Expert Systems with Applications 28 (2005) 799–812

Expert Systems with Applications

www.elsevier.com/locate/eswa

MMDT: a multi-valued and multi-labeled decision tree classifier for data mining

Shihchieh Chou[a], Chang-Ling Hsu[b,*]

[a]Department of Information Management, National Central University, Taiwan, ROC
[b]Department of Computer Science and Information Management, Hungkuang University of Technology, 34 Chung-Chie Road, Shalu, Taichung County, 433 Taiwan, ROC

Abstract

We have proposed a decision tree classifier named *MMC* (multi-valued and multi-labeled classifier) before. MMC is known as its capability of classifying a large multi-valued and multi-labeled data. Aiming to improve the accuracy of MMC, this paper has developed another classifier named *MMDT* (multi-valued and multi-labeled decision tree). MMDT differs from MMC mainly in attribute selection. MMC attempts to split a node into child nodes whose records approach the same multiple labels. It basically measures the average similarity of labels of each child node to determine the goodness of each splitting attribute. MMDT, in contrast, uses another measuring strategy which considers not only the average similarity of labels of each child node but also the average appropriateness of labels of each child node. The new measuring strategy takes scoring approach to have a look-ahead measure of accuracy contribution of each attribute's splitting. The experimental results show that MMDT has improved the accuracy of MMC.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Multi-valued attribute; Multiple labels; Classification; Decision tree; Data mining

1. Introduction

The purpose of the decision tree classifier is to classify instances based on values of ordinary attributes and class label attribute. Traditionally, the data set is single-valued and single-labeled. In this data set, each record has many single-valued attributes and a given single-labeled attribute (i.e. class label attribute), and the class labels that can have two or more than two types are exclusive to each other or one another. Prior art decision tree classifiers, such as *ID3* (Quinlan, 1979, 1986), *Distance-based method* (Mantaras, 1991), *IC* (Agrawal, Ghosh, Imielinski, Iyer, & Swami, 1992), *C4.5* (Quinlan, 1993), *Fuzzy ID3* (Umano et al., 1994), *CART* (Steinberg & Colla, 1995), *SLIQ* (Mehta, Agrawal, & Rissanen, 1996), *SPRINT* (Shafer, Agrawal, & Mehta, 1996), *Rainforest* (Gehrke, Ramakrishnan, & Ganti, 1998) and *PUBLIC* (Rastogi & Shim, 1998), all focus on this single-valued and single-labeled data set.

However, there is multi-valued and multi-labeled data in the real world as shown in Table 1. Multi-valued data means that a record can have multiple values for an ordinary attribute. Multi-labeled data means that a record can belong to multiple class labels, and the class labels are not exclusive to each other or one another. Readers might have difficulties to distinguish multi-labeled data from two-classed or multi-classed data mentioned in some related works. To clarify this confusion, we discuss the exclusiveness among classes, number of class and representation of the class label attribute in the related works as follows:

1. *Exclusiveness:* Each data can only belong to a single class. Classes are exclusive to one another. ID3, Distance-based Method, IC, C4.5, Fuzzy ID3, CART, SLIQ, SPRINT, Rainforest and PUBLIC are such examples.
2. *Number of class:* Data with classes classified into two types in the class label attribute is called two-classed data. ID3 and C4.5 are such examples. Data with classes

* Corresponding author. Tel.: +886 4 2631 8652x5400; fax: +886 4 2652 1921.
E-mail addresses: scchou@mgt.ncu.edu.tw (S. Chou), johnny@sunrise.hk.edu.tw (C.-L. Hsu).

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2004.12.035

Table 1
A training data set for 15 products

Id	Maker	Price	Performance	Color	Class label
p1	A	\$100	Not good	Yellow	A, B, C
p2	B	\$880	Good	Yellow	B, C
p3	A	\$370	Not good	Yellow, green	A
p4	C	\$1230	Good	Blue	B
p5	B	\$910	Good	Yellow, blue	B, C
p6	B	\$770	Not good	Yellow	A, B, C
p7	B	\$590	Not good	Yellow, green	A, B
p8	C	\$1350	Good	Green	A, B, C
p9	C	\$1250	Good	Yellow, green	A, B, C
p10	B	\$1140	Good	Yellow, green	A
p11	A	\$340	Not good	Yellow, blue	A, C
p12	C	\$1300	Good	Yellow	A, B
p13	B	\$1090	Good	Blue	C
p14	B	\$810	Good	Green	A
p15	B	\$520	Not good	Yellow, blue, green	C

classified into more than two types in the class label attribute is called multi-classed data. IC, CART and Fuzzy ID3 are such examples.

3. *Label representation:* Data with a single value for the class label attribute is called single-labeled data. ID3, Distance-based Method, IC, C4.5, Fuzzy ID3, CART, SLIQ, SPRINT, Rainforest and PUBLIC are such examples.

According to the discussion above, a multi-valued and multi-labeled data as we defined here can be regarded as a non-exclusive, multi-classed and multi-labeled data.

In our previous work (Chen, Hsu, & Chou, 2003), we have explained why the traditional classifiers are not capable of handling this multi-valued and multi-labeled data. To solve this multi-valued and multi-labeled classification problem, we have designed a decision tree classifier named *MMC* (Chen et al., 2003) before. MMC differs from the traditional ones in some major functions including growing a decision tree, assigning labels to represent a leaf and making a prediction for a new data. In the process of growing a tree, MMC proposes a new measure named *weighted similarity* for selecting multi-valued attribute to partition a node into child nodes to approach perfect grouping. To assign labels, MMC picks the ones with numbers large enough to represent a leaf. To make a prediction for a new data, MMC traverses the tree as usual, and as the traversing reaches several leaf nodes for the record with multi-valued attribute, MMC would union all the labels of the leaf nodes as the prediction result. Experimental results show that MMC can get an average predicting accuracy of 62.56%.

Having a decision classifier developed for the multi-valued and multi-labeled data, this research steps further to improve the classifier's accuracy. Considering the following over-fitting problems (Han & Kamber, 2001; Russell & Norving, 1995) of MMC, improvement on its predicting accuracy seems possible. First, MMC neglects to avoid the situation when the data set is too small. Therefore, it may choose some attributes irrelevant to the class labels. Second, MMC appears to prefer the attribute which splits into child nodes with larger similarity among multiple labels. Therefore, MMC exists inductive bias (Gordon & Desjardins, 1995).

Trying to minimize the over-fitting problems above, this paper proposes solutions as: (1) Set a constraint of size for the data set in each node to avoid the data set being too small. (2) Consider not only the average similarity of labels of each child node but also the average appropriateness of labels of each child node to decrease the bias problem of MMC.

Based on the propositions above, we have designed a new decision tree classifier to improve the accuracy of MMC. The decision tree classifier, named *MMDT* (multi-valued and multi-labeled decision tree), can construct a multi-valued and multi-labeled decision tree as Fig. 1 shows.

