Discovering Statistically Non-Redundant Subgroups

Jiuyong Li[a,*], Jixue Liu[a], Hannu Toivonen[b], Kenji Satou[c], Youqiang Sun[e,d], Bingyu Sun[d]

[a]*School of Information Technology & Mathematical Sciences, University of South Australia, Australia*
[b]*Department of Computer Science, University of Helsinki, Finland*
[c]*Graduate School of Natural Science & Technology, Kanazawa University, Japan*
[d]*Institute of Intelligent Machines, Chinese Academy of Sciences, China*
[e]*School of Information Science & Technology, University of Science and Technology of China, China*

Abstract

The objective of subgroup discovery is to find groups of individuals who are statistically different from others in a large data set. Most existing measures of the quality of subgroups are intuitive and do not precisely capture statistical differences of a group with the other, and their discovered results contain many redundant subgroups. Odds ratio is a statistically sound measure to quantify the statistical difference of two groups for a certain outcome and it is a very suitable measure for quantifying the quality of subgroups. In this paper, we propose a statistically sound framework for statistically non-redundant subgroup discovery: measuring the quality of subgroups by the odds ratio and defining statistically non-redundant subgroups by the error bounds of odds ratios. We show that our proposed method is faster than most existing methods and discovers complete statistically non-redundant subgroups.

Keywords: Subgroups, non-redundancy, odds ratio, rules, search space pruning

1. Introduction

1.1. Subgroups and rules

The task of subgroup discovery was defined by Klösgen [24] and Wrobel [41] as follows: "Given a population of individuals and a property of those individuals that we are interested in, find population subgroups that are statistically 'most

*Corresponding author. Email: jiuyong.li@unisa.edu.au; Tel: 61883023898.

interesting', for example, are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest". For example, let the cancer be a property that we are interested in a medical data set. Subgroups indicate people who are more vulnerable to the cancer than others. Subgroup discovery has wide applications, such as [5, 17, 26, 27]. The subgroup discovery has extended from nominal data to numerical data [19, 33]. In this paper, we consider subgroup discovery in nominal data.

Subgroup discovery is closely related to rule discovery [37, 40]. A subgroup is a subset of a data set, and can be described by a rule. For example, subgroup "male and 40-50" comprises 40 to 50 year old male individuals. Assume that this group has a higher chance to suffer from depression than others not in the group. Equivalently, this subgroup can be represented as a rule "Gender = male and Age = (40-50) $\rightarrow$ depression" with a high confidence. The discovery of subgroups is equivalent to the discovery of rules. Subgroup discovery has been shown equivalent to the discovery of contrast sets [6] and emerging patterns [12]. Association rule discovery algorithms [2, 21] have been adapted to subgroup discovery [4, 23]. We refer readers to Novak et al.'s comprehensive survey [37] for detailed discussions.

In the discussions in this paper, we use subgroups and rules exchangeably and the right hand sides (RHS) of rules are fixed to a value of the outcome attribute, the property of interest.

One major problem of subgroup discovery and application is redundancy. Subgroups are represented as descriptors (or left hand sides of rules). A number of descriptors may correspond to the same set or overlapped sets of records (or instances) in a data set. For example, "pregnant $\rightarrow$ depression " and "female and pregnant $\rightarrow$ depression" map to the same set of records in a data set. Semantically, they are the same and hence one is redundant. If we relax the definition of redundant subgroups to include the descriptors mapping to record sets that are overlapped (not necessarily identical), the number of redundant subgroups gets very large. The relaxation of redundancy definition makes redundancy detection more challenging. It is easily to find hundreds to thousands of subgroups that slightly vary from each other. For example, we may find a number of subgroups, such as "40-50 $\rightarrow$ depression", "40-50 and male $\rightarrow$ depression", "40-50 and male and in Australia $\rightarrow$ depression", and so on. They carry the similar information and some are redundant to others.

The redundancy reduces the efficiency of the discovery. It is common that a redundant subgroup set is tens to hundreds of times larger a non-redundant subgroup

set because there are so many redundant subgroups. The search space for redundant subgroups is also significantly larger than that for non-redundant subgroups. The redundancy reduces the efficiency of subgroup discovery significantly.

The redundancy also reduces the usability of subgroups. The number of subgroups can be unrealistically large, for example, hundreds of thousands to millions. Users are only interested in and are only able to assess tens to a few hundreds of subgroups. It is necessary to filter out "uninteresting" subgroups and present a small number of subgroups to users. Top k seems an effective method for such selection, however, it was found that 99 out of top 100 subgroups are redundant in a data set [29]. A top k method that cannot remove redundant subgroups is not helpful.

The problem of redundancy has been studied for a few years. Most work relied on the top-k solutions with various interesting metrics to sort the subgroups [4, 20, 28, 36]. The various interesting metrics remedy the problem but does not solve the problem caused by the redundancy. Other studies use some syntactical definitions to remove redundancies, for example, closure [29, 30], productivity [39], relevance [18], constraints [7] and so on. They have achieved various successes, but they rely on a constant threshold in the quality improvement to remove redundant subgroups.

There is a general consensus in the previous research that a non redundant subgroup should have a quality improvement over all its super subgroups. Let us consider two subgroups "40-50 $\rightarrow$ depression" and "40-50 and male $\rightarrow$ depression". If the second subgroup has a lower quality than the first one, it is of no interest to users when users are looking for high quality subgroups. When the second one has a marginal quality improvement over the first one, it is not of interest either since the improvement may be caused by fluctuations of data. For example, a quantified quality value may change in data everyday with new added instances. The second subgroup is only of interest to users when its quality is significantly higher than that of the first one and the improvement is unlikely caused by data fluctuations.

A question is how big an improvement should be. Most work chooses a baseline, any non-zero improvement, or leaves the choice to users as a user specified parameter [29, 30, 18, 7]. In fact, it is a difficult choice and any constant thresholds are unsuitable. Fundamentally, we try to make a judgement that two subgroups are not equivalent so one is not a redundant version of the other. In other words, we would like to know if the qualities of two subgroups are statistically equivalent. The answer to this question depends on the size of a subgroup and the counts of the value of interest within and outside the subgroup. It will not be a con-

stant threshold for all subgroups. Some work [39] answering this question uses a hypothesis test whether two qualities of subgroups are equivalent.

In this paper, we propose a statistical sound definition of non-redundant subgroups where the quality improvement for determining redundancy vary with the sizes of subgroups and the counts of the value of interest within and outside the subgroups. We explore the computational properties for efficient discovering the statistically non-redundant subgroups so the pruning redundant subgroups can be achieved in the discovery process. We present an efficient optimal algorithm for the efficient discovery of statistically non-redundant subgroups. The algorithm has been shown more efficient than other classic subgroup discovery methods and new non-redundant subgroup discovery methods especially in large data sets.

2. Related work

In this section, we mainly differentiate our work from other most related work.

2.1. Measures of the quality of subgroups

A measure of the quality of subgroups aims at capturing statistical difference of a group with the other group including remaining individuals in a population. Most quality measures of subgroups, such as generalisation quotient [16], binomial test quality function [24], weighted relative accuracy [23], wighted Krimp gain [29], support difference [6], and growth rate [12] (please refer to reviews [22, 37] for more definitions), intuitively capture the difference between two groups, and they may not be statistically sound. One different measure [14] is to compare the differences of Bayesian networks constructed from the subgroup and the whole data. The larger the difference is, the higher quality is. However, it is only possible to construct optimal Bayesian networks in very low dimensional data sets. This measure has a significant limitation.

Odds ratio is a measure for quantifying the statistical difference of two groups with regard to an outcome [15], and it has been widely used in health and medical research and practice. It is very suitable for measuring the quality of subgroups. Odds ratio has good interpretability. For example, if the odds ratio of a group of patients to a cancer is 3, the group is 3 times more likely to suffer from the cancer than others.

Odds ratio has an advantage over other statistical measures of the quality of subgroups. Other statistical measures used in rule discovery such as Chi-square [35] and p-value [13, 39], do not indicate the strength of the association [15, 34]. They are unsuitable for comparing values of quality of two sub-

groups and unsuitable for choosing top subgroups. In contrast, odds ratio indicates the strength of an association [15]. The degree of association is necessary to compare the strength of association of two subgroups to define redundancies.

2.2. Redundancy

There generally are two types of redundant subgroups.

The first type of redundant subgroups are those that have different descriptors but contain the same set of data records. For example, sub groups "pregnant $\rightarrow$ oedema" and " female and pregnant $\rightarrow$ oedema" are the same subgroup. In many cases, the equivalence of redundant subgroups are not so obvious to tell. Formally, all such subgroups have the same closure in the attribute lattice of a data set [38]. This type of redundancy has been well defined in subgroup and rule discovery, and typical works are [8, 31, 42]. Such redundant subgroups only account for a small proportion of all redundant subgroups.

The second type of redundant subgroups are those that cover a subset (or a similar set) of data records of some other subgroups. But their qualities are lower than the qualities of other subgroups respectively. For example, subgroup "40-50 and male $\rightarrow$ depression" is a redundant subgroup if its quality is lower than that of subgroup "40-50 $\rightarrow$ depression". Examples of this type of redundancy are relevant subgroups [18], non-redundant subgroups [29, 30], productive rules [39], optimal rules [32] and constraint rules [7].

The first type of redundancy can be considered as a special case of the second type of redundancy when two subgroups cover the same set of records and have the same quality. Non-redundant subgroups of the second type still involve many statistically redundant subgroups. Let us assume that "40-50 $\rightarrow$ depression" is a quality subgroup. There will be a lot of sub subgroups like "40-50 and male $\rightarrow$ depression", "40-50 and male and in Australia $\rightarrow$ depression" and so on. If those subgroups have higher quality than sub group "40-50 $\rightarrow$ depression", they are not detected by the existing definitions. However, those slight quality improvements are likely caused by data fluctuations. A constant minimum quality improvement does not solve the problem since the required improvements vary with the sizes of subgroups, and the counts of interesting value within and outside the subgroups. A sound solution is to test if the quality improvement is statistically significant. Webb [39] has used a hypothesis test approach to identify / remove redundant patterns by holdout evaluation or direct adjustment. This paper will use the confidence intervals of odds ratios to measure the significance of quality improvement. The measurement results in effective pruning criteria for subgroup discovery. Note that work in [13] made use of statistical test to validate

the subgroups discovered. However, the test is for subgroups, not for the quality improvement over other subgroups. Our proposed method tests both.

