Knowledge-Based Systems xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Learning conditional preference network from noisy samples using hypothesis testing

Juntao Liu [a,b], Zhijun Yao [a], Yi Xiong [a], Wenyu Liu [a,*], Caihua Wu [c]

[a] Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
[b] Department of Computer Engineering, Mechanical Engineering Institute, Shijiazhuang 050003, China
[c] Information Combat Commanding Teaching and Research Section, Information Countermeasure Department, Air Force Radar Academy, Wuhan 430010, China

ARTICLE INFO

Article history:
Received 17 May 2012
Received in revised form 3 November 2012
Accepted 7 November 2012
Available online xxxx

Keywords:
Preference learning
Conditional preference
Conditional preference networks
Hypothesis testing
Chi-squared testing

ABSTRACT

The problem of learning Conditional Preference Networks (CP-nets) from a set of pairwise comparisons between outcomes has received great attention recently. However, because of the randomicity of the users' behaviors or the observation errors, there exists some noise (errors) in the training samples. Most existing methods neglect to handle the case with noisy samples. In this work, we introduce a new model of learning CP-nets from noisy samples. Based on chi-squared testing, we propose an algorithm to solve this problem in polynomial time. We prove that the obtained CP-net converges in mean to initial CP-net as sample size increases. The proposed method is verified on both simulated data and real data. Compared with the previous methods, our method achieves more accurate results on noisy sample sets.

1. Introduction

Preference learning (or preference elicitation) is a critical issue in many scientific fields, such as decision theory [1,2], economics [3,4] and database [5]. Conditional Preference Network (CP-net) [6], as a simple and intuitive graphical tool for representing and reasoning conditional preferences over outcome space, has received increasing attentions in recent years. In the outcome space, each outcome is represented by several variables(features). A CP-net is a digraph, whose nodes correspond to variables. Each node is annotated with a conditional preference table. The preferential dependence is represented by the graphical structure. CP-nets have various extensions [7–12]. Although CP-nets represent the preferential dependence between variables compactly by graphical structure, learning CP-nets in complex domains is a difficult task.

CP-nets learning methods can be divided into active learning methods [13] and passive learning methods [14–16]. Different from active learners, passive learners do not interact with users. They collect the set $S = \{o_i \succ o_i'\}$ of samples by observing users passively, where a sample $o_i \succ o_i'$ means that the outcome o_i is strictly preferred over outcome o_i'. Such set of samples may be gathered, for instance, by observing online users' choices. Because of randomicity of the users' behaviors or the observation errors, there exists some noise in set S. Let CP-net N_0 represent a user's real preference. Let $\succ_{N_0}$ denote the preference relation entailed by N_0. If $o \succ_{N_0} o'$, sample $o' \succ o$ is observed with probability p, and sample $o \succ o'$ is observed with probability $1-p$. Here, $p < 0.5$ is called noise level. The learner does not have any prior about CP-net N_0 and noise level p. A CP-net N is learned just from the noisy samples S. To predict the user's preference accurately, the learned CP-net N should converge to CP-net N_0 as long as the learner get sufficient samples. Formally, we can give the model:

$$\lim_{|S|\to\infty} N = N_0 \tag{1}$$

Existing methods [9,14–16], assuming the set S of samples is noise-free, are different from our model and cannot solve formula (1). For instance, active learning method proposed in [13] cannot solve passive learning problem. The learning method proposed in [16] assumes that every variable is preferentially independent, which means no edges in CP-nets. The algorithm for learning the CP-nets over a fixed acyclic digraph was proposed in [15]. Constraints on structure of CP-nets restrict the application of these two methods. In [14], conditional preference tables are learned by solving a set of 2-satisfiability (2SAT) problems, and it assumes that the training samples are transparent entailed (see Definition 5 in [14]) by an acyclic CP-net. If this assumption is not satisfied, some of the 2SAT problems are unresolvable.

* Corresponding author.
E-mail addresses: prolay@163.com (J. Liu), alexyau78@gmail.com (Z. Yao), bearone@live.cn (Y. Xiong), liuwy@hust.edu.cn (W. Liu), wucaihua_1999@yahoo.com.cn (C. Wu).

0950-7051/$ - see front matter Crown Copyright © 2012 Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.11.006

To learn a CP-net, one should decide whether a variable X is preferentially dependent on a given set U of variables, i.e., whether the preference relation over values of variable X is dependent on the values of variables in U. In this work, we adopt the hypothesis testing to solve this problem. The hypotheses are:

H_0 : X is not preferentially dependent on U

H_1 : X is preferentially dependent on U

We find that the distributions of variables in U in preferred outcomes are different from those in all outcomes in training samples, even if the training samples are noisy. So the above hypotheses can be tested by chi-squared testing. In this way, we propose an algorithm to solve model (1).

The main contributions of this work are twofold. (1) We propose a new model of learning CP-nets from noisy samples. Although the problem of learning CP-nets has received great attention, the problem of learning CP-nets from noisy samples has seldom been addressed. (2) We propose a method to solve the new model in polynomial time. The obtained CP-net converges in mean to initial CP-net as the size of samples increases, which means the difference between the CP-net learned by the proposed algorithm and the initial CP-net tends to be 0 as the number of training samples increases.

The rest of the paper is organized as follows. In Section 2, we briefly review the related work. In Section 3, we introduce necessary background on CP-nets. In Section 4, the learning algorithm using hypothesis testing is presented. Its properties are given. In Section 5, the proposed method is verified on simulated data and real data, and it is also compared with RankNet [17], AdaRank [18], Coordinate Ascent [19] and Dimopoulos's method [14]. Finally, we conclude and mention some possible future researches in Section 6.

2. Related work

A CP-net represents a partial order over outcome space. Learning CP-net can be regarded as learning a preference order from pairwise comparisons. So our work is related to learning preference order. This problem is also called learning to order things [20] or supervised ordering [21]. In this scenario, the learner is provided with a set of pairwise comparisons over a set of outcomes, and the task is to predict the preference order for a new set of outcomes. If the predicted preference order is required to be a total order, the problem is also called the ranking problem. The existing approaches for learning preference order can be divided into three categories: the utility function based approaches [21–25,19,18,26,27,17,28,29], the related preference order based approaches [20] and the preference model based approaches [30–32,13–16]. Now, we give a brief survey of these approaches.

2.1. Utility function based approaches

The value of utility function, numerical or ordinal, represents a degree of preference for each outcome. Some special constraints should be taken into consideration such as the bound of the bounded range and the monotonicity of the utility function. The utility function can represent a total order. So learning utility function is called learning to rank in information retrieval.

Expected rank regression method [21] proposed by Kamishima et al. learns the utility function by the common regression method. The order is estimated based on Thurstonian model according to the utility function. Metzler and Bruce Croft [19] proposed a linear utility function and minimized the ranking error by coordinate ascent.

RankBoost [22] proposed by Freund is a boosting algorithm. The algorithm trains a set of weak learners and selects weights for them. The final order is predicted according to the weighted sum of outputs of weak learners, which can be regarded as a utility function. RankBoost minimizes Kendall distance between training samples and the predicted order. AdaRank [18] is also a boosting algorithm. Different from RankBoost, AdaRank optimizes information retrieval performance measure function, such as Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG), directly.

Some approaches based on SVM, such as Order SVM [23], Support Vector Ordinal Regression [24] and Ranking SVM [25], learn the utility function by SVM. Their formulation are very similar to standard SVM. Sometimes a single hyperplane in SVM based approaches cannot tackle complex ranking problems. To overcome this shortcoming, Multi-Hyperplane Ranker [26] trains several base rankers using Ranking SVM. The orders obtained by base rankers are aggregated as the final order. To handle outliers in training samples robustly, Carvalho et al. [27] introduced a sigmoid-based loss function to replace the objective function in Ranking SVM.

