International Journal of Innovative
Computing, Information and Control
Volume **15**, Number **1**, February **2019**

ICIC International ©2019 ISSN 1349-4198
pp. **53–66**

PREFERENCE RULE TREES FOR PAIRWISE PREFERENCE LEARNING

Qiang Li[1] and Lihong Wang[2,*]

[1]School of Economics and Management
[2]School of Computer and Control Engineering
Yantai University
No. 30, Qingquan Road, Laishan Dist., Yantai 264005, P. R. China
liqiang@ytu.edu.cn; *Corresponding author: wanglh@ytu.edu.cn

Received April 2018; revised August 2018

Abstract. *This paper investigates pairwise preference learning from a preference database and an object database. We propose preference rule trees (PRTs) for preference learning based on the combination of classical decision trees and CP-nets. PRTs are able to extract the preference rules quickly and represent them efficiently. Specially, PRTs can handle binary, categorical and numeric preference data with low time complexity, so PRTs are feasible on diversified big datasets. Confidence maximum is pursued in the attributes selection and rules generation in order to make PRTs predict pairwise preferences of objects with more confidence. Experiments on the artificial datasets and SUSHI dataset show that, PRTs predict the pairwise preferences between objects with desirable accuracy even with little training data, and the found preference rules are interpretable.*
Keywords: Preference learning, Decision trees, Conditional preference networks, Preference rules, Rule confidence

1. **Introduction.** Preferences have been studied in many scientific fields, such as decision theory, economics, operations research and computer science. One of the main problems in preference elicitation is automatic acquisition of preference rules, which is also the main issue of recommendation systems. A recommendation system suggests the items expected to be preferred by the users. Recommendation systems usually use collaborative filtering to recommend items by summarizing the preferences of people who have tendencies similar to the user preference [1-4]. The potential assumption of collaborative filtering is that some users share the same or similar preference patterns. In this paper, we try to acquire the common pairwise preference patterns from a preference database and an object database and represent them efficiently by a combination of CP-nets and decision trees.

CP-nets (conditional preference networks) are widely studied in preference learning [5-9], since a CP-net is a simple and intuitive graphical tool that can easily express conditional preferences between attribute variables [5]. A CP-net is a directed acyclic graph whose nodes correspond to attribute variables, each annotated with preferences over the variable's values [7]. The number of conditional preference rules is exponential to the number of the attribute variables. This makes the space complexity of CP-nets too high [9].

Decision trees are well-known classification models with clear structures and semantic interpretations [10]. A decision tree is a directed tree that consists of internal nodes and leaves. In a decision tree, each internal node splits the instance space according to the test attribute's value, while each leaf is assigned with the most appropriate class label.

DOI: 10.24507/ijicic.15.01.53

Decision trees can express conditional dependences between a node and its children nodes conveniently; however, decision trees are classification models and it is not straightforward to use in preference learning.

In this paper, preference learning between pairwise objects is regarded as a classification problem, and new models, named *preference rule trees*, are proposed for preference learning based on the combination of decision trees and CP-nets.

The main contributions of this paper are as follows.

(1) Preference rule trees (PRTs) are proposed. PRTs have the advantages of decision trees and CP-nets, i.e., PRTs have distributed preference rules similar to CP-nets and distributed conditional preference dependences similar to decision trees.

(2) By maximizing confidence of preference rules, we provide the theoretical basis for PRTs. The proposed preference rules are not limited to Ceteris Paribus rules but have larger coverages than Ceteris Paribus rules.

(3) The algorithm for creating PRTs has a linear time complexity with the size of preference database and the number of attributes, so PRTs are suitable in the preference learning with big preference databases.

(4) The simulation experiments are carried out on the artificial datasets and SUSHI data set. Best to our knowledge, it is the first time to give preference rules explicitly based on data for SUSHI preferences. Experiments on the artificial datasets show that PRTs are able to predict pairwise preferences even with little pairwise preference training data.

The rest of the paper is organized as follows: Section 2 describes the preference learning problem and related work; Section 3 proposes the definition, creating and application of PRTs; Section 4 gives a theoretical analysis for the algorithms; In Section 5, experiments on the artificial datasets and SUSHI dataset are carried out and the results of the experiments are analyzed; Section 6 summarizes the full paper and gives the future work.

2. **Related Work.**

2.1. **Preference learning problem.**

Let ODB be an object database, ODB $= \{O_1, O_2, \ldots, O_n\}$, and each object $O_i \in$ ODB be represented by an attribute set A, $A = \{A_1, A_2, \ldots, A_m\}$, where n is the number of objects and m is the number of attributes respectively. A preference database whose tuples are pairwise preferences between objects [11] is defined as follows.

Definition 2.1. *Preference database (PDB)*

A preference database (or PDB for short) is a strict partial order over ODB, which is a binary relation PDB $\subseteq$ ODB $\times$ ODB satisfying the irreflexivity and transitivity properties. A PDB is anti-symmetric, i.e., $(O_j, O_i) \notin$ PDB if $(O_i, O_j) \in$ PDB, where $(O_i, O_j) \in$ PDB means a user prefers O_i to O_j.

Many users may have the same pairwise preferences, so we add a field named *count* to denote the occurrence number for each pairwise preference. A simple example for ODB and PDB is shown in Table 1. There are 7 objects in the ODB and 7 pairwise preferences in the PDB. Preference $(1, 2)$ with *count* 10 means there are 10 users who prefer object 1 to object 2, i.e., $O_1 \succ O_2$.

Preference is a partial order over ODB. For any objects O_1 and O_2, one and only one of the three propositions, $O_1 \succ O_2$, $O_1 \prec O_2$ or $O_1 \sim O_2$, must hold, where $O_1 \sim O_2$ denotes that O_1 and O_2 are incomparable or equal. This paper regards preference learning as a triple-classification problem, where $\{\succ, \prec, \sim\}$ are classes. The main objective of this paper is to extract a preference model from a preference database. The preference model is specified by a preference rule tree defined next.

