Bloomy Decision Tree for Multi-objective Classification

Einoshin Suzuki, Masafumi Gotoh, and Yuta Choki

Division of Electrical and Computer Engineering
Yokohama National University, Japan
{suzuki,dandy,choki}@slab.dnj.ynu.ac.jp

Abstract. This paper presents a novel decision-tree induction for a multi-objective data set, i.e. a data set with a multi-dimensional class. Inductive decision-tree learning is one of the frequently-used methods for a single-objective data set, i.e. a data set with a single-dimensional class. However, in a real data analysis, we usually have multiple objectives, and a classifier which explains them simultaneously would be useful and would exhibit higher readability. A conventional decision-tree inducer requires transformation of a multi-dimensional class into a single-dimensional class, but such a transformation can considerably worsen both accuracy and readability. In order to circumvent this problem we propose a bloomy decision tree which deals with a multi-dimensional class without such transformations. A bloomy decision tree has a set of split nodes each of which splits examples according to their attribute values, and a set of flower nodes each of which predicts a class dimension of examples. A flower node appears not only at the fringe of a tree but also inside a tree. Our pruning is executed during tree construction, and evaluates each class dimension based on Cramér's V. The proposed method has been implemented as D3-B (Decision tree in Bloom), and tested with eleven data sets. The experiments showed that D3-B has higher accuracies in nine data sets than C4.5 and tied with it in the other two data sets. In terms of readability, D3-B has a smaller number of split nodes in all data sets, and thus outperforms C4.5.

1 Introduction

Given a set of training examples, learning from examples aims at constructing a classifier which predicts the class of an unseen example. Here, learning from examples assumes that each example has a single-dimensional class, and can thus be called as single-objective. Inductive decision-tree learning [2,10,11] has been successfully used in various fields as one of the most useful methods in learning from examples.

In dealing with real data, however, we often have multiple objectives, and may wish to predict a multi-dimensional class [3,4]. Building a separate decision tree for each objective would be problematic in terms of readability because decision trees differ in their structures and in their split attributes, and are thus difficult to be compared. A single classifier which predicts this multi-dimensional class

L. De Raedt and A. Siebes (Eds.): PKDD 2001, LNAI 2168, pp. 436–447, 2001.

would be more useful. For example, suppose an analyst constructing a classifier from an agricultural data set about various crops. Rather than having a decision tree which predicts only corn, the analyst would prefer a decision tree which predicts corn and wheat simultaneously since it would be comprehensible.

In such a case, a conventional decision-tree learning algorithm can construct a classifier if the multi-dimensional class is transformed into a single-dimensional class. This idea is described in [3,4] briefly without experimental justification[1]. However, a transformation without loss of information, such as assigning a new class value to each combination of class values, considerably increases the number of class values. This tendency causes a fragmentation problem [11]: each class value has only a few training examples in a split node at the bottom of a decision tree, and appropriate selection of an attribute would be difficult. A transformation with loss of information such as principle component analysis [6, 7] could overlook useful knowledge.

In order to circumvent this problem we propose a bloomy decision tree in which each class dimension is independently predicted by a flower node. Since a flower node can be constructed inside a tree, the number of class dimensions gradually decreases as an example descend the tree. This corresponds to coping with the fragmentation problem by simplifying the classification task in order to construct a small decision tree with high accuracy. We have implemented an induction algorithm of bloomy decision trees as D3-B (Decision tree in Bloom), and demonstrate its effectiveness as a multi-objective classifier with eleven data sets.

2 Decision Tree

In this section, we give a simple explanation of inductive decision-tree learning [2,10,11]. For various problems in this field, please refer to a recent survey [9].

2.1 Construction of a Decision Tree

A decision tree represents a tree-structured classifier which consists of a set of nodes and a set of edges. A node is either a split node which tests an attribute or a leaf node which predicts a class of an example. Given an unseen example, a split node assigns the example to one of its subtrees according to the value of its attribute, and a leaf node predicts the class value of the example.