RankNet [17] tries to minimize the ranking error represented by the cross entropy cost function using neural network. FRank [33] uses Fidelity loss function instead. To optimize Normalized Discounted Cumulative Gain (NDCG) directly, LambdaRank [28] simply multiplies gradient of loss function in RankNet by the size of the change in NDCG. Integrating LambdaRank and Multiple Additive Regression Trees (MARTs) [34], Wu et al. [29] proposed LambdaMART. Ranking methods stemming from RankNet, such as LambdaRank, LambdaMART, gained great success in information retrieval and won Track 1 of the 2010 Yahoo! Learning To Rank Challenge [35].

In recent years, some list-wise ranking methods, such as ListNet [36] and Top-k ListNet [37], have been proposed. List-wise methods take order lists of outcomes as training samples. In this paper, we mainly focus on learning preference from pairwise comparisons, list-wise methods are out of the scope.

2.2. Related preference order based approaches

The main task of related preference order based approaches is to learn a binary preference relation over outcomes. The preference order is extracted from the learned binary preference relation. Because the pairwise comparisons in training samples can be used directly, the models in this kind of approaches are very simple. Related preference order based approaches generally try to minimize the number of outcome pairs conflicted with the training samples, which leads to NP-hard problem. So approximation algorithms should be found. Cohen et al. [20] proposed a method to maximize the sum of the binary preference function for all possible outcome pairs. This maximization problem is known as the weighted feedback arc set problem, and is NP-hard. The greedy algorithm in [20] can find an approximate solution in $O(n^2)$. Besides the computational complexity, the other drawback of this kind of approaches is that the extracted preference order is normally not unique, because the preference relation may not be consistent or complete.

2.3. Preference model based approaches

Besides two kinds of approaches above, there are approaches based on special preference models. The main difference between this kind of approaches and related preference order based approaches is that the preferences are represented by preference models in preference model based approaches. The task of preference model based approaches is to learn the preference model from training samples. The learning methods in this kind of approaches are very different from those in related preference order based approaches.

Please cite this article in press as: J. Liu et al., Learning conditional preference network from noisy samples using hypothesis testing, Knowl. Based Syst. (2012), http://dx.doi.org/10.1016/j.knosys.2012.11.006

Lexicographic Order Model (LPM) [30], for example, is a simple preference model. In LPM, importance order over variables are defined, which is used to determine the preference order between outcomes. Schmitt and Martignon [30], proposed a greedy variable permutation algorithm guaranteeing to find one of the LPMs that is consistent with the training samples, if such one exists. They also proved that for the noisy training samples, the problem of finding a LPM that does not violate more than a constant number of pairwise comparisons in training samples is NP-complete. Based on the algorithm in [30], Yaman et al. [31] proposed a method to predict the preference order from a set of LPMs consistent with the training samples. The assumption of LPM is strong, because the preference on individual variable is sometimes independent on other variables.

CP-net is proposed to address the conditional preference. In term of learning CP-nets from consistent examples, Koriche and Zanuttini [32,13] proposed an active learning algorithm. In [16], the algorithm for learning separable ceteris paribus structures (SCP-structures) was proposed. SCP-structure is the simplified CP-net, in which parent variable set of every variable is empty. The algorithm for learning CP-nets over a fixed acyclic digraph was proposed in [15]. In [14], an algorithm for generating an acyclic CP-net entailing all training samples was proposed. This algorithm was proved to be a PAC-learner under the transparent entailment assumption (see Definition 5 in [14]). It was proved that the problem of learning CP-nets is intractable even under some simplifying assumptions [14,38]. The method proposed in [14] is the most related to our method. We compare this method with ours in Section 5.

In this paper, we propose an approach to learn CP-nets from inconsistent pairwise training samples, which is a preference model based method. The main differences between previous approaches and ours are: (1) Compared with utility function based approaches and related preference order based approaches, our approach learns a preference model, CP-net, which can represent conditional preference. (2) Compared with other CP-net learning approaches, our approach learns CP-nets from inconsistent training samples. We do not require that all of the training samples are entailed by CP-nets. Our method can be applied widely than other CP-nets learning methods. (3) Compared with active CP-net learning approaches, our approach learns CP-net passively. The training examples are collected silently and the users do not need to be disturbed.

3. Background on CP-nets

We start by briefly reviewing the basics of CP-net.

Let $V = \{X_1, X_2, \ldots, X_n\}$ be a set of variables. Each variable X_i is associated with a domain of values $Dom(X_i)$. For a set $U \subset V$ of variables, $Dom(U) = \prod_{X_i \in U} Dom(X_i)$. $\Omega = Dom(V)$ denotes the outcome space. $o \in \Omega$ is called an outcome. $o[U]$ denotes the projection of o on a set U of variables. Let $P(x) = P(X_i = x)$ denote the probability of a value $x \in Dom(X_i)$ such that $\sum_{x \in Dom(X_i)} P(x) = 1$. In this paper, we assume the distributions of variables are independent, that is $P(X = x, Y = y) = P(X = x)P(Y = y)$.

Preference relation $\succ$ is a strict partial order on Ω, i.e., an irreflexive, transitive and asymmetric relation. $o \succ o'$ means outcome o is strictly preferred over outcome o'. A conditional preference rule (CPR) on a variable X is an expression: u: $x \succ x'$, where x, $x' \in Dom(X)$, u is the assignment of set $U \subset V$ of variables, and the variables in U are called the parent variables of X denoted by $Pa(X)$. Such a rule means given that u hold, x is preferred over x', all the other variables being equal. A conditional preference table (CPT) on a variable X, $CPT(X)$, is the set of CPRs on X. If for any two values x and x' of variable X, a CPR is associated with each assignment of $Pa(X)$, $CPT(X)$ is complete. The definition of CP-nets given in [39] is as follows:

Definition 1. A **CP-net** N over set $V = \{X_1, X_2, \ldots, X_n\}$ of variables is a labeled digraph over V. Each node X_i is annotated with a conditional preference table $CPT(X_i)$, with respect to the set $Pa(X_i)$ of parents of X_i in the digraph.

If all conditional preference tables in a CP-net are complete, this CP-net is **complete**. If the digraph of a CP-net is acyclic, this CP-net is **acyclic**. Some cyclic CP-nets are not satisfiable [39], and cannot represent real-life users' preferences. While, every acyclic CP-net is satisfiable [39]. So in this paper, we only consider acyclic CP-nets.

In this paper, we define the size $|N|$ of a CP-net N as the total number of conditional preference rules in N. For a complete CP-net N,

$$|N| = \sum_{i=1}^{|V|} \frac{1}{2} |Dom(X_i)|(|Dom(X_i)| - 1)|Dom(Pa(X_i))| = \frac{1}{2} \sum_{i=1}^{|V|} |Dom(X_i)|(|Dom(X_i)| - 1) \prod_{X_j \in Pa(X_i)} |Dom(X_j)| \quad (2)$$

where $|\cdot|$ denotes the size of a set. For general CP-nets, the maximal size of CP-net N over V is

$$|N|_{max} = \frac{1}{2} \sum_{i=1}^{|V|} |Dom(X_i)|(|Dom(X_i)| - 1) \prod_{j \neq i} |Dom(X_j)| \quad (3)$$

In this case, every variable is preferential dependent on the other variables, and every conditional preference table is complete.

In [39], preference graphs are used to describe the preference relation between outcomes directly induced from CP-nets. Now we give the formal definition of preference graphs.

Definition 2. A **preference graph** G is a digraph over outcome space Ω. Each node corresponds to an outcome in Ω. For any two outcomes o, $o' \in \Omega$ differing on only one variable, if $o \succ o'$, $(o', o) \in E$, where E is the set of edges in G. If for any two outcomes x and y differing on only one variable $(y, x) \in E$ or $(x, y) \in E$ hold, the preference graph is **complete**.

There are $|\Omega|$ nodes in a preference graph. Each node (outcome) connects to other nodes differing on only one variable with it. So each node in the preference graph connects to $m = \sum_{i=1}^{n} (|Dom(X_i)| - 1)$ other nodes at most. Considering the antisymmetry of preference relation, there are $k = m|\Omega|/2$ edges in a preference graph at most.