Redundancy results in too many subgroups, but few subgroups do not mean that there is no redundancy. Due to the large number of subgroups, most discovery methods are top k based ones, such as [4, 20, 28, 36]. A top k method only selects subgroups with the highest quality up to the number k. The top k subgroups may still contain redundant subgroups. This has been observed by [29]. For example, in our experiment, all top 258 subgroups are variants of a statistically non-redundant subgroup ranked 259 in the subgroup list. So, it is possible that all top k subgroups are redundant if we do not use a non-redundant subgroup discovery method. More likely, all top k subgroups come from a dense subsection of a data set and no subgroups cover other data sections. Therefore, when there is a computational feasibility, an optimal discovery method is desirable. The algorithm presented in this paper is an optimal one. It is efficient and guarantees the completeness of results.

3. Statistically non-redundant subgroups and pruning criteria

3.1. Problem definition

Let data set D contain m descriptive attributes, $A_1, A_2, \ldots, A_m$, and one outcome attribute, Z, and let all values be categorical. If there are numerical attributes, they will be discretised firstly. One value in the outcome attribute is a property that users are interested in. Let the data set contain n records, each of which is a description of an individual. Let a descriptor be a set of values of some attributes. A subgroup is a subset of D with the same descriptor. A subgroup is interesting if it is statistically different from other individuals.

We link the subgroup definition to rule definition. Let each value in attributes $A_1, A_2, \ldots, A_m$ be uniquely coded, called an item. A set of items is called an itemset. A set of records containing an itemset form a subgroup, and the itemset is called the descriptor of the subgroup. The subgroup discovery problem is the problem of discovering association rules with a fixed consequence.

In this paper, the RHS (the right hand side) of rules are fixed to a value of the outcome attribute. In the following discussions, we do not present the RHS and use the LHS (the left hand side) of a rule as a subgroup.

Given the contingency table of a subgroup in Table 1, P is the descriptor of a subgroup and z is the value of interest in the outcome attribute Z. $\overline{P}$ indicates all other individuals apart from the individuals included by subgroup P. Numbers in cells are represented following the statistical convention. We also use them

	z	$\overline{z}$	total
P	n_{11}	n_{12}	n_{1*}
$\overline{P}$	n_{21}	n_{22}	n_{2*}
total	n_{*1}	n_{*2}	n

Table 1: The contingency table of subgroup P. z is the value of interest.

to indicate different cells in the following discussions. The number of records containing P in a data set is denoted as n_{1*} where n_{11} records include the value of interest and n_{12} records do not. $\overline{P}$ is the complementary subgroup of P and $n_{2*} = n - n_{1*}$. Again, n_{21} records in subgroup $\overline{P}$ include the value of interest, and n_{22} records do not. When the context is clear, we use matrix $[n_{11}, n_{12}; n_{21}, n_{22}]$ to represent a contingency table.

The *support* of subgroup P in a data set is defined as $\text{supp}(P) = n_{11}/n$. We call n_{1*} the size of subgroup P.

The *odds ratio* of subgroup P is defined as the following.

$$\text{OR}(P) = \frac{n_{11}}{n_{12}} / \frac{n_{21}}{n_{22}} = \frac{n_{11} * n_{22}}{n_{12} * n_{21}}$$

An odds ratio of 1 indicates that z is equally likely to occur in both subgroups P and $\overline{P}$. An odds ratio greater than 1 indicates that z is more likely to occur in the subgroup P than $\overline{P}$. And an odds ratio less than 1 indicates that z is less likely to occur in the P than $\overline{P}$. In statistical terms, an odds ratio greater than 1 indicates the strength of positive association between the subgroup and the property of interest.

Suppose that we have two inclusive subgroups P_1 = (male, age 40-50) and P_2 = (male, age 40-50, living in City X). P_2 is a sub subgroup of P_1, and all individuals included by P_2 are included by P_1. In principle, P_2 is of interest only when its odds ratio is significantly higher than P_1 since our objective is to find subgroups with high odds ratios. Note that subgroups of low odds ratios may be of interest to users too, but it is a symmetric problem of the problem discussed in this paper when z and $\overline{z}$ are swapped, and hence we do not consider it here.

Let $\text{OR}(P_2) = 2.2$ and $\text{OR}(P_1) = 2.0$. Is P_2 interesting? Before we are able to answer this question, we need to answer the following question: "Is there a difference between $\text{OR}(P_2) = 2.2$ and $\text{OR}(P_1) = 2.0$ statistically?" In other words, if one randomly samples a set of records with the size of P_2 from subgroup P_1, how likely would s/he get an odds ratio of 2.2 or higher? If it is likely, P_2 is a random variation of P_1 and we consider that it is redundant.

Definition 1 (Statistical redundancy) *Consider subgroups P_1 and its sub sub-*

group P_2 where $\mathrm{OR}(P_1) > 1$ and $\mathrm{OR}(P_2) > 1$, subgroup P_2 is statistically redundant with respect to P_1 if (1) $\mathrm{OR}(P_2) \leq \mathrm{OR}(P_1)$ or (2) $\mathrm{OR}(P_2) > \mathrm{OR}(P_1)$ but the difference between $\mathrm{OR}(P_2)$ and $\mathrm{OR}(P_1)$ is statistically insignificant.

For example, if $(\mathrm{OR}(P_2) = 2.2) - (\mathrm{OR}(P_1) = 2.0)$ is statistically insignificant in the previous example, then P_2 is a statistically redundant subgroup of P_1. Such a significance depends on the size of subgroup P_2. For example, if P_2 is a large subgroup, the 10% improvement in odds ratio is significant. In contrast, if P_1 is a small subgroup, the 10% improvement in odds ratio may be caused by the data fluctuation and hence insignificant.

Now, we can summarise the objective of statistically non-redundant subgroup discovery.

Problem 1 *Given a data set and a value of interest, a subgroup is a subset of data with the same descriptor. An interesting subgroup has at least the minimum support and its odds ratio for the value of interest is greater than 1 and the difference is statistically significant. Non-redundant subgroup discovery is to find all statistically non-redundant interesting subgroups.*

3.2. A criterion for statistical redundancy and its computational properties

A measure of the quality of a subgroup is an approximation. When we say that the quality of a subgroup is 3, it might be estimated as 2.9 when it is calculated in a data set with two more days' data, or as 3.15 in a data set with one more week data. Such a difference is called a variation (or an uncertainty or an error) in the measurements. So when we say that an estimation of the quality of a subgroup is 3, we imply that the quality is in an interval, e.g. $[2.8, 3.2]$, called a confidence interval. In the same measurement, 3.15 is not necessarily larger than 2.9 when uncertainties are taken into consideration. When we see the qualities of two subgroups are 2.9 and 3.15 respectively, both subgroups very likely have the same quality. In this section, we will discuss how to estimate the confidence interval of an odd ratio, and how to use confidence intervals to determine the equality or inequality of the qualities of two subgroups to define the redundancy.

Given a subgroup P with the contingency table in Table 1, the confidence interval of the odds ratio is calculated as the following [15]
$[\mathrm{OR}(P)exp(-\omega), \mathrm{OR}(P)exp(\omega)]$ where $\omega = z_{\alpha/2}\sqrt{\frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}}$. $z_{\alpha/2}$ is the critical value of the confidence interval. When the confidence is 95%, $z_{\alpha/2}$ = 1.96. In other words, $\mathrm{OR}(P)$ is an estimation. Its real value is in the interval $[\mathrm{OR}(P)_-, \mathrm{OR}(P)_+]$ with 95% probability, where $\mathrm{OR}(P)_-$ and $\mathrm{OR}(P)_+$ stand

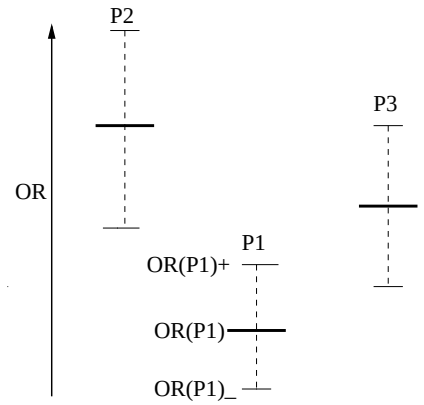

Figure 1: A demonstration of statistically non-redundant subgroups. Subgroup P_3 is statistically redundant with respect to subgroup P_1 and subgroup P_2 is not.

for $\mathrm{OR}(P)exp(-\omega)$ and $\mathrm{OR}(P)exp(\omega)$ respectively. $\mathrm{OR}(P)_-$ and $\mathrm{OR}(P)_+$ are called the left bound and the right bound of odds ratio $\mathrm{OR}(P)$.

Given P_1 and its sub subgroup P_2, their odds ratios are estimated as $[\mathrm{OR}(P_1)_-, \mathrm{OR}(P_1)_+]$ and $[\mathrm{OR}(P_2)_-, \mathrm{OR}(P_2)_+]$. Suppose that $\mathrm{OR}(P_2) > \mathrm{OR}(P_1)$. The difference between the two odds ratios is statistically significant if $\mathrm{OR}(P_2)_- > \mathrm{OR}(P_1)_+$. In other words, we have a high confidence to conclude that two odds ratios are different if their confidence intervals do not overlap. Otherwise, the difference is insignificant. The wider the gap between the intervals, the smaller the chance for two odds ratios being the same.