The input to a decision-tree inducer is a set E of examples. An example e_i has n attribute values $a_{1i}, a_{2i}, \cdots, a_{ni}$ for attributes $a_1, a_2, \cdots, a_n$ and a class value c_i to a class c, and is represented as $e_i = (a_{1i}, a_{2i}, \cdots, a_{ni}, c_i)$.

[1] Caruana formalized a learning problem with multiple objectives as multitask learning [3,4], which includes our multi-objective classification. He almost exclusively worked with neural networks, and only dropped a few remarks about decision trees. His remarks mainly concern proposal of novel split criteria, and no proposals are given for knowledge representation.

A decision tree is typically constructed by a recursive split of example space with a divide and conquer method. The split typically employs greedy search, and, for each split node, the best attribute is selected as the attribute of the node based on an evaluation function. We will explain a typical function in Sect. 2.2. The class value of a leaf node is determined if all training examples in the node have the same class value. If a leaf node has no training examples, the most frequent class value of examples in its parent is assigned as the class value of the leaf. If the training examples in a split node can no longer be split, the node becomes a leaf node and the most frequent class value of examples in the node is assigned as the class value of the leaf. Here, a decision tree which perfectly predicts the class of a training example tends to perform poorly for test examples. A procedure called pruning replaces a subtree which is judged irrelevant in the prediction with a leaf node. We will explain a pruning method in Sect. 2.3.

2.2 Attribute Selection

Here, we explain gain ratio [10,11] as one of the evaluation criteria. Gain ratio $G(a, c, E)$ is a criterion based on mutual entropy of an attribute a and the class c for a set E of examples. Let $|E|$, $E_{a=i}$, and $E_{c=i}$ be the number of examples in E, the set of examples each of which satisfies $a = i$ in E, and the set of examples each of which satisfies $c = i$ in E respectively, then

$$G(a, c, E) \equiv \frac{H(c, E) - J(a, c, E)}{H(a, E)} \tag{1}$$

where

$$H(c, E) \equiv -\sum_i \frac{|E_{c=i}|}{|E|} \log_2 \left(\frac{|E_{c=i}|}{|E|} \right) \tag{2}$$

$$J(a, c, E) \equiv \sum_i \frac{|E_{a=i}|}{|E|} H(c, E_{a=i}) \tag{3}$$

2.3 Pruning

Pruning can be classified as either pre-pruning, which is executed during tree construction, or post-pruning, which is executed after tree construction [8,10, 11]. Compared with post-pruning, pre-pruning is time-efficient since it does not require construction of a complete tree. However, experimental evidence shows that pre-pruning leads to low accuracy [8], and most decision-tree inducers employ post-pruning.

Pruning based on χ^2 is employed in decision-tree inducers such as ID3 [10]. This method first calculates a χ^2 value $\chi^2(a, c, E)$ of an attribute a and the class c for a set E of examples in a node. Let $E_{a=i,c=j}$ be the set of examples each of which satisfies both $a = i$ and $c = j$ in E, then

$$\chi^2(a, c, E) \equiv \sum_i \sum_j \frac{(|E_{a=i,c=j}| - \epsilon_{a=i,c=j}(E))^2}{\epsilon_{a=i,c=j}(E)} \tag{4}$$

where

$$\epsilon_{a=i,c=j}(E) \equiv \frac{|E_{a=i}||E_{c=j}|}{|E|} \tag{5}$$

Let $\chi^2_r(\alpha)$ be a χ^2 value with a degree of freedom r and a significance level α, then when $\chi^2(a, c, E)$ is smaller than a threshold $\chi^2_r(\alpha)$, the attribute a and the class c are considered to have no relevance, and the split node is replaced by a leaf node. A shortcoming of this approach is that $\chi^2(a, c, E)$ tends to be overly large when the number of examples in the training set is large, and a decision tree is often under-pruned [8].