Here we use the example in [39] to illustrate a CP-net and the induced preference graph.

Example 1. Let us consider the evening dress scenario [39]. The set $V = \{J, S, P\}$ of variables stands for jacket, pants and shirt, respectively. The domains $Dom(J) = \{J_b, J_w\}$ stands for black jacket and white jacket, respectively; $Dom(P) = \{P_b, P_w\}$ stands for black pant and white pant, respectively; $Dom(S) = \{S_r, S_w\}$ stands for red shirt and white shirt, respectively. The user unconditionally prefers black to white as a color for both the jacket and the pants. While her preference between the red and white shirts is conditioned on the combination of jacket and pants: if they have the same color, the user prefers a red shirt. Otherwise, if the jacket and the pants are of different colors, the user prefers a white shirt. Fig. 1 shows the CP-net and the induced preference graph.

Next, we define the difference between two CP-nets as the difference between the induced preference graphs.

Definition 3. Given two preference graphs G_1 and G_2 over same outcome space, the difference between G_1 and G_2 is defined as:

Please cite this article in press as: J. Liu et al., Learning conditional preference network from noisy samples using hypothesis testing, Knowl. Based Syst. (2012), http://dx.doi.org/10.1016/j.knosys.2012.11.006

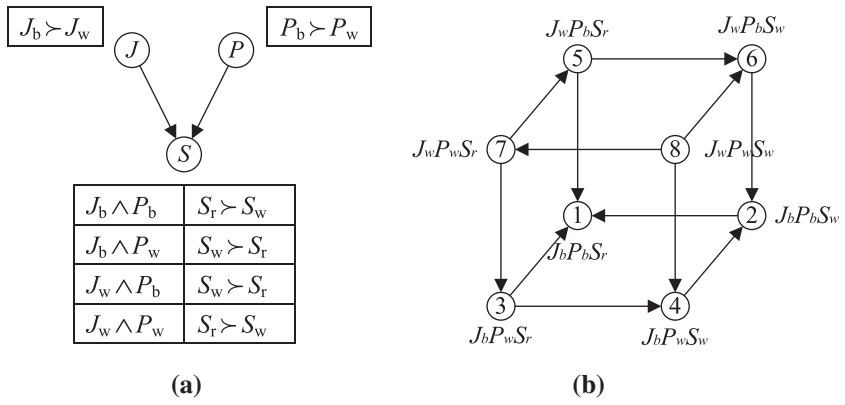

Fig. 1. (a) The CP-net for evening dress scenario and (b) the induced preference graph.

$$diff(G_1, G_2) = |\{(o, o') | (o, o') \in E_1, (o, o') \notin E_2\}|/k \quad (4)$$

where E_1 and E_2 are the sets of edges in G_1 and G_2. k is the possible maximal number of edges in G_1 and G_2, $k = m|\Omega|/2, m = \sum_{i=1}^{n}(|Dom(X_i)| - 1)$.

In general cases, $diff(G_1, G_2) \neq diff(G_2, G_1)$. But if G_1 and G_2 are complete, $diff(G_1, G_2) = diff(G_2, G_1)$, here G_1 and G_2 are two preference graphs over same outcome space.

Definition 4. Given two CP-nets $N_1$and N_2 over same variables, their preference graphs are G_1 and G_2, the difference between N_1 and N_2 is defined as the difference between G_1 and G_2,

$$diff(N_1, N_2) = diff(G_1, G_2) \quad (5)$$

In order to compute the difference between two CP-nets, the CP-nets should be transformed into preference graphs firstly. And the different edges in preference graphs are then counted. Edges in preference graph can be obtained by processing every conditional preference rule one time [39]. So the complexity of computing the difference between two CP-nets is polynomial to the size of CP-nets.

4. Learning CP-nets using hypothesis testing

According to the definition of CP-nets, the task of learning CP-nets includes two aspects: (1) learning conditional preference dependency between variables and (2) learning the conditional preference tables for every variable. In this section, we introduce the method of learning conditional preference dependency and conditional preference table using hypothesis testing.

4.1. Learning conditional preference dependency

We use hypothesis testing to determine whether a variable X is preferentially dependent on a given set U of variables. The hypotheses are:

$$H_0: \quad Pa(X) \neq U$$
$$H_1: \quad Pa(X) = U$$

To test the above hypotheses, we compare two distributions using chi-squared testing. Let us consider the following conditional probability:

$$P_{u:x\succ x'} = P(o[U] = u | o \succ o', o[X] = x, o'[X] = x') \quad (6)$$

where $o, o' \in \Omega$, $u \in Dom(U)$. Because the variables are independent, $P(o[U] = u | o \succ o', \ o[X] = x, \ o'[X] = x') = P(o'[U] = u | o \succ o', \ o[X] = x, o'[X] = x')$, we only consider $P(o[U] = u | o \succ o', o[X] = x, o'[X] = x')$ in formula (6).

If H_0 is true, because the variables are independent with each other, the event "$o[U] = u$" and the event "$o \succ o' \wedge o[X] = x \wedge o'[X] = x'$" are independent. We have $P_{u:x\succ x'} = P(o[U] = u) = P_u$.

If H_1 is true, there must exist a conditional preference rule, such as u: $x \succ x'$, in the conditional preference table of variable X. So the event "$o[U] = u$" and the event "$o \succ o' \wedge o[X] = x \wedge o'[X] = x'$" are dependent. $P_{u:x\succ x'} \neq P_u$. Note that the $P_{u:x\succ x'} \neq P_u$ still holds even when the training samples are noisy.

So H_0 and H_1 are equivalent to the following hypotheses:

$$H_0': \quad P_{u:x\succ x'} = P_u$$
$$H_1': \quad P_{u:x\succ x'} \neq P_u \quad x, x' \in Dom(X)$$

To test the hypotheses above, we need to compare two distributions. Chi-squared goodness-of-fit testing, which is one of the most popular hypotheses testing methods, can decide whether or not an observed frequency distribution differs from a theoretical distribution. Compared with other hypotheses testing methods, chi-squared testing has fewer assumptions. It requires the observations are independent, which is consistent to our assumption. Furthermore, chi-squared testing is distribution-free. We need not know the parameters and form of the compared distributions when we use chi-squared testing. So we choose chi-squared testing to test the hypotheses.

The observed frequencies are:

$$n_{u:x\succ x'} = |\{o \succ o' | o[X] = x, \ o'[X] = x', \ o[U] = u\}| \quad (7)$$

The theoretical frequencies to be compared are $E_{u:x\succ x'} = n_{x\succ x'}P_u$, here $n_{x\succ x'} = \sum_{u\in Dom(U)} n_{u:x\succ x'}$. Probability $P_u = P(o[U] = u)$ can be estimated from the training samples. The value of the test-statistic is:

$$\chi^2 = \sum_{u\in Dom(U)} \frac{(n_{u:x\succ x'} - E_{u:x\succ x'})^2}{E_{u:x\succ x'}} \quad (8)$$

The degrees of freedom are $d = |Dom(U)| - 1$.

4.2. Learning conditional preference rule

The next step is to learn the conditional preference rules for every variable. For a variable X, let $Pa(X) = U$. We should determine whether u: $x \succ x'$ holds, where $u \in Dom(U), x, x' \in Dom(X)$. Consider the conditional probability:

$$P_{x\succ x'|u} = P(o \succ o' | o[X] = x, \ o'[X] = x', \ o[U] = o'[U] = u) \quad (9)$$

If u: $x \succ x'$ holds, at noise level p, $P_{x'\succ x|u} = p$. We compare the sizes of two sets, $n_{x\succ x'|u}$ and $n_{x'\succ x|u}$, here,

$$n_{x\succ x'|u} = |\{o \succ o' | o[X] = x, o'[X] = x', o[U] = o'[U] = u\}| \quad (10)$$

Please cite this article in press as: J. Liu et al., Learning conditional preference network from noisy samples using hypothesis testing, Knowl. Based Syst. (2012), http://dx.doi.org/10.1016/j.knosys.2012.11.006

Suppose the noise level $p < 0.5$, if $n_{x\succ x'|u} > n_{x'\succ x|u}$, set $\{o \succ o' | o[X] = x', o'[X] = x, o[U] = o'[U] = u\}$ of samples is noise and conditional preference rule u: $x \succ x'$ should be added into conditional preference table of variable X, $CPT(X)$, otherwise u: $x' \succ x$ should be added into $CPT(X)$.