TABLE 1. An object database and a preference database with *count*

ODB (object database)					PDB (preference database)		
ID	A_1	A_2	A_3	A_4	PID	Preference	Count
1	1	1	2	1	1	$(1, 2)$	10
2	1	2	2	2	2	$(3, 4)$	2
3	1	2	1	2	3	$(5, 6)$	4
4	1	1	1	1	4	$(4, 7)$	3
5	2	2	2	2	5	$(7, 6)$	5
6	2	1	2	1	6	$(4, 5)$	1
7	1	2	1	1	7	$(1, 6)$	1

2.2. **Learning of CP-nets and decision trees.** In CP-nets, each attribute node has a CPT (conditional preference table), which represents user's preferences between the attribute values under different assignments of the parent nodes. CP-nets learning can be grouped into active learning [6] and passive learning [7,8]. Active learning sets up the structure of a CP-net by asking questions to obtain the necessary information, such as equivalence relationships and membership relations [6]. On the contrary, passive learning is conducted based on an ODB and a PDB to create a CP-net satisfying the PDB. For example, Dimopoulos et al. constructed a CP-net by adding nodes step by step where all possible combinations of existing nodes were tested for searching the parent nodes of a candidate attribute [7]. Consequently, the more attributes, the higher time complexity. In order to equip every node with a complete CPT, satisfiability problem (SAT) had been solved. Thus, the CPT may have preference rules that have no supports in the PDB. Liu et al. considered the possible noise problem in a PDB, and used hypothesis test to test the parent nodes of each attribute to find the candidate CP-net best matching the PDB [8]. Since the space complexity of CP-nets is high, i.e., the number of preference rules is exponential to the number of attributes [9], only 3 attributes were tested for the SUSHI dataset [8]. CP-nets express the Ceteris Paribus rules, where two objects are only different in one attribute. Obviously, it is complicated to derive the preference between two objects with multiple different attributes. This paper focuses on passive preference learning and constructs a tree-like structure for pairwise preference rules, which are not limited to the Ceteris Paribus rules.

Decision trees can deal with many kinds of data, such as binary, categorical and numerical data [10]. A labeled dataset is used as training set to produce a decision tree, using an induction algorithm such as C4.5 or CART. Considering that the misclassification costs of instances may be different, the cost-sensitive decision trees were studied in recent years [12,13]. For cost-sensitive decision trees, the commonly used index, such as Gini or entropy, was replaced by an example-dependent cost-based non-purity measure to get better classification performance [12]. Some algorithms took account of the costs of acquiring attribute characteristics and used multiple machine learning techniques to minimize the costs of decision trees [13]. However, decision trees and variations are classification models and it is not straightforward to use in preference learning.

In addition, Hüllermeier and Fürnkranz regarded classification task as preference ranking for labels [14]. This work focuses on a particular learning scenario called label ranking, where the problem is to learn a mapping from instances to rankings over a finite number of labels. They first induced a binary preference relation from training data using pairwise classification, and then derived a ranking from the preference relation by a ranking procedure, whereby different ranking methods can be used for minimizing different loss functions [14].

Furthermore, fuzzy preference relations can be used to represent pairwise preference values, such as the intensity of preference and the uncertainty about the preference [15,16]. The missing values in a fuzzy preference relation might be caused by the incapacity of experts to quantify the preference degree due to lack of knowledge or data. Based on additive consistency, Herrera-Viedma et al. developed an iterative estimation procedure, called AC_IOWA, for estimating the missing values of an incomplete fuzzy preference relation in group decision making. Missing values can be estimated from the existing information using the consistency degree of information [17]. Fedrizzi and Giove proposed a different method, called GC_OPT, for calculating the missing values based on the resolution of an optimization problem to derive a complete fuzzy preference relation with maximum global consistency property [18]. Chiclana et al. proposed a reconstruction policy of incomplete fuzzy preference relations using both AC_IOWA and GC_OPT methods [19]. However, the underlying knowledge of preferences is not explicit even if the unknown preference of a pair of objects can be estimated by these methods.

In this paper, we regard pairwise preference learning as a classification task, and propose a new method, named *preference rule trees*, for user preference rules mining based on the combination of decision trees and CP-nets. By mining a preference database, we find the conditional dependences between attributes.

3. Proposed Preference Rule Trees.

3.1. Definition of preference rule trees.

Considering a rooted tree, a node with out-degree 0 is defined as a *leaf node* and the rest are *internal nodes*. Let n_0, n_1, ... n_k be a path from the root node n_0 to node n_k, then $n_0, n_1, \ldots, n_{k-1}$ is the prefix path of n_k.

Definition 3.1. PRT (preference rule tree) *A preference rule tree is a rooted tree whose nodes correspond to attribute variables, each annotated with an attribute $A_i \in A$. In a PRT, an internal node has one or more branches labeled with corresponding A_i values, while leaf nodes have no branches. Each node n_i in a PRT has a conditional preference table (or CPT for short) to record the preference between A_i values under the attribute values of the prefix path of n_i.*

Figure 1 shows an example of PRT, where the attribute set is $A = \{A_1, A_2, \ldots, A_5\}$.

In Figure 1, the root node has an attribute A_1 with value set $\{0, 1\}$. The preference rule $0 \succ 1$ in the CPT of the root node shows that, for any two objects $O_1 = 0^{****}$ and $O_2 = 1^{****}$, $O_1 \succ O_2$ holds with confidence 0.93. Obviously, this preference rule is independent of the rest attributes and strongly different from Cateris Paribus rules, in which the compared objects are required to have different values on one and only one attribute. Similarly, the left sub-tree represents preference rules $01^{***} \succ 00^{***}$ with

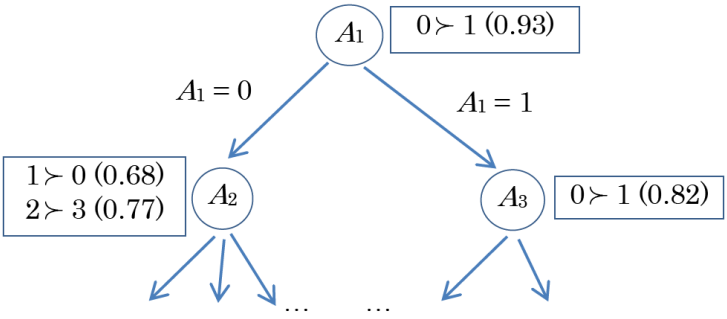


FIGURE 1. An example of PRT

confidence 0.68, and $02^{***} \succ 03^{***}$ with confidence 0.77. Meanwhile, the right sub-tree has rule $1^*0^{**} \succ 1^*1^{**}$ with confidence 0.82.