We note that the above analysis is based on the fact that the odds ratios of P_1 and P_2 are independent. In our case, they are not since P_2 is a subset of P_1. A difference between the odds ratios of P_1 and P_2 is more significant than the same difference between two independent odds ratios. In practice, a critical value for a dependent case should be smaller than that for an independent case.

See Figure 1 for example. Odds ratios of subgroups P_2 and P_1 are statistically different since their confidence intervals are not overlapped. In contrast, the difference between the odds ratios of subgroups P_3 and P_1 is insignificant since their confidence intervals are overlapped. We have the following definition.

Definition 2 (A criterion for statistically redundant subgroups) *Consider subgroup P_1 and its sub subgroup P_2, subgroup P_2 is statistically redundant with respect to subgroup P_1, if* $\mathrm{OR}(P_1)_+ > \mathrm{OR}(P_2)_-$.

When $\mathrm{OR}(P_2) \leq \mathrm{OR}(P_1)$, $\mathrm{OR}(P_1)_+ > \mathrm{OR}(P_2)_-$ trivially holds. When

$\mathrm{OR}(P_2) > \mathrm{OR}(P_1)$, we need to compare $\mathrm{OR}(P_1)_+ > \mathrm{OR}(P_2)_-$ case by case. When we search for statistically non-redundant subgroups using a branch and bound algorithm, it will be good to know the upper bound of the left bounds of odds ratios of all the sub subgroups of P_1.

In order to obtain the upper bound of left bounds of all subgroups of subgroup P_1, we define variables x and y to quantify value changes between contingency tables of subgroup P_1 and its sub subgroup P_2. Let the contingency table of subgroup P_1 be as follow.

Table I	z	$\overline{z}$	total
P_1	a	b	$a+b$
$\overline{P_1}$	c	d	$c+d$
total	$a+c$	$b+d$	n

We use a, b, c and d to indicate counts to avoid subscripts. $a+b+c+d=n$. The contingency table of P_2 is given as follow.

Table II	z	$\overline{z}$	total
P_2	$a-x$	$b-y$	$a+b-x-y$
$\overline{P_2}$	$c+x$	$d+y$	$b+c+x+y$
total	$a+c$	$b+d$	n

$0 \leq x \leq a$ and $0 \leq y \leq b$. The size of subgroup P_2 is at most the same as the size of P_1. Therefore, both counts in Cells n_{11} and n_{12} drop. However, the total counts of z and $\overline{z}$ are constant, and hence the drops are added to Cells n_{21} and n_{22}. The variables in the specialisation of a subgroup to its sub subgroup are x and y.

In the following, we will discuss how the odds ratio of P_2 and the left bound of the odds ratio change with x and y.

Lemma 1 *Given a subgroup P_1, and one of its sub subgroups P_2, their contingency tables are described in Tables I and II where a, b, c and d are constants. Let $x = 0$. $\mathrm{OR}(P_2)$ monotonously increases with y. $\mathrm{OR}(P_2)_-$ monotonously increases with y when $d + y > b - y$.*

Proof

$$\begin{aligned} \mathrm{OR}(P_2) &= \frac{a}{c} * \frac{d+y}{b-y} \\ \frac{\mathrm{d}\,\mathrm{OR}(P_2)}{\mathrm{d}\,y} &= \frac{a}{c} * \frac{b+d}{(b-y)^2} > 0 \end{aligned}$$

Therefore, $\mathrm{OR}(P_2)$ monotonously increases with y.

$$\begin{aligned} \omega &= z_{\alpha/2}\sqrt{\frac{1}{a}+\frac{1}{c}+\frac{1}{b-y}+\frac{1}{d+y}} = z_{\alpha/2}\sqrt{w_1} \\ \frac{\mathrm{d}(-\omega)}{\mathrm{d}\,y} &= \frac{z_{\alpha/2}}{2} w_1^{-\frac{1}{2}} (\frac{1}{(b-y)^2} - \frac{1}{(d+y)^2}) > 0 \\ \mathrm{OR}(P_2)_- &= \mathrm{OR}(P_2) e^{-\omega} \\ \frac{\mathrm{d}\,\mathrm{OR}(P_2)_-}{\mathrm{d}\,y} &= \frac{\mathrm{d}\,\mathrm{OR}(P_2)}{\mathrm{d}\,y} e^{-\omega} + \mathrm{OR}(P_2) e^{-\omega} \frac{\mathrm{d}(-\omega)}{\mathrm{d}\,y} > 0 \end{aligned}$$

Therefore, $\mathrm{OR}(P_2)_-$ monotonously increases with y when $d + y > b - y$. □

The practical meanings of Lemma 1 is that with the increase of y, $\mathrm{OR}(P_2)$ increases and so does $\mathrm{OR}(P_2)_-$. Therefore, in order to have a sub subgroup with the largest odds ratio and the largest left bound of the odds ratio, value y should be maximised.

The following example shows that condition $(d+y) > (b-y)$ (or equivalently $n_{22} > n_{21}$) in Lemma 2, is easily satisfied in subgroup discovery. This is because n_{21} indicates the count of false positives of a subgroup, and this count needs to be small to make a subgroup interesting.

	depression	normal	total
(male, 40-50)	15	5	20
Others	35	45	80
total	50	50	100

Lemma 2 *Given a subgroup P_1, and one of its sub subgroups P_2, their contingency tables are described Tables I and II where a, b, c and d are constants. Let $y = 0$. $\mathrm{OR}(P_2)$ monotonously decreases with x. $\mathrm{OR}(P)_-$ monotonously decreases with x.*

Proof

$$\begin{aligned}
\mathrm{OR}(P_2) &= \frac{a-x}{c+x} * \frac{d}{b} \\
\frac{\mathrm{d}\,\mathrm{OR}(P_2)}{\mathrm{d}\,x} &= -\frac{d}{b} * \frac{a+c}{(c+x)^2} < 0
\end{aligned}$$

Therefore, $\mathrm{OR}(P_2)$ monotonously decreases with x.

$$\begin{aligned}
\omega &= z_{\alpha/2}\sqrt{\frac{1}{a-x}+\frac{1}{c+x}+\frac{1}{b}+\frac{1}{d}} = z_{\alpha/2}\sqrt{w_2} \\
\frac{\mathrm{d}(-\omega)}{\mathrm{d}\,x} &= \frac{z_{\alpha/2}}{2} w_2^{-\frac{1}{2}} (\frac{1}{(a-x)^2} - \frac{1}{(c+x)^2}) \\
\mathrm{OR}(P_2)_- &= \mathrm{OR}(P_2) e^{-\omega} \\
\frac{\mathrm{d}\,\mathrm{OR}(P_2)_-}{\mathrm{d}\,x} &= \frac{\mathrm{d}\,\mathrm{OR}(P_2)}{\mathrm{d}\,x} e^{-\omega} + \mathrm{OR}(P_2) e^{-\omega} \frac{\mathrm{d}(-\omega)}{\mathrm{d}\,x} \\
&= \frac{d}{b} e^{-\omega} (-\frac{a+c}{(c+x)^2} + \frac{a-x}{c+x} * \frac{z_{\alpha/2}}{2} w_2^{-\frac{1}{2}} (\frac{1}{(a-x)^2} - \frac{1}{(c+x)^2}))
\end{aligned}$$

Let $A = \frac{a+c}{(c+x)^2}$ and $B = \frac{a-x}{c+x} * \frac{z_{\alpha/2}}{2} w_2^{-\frac{1}{2}} (\frac{1}{(a-x)^2} - \frac{1}{(c+x)^2})$.

$$\begin{aligned}
B &< \frac{a-x}{c+x} * \frac{z_{\alpha/2}}{2} w_2^{-\frac{1}{2}} (\frac{1}{(a-x)^2}) \\
&= \frac{z_{\alpha/2}}{2} w_2^{-\frac{1}{2}} \frac{1}{(c+x)(a-x)}
\end{aligned}$$

To have $B < A$ (so that $\frac{\mathrm{d}\,\mathrm{OR}(P_2)_-}{\mathrm{d}\,x} < 0$), it is necessary that the following

inequations hold.

$$\begin{aligned}
\frac{z_{\alpha/2}}{2} w_2^{-\frac{1}{2}} \frac{1}{(c+x)(a-x)} &< \frac{a+c}{(c+x)^2} \\
\frac{z_{\alpha/2}}{2} w_2^{-\frac{1}{2}} \frac{1}{a-x} &< \frac{a+c}{c+x} \\
w_2^{-\frac{1}{2}} &< \frac{2}{z_{\alpha/2}} * \frac{(a+c)(a-x)}{c+x} \\
\sqrt{\frac{1}{a-x} + \frac{1}{c+x} + \frac{1}{b} + \frac{1}{d}} &> \frac{z_{\alpha/2}}{2} * \frac{c+x}{(a+c)(a-x)} \\
\sqrt{\frac{1}{a-x}} &> \frac{z_{\alpha/2}}{2} * \frac{c+x}{(a+c)(a-x)} \\
(a-x) &< \frac{4}{z_{\alpha/2}^2} (\frac{(a+c)(a-x)}{c+x})^2 \\
1 &< \frac{4(a-x)}{z_{\alpha/2}^2} (\frac{a+c}{c+x})^2 \qquad (1)
\end{aligned}$$

Because $(a+c) > (c+x)$, Inequation (1) holds when $z_{\alpha/2}$ is smaller than $2\sqrt{\#\text{min support count}}$ since $(a-x)$ is bounded by the count of the minimum support. Normally, $z_{\alpha/2} = 1.96$, corresponding to the 95% confidence level, requires the count of the minimum support as 1. $z_{\alpha/2} = 4$, corresponding to the 99.99% confidence level, requires the count of the minimum support as 4. The count of the minimum support is at least 5 in all subgroup discovery applications. So, Inequation (1) holds.

Therefore, $\mathrm{OR}(P)_{-}$ monotonously decreases with x.

□

The practical meanings of Lemma 2 is that with the increase of x, $\mathrm{OR}(P_2)$ decreases, and so does $\mathrm{OR}(P)_{-}$. Therefore, in order to have a sub subgroup with the largest odds ratio and the largest left bound of the odds ratio, x should be minimised.

