Abstract: Multi label classification is variation of single label classification where each instance is associated with more than one class labels. Multi label classification is used in many applications like text classification, gene functionality, image processing etc. Hierarchical multi-label classification problems combine the characteristics of both hierarchical and multi-label classification problems. This paper introduced k binary classifier and one classifier approaches of hierarchical multi label classification. These approaches are explained with two algorithms to solve hierarchical multi label classification problems. One is the C4.5H algorithm (extension of multi label decision tree) and second is Predictive Clustering Tree (PCT) algorithm. From theoretical and experimental study on yeast data set shows that PCT algorithm is the best option for hierarchical multi label classification. PCT algorithm is implemented on Clus. This paper introduced three approaches of Clus: Single Classification (SC), Hierarchical Single Label Classification (HSC) and Hierarchical Multi label Classification (HMC). From theoretical and experimental study, HMC performs better compare to remaining two approaches.

Index Terms: Classification, Decision Tree, Hierarchical Classification, Multi Label Classification, Predictive clustering tree.

I. INTRODUCTION

In classification problems, each instance of a dataset is associated with just one class label. However, there are many classification tasks where each instance can be associated with one or more class labels. This group of problems represents an area known as Multi-Label Classification. Basically there are two ways for representation of multiple labels: flat structure and hierarchical structure. In flat structure all class labels are arranged at same level. In hierarchical structure all class labels are organized in hierarchy according to label correlation ships.

In Hierarchical Classification, classes are organized in a hierarchy: an example that belongs to some class automatically belongs to all its super classes [1]. Hierarchical Multi-label Classification (HMC) problems combine the characteristics of both hierarchical and multi-label classification problems: (1) class labels are organized in a hierarchical structure (e.g. a tree or DAG structure); (2) Examples may be associated with more than one class labels [2, 12]. Here tree structure is used for hierarchical structure.

Applications of hierarchical multi label classification are found in many areas, including text classification [9, 10], functional genomics, image recognition, face recognition, emotion detection, etc.

The rest of the paper is organized as follows. Section II explains approaches of hierarchical multi label classification. Decision tree for HMC and Multiple prediction tree for HMC discussed in Section III and IV respectively. Section V explains evaluation measures for HMC. Section VI shows experimental analysis on toyHMC and yeast dataset. At last current research challenges and conclusion presented in Section VII and VIII respectively.

II. APPROACHES FOR HMC

There are mainly two approaches for hierarchical multi label classification: k–Binary Classifier approach and Single Classifier approach.

A. K-BINARY CLASSIFIER

This approach transforms an HMC task into a separate binary classification task for each class in the hierarchy. This referred as k- Binary Classifier approach or Single classifier (SC) for each label. After transformation apply basic classification algorithm for classification.

\[
\begin{align*}
m1 & \quad m2 & \quad \ldots \ldots & \quad mk \\
\downarrow & \quad \downarrow & \quad \ldots \ldots & \quad \downarrow \\
c1 & \quad c2 & \quad \ldots \ldots & \quad ck
\end{align*}
\]

Figure I: Model for k-Binary Classifier

Binary classification has several drawbacks. It is less efficient, because the learner has to run dataset [L] times, with L is set of class labels, which can be hundreds or thousands. It often results in strongly skewed class distributions: in HMC applications classes at lower levels of the hierarchy often have very small frequencies, while the frequency of classes at higher levels tends to be very high. Many learners have problems with strongly skewed class distributions. Hierarchical relationships between classes are not considered. Means an instance belonging to a class must belong to all its super classes is not automatically imposed. The learned models identify features relevant for one class, rather than identifying features with all classes. [1, 3, 4]
A Comprehensive and Comparative Study on Hierarchical Multi Label Classification

B. ONE CLASSIFIER

This approach developed one single multi-label model which predicts multiple class labels at once. Means it predicts all the classes of examples at once.

\[
M
\]

\[
c1 \quad c2 \quad \ldots \ldots \quad ck
\]

Figure II: Model for One Classifier

This approach takes hierarchical constraint into account. It is also able to identify features that are relevant to all classes. [1, 3, 4]

Table I: Comparison of Hierarchical Multi-label Classification Approach

<table>
<thead>
<tr>
<th>k – Binary Classifier</th>
<th>One classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time consuming prediction</td>
<td>Faster to learn</td>
</tr>
<tr>
<td>Model interpretation is difficult</td>
<td>Easy to interpret model</td>
</tr>
<tr>
<td>Difficult to generate hierarchical constraint</td>
<td>Hierarchical constraint automatically imposed</td>
</tr>
</tbody>
</table>

III. DECISION TREE FOR HMC

In Multi-Label Classification, we have multiple class labels. But decision tree predict only one class label, not a set of labels. Using decision tree, separate tree will be created for each label.

\[
\text{Figure III Decision Tree for each individual label}
\]

According to above figure, for 250 labels separate decision tree will be created.