3 Bloomy Decision Tree for Multi-objective Classification

3.1 Bloomy Decision Tree

In a data set E for multi-objective classification, i.e. with a multi-dimensional class, each example e_i has n attribute values $a_{1i}, a_{2i}, \cdots, a_{ni}$ for attributes $a_1, a_2, \cdots, a_n$, and m class values $(c_{1i}, c_{2i}, \cdots, c_{mi})$ for a m-dimensional class $(c_1, c_2, \cdots, c_m)$.

The problems in Sect. 1 have led us to invent a bloomy decision tree for multi-objective classification. In a decision tree for multi-objective classification, several class dimensions can be predicted accurately even at an internal node. A bloomy decision tree predicts such dimensions in an internal node which is called a flower node, and typically reduces the number of class dimensions to be predicted as an example descends the tree from its root to one of its leaves. This corresponds to simplifying a multi-objective classification downward a decision tree, and can be considered as an efficient solution to the fragmentation problem described in Sect. 1. A bloomy decision tree is expected to show high accuracy with a simpler structure compared with a conventional decision tree. A flower node corresponds to a leaf node which predicts a set of class dimensions, and appears not only at the fringe of a tree but also inside a tree.

Similar to a conventional decision tree, a bloomy decision tree T has a recursive tree structure. A node N of a bloomy decision tree T is classified as either a flower node N_{bloom} which predicts values of a set of class dimensions, or a split node N_{split} which splits examples according to their attribute values. Figure 1 shows an example of a bloomy decision tree for a 2-dimensional class (c_1, c_2), where an oval and a rectangle represent a flower node and a split node respectively. Note that a flower node appears inside the tree since a class dimension is predicted as $c_1 =$ p for examples each of which satisfies $Attribute1 =$Y.

A root node of a bloomy decision tree is a split node. A split node N_{split} has an attribute a which is selected according to a procedure in the next section. Let the number of values for an attribute a be v_a, then there are v_a child nodes for N_{split}, and each child node is assigned a subset $E_{a=a_i}$ where a_i is the i-th value of a. A child node of N_{split} is either a split node or a flower node.

A flower node N_{bloom} consists of l $(\leq m)$ petals $p_1, p_2, \cdots, p_l$ each of which predicts a class dimension. Alternatively, a petal p_i represents that a predicted

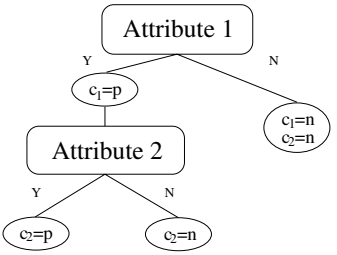

Fig. 1. Example of a bloomy decision tree

value is assigned to a class dimension c_j. The predicted value of c_j is fixed in the petal and remain unchanged in the child nodes of N_{bloom}. If some of the class dimensions remain unpredicted in N_{bloom}, N_{bloom} has a child node which is a split node. Note that a flower node can be an internal node as well as a leaf node.

3.2 Attribute Selection

Similar to the construction of a decision tree, we employ a divide and conquer method based on an attribute selection function $F(a, E)$. Gain ratio presented in Sect. 2.2 is an evaluation function for a single-dimensional class, and cannot be employed without modification. In this paper, we employ the add-sum $F(a, E)$ of gain ratio $G(a, c_j, E)$ for each class dimension c_j.

$$F(a, E) \equiv \sum_{c_j} G(a, c_j, E) \tag{6}$$

Given a set E of training examples, the attribute a which maximizes $F(a, E)$ is selected in a split node. We call this approach as the add-sum criterion.

Instead of using the add-sum of gain ratios, we can also consider their product or the add-sum of their squares, which we call the product criterion and the squares-sum criterion respectively. However, the product criterion would be overly pessimistic, avoiding an attribute which has at least a nearly-zero gain ratio. The squares-sum criterion, on the other hand, would be overly optimistic, preferring an attribute which relies on a few large gain ratios. The former neglects an attribute which works well for a subset of dimensions, and the latter criterion is typically dominated by outliers. Therefore, we use the add-sum criterion in our approach. Note that this analysis could be justified experimentally under various settings, but we leave this for future work due to space constraint.

