Finding Low-Entropy Sets and Trees from Binary Data

Hannes Heikinheimo
HIIT, CIS Lab, Helsinki
University of Technology

Eino Hinkkanen
HIIT, Dept. Computer Science,
University of Helsinki

Heikki Mannila
HIIT, Dept. Computer Science,
University of Helsinki

Taneli Mielikäinen
HIIT, Dept. Computer Science,
Univ. Helsinki and Nokia
Research Center Palo Alto

Jouni K. Seppänen
HIIT, CIS Lab, Helsinki
University of Technology

ABSTRACT

The discovery of subsets with special properties from binary data has been one of the key themes in pattern discovery. Pattern classes such as frequent itemsets stress the co-occurrence of the value 1 in the data. While this choice makes sense in the context of sparse binary data, it disregards potentially interesting subsets of attributes that have some other type of dependency structure.

We consider the problem of finding all subsets of attributes that have low complexity. The complexity is measured by either the entropy of the projection of the data on the subset, or the entropy of the data for the subset when modeled using a Bayesian tree, with downward or upward pointing edges. We show that the entropy measure on sets has a monotonicity property, and thus a levelwise approach can find all low-entropy itemsets. We also show that the tree-based measures are bounded above by the entropy of the corresponding itemset, allowing similar algorithms to be used for finding low-entropy trees. We describe algorithms for finding all subsets satisfying an entropy condition. We give an extensive empirical evaluation of the performance of the methods both on synthetic and on real data. We also discuss the search for high-entropy subsets and the computation of the Vapnik-Chervonenkis dimension of the data.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database applications—*data mining*

General Terms: Algorithms, Experimentation, Theory

Keywords: Local Models, Pattern Discovery

1. INTRODUCTION

Pattern discovery is the subarea of data mining concerned with finding combinations of variables that are in some sense locally interesting. The archetypical example is the pattern class of frequent itemsets [2], and it sometimes seems that this is the only such class that attracts significant research. Our aim in this paper is to chart one part of the space of more general pattern classes in binary data. Perhaps the most obvious way to generalize frequent itemsets is to allow not only combinations of variables that have the value 1 in the same records but also combinations whose distribution of values is significantly skewed in other ways. This leads naturally to examining the entropy of the variables: if the combination of variables has a skewed joint distribution, their joint entropy is low. Finding all low-entropy itemsets is easy because entropy is monotonic with respect to set inclusion: adding a variable to a set cannot decrease the entropy.

While the property of having low entropy is interesting in itself, it would often be beneficial to have an explanation of how the variables in the set are connected to each other. This leads us to examine two other pattern classes, which we call U-trees and D-trees (seen as Bayes networks, the trees have edges going up and down, respectively). U-trees are a subclass of tree-structured Bayes networks. We give an algorithm for mining U-trees and show by experiments that it is feasible for moderate-sized data. On the other hand, D-trees are also Bayes nets, but simpler than U-trees as every node has at most one parent. The best D-tree can be computed fast even for very large data sets. They have also been called Bayes trees, Markov trees [21], dependence trees [6], and Chow-Liu trees [18], but an important distinction is that we do not attempt to model the complete joint distribution but to find interesting local patterns. In other words, a D-tree is the Chow-Liu tree for some subset of the variables.

Thus there is a continuum of local pattern classes starting from frequent itemsets, which are inflexible but for which a multitude of fast mining algorithms has been developed. D-trees are more flexible and still fairly easy to find; U-trees are still more flexible, but our current algorithms are not as efficient as those for the less flexible classes. At the high end of flexibility are arbitrary low-entropy itemsets, which can capture any kind of interaction, and somewhat paradoxically can be found efficiently using an algorithm similar to frequent itemset mining algorithms. However, while the algorithm is efficient, the size of the output can be very large, which motivates the more direct algorithms for finding low-entropy trees. A drawback to low-entropy itemsets compared to the less flexible trees is that they are not as easily interpreted. From another viewpoint, there is a continuum from frequent itemsets via fragments of order [10] and the trees of [14] to U-trees, where each pattern class has more structure than the previous one.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'07, August 12–15, 2007, San Jose, California, USA.
Copyright 2007 ACM 978-1-59593-609-7/07/0008 ...$5.00.