C4.5H ALGORITHM

In this algorithm tree is constructed in top-down manner. Best attribute is chosen by considering information gain. Information gain is a difference between the entropy of whole set of remaining training examples and the weighted sum of the entropy of the subsets caused by partitioning on the values of that attribute.

\[
\text{Information gain}(D, A) = \left\{ \begin{array}{l}
\text{entropy}(D) - \sum_{v \in A} \left[ \frac{|D_v|}{|D|} \cdot \text{entropy}(D_v) \right] \\
\text{if A is the attribute being considered, D is the set of training examples being considered, and Dv is the subset of D with value v for attribute A.}
\end{array} \right.
\]

(1)

\[
\text{Entropy (D)} = -\sum_{j=1}^{q} p(\lambda j) \log p(\lambda j) + q(\lambda j) \log q(\lambda j) \quad \text{Number of descendant classes of class } \lambda j
\]

(2)

Where,

\[
p(\lambda j) = \text{Relative frequency of class } \lambda j
\]

\[
q(\lambda j) = 1 - q(\lambda j)
\]

For generating hierarchy of classes several modifications are required: reading and storing the class hierarchy, testing for membership of a class, finding the best class or classes to represent node and performing entropy calculations.

A. READING AND STORING CLASS HIERARCHY:

The class hierarchy is read from a file, along with the data values and attributes names. We require the user to unambiguously specify the hierarchical relationships by requesting the hierarchy already in tree format. The format uses indentation by spaces to show the parent and child relationships, with children indented more than their parents.

B. TESTS FOR MEMBERSHIP OF A CLASS:

Testing a data item for membership of a class is now trivial, due to our class representation: simply check if the appropriate array element is set to “true” or not. Membership of parent classes was calculated once at the start, and is now explicit. When doing calculations which involve looking at a specific level in the hierarchy, we can use a Boolean mask to hide classes belonging to other levels. Since the class array consists of Booleans, simply AND-ing a Boolean array mask with the class array will show the classes of interest.

C. FINDING THE BEST CLASS OR CLASSES TO REPRESENT A NODE:

Nodes in the decision tree are labeled with the classes which best represent the data in each node. These classes are the classification which is to be made by following a path from the root down to that node. Since we are still dealing with the multi-label problem, there may be several classes used to label a node in the tree. When dealing with the class hierarchy we could choose to label the node with all appropriate classes from most specific to most general, or just the most specific classes.

D. ENTROPY CALCULATIONS:

To deal with hierarchies the entropy calculations again need to be modified. The basic entropy calculations developed for multi-label learning still apply, but we also need to consider the differences between levels of the hierarchy.

\[
\text{Entropy (D)} = -\sum_{j=1}^{q} p(\lambda j) \log p(\lambda j) + q(\lambda j) \log q(\lambda j) - \alpha(\lambda j) \log \text{treesize}(\lambda j)
\]

(3)

Where,

\[
p(\lambda j) = \text{Relative frequency of class } \lambda j
\]

\[
q(\lambda j) = 1 - q(\lambda j)
\]

\[
\alpha(\lambda j) = 0, \text{if } p(\lambda j) = 0
\]

\[
= \text{a user defined constant otherwise (default=1)}
\]

\[
\text{treesize}(\lambda j) = 1 + \text{Number of descendant classes of class } \lambda j
\]
MULTIPLE PREDICTION TREE VS. SEPARATE DECISION TREE

- Identify features with high relevance for multiple classes.
- Hierarchical constraint is maintained.
- More efficient than separate decision tree.
- Simpler models if the classes are not independent.
- Learning from skewed distribution.