Now we are able to deduce the upper bound of odds ratios and the upper bound of the left bounds of odds ratios of all sub subgroups of subgroup P_1.

Theorem 1 *Given a subgroup P_1, its contingency table is described in Table I where a, b, c and d are constants. The largest possible odds ratio of all its sub subgroups is $\mathrm{OR}(P_1)^{ub} = \frac{a(b+d-\beta)}{\beta c}$ where $\beta \geq 5$. The largest possible left bound*

of the odds ratios of all sub subgroups of P_1 *is* $\mathrm{OR}(P_1)^{ub}_{-} = \mathrm{OR}(P_1)^{ub}e^{-\omega}$ *where* $\omega = z_{\alpha/2}\sqrt{\frac{1}{a}+\frac{1}{c}+\frac{1}{\beta}+\frac{1}{b+d-\beta}}$.

Proof Let P_2 be a sub subgroup of P_1. Its contingency table is described as Table II. By Lemma 2, we have the following inequation when x takes the smallest value.

$$\mathrm{OR}(P_2) = \frac{(a-x)(d+y)}{(c+x)(b-y)} \leq \frac{a(d+y)}{c(b-y)}$$

By Lemma 1, we have the following inequation when y takes the largest possible number, i.e. $b-\beta$.

$$\frac{a(d+y)}{c(b-y)} \leq \frac{a(d+b-\beta)}{\beta c}$$

Therefore, the largest odds ratio of $\mathrm{OR}(P_2)$ is $\frac{a(b+d-\beta)}{\beta c}$. It is the upper bound of odds ratios of all sub subgroups of P_1, denoted as $\mathrm{OR}(P_1)^{ub}$.

Since both $\mathrm{OR}(P_2)$ and $\mathrm{OR}(P_2)_-$ satisfy the same monotonous properties Lemmas 1 and 2, the points of their largest values overlap. The upper bound of the left bounds of the odds ratios of all sub subgroups of P_1 is $\mathrm{OR}(P_1)^{ub}_{-} = \mathrm{OR}(P_1)^{ub}e^{-\omega}$ where $\omega = z_{\alpha/2}\sqrt{\frac{1}{a}+\frac{1}{c}+\frac{1}{\beta}+\frac{1}{b+d-\beta}}$. □

Now we explain the practical meanings of β. Let us start with an example. If a subgroup has the following contingency table $x_{11} = 20, x_{12} = 2, x_{21} = 30, x_{22} = 48$, its odds ratio is 16. Assume that its sub subgroup has the following contingency table $x_{11} = 20, x_{12} = 1, x_{21} = 30, x_{22} = 49$. The odds ratio of the sub subgroup is 32.7. The difference of the two contingency tables is that one instance is moved from Cell n_{12} to Cell n_{22} from the subgroup to the sub subgroup. Such a change could be caused by a data fluctuation. However, the odds ratio is doubled. As χ square for testing association [10], the counts in a contingency table should be at least 5 to be reliable. The count of n_{21} and n_{22} are much larger than 5 as long as a data set is not too small. The count of n_{11} is bounded by the minimum support. Here β is the lower bound for n_{12} for a reliable odds ratio estimation. When the count in n_{12} falls lower than β, we use the β as a count.

The theorem can be used for the efficient discovery of subgroups in a branch and bound search. When we search for the current subgroup P_1, we can obtain the upper bound of left-bounds of confidence intervals of odds ratios of all sub

subgroups of subgroup P_1. If $\mathrm{OR}(P_1)^{ub}_{-} \leq \mathrm{OR}(P_1)_{+}$, then all sub subgroups of P_1 are statistically redundant. More specifically, we have the following pruning criterion.

Criterion 1 (Upper bound pruning) *Consider a subgroup P_1 and let $\mathrm{OR}(P_1)^{ub}_{-}$ be the upper bound of left bounds of odds ratios of all sub subgroups of P_1. If $\mathrm{OR}(P_1)^{ub}_{-} \leq \mathrm{OR}(P_1)_{+}$, all sub subgroups of P_1 are prunable.*

Proof When $\mathrm{OR}(P_1)^{ub}_{-} \leq \mathrm{OR}(P_1)_{+}$, the highest possible left bound of the odds ratios of all sub subgroups of P_1 is not as high as the right bound of the odd ratio of P_1. So the confidence interval of odds ratio of P_1 overlaps with that of any of its sub subgroups. According to Definition 2, all sub subgroups of P_1 are statistically redundant. So they should be pruned. □

This criterion is very effective in a branch and bound search. For example, let a data set contain ten attribute values $S = \{a, b, c, d, e, f, g, h, i, j\}$ and a value of interest z. Let the length of the descriptor of a subgroup be up to 5. The candidate subgroups include the sets in the power set of S with the cardinality up to 5, and the number of the candidates is 637. Let us assume that all candidate subgroups satisfy the minimum support requirement, and $P_1 = a$ is the current subgroup under consideration. The number of all sub subgroups of P_1 is $\sum_{i=1}^{i=4} \binom{9}{i} = 255$. If $\mathrm{OR}(P_1)^{ub}_{-} < \mathrm{OR}(P_1)_{+}$. The 255 sub subgroups should not be searched, and the search space is reduced greatly.

Another pruning criterion extended from the optimality pruning criterion in [32].

Criterion 2 (Optimality pruning) *Let P_2 be a sub subgroup of P_1. The contingency tables of the two subgroups are described in Table I and II. If $y = 0$, then P_2 and all its sub subgroups are prunable.*

Proof Firstly, we prove that P_2 is prunable. When $y = 0$, $\mathrm{OR}(P_2) \leq \mathrm{OR}(P_1)$ following Lemma 2 because of $x \geq 0$, and hence $\mathrm{OR}(P_2)_{-} < \mathrm{OR}(P_1)_{+}$. If P_1 is statistically non-redundant, P_2 is statistically redundant with respect to P_1. If P_1 is statistically redundant with respect to another subgroup P, P_2 is also statistically redundant to P. Therefore, P_2 is statistically redundant.

Secondly, we will prove that all sub subgroups of P_2 are prunable. We start with some notations. Let $I(P_1)$ include all instances containing the descriptor of subgroup P_1 in a data set. $I(P_1) = I^{+}(P_1) \cup I^{-}(P_1)$ where $I^{+}(P_1)$ contains all instances in $I(P_1)$ with the property of interest and $I^{-}(P_1)$ contains those without.

Let the set difference of descriptors of P_1 and P_2 be Q, i.e. $P_2 \backslash P_1 = Q$. Consider that Q is also a subgroup. $I^-(P_2) = I^-(P_1) \cap I^-(Q)$. When $y = 0$, $I^-(P_2) = I^-(P_1)$ and equivalently $I^-(P_1) \subseteq I^-(Q)$.

Let P_2R be a sub subgroup of P_2 and P_2R is also a sub subgroup of P_1R. $I^-(P_1R) \subseteq I^-(P_1) \subseteq I^-(Q)$, and hence $y' = |I^-(P_1R)| - |I^-(P_2R)| = 0$ by noting that $P_2R = P_1QR$. Following Lemma 2, $\mathrm{OR}(P_2R) \leq \mathrm{OR}(P_1R)$ because of $y' = 0$, and hence $\mathrm{OR}(P_2R)_- < \mathrm{OR}(P_1R)_+$. If P_1R is statistically non-redundant, P_2R is statistically redundant with respect to P_1R. If P_1R is statistically redundant with respect to another subgroup P', P_2R is also statistically redundant to P'. Therefore, subgroup P_2R is statistically redundant.

P_2R is any sub subgroup of P_2 and hence all sub subgroups of P_2 are statistically redundant. □

Now we discuss the practical meanings of Criterion 2. In a branch and bound search, we only search sub subgroups that have non-zero y values from their super subgroups. For example, let P_1 = (age 40-50) and P_2 = (age 40-50, male). Assume that $y = 0$ in the specialisation of subgroup P_1 to P_2. In other words, when subgroup P_1 is specialised to subgroup P_2, no false positive instances have been reduced. So all sub subgroups of P_2 should not be searched because they are redundant. For example, subgroup (age 40-50, male, living in City X) is redundant to subgroup (age 40-50, living in City X). Note that the criterion makes no comparison between subgroups (age 40-50, male, living in City X) with subgroup (age 40-50). The comparisons are between a super subgroup and a sub subgroup with and without term "male", the set difference between descriptors of P_1 and P_2.

The closure pruning [8, 31, 42] is implied by the above optimality pruning. When subgroups P_1 and P_2 have the same closure, P_1 and P_2 have the same support and the same contingency tables. Equivalently, $x = 0$ and $y = 0$ (the x and y in Tables I and II). In contrast, the optimality pruning only needs $y = 0$. So, the closure pruning is implied by the optimality pruning.

4. Algorithm

We design an efficient algorithm to discover statistically non-redundant subgroups by using Criteria 1 and 2. The algorithm is based on a branch and bound search. The algorithm is an optimal one because it returns all statistically non-redundant subgroups.

4.1. Data structure

This algorithm employs the prefix tree as the base data structure for subgroup discovery. A prefix tree is an ordered tree to store ordered sets. Each node stores an ordered set. A node stores the maximal common set (prefix set) of its child nodes. The label of a node is the different object between its set and the set of its parent (the last object in its ordered set). The root of the prefix tree stores an empty set. Given a set of objects, called O, a complete prefix tree will store the power set of the set O. Figure 2 shows an exemplar prefix tree. A prefix tree has been shown to be a good data structure for branch and bound search [7, 9].

For a data set stored as a table, we firstly code each distinct value uniquely, and call the coded value as an item. Let us assume that all items are ordered. The way of ordering does not matter at this stage (see the end of this section for more discussions). The order is used to prevent the generation of duplicate candidates, such as abc and cba. When the prefix tree is used for subgroup discovery, each node stores a subgroup and is labelled by an item, which is the last item in the descriptor of the subgroup stored at the node. The prefix of the descriptor of a subgroup is the descriptor of its directly linked parent nodes in the prefix tree.