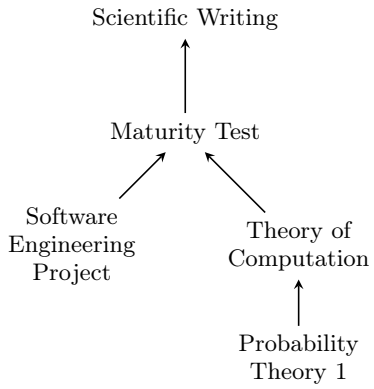

Figure 1: An example U-tree in course data.

Comp.	Prob.	
no	no	1887
yes	no	156
no	yes	143
yes	yes	219

Wri.	Mat.	
no	no	1636
yes	no	74
no	yes	14
yes	yes	681

Mat.	Eng.	Comp.	
no	no	no	1570
yes	no	no	79
no	yes	no	99
yes	yes	no	282
no	no	yes	28
yes	no	yes	164
no	yes	yes	13
yes	yes	yes	170

Table 1: The marginal distributions related to the tree in Figure 1. Legend: Comp. = Theory of Computation; Prob. = Probability Theory; Wri. = Scientific Writing; Mat. = Maturity Test.

Figure 1 shows an example U-tree found from course enrolment data at the University of Helsinki Department of Computer Science. The tree is the lowest-entropy 5-node U-tree found in one of the experiments described later. All courses in the tree are required for graduating, and the tree shows that students have completed different combinations of them. Table 1 shows the marginal distributions whose entropies are embodied in the tree. We see that the two theoretical courses are correlated quite strongly with each other and less strongly with the other courses. Scientific Writing and Maturity Test[1] are even more strongly correlated, which makes sense since both are related to the final phase of writing the Bachelor's thesis. Software Engineering Project is a demanding practical course, which theoretically minded students may leave for last, while practical students often postpone taking the theoretical courses perceived as difficult.

Related to low-entropy itemsets are of course high-entropy itemsets, and itemsets having high Vapnik-Chervonenkis dimension; we discuss these pattern classes briefly. Since the entropy function is decreasing with respect to itemset inclusion, high-entropy itemsets cannot be found with the same approach as low-entropy ones, and in fact the highest-entropy itemset is always the one containing all items. Thus we introduce two kinds of scaling for entropy, defining an itemset to be interesting either if its entropy is high among all itemsets of the same size, or if its entropy is high when compared to the entropies of its singleton subsets. We show that these properties are weakly monotonic: if an itemset X has high scaled entropy, then it has at least one subset $X \setminus \{ A \}$ having even higher scaled entropy. This again allows a levelwise algorithm to be used, albeit with less efficient pruning.

The rest of this paper is structured as follows. We introduce entropy and discuss finding low-entropy sets in Section 2. Then we move to low-entropy trees in Section 3, and address high-entropy sets and sets of high Vapnik-Chervonenkis dimension in Section 4. Section 5 describes extensive experiments, Section 6 discusses related work, and Section 7 is a brief conclusion.

2. LOW-ENTROPY SETS

We start by defining some notation and terminology. A $0/1$ dataset $\mathcal{D}$ is an $n \times m$ binary matrix. The columns are often called items and the set of all items is denoted by $\mathcal{I}$. A set of items is called an itemset. We denote items by $A, B, \ldots$ and itemsets by $X, Y, Z, \ldots$. We omit the braces around singleton sets, e.g., we write A instead of $\{ A \}$. We also follow the convention in information theory of denoting set union by a comma within the parentheses of entropy and information functions; e.g., $H(X, Y)$ is the entropy of the random variable $X \cup Y$, and $I(X, Y; Z)$ is the mutual information between the two random variables $X \cup Y$ and Z.