So PRTs can be seen as extended decision trees with CPTs in which the preference rules are not limited to the Ceteris Paribus rules.

3.2. **An algorithm for creating preference rule trees.** How to select the attribute for each node is a key problem in the creating of PRTs. Let D_t denote the preference data corresponding to node t, $D_t \subseteq$ PDB. We employ the information entropy to evaluate the preference impurity in node t for a candidate attribute A_i and assign node t with the minimal entropy attribute. The theoretical analysis is shown in Section 4.1.

$$\begin{aligned} &\mathit{entropy}\,(D_t, A_i, a_{ij}, a_{ik}) \\ &= -P\,(a_{ij} \succ a_{ik}) \log\,(P(a_{ij} \succ a_{ik})) - P(a_{ik} \succ a_{ij}) \log(P(a_{ik} \succ a_{ij})) \end{aligned} \tag{1}$$

where $P(a_{ij} \succ a_{ik})$ denotes the probability of $a_{ij} \succ a_{ik}$ in D_t.

$P(a_{ij} \succ a_{ik})$ can be represented as the probability of $O_j \succ O_k$, where $A_i(O_j) = a_{ij}$, $A_i(O_k) = a_{ik}$, $a_{ij} \neq a_{ik}$, and $P(a_{ij} \succ a_{ik}) = 1 - P(a_{ik} \succ a_{ij})$.

$$P(a_{ij} \succ a_{ik}) = \frac{\mathit{freq}(a_{ij} \succ a_{ik})}{\mathit{freq}(a_{ij} \succ a_{ik}) + \mathit{freq}(a_{ik} \succ a_{ij})} \tag{2}$$

where $\mathit{freq}(a_{ij} \succ a_{ik})$ denotes the frequency of $a_{ij} \succ a_{ik}$ in D_t.

$$\mathit{freq}(a_{ij} \succ a_{ik}) = \sum_{\substack{e\,=\,(O_j, O_k)\,\in\, D_t \\ A_i(O_j)\,=\,a_{ij},\, A_i(O_k)\,=\,a_{ik}}} e.\mathit{count} \tag{3}$$

where $e.\mathit{count}$ denotes the occurrence number of $e = (O_j, O_k)$ in the PDB, see Definition 2.1 and Table 1 for details.

When there are multiple pairwise comparisons of an attribute A_i in D_t, such as $^*0^* \succ {}^*1^*$, $^*3^* \succ {}^*2^*$, $^*2^* \succ {}^*1^*$, we need to calculate the entropy for each pair, and adopt the maximum one to evaluate the attribute A_i.

$$\mathit{entropy}(D_t, A_i) = \max_{\substack{(O_j, O_k)\,\in\, D_t \\ A_i(O_j)\,=\,a_{ij},\, A_i(O_k)\,=\,a_{ik}}} \mathit{entropy}(D_t, A_i, a_{ij}, a_{ik}) \tag{4}$$

Finally, the attribute A_t with minimal $\mathit{entropy}(D_t, A_i)$ is selected for node t.

$$A_t = \arg\min_{A_i \in A} \mathit{entropy}(D_t, A_i) \tag{5}$$

The attribute selection above is more complex than that of decision trees, for we consider multiple pairwise comparisons and minimize the maximum of preference impurity to ensure the minimal misclassification.

Now, we create a PRT in the following way:

Function t = Create_PRT (ODB, PDB, A) //Preference rule tree construction
Input: ODB, PDB, A (attribute set)
Output: PRT node t
Step1: **if** PDB is empty or A is empty **then**
return null.
Step2: [A_t, CPT_1, *subPDB*] = get_attribute (ODB, PDB, A);
Create a new node t, equip t with attribute A_t, and assign CPT_1 to the CPT of t.
Step3: **for** each value a_{ti} of attribute A_t appearing in *subPDB*,
3.1 find from *subPDB* the pairwise preferences with $A_t = a_{ti}$:

$$\mathit{Smallset}(a_{ti}) = \{(O_j, O_k) : (O_j, O_k) \in \mathit{subPDB} \wedge (A_t(O_j) = A_t(O_k) = a_{ti})\}$$

3.2 extend a child node n_1 for node t, assign $A_t = a_{ti}$ to the edge (t, n_1).

$$n_1 = Create\_PRT(ODB, Smallset(a_{ti}), A - \{A_t\});$$

end for

Step4: **return.**

Create_PRT () establishes the PRT by recursion. The function get_attribute () is called in Step2 to determine the attribute A_t and the CPT for the current node t, at the same time, *subPDB*, the set of pairwise objects with the same A_t value, is returned. Step3.2 creates recursively a subtree for each different A_t value and marks the A_t value on the edge. Since *subPDB* is a subset of PDB and $Smallset(a_{ti})$ is a subset of *subPDB*, the algorithm will terminate finally when $Smallset(a_{ti})$ becomes empty.

Function [A_t, CPT, *subPDB*] = get_attribute (ODB, PDB, A);
Input: ODB, PDB, A (attribute set)
Output: attribute A_t, CPT, *subPDB*.
Step1: Evaluate each attribute in A by Equation (4);
Step2: Find the attribute A_t by Equation (5). Set CPT as null.
Step3: **for** each pairwise A_t values a_{tj} and a_{tk}, $a_{tj} \neq a_{tk}$
if $P(a_{tj} \succ a_{tk}) > P(a_{tk} \succ a_{tj})$, **then**
Append a rule $a_{tj} \succ a_{tk}$ in the CPT with confidence $P(a_{tj} \succ a_{tk})$;
else
Append a rule $a_{tk} \succ a_{tj}$ in the CPT with confidence $P(a_{tk} \succ a_{tj})$.
end if
end for
Step4: find from PDB the pairwise objects with the same A_t values.

$$subPDB = \{(O_j, O_k) : (O_j, O_k) \in PDB \wedge (A_t(O_j) = A_t(O_k))\}$$

Step5: **return**

Function get_attribute () selects an attribute A_t for the current node t, appends rules into its CPT, and prepares preference data *subPDB* for its children nodes. Taking the ODB and the PDB in Table 1 as an example, Create_PRT () draws a PRT as shown in Figure 2.