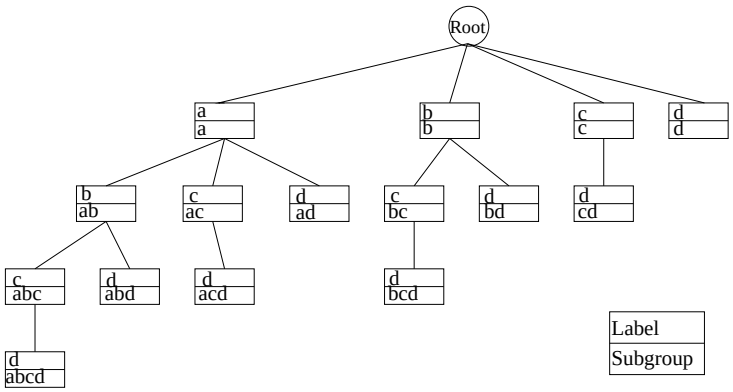

Figure 2: An example of prefix tree. a, b, c, d here stand for items, uniquely coded attribute values. An itemset is a descriptor of a subgroup.

To facilitate pruning by Criteria 2, backtrack links to all of its parent nodes of a subgroup should be added. For example, nodes ab, bc and ac are parent nodes of node abc. There is a direct link between nodes abc and ab, but there are no links between node abc and nodes ac and bc. Adding backtrack links between node abc and nodes ac and bc is easy for finding super subgroups of a subgroup in the previous level. For the simplicity of illustration, we do not draw backtrack links in the figure.

a			d	y
a		c		y
a	b	c		y
a	b	c	d	y
	b	c	d	y
a			d	n
	b		d	n
	b	c	d	n
	b			n
		c	d	n

Table 2: An itemised data set for the running example. a, b, c, d indicate values of descriptive attributes. When a value does not occur in a row, we leave the cell empty. y and n indicate values of the outcome attribute.

There are some advantages for using a prefix tree as the data structure.

- Pairs for candidate generation are naturally grouped under a node. For example, abc and abd are under node ab. We do not have to search for them in a list or a hash tree. See Subsection 4.3 for details.
- A stored subgroup is easily accessed because its descriptor shows the path from the root to the node storing itself. For example, labels a, c and d in sequence lead to subgroup acd.
- All candidates can be counted efficiently against a record by a tracing operation in one run. See subsection 4.4 for more details.

In this algorithm, we assume that the order is fixed. The order can be changed in each branch based on the frequencies of subgroups. Such a change would improve the efficiency of algorithm [9]. We have not employed such an optimisation in this implementation.

4.2. A running example

We use the following running example to explain the algorithm and to demonstrate the effectiveness of pruning.

Example 1 Values in Table 2 are itemised. This is a small data set, and we mainly used it to show how Criteria 1 and 2 work.

Frequent itemsets	Closed frequent itemsets	Itemsets searched
a, b, c, d, ab, ac, ad, bc, bd, cd, abc, abd, acd, bcd abcd	a, b, c, d, ac, ad, bc, bd, cd abc, bcd abcd	a, b, c, d bc, bd, cd.

Table 3: The sets of frequent itemsets, closed frequent itemsets and itemsets searched by the algorithm. Itemsets are descriptors of subgroups.

Let the count of the minimum support be 1. Let 'y' be the value of interest. To avoid the division by zero, we set a count as 0.5 when it is 0. β is also set as 0.5 (equivalent to a 0 count) for computing the upper bound of the left bounds of the odds ratios of all the sub subgroups of a subgroup. The confidence level is set as 68%, corresponding to the critical value of 1 because the size of the data set is small.

Itemsets are descriptors of subgroups and represent subgroups. The sets of frequent itemsets, frequent closed itemsets and the itemsets searched by the algorithm are listed in Table 3. The proposed algorithm does not search for any Level 3 and 4 itemsets. Super itemsets of a and bc are pruned by Criterion 1, and super itemsets of cd are pruned Criterion 2. Therefore, no Level 3 candidates will be generated as shown in Function 1.

Since sub subgroups of both a and bc are pruned by the same criterion, we use bc as an example to show how they are pruned. The contingency table associated with subgroup bc is $[3, 1; 2, 4]$. $\mathrm{OR}(bc) = 6$ corresponding to the confidence interval $[1.4, 25]$. The confidence interval is quite large since the data set is very small. The significance test penalises subgroups of small supports.

The contingency table associated with the upper bound of all the sub subgroups of subgroup bc is $[3, 0.5; 2, 5]$. The upper bound of the left bounds of the odds ratios of all sub subgroups of bc is $\mathrm{OR}(bc)^{ub}_{-} = 2.6$. Since $\mathrm{OR}(bc)^{ub}_{-} < \mathrm{OR}(bc)_{+}$, all sub subgroups of bc are statistically redundant and they should pruned. Therefore, node bc is removed.

The contingency tables of subgroups c and cd are $[4, 2; 1, 3]$ and $[2, 2; 3, 3]$ respectively. The difference of Cells n_{12} of the two contingency tables is zero, and hence $y = 0$. According to Criterion 2, all sub subgroups of cd and itself are pruned.

We will return to this example after we present the complete algorithm.

4.3. Candidate generation and first pruning

Candidate generation and pruning in the prefix tree are presented in the following function. l-unordered subsets of set P are subsets of set P with the cardinality of l.

Function 1 (NewLevel (prefix tree))
Input: prefix tree T built and pruned to Level k.
Output: prefix tree T with $k+1$ level.

1: **for** each node W at Level $k-1$ **do**
2: **for** each pair of child nodes U and V under W **do**
3: union the sets of U and V to form a new set P
4: check if all unordered l-subsets (unordered) of the set P are stored in T
5: **if** any unordered subset is not stored in T **then**
6: skip the current pair and continue to the next pair
7: **end if**
8: store set P in a child of U (or V) which stores the prefix set of the set P
9: record backtrack links to nodes storing the l-unordered subsets of set P
10: **end for**
11: **end for**

We use the example in Figure 2 to elaborate the function. Let $W = a$ be a node in Level 1. Nodes ab, ac and ad are child nodes of node W. Let $U = ab$ and $V = ac$. The union of U and V form a new candidate $P = abc$ in Level 3 of the prefix tree. These have been done by Lines 2 and 3. Note that we do not try the union of ab and bc, which are from different parent nodes. The candidates by combining non-sibling nodes are redundant. For example, the union of ab and bc is redundant to the combination of ab and ac.

The validity of a new candidate is determined by whether all its super subgroups are in the prefix tree. For candidate abc, we will need to check if nodes ab, ac and bc exist. Since abc is formed by ab and ac, they do exist. The only unchecked super subgroup is bc. This check is done by Line 4. Assume that node bc exists. abc is added as a descendant node of ab, and this is done by Line 8. Assume that node bc does not exist. Candidate abc is discarded and other sibling nodes are tested, such as, ab and ad. For the ease of accessing the super subgroups of a candidate subgroup, the backtrack links are added to super subgroups of a subgroup. This has been done by Line 9.

Note that the function will ensure that no sub subgroups will be generated if a subgroup is removed. In Example 1, node a is pruned by Criterion 1. The whole

branch rooted by a will not generated in the prefix tree. In other words, all sub subgroups of subgroup a are removed.

4.4. Support counting

Counting by the prefix tree is efficient since descriptors of subgroups indicate pathes to nodes storing them. All candidates at one level can be counted against a record in one run. Count is a recursive function.

Function 2 (SupportCount (prefix tree, dataset))
Input: Prefix tree T with Level $k+1$ uncounted, and data set D.
Output: Prefix tree T with Level $k+1$ counted.

for each record r in D **do**
  Count (T, r)
end for

Count (T', r')

1: let p be the label of the root of T'
2: **if** $p \notin r'$ or p is not the root of T **then**
3: return
4: **end if**
5: let P be the subgroup stored at the root of T'
6: **if** the length of the descriptor of $P == (k+1)$ **then**
7: increment the count of P by 1
8: return
9: **end if**
10: remove items preceding p and p from r'
11: **if** $r' == \emptyset$ **then**
12: return
13: **end if**
14: **for** each child node Y of node P **do**
15: let T_Y be the subtree rooted by node Y
16: Count(T_Y, r')
17: **end for**

We use an example to show how the function works. Suppose that we count the tree in Figure 2 against record $\{b, c, d\}$ for subgroups stored at Level 2 (and imagine that Levels 3 and 4 do not exist). Now, $k = 1$.

Count is a recursive function. It is firstly called by Count (the root of T, $\{b, c, d\}$). Then four subroutines will be called by following Lines 14-17. They

are: (I) Count (node a under the root, $\{b, c, d\}$), (II) Count (node b under the root, $\{b, c, d\}$), (III) Count (node c under the root, $\{b, c, d\}$), and (IV) Count (node d under the root, $\{b, c, d\}$). We use Subroutines (I), (II), and (IV) to further show how the function works.

Subroutine (I) Count (node a under the root, $\{b, c, d\}$) will be terminated by Lines 2-4 since a is not in record $\{b, c, d\}$. Therefore, the whole branch under node a will be bypassed and this is a reason for the efficiency.

The following two subroutines will be called by Subroutine (II): (A) Count (node c under node b, $\{c, d\}$), (B) Count (node d under node b , $\{c, d\}$). Note that item b is removed from record $\{b, c, d\}$ by Line 10 in Subroutine II. Again we only explain subroutine (A) Count (node c under node b, $\{c, d\}$) since both Subroutines (A) and (B) work in the same way.

In Lines 6-9 of subroutine (A) Count (node c under node b, $\{c, d\}$), the count of subgroup bc will be incremented by 1 and the subroutine will be terminated. Similarly subgroups bd and cd will be counted.