The rest of the paper is organized as follows. In Section 2, the symbols will be introduced first. In Section 3, the tree construction and data prediction algorithms are described. In Section 4, the experiments are presented. And, finally, Section 5 makes summaries and conclusions.

2. Notation

The symbols for the multi-valued and multi-labeled classification problem are formally stated as follows:

(a) Given that D is a training set, $|D|$ denotes the number of records.
(b) C denotes a set of class labels, $C=\{C_i|C_i$ is a class label, $i=1,\ldots,k\}$. The number of class labels in C is known in advance. $|C|$ denotes the number of class labels k.
(c) A denotes a set of attributes, $A=\{A_i|A_i$ is any ordinary attribute of D, $i=1,\ldots,n\}$. $|A|$ denotes the number of attributes n.

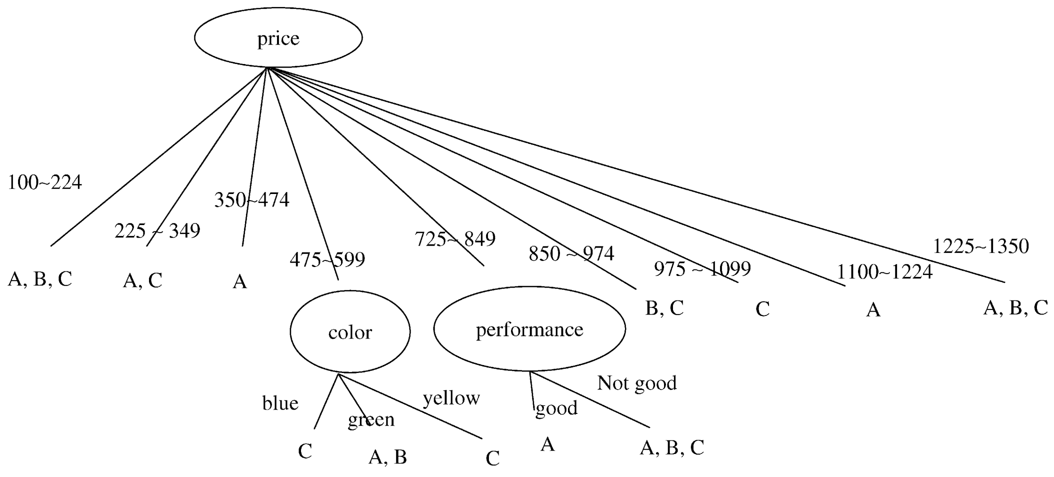

Fig. 1. A multi-valued and multi-labeled decision tree.

(d) 'Multiple labels' is represented as label-set. L_j denotes a label-set to represent a set of labels in C, $L_j=\{C_1^*, C_2^*, \ldots, C_k^*\}-\phi$, where $*=0$ or 1, $C_k^0=\phi$, $C_k^1=C_k$, $1\leq j\leq 2^k-1$.
(e) L denotes a set of label-sets which could be single-labeled or multi-labeled. $L=\{L_j|L_j\subseteq 2^C\}$.
(f) $T(V,E)$ denotes a rooted and multi-degreed decision tree in which V is a set of nodes and E is a set of branches. Each internal node of T contains numeric and non-numeric attributes, and each leaf node of T contains 'multiple labels' which belongs to L.
(g) To restrict the size of the tree, we define ub as a user-specified degree-restricted parameter, such that $2\leq \text{Degree}(v_i)\leq \text{ub}$, where internal node $v_i\in V$ and v_i contains a numeric attribute.

3. The algorithms

The development of MMDT is based on MMC (Chen et al., 2003). For completeness, we briefly describe the related part of the MMC algorithm in Section 3.1 first. Then, we present the MMDT algorithm in Section 3.2. Finally, we describe how MMDT makes a prediction and how to evaluate the accuracy of the prediction in Section 3.3.

3.1. The MMC algorithm

We have designed the MMC algorithm (Chen et al., 2003) before to construct a multi-valued and multi-labeled decision tree. It follows the standard framework adopted by the classical classification methods such as ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), MIND (Wang, Iyer, & Vitter, 1998), IC (Agrawal et al., 1992), SLIQ (Mehta et al., 1996) and SPRINT (Shafer et al., 1996). Fig. 2 explains how MMC goes.

In Fig. 2, steps 4–6 determine the internal node and its branches for a tree; step 10 determines a leaf node. In the following, we have further clarified them.

3.1.1. To determine the internal node and its branches

For a non-STOP node, MMC has to select the most discriminatory attribute for splitting and partitioning its intervals. In the following, we discuss how MMC partitions the data set into subsets according to the intervals they fall first. Then, we discuss how MMC selects the best splitting attribute.

Suppose that MMC partitions a data set according to a numeric attribute A_l. Let D_{CN} denote the data set stored in node CN, let the number of D_{CN} be n, and let the jth percentile value of the attribute A_l in D_{CN} be denoted as percentile(A_l,j) after sorting D_{CN} by the attribute A_l as an ascending sequence. Then, MMC can partition node CN into at most k intervals according to the attribute A_l by the split-intervals function as Fig. 3 shows.

Viewing an interval as a value range of a numeric attribute or a single value of a categorical attribute, we can partition D_{CN} into k subsets according to the interval they fall. Let $|D_{\text{CN}}(i)|=n_i$, where $D_{\text{CN}}(i)$ denotes the ith subset, $i=1,\ldots,k$, and let $n'=\sum_{i=1}^{k} n_i$. Then, we have $n'\geq n$ as an attribute may have multiple values and a data record can belong to multiple intervals.

To measure the goodness of this splitting, MMC defines the measure of weighted similarity of the splitting of attribute A_l on node CN by the following equation

$$\text{weighted-similarity}(\text{CN}, A_l, k) = \sum_{i=1}^{k} \text{set-similarity}_i \times \frac{n_i}{n'}, \tag{1}$$

where set-similarity(L) is the similarity among a set of label-sets $L=\{L_1,L_2,\ldots,L_m\}$.

Many label-sets in L may be the same, and hence the original L can be rewritten as the form

```
1.  MMC (Training Set D)
2.  Initialize tree T and put all records of D in the root;
3.  While (some leaf in T is not a STOP node)
4.      For each attribute i of each non-STOP node do
5.          For each split value of attribute i do
6.              Evaluate how good of this splitting of attribute i;
7.      For each non-STOP leaf do
8.          Get the best split for it;
9.          Partition the records and grow the tree for one more level according to the
            best splits;
10.     Identify the nodes that can be stopped, and mark them as STOP nodes and
        determine their result label-set;
11. Return T
```

Fig. 2. The MMC algorithm.