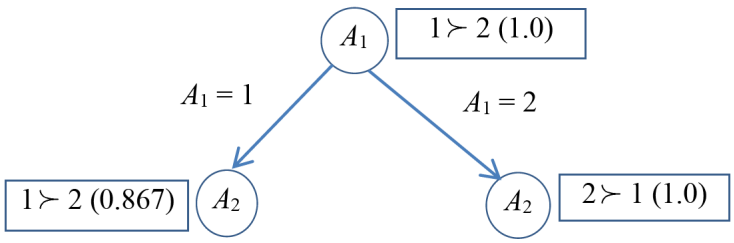


FIGURE 2. The preference rule tree for Table 1

In Figure 2, the root node is equipped with attribute A_1 and annotated with rule $1^{***} \succ 2^{***}$, which covers three tuples with PID = 5, 6, 7. The left child node has rule $11^{**} \succ 12^{**}$, which covers three tuples with PID = 1, 2, 4. The tuple with PID = 2 is negative to the rule, so the confidence of $11^{**} \succ 12^{**}$ is $(10 + 3)/(10 + 3 + 2) = 0.867$. The right child node has rule $22^{**} \succ 21^{**}$, which covers the tuple with PID = 3. We can see that the PDB is covered by these rules and the rules in the CPTs are not limited to Ceteris Paribus rules.

3.3. **Preference prediction based on preference rule trees.** PRTs are built to find the intrinsic preference rules for the pairwise preference data. For a new pairwise objects, the preference is predicted by matching the rules extracted from the training data. The details are as follows:

Function [*label*, *conf*] = classify(O_1, O_2, PRT)
Input: O_1, O_2 (pairwise objects), PRT
Output: *label* (+1 denotes $O_1 \succ O_2$, –1 denotes $O_2 \succ O_1$, and 0 denotes rejected)
conf (confidence of preference prediction)
Step1: **if** PRT is null **then**
Return 0 with confidence 1.0 // rejected
return
Step2: read the attribute A_r of the root node;
Step3: **if** $A_r(O_1) = A_r(O_2)$ //O_1 and O_2 have the same A_r value
then find the corresponding PRT_1 of $A_r(O_1)$, set PRT_1 = null if not found.
[*label*, *conf*] = classify(O_1, O_2, PRT_1);
return
else //matching the rules in the CPT of the root node
for each rule $A_{ri} \succ A_{rj}$ with confidence *cf*
if $A_r(O_1) = A_{ri}$ and $A_r(O_2) = A_{rj}$
then *label* = 1, *conf* = *cf*, return
if $A_r(O_1) = A_{rj}$ and $A_r(O_2) = A_{ri}$
then *label* = –1, *conf* = *cf*, return
endfor //end of rules matching
label = 0, *conf* = 1.0, return //no rule is matched, rejected
end // end of if
Step4: **return**

PRTs have a prediction procedure similar to decision trees. The difference is that a PRT may predict preference for a new pairwise objects in a non-leaf node, while a decision tree gets the label only at a leaf node, e.g., in Figure 2, 1122 $\succ$ 2122 can be determined at the root node. Thus, the rule matching is efficient in PRTs.

3.4. **Discussion on preference rule trees.** PRTs are different from classical decision trees. Firstly, node structures are different: each node in a PRT has a CPT to represent the preference rules, but a decision tree only assigns class labels to the leaf nodes. Secondly, decision positions are different: a PRT can make preference prediction at any node whose rule matches the pairwise objects, while a decision tree needs to reach the leaf node to determine the class label. Finally, the tree size (i.e., total number of the nodes) of a PRT is less than that of a decision tree if they have the same number of rules to represent. Each node in a PRT has a CPT to represent one or more rules, so the size of a PRT is less than or equal to the number of rules; however, a decision tree assigns class labels to the leaf nodes, hence the number of leaf nodes is equal to the number of rules and the size of a decision tree is larger than the number of rules.

There are also gaps between PRTs and CP-nets. On the one hand, a PRT is a tree, while a CP-net is a directed acyclic graph, not necessarily a tree. On the other hand, CP-nets are limited to the Ceteris Paribus rules, while PRTs derive the preference rules not limited to the Ceteris Paribus rules. Finally, the time complexity of the CP-nets learning algorithm is so high that it is difficult to learn even under some simplified assumptions [9], while the training and test time complexity of a PRT are approximately linear to the

size of PDB and the attribute number (see details in Section 4.2), so PRTs are suitable for discovering rules from large-scale preference data sets.

4. Algorithm Theoretical Analysis.

4.1. Algorithm basis.

Let D_t denote the PDB corresponding to the current node t, and $\mathrm{dom}(A_i) = \{a_{i1}, a_{i2}, \ldots, a_{is}\}$ be the domain of the attribute A_i, $A_i \in A$. All the numerical data have been discretized. If objects O_i and O_j have the same A_i value, then the preference between O_i and O_j is not determined by the attribute A_i, for O_i and O_j are indistinguishable on A_i. Creating a PRT, we focus on the set of pairwise objects with different A_i values, by which we derive some preferences between A_i values for node t.

Theorem 4.1. *If attribute A_i is selected for the current node t, then the conditional preference table (CPT) can be regarded as a rule set derived by a group of binary classifiers, each of which is determined by two different A_i values occurring in D_t.*

Proof: Let *Dpair* denote the set of pairs of different A_i values in D_t:

$$Dpair(D_t, A_i) = \{(a_{ij}, a_{ik}) : \exists (O_j, O_k) \in D_t, A_i(O_j) = a_{ij}, A_i(O_k) = a_{ik}, a_{ij} \neq a_{ik}\} \quad (6)$$

where D_t denotes the PDB corresponding to the current node t.

For each $(a_{ij}, a_{ik}) \in Dpair(D_t, A_i)$, we count the occurrence number of $a_{ij} \succ a_{ik}$ in D_t by counting the occurrence numbers of $(O_j, O_k) \in D_t$ in D_t, where $A_i(O_j) = a_{ij}$, $A_i(O_k) = a_{ik}$. In the same way, we obtain the occurrence number of $a_{ik} \succ a_{ij}$ in D_t.