IV. MULTIPLE PREDICTION TREE

Predictive Clustering Tree (PCT) can be constructed with a standard “Top-Down Induction of Decision Tree” algorithm, similar to CART and C4.5. PCT generates multiple prediction trees, which makes multiple predictions at once. PCT framework views decision tree as a hierarchy of clusters. Top node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters. Each split reduces intra-cluster variance. [4, 5]

The resulting tree using PCT algorithm:

```
    A = "a"
   /   |
  /     |
B = "b" N N
```

PCT: ALGORITHM

Input: Current training instances I
Output: Tree

Procedure: PCT (I)
1. \((t^*, p^*) = \text{BestTest}(I)\)
2. if \(t^* \neq \text{none}\)
3. for each \(I_k \in P^*\)
4. \(\text{tree}_k = \text{PCT}(I_k)\)
5. return node \((t^*, U_k\{\text{tree}_k\})\)
6. else
7. return leaf \((\text{Prototype}(I))\)

Procedure: BestTest (I)
1. \((t^*, h^*, p^*) = (\text{none}, 0, \emptyset)\)
2. for each possible test \(t\)
3. \(p = \text{partition induced by } t \text{ on } I\)
4. \(h = \text{Var}(I) - \sum_{h \in p} \frac{|I_h|}{|I|}\text{Var}(I_h)\)
5. if \((h > h^*) \wedge \text{Acceptable}(t, p)\)
6. \((t^*, h^*, p^*) = (t, h, p)\)
7. return \((t^*, p^*)\)

The algorithm takes as input a set of training instances I. The main method searches for the best acceptable attribute-value test that can be put in a node. If such a test \(t^*\) can be found then the algorithm creates a new internal node labeled \(t^*\) and calls itself recursively to construct a sub tree for each subset in the partition \(p^*\) induced by \(t^*\) on the training instances. To select the best test, the algorithm scores the tests by the reduction in variance. Maximizing variance reduction maximizes cluster homogeneity and improves predictive performance. If no acceptable test can be found, that is, if no test significantly reduces variance, then the algorithm creates a leaf and labels it with a representative case of the given instances. [4, 5]

Here toyHMC dataset is used to trace the PCT algorithm which generates hierarchical tree. This dataset has three attributes. From that one attribute is used to represent class hierarchy.

```
@relation simple
@attribute A {a,b}
@attribute B {1,2}
@attribute class hierarchical
rec/sport/swim, rec/sport/run, rec/auto

@data
a, 1, rec/sport/swim
a, 2, rec/sport/run
a, 2, rec/sport/run@rec/sport/swim
b, 1, rec/sport
b, 2, rec/auto
```

Figure VI Tree using PCT

Clus is a decision tree and rule induction system that implements the predictive clustering framework. This framework unifies clustering and predictive modeling and allows for a natural extension to more complex prediction settings such as multi-task learning and multi-label classification. It uses ideas of: C4.5 and CART. Most decision tree learners induce classification or regression trees, Clus generalizes this approach by learning trees that are interpreted as cluster hierarchies. We call such trees predictive clustering trees or PCTs. Depending on the learning task at hand, different goal criteria are to be optimized while creating the clusters, and different heuristics will be suitable to achieve this. [6]

Clus is written in Java programming language. So to run Clus, it suffices to install the Java Runtime Environment (JRE). Clus uses two input files and these are named filename.s and filename.arff. The file filename.s contains the parameter settings for Clus. The file filename.arff contains the training data to be read. The results of Clus run are put in an output file filename.out. [6]
There are three approaches of Clus: Clus-SC, Clus-HSC and Clus-HMC [1, 5]. Class Hierarchy: Figure VII(c) shows small class hierarchy. Class label names reflect the position in the hierarchy, e.g., ‘2.1’ is a subclass of ‘2’ [1].

**Clus-SC:**

In this approach, separate tree is created for each class label. Each of these trees is a single label binary classification tree. So this approach is an independent single label classification problem. The hierarchy introduces dependencies between class labels. But hierarchy is ignored by this approach. This approach has several disadvantages:

1. It is inefficient, because the learner has to be run |C| times, with |C| the total number of classes.
2. It often results in learning from strongly skewed class distribution: classes at lower levels of the hierarchy often have very small frequencies, while the frequency of classes at higher levels tends to be very high.
3. The knowledge discovery point of view, the learned models identify features relevant for one class, rather than identifying features with high overall relevance.

4. The hierarchical constraint is not taken into account, i.e. it is not automatically imposed that an instance belonging to a class should belong to all its super classes.

Figure VII(a) shows Clus-SC approach for the given class hierarchy (Figure VII(c)).

**Clus-HSC:**

This approach learns a separate tree for each hierarchical edge. For a non top-level class c, it holds that an instance can only belong to c if it belongs to c’s parent par(c). An alternative approach to learning a tree that directly predicts c, is therefore to learn a tree that predicts c given that the instance belongs to par(c). Learning such a tree requires fewer training instances: only the instances belonging to par(c) are relevant. The subset of these instances that also belong to c become the positive instances and the other instances the negative
instances. The resulting tree predicts the conditional probability P(c|par(c)). To make predictions for a new instance, we use the product rule P(c) = P(c|par(c)) P(par(c)). This rule applies the trees recursively, starting from the tree for a top-level class. Figure VII(b) shows Clus-HSC approach.