Subroutine (IV) Count (node d under the root, $\{b, c, d\}$) is terminated at Lines 11-13. After removing item d and items preceding item d, namely b and c from record $r' = \{b, c, d\}$. Record $r' = \emptyset$, and the subroutine is terminated.

Count function does not search for the whole prefix tree, and only searches for parts of the prefix tree storing subgroups which descriptors are subsets of the record. For example, in the above example, the branch under node a have not been searched.

4.5. Pruning and subgroup selection

Pruning is implemented by Criteria 1 and 2 and the frequency requirement.

Function 3 (Pruning&Selection (prefix tree, subgroup list))
Input: Prefix tree T with Level $k+1$ counted. A list of statistically non-redundant subgroups L
Output: Prefix tree T with Level $k+1$ pruned. Subgroup list L with new statistically non-redundant subgroups at Level $k+1$ added

1: **for** each node P at Level $k+1$ **do**
2: **if** the support of subgroup $P \leq$ the minimum support **then**
3: remove node P from tree T and continue to the next node
4: **end if**
5: compute $\mathrm{OR}(P)$, $\mathrm{OR}(P)_+$, $\mathrm{OR}(P)_-$ and $\mathrm{OR}(P)_-^{ub}$
6: **if** $(\mathrm{OR}(P)_- - 1) > 0$ and P is statistically non-redundant **then**
7: add P to L

8: **if** $\mathrm{OR}(P)_+ > \mathrm{OR}(P)^{ub}_-$ **then**
9: remove node P from tree T and continue to the next node
10: **end if**
11: **end if**
12: **for** each parent node R of P at Level k **do**
13: compute change y in cell n_{12} from R to P
14: **if** $y == 0$ **then**
15: remove node P from tree T and continue to the next node
16: **end if**
17: **end for**
18: **end for**

The first pruning process is based on the frequency requirement. If a subgroup is infrequent, so are all its sub subgroups and hence they are all pruned. This has been implemented by Lines 2-4. In Example 1, no node is pruned by the frequency requirement. However, if a node is pruned, none of its sub subgroups will be generated in the prefix tree following the candidate generation function.

The second pruning process is based on Criterion 1. All statistically redundant subgroups judged by the criterion will be pruned by Lines 8-10. In Example 1, both nodes a and bc are pruned by this criterion. The pruned statistically redundant subgroups have the same anti-monotonic property as infrequent subgroups. For example, the whole branch under node a will be pruned.

The third pruning process is based on Criterion 2 in Lines 12-16. In Example 1, node cd is pruned in this way. Again, the pruned statistically redundant subgroups follow the same anti-monotonic property as infrequent subgroups. No sub subgroups of cd will be generated or tested.

All statistically non-redundant subgroups are added to the subgroup list according to Definition 1 in Lines 6-7. Backtrack links are used in Line 12 to fetch parent nodes of P.

4.6. The overall algorithm

The overall algorithm is presented in Algorithm 1, which firstly transfers attribute value data to transactional data with items and then initiates a prefix tree. It subsequently calls Functions 1, 2, and 3 in a loop until there is no new candidate in the prefix tree.

We run the algorithm on the data set in Example 1. In the first level, nodes a, b, c, d are generated by Line 4. In Line 5, subgroup a is tested as a statistically non-redundant subgroup and is added to L. Subsequently, a is pruned by Lines

Algorithm 1 Statistically Non-redundant Subgroup discovery (SNS)

Input: data set D, the minimum support, significance critical value.
Output: a list of all statistically significant subgroups L

1: let $L = \emptyset$
2: code attribute values in D to items
3: count support of items
4: initiate prefix tree T
5: Pruning&Selection (T)
6: let $k = 1$
7: NewLevel (T)
8: **while** $k + 1$ level of tree T is non-empty **do**
9: SupportCount (T, D)
10: Pruning&Selection (T)
11: $k = k + 1$
12: NewLevel (T)
13: **end while**
14: output L

8-9 in Function Pruning&Selection. The remaining nodes are b, c and d.

In the second level, nodes bc, bd and cd are generated by Line 7. Their frequencies will be counted by Line 9. In Line 10, bc is tested as a statistically non-redundant subgroup and is added to L. Subsequently, node bc is pruned by Lines 8-9 in Function Pruning&Selection. Node cd is pruned by Lines 14-15 in Function Pruning&Selection. The remaining node is bd.

No node will be generated in level 3 by Line 12. The program is terminated and $L = \{a, bc\}$ is returned. A summary of generated candidates, statistically non-redundant subgroups and the pruned prefix tree is given in Figure 3.

Next, we discuss correctness of Algorithm 1.

Theorem 2 *Algorithm 1 produces the complete set of statistically non-redundant subgroups correctly.*

Proof Firstly, if there are no pruning processes, the algorithm will produce the complete prefix tree. This means that the algorithm will search for all candidates for statistically non-redundant subgroups.

Secondly, both Criteria 1 and 2 have been proved that they prune the search space that does not produce statistically non-redundant subgroups.

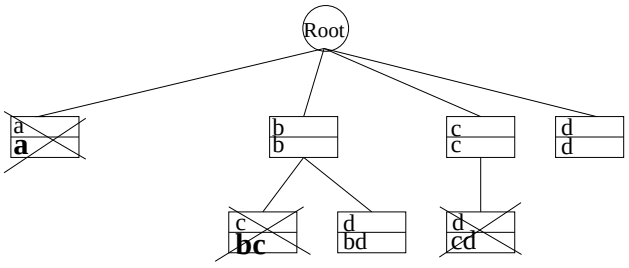

Figure 3: The pruned prefix tree in Example 1.The subgroups highlighted in bold are statistically non-redundant subgroups. Pruned candidates are crossed.

Therefore, the algorithm will produce the complete set of statistically non-redundant subgroups correctly. □

5. Experiments

In this section, we will demonstrate the efficiency of the proposed method in comparison with six classic and recent subgroup discovery methods. We will also point out a potential for relaxing pruning criteria for more efficient heuristic subgroup discovery.

5.1. Data

We employ ten frequently used data sets from UCIML repository [3]. Data sets Hypothyroid and Sick are in the directory of Thyroid Disease data set. Their size ranges from small to large, and their dimensionality vary from very small to medium. The numeric attributes have been removed in Adult and Census-Income data sets. Numeric attributes in other data sets have been discretized by using the discretization function of MLC++ [25]. For Census-Income data, the combined training and test data set D contains in total 299,285 records. We randomly sampled 50K from data set D without replacement and form the first sub data set D_1. We then randomly sampled 50K from the remaining records of D without replacement and combined the newly sampled data with the previous data set D_1 to form the second sub data set D_2. In this way, we formed five sub data sets D_1 to D_5 with the size of 50K, 100K, 150K, 200K and 250K respectively. These data sets were used for the scalability evaluation.

A description of data sets is listed in Table 4. In the experiments, we set the smaller class as the value of interest.

Name	#Records	#Attributes	Distributions
Adult	48842	8	23.9% & 76.1%
Breast cancer Wisconsin	699	9	34.5% & 65.5%
Census-Income	250000	33	6.2% & 93.8%
German Credit	1000	15	30% & 70%
Hypothyroid	3163	23	4.8% & 95.2%
kr-vs-kp	3196	36	47.8% & 52.2%
Mushroom	8124	22	48.2% & 51.8%
Sick	2800	26	6.1% & 93.9%
Tic-Tac-Toe	968	9	34.7% & 65.3%
Congressional voting	435	16	38.6% 61.4%

Table 4: A brief description of data sets used in experiments

The parameters of SNS algorithm is set as the following. β value is set as 5. The critical value of an odds ratio greater than 1 is set as 1.96 corresponding to 95% confidence level. The critical value for testing overlapping confidence intervals of odds ratios is set as 1. This is because that when we compare the lower bound of the odds ratio of a subgroup with the upper bound of the odds ratio of another subgroup, the uncertainty has been considered in both ends. Therefore, it is fair to use a smaller critical value than that for testing an odds ratio greater than 1. Other parameters are set differently to match comparison methods and are detailed in the following sections.

All experiments are performed on a computer with 4 core Intel i7-3370 CPU@ 3.4 GHz and 16 GB RAM. Our method is running on Ubuntu operating system and other comparison methods are running on 64-bit Windows operating system.

5.2. The efficiency of the algorithm

We compare the proposed method with six subgroup discovery methods for efficiency, including four classic methods and two recent non-redundant subgroup discovery methods. SD [16] is a beam search based rule induction method for subgroup discovery. CN2-SD [28] extends CN2 [11], a beam search based rule induction algorithm, for subgroup discovery. Apriori-SD [23] is an association rule mining based subgroup discovery algorithm, and its base algorithm is Apriori [2]. SD-Map [4] is another association rule mining based subgroup discovery method, and its base algorithm is FP-growth [21]. DSSD [29, 30] is a non-redundant subgroup discovery method based on the beam search strategy. ID-RSD [18] is a recent top-k relevant subgroup discovery method.

The parameter setting follows the setting in DSSD paper [29]. The count of the minimum support of a subgroup is 10 (except 100 for the Census data), and the maximum length of a subgroup is 5. For a top k method, the k is set as 10,000. k=10,000 looks large in a top k discovery method, but it is not large if we search for non-redundant subgroups. In work [29], 10,000 top subgroups are used to find 100 non-redundant subgroups since a small k may contain only one or two non-redundant subgroups. If the total number of subgroups is fewer than 10,000. A top k method is similar to an optimal discovery method, such as SNS, since both discover all the best subgroups (note that a beam search based method does not still). This is what we need for a fair comparison. In our demonstrations in the next section, top 10,000 subgroups do not include all statistically non-redundant subgroups in Hypothyroid data set. So k=10,000 is not large.

DSSD, ID-RSD and SD-Map programs were provided by authors. Other methods were implemented in Orange data mining software tool [1]. Other parameters use the default ones in the software tools.