Subsequently, $a_{ij} \succ a_{ik}$ is selected as the preference rule if the occurrence number of $a_{ij} \succ a_{ik}$ is bigger than that of $a_{ik} \succ a_{ij}$, and $a_{ik} \succ a_{ij}$ is selected as the preference rule otherwise. Hence selecting a preference rule can be regarded as a binary classifier, which is determined by two different A_i values occurring in D_t, as shown in Figure 3. After all tuples in $Dpair(D_t, A_i)$ have been processed, we assign the CPT of node t with the set of selected rules. □

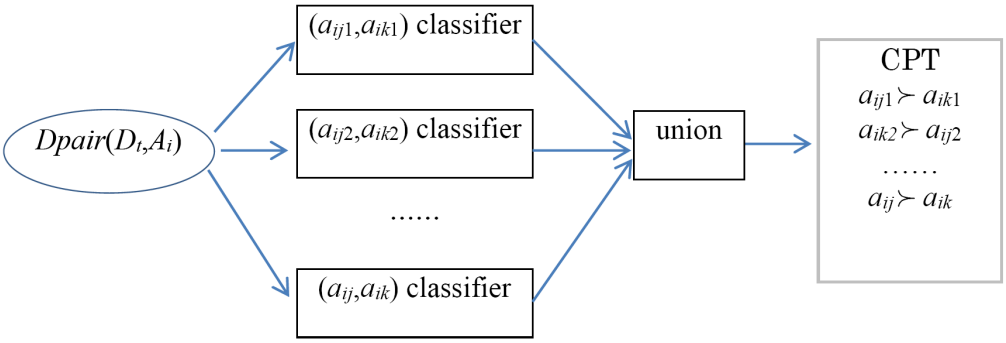


FIGURE 3. Preference rules selection

A rule can be represented as (LHS) → (RHS), where LHS is the condition of the rule and RHS is the conclusion of the rule. Coverage (support) and confidence are employed to evaluate a rule. Coverage is the proportion of the object number satisfying LHS to the total number of objects, while confidence is the proportion of the object number satisfying LHS and RHS to the object number satisfying LHS. Coverage emphasizes the generality of a rule, while confidence is the base of reasoning and classification, since only the rules with enough confidence are meaningful in applications.

Theorem 4.2. *Equation (5) selects the attribute which derives the preference rules with maximum confidences.*

Proof: $Dpair(D_t, A_i)$ is defined as Equation (6).

For any $(a_{ij}, a_{ik}) \in Dpair(D_t, A_i)$,

(1) If $(a_{ik}, a_{ij}) \notin Dpair(D_t, A_i)$, then $a_{ij} \succ a_{ik}$ is reasonable with confidence 100% in D_t.

(2) If $(a_{ik}, a_{ij}) \in Dpair(D_t, A_i)$, then the superior one of $a_{ij} \succ a_{ik}$ and $a_{ik} \succ a_{ij}$ is selected as the preference rule, as described in Theorem 4.1. Information entropy is a reasonable evaluation index, because the smaller entropy in Equation (1), the more uneven distribution of $a_{ij} \succ a_{ik}$ and $a_{ik} \succ a_{ij}$. In detail,

Assuming $x = \max\{P(a_{ij} \succ a_{ik}), P(a_{ik} \succ a_{ij})\}$, Equation (1) becomes

$$entropy(D_t, A_i, a_{ij}, a_{ik}) = -x\log(x) - (1-x)\log(1-x) \tag{7}$$

Because $f(x) = -x\log(x) - (1-x)\log(1-x)$ is a monotone decreasing function in $[0.5, 1)$, smaller value of Equation (7) corresponds to larger x and hence larger confidence of the preference rule.

Aiming at maximizing the confidences of rules derived by attribute A_i, Equation (4) gets the maximum entropy for each attribute, and then Equation (5) selects the attribute A_i which derives the preference rules with maximum confidences. □

4.2. **Time complexity analysis.**

(1) function get_attribute ()

This function calculates the entropy for each pairwise attribute value and finds the minimal maximum value for each attribute, so the time complexity is proportional to m (the number of attributes) and n (the size of PDB corresponding to current node), i.e., the time complexity of get_attribute () is $O(n * m)$.

(2) function Create_PRT ()

Step2 is the same as get_attribute (), while Step3 is more complex. Let k denote the number of different A_t values in D_t (the PDB corresponding to current node t), and n denote the size of D_t.

Let

$$r = |Dpair(D_t, A_t)|/n.$$

where $|Dpair(D_t, A_t)|$ is the size of $Dpair(D_t, A_t)$.

For simplification, we assume the *subPDB* size for each A_t value is approximately equal, and then we can estimate the size of *Smallset* as $(1-r) * n/k$.

Now, $T(n)$, the time complexity of Create_PRT (), is calculated as follows:

$$T(n) = \begin{cases} 1 & n = 0 \\ k * T(n * (1-r)/k) + O(n * m) & n > 0 \end{cases}$$

where $r > 0$, and $k < k/(1-r)$.

Thus, $T(n) = O(m * n/r)$ is derived according to the time complexity of recursive function. Larger r implies that an internal node covers more instances by itself and passes fewer instances to its children nodes; therefore, less time is cost.

(3) function classify ()

In the worst case, from the root to a leaf node, a pairwise objects pass the longest path whose length equals the number of attributes, so the time cost of passing is proportional to m. At the leaf node, we need to match the preference rules in the CPT to classify the pairwise objects. The number of rules in the CPT is $O(n)$, so the total time complexity of classify () is $O(m + n)$, where n is the PDB size.

In summary, we have the following Theorem 4.3.

Theorem 4.3. *The time complexity of creating a PRT is $O(m * n/r)$. In the worst case, the time complexity of predicting a pairwise preference is $O(m + n)$.*

5. Experiments and Results.

5.1. Introduction to databases.

(1) Artificial PDBs

We design 4 artificial PDBs, namely D200, D300, D500, and D500N, to evaluate the proposed PRTs. D500N is generated randomly as follows.