**Clus-HMC:**

This approach is to develop learners that learn a single multi-label model that predicts all the classes of an example at once. It considers hierarchy constraint into account. This approach is also able to identify features that are relevant to all classes. This approach follows PCT algorithm for hierarchical multi-label classification. Clus-HMC approach is shown in Figure VII(d) for given class hierarchy in Figure VII(c).

**Table II Comparing the three Clus approach**

<table>
<thead>
<tr>
<th>Measure</th>
<th>SC</th>
<th>HSC</th>
<th>HMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>-</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Hierarchical constraint</td>
<td>-</td>
<td>-</td>
<td>√</td>
</tr>
<tr>
<td>Identifying global features</td>
<td>-</td>
<td>-</td>
<td>√</td>
</tr>
</tbody>
</table>

**V. EVALUATION MEASURES**

Let D be a multi-label evaluation data set, consisting of |D| multi-label instances (x_i, y_i), i=1,...,|D|, y_i Y. Y= {1,2,...,k} is a set of labels. Let z_i be the set of labels predicted by classifier for example x_i. Three metrics are defined below. [1, 7]

Label Cardinality of D is the average number of labels of the instances in D.

\[ LC(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} |Y_i| \]  

(4)

Label Density of D is the average number of labels of the instances in D divided by |L|.

\[ LD(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i|}{|L|} \]  

(5)

Precision is the percentage of predicted labels that were correct.

\[ \text{Precision}(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Z_i|} \]  

(6)

Recall is the percentage of correct labels that were predicted.

\[ \text{Recall}(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i |}{|Z_i|} \]  

(7)

F-score is a combination of precision and recall. It is the harmonic average of the two metrics and it is used as an aggregated performance score.

\[ F - \text{score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]  

(8)

There is common characteristic for above measures, the larger measure value, and the better classification performance. The performance is perfect value when its value equals one.

Coverage is defined as the distance to cover all possible labels assigned to a sample x. It is loosely related to precision at the level of perfect recall. The smaller value of coverage is, the better the performance is.

\[ \text{Coverage} = \frac{1}{N} \sum_{i=1}^N \max_r(\lambda) - 1 \]  

(9)

Hierarchical Loss considers mistakes made at higher levels in the class hierarchy more important than mistakes made at lower levels.

\[ l_H = \sum_i [v_{k,i} \neq y_{k,i} \text{ and } \forall c_j \leq_h c_i : v_{k,j} = y_{k,j}] \]  

(10)

**VI. EXPERIMENT**

To simulate toyHMC.arff (Figure V) file on Clus, this generates hierarchy of class labels in one text document. Class hierarchy is given as below:

Hier #nodes: 5  
Hier classes by level: [1, 1, 2, 2]  
rec: 10  
auto: 2  
sport: 8  
run: 4  
swim: 4

**Figure VII hierarchy.txt file**

Below figure shows output file generated on Clus:

A = a  
++--yes: B = 1  
|  | +--yes: [1,0,1,0,9,0,2,0,7]: 1  
|  | +--no: [1.0,0666667,0.933333,0.8,0.466667]: 2  
|  | +--no: B = 1  
|  | +--yes: [1.0,1,0,9,0,2,0,2]: 1  
|  | +--no: [1.0,1,0,9,0,2,2,2]: 1

**Figure VIII toyHMC.out - Output file in Clus**

In table III, the data sets describe different aspects of the genes. (yeast dataset for biology) Each gene included in the datasets is annotated with one or more classes selected from the MIPS FunCat hierarchical classification scheme. In below table, |D| is the number of instances, and |A| number of attributes.

<table>
<thead>
<tr>
<th>Table III Dataset Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
</tr>
<tr>
<td>D1</td>
</tr>
<tr>
<td>D2</td>
</tr>
<tr>
<td>D3</td>
</tr>
<tr>
<td>D4</td>
</tr>
<tr>
<td>D5</td>
</tr>
<tr>
<td>D6</td>
</tr>
<tr>
<td>D7</td>
</tr>
<tr>
<td>D8</td>
</tr>
<tr>
<td>D9</td>
</tr>
<tr>
<td>D10</td>
</tr>
</tbody>
</table>
Below table shows comparison of Clus and C4.5H algorithm on yeast dataset mentioned in table III.