3.3 Pruning

In order to obtain an accurate classifier for all class dimensions, each dimension should be evaluated independently. Our method employs pre-pruning for each

dimension immediately after constructing a split node, and assigns a predicted value for a dimension which is pruned.

As explained in Sect. 2.3, χ^2 pruning tends to produce an overly large decision tree when the number of examples in the training set is large. Therefore, we use Cramér's V [7] to cope with this problem. Cramér's V $V(a, c_j, E)$ is an index of relevance of an attribute a and a class dimension c_j for a set E of examples.

$$V(a, c_j, E) \equiv \sqrt{\frac{\chi^2(a, c_j, E)}{|E|(\min(v_a, q(c_j)) - 1)}} \tag{7}$$

where $q(c_j)$ is the number of values of c_j. This index satisfies $0 \leq V(a, c_j, E) \leq 1$.

Since this index is, unlike χ^2, simply employed to compare its value and has no theoretical interpretation, we use the following value $V_r(a, c_j, E_{\mathrm{Z}}, \alpha)$ as a threshold for pruning.

$$V_r(a, c_j, E_{\mathrm{Z}}, \alpha) \equiv \sqrt{\frac{\chi_r^2(\alpha)}{|E_{\mathrm{Z}}|(\min(v_a, q(c_j)) - 1)}} \tag{8}$$

where $|E_{\mathrm{Z}}|$ is the expected number of examples assigned to the split node. Let $|E_{\mathrm{P}}|$ and $|N_{\mathrm{PC}}|$ be the number of examples in the parent split node and the number of child nodes of the parent split node respectively, then

$$|E_{\mathrm{Z}}| \equiv \frac{|E_{\mathrm{P}}|}{|N_{\mathrm{PC}}|} \tag{9}$$

For a root node, we define that $|E_{\mathrm{P}}|$ is equivalent to the number of training examples in the data set. In our method, a split node is pruned with respect to a dimension c_j if $V(a, c_j, E) < V_r(a, c_j, E_{\mathrm{Z}}, \alpha)$, and decision-tree construction is continued with the remaining dimensions of the class.

As explained in Sect. 2.3, post-pruning produces an accurate tree but is time-consuming [8]. Our method, unlike a conventional decision-tree inducer, continues construction of a decision tree with the remaining dimensions even after pruning. Therefore, we do not employ post-pruning since it requires iterations of construction and pruning, and is thus time-consuming.

4 D3-B

4.1 Construction of a Bloomy Decision Tree

We have implemented our method as D3-B. Its algorithm is shown below, where each attribute and each class dimension is assumed to have a discrete value. Given a set E of examples, D3-B outputs a bloomy decision tree T using *D3-B*(training set).

Given a set E of examples, algorithm *D3-B* recursively constructs a bloomy decision tree T with a divide and conquer method. If the training set E in a node can no longer be divided, we add, to T, a flower node which predicts all

dimensions in E. We define that a node can no longer be divided if and only if at least one of the following conditions hold: 1) no class dimension in E, 2) values of each class dimension are identical, or 3) E is empty. A predicted value is the majority of E if E is non-empty, otherwise the majority of the training set.

If E can be divided, an attribute a is first selected according to the procedure described in Sect. 3.2. Next, based on the pruning procedure which will be described in the next section, a set P of class dimensions to be pruned are obtained. If P is non-empty, we add, to T, a flower node which predicts the class dimensions in P, and those class dimensions are deleted from E. From Sect. 3.1, in this case, there is only one child node, which will be constructed by *D3-B*(E). For instance, in Fig. 1 c_1 was pruned at the left child of the root node. If P is empty, child nodes are constructed by *D3-B*($E_{a=v}$) for each value v of a.