$\{(NL_1,\text{count}_1),(NL_2,\text{count}_2),\ldots,(NL_r,\text{count}_r)\}$, where NL_i denotes the ith distinctive label-set and count_i is the number of label-sets in L whose label-sets are NL_i. Set-similarity(L) is calculated by the following equation

$$\text{set-similarity}(L) = \frac{\sum_{i=1}^{r} C_2^{\text{count}_i} + \sum_{i<j} \text{count}_i \times \text{count}_j \times \text{similiarity}(NL_i,NL_j)}{m(m-1)/2}, \tag{2}$$

where $m \neq 1$, and similarity(NL_i,NL_j) denoting similarity between NL_i and NL_j is calculated by the following equation

$$\text{similarity}(NL_i,NL_j) = \left(\frac{\text{same}(NL_i,NL_j)}{\text{cardinality}(NL_i,NL_j)} - \frac{\text{different}(NL_i,NL_j)}{\text{cardinality}(NL_i,NL_j)} + 1\right)\Big/ 2, \tag{3}$$

where same(NL_i,NL_j) is the number of labels that appear in both NL_i and NL_j, different(NL_i,NL_j) is the sum of the number of labels that appear in NL_i only and the number of labels that appear in NL_j only, cardinality(NL_i,NL_j) is the number of distinctive labels that appear in NL_i and NL_j.

Table 2, called *similarity table* here, shows the similarity measures for all the label-sets obtained from C_1, C_2 and C_3 in MMC. The similarity table has been pre-computed for calculation performance.

Based on the measure of weighted similarity, MMC can select the best splitting attribute for a node CN as Fig. 4 shows.

3.1.2. To determine the leaf node

Let $C=\{C_1,C_2,\ldots,C_k\}$ be a set of all the class labels in training set D. Let $d_1,d_2,\ldots,d_r$ be the data records associated with the current node CN, and let $L_1,L_2,\ldots,L_r$ be their label-sets, respectively, where $L_i \subset 2^C$ for each i. Then, the term support of class label C_i can be defined as the percentage of data records in CN that contains C_i. If the support of a class label C_i is greater than or equal to the user-specified minimum support (called minsup), we call C_i a large label. Otherwise, we call it a small label. Therefore, all the class labels in CN can be classified into two sets: small(CN) and large(CN), where small(CN) contains all the small labels in CN and large(CN) contains all the large labels in CN.

```
1. Split-intervals(CN, $A_l$, k)
2. Let $b_1$ denote the smallest value of attribute $A_l$ in CN;
3. left = $b_1$-2Δ;   /Here, Δ denotes a very small number./
4. For j = 1 to k do
5.   index = (j/k) ×100%;
6.   right = percentile($A_l$, index);
7.   If left = right then skip this iteration;
8.   Else generate interval [left+Δ, right];
9. Endfor
```

Fig. 3. The split-intervals function.

Table 2
The similarity table for all the label-sets obtained from C_1, C_2 and C_3

	C_1	C_2	C_3	C_1, C_2	C_1, C_3	C_2, C_3	C_1, C_2, C_3
C_1	1	0	0	1/2	1/2	0	1/3
C_2	0	1	0	1/2	0	1/2	1/3
C_3	0	0	1	0	1/2	1/2	1/3
C_1, C_2	1/2	1/2	0	1	1/3	1/3	2/3
C_1, C_3	1/2	0	1/2	1/3	1	1/3	2/3
C_2, C_3	0	1/2	1/2	1/3	1/3	1	2/3
C_1, C_2, C_3	1/3	1/3	1/3	2/3	2/3	2/3	1

Further, we define the term difference of node CN as

$$\text{difference(CN)} = \min\{\text{support}(C_i)|C_i \text{ in large(CN)}\} - \max\{\text{support}(C_i)|C_i \text{ in small(CN)}\}.$$

If difference of a node CN is greater than or equal to the user-specified minimum difference (called mindiff), then we call CN a clear node. Otherwise, we call CN an unclear node.

CN will continue to grow until matching one of the following conditions:

1. If node CN is clear, then assign all labels in large(CN) as its result label-set.
2. If node CN is unclear and all the attributes have been used up in the path from root down to CN, then
   2.1 If large(CN) is not empty, then assign all labels in large(CN) as its result label-set.
   2.2 If large(CN) is empty, then assign the label with the maximum support as its result label-set.
3. If node CN is unclear and the number of data records is smaller than the user-specified minimum quantity (called minqty), then deal it the same way as in conditions 2.1 and 2.2.

3.2. Label ratio and the MMDT algorithm

Based on MMC, we have further developed MMDT algorithm. The main idea is to refine MMC's measure of the goodness of splitting attribute. MMDT has developed a new measure called weighted label ratio, which chooses the attribute that can partition data set into better label representations by measuring ratio of each label-set representing a subset. The weighted label ratio in MMDT works together with MMC's weighted similarity to measure the goodness of splitting the attribute. In the following, we will describe the label ratio approach first in Section 3.2.1, and then describe the MMDT algorithm in Section 3.2.2.

3.2.1. Label ratio

In the process of constructing a tree, each node continues to choose a label-set to represent itself to approach perfect grouping. The measure of the appropriateness of each label-set representing a node is called label ratio, denoted as R_{L_j}, and calculated by the following equation

$$R_{L_j} = S_{L_j}/|D|, \tag{4}$$

where S_{L_j} is the label score of L_j (label score is the score given to L_j about the degree of representing the node D) and calculated by the following equation

$$S_{L_j} = \vec{\mathbf{D}} \cdot \vec{\mathbf{S}}_{L_j} = (|D^{L_1}|, \ldots, |D^{L_j}|, \ldots, |D^{L_{2^k-1}}|) \cdot (\text{Sim}^{L_jL_1}, \ldots, \text{Sim}^{L_jL_i}, \ldots, \text{Sim}^{L_jL_{2^k-1}}) = \sum_{i=1}^{2^k-1} |D^{L_i}|\text{Sim}^{L_jL_i}, \tag{5}$$

with

$\vec{\mathbf{D}}$	being a vector of label number.
$\vec{\mathbf{S}}_{L_j}$	being a vector of label similarities of L_j.
D^{L_i}	being a subset of D whose label-set is L_i.
$\|D^{L_i}\|$	being the cardinality of D^{L_i}.

```
1.  Next_attribute(CN)
2.  For each attribute i do
3.      If it is a categorical attribute then compute its weighted similarity;
4.      If it is a numeric attribute then compute $\overset{ub}{\underset{k=2}{Max}}$ weighted similarity;
5.  Endfor
6.  Choose the attribute and split which can get the largest weighted similarity;
```

Fig. 4. The next_attribute function of MMC.

Table 3
Appearance-based label similarity of {A, B}

L_j	L_i						
A, B (L_4)	A (L_1)	B (L_2)	C (L_3)	A, B (L_4)	B, C (L_5)	A, C (L_6)	A, B, C (L_7)
Consistent (L_j, L_i)	2	2	0	3	1	1	2
Inconsistent (L_j, L_i)	1	1	3	0	2	2	1
$\mathrm{Sm}^{L_jL_i}$ (un-normalized)	1.5	1.5	0.25	4	0.667	0.667	1.5
$\mathrm{Sm}^{L_jL_i}$ (normalized)	0.33	0.33	0	1	0.11	0.11	0.33

$\mathrm{Sim}^{L_jL_i}$ (formally stated in Section 3.2.1.1) being the label similarity between L_j and L_i.

