsymbol{C}$, $c_i \prec c_j$ *and* $c_j \prec c_k$ *imply* $c_i \prec c_k$.

The tree structured class taxonomy represents class memberships at different levels of abstraction. The root of class taxonomy is the most general label (i.e., "ANY") that is applicable to any instance. The leaves of class taxonomy indicate the most specific labels. The tree structure imposes strict constraints on these class memberships. Therefore, in a given class taxonomy, when the most specific label is given, all its ancestral class labels are included automatically.

Now, we can formally define the structured label based on the given class taxonomy $\mathcal{CT}$.

Definition 2 (Structured label). *Any structured label* $\mathcal{C}_s$ *is represented graphically by a subtree of* $\mathcal{CT}$. $\mathcal{C}_s$ *is still a partially order set* $(\boldsymbol{C}_s, \prec)$ *that defines the same* is-a *relationships as in* $\mathcal{CT}$. $\forall c_i \in \boldsymbol{C}_s$, c_i *is ANY or* $c_i \prec parent(c_i)$, *where* $parent(c_i) \in \boldsymbol{C}_s$ *is the direct parent of* c_i.

The definition specifies a structured constraint on the integrity and validity of the structured labels. The integrity states that $\mathcal{C}_s$ is a subtree structure of $\mathcal{CT}$ sharing the same root. Structured label is not arbitrary fragmentation of the class taxonomy. The validity captures all valid *is-a* relationships among class labels and their parent labels. Invalid structured labels include the cases when a certain class label c_j is chosen, but $parent(c_j)$ is not chosen. Further constraints may apply to structured labels in the class taxonomy too, including additional incompatible class labels that may appear at different levels in the class taxonomy.

In structured label classification settings, each instance is labelled using a structured label basing on a predefined $\mathcal{CT}$.

Figure 1 shows a class taxonomy with class labels *A*-*H*, while $\{A, C, E, F, H\}$ is a valid structured label.

3 Methods

Most current classifier learning algorithms that work with single label data can not be directly applied to learning from data that are multi label or structured label. The direct way to deal with multi label and structured label problems is to revise the current learning algorithms, and make them be capable of learning from multi label and structured label data.

In what follows, we will focus on the structured label problem, which in turns will be a generalization of both single label problem and multi label problem. We consider the structure constraints over class labels directly in constructing a meta classifier, and eliminate the invalid predictions. We propose two learning methods that can learn from structured label data directly.

3.1 Binarization on Class Taxonomy

One simple approach is to build a classifier consisting of a set of binary classifiers (one for each class). However, the drawbacks of this approach are obvious: (1) When making predictions for unlabelled instances, the classification results may violate the membership constraints. That is, the predication may contradict the implications of *is-a* labels by the structure of class taxonomy and will not be a valid structured label. (2) The set of binary classifiers fails to take into account the structure information of the class taxonomy during learning.

To overcome those disadvantages, we build a set of hierarchically organized classifiers having the same structure as the class taxonomy $\mathcal{CT}$. The classifiers can be seen as a partially ordered set $(h_{\mathcal{CT}}, \prec)$, where $h_{\mathcal{CT}} = \{h_{C_1}, \cdots, h_{C_M}\}$ is the set of classifiers, and $\prec$ represents partial orders among classifiers. If C_j is a direct child node of C_i in $\mathcal{CT}$, then there is a partial order for the corresponding two classifiers $h_{C_j} \prec h_{C_i}$. This partial order on classifiers decides how an instance is classified. If $h_{C_j} \prec h_{C_i}$, an instance will not be classified using h_{C_j} if it has been classified as not belonging to C_i (i.e., output of h_{C_j} is 0). We call our method of building this hierarchically structured classifiers "Binarization on Class Taxonomy".

For example, if we have a class taxonomy defined in Figure 1, we will have corresponding hierarchically organized classifiers. The root of this hierarchy is a classifier h_{ANY}, which will always output"1" for any instance. Each node in the hierarchy has a corresponding classifier. When an unlabelled instance is given, it goes to h_{ANY} and h_{ANY} will always output "1", then it will be given as an input to h_A, h_B and h_C. If h_A output "1", which means the instance has label A, the instance is then given to h_D, h_E and h_F for further classification. Otherwise, if h_A output "0", this instance will not be further classified. Similarly, if h_C output "1", the instance will be sent to h_G and h_H for further classification. By following the structure of classifiers, we can guarantee the satisfaction of structure constraint (integrity and validity of structured labels), and no invalid labels will be generated by this partially structured meta classifier.