4.3. Algorithm

To learn a CP-net from noisy training samples, we first determine the parent variables for each variable using chi-squared testing, and then generate the conditional preference tables according to the preferential dependence. Finally, conditional preference tables are reduced by removing the redundant parent variables and merging conditional preference rules to make the CP-net simple. The pseudo code of the proposed algorithm is summarized in Algorithm 1.

Algorithm 1. Learning CP-net using hypothesis testing

```
Input: set V of variables and set S of training sample;
Output: CP-net N;
01  For each variable X ∈ V
02    For each pair x, x' ∈ Dom(X)
03      If(ParentTest(X, V/X, x, x'))
04        Pa(X) = V/X;
05        For each u ∈ Dom(Pa(X))
06          GenerateCPR(X, x, x', u);
07        End
08      Else
09        GenerateCPR(X, x, x', ε);
10      End
11    End
12    If Pa(X) ≠ ∅
13      ReduceCPT(CPT(X));
14    End
15  End
```

In Algorithm 1, function $ParentTest(X, U, x, x')$ tests whether the variables in U are the parent variables of variable X with respect to the values $x, x' \in Dom(X)$ using chi-squared testing. If U is the parent variable set of variable X with respect to the values x and x', it returns true, otherwise it returns false. Function $GenerateCPR(X, x, x', u)$ generates a conditional preference rule by comparing $n_{x\succ x'|u}$ and $n_{x'\succ x|u}$, and adds this rule into the conditional preference table of variable X. If $ParentTest(X, U, x, x')$ returns false, preference relation over x and x' is independent. Algorithm 1 calls $GenerateCPR(X, x, x', \varepsilon)$ comparing $n_{x\succ x'}$ with $n_{x'\succ x}$, where ε is empty value, just as line 9.

Because Algorithm 1 tests the preferential dependence between a variable in V and the rest of V, there may be exist redundant variables in the set of parent variables. Function $ReduceCPT(CPT)$ reduces the conditional preference table CPT by removing redundant parent variables and merging corresponding conditional preference rules to make the CP-net simple. If the change of a parent variable does not effect the preference relation between values of child variable in the conditional preference table, this parent variable is redundant. Formally, given a variable $Y \in Pa(X)$, for any two conditional preference rules u_1: e_1 and u_2: e_2, if $u_1[Y] \neq u_2[Y]$, $u_1[Pa(X)/Y] = u_2[Pa(X)/Y]$ and $e_1 = e_2$ hold, variable Y is a redundant variable in $Pa(X)$. After removing the redundant parent variables, the corresponding conditional preference rules should be merged. For example, let $V = \{A, B, C\}$ denote a set of variables. The domains of variables are $Dom(A) = \{a_0, a_1\}$, $Dom(B) = \{b_0, b_1\}$ and $Dom(C) = \{c_0, c_1\}$. Conditional preference table of variable A is shown in Fig. 2a. By observing the conditional preference rule (1) and (2) in Fig. 2a, we can find the change of variable C does not change the preference relation between a_0 and a_1. We can draw the same conclusion from rule (3) and rule (4). So variable C should be removed from $Pa(A)$. The conditional preference rule (1) and (2) should be merged. The rule (3) and rule (4) should also be merged. The reduced conditional preference table is shown in Fig. 2b.

(a)

(1)	$b_0c_0 : a_0 \succ a_1$
(2)	$b_0c_1 : a_0 \succ a_1$
(3)	$b_1c_0 : a_1 \succ a_0$
(4)	$b_1c_1 : a_1 \succ a_0$

(b)

(1-2)	$b_0 : a_0 \succ a_1$
(3-4)	$b_1 : a_1 \succ a_0$

Fig. 2. (a) Initial conditional preference table and (b) the reduced conditional preference table.

In Algorithm 1, chi-squared testing (line 03) is performed $n_t = \sum_{i=1}^{|V|} |Dom(X_i)|(|Dom(X_i)| - 1)/2$ times. To ensure that the significance level for the whole family of tests should be (at most) α, we apply Bonferroni adjustment, i.e. testing each of the individual test at a significance level of α/n_t.

In this paper, we only consider acyclic CP-nets. If the training samples are generated from an acyclic CP-net, CP-net obtained by Algorithm 1 approaches this acyclic CP-net, which is guaranteed by Theorem 1 in Section 4.6. If Algorithm 1 obtains a cyclic CP-net, more training samples are needed to obtain an acyclic CP-net.

4.4. An example

Here we give an example to demonstrate the solution process for Algorithm 1. In this example, there are two binary variables, and the outcome space is small. This example also demonstrates that our method is able to adapted to the CP-nets over very small outcome space.

Example 2. For simplicity, consider the dinner scenario in [39]. The set $V = \{S, W\}$ of variables stands for soup and wine. The domains $Dom(S) = \{S_f, S_v\}$ stands for fish soup and vegetable soup; $Dom(W) = \{W_r, W_w\}$ stands for red wine and white wine. We have collected the training samples presented in Table 1 by observing the user for a period of time, and some pairwise comparisons in training samples are observed several times. In Table 1 we list the pairwise comparisons and the observing count. From Table 1, distributions of variables are estimated: $P_{S_f} = 0.4935$, $P_{S_v} = 0.5065$, $P_{W_w} = 0.5000$, $P_{W_r} = 0.5000$.

Firstly, Algorithm 1 decides whether variable S is preferential dependent on variable W. The observed frequencies are $n_{W_w:S_f\succ S_v} = 11$ and $n_{W_r:S_f\succ S_v} = 10$. $n_{S_f\succ S_v} = n_{W_w:S_f\succ S_v} + n_{W_r:S_f\succ S_v} =$

Table 1
Training samples in Example 2.

Pair-wise comparison	Observing count
$S_fW_w \succ S_vW_w$	11
$S_fW_r \succ S_vW_r$	10
$S_fW_w \succ S_fW_r$	20
$S_vW_r \succ S_vW_w$	22
$S_vW_r \succ S_fW_r$	2
$S_vW_w \succ S_fW_w$	1
$S_fW_r \succ S_fW_w$	6
$S_vW_w \succ S_vW_r$	5

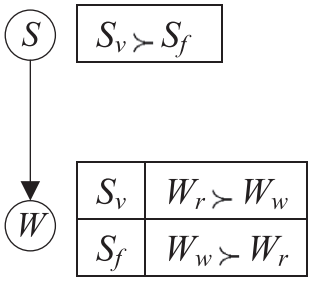

Fig. 3. The obtained CP-net in Example 2.

21. The compared theoretical frequencies are $E_{W_w:S_f\succ S_v} = n_{S_f\succ S_v} * P_{W_w} = 21 * 0.5 = 10.5$ and $E_{W_r:S_f\succ S_v} = n_{S_f\succ S_v} * P_{W_r} = 21 * 0.5 = 10.5$. The value of the test-statistic is $\chi^2 = (10-10.5)^2/10.5 + (11-10.5)^2/10.5 = 0.0476$. This value is out of the reject region, so hypothesis H_0 and H_0' are accepted. Variable W is not the parent variable of variable S, $Pa(S) = \emptyset$. Next, Algorithm 1 constructs the conditional preference table of variable S. Because $Pa(S)$ is empty, $n_{S_f\succ S_v}$ and $n_{S_v\succ S_f}$ are compared. Because $n_{S_f\succ S_v} = 21 > n_{S_v\succ S_f} = 3$, $S_v \succ S_f$ is added into the conditional preference table of variable S.