Algorithm: *D3-B*(E)
Input: data set E
Return value: bloomy decision tree T
begin
If ((E can no longer be split) **and** (E has a class dimension))
  Add a flower node which predicts all class dimensions in E to T
Else
  begin
  $a \leftarrow \arg\max_{a'} F(a', E)$
  $P \leftarrow$ *Prune*(E,a)
  If(P is non-empty)
    begin
    Add a flower node which predicts class dimensions in P to T
    Delete class dimensions in P from E
    Construct a child node of T by *D3-B*(E)
    end
  Else
    Foreach(value v of attribute a)
      Construct child nodes of T by *D3-B*($E_{a=v}$)
  end
Return T
end

4.2 Pruning of a Bloomy Decision Tree

Given a set E of examples and a selected attribute a, our algorithm returns a set P of class dimensions each of which should be predicted in the node. For each dimension c_i in E, the following procedure checks the pruning condition explained in Sect. 3.3, and adds the class dimension if it satisfies the condition to P. This procedure is done in lexical order of class dimensions for simplicity, and a class dimension c_i which satisfies the condition is not employed in the calculation of $V(a, c_j, E)$ and $V_r(a, c_j, E_{\mathrm{Z}}, \alpha)$ for a subsequent class dimension c_j.

Procedure: Prune(E,a)
Input: data set E, selected attribute a
Return value: a set of class dimensions P
begin
$P \leftarrow \phi$
Foreach(class dimension c_i in E)
If($V(a, c_i, E) < V_r(a, c_i, E_{\mathrm{Z}}, \alpha)$)
begin
$P \leftarrow P \cup \{c_i\}$
c_i is not employed in the calculation of $V(a, c_j, E)$ and $V_r(a, c_j, E_{\mathrm{Z}}, \alpha)$ for a subsequent class dimension c_j
end
Return P
end

5 Experimental Evaluation

5.1 Conditions of Experiments

We demonstrate the effectiveness of D3-B as a multi-objective classifier by experiments with eleven data sets. In the rest of this paper, thresholds for pruning were settled with $r = 1$ and $\alpha = 5\%$.

"Agriculture" is a series of data sets which describe agricultural statistics for 3,246 municipalities in Japan. In this experiment, we employed the 1991 version. Each example is represented by 37 attributes such as areas, populations, financial statistics, and industrial statistics; and has a 25-dimensional class about gross products of crops. A continuous attribute was first discretized with equal-frequency method [5] of five bins. We employed a simple class-blind discretization method, rather than a class-driven discretization method, for the sake of simplicity and speed-up. Before discretization, we visualized distributions of values for several attributes, and chose the number five arbitrarily. A continuous class dimension was first discretized with equal-frequency method of two bins. The number two was chosen after we observed poor performance of induction algorithms with five bins. We consider that information contained in this data set is insufficient in order to learn a classifier which correctly predicts a fine-defined multi-dimensional class.

"Kiosk" is a data set which describes inventories about 52 kinds of merchandise in 232 shops of a Japanese company in 1994. In this data set, each kind of merchandise has at least 83 zeros. In discretizing a continuous attribute, we treated 0 as a value, and applied equal-frequency method of three bins to the other values. The number three was chosen because only a small number of examples were used in the discretizing procedure.

The other nine data sets comes from the UCI Repository [1]. We discretized a continuous attribute with equal-frequency method of four bins, where a missing value was left unchanged. The number four was chosen due to the wide variety of attributes concerning distributions of values: the distribution was relatively

balanced for some attributes such as those in "Agriculture", and was skewed for other attributes such as those in "Kiosk".