First of all, we randomly generate 500 different objects, each simulated by a 10 bits string to represent 10 binary attributes.

Secondly, we randomly select a pair of objects (O_i, O_j) and a positive integer *count* to compose a tri-tuple $< count, O_i, O_j >$. For example, we use tri-tuple $< 28, 13, 46 >$ to express preference $13 \succ 46$ with *count* 28.

Finally, 200 tuples are generated as above and saved in D500N. The random preferences between objects make D500N present no preference rules.

However, D500 is designed purposely, since there are four rules to follow for objects O_1 and O_2:

R_1: $O_1 \succ O_2$, if $A_1(O_1) = 0$ and $A_1(O_2) = 1$;

R_2: $O_1 \succ O_2$, if $A_1(O_1) = A_1(O_2) = 0$ and $A_2(O_1) = 0$ and $A_2(O_2) = 1$;

R_3: $O_1 \succ O_2$, if $A_1(O_1) = A_1(O_2) = 1$ and $A_9(O_1) = 1$ and $A_9(O_2) = 0$;

R_4: Randomly set $O_1 \succ O_2$ or $O_2 \succ O_1$, otherwise.

Similar to D500N, a positive integer is randomly assigned to the *count* field of a preference in D500.

Besides D500, other two PDBs, namely D200 and D300, are designed with the same rules $R_1 \sim R_4$, see Table 2 for details. In these PDBs, consistency (anti-symmetric relation) is guaranteed by deleting $O_1 \succ O_2$, if $O_2 \succ O_1$ exists or can be derived by transitivity.

(2) SUSHI

Besides these artificial PDBs, we employ SUSHI to evaluate our PRTs. SUSHI dataset is an online survey of Japanese preferences for sushi [3]. We downloaded two datasets SUSHI10 and SUSHI100 from http://www.preflib.org. SUSHI10 involves 10 kinds of SUSHI, and SUSHI100 involves 100 kinds of SUSHI. Each kind of SUSHI has 7 attributes, namely style, major group, minor group, heaviness/oiliness, eating frequency, normalized price and sell frequency (attribute item ID and name are deleted). We employ A_1, $A_2, \ldots, A_7$ to denote these attributes respectively. Table 2 shows details of all PDBs.

Table 2. Details of object database and preference database

ODB (object database)			PDB (preference database)
ID	Attributes	Instances	Instances
D200	10	200	200
D300	10	300	200
D500	10	500	200
D500N	10	500	200
SUSHI10	7	10	45
SUSHI100	7	100	4503

5.2. Experimental results.

There are two binary attributes (style, major group), one categorical attribute (minor group) with 12 available values, and four numerical attributes in SUSHI datasets. Liu et al. selected the attributes major group (A_2), heaviness (A_4), and normalized price (A_6) to create CP-nets, where normalized price (A_6) was discretized

to be a binary attribute [8]. We employ all the seven attributes to create PRTs and discretize four numerical attributes by k-means algorithm with cluster number 3.

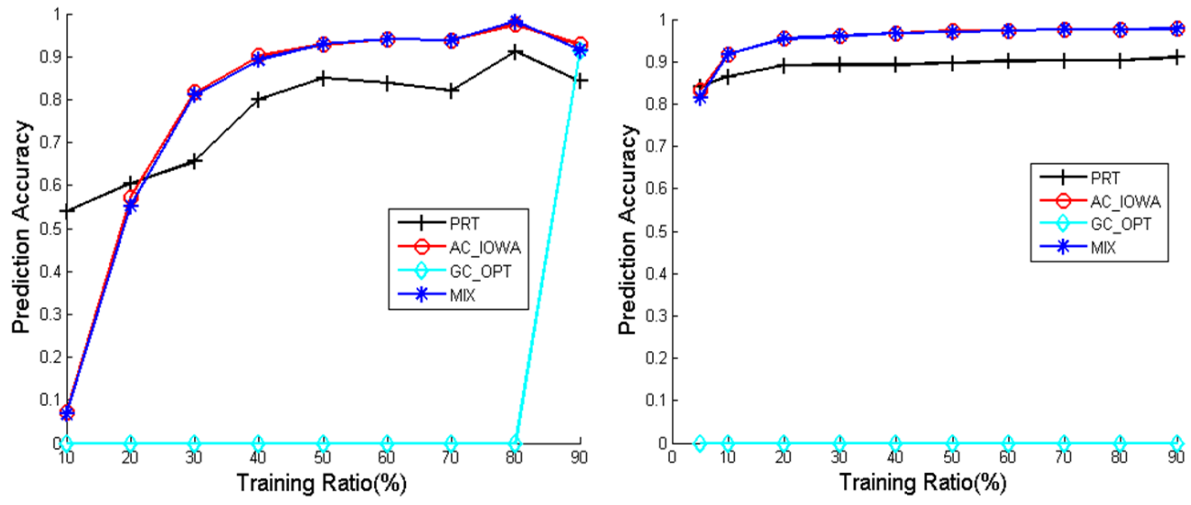
5.2.1. *Prediction accuracy of PRTs.* In order to evaluate the prediction accuracy of PRTs, we divide the preference dataset into training set and test set. A PRT is built based on the training set and tested by the test set. The prediction accuracy is defined as follows:

$$acc = \left(\sum_{\substack{e=(O_i,O_j)\in \mathit{Test} \\ O_i \succ' O_j}} e.count \right) \Big/ \left(\sum_{e=(O_i,O_j)\in \mathit{Test}} e.count \right)$$

where *Test* denotes the test set and $(O_i, O_j) \in \mathit{Test}$ means $O_i \succ O_j$. We use $O_i \succ' O_j$ to indicate that a PRT predicts "O_i prefers to O_j". The prediction is correct if $O_i \succ' O_j$ holds for $e = (O_i, O_j) \in \mathit{Test}$. Here, $e.count$ denotes the occurrence number of $e = (O_i, O_j)$ in the PDB, see Definition 2.1 and Table 1 for details.