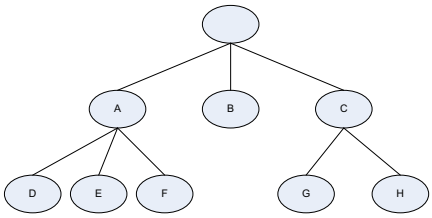

Fig. 1. class taxonomy

Another benefit gained by exploiting the class taxonomy is the efficient use of the training data. For independently trained binary classifier h_{C_i}, the positive instances are those have been labelled with C_i. We denote the whole data set

S, and instances with a C_i label S_{C_i}. The negative instances will be all the remaining ones that are not labelled with C_i, namely $S - S_{C_i}$. The training data for building h_{C_i} are largely imbalanced because of the large number of negative instances.

However, if the class taxonomy structure is considered, the number of negative instances is reduced. This is due to the fact that we consider all instances labelled with a class label $parent(c_i)$ but not c_i to be negative instance for learning h_{C_i}. Consider the example using the class taxonomy in Figure 1. The following table shows the difference on negative instances. However, the positive instances will be the same.

Table 1. Negative Instances for Independent Classifiers and Binarization Classifiers

Classifiers	Independent	Binarization
h_A	$S - S_A$	$S - S_A$
h_B	$S - S_B$	$S - S_B$
h_C	$S - S_C$	$S - S_C$
h_D	$S - S_D$	$S_A - S_D$
h_E	$S - S_E$	$S_A - S_E$
h_F	$S - S_F$	$S_A - S_F$
h_G	$S - S_G$	$S_C - S_G$
h_H	$S - S_H$	$S_C - S_H$

3.2 Split-based Classifiers

One observation on multi label problem is that in real world applications it is very rare that we will have 2^M multi label combinations all appeared in the training data. The actual number of multi labels is much smaller than the possible number 2^M. There are two alternatives here, one is to set an upper limit number of possible class combinations. For example, we can set the number to be 2, such that we will only consider the combinations of two class labels instead of M class labels. Another option is to consider only the multi labels that appear in the training data.

However, the problem arises when we add those combinatorial multi labels to the original set of single class labels. We can not apply standard learning algorithms directly by extending multi labels as new class labels, because the multi label and the individual class labels are not mutually exclusive. Therefore, in order to apply standard learning algorithms, we need to generate mutually exclusive classes. We can generate one extended label out of each original class label, which only represents those instances with that original label only; Then we add new extended labels which each represents the instances labelled with 2 original class labels. For example, consider $\mathbf{C} = \{A, B, C\}$ with instances set S_A, S_B, S_C respectively. Suppose the only multi label observed in the training data is $\{A, B\}$.

Note that $S_A \bigcap S_B \neq \emptyset$. So the extended class set is $\mathbf{C}' = \{\hat{A}, \hat{B}, C, A\&B\}$, which represents instance set $S_A - S_A \bigcap S_B, S_B - S_A \bigcap S_B, S_C, S_A \bigcap S_B$.

This approach to transforming class labels to obtain mutually exclusive class labels can be applied to structured label problem by building split-based classifiers. We will first define an split in class taxonomy $\mathcal{CT}$, and then for each split we show how to learn a respective classifier by learning from instances with combinatorial extensions on class labels.

Definition 3 (Split). *A split is a one level subtree within a class taxonomy, which includes one parent node and all its children nodes, and the links between the parent node and children nodes.*

Obviously, the number of splits in the class taxonomy is smaller than the number of nodes. We can build a set of classifiers on the splits to solve structured label problem so to decrease the number of resulting classifiers. Within each split, the structured label problem will be reduced to a multi label problem, and we only need to consider the combinatorial extensions on class labels at that particular level. Additionally, the split-based classifiers also form partial orderings on the class taxonomy. Any instance to be classified will follow this topological order of the split-based classifiers: start from the classifier for the split at first position, continue to run a split-based classifier only when predicted to be "1" by the parent split-based classifier.