For each data set, 100 multi-objective classification tasks were settled. For each classification task except for those with the agriculture data, we chose six attributes randomly, and regarded them as a 6-dimensional class. In choosing these attributes for UCI data sets, we considered their appropriateness as a class dimension. For each attribute, a single-objective classification task was settled, and an attribute with which C4.5[11]'s accuracy is less than 63.5% with 5-fold cross-validation was ignored in selecting a class dimension. As the result, the Australian data and the mushroom data have only 28 ($= {}_8\mathrm{C}_6$) possible sets of 6-class tasks, so we checked all these 28 tasks instead of randomly-chosen 100 tasks for these data sets. For the agriculture data, we employed the 25-dimensional class. Initial experiments revealed that 3.0 % difference for average accuracy and 1.5 difference for average number of nodes can be each considered as significant.

Note that, in practice, a class dimension should be settled in terms of its importance in the domain. An interesting research avenue would be to measure effectiveness of learning algorithms by constructing a class attribute with attributes that can be more naturally used as a class dimension, in the sense that their prediction is useful for the user.

In order to evaluate the effectiveness of our approach, we compared D3-B with six learning algorithms including C4.5 and variants of D3-B. In the experiments, average accuracies for class dimensions and average number of split nodes were measured by 5-fold cross-validation as evaluation indices. First, C4.5 was chosen as the representative of conventional decision-tree inducers. In applying C4.5, a multi-dimensional class was transformed to a single-dimensional class by assigning a new class value to each combination of class values. Second, D3-B was also applied to data sets each of which was produced with this transformation in order to evaluate the effectiveness of flower nodes. Third, in order to evaluate the effectiveness of Cramér's V pruning, we also employed D3-B with χ^2 pruning. Fourth and fifth, the add-sum criterion in Sect. 3.2 was evaluated by using D3-Bs with the product criterion and the squares-sum criterion. Sixth, we compared D3-B with a method which constructs a decision tree for each class dimension.

5.2 Experimental Results

Figure 2a shows the accuracies and the numbers of split nodes of D3-B and C4.5. Concerning accuracy, our method outperforms C4.5 in nine data sets by 3.4 % - 19.3 %, and approximately ties with it in the other two data sets (our advantage is less than 2.4 %). Concerning the number of split nodes, D3-B constructs smaller trees in all data sets by 9.1 - 452.9. This shows that our D3-B outperforms C4.5 in accuracy and readability due to its appropriateness to a multi-objective classification task.

Figure 2b shows the effect of flower nodes on the accuracy and the number of split nodes. Concerning the accuracy, D3-B outperforms D3-B without flower nodes in "hepatitis", "Australian", and "German" by 7.9 % - 11.8 %; and ap-