3.2.1.1. Label similarity. Label similarity is the measurement of the similarity between two label-sets. There are two measuring strategies: appearance basis and similarity basis.

(1) *Appearance basis*. Let L_j and L_i be the label set, $\mathrm{Sim}^{L_jL_i}$ be the label similarity between L_j and L_i. $\mathrm{Sim}^{L_jL_i}$ is derived by calculating and normalizing $\mathrm{Sm}^{L_jL_i}$ as follows

$$\mathrm{Sm}^{L_jL_i} = \frac{1+\mathrm{consistent}(L_j,L_i)}{1+\mathrm{inconsistent}(L_j,L_i)}, \tag{6}$$

where consistent(L_j,L_i) is the sum of the number of labels of the class set that appear in both L_j and L_i and the number of labels of the class set that do not appear in both L_j and L_i, $i=1\ldots2^k-1$; inconsistent(L_j,L_i) is the sum of the number of labels that appear in L_j only and the number of labels that appear in L_i only, $i=1\ldots2^k-1$.

$\mathrm{Sm}^{L_jL_i}$ is then normalized to be $\mathrm{Sim}^{L_jL_i}$ whose value falls in [0…1] by the following equation

$$\mathrm{Sim}^{L_jL_i} = \frac{\mathrm{Sm}^{L_jL_i}-\mathrm{Minscore}(L_j)}{\mathrm{Maxscore}(L_j)-\mathrm{Minscore}(L_j)}, \tag{7}$$

where Maxscore(L_j) is the maximum score of L_j among each un-normalized $\mathrm{Sm}^{L_jL_i}$, $i=1\ldots2^k-1$; Minscore(L_j) is the minimum score of L_j among each un-normalized $\mathrm{Sm}^{L_jL_i}$, $i=1\ldots2^k-1$.

In Eq. (6), inconsistent is a penalty for consistent to avoid that more labels get more score.

(2) *Similarity basis*: Let L_j and L_i be the label set, $\mathrm{Sim}^{L_jL_i}$ be the label similarity between L_j and L_i. $\mathrm{Sim}^{L_jL_i}$ equals to similarity(L_i,L_j) calculated by Eq. (3) in MMC.

3.2.1.2. Label ratio calculation. Following is an example to demonstrate the calculation of label ratio based on appearance basis. Suppose a node D has five records. The label-set of each record belongs to a set L, $L=\{\{B\},\{C\},\{C\},\{A,C\},\{B,C\}\}$. Label ratio based on appearance basis of the label-set {A,B} can be calculated according to the following steps.

Step 1: Getting label similarity. We first determine the consistent and inconsistent value of the label set {A,B} (see Eq. (6) for definition). For example, for $L_j=\{A,B\}$ and $L_i=\{B\}$, consistent ({A,B}, {B}) equals to 2 as they both have 'B' and both do not have 'C', and inconsistent ({A,B}, {B}) equals to 1 as only label 'A' appears in L_j only; for $L_j=\{A,B\}$ and $L_i=\{A,C\}$, consistent ({A,B}, {A,C}) equals to 1 as they both have 'A', and inconsistent ({A,B}, {A,C}) equals to 2 as label 'B' appears in L_j only and label 'C' appears in L_i only. Then, each $\mathrm{Sm}^{L_jL_i}$ can be derived from Eq. (6) and each $\mathrm{Sim}^{L_jL_i}$ can be derived from Eq. (7). Table 3 has shown the calculating result. For the consideration of calculation performance, the reference table of appearance-based label similarity can be built in advance to reduce the computing time as shown in Table 4.

Step 2: Calculating label score. We first get the vector of label number in the node: $\vec{\mathbf{D}}=(|D^{L_1}|,|D^{L_2}|,\ldots,|D^{L_7}|)=(0,1,2,0,1,1,0)$. Label score of {A,B} (i.e. $S_{\{A,B\}}$) is calculated by $\vec{\mathbf{D}}\cdot\vec{S}_{\{A,B\}}=(0,1,2,0,1,1,0)\cdot(0.33,0.33,0,1,0.11,0.11,0.33)=0.55$.

Step 3: Calculating label ratio. Label ratio of {A, B} (i.e. $R_{\{A,B\}}$) is calculated by $S_{\{A,B\}}/|D|=0.55/5=11\%$.

3.2.2. The MMDT algorithm

The MMDT algorithm, depicted in Fig. 5, is designed to construct a multi-valued and multi-labeled decision tree.

Table 4
The reference table of appearance-based label similarity

Label L_i	Label L_j						
	A (L_1)	B (L_2)	C (L_3)	A, B (L_4)	B, C (L_5)	A, C (L_6)	A, B, C (L_7)
A (L_1)	1	0.11	0.11	0.33	0	0.33	0
B (L_2)	0.11	1	0.11	0.33	0.33	0	0
C (L_3)	0.11	0.11	1	0	0.33	0.33	0
A, B (L_4)	0.33	0.33	0	1	0.11	0.11	0.25
B, C (L_5)	0	0.33	0.33	0.11	1	0.11	0.25
A, C (L_6)	0.33	0	0.33	0.11	0.11	1	0.25
A, B, C (L_7)	0.11	0.11	0.11	0.33	0.33	0.33	1

```
Input: $D$ is a data set, and $A$ is an attribute set.
Output: A multi-valued and multi-labeled decision tree $T$.
MMDT ($D$, $A$)
1.  If $D$ is empty Then return NULL;
2.  If $D$ satisfies one of the following stop conditions:
    2.1.  difference($D$)=min{support($C_i$) | $C_i$ in large($D$)} – max{support($C_i$) | $C_i$ in
          small($D$)} > mindiff, and |$D$| >= minqty;
    2.2.  $|D| \neq 0$, and $A$ is empty;
    2.3.  $|D| \neq 0$, and $|D| < minqty$;
      Then
      2.3.1. If large($D$) is not empty Then assign all labels in large($D$) as its result
             label-set, and return $T$;
      2.3.2. If large($D$) is empty Then assign the label with the maximum support as
             its result label-set, and return $T$;
3.  ($A_i$, $k$) = next_attribute($A$, $D$);
4.  $T$ = $T$ with the growth of node $A_i$;
5.  $I$ = split-intervals($D$, $A_i$, $k$);
6.  For (each interval $I_j \in I$ of $A_i$, $j$ = 1.. $k$) loop
7.      $D^j_{A_i,k}$ = partitioning $D$ into the $j$-th child subset based on $A_i$ in the interval $I_j$;
8.      aSubtree = MMDT ( $D^j_{A_i,k}$ , $A - \{A_i\}$);
9.      $T$ = adding a branch labeled interval $I_j$ to connect $T$ with aSubtree;
10. $j = j+1$;
11. End loop;
12. Return $T$;
```

Fig. 5. The MMDT algorithm.