Taking the class taxonomy shown in Figure 1 as an example, we only need to build three classifiers because there are only three splits in the graph. We will build a classifier h_1 for the first split which includes the root and the nodes A, B, C and corresponding links, a classifier h_2 for the second split with nodes A, D, E, F and links, and a classifier h_3 for the third split with nodes C, G, H and links. Each classifier needs to work with the extended and mutually exclusive class labels, including some multi labels. For example if we have a multi label $A\&C$ occur at first level of the class taxonomy, then the classifier h_1 will be built on the class set $\{\hat{A}, B, \hat{C}, A\&C\}$. Accordingly, for predicting an instance with structured label, this instance is sent to h_1 firstly. Based on the classification result of h_1, the instance will be decided if it will be sent to lower level classifiers h_2 and/or h_3. For example, if an instance has been labelled with A by h_1, it will be sent to h_2 to decide its further labels. If an instance has been labelled with $A\&C$, it will be sent to both h_2 and h_3 for further classification.

4 Evaluation Method

4.1 Distance Metrics

In single label classification, a loss function $\mathbf{s}(c_p, c_o)$ can be defined for each test instance to evaluate the cost of misclassifying the instance with observed class label c_o to the predictive class label c_p, including the standard 0-1 loss function. However, in multi label or structured label problems, loss functions defined over single label problems are not suitable since now we need to measure the difference between two label sets $\mathbf{C}_p$ and $\mathbf{C}_o$. Each label set corresponds to a subtree

of the class taxonomy in structured label problem, and we transform the misclassification score defined over the two label sets to a distance measure between two subtrees and apply "label mapping" strategy to calculate the distance based on the so called "node distance" between two single class labels in class taxonomy. The node distance between two nodes in class taxonomy directly relates to the difference between two class labels. The longer the distance, the more the two labels differ.

Definition 4 (Node Distance). *Node distance is a value $d(c_i, c_j)$ denoting the difference of labels c_i, c_j. It has the following properties:*

- $d(c_i, c_j) \geq 0$
- $d(c_i, c_j) = d(c_j, c_i)$
- $d(c_i, c_i) = 0$

In order to describe the procedure to compute the score $s(\mathbf{C}_p, \mathbf{C}_o)$, we define the following notions.

Definition 5 (Dummy Label). *Dummy label θ is an"add-on" label to the class taxonomy which acts as a predicted value to the instance when a classifier can not decide the class label and does nothing. Thus this is a "label by default". It has the following properties:*

- $d(\theta, c_i) = d(\theta, c_j)$
- $d(c_i, c_j) \leq d(\theta, c_i)$

Definition 6 (Non-Redundant Operation). *A non-redundant operation (with Φ as the operator) to a label set $\mathbf{C}_i$ is to keep the children labels when both children labels and their parent labels are present, so that we eliminate the redundancy in the label set $\mathbf{C}_i$.*

Definition 7 (Mapping). *A mapping f between two label sets $\mathbf{C}_1, \mathbf{C}_2$ with the same cardinality is a bijection $f : \mathbf{C}_1 \rightarrow \mathbf{C}_2$.*

To calculate the score $s(\mathbf{C}_p, \mathbf{C}_o)$, there are two cases concerning the cardinality difference between $\mathbf{C}_p$ and $\mathbf{C}_o$:

- In case that their cardinality are equal, then find a mapping to minimize the score of the sum of node distance and average the sum over the cardinality.

- In case that their cardinality are not equal, then add dummy labels θ to the label set with fewer elements till their cardinality are equal (Note: in this scenario, one label set may have repetitive dummy labels), then calculate the score as the upper case.

Meanwhile, the optimal mapping between equal-cardinality label sets can be reduced to classic assignment problem and solved with Munkres algorithm[10] in polynomial time.