The proposed method SNS is faster than other methods as shown in Table 5, which lists the execution time of all methods on ten data sets. In a large data set, the Census data, our method is the only one that can find results in two hour time frame. In a medium data set, the Adult data set, our method is significantly faster than all other methods. In kr-vs-kp data set, SD and DSSD are faster than SNS. Both SD and DSSD are beam search based methods. A beam search based method searches the candidate space using a fixed beam width, and it misses quality subgroups when the number of attributes is much larger than the beam width. This is a case in kr-vs-kp data set. SD finds 20 subgroups and DSSD finds 3 unrepeated non-redundant subgroups. In contrast, SNS finds 1638 subgroups with the same setting. Note that kr-vs-kp is a chess data set. There are many ways to win (and lose) a chess game, and a large number of subgroups is expected. This further shows that an optimal method is necessary for non-redundant subgroup discovery when it is computationally feasible.

To demonstrate that the SNS algorithm does not use a high memory usage to trade for a fast speed. We list the peak memory usage of the SNS and other algorithms in Table 5.2. We only list the data sets that all algorithms return results within two hours. The results show that the SNS uses a small amount of memory, and its memory usage is the second smallest on average in all algorithms. CN2-SD makes use the smallest memory on average but it is a heuristic algorithm and does not guarantee the completeness of discoveries. Note that the memory usage of Apriori-SD in data set Tic-tac is the smallest, but Apriori-SD does not return any subgroup in the data set. In contrast, SNS discovers 71 statistically non-redundant

	SNS	SD	CN2-SD	Apriori-SD	SD-Map	DSSD	ID-RSD
Adult	0	864	31.3	zero	38 m	35.7 m	138
BCW	0	3	20	4	0	28	0
Census	60.9m	-	-	zero	$>$ 2h	$>$ 2h	-
German	0	16	38	zero	21.9 m	39	67
Hypothyroid	3	32	217	-	22.1	226	10
Kr-vs-kp	145	34	304	-	83.1m	61	29.6m
Mushroom	1	50	12.4m	-	121.4m	298	9
Sick	14	25	255	-	30.3m	48	18
Tic-tac-toe	0	1	9	2	0	5	1
Voting	0	1	10	190	23m	14	1

Table 5: A comparison of execution time of different methods on ten data sets. The default time unit is in second. '-' indicates running out of memory. 'zero' indicates no subgroup returned. We terminated a program after 2 hours running.

Census	SNS	SD	CN2-SD	Apriori-SD	SD-Map	DSSD	ID-RSD
BCW	1.1	8.4	1.9	2.4	177.2	48.4	135.3
German	9.4	20.8	3.0	319.7	419.7	53.4	3858.6
Tic-tac-toe	2.1	11.6	1.7	1.5	237.9	31.9	1070.8
Voting	1.2	10.8	1.8	39.1	440.8	35.1	943.6

Table 6: A comparison of peak memory usage of the algorithms. Only data sets that all algorithms return results within two hours are listed.

subgroups in the data set with the same setting.

The scalability of the SNS with data size is shown in Figure 4. The parameters are set as before. Both running time and memory usage of the SNS scale well with the size of data sets. This is a reason that the SNS is the only algorithm running on the Census data set, the largest data set in all experimental data sets. However, SNS does not scale well with the the number of attributes since the number of candidates is ultimately exponentially increasing with the number of attributes. This is a common drawback for optimal subgroup (and rule) discovery methods.

SNS is designed for discovering all the statistically non-redundant subgroups constrained only by the minimum support and the maximum length of descriptors of subgroups. Other algorithms are designed for discovering top k subgroups which include statistically redundant subgroups, and they do not guarantee the completeness of the discoverers. For a fair comparison, k for a top k method should be large to ensure its discovery is compete or near complete. For example, top 10,000 non-redundant subgroups do not include all 29 statistically non-

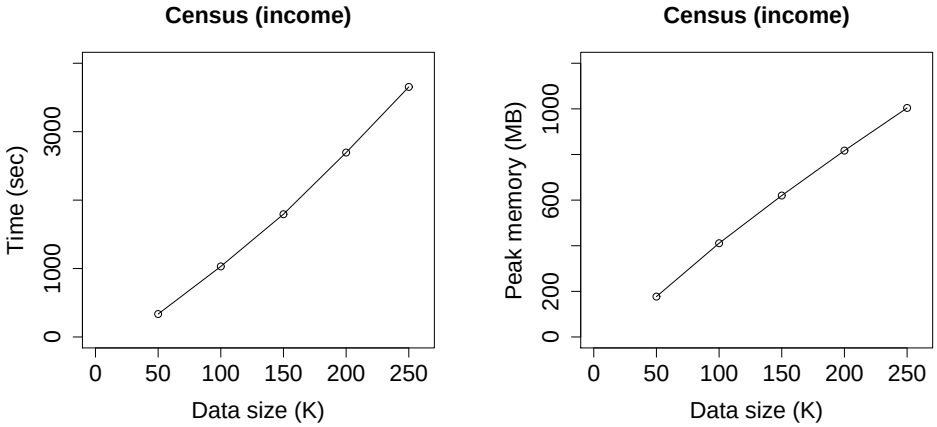

Figure 4: The scalability of the SNS algorithm with data size.

redundant subgroups as shown in the following section. A large k makes those algorithms inefficient. So the experimental results show that those methods are incapable of discovering statistically non-redundant subgroups. An optimal approach is necessary since there is a significant loss of the quality subgroups by a top k discovery method. SNS is an efficient and optimal algorithm to discover all statistically non-redundant subgroups.

5.3. Improvement over top k subgroup discovery

It is well known that an optimal search method produces many subgroups, and in some cases, too many. Top k search is a widely used strategy to avoid too many subgroups discovered. However, we will show that even if for a top k method, it is necessary to have a mechanism to remove statistically redundant subgroups to avoid significant loss of subgroups caused by that all top k subgroups come from one subsection of a data set.

Let us use the Hypothyroid data set with the minimum local support of 0.5 to show the above points. The minimum local support of 0.5 means that each subgroup will contain at least 50% samples in the disease group. This support is significant large and is not a cause for too many subgroups. The quality (or interestingness) measure is odds ratio. All top 258 discovered non-redundant subgroups (defined by the first type definition in the Related Work section by using minimal generators [38, 42]) do not include a single statistically non-redundant subgroup. All the subgroups are variations of the first statistically non-redundant subgroup which is ranked 259 in the list. The statistically non-redundant subgroup is "TSH

ID	Subgroups	Size	OR	OR_-	OR_+
1	age $>$ 71.5	477	3.99	3.39	4.71
2	TT4 $<$ 87.5	641	3.88	3.30	4.55
3	T4U $<$ 0.895	715	8.04	6.76	9.57
4	1 & 2	107	9.14	7.30	11.44
5	1 & 2 & 3	60	18.43	14.04	24.20

Table 7: Examples of sub and super subgroups in discovered statistically non-redundant subgroups

$>$ 27.5". Other statistically redundant subgroups are like "TSH$>$27.5, sick=f", "TSH$>$27.5, sick=f, pregnant=f", "TSH$>$27.5, sick=f, pregnant=f, thyroid_surgery=f", and so on, which have marginally larger odds ratios than that of "TSH$>$27.5". When $k = 500$, only 1 statistically non-redundant subgroup is included. When $k = 1000$, 5 statistically non-redundant subgroups are included. In other words, 24 (83%) statistically non-redundant subgroups are missed. When $k = 47762$, all (29) statistically non-redundant subgroups are discovered. At this point, post pruning is necessary to find interesting subgroups.

A remedy to the above problem is to find the top k largest subgroups (with the highest support) satisfying users constraints. In the implementation, when a subgroup is discovered to satisfy the user's constraints, we do not search for any of its more specific subgroups. As a result, many statistically redundant subgroups will be avoid. However, there are two problems associating with the solution. Firstly, users normally do not know how to set the right parameters for the constraints. A wrong setting means a significant loss of quality subgroups. Secondly, such a method will lose the subgroups with the highest quality since many subgroups with high quality are hid deeply in the search lattice. More than a half statistically non-redundant subgroups from the Hypothyroid data set have sub or super subgroups. Table 7 shows a few examples. A simple exclusion of more specific subgroups will lead to a significant loss of such subgroups.

5.4. A further exploration of pruning criteria

One interesting question is how much more reduction in the search space that we can achieve without losing the completeness of discoveries. Pruning criterion 2 requires $y = 0$, or the change in cell n_{12} to be zero in a subgroup specialisation process. As an approximation, we may set $y \leq \delta$ where δ is a small number instead of 0. In other words, we stop searching for sub subgroups of the current subgroup (P) if the change in cell n_{12} is very small in the specialisation process from a super subgroup of P to P. We should assess the results of this change in

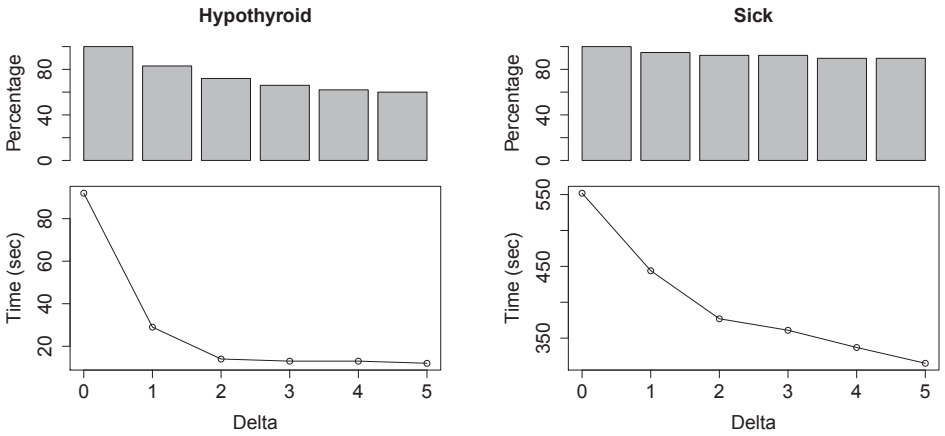

Figure 5: The completeness drop versus the running time improvement with different δ. Bottom graphs show the running time reduction. Top graphs show the percentage of retained statistically non-redundant subgroups.

this experiment.