Initially, MMDT is given a training data set D with values of attributes sorted for analyzing. In step 1, it judges whether D is empty. In step 2, if D satisfies one of the three conditions in step 2.1 through step 2.3, then if large(D) (referring to Section 3.1.2) is not empty, MMDT will assign all labels in large(D) as its result label-set and stop growing; otherwise, MMDT will assign the label with the maximum support as its result label-set and stop growing.

If D does not satisfy any stop conditions in step 2, the function next_attribute (discussed in Section 3.2.2.1) in step 3 chooses the attribute A_i from the attribute set A as the next splitting attribute and finds the best interval number k. In step 4, the tree T grows a node A_i. In step 5, split-intervals(D,A_i,k) partitions D into at most k intervals (sub-domains) according to the attribute A_i. Steps 6–11 which is a loop recursively calls MMDT to grow the tree. In step 12, MMDT returns the decision tree T.

3.2.2.1. Function Next_attribute. The main function of function next_attribute is to choose the next splitting attribute by measuring the goodness of each attribute. The measurement ‘similarity ratio’ is formed by two sub-measurements ‘weighted similarity’ and ‘weighted label ratio’ (discussed in Section 3.2.2.1.1). Weighted similarity (referring to Section 3.1.1) chooses the attribute for partitioning a data set into more similar subsets. Weighted label ratio chooses the attribute for partitioning a data set into subsets with appropriate label-sets.

Function next_attribute is depicted in Fig. 6. Steps 1–3 are a loop to compute similarity ratio and determine the number of interval for each attribute. If the attribute is numeric, then it sums the two measurements of each interval of A_i and assigns the maximum value of the sum as the similarity ratio. It also assigns the number of interval of the similarity ratio as K_i (the best number of interval of A_i). If the attribute is categorical, then it assigns the sum of the two sub-measurements as the similarity ratio and the number of interval of A_i as K_i. After calculating the similarity ratio for each attribute, step 4 will be able to choose the best attribute A_m which has the maximum similarity ratio. In step 5,

```
    Input: $A$ is an attribute set, and $D$ is a data set.

    Output: A pair (the best attribute $A_m$, and the number of interval of $A_m$).

    Next_attribute($A$, $D$)
1.  For each attribute $A_i$ do
2.  If $A_i$ is a numeric attribute
      Then similarity-ratio($A_i$, $D$) = $\underset{v=2}{\overset{ub}{Max}}$ (weighted-similarity($D$, $A_i$, $v$) + weighted-label-ratio($A_i$, $v$, $D$));
           $K_i = v$ of similarity-ratio($A_i$, $D$);
      Else similarity-ratio($A_i$, $D$) = weighted-similarity($D$, $A_i$, $v$) + weighted-label-ratio($A_i$, $v$, $D$);
           $K_i = v$;
3.  EndFor
4.  $A_m = \underset{i=1}{\overset{n}{Max}}$ similarity-ratio($A_i$, $D$);
5.  If similarity-ratio($A_m$, $D$) $\neq 0$
      Then return ($A_m$, $v$ of $A_m$);
      Else return NULL;
```

Fig. 6. The next_attribute function of MMDT.

if similarity ratio is not equal to zero, then function next_attribute returns A_m and A_m's best number of interval; otherwise, it returns NULL.

Function weighted-label-ratio. The main function of function weighted-label-ratio is to measure the average appropriateness of labels of each subsets after partitioning D into v subsets by splitting the attribute A_i. Function weighted-label-ratio is calculated by the following equation

$$\text{weighted-label-ratio}(A_i, v, D) = E(A_i, v, D) - \text{maximum-label-ratio}(D), \quad (8)$$

where

$$E(A_i, v, D) = \sum_{j=1}^{v} (P_{ij} \times \text{maximum-label-ratio}(D^j_{A_i,v})), \quad (9)$$

$$\text{maximum-label-ratio}(D) = \underset{j=1}{\overset{2^k-1}{\text{Max}}} R_{L_j}, \quad (10)$$

$$P_{ij} = |D^j_{A_i,v}| \Big/ \sum_{j=1}^{v} |D^j_{A_i,v}|. \quad (11)$$

In Eq. (9), $E(A_i,v,D)$ is the weighted maximum label ratio of each subset $D^j_{A_i,v}$ after partitioning D. In Eq. (10), maximum-label-ratio(D) is the maximum label ratio among each label ratio R_{L_j} (referring to Eq. (4)) before partitioning D. In Eq. (11), $\sum_{j=1}^{v} |D^j_{A_i,v}| \geq |D|$ because a data record with multi-valued attribute can belong to multiple intervals. Thus, the denominator should be the sum of the data quantity of each subsets (i.e. $\sum_{j=1}^{v} |D^j_{A_i,v}|$) instead of being $|D|$.

3.3. Label prediction for new data and evaluation on the label prediction

MMDT predicts the class label of a record by traversing the decision tree as starting from the root node and find a path to the leaf node and use the label-set of this leaf as the predicted result. When a record has a multi-valued attribute, it may reach several leaf nodes. MMDT unions all these labels as the prediction result. The function predict_data in MMDT performs the label prediction work. Fig. 7 displays the procedure of function predict_data.

MMDT uses the same way as MMC did to evaluate the accuracy of the label prediction. It determines the accuracy of a record by measuring the similarity between the record's label-set and the predicted label-set first. Then calculate the average accuracy of the records in the test data set as the accuracy of the test data set.

```
1. Predict_data(u)
2. If u is a leaf node then return the labels of u;
3. result = ϕ;
4. For each child v of node u
5.     If the condition of arc (u,v) is satisfied by the test data;
6.     Then result = result ∪ Predict_data(v);
7. Endfor
8. Return result;
```

Fig. 7. The predict_data function.

4. Experiments

Han and Kamber addressed that the classification and prediction methods could be compared and evaluated according to the six criteria (Han, 2000; Han & Kamber, 2001): predictive accuracy, speed, robustness, scalability, interpretability and goodness of rules. This paper focuses on predictive accuracy and goodness of rules. We describe the experimental design in Section 4.1 first, then present the experimental results in Section 4.2.

4.1. Experimental design

In the experiment, MMDT with similarity basis and appearance basis, respectively, has been applied to a multi-valued and multi-labeled data set. To compare MMDT with MMC, MMDT uses the same data set as MMC did. As we have stated before (Chen et al., 2003), due to the lack of a benchmark containing multi-valued and multi-labeled data sets, we have modified a well-known data generator (Agrawal et al., 1992; Agrawal, Imielinski, & Swami, 1993; Shafer et al., 1996; Wang et al., 1998) to develop a synthetic data set for MMC. Based on the developed synthetic data set, the training set and the test set are generated. The synthetic set is a customer database for a tour company. It has nine ordinary attributes as shown in Table 5, where the attribute car, hobby, and occupation are multi-valued and the attribute class label is multi-labeled. The value of the class label attribute belongs to the set of label-sets: {{A},{B},{C},{A,B},{A,C},{B,C},{A,B,-C},…,{A,B,C,D,E}}. The data generator comprises five functions as Fig. 8 shows for assigning labels to the records generated. As a record fits the condition of a function, the corresponding label will be assigned to it. Certainly, a record may fit several conditions of the five functions and be tagged with multiple labels.