Procedure 1 score calculation $s(\mathbf{C}_p, \mathbf{C}_o)$

$\mathbf{C}_1 \leftarrow \Phi(\mathbf{C}_p)$
$\mathbf{C}_2 \leftarrow \Phi(\mathbf{C}_o)$
$m \leftarrow |\mathbf{C}_1|$
$n \leftarrow |\mathbf{C}_2|$
if $m > n$ **then**
    add m-n dummy labels θ to $\mathbf{C}_2$
end if
if $n >$m **then**
    add n-m dummy labels θ to $\mathbf{C}_1$
end if
$k \leftarrow \mathbf{max}(m, n)$
$s \leftarrow argmin_f \frac{\sum_{c_i \in \mathbf{C}_1} d(c_i, f(c_i))}{k}$ where f is a mapping between $\mathbf{C}_1, \mathbf{C}_2$
return **s**

After calculating the score of each instance in the testing data set $\mathbf{T}$, the average score $\bar{s} = \frac{\sum_{\mathbf{T}} s(\mathbf{C}_p, \mathbf{C}_o)}{|\mathbf{T}|}$ is taken to estimate the effectiveness of classifiers. The smaller the score, the more accurate the classifier.

Using the class taxonomy in Figure 1 as an example and defining node distances in table 2, we demonstrate how to calculate the score between two label sets as follows:

non-redundant operation: $\Phi(\{A, B, C, H\}) = \{A, B, H\}$

mapping between $\{A, B\}$ and $\{F, C\}$: $f(A) = F, f(B) = C$.

score calculation:

$$\begin{aligned} \mathbf{s}(\{A, B\}, \{F, C\}) &= \frac{d(A, F) + d(B, C)}{2} \\ &= 1.5 \\ \mathbf{s}(\{A, B, H\}, \{F, C\}) &= \mathbf{s}(\{A, B, H\}, \{F, C, \theta\}) \\ &= \frac{d(A, F) + d(B, \theta) + d(H, C)}{3} \\ &= 2 \end{aligned}$$

Table 2. node distances definition

	F	C	θ
A	1	2	4
B	3	2	4
H	3	1	5

5 Experimental Results

For our experiments on artificial data and mutant phenotype data, we implement our hierarchical learning methods using base classifier from Weka package[12]. For the experiment on Reuters-21578 data, we implement our learning approaches using bow[11] toolkit as subroutines.

Given a structured label data set, we need to decide the node distances between class labels. Generally, this could be specified by a domain expert. An alternative is to calculate them according to the training set as follows: for each level in the class taxonomy, we calculate the occurrence of classes in the training set, divide it by the number of labels in the same level of the the class taxonomy, then normalize them, thus we get the number to estimate the average weight of label in that level, which approximately denotes the importance extent of class labels in that level. Table $3, 4, 5$ list the number of labels and average weight in each level of the three data sets: artificial data, Reuters-21578 data and phenotype data respectively.

Table 3. number of labels and average weight in each level of artificial data

	level1	level2
#labels	3	5
average weight	1	0.465

Table 4. number of labels and average weight in each level of Reuters-21578 data

	level1	level2
#labels	2	7
average weight	1	0.268

Table 5. number of labels and average weight in each level of phenotype data

	level1	level2	level3	level4
#labels	13	99	60	26
average weight	1	0.148	0.068	0.038

Our calculation of the distance between class labels reflects the intuition that disagreements in labels at higher levels in CT(class taxonomy) should have a greater influence on the evaluation of the classifier. Then we calculate the

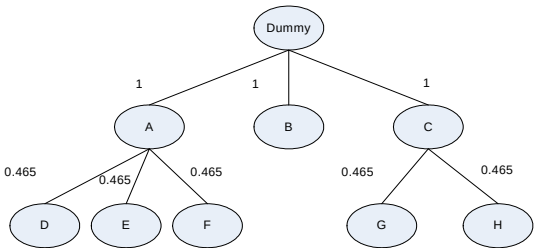

Fig. 2. edge distance of the artificial data class taxonomy

distance between class labels as follows: we place the "add-on" label θ in the root node of the class taxonomy tree and set the edge distance as the level weight. For two nodes, if one is ancestor of the other, the node distance will be the sum of the edge distance between them; if neither is ancestor of the other, the node distance will be the sum of their edge distance to their nearest common ancestor divided by 2. For example,in figure 2, $d(A, E) = 0.465$ and $d(D, C) = \frac{1+1+0.465}{2} = 1.233$. Here we can see this assigned node distance 1 to any two labels in the level 1 together with the "add-on" label θ, and the maximal node distance equals to the summation of all the level weights as 1.465 in artificial data set, 1.268 in Reuters-21578 data and 1.254 in phenotype data set.