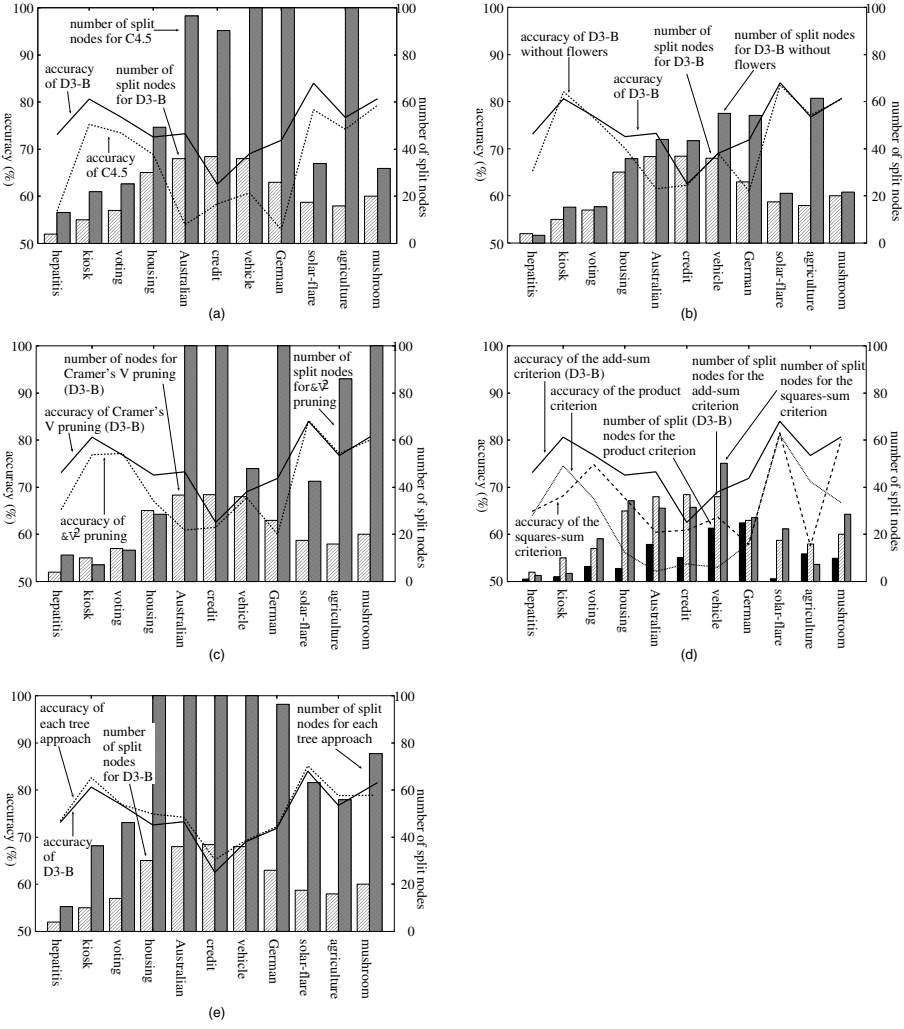

Fig. 2. Experimental results. (a) Accuracies and numbers of split nodes of D3-B and C4.5. For C4.5, the numbers of nodes for "vehicle", "German", and "agriculture" are 109, 152, and 468 respectively. (b) Effect of flower nodes on the accuracies and the numbers of split nodes. (c) Accuracies and numbers of split nodes of D3-Bs with Cramér's V pruning and χ^2 pruning. The numbers of nodes of the latter method for "Australian", "credit", "German", and "mushroom" are 132, 125, 179, and 103 respectively. (d) Accuracies and numbers of split nodes of D3-Bs with the add-sum criterion, the product criterion, and the squares-sum criterion. (e) Comparison of D3-B and a method which constructs a decision tree for each class dimension. The numbers of nodes of the latter method for "housing", "Australian", "credit", and "vehicle" are 101, 144, 143, and 119 respectively. Note that this comparison should be treated differently since the latter method constructs multiple trees.

proximately ties with it in the rest (the difference is less than 1.6 %). Concerning the number of split nodes, our method has a smaller number in eight data sets by 3.5 - 45.5, and approximately ties with the other in the rest (the difference is less than 1.7). We can conclude that the use of flower nodes almost always improves readability and occasionally improves accuracy. This result is due to the fact that a bloomy decision tree, unlike a conventional decision tree, gradually simplifies a multi-objective classification task with flower nodes inside the tree.

Figure 2c shows the influence of pruning methods on the accuracy and the number of split nodes. Since χ^2 pruning is known to produce an overly large decision tree for a large data set, data sets on the horizontal axis are sorted in ascending order with respect to their numbers of examples. Concerning the accuracy, our method outperforms χ^2 pruning in five data sets by 3.7 % - 12.4 %, and approximately ties with it in the rest six (the difference is less than 1.1 %). Concerning the readability, our method outperforms χ^2 pruning in eight data sets (7.2 - 152.8 smaller), and approximately ties with it in the rest three (0.6 - 2.9 larger). These three data sets correspond to the second, the third, and the fourth smallest data sets. These facts show that our Cramér's V pruning tolerates the ineffectiveness of χ^2 pruning in readability when the training set is large.