Secondly, Algorithm 1 decides whether variable S is the parent of variable W. The observed frequencies are $n_{S_f:W_w\succ W_r} = 20$ and $n_{S_v:W_w\succ W_r} = 5$. We have $n_{W_w\succ W_r} = n_{S_f:W_w\succ W_r} + n_{S_v:W_w\succ W_r} = 25$. The compared theoretical frequencies are $E_{S_f:W_w\succ W_r} = n_{W_w\succ W_r} * P_{S_f} = 25 * 0.4935 = 12.3375$ and $E_{S_v:W_w\succ W_r} = n_{W_w\succ W_r} * P_{S_v} = 25 * 0.5065 = 12.6625$. The value of the test-statistic is $\chi^2 = (20-12.3375)^2/12.3375 + (5-12.6625)^2/12.6625 = 9.3958$. This value is in the reject region, so hypothesis H_1 and H_1' are accepted. Variable S is the parent variable of variable W, $Pa(W) = S$. Next, Algorithm 1 constructs the conditional preference table of variable W. Because $n_{S_f:W_w\succ W_r} = 20 > n_{S_f:W_r\succ W_w} = 6$, S_f: $W_w \succ W_r$ is added into the conditional preference table of variable W. Because $n_{S_v:W_r\succ W_w} = 22 > n_{S_v:W_w\succ W_r} = 5$, S_v: $W_r \succ W_w$ is added.

The conditional preference tables of variable S and W are the simplest, and need not to be reduced. The obtained CP-net is shown in Fig. 3.

4.5. Computational complexity analysis

Line 06 in Algorithm 1 executes $\frac{1}{2}\sum_{i=1}^{|V|}|Dom(X_i)|(|Dom(X_i)|-1)\prod_{j\neq i}|Dom(X_j)|$ times at most, which is equal to the maximal size of general CP-nets $|N|_{max}$ (Eq. (3)). Note all of the test-statistics can be computed by processing every training sample one time. So Algorithm runs in time $poly(|N|_{max}, |S|)$.

4.6. Properties of the algorithm

We assume the training samples are randomly drew from an unknown distribution. Referring to the definition of Random Instance Oracle [14], we define Random Instance Oracle with Noise.

Definition 5. Given a set V of variables, a probability of independent distribution D over unordered outcome pairs over V where $\forall o \in \Omega$, $P(o) \neq 0$, a CP-net N over V and noise level $p \in [0, 0.5)$, the **Random Instance Oracle with Noise** $RN(D, N, p)$ is a procedure running in unit time, and on each call $RN(D, N, p)$ returns comparison $o \succ o'$ with probability $1-p$ and returns comparison $o' \succ o$ with probability p, where pair o, o' of outcome draws randomly and independently from D, and $o \succ_N o'$, i.e.,

$$P(o \succ o' \leftarrow RN(D,N,p) | o \succ_N o') = 1-p \tag{11}$$

$$P(o' \succ o \leftarrow RN(D,N,p) | o \succ_N o') = p \tag{12}$$

Next, we give some useful properties of Algorithm 1 by the following theorems.

Theorem 1. *Let N_0 be the initial CP-net. Set S of training samples is drew from $RN(D, N_0, p)$. N is the CP-net returned by Algorithm 1, N converges in mean to N_0, i.e., $\lim_{|S|\to\infty} E(diff(N_0, N)) = 0$.*

Proof. Let G_0 and G denote the preference graphs induced from CP-net N_0 and N, respectively. Let E_0 and E denote the sets of edges in G_0 and G. According to Definitions 2–4, we have:

$$E(diff(N_0, N)) = \frac{1}{k}\sum_{(o,o')\in E_0} P((o,o') \notin E | (o,o') \in E_0) \tag{13}$$

For a given edge $(o, o') \in E_0$, let outcomes o, o' differ on variable X. This edge can be regarded as a conditional preference rule: $o[V/X]$: $o'[X] \succ o[X]$. So the probability

$$\begin{aligned} &P((o,o') \notin E | (o,o') \in E_0) \\ &\quad = P(\text{accept } H_0' | H_1' \text{ is true}) + P(\text{accept } H_1', o[X] \\ &\quad \succ o'[X] | H_1' \text{ is true}) \end{aligned} \tag{14}$$

$P(\text{accept} H_0' | H_1' \text{ is true}) = P(\chi^2 < \chi_\alpha^2 | H_1') = \beta$ is the probability of type II error of the hypothesis testing. Here, χ^2 is the test-statistic, χ_α^2 is the α-significance point of the chi-squared distribution with d degrees of freedom. When H_1' is true, the test-statistic χ^2 follows non-central chi-squared distribution with d degrees of freedom [40],

$$\beta = F_{\chi'^2}(\chi_\alpha^2; d, \lambda) \tag{15}$$

where $F_{\chi'^2}(x; d, \lambda)$ is the cumulative distribution function of non-central chi-squared distribution with d degrees of freedom. λ is the non-central parameter, $\lambda = n\Delta(H_0', H_1')$, here n is the number of samples for the hypothesis testing. $\Delta(H_0', H_1')$ is the measure of discrepancy between two distributions specified by H_0' and H_1', which is a constant for the given H_0' and H_1'. We denote λ as $\lambda = n\Delta$.

$$\begin{aligned} \beta &= F_{\chi'^2}(\chi_\alpha^2; d, n\Delta) \\ &= \exp(-n\Delta/2)\sum_{j=0}^{\infty} F_{\chi^2}(\chi_\alpha^2; d+2j)(n\Delta/2)^j/j! \leqslant \exp(-n\Delta/2) \end{aligned} \tag{16}$$

where $F_{\chi^2}(x; d)$ is the cumulative distribution function of central chi-squared distribution with d degrees of freedom.

$$\begin{aligned} &P(\text{accept } H_1', o[X] \succ o'[X] | H_1' \text{ is true}) \\ &\quad = P(\text{accept } H_1' | H_1' \text{ is true}) \cdot P(\text{accept } o[X] \\ &\quad \succ o'[X] | H_1' \text{ is true}) \\ &\quad = (1-\beta)F_B(n/2; n, 1-p) \end{aligned} \tag{17}$$

where $F_B(k; n, p)$ is the cumulative distribution function of binomial distribution. According to Hoeffding's inequality, when $p < 0.5$, $F(n/2, n, 1-p) \leqslant \exp(-2n(1/2-p)^2)/2$. So,

$$\begin{aligned} &P((o,o') \notin E | (o,o') \in E_0) \\ &\quad \leqslant 1 - (1 - \exp(-n\Delta/2))(1 - \exp(-2n(1/2 \\ &\quad\quad - p)^2)/2) \\ &\quad \leqslant \exp(-n\Delta/2) + \exp(-2n(1/2-p)^2)/2 \\ &\quad \leqslant 3\exp(-C_{o,o'}n)/2 \end{aligned} \tag{18}$$

where $C_{o,o'} = min\{\Delta/2, 2(1/2-p)^2\} > 0$. When $n \to \infty$, $P((o,o') \notin E | (o,o') \in E_0) \leqslant 3\exp(-C_{o,o'}n)/2 \to 0$. Since training samples are drew from $RN(D, N_0, p)$, $\forall o \in \Omega$, $P(o) \neq 0$, when $|S| \to \infty$, $n \to \infty$. So,

Please cite this article in press as: J. Liu et al., Learning conditional preference network from noisy samples using hypothesis testing, Knowl. Based Syst. (2012), http://dx.doi.org/10.1016/j.knosys.2012.11.006

$$\lim_{|S|\to\infty} E(diff(N_0,N)) = \sum_{(o,o')\in E_0} \lim_{n\to\infty} P((o,o') \notin E|(o,o') \in E_0)/k = 0 \quad \square \tag{19}$$

Theorem 1 guarantees the accuracy of Algorithm 1. Given sufficient training samples, the proposed algorithm can return a CP-net the same as the user's real preference represented by the initial CP-net. In practice, to achieve a specified accuracy, only finite number of training samples are needed (for more details see the experimental results). However, Algorithm 1 is not suitable to deal with very small set of training samples (for example, the number of training samples is much less than $|\Omega|$). The reason is that it is required the value for every observed frequency is not less than 5 for chi-squared testing.