Conducting on a Pentium4-2 GHZ computer with 1024 MB of main memory, algorithms of MMDT and MMC written in C++ are applied to the experimental processes as follows. First, analyzing the training set and constructing a multi-valued and multi-labeled decision tree; second, transforming the tree into a rule set; third, making a prediction for a test set by the rule set; fourth, evaluating the accuracy of the prediction.

In each experiment, we use 5000 records as the test set, then compare the results by fixing four of the five parameters: size of training set = 6000, minsup = 50%, mindiff = 15%, minqty = 6 and ub = 6. Values of the parameters are specified according to the following: Size of training set is increased from 2000 to 10,000 by 2000; minsup is increased from 40 to 60% by 5%; mindiff is increased from 5 to 25% by 5%; minqty is increased from 2 to 10 by 2; and ub is increased from 2 to 10 by 2.

4.2. Experimental results

The results show that MMDT and MMC with different values of the five experimenting parameters (size of training set, minsup, mindiff, ub and minqty) produce different accuracy and different number of rules. Fig. 9 shows the average accuracy for each algorithm and Fig. 10 shows the average number of rules for each algorithm. To get a more detailed understanding of MMDT and MMC, we show in Table 6 a breakdown of the accuracies and the numbers of rules varied by the five parameters. We compare MMDT with MMC at the accuracy and the number of rules first in Section 4.2.1, then discuss the behavior of MMDT in Section 4.2.2.

Table 5
Description of attributes

Attribute	Description	Number of values	Value
Salary	Salary	1	Uniformly distributed from $20,000 to $150,000
Gender	Gender	1	Uniformly chosen from 1 to 2
Age	Age	1	Uniformly chosen from 20 to 80
Car	Make of the car	Uniformly chosen from 1 to 3	Uniformly chosen from 1 to 20
Hobby	Hobby	Uniformly chosen from 1 to 5	Uniformly chosen from 1 to 20
Hvalue	Value of the houses	1	Uniformly distributed from $50,000 to $150,000
Elevel	Education level	1	Uniformly chosen from 1 to 5
m-Status	Marital status	1	Uniformly chosen from 1 to 3
Occupation	Occupation	Uniformly chosen from 1 to 2	Uniformly chosen from 1 to 10

Function of label 'A':

[(gender = 1) $\wedge$ (\$20,000 $\le$ salary $\le$ \$100,000) $\wedge$ (car $\in$ [1..10])] $\vee$

[(gender = 2) $\wedge$ (\$100,000 $\le$ salary $\le$ \$150,000) $\wedge$ (car $\in$ [11..20])] $\vee$

[occupation $\in$ [1, 6]]

Function of label 'B':

[(age < 40) $\wedge$ (car $\in$ [1..10])] $\vee$

[(40 $\le$ age < 60) $\wedge$ (car $\in$ [11..15])] $\vee$

[occupation $\in$ [2]]

Function of label 'C':

[(m-status = 1) $\wedge$ (\$20,000 $\le$ salary $\le$ \$100,000) $\wedge$ (hobby $\in$ [1..9] $\vee$ car = 10)] $\vee$

[(m-status $\in$ [2,3]) $\wedge$ (\$100,000 $\le$ salary $\le$ \$150,000) $\wedge$ (hobby $\in$ [11..19] $\vee$ car = 20)] $\vee$

[occupation $\in$ [3..5]]

Function of label 'D':

[(m-status = 1) $\wedge$ (\$20,000 $\le$ salary $\le$ \$40,000)] $\vee$

[m-status = 2 $\wedge$ (\$40,000 < salary $\le$ \$80,000)] $\vee$

[occupation $\in$ [7..8]]

Function of label 'E':

[(Elevel = 1) $\wedge$ (hobby $\in$ [1..9]) $\wedge$ (m-status = 1)] $\vee$

[(Elevel = 5) $\wedge$ (hobby $\in$ [10..20]) $\wedge$ (m-status $\in$ [2..3])] $\vee$

[occupation $\in$ [9..10]]

Fig. 8. The five functions for each label.

4.2.1. Comparisons between MMDT and MMC

Fig. 9 shows that both MMDT using appearance basis and MMDT using similarity basis perform better accuracy than MMC. Since all accuracies of the algorithms are apparently larger than and different from the average randomly-guessed accuracy of partial correctness 35.48% (i.e. the average of all entries of similarity table with five labels) and the average randomly-guessed accuracy of exact correctness 3.23%, the result is meaningful.

Fig. 10 shows that MMC performs better number of rules than both MMDT using appearance basis and MMDT using similarity basis. MMDT using appearance basis produces about 1.4 times the number of rules of MMC; and MMDT using similarity basis produces about 3.3 times the number of rules of MMC.

The results given above make it clear that MMDT achieves better accuracy, and MMC achieves smaller tree.

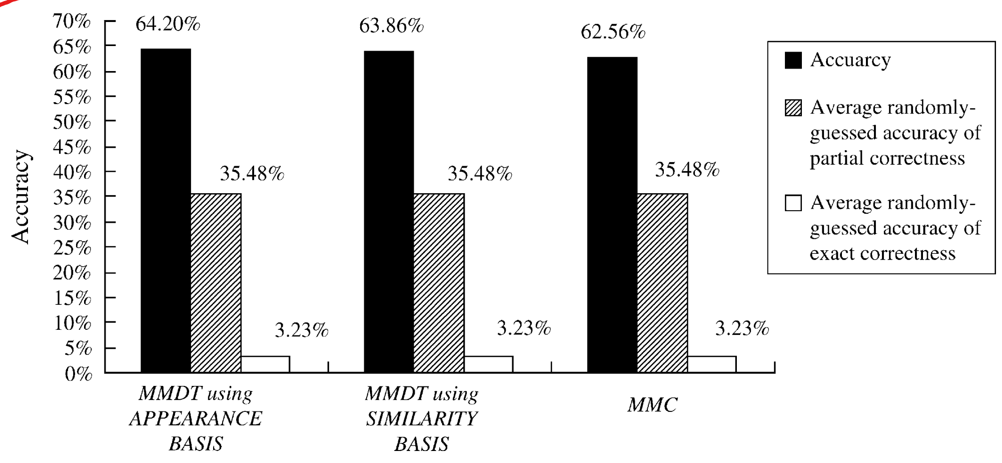

Fig. 9. The comparisons of accuracy among MMDT using appearance basis, MMDT using similarity basis and MMC.