Figure 2d shows the effect of criteria on the accuracy and the number of split nodes. Concerning the readability, the product criterion produces trees with the smallest number of nodes in seven data sets by 7.5 - 24.4, which seems significant, and also trees with the smallest number of nodes in three data sets although the difference is smaller (less than 1.6). "Agriculture" is the only exception since the squares-sum criterion produces trees with a smaller number of nodes by 4.4. We attribute these results to the fact that the product criterion selects an attribute which splits all class dimensions well. We could not judge superiority between the other two criteria from these experiments. Concerning the accuracy, the add-sum criterion outperforms other criteria in all data sets. Especially, the difference seems significant in seven data sets (4.9% - 13.7%). We attribute these results to the fact that our criterion is, as we discussed in Sect. 3.2, robust concerning the distribution of gain ratios of class dimensions, and is thus adequate for prediction of multi-objective classification.

Figure 2e shows comparison of D3-B and a method which constructs a decision tree for each class dimension. We see that the differences of accuracies seem not significant since they are less than 2.7% in all data sets. For readability, however, D3-B always constructs a smaller tree (the difference is at least 6.6, which seems significant). We consider that these results are due to the fact that D3-B constructs a single tree while the other method constructs multiple trees. Moreover, as we mentioned in Sect 1, readability is much worse than it appears in the latter method since analysis based on multiple trees is difficult.

We also investigated these methods by varying the number m of class dimensions from two to five. Our method typically has no clear advantage for $m = 2$, but gradually outperforms other methods as m increases. It should be noted that no clear difference was observed for relatively "easy" data sets such

as "housing". We didn't tried for $m \geq 7$ to avoid the effect that the number of attributes becomes smaller as m increases.

From these results, the superiority of our method in accuracy and/or readability has been empirically proved against other methods. We consider that this superiority demonstrate that our proposals of the flower node, the add-sum criterion, and the Cramér's V pruning are effective in multi-objective classification.

6 Conclusion

In this paper, we proposed a learning algorithm for a novel decision tree from a multi-objective data set. Conventional learning algorithms are ineffective in constructing an accurate and readable decision tree in multi-objective classification. Our D3-B constructs a single classifier: a bloomy decision tree for such a data set. In a bloomy decision tree, the number of class dimensions gradually decreases by the use of flower nodes inside the tree. Experiments with eleven data sets showed that our D3-B, compared with C4.5 and other methods, typically constructs more accurate and/or smaller trees.

References

1. Blake, C., Keogh, E., and Merz, C. J.: UCI Repository of Machine Learning Databases, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, Univ. California, Irvine, Dept. Information and Computer Science (1998).
2. Breiman, L., Friedman, J., Olshen, R., and Stone, C. A.: *Classification and Regression Trees*, Chapman & Hall, New York (1984).
3. Caruana, R.: "Multitask Learning", *Machine Learning*, Vol. 28, No. 1, pp. 41–75 (1997).
4. Caruana, R.: "Multitask Learning", *Ph. D. Thesis, CMU-CS-97-203*, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, Pa. (1997).
5. Dougherty, J., Kohavi, R., and Sahami M.: "Supervised and Unsupervised Discretization of Continuous Features", *Proc. Twelfth Int'l Conf. on Machine Learning (ICML)*, pp. 194–202 (1995).
6. Forouraghi, B., Schmerr, L. W., and Prabhu, G. M.: "Induction of Multivariate Regression Trees for Design Optimization", *Proc. Twelfth Nat'l Conf. on Artificial Intelligence (AAAI)*, pp. 607–612 (1994).
7. Kendall, M. G.: *Multivariate Analysis, second edition*, Charles Griffin, High Wycombe, England (1980).
8. Mingers, J.: "An Empirical Comparison of Pruning Methods for Decision-Tree Induction", *Machine Learning*, Vol. 4, No. 2, pp. 227–243 (1989).
9. Murthy, S. K.: "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey", *Data Mining and Knowledge Discovery*, Vol. 2, No. 4, pp. 345–389 (1998).
10. Quinlan, J. R.: "Induction of Decision Trees", *Machine Learning*, Vol. 1, No. 1, pp. 81–106 (1986).
11. Quinlan, J. R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, Calif. (1993).