Figure 5 lists the reduction of the running time of the implemented statistically non-redundant subgroup discovery method with different δ and the retention rate of statistically non-redundant subgroups in comparison with those with $\delta = 0$. The retention rate is the ratio of the number of subgroups discovered to the total number of subgroups that should be discovered.

Any non-zero δ results in a loss in the discovery of statistically non-redundant subgroups. This shows that the pruning criterion is very tight already. However, if we can tolerate a certain level of incompleteness in the discoveries, the time efficiency can be improved significantly. For example, when $\delta = 2$, 85% reduction in running time for the Hypothyroid data set and 32% reduction for the Sick data set, whereas the loss of statistically non-redundant subgroups is 28% and 7.7% respectively.

We note the dual speeds in the reduction of the running time, and of the completeness of the discovered statistically non-redundant subgroups in Figure 5. The reduction in running time is significantly faster than the reduction of the completeness of the discoveries. This makes a good tradeoff for a big time efficiency improvement with a small loss in the discoveries.

6. Conclusion

Most existing subgroup discovering methods use some intuitive measures to capture statistical difference of a subgroup with the other, and they do not exclude statistically redundant subgroups in their discoveries. Odds ratio is a statistical measure widely used in practice to measure statistical differences of two groups and is very suitable for measuring the quality of subgroups. In this paper, we have proposed a framework for efficient non-redundant subgroup discovery by using the odds ratio to define subgroups and the confidence intervals of odds ratios to define statistically non-redundant subgroups. We have studied criteria for pruning redundant subgroups, and proposed an algorithm for efficient statistically non-redundant subgroup discovery. We show that the proposed algorithm is faster than six classic and recent subgroup discovery algorithms, especially for large data sets. We demonstrate that a top k subgroup discovery method needs pruning statistically redundant subgroups to avoid the loss of quality subgroups discovered. We also show that a proposed pruning criterion is very tight in term of pruning the search space while keeping the completeness of discoveries.

As an optimal subgroup discovery algorithm, the proposed method is very efficient and scales well with data size. However, it does not scale well with dimensions. The number of candidate subgroups increases exponentially with the total number of values in attributes and the large number of candidate subgroups reduce the speed and consume memory quickly. A solution is by feature selection. It is our future work to study suitable feature selection methods for subgroup discovery.

Acknowledgements

This work has been supported by Australian Research Council Discovery Grant DP130104090, the Academy of Finland grant 118653 for the Finnish Center of Excellence for Algorithmic Data Analysis Research (Algodan), Nokia Foundation, and Japan Society of Promotion of Science. We thank authors of papers [4, 18, 29, 30] for kindly providing their implementations for the comparison. We also thank developers of Orange Data Mining for their open source software tools. We appreciate constructive comments of anonymous reviewers for improving the quality and presentation of this paper.

References

[1] http://orange.biolab.si/, Access 1 April 2013.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A.I. Verkamo, Fast discovery of association rules, in: Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996, pp. 307–328.

[3] A. Asuncion, D. Newman, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/, 2007.

[4] M. Atzmueller, F. Puppe, SD-Map: a fast algorithm for exhaustive subgroup discovery, in: Proceedings of European Conference on Principle and Practice of Knowledge Discovery in Databases (PKDD), Springer-Verlag, Berlin, Heidelberg, 2006, pp. 6–17.

[5] J. Baumeister, M. Atzmueller, F. Puppe, Introspective subgroup analysis for interactive knowledge refinement, in: International Florida Artificial Intelligence Research Society Conference (FLAIRS), AAAI Press, 2006, pp. 402–407.

[6] S.D. Bay, M.J. Pazzani, Detecting group differences: Mining contrast sets, Journal of Data Mining and Knowledge Discover 5 (2001) 213–246.

[7] R. Bayardo, R. Agrawal, D. Gunopulos, Constraint-based rule mining in large, dense database, Data Mining and Knowledge Discovery Journal 4(2/3) (2000) 217–240.

[8] M. Boley, H. Grosskreutz, Non-redundant subgroup discovery using a closure system, in: Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD), 2009, pp. 179–194.

[9] C. Borgelt, Efficient implementations of Apriori and Eclat, in: Proceedings of IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI), 2003, pp. 24–32.

[10] S. Brin, R. Motwani, C. Silverstein, Beyond market baskets: generalizing association rules to correlations, SIGMOD Record (ACM Special Interest Group on Management of Data), 26 2 (1997) 265–276.

[11] P. Clark, T. Niblett, The CN2 induction algorithm, Machine Learning 3 (1989) 261–283.

[12] G. Dong, J. Li, Efficient mining of emerging patterns: discovering trends and differences, in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 1999, pp. 43–52.

[13] W. Duivesteijn, A.J. Knobbe, Exploiting false discoveries - statistical validation of patterns and quality measures in subgroup discovery, in: Proceedings of IEEE International Conference on Data Mining (ICDM), 2011, pp. 151–160.

[14] W. Duivesteijn, A.J. Knobbe, A. Feelders, M. van Leeuwen, Subgroup discovery meets bayesian networks – an exceptional model mining approach, in: Proceedings of IEEE International Conference on Data Mining (ICDM), 2010, pp. 158–167.

[15] J.L. Fleiss, B. Levin, M.C. Paik, Statistical Methods for Rates and Proportions, 3rd ed., Wiley, 2003.

[16] D. Gamberger, N. Lavrac, Expert-guided subgroup discovery: methodology and application, Journal of Artificial Intelligence Research 17 (2002) 501–527.

[17] H. Grosskreutz, M. Boley, M. Krause-Traudes, Subgroup discovery for election analysis: a case study in descriptive data mining, in: Proceedings of the International Conference on Discovery Science (DS), 2010, pp. 57–71.

[18] H. Grosskreutz, D. Paurat, Fast and memory-efficient discovery of the top-k relevant subgroups in a reduced candidate space, in: Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I (PKDD), 2011, pp. 533–548.

[19] H. Grosskreutz, S. Rüping, On subgroup discovery in numerical domains, Data Mining and Knowledge Discovery 19 (2009) 210–226.

[20] H. Grosskreutz, S. Rüping, S. Wrobel, Tight optimistic estimates for fast subgroup discovery, in: Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD), 2008, pp. 440–456.

[21] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: A frequent-pattern tree approach, Data Mining and Knowledge Discovery 8(1) (2004) 53–87.

[22] F. Herrera, C. Carmona, P. González, M. Jesus, An overview on subgroup discovery: foundations and applications, Knowledge and Information Systems 29 (2010) 495–525.

[23] B. Kavšek, N. Lavrač, Apriori-sd: Adapting association rule learning to subgroup discovery, Applied Artificial Intelligence 20 (2006) 543–583.

[24] W. Klösgen, Explora: a multipattern and multistrategy discovery assistant, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, 1996, pp. 249–271.

[25] R. Kohavi, D. Sommerfield, J. Dougherty, Data mining using MLC++: A machine learning library in C++, in: Tools with Artificial Intelligence, 1996, pp. 234–245.

[26] R.M. Konijn, W. Duivesteijn, W. Kowalczyk, A.J. Knobbe, Discovering local subgroups, with an application to fraud detection, in: Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) Part I, 2013, pp. 1–12.

[27] P. Kralj, N. Lavrač, D. Gamberger, A. Krstačic, Contrast set mining through subgroup discovery applied to brain ischaemina data, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2007, pp. 579–586.

[28] N. Lavrač, B. Kavšek, P. Flach, L. Todorovski, Subgroup discovery with CN2-SD, Journal of Machine Learning Research 5 (2004) 153–188.

[29] M. van Leeuwen, A. Knobbe, Non-redundant subgroup discovery in large and complex data, in: Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD), 2011, 459–474.

[30] M. van Leeuwen, A.J. Knobbe, Diverse subgroup set discovery, Data Mining and Knowledge Discovery 25 (2012) 208–242.

[31] H. Li, J. Li, L. Wong, M. Feng, Y.P. Tan, Relative risk and odds ratio: a data mining perspective, in: Proceedings of ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), 2005, pp. 368–377.

[32] J. Li, On optimal rule discovery, IEEE Transactions on Knowledge and Data Engineering 18(4) (2006) 460 – 471.

[33] M. Mampaey, S. Nijssen, A. Feelders, A.J. Knobbe, Efficient algorithms for finding richer subgroup descriptions in numeric and nominal data, in: Proceedings of IEEE International Conference on Data Mining (ICDM), 2012, pp. 499–508.

[34] G.A. Morgan, N.L. Leech, G.W. Gloeckner, K.C. Barrett, SPSS for Introductory Statistics: Use and Interpretation, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 2006.

[35] S. Morishita, J. Sese, Transversing itemset lattices with statistical metric pruning, in: Proceedings of ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2000, pp. 226–236.

[36] S. Nijssen, T. Guns, L. De Raedt, Correlated itemset mining in roc space: a constraint programming approach, in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2009, pp. 647–656.

[37] P.K. Novak, N. Lavrač, G.I. Webb, Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining, Journal of Machine Learning Research 10 (2009) 377–403.

[38] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Discovering frequent closed itemsets for association rules, in: Proceedings of International Conference on Database Theory (ICDT), 1999, pp. 398–416.

[39] G.I. Webb, Discovering significant patterns, Machine Learning 71 (2008) 1–31.

[40] G.I. Webb, S. Butler, D. Newlands, On detecting differences between groups, in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2003, pp. 256–265.

[41] S. Wrobel, An algorithm for multi-relational discovery of subgroups, in: Proceedings of European Conference on Principles of Data Mining and Knowledge Discovery (PKDD), 1997, pp. 78 – 87.

[42] M.J. Zaki, Mining non-redundant association rules, Data Mining and Knowledge Discovery 9 (2004) 223–248.