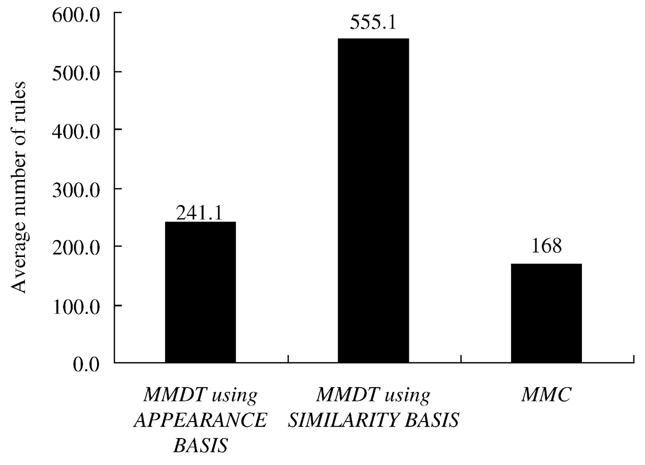

Fig. 10. The comparisons of number of rules among MMDT using appearance basis, MMDT using similarity basis and MMC.

4.2.2. The behavior of MMDT

With four of the five parameters (i.e. size of training set, minsup, mindiff, ub and minqty) fixed, a series of experiments have been carried out to examine the effect of the left parameter. The experimental results show that the behavior of MMDT using appearance basis and the behavior of MMDT using similarity basis are basically consistent. Following discusses the experimental results:

1. With parameters minsup = 50%, mindiff = 15%, minqty = 6 and ub = 6. Table 6(1) is the experimental results of MMDT with different training data set sizes (i.e. 2000 records, 4000 records, 6000 records, 8000 records and 10,000 records). It shows that as the size of the training data set exceeds a certain threshold, the classification accuracy decreases. The reason could be: When the experimenting training data set is small, the expansion provides more useful information for growing a tree. However, when the size of the data set reaches the threshold, the information needed to grow a tree is enough, and further expansion of the training data set provides more noises than information. This might explain why the classification accuracy declines when the training data set exceeds 6000 records in appearance basis or exceeds 8000 records in similarity basis.
2. With parameters size of training set = 6000, mindiff = 15%, minqty = 6 and ub = 6. Table 6(2) is the experimental results of MMDT with different value of minsup (i.e. 40, 45, 50, 55 and 60%). It shows that the accuracy will increase and the number of rules have a decline tendency as minsup increases. With regard to the increase of accuracy, the reason could be: When the value of minsup becomes larger, the majority value of each leaf node (i.e. large set) gets better support. This makes the labels of each leaf node have higher similarity and label ratio, and hence improves the accuracy of the prediction. With regard to the decline tendency of the number of rules, the reason could be: When the value of minsup is larger, it is more likely that a node will stop growing. This will make the tree smaller and lessen the number of rules. Following two cases demonstrate the tendency: When minsup = 40%, a node will stop growing when the support of a label larger than minsup and the largest support of the small labels falls in [0,25%] (as mindiff = 15%), and the probability is 10.5%; when minsup = 60%, a node will stop growing when support of a label larger than minsup and the largest support of the small labels falls in [0,45%], and the probability is 18.5%. Therefore, as minsup increases, there is a tendency to make the tree smaller and lessen the number of rules.
3. With parameters size of training set = 6000, minsup = 50%, minqty = 6 and ub = 6. Table 6(3) is the experimental results of MMDT with different value of mindiff (i.e. 5, 10, 15, 20 and 25%). It shows that the accuracy has an ascending tendency and the number of rules will increase as mindiff increases. With regard to the ascending tendency of accuracy, the reason could be: When the value of mindiff gets larger, there is a sharper boundary between large set and small set in a node. Thus, the distinction between large set and small set is more clear. This makes the labels of a leaf node have a higher similarity, and hence improves the accuracy of the prediction. With regard to the increase of the number of rules, the reason could be: When the value of mindiff is smaller, it is more likely that a node will stop growing. This will make the tree smaller and lessen the number of rules. Following two cases demonstrate the tendency: When mindiff = 10%, a node will stop growing when the support of a label larger than minsup and the largest support of the small labels falls in the interval [0,40%] (as the minimum support of the large labels must be larger than minsup (50%)), and the probability is 20.5%; when mindiff = 25%, a node will stop growing if the support of a label larger than minsup and the largest support of the small labels falls in the interval [0,25%], and the probability is 13%. Therefore, as mindiff increases, there is a tendency to make the tree bigger and increase the number of rules.
4. With parameters size of training set = 6000, minsup = 50%, mindiff = 15% and ub = 6. Table 6(4) is the experimental results of MMDT with different value of minqty (i.e. 2, 4, 6, 8 and 10). It shows that the number of rules has a decline tendency as minqty increases. The reason could be: When the value of minqty is bigger, it is more likely that the data quantity of each node is smaller than minqty, and hence each node might stop growing earlier. Therefore, as minqty increases, there is a tendency to make the tree smaller and lessen the number of rules.
5. With parameters size of training set = 6000, minsup = 50%, mindiff = 15% and minqty = 6. Table 6(5) is the experimental results of MMDT with different value of ub

Table 6
The experimental results of MMC, MMDT using appearance basis and MMDT using similarity basis with different parameter values

Training set size	Accuracy			Number of rules		
	MMC	MMDT using appearance basis	MMDT using similarity basis	MMC	MMDT using appearance basis	MMDT using similarity basis
(1) minsup=50%, mindiff=15%, minqty=6 and ub=6						
2000	0.6035	0.6447	0.6413	100	211	263
4000	0.6428	0.6468	0.6444	99	603	732
6000	0.6425	0.6561	0.6500	137	158	710
8000	0.6397	0.6552	0.6552	195	259	140
10000	0.6361	0.6537	0.6538	212	355	243
Average	0.6329	0.6513	0.6490	148.6	317.2	417.6

Minsup	Accuracy			Number of rules		
	MMC	MMDT using appearance basis	MMDT using similarity basis	MMC	MMDT using appearance basis	MMDT using similarity basis
(2) Training set size=6000, mindiff=15%, minqty=6 and ub=6						
40%	0.6122	0.5957	0.6029	270	821	970
45%	0.6324	0.6498	0.6499	234	955	905
50%	0.6425	0.6561	0.6500	137	158	710
55%	0.6434	0.6562	0.6562	130	10	10
60%	0.5963	0.6562	0.6562	69	10	10
Average	0.6254	0.6428	0.6430	168	390.8	521

Mindiff	Accuracy			Number of rules		
	MMC	MMDT using appearance basis	MMDT using similarity basis	MMC	MMDT using appearance basis	MMDT using similarity basis
(3) Training set size=6000, minsup=50%, minqty=6 and ub=6						
5%	0.3986	0.3986	0.3986	1	1	1
10%	0.5905	0.6569	0.6536	75	55	162
15%	0.6425	0.6561	0.6499	137	158	710
20%	0.6518	0.6554	0.6522	259	291	812
25%	0.6600	0.6537	0.6514	490	480	1004
Average	0.5887	0.6041	0.6011	192.4	197	537.8