### Table IV

<table>
<thead>
<tr>
<th>Name</th>
<th>Clus</th>
<th>C4.5H</th>
<th>Clus</th>
<th>C4.5H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq</td>
<td>61</td>
<td>71</td>
<td>80.18</td>
<td>14.16</td>
</tr>
<tr>
<td>Phenol</td>
<td>67</td>
<td>68</td>
<td>3.09</td>
<td>3.26</td>
</tr>
<tr>
<td>Struc</td>
<td>68</td>
<td>58</td>
<td>29.71</td>
<td>2.67</td>
</tr>
<tr>
<td>Hom</td>
<td>64</td>
<td>55</td>
<td>81.41</td>
<td>12.06</td>
</tr>
<tr>
<td>Cellcycle</td>
<td>82</td>
<td>54</td>
<td>0.86</td>
<td>71.34</td>
</tr>
<tr>
<td>Church</td>
<td>75</td>
<td>53</td>
<td>8.18</td>
<td>58.64</td>
</tr>
<tr>
<td>Derisi</td>
<td>77</td>
<td>61</td>
<td>2.90</td>
<td>8.39</td>
</tr>
<tr>
<td>Eisen</td>
<td>88</td>
<td>48</td>
<td>5.73</td>
<td>37.63</td>
</tr>
<tr>
<td>gasch1</td>
<td>67</td>
<td>38</td>
<td>15.77</td>
<td>47.24</td>
</tr>
<tr>
<td>gasch2</td>
<td>96</td>
<td>60</td>
<td>3.09</td>
<td>64.06</td>
</tr>
<tr>
<td>Spo</td>
<td>79</td>
<td>46</td>
<td>3.70</td>
<td>12.82</td>
</tr>
<tr>
<td>Expr</td>
<td>77</td>
<td>75</td>
<td>27.28</td>
<td>5.56</td>
</tr>
</tbody>
</table>

### Figure IX

**Yeast Data set**

**Figure IX** Precision for Clus and C4.5H

### Figure X

**Yeast Data set**

**Figure X** Coverage for Clus and C4.5H

Table IV presents the obtained average precision and coverage for each data set. The average precision obtained by Clus is higher than C4.5H, this increased precision comes in most cases at the expense of lower coverage. The coverage of Clus is lower than C4.5H.

**VII. CURRENT RESEARCH CHALLENGES**

Following are the research challenges of the existing hierarchical multi label classification problem:

- Some of the instances are less informative because of irrelevant labels are existing in dataset. So try to extract only relevant label set from multi label set.
- To apply data preprocessing techniques such as pruning, feature selection etc. to improve the performance of multi label classification problem.

- One critical issue is to build hierarchical structure for large dataset with large number of attributes.
- In most of the cases instances are ignored who has missing values. So training instances are reduced.

**REFERENCES**


**AUTHORS PROFILE**

Purvi Prajapati has received her B.E degree in Information Technology from Sardar Patel University, Vidhyanagar, Gujarat, India in 2004. Since 2006 she has been with faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat, Where she is currently working as an Assistant Professor in the Department of Information Technology. She has joined M.Tech at Charotar University of Science and Technology, Changa, Gujarat, India in 2010. Her current research interest includes multi-label classification.
Amit Thakkar has received his B.E degree in Information Technology from Gujarat University, Gujarat, India in 2002 and master Degree from Dharmsinh Desai University, Gujarat, India in 2007. He has joined his Ph.D in the area of Multi relational Classification at KadiSarvavishvidhalayaUniversity, Gandhinagar, India in June 2010. Since 2002 he has been with faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat, Where he is currently working as an Associate Professor in the Department of Information Technology. He has published more than 20 research papers in the field of data mining and web technology. His current research interest includes Multi relational Data Mining, Relational Classification and Associate Classification.

Amit Ganatra has received his B.E degree in Computer Engineering from Gujarat University, Gujarat, India in 2000 and master Degree from Dharmsinh Desai University, Gujarat, India in 2004. He has joined his Ph.D in the area of Multiple Classifier System (InformationFusion) at KadiSarvavishvidhalayaUniversity, Gandhinagar, India in August 2008. Since 2000 he has been with faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat, Where he is currently working as an Associate Professor in the Department of Computer Engineering. He has published more than 50 research papers in the field of data mining and Artificial Intelligence. His current research interest includes Multiple Classifier System, Sequence Pattern Mining.