5.1 Artificial Data Set

Because regular structured label data set is rare, we try our learning approaches in an artificial data set with the class taxonomy in Fig 1. The data set has 2 levels, 8 class labels as A, B, C, D, E, F, G, H and 8 binary value attributes as $A_1, ..., A_8$. An instance $I(A_1, ..., A_8)$ with label set $\mathbf{C}_I$ is labelled according to the following 8 logical rules:

- Initialize $\mathbf{C}_I = \emptyset$.
- If $A_1 = 1$ then add label A to $\mathbf{C}_I$.
- If $A_2 = 1$ then add label B to $\mathbf{C}_I$.
- If $A_3 = 1$ then add label C to $\mathbf{C}_I$.
- If $A_4 = 1$ and $A_1 = 1$ then add label D to $\mathbf{C}_I$.
- If $A_5 = 1$ and $A_1 = 1$ then add label E to $\mathbf{C}_I$.
- If $A_6 = 1$ and $A_1 = 1$ then add label F to $\mathbf{C}_I$.
- If $A_7 = 1$ and $A_3 = 1$ then add label G to $\mathbf{C}_I$.
- If $A_8 = 1$ and $A_3 = 1$ then add label H to $\mathbf{C}_I$.

For the integrity of the data set, instances with $A_1 = A_2 = A_3 = 0$ are not allowed. 172 instances are generated with uniform distribution.

We use C4.5 decision tree as the base classifier on the artificial data set with 3 fold cross validation. The results are listed in Table 6, 7. The resulting

average distance 0.134 denotes the split-based learning performs well enough on the artificial data set. It can almost predict correctly on the 1_{st} level. That's because the artificial data set is a ideally balanced data set and the data gives out enough information on the upper level of the class taxonomy. So hierarchical learning methods tend to predict better on the upper level. Binarization learning is less better because the binarization process will introduce some unbalance to the data set, especially to those class labels with less instances as class label B.

Table 6. average score: learning on artificial data set

	binarization learning	split-based learning
$\bar{s}$	0.846	0.134

Table 7. recall&precision: learning on artificial data set

	binarization learning		split-based learning	
	recall	precision	recall	precision
A	1.0	1.0	0.912	1.0
B	0.499	0.477	0.967	1.0
C	0.466	0.475	0.918	1.0
D	1.0	1.0	0.634	1.0
E	0.490	0.467	0.948	0.913
F	0.509	0.620	0.888	0.888
G	0.439	0.478	0.920	1.0
H	0.174	0.156	0.945	0.678

5.2 Reuters-21578 data Set

Reuters-21578 data, originally collected by Carnegie Group for text categorization, does not have a predefined hierarchical class taxonomy. However, many documents are labelled with multiple topic classes. We extract 670 documents with more than 72% documents having multiple class labels, and we create a two-level class taxonomy using current categories of the documents as follows:

grain(barley, corn, wheat, oat, sorghum)
livestock(l-cattle, hog)

We use Naive Bayes Classifier implemented in the bow toolkit [11] as the base classifier and test using 5 fold cross validation. The results in table 8, 9 demonstrate that binarization method performs as well as split-based method

Table 8. average score: learning on Reuters-21578 data set

	binarization learning	split-based learning
$\bar{s}$	0.217	0.251

Table 9. recall&precision: learning on Reuters-21578 data set

	binarization learning		split-based learning	
	recall	precision	recall	precision
grain	0.993	0.964	0.993	0.968
livestock	0.766	0.893	0.752	0.917
barley	0.498	0.440	0.454	0.442
wheat	0.852	0.735	0.859	0.724
corn	0.839	0.721	0.818	0.726
oat	0.270	0.75	0.167	0.75
sorghum	0.408	0.560	0.324	0.591
l-cattle	0.146	0.417	0.167	0.339
hog	0.729	0.786	0.717	0.686

in this case. Both have good predictive accuracy in the first level classes: grain, livestock. We note that distance metric which assigns equal distances between any node and its parents in the CT provides a coarse evaluation of the classifier owing to our definition of node distance. The average score between these two methods is slightly different, while the average recall and precision calculated over the entire class hierarchy are very close.