In this paper, we compare PRTs mainly with fuzzy preference relation prediction algorithms. A fuzzy preference relation can be used to represent pairwise preference values, where the missing values can be estimated by algorithms AC_IOWA [17], GC_OPT [18] or MIX [19]. Figure 4 shows the comparison of PRTs with these algorithms on SUSHI10 and SUSHI100. In Figure 4(a), MIX overlaps with AC_IOWA when the training ratio is less than 90% and overlaps with GC_OPT when the training ratio is 90%. This is easy to interpret since the MIX algorithm was designed by using AC_IOWA and GC_OPT alternatively. The same is shown in Figure 4(b), where MIX overlaps with AC_IOWA on SUSHI100 all the time. Although the prediction accuracies of PRTs are inferior to AC_IOWA in most cases, PRTs outperform AC_IOWA and GC_OPT on SUSHI10 and SUSHI100 when the training ratio is small, i.e., only a little training data are available.

These experiments suggest that PRTs are capable of predicting preferences with little training data; therefore, we conducted experiments on the artificial datasets, see Table 2 in Section 5.1 for details. The algorithms AC_IOWA, GC_OPT, and MIX all failed to estimate missing values on these artificial datasets, so we only show the performance of PRTs in Figure 5. The prediction accuracies on D500N are less than 0.5 as expected since



(a) SUSHI10 (b) SUSHI100

FIGURE 4. Comparison with other algorithms

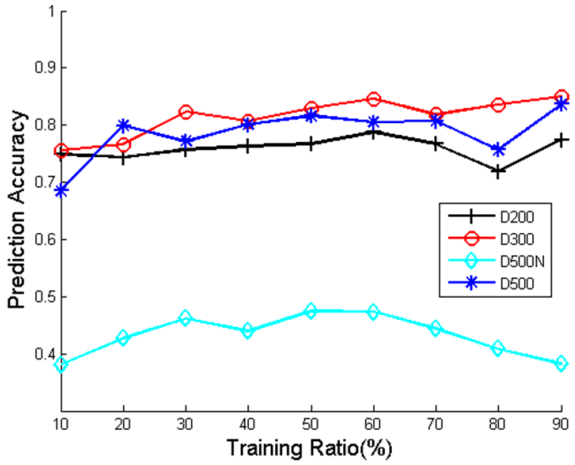


FIGURE 5. Performance on artificial datasets

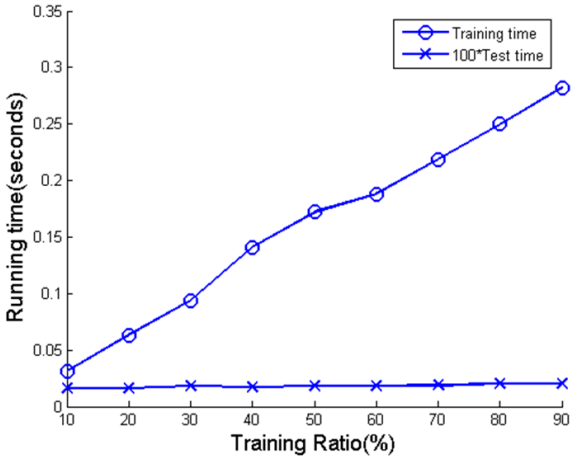


FIGURE 6. Running time of a PRT

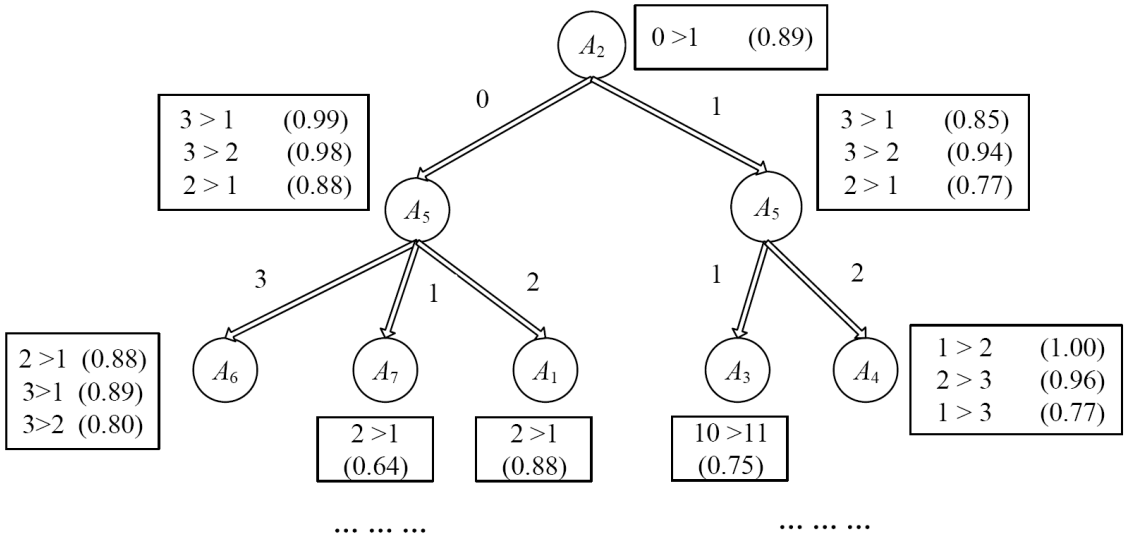


FIGURE 7. A PRT (part)

D500N was designed with random pairwise preferences. PRTs show preferable accuracies on other PDBs; thus, we can draw a conclusion that PRTs are useful in prediction preferences even if we have little pairwise preferences training data.

5.2.2. *Running time of PRTs.* In order to validate that PRTs can handle large-scale preference databases, the average running time of 30 runs on SUSHI100 is shown in Figure 6.

Figure 6 shows that the training time of a PRT is approximately linear with the training data size, while the test time of a pairwise preference is about 10e-3 seconds. Figure 6 magnifies the actual test time one hundred times in order to display the training time and the test time in the same figure. The experimental results are consistent with Theorem 4.3 which gives the complexity analysis for training time and test time.

5.2.3. *Some preference rules derived from a PRT.* We got a PRT when the training set was randomly selected from SUSHI100 and the training ratio was 70%. The top three levels are shown in Figure 7 due to space limits, see Section 5.1 for attribute descriptions.

In Figure 7, we have the following observations.

(1) The CPT of attribute A_2 (major group) illustrates that seafood is preferred by Japanese. It is easy to see that Japan is an island country with rich fishery resources, so fresh seafood is preferred naturally.