Minqty	Accuracy			Number of rules		
	MMC	MMDT using appearance basis	MMDT using similarity basis	MMC	MMDT using appearance basis	MMDT using similarity basis
(4) Training set size=6000, minsup=50%, mindiff=15% and ub=6						
2	0.6429	0.6561	0.6499	137	158	710
4	0.6429	0.6561	0.6499	137	158	710
6	0.6425	0.6561	0.6499	137	158	710
8	0.6430	0.6560	0.6454	134	143	587
10	0.6424	0.6559	0.6501	125	92	529
Average	0.6427	0.6560	0.6490	134	141.8	649.2

ub	Accuracy			Number of rules		
	MMC	MMDT using appearance basis	MMDT using similarity basis	MMC	MMDT using appearance basis	MMDT using similarity basis
(5) Training set size=6000, minsup=50%, mindiff=15% and minqty=6						
2	0.6273	0.6560	0.6499	16	171	784
4	0.6550	0.6560	0.6499	85	171	784
6	0.6425	0.6561	0.6499	137	158	710
8	0.6485	0.6560	0.6519	250	147	589
10	0.6188	0.6560	0.6534	495	147	383
Average	0.6384	0.6560	0.6510	196.6	158.8	650

(i.e. 2, 4, 6, 8 and 10). It shows that the number of rules has a decline tendency as ub increases. Generally, when the value of ub is larger, a node is more likely to have more branches and leaves which result in more number of rules. The contrary tendency of the experimenting result might be related to early stop of tree growing. The reason could be: A node with more branches has less data quantity than a node with fewer branches. It is more likely that the data quantity of each node is smaller than minqty. Therefore, as ub increases, there is a tendency to make the tree smaller and lessen the number of rules.

5. Summary and conclusion

This research has designed a decision tree classifier, MMDT, to improve the accuracy of MMC by minimizing the over-fitting problems. In MMDT, we set a constraint of size for the data set in each node to avoid the data set being too small; and we consider not only the average similarity of labels of each child node but also the average appropriateness of labels of each child node to decrease the bias problem. The experimental results show that MMDT has improved the accuracy of MMC.

The main works of MMDT can be summarized as follows:

1. To select the best multi-valued attribute: MMDT proposes a new measure named similarity ratio, which combines the measure of weighted label ratio with the measure of weighted similarity, as the measure for selecting the best multi-valued attribute.
2. To grow a multi-valued and multi-labeled decision tree: MMDT constructs a decision tree by traversing and splitting each internal node recursively. When the traversing has reached a leaf node, a label-set would be assigned to the leaf node. To avoid the data set in each node being too small, MMDT sets a threshold. To acquire a smaller decision tree, MMDT sets a degree constraint for each internal node containing numeric attribute to reduce branches of the internal node. The constraint can reduce the number of rules.
3. To assign multiple labels to a leaf: Several thresholds and parameters of MMDT are used to decide whether a node can be a leaf and what label-set can be assigned to the leaf.
4. To make a prediction based on the decision tree: MMDT predicts the class label of a record by traversing the tree from its root until the traversing finally reaches a leaf node. If a record has a multi-valued attribute and the traversing reaches several leaf nodes, MMDT will union all the leaves' label-sets as the prediction result.

The capability of classifying multi-valued and multi-labeled data could be applied to many real-world commercial data. It also could be applied to the handling of more complicated data. For example, it makes a multi-valued and multi-labeled meta-data for semi-structured data (Wang, Zhou, & Liew, 1999; Zaiane & Han, 1995) or object-oriented data (Han, Nishio, Kawano, & Wang, 1998) meaningful and manageable. This certainly extends our ability in data management. Therefore, continued improvement on the multi-valued and multi-labeled classification work is important.

References

Agrawal, R., Ghosh, S., Imielinski, T., Iyer, B., & Swami, A. (1992). An interval classifier for database mining applications. *Proceedings of the 18th international conference on very large databases* (pp. 560–573). Vancouver, BC.

Agrawal, R., Imielinski, T., & Swami, A. (1993). Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, *5*(6), 914–925.

Chen, Y., Hsu, C., & Chou, S. (2003). Constructing a multi-valued and multi-labeled decision tree. *Expert Systems with Applications*, *25*(2), 199–209.

Gehrke, J., Ramakrishnan, R., & Ganti, V. (1998). Rainforest: A framework for fast decision tree construction of large datasets. *Proceedings of the 24th international conference on very large databases*. New York.

Gordon, D. F., & Desjardins, M. (1995). Evaluation and selection of biases in machine learning. *Machine Learning*, *20*(1–2), 5–22.

Han, J. (2000). From data mining to web mining: An overview. *Conference tutorial (in PowerPoint), 2000 international database systems conference*. Hong Kong. ftp://ftp.fas.sfu.ca/pub/cs/han/slides/hkw00.ppt

Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques.*(pp. 279–333). San Francisco, CA: Morgan Kaufmann.

Han, J., Nishio, S., Kawano, H., & Wang, W. (1998). Generalization-based data mining in object-oriented databases using an object-cube model. *Data and Knowledge Engineering*, *25*(1–2), 55–97.

Mantaras, R. L. D. (1991). A distance-based attribute selection measure for decision tree induction. *Machine Learning*, *6*, 81–92.

Mehta, M., Agrawal, R., & Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. *Proceedings of the fifth international conference on extending database technology*.

Quinlan, J. R. (1979). Discovering rules from large collections of examples: A case study. In D. Michie (Ed.), *Expert Systems in the Microelectronic Age*. Edinburgh, Scotland: Edinburgh University Press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Rastogi, R., & Shim, K. (1998). Public: A decision tree classifier that integrates building and pruning. *Proceedings of the 24th international conference on very large databases*.

Russell, S., & Norving, P. (1995). *Artificial intelligence—A modern approach*. Upper Saddle River, NJ: Prentice-Hall.

Shafer, J. C., Agrawal, R., & Mehta, M. (1996). SPRINT: A scalable parallel classifier for data mining. *Proceedings of the 22nd international conference on very large databases* (pp. 544–555). Mumbai (Bombay), India.

Steinberg, D., & Colla, P. L. (1995). *CART: Tree-structured nonparametric data analysis*. San Diego, CA: Salford Systems.

Umano, M., Okamoto, H., Hatono, I., Tamura, H., Kawachi, F., & Umedzu, S., et al. (1994). Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems. *Proceedings of the third IEEE international conference on fuzzy systems* (Vol. 3) (pp. 2113–2118). Orlando, FL.

Wang, M., Iyer, B., & Vitter, J. S. (1998). Scalable mining for classification rules in relational databases. *Proceedings of international database engineering and applications symposium* (pp. 58–67). Cardiff, Wales, UK.

Wang, K., Zhou, S., Liew, S. C. (1999). Building hierarchical classifiers using class proximity. *Proceedings of the 25th international conference on very large data bases* (pp. 363–374). Edinburgh, Scotland.

Zaiane, O. R., Han, J. (1995). Resource and knowledge discovery in global information systems: A preliminary design and experiment. *Proceedings of the first international conference on knowledge discovery and data mining* (pp. 331–336). Montreal, Quebec.