Because the obtained CP-net N converges to the initial CP-net N_0, N entails pairwise comparisons drew from $R(D, N_0, p)$ with probability $1-p$. This conclusion is summarized in the following theorem.

Theorem 2. *Let N_0 be the initial CP-net. Set S of training sample is drew from RN(D, N_0, p). N is the CP-net returned by Algorithm 1. For any comparison $o \succ o'$ drew from R(D, N_0, p), $\lim_{|S|\to\infty} P(o \succ_N o') = 1-p$.*

Proof. According to total probability formula, we have:

$$\begin{aligned} P(o\succ_N o') &= P(o \succ o' \text{ is noise})P(o\succ_N o'|o \succ o' \text{ is noise}) + P(o \\ &\succ o' \text{ is not noise})P(o\succ_N o'|o \succ o' \text{ is not noise}) \\ &= pP(o\succ_N o'|o'\succ_{N_0} o) + (1-p)P(o\succ_N o'|o\succ_{N_0} o') \end{aligned} \tag{20}$$

According to the proof of Theorem 1, when $|S| \to \infty, P(o\succ_N o'|o'\succ_{N_0} o) \to 0$. So $\lim_{|S|\to\infty} P(o \succ_N o') = 1-p$. $\square$

5. Experimental results

The performance of proposed algorithm is verified by simulated data and SUSHI data set [41]. To the best of our knowledge, there does not exist study in learning CP-nets from inconsistent training samples. So we choose the most related method which is proposed in [14] for comparison. Both the method in [14] and our method learn CP-nets passively, but there are more constraints on training samples in [14]'s method. Although we mainly focus on learning CP-nets passively, considering the methods for learning to rank are partly related to our work, we also compare our method with popular learning to rank methods, such as RankNet [17], AdaRank [18] and Coordinate Ascent [19], which can handle noisy training samples.

5.1. Experimental results on simulated data

We generate an acyclic CP-net with five binary variables randomly to represent a user's preference. Pairs of outcomes are drew randomly according to uniform distribution. Preference order between two outcomes are determined by the generated CP-net. Considering that the user may not make every decision exactly consistent with her/his preference, we add some noise on the training samples. Namely, we change the preference order between two outcomes at a probability of p, which is called noise level. We generate 100, 200, 500, 1000, 2000 and 5000 pairs of comparisons as the training samples at noise level 0, 0.01, 0.05, 0.1 and 0.2, respectively. A CP-net is learned from training samples using Algorithm 1. For each configuration, the experiments are repeated 1000 times. We calculate the average similarity between initial CP-net N_0 and learned CP-net N, $1-diff(N_0, N)$. The average sample agreement, which is defined as the proportion of samples entailed by learned CP-net N in training samples, is calculated too. The results at different noise level on different sample size are show in Fig. 4.

From Fig. 4a we can see at the same noise level, as the sample size increases, the average CP-net similarity increases and converges at 1. That means the rebuilt CP-nets will approach the initial CP-nets even at high noise level, just as Theorem 1 guarantees. The converging speed is low at high noise level, so more samples are needed to get more accurate result. Although a large number of training samples are needed to rebuild the initial CP-net accurately, only a small number of training samples are needed to get satisfied results. For example, at noise level of 0.05, using 200 comparisons, the learner get CP-net similarity of 0.7620, while using 500 comparisons, the learner get CP-net similarity of 0.9186. Considering there are $2^5 * (2^5 - 1) = 992$ possible pairwise of comparisons, these sizes of training samples are small. While in Fig. 4b we can see at noise level p the average sample agreement increases and converges at $1-p$ as sample size increases, which is proved by Theorem 2.

5.2. Experimental results on SUSHI data set

SUSHI data set [41] contains 5000 preferences orders of users on 100 kinds of sushi. Each kind of sushi is represented by several features. To simplify the problem, we select three features: the major group, the heaviness and normalized price. These features are considered to be the most important features affecting preferences based on common sense. The "major group" feature is binary, while the other two features are continuous. Because CP-net cannot process continuous values, we choose to use the average values of the features to discretize them. All the data lower than the

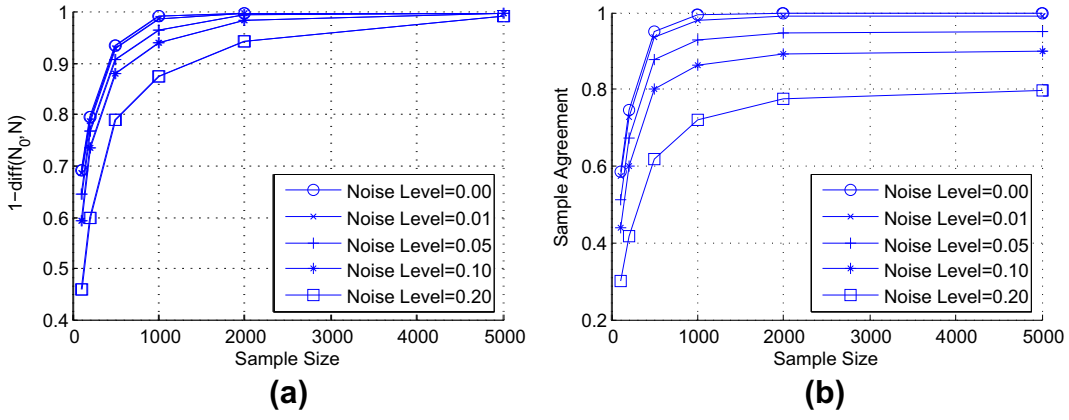

Fig. 4. The average CP-net similarity (a) and the average sample agreement (b) at different noise level on simulated data.

Please cite this article in press as: J. Liu et al., Learning conditional preference network from noisy samples using hypothesis testing, Knowl. Based Syst. (2012), http://dx.doi.org/10.1016/j.knosys.2012.11.006

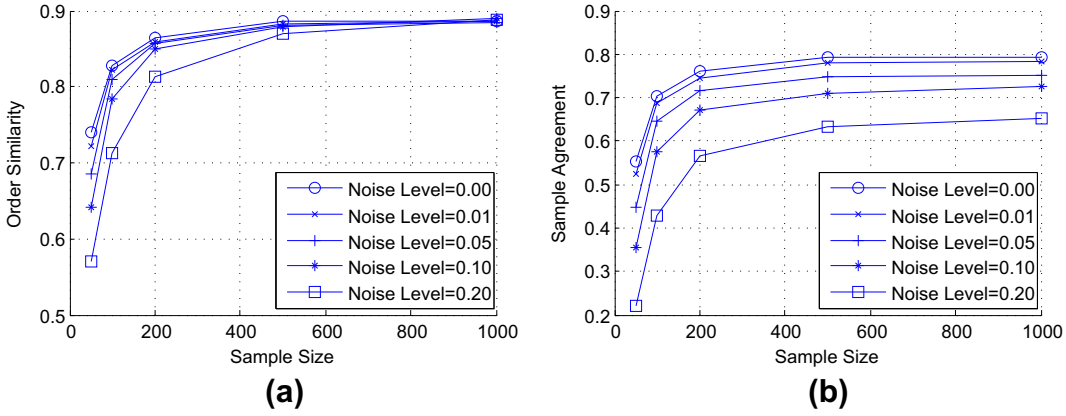

Fig. 5. The average order similarity (a) and the average sample agreement (b) at different noise level on SUSHI data set.

Table 2
Comparison the proposed method with Dimopoulos's method on simulated data.