(2) Frequency of eating (A_5) reveals people's food preferences, the more food they eat, the more they like it. Whether seafood or not, frequency of eating shows the preference $3 \succ 2 \succ 1$, where attribute A_5 is discretized as $1 \leftrightarrow [0.4079, 0.9565]$, $2 \leftrightarrow [0.9931, 1.4939]$, $3 \leftrightarrow [1.5662, 2.3485]$.

(3) The combination of major group (A_2) and frequency of eating (A_5) determinates what is concerned next. For example, minor group (A_3) is concerned when $A_2 = 1$ and $A_5 = 1$, while normalized price (A_6) is observed when $A_2 = 0$ and $A_5 = 3$. The PRT in Figure 7 reflects multiple considerations in SUSHI preferences.

However, major group (A_2) and eating frequency (A_5) are more important attributes in SUSHI preferences.

6. **Conclusions.** Preference elicitation has been studied for a long time. This paper regarded pairwise preference prediction as a classification problem and proposed new preference rule trees (PRTs) to learn pairwise preferences from PDB and ODB. PRTs have the advantages of decision trees and conditional preference networks, i.e., preference rules are distributed in the tree, and the conditional preference dependences are represented by the attribute values on the branches. Theoretical analysis and experiments validated that the creating time of a PRT is generally linear with the number of attributes and the size of training dataset, so PRTs are feasible on big data sets. In the experiments on SUSHI datasets, we established a specific PRT in which the preference rules are interpretable. In the future research, we plan to investigate PRTs in the following aspects.

- Assess PRTs stability. PRTs are designed based on decision trees and CP-nets. A decision tree is easy to change when the training dataset is changed, this instability raises suspicion about the validity of the decision tree [20], so the stability of PRTs should be checked and quantified.
- Investigate ensemble techniques to further improve the prediction accuracies of PRTs.
- Compare PRTs with CP-nets on prediction accuracy and structure complexity.

Acknowledgments. This work is supported by National Key R&D Program of China (No. 2017YFB1400105) and a Project of Shandong Province Higher Educational Science and Technology Program (No. J17KA091). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] Y. Shi, M. Larson and A. Hanjalic, Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges, *ACM Computing Surveys*, vol.47, no.1, pp.1-45, 2014.

[2] H. Liu, Z. Hu, A. Mian, H. Tian and X. Zhu, A new user similarity model to improve the accuracy of collaborative filtering, *Knowledge-Based Systems*, vol.56, pp.156-166, 2014.

[3] T. Kamishima, Nantonac collaborative filtering: Recommendation based on order responses, *Proc. of ACM SIGKDD'03*, New York, USA, pp.583-588, 2003.

[4] J. Kim, D. Lee and K. Y. Chung, Item recommendation based on context-aware model for personalized u-healthcare service, *Multimedia Tools and Applications*, vol.71, no.2, pp.855-872, 2014.

[5] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos and D. Poole, CP-nets: A tool for representing and reasoning with conditional Ceteris Paribus preference statements, *Journal of Artificial Intelligence Research*, vol.21, no.1, pp.135-191, 2004.

[6] F. Koriche and B. Zanuttini, Learning conditional preference networks, *Artificial Intelligence*, vol.174, pp.685-703, 2010.

[7] Y. Dimopoulos, L. Michael and F. Athienitou, Ceteris Paribus preference elicitation with predictive guarantees, *Proc. of IJCAI'09*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp.1890-1895, 2009.

[8] J. Liu, Z. Yao, Y. Xiong, W. Liu and C. Wu, Learning conditional preference network from noisy samples using hypothesis testing, *Knowledge-Based Systems*, vol.40, pp.7-16, 2013.

[9] W. Liu, C. Wu, B. Feng and J. Liu, Conditional preference in recommender systems, *Expert Systems with Applications*, vol.42, no.2, pp.774-788, 2015.

[10] J. R. Quinlan, Induction of decision trees, *Machine Learning*, vol.1, no.1, pp.81-106, 1986.

[11] S. de Amo, M. L. P. Bueno, G. Alves and N. F. Silva, CPrefMiner: An algorithm for mining user contextual preferences based on Bayesian networks, *IEEE, International Conference on Tools with Artificial Intelligence (ICTAI)*, vol.1, no.4, pp.114-121, 2012.

[12] A. C. Bahnsen, D. Aouada and B. Ottersten, Example-dependent cost-sensitive decision trees, *Expert Systems with Applications*, vol.42, no.19, pp.6609-6619, 2015.

[13] S. Lomax and S. Vadera, A survey of cost-sensitive decision tree induction algorithms, *ACM Computing Surveys*, vol.45, no.2, pp.1-35, 2013.

[14] E. Hüllermeier and J. Fürnkranz, Label ranking by learning pair-wise preference, *Artificial Intelligence*, vol.172, no.16, pp.1897-1916, 2008.

[15] J. Bezdek, B. Spillman and R. Spillman, A fuzzy relation space for group decision theory, *Fuzzy Sets and Systems*, vol.1, no.4, pp.255-268, 1978.

[16] H. Nurmi, Approaches to collective decision making with fuzzy preference relations, *Fuzzy Sets and Systems*, vol.6, no.3, pp.249-259, 1981.

[17] E. Herrera-Viedma, F. Chiclana, F. Herrera and S. Alonso, Group decision-making model with incomplete fuzzy preference relations based on additive consistency, *IEEE Trans. Systems, Man, and Cybernetics – Part B: Cybernetics*, vol.37, no.1, pp.176-189, 2007.

[18] M. Fedrizzi and S. Giove, Incomplete pairwise comparison and consistency optimization, *European Journal of Operational Research*, vol.183, no.1, pp.303-313, 2007.

[19] F. Chiclana, E. Herrera-Viedma and S. Alonso, A note on two methods for estimating missing pairwise preference values, *IEEE Trans. Systems, Man, and Cybernetics – Part B: Cybernetics*, vol.39, no.6, pp.1628-1633, 2009.

[20] L. Wang, Q. Li, Y. Yu and J. Liu, Region compatibility based stability assessment for decision trees, *Expert Systems with Applications*, vol.105, pp.112-128, 2018.