5.3 Phenotype Data Set

Phenotype data set is introduced by Clare and King[5, 6]. This data describes the reaction of mutant in various growth media. The data set has 1461 instances in total, every instance has 69 attributes, the former 68 attributes represents the reaction of the mutant in one dedicated growth media. Each attribute can take one of the four values: n-missing data; w-wild type; s-snesitive; r-resistance.

The data set has 84.41% missing data in the attributes values owing to the unavailability of the experiment data in corresponding growth media. The 69_{th} attribute, which we do not use, is the sum of s and r reactions. Each instance has a structured label, denoting the function class of the ORF, which is from the MIPS classification scheme. The class taxonomy of the data set is a hierarchical tree with 4 levels and 198 labels. For example, 30/0/0/0 is a 1_{st} label representing ORF with cell growth/division and DNA synthesis function; 30/16/0/0 is a child of 30/0/0/0 in the 2_{nd} level label representing ORF with mitochondrial organization function.

We choose the C4.5 decision tree and Naive Bayes classifier as the base classifier to run the binarization learning and split-based learning. We perform a 5-fold cross validation to get the average scores shown in table 10. Split-based

Table 10. average score: learning on phenotype data set

	binarization learning		split-based learning	
	Decision Tree	Naive Bayes	Decision Tree	Naive Bayes
$\bar{s}$	1.171	1.147	0.790	0.834

learning shows better performance than binarization learning on this data set. With C4.5 decision tree, the average score is 0.79. Thus, the split-based learning generally can predict 1 out of 4 class labels correctly in the 1_{st} level branches. Compared to the artificial data set and Reuters-21578 data set, the phenotype data set has a greater fraction of missing attribute values and much larger class label space. This might explain the fact that the results are not as good as those of artificial data set and Reuters-21578 data set.

We also calculate accuracy, recall and precision of each class labels. It turns out that the accuracy of each class label is quite high(95%). This is due to the fact that this data set is highly unbalanced and each hypothesis generates high true negative rate. Figures $3, 4$ present the recall and precision of each class label with binarization and split-based learning. Owing to the sparseness of the data set, many class labels do not appear in the testing data set. This leads to undefined recall and precision estimates because of division by 0. Hence, only those class labels with recall and precision estimates available are listed.

We can see that split-based learning can effectively learn about 20% class labels from phenotype data and nearly all the labels in the 1_{st} level are included. We notice that our classifiers generally do better in the upper levels of the class taxonomy. This is because statistical estimates at class labels that correspond to the upper levels of the CT are more reliable than those at lower levels. So paths from the root of the CT to nodes, classifiers that have high precision and recall values, such as 30/0/0/0-30/16/0/0 and 9/0/0/0-9/1/1/0 can be reliably annotated in practice.

Our results also show that split-based learning performs better in terms of recall and precision, which is consistent with the average score.

6 Conclusion and Future work

6.1 Summary

In this paper, we formalize structured label learning problem, and explore approaches that exploit the hierarchical structure over class labels in constructing classifiers.

Compared to single label classification problem, structured label problem with class taxonomy poses new challenges in terms of learning a classifier as well as evaluation of the result. Our main contributions in this paper are:

– We formalize the structured label learning problem as a generalization of single label and multi label problems.