Sample size	Dimopoulos's method			Proposed method		
	Success rate	CP-net similarity	Sample agreement	Success rate	CP-net similarity	Sample agreement
100	0.6100	**0.8608**	**0.8072**	**1.0000**	0.6412	0.5255
200	0.3790	0.8080	0.7125	**1.0000**	**0.8156**	**0.7661**
500	0.3260	0.7972	0.6928	**1.0000**	**0.9443**	**0.9311**
1000	0.3300	0.7964	0.6905	**1.0000**	**0.9850**	**0.9846**

average value is set to 0, all the data higher than average value is set to 1.

After preprocessing, all the sushi are categorized into $2^3 = 8$ types. Knowing the preference order on these types, we select sushi randomly according to uniform distribution and generate pairwise comparisons of sushi with noise as training samples. For each user's preference order, we generate 50, 100, 200, 500 and 1000 pair of comparisons of sushi at noise level 0, 0.01, 0.05, 0.1 and 0.2, respectively. For each configuration, we choose 1000 preference orders of users randomly to repeat the experiment. The similarity between the user's preference order and the rebuilt CP-net is defined as:

$$\text{similarity} = \frac{|\{o, o' | o \succ_N o', o \succ_{PO} o'\}|}{|\{o, o' | o \succ_{PO} o'\}|} \tag{21}$$

where $o, o' \in \Omega$, N denotes the rebuilt CP-net, PO denotes the user's preference order on Ω. The numerator is the number of pairs of outcomes entailed by both CP-net N and preference order PO. The denominator is the number of pairs of outcomes entailed only by preference order PO. We also compute the average sample agreement. The results are shown in Fig. 5. Comparing with the first experiment, we can find out that the trends of the curves are similar. So all the conclusions drew from the first experiment still make sense. Besides the similarity, there are also some differences between the two experiments. CP-net can only represent partial order, while the users' preferences in SUSHI data set are total orders. So there must be some preferences that cannot be satisfied by CP-nets. For this reason, although we use 1000 noise-free samples, the order similarity and sample agreement do not approach 1. And the curves are lower than those in the first experiment.

5.3. Comparison with Dimopoulos's method

We compare our method with Dimopoulos's method [14] on simulated data and SUSHI data set. Because Dimopoulos's method cannot handle noisy samples, we fix noise level at 0. We generate 100, 200, 500 and 1000 pairwise comparisons without noise and rebuild CP-nets using Dimopoulos's method and ours. For each sample size, the experiments are repeated 1000 times. The results are shown in Tables 2 and 3. The bold values in Table 2– Table 7 are the best results in the experiments. In many cases, the training samples are not transparently entailed by a CP-net (see Definition 5 in [14]), Dimopoulos's method fails to return a CP-net, and it is more likely to fail by given more training samples, while our method can always obtain a CP-net. For this reason, as the sample size

Table 3
Comparison the proposed method with Dimopoulos's method on SUSHI data set.

Sample size	Dimopoulos's method			Proposed method		
	Success rate	Order similarity	Sample agreement	Success rate	Order similarity	Sample agreement
100	0.0490	0.5405	0.1557	**1.0000**	**0.8285**	**0.7027**
200	0.0580	0.5462	0.1674	**1.0000**	**0.8637**	**0.7598**
500	0.0400	0.5376	0.1545	**1.0000**	**0.8854**	**0.7926**
1000	0.0540	0.5428	0.1640	**1.0000**	**0.8889**	**0.7980**

Table 4
Comparison the proposed method with learning to rank methods on simulated data (sample size = 500).

Noise level (p)	0		0.01		0.05		0.10	
Method	Similarity	Agreement	Similarity	Agreement	Similarity	Agreement	Similarity	Agreement
RankNet	0.8540	0.8325	0.8525	0.8224	0.8533	0.8042	0.8469	0.7670
AdaRank	0.8233	0.7083	0.8291	0.7087	0.8249	0.7027	0.8283	0.7061
Coordiante Ascent	0.8774	0.8853	0.8721	0.8722	0.8757	0.8553	0.8736	**0.8245**
Our method	**0.9345**	**0.9501**	**0.9295**	**0.9349**	**0.9071**	**0.8763**	**0.8797**	0.8002

Please cite this article in press as: J. Liu et al., Learning conditional preference network from noisy samples using hypothesis testing, Knowl. Based Syst. (2012), http://dx.doi.org/10.1016/j.knosys.2012.11.006

Table 5
Comparison the proposed method with learning to rank methods on simulated data (sample size = 1000).

Noise level (p)	0		0.01		0.05		0.10	
Method	Similarity	Agreement	Similarity	Agreement	Similarity	Agreement	Similarity	Agreement
RankNet	0.8700	0.8602	0.8643	0.8474	0.8660	0.8242	0.8626	0.7928
AdaRank	0.8244	0.7098	0.8248	0.6989	0.8238	0.7039	0.8294	0.7083
Coordiante Ascent	0.8762	0.8805	0.8710	0.8698	0.8739	0.8481	0.8708	0.8187
Our method	**0.9923**	**0.9952**	**0.9869**	**0.9817**	**0.9657**	**0.928**	**0.9412**	**0.8629**

Table 6
Comparison the proposed method with learning to rank methods on sushi dataset (sample size = 200).

Noise level (p)	0		0.01		0.05		0.10	
Method	Similarity	Agreement	Similarity	Agreement	Similarity	Agreement	Similarity	Agreement
RankNet	0.6578	0.6616	0.6611	0.6619	0.6489	0.6378	0.6481	0.6232
AdaRank	0.6781	0.6826	0.6754	0.6760	0.6836	0.6788	0.6869	0.6723
Coordiante Ascent	0.8506	**0.8589**	0.8439	**0.8488**	0.8465	**0.8233**	0.8408	**0.7877**
Our method	**0.8637**	0.7598	**0.8588**	0.7463	**0.8571**	0.7152	**0.8493**	0.6719

Table 7
Comparison the proposed method with learning to rank methods on sushi dataset (sample size = 500).

Noise level (p)	0		0.01		0.05		0.10	
Method	Similarity	Agreement	Similarity	Agreement	Similarity	Agreement	Similarity	Agreement
RankNet	0.7612	0.7635	0.7606	0.7575	0.7491	0.7255	0.7460	0.6996
AdaRank	0.6788	0.6814	0.6858	0.6860	0.6864	0.6797	0.6859	0.6721
Coordiante Ascent	0.8529	**0.8575**	0.8434	**0.8419**	0.8448	**0.8176**	0.8462	**0.7869**
Our method	**0.8854**	0.7926	**0.8828**	0.7797	**0.8815**	0.7492	**0.8783**	0.7091

increases, the CP-net similarity, order similarity and sample agreement provided by Dimopoulos's method decrease on both simulated data and SUSHI data set, which is opposite to that of ours. Dimopoulos's method only get higer CP-net similarity and sample agreement from 100 training samples on simulated data. Our method gets more accurate results in other cases (see Tables 2 and 3).

5.4. Comparison with learning to rank methods

We compare our method with RankNet [17], AdaRank [18] and Coordinate Ascent [19] on simulated data. These methods are the most popular methods for learning to rank in recent years, and can handle noisy training samples. The results, shown in Tables 4 and 5, indicate that our method outperforms these methods. The reason is that the methods for learning to rank, such as RankNet [17], AdaRank [18] and Coordinate Ascent [19], cannot represent conditional preference perfectly.

We compare our method with RankNet [17], AdaRank [18] and Coordinate Ascent [19] on sushi dataset. The results are shown in Tables 6 and 7. Although Coordinate Ascent [19] gets the highest sample agreement, our method gets the highest order similarity in all of the cases.

6. Conclusion

Within artificial intelligence, the problem of preference learning or preference elicitation is valuable in theory and application. In this paper, we propose a model of learning CP-nets from noisy samples and present a method to solve this model. We learn the preferential dependence between variables by hypothesis testing, and then we remove redundant parent variables and merge corresponding conditional preference rules to make the obtained CP-nets simple. We prove the obtained CP-net converges in mean to initial CP-net as the size of training samples increases.

In further research, we plan to study the learning methods for other CP-nets classes, including TCP-nets [8] and Stronger Conditional Preference Statements [11], which are more expressive than CP-nets. Online learning is another research direction, where the learner receives the samples one by one, and each time the learner updates the user's preference.

References

[1] L. Majid Zerafat Angiz, Ali Emrouznejad, A. Mustafa, A.S. Al-Eraqi, Aggregating preference ranking with fuzzy data envelopment analysis, Knowledge-Based Systems 23 (6) (2010) 512–519.
[2] H. Chen, L. Zhou, B. Han, On compatibility of uncertain additive linguistic preference relations and its application in the group decision making, Knowledge-Based Systems 24 (6) (2011) 816–823.
[3] I. Yakut, H. Polat, Estimating nbc-based recommendations on arbitrarily partitioned data with privacy, Knowledge-Based Systems (2012).
[4] V. Moscato, A. Picariello, A.M. Rinaldi, Towards a user based recommendation strategy for digital ecosystems, Knowledge-Based Systems (2012).
[5] D. Mindolin, J. Chomicki, Contracting preference relations for database applications, Artificial Intelligence 175 (7C8) (2011) 1092–1121.
[6] C. Boutilier, R.I. Brafman, H.H. Hoos, D. Poole, Reasoning with conditional ceteris paribus preference statements, in: Proceedings of UAI'99, 1999, pp. 71–80.
[7] C. Boutilier, F. Bacchus, R.I. Brafman, Ucp-networks: a directed graphical representation of conditional utilities, in: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, UAI'01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 56–64.
[8] R.I. Brafman, C. Domshlak, S.E. Shimony, On graphical modeling of preference and importance, Journal of Artificial Intelligence Research 25 (2006) 398–424.
[9] P. Chatel, I. Truck, J. Malenfant, LCP-Nets: a linguistic approach for non-functional preferences in a semantic soa environment, Journal of Universal Computer Science 16 (1) (2010) 198–217.
[10] S. Kaci, H. Prade, Relaxing ceteris paribus preferences with partially ordered priorities, in: Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 660–671.
[11] N. Wilson, Extending CP-Nets with stronger conditional preference statements, in: Proceedings of AAAI'04, 2004, pp. 735–741.
[12] F. Yaman, M. desJardins, More-or-less CP-Networks, in: Proceedings of UAI'07, 2007, pp. 434–441.
[13] F. Koriche, B. Zanuttini, Learning conditional preference networks, Artificial Intelligence 174 (2010) 685–703.

Please cite this article in press as: J. Liu et al., Learning conditional preference network from noisy samples using hypothesis testing, Knowl. Based Syst. (2012), http://dx.doi.org/10.1016/j.knosys.2012.11.006

[14] Y. Dimopoulos, L. Michael, F. Athienitou, Ceteris paribus preference elicitation with predictive guarantees, in: Proceedings of IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, pp. 1890–1895.
[15] J. Lang, J. Mengin, Learning preference relations over combinatorial domains, in: Proceedings of NMR08, 2008, pp. 207–214.
[16] J. Lang, J. Mengin, The complexity of learning separable ceteris paribus preferences, in: Proceedings of IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, pp. 848–853.
[17] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: Proceedings of the 22nd International Conference on Machine Learning, ICML '05, ACM, New York, NY, USA, 2005, pp. 89–96.
[18] J. Xu, H. Li, Adarank: a boosting algorithm for information retrieval, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, ACM, New York, NY, USA, 2007, pp. 391–398.
[19] D. Metzler, W. Bruce Croft, Linear feature-based models for information retrieval, Information Retrieval 10 (3) (2007) 257–274.
[20] W.W. Cohen, R.E. Schapire, Y. Singer, Learning to order things, Journal of Artificial Intelligence Research 10 (1) (1999) 243–270.
[21] T. Kamishima, H. Kazawa, S. Akaho, Supervised ordering: an empirical survey, in: Proceedings of the Fifth IEEE International Conference on Data Mining, ICDM '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 673–676.
[22] Y. Freund, R. Iyer, R.E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, Journal of Machine Learning Research 4 (2003) 933–969.
[23] H. Kazawa, T. Hirao, E. Maeda, Order svm: a kernel method for order learning based on generalized order statistics, Systems and Computers in Japan 36 (1) (2005) 35–43.
[24] R. Herbrich, T. Graepel, P. Bollmann-sdorra, K. Obermayer, Learning preference relations for information retrieval, in: ICML1998 Workshop: Text Categorization and Machine Learning, ICML 1998, 1998, pp. 83–86.
[25] T. Joachims, Optimizing search engines using clickthrough data, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, ACM, New York, NY, USA, 2002, pp. 133–142.
[26] T. Qin, X.-D. Zhang, D.-S. Wang, T.-Y. Liu, W. Lai, H. Li, Ranking with multiple hyperplanes, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, ACM, New York, NY, USA, 2007, pp. 279–286.
[27] Vitor R. Carvalho, Jonathan Elsas, William W. Cohen, Jaime G. Carbonell, A meta-learning approach for robust rank learning, in: SIGIR 2008 Workshop on Learning to Rank for Information Retrieval, 2008, pp. 15–23.
[28] C.J. Burges, R. Ragno, Q.V. Le, Learning to rank with nonsmooth cost functions, in: B. Schölkopf, J. Platt, T. Hoffman (Eds.), Advances in Neural Information Processing Systems, vol. 19, MIT Press, Cambridge, MA, 2007, pp. 193–200.
[29] Q. Wu, C.J. Burges, K.M. Svore, J. Gao, Adapting boosting for information retrieval measures, Information Retrieval 13 (3) (2010) 254–270.
[30] M. Schmitt, L. Martignon, On the complexity of learning lexicographic strategies, Journal of Machine Learning Research 7 (2006) 55–83.
[31] F. Yaman, T.J. Walsh, M.L. Littman, M. desJardins, Democratic approximation of lexicographic preference models, Artificial intelligence 175 (7–8) (2011) 1290–1307.
[32] F. Koriche, B. Zanuttini, Learning conditional preference networks with queries, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, pp. 1930–1935.
[33] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, W.-Y. Ma, Frank: a ranking method with fidelity loss, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, ACM, New York, NY, USA, 2007, pp. 383–390.
[34] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Annals of Statistics 29 (2000) 1189–1232.
[35] C.J.C. Burges, K.M. Svore, P.N. Bennett, A. Pastusiak, Q. Wu, Learning to rank using an ensemble of lambda-gradient models, Journal of Machine Learning Research – Proceedings Track (2011) 25–35.
[36] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li, Learning to rank: from pairwise approach to listwise approach, in: Proceedings of the 24th International Conference on Machine Learning, ICML '07, ACM, New York, NY, USA, 2007, pp. 129–136.
[37] F. Xia, T.-Y. Liu, H. Li, Statistical consistency of top-k ranking, in: Y. Bengio, D. Schuurmans, J. Lafferty, C.K.I. Williams, A. Culotta (Eds.), Advances in Neural Information Processing Systems, vol. 22, 2009, pp. 2098–2106.
[38] Y. Chevaleyre, F. Koriche, J. Lang, J. Mengin, B.Z. Abstract, Learning ordinal preferences on multiattribute domains: the case of cp-nets, in: Preference Learning, Springer, Berlin Heidelberg, 2010, pp. 273–296.
[39] C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, D. Poole, CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements, Journal of Artifical Intelligence Research 21 (2004) 135–191.
[40] P.B. Patnaik, The non-central χ^2- and F-distribution and their applications, Biometrika 36 (1949) 202–232.
[41] T. Kamishima, Nantonac collaborative filtering: recommendation based on order responses, in: Proceedings of ACM SIGKDD'03, ACM, New York, NY, USA, 2003, pp. 583–588.

Please cite this article in press as: J. Liu et al., Learning conditional preference network from noisy samples using hypothesis testing, Knowl. Based Syst. (2012), http://dx.doi.org/10.1016/j.knosys.2012.11.006