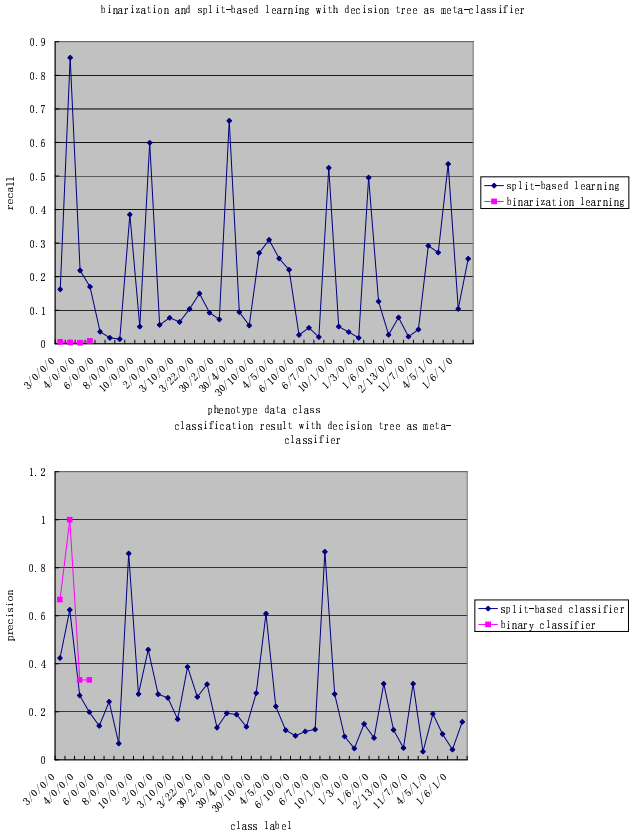

Fig. 3. recall&precision: learning on phenotype data set with decision tree

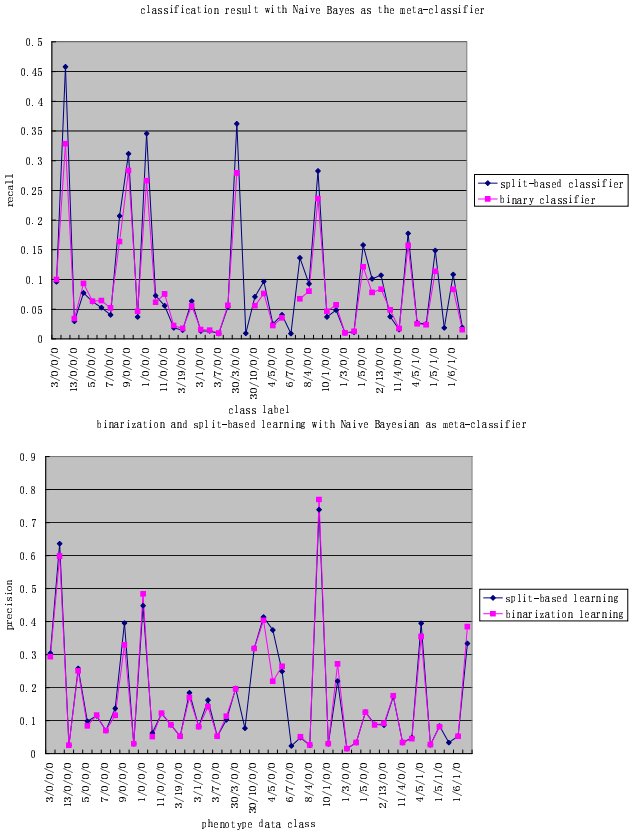

Fig. 4. recall&precision: learning on phenotype data set with Naive Bayes

- We describe two learning methods: binarization learning and split-based learning that directly exploit class taxonomy to build pattern classifiers.
- We discuss the evaluation of structured label learning problem and we propose a new strategy "label mapping" in distance metrics to evaluate structured label learning result.

In summary, for dealing with the structured label classification problem, we take into account of the structure constraints over class labels directly in constructing a classifier, and eliminate any class label assignments that violate the constraints imposed by class taxonomy. Our approaches are both simple and flexible, and can utilize any available classifier learning algorithm as base learner. We combine novel distance metric with the traditional means of precision and recall to evaluate the effectiveness of the learned classifiers. Experimental results show that our approach can effectively build classifiers from class taxonomy and data with structured labels. Empirical results also show our methods perform well on artificial data and Reuters-21578 data but need more refinement on sparse data sets like Phenotype data.

6.2 Related work

McCallum [1] built mixture model and trained it with EM algorithm to generate mixture weights of the model nodes, then used Bayes rules to do text classification. Although he considered the correlation between class labels, he did not take into account of the class taxonomy directly. Joachims[2] applied support vector machine to each class label in document classification which is similar to our implementation of binarization classification, but he treated each class label separately and did not combine the binary results with respect to the class taxonomy as we do. Kriegel et al [4] also used support vector machine, but their work was in predicting biological entities and emphasized in multiple data source integration. They trained a classifier for each class label, and they used classifier voting to decide the final prediction result and did not integrate the result together with the class taxonomy constraints. Wang et al [9] studied the document classification problem by constructing a single optimal classifier, but their work emphasized on the rules learning and cast the structured label classification as a flat, non-structured label classification. In scene classification, Shen et al [3] used support vector machine for each class label, and used maximum a posteriori(MAP) to do a cluster to get the final result, therefore the problem they studied is not a general structured label classification but a flat one. Blockeel et al [7, 8] induced a clustering tree (essentially a decision tree) by maximizing the distance between clusters. Their hierarchal multi label classification problem is very close to our definition of structured label classification. They regard each class as an vector and calculate the distance of vector to get the distance between classes, which is more complex than our "label-mapping" method. Clare and King [5, 6] proposed a modified C4.5 algorithm to solve the genome function classification by modifying the calculation of entropy of a data set with multiple classes. However, this work focused on decision tree and the

resulting classifier were evaluated primarily in terms of classifications relative to experimentally determined annotations.

6.3 Future work

Some directions for future work include:

- Development of algorithms to incorporate techniques for exploiting CT (class taxonomies) to handle partially specified class labels.
- Development of more sophisticated metrics for evaluation of structured label classification.

Acknowledgements: This research was supported in part by a grant from the National Institutes of Health (GM066387) to Vasant Honavar

References

1. A. McCallum "Multi label text classification with a mixture model trained by EM". AAAI'99 Workshop on Text Learning, 1999.
2. T,Joachims. "Text categorization with Support Vector Machines: Learning with many relevant features". In Machine Learning: ECML-98, Tenth European Conference on Machine Learning, pp. 137–142. 1998
3. X. Shen, M. Boutell, J. Luo, and C. Brown "Multi label Machine learning and its application to semantic scene classification", in Proceedings of the 2004 International Symposium on Electronic Imaging (EI 2004), Jan. 18-22, 2004
4. H.-P. Kriegel, P. Kroeger, A. Pryakhin, and M. Schubert "Using Support Vector Machines for Classifying Large Sets of Multi-Represented Objects", in Proc. 4th SIAM Int. Conf. on Data Mining, pp. 102-114, 2004
5. A. Clare and R. D. King "Knowledge Discovery in Multi label Phenotype Data", 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2001), volume 2168 of Lecture Notes in Arti cial Intelligence, pages 42-53, 2001
6. A. Clare and R. D. King "Machine learning of functional class from phenotype data", Bioinformatics 18(1) 2002 pp 160-166
7. H. Blockeel, M. Bruynooghe, S. Dzeroski, J. Ramon, and J. Struyf. "Hierarchical Multi-Classification", Proceedings of the First SIGKDD Workshop on Multi-Relational Data Mining (MRDM-2002), pages 21–35, July 2002
8. H. Blockeel, L. De Raedt, and J. Ramon "Top-down induction of clustering trees", In Proceedings of the 15th International Conference on Machine Learning, pages 55–63, 1998
9. K. Wang, S. Zhou, S.C. Liew, "Building hierarchical classifiers using class proximity", Technical Report, National University of Singapore, 1999
10. K. Wang, S. Zhou, S.C. Liew, J. Munkres, "Algorithms for the assignment and transportation problems" J. SIAM 5 (1957) 32-38.
11. A. McCallum, "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering", http://www.cs.cmu.edu/ mccallum/bow 1996.
12. "Data Mining: Practical machine learning tools with Java implementations," by Ian H. Witten and E. Frank, M. Kaufmann, San Francisco, 2000.
13. The Reuters-21578, Distribution 1.0 test collection is available from http://www.daviddlewis.com/resources/testcollections/reuters21578