We denote by Ω_X the set $\{ 0, 1 \}^{|X|}$ of all $0/1$ vectors of length $|X|$. For a vector $\vec{x} \in \Omega_X$, the probability $\mathbf{P}(X = \vec{x})$ is the fraction of rows whose projection onto X is equal to $\vec{x}$. We sometimes use the shorthand $\mathbf{P}(\vec{x})$ to denote $\mathbf{P}(X = \vec{x})$.

The entropy of an itemset X in $\mathcal{D}$ is

$$H(\mathcal{D}, X) = -\sum_{\vec{x} \in \Omega_X} \mathbf{P}(X = \vec{x}) \log \mathbf{P}(X = \vec{x}).$$

All logarithms are in base 2, and $0 \log 0 = 0$ by convention. We denote $H(X) = H(\mathcal{D}, X)$, leaving out the $\mathcal{D}$ when it is clear from the context. Entropy can be seen as a measure of how surprising the outcome is likely to be when we select a random vector $\vec{x} \in \Omega_X$. Intuitively, entropy is increased by outcomes that are unlikely, because then $-\log \mathbf{P}(X = \vec{x})$ is high; but since these contributions are weighted by the probabilities, a single unlikely outcome contributes little, and entropy is only maximized when each outcome is unlikely. Indeed, a simple application of Lagrange multipliers shows that the entropy function $H(X)$ has its maximum value when all probabilities $\mathbf{P}(X = \vec{x})$ are equal; in this case the entropy is $\log |\Omega_X| = |X|$.

Entropy is small when the most outcome are not surprising. The minimum value $H(X) = 0$ is achieved when there is one value $\vec{x}_1$ that is always the outcome, i.e., when $\mathbf{P}(X = \vec{x}_1) = 1$. This can be contrasted with frequent itemsets: an itemset X has maximal frequency if $\mathbf{P}(X = \vec{1}) = 1$. Choosing $\vec{x}_1 = \vec{1}$ shows that then entropy is also minimized. More generally, high frequency implies low entropy, but not neces-

[1]The Maturity Test is required in the Finnish system to complete a Bachelor's degree after submitting the thesis; the candidate has to answer questions regarding the subject of the thesis in his or her mother tongue.

sarily vice versa. This motivates the following generalization of frequent itemsets:

DEFINITION 1. *Given an entropy threshold ϵ, an itemset X is a low-entropy set in $\mathcal{D}$ if $H(X) \leq \epsilon$.*

Low-entropy sets have a monotonicity property like that of frequent itemsets, which allows us to use e.g. the levelwise search [3] to find low-entropy sets.

PROPOSITION 2. *For all datasets $\mathcal{D}$,*

$$H(X) \geq H(X \setminus A)$$

for all $A \in X$ and $X \neq \emptyset$.

PROOF. $H(X) = H(X \setminus A) + H(A \mid X \setminus A) \geq H(X \setminus A)$ with equality holding only when $H(A \mid X \setminus A) = 0$, i.e., when there is a functional dependency $X \setminus A \to A$. □

In the proof, we used the concept of conditional entropy: $H(X \mid Y)$ is defined as $H(X, Y) - H(Y)$, and it measures the uncertainty about X when Y is known. We used the fact that conditional entropy is always nonnegative. For the proof of this fact and other basic properties of entropy, the reader is referred to e.g. [8, Chapter 2].

Proposition 2 implies that low-entropy sets can be found by using a levelwise approach. One seeming complication in mining low-entropy sets is that the entropy $H(X)$ is a sum over all possible combinations of values that the attributes in X can take, which involves $2^{|X|}$ different values. However, because of the convention $0 \log 0 = 0$ used in the definition of entropy, we only need to count those value combinations that appear in the data. Thus the database pass of the levelwise algorithm can be done using $O(c\ell n \log n)$ auxiliary space, where c is the number of candidate itemsets being examined, ℓ is the size of the candidates, and n is the number of records in the data. For each candidate X, we keep track of the combinations seen and a count for each combination; there are at most n combinations, each requiring space $O(\ell)$, and each count takes $O(\log n)$ space. Further savings can be obtained by storing the combinations in a binary tree. If the data fits into main memory, then the entropy computation can easily be performed individually for each itemset.

3. LOW-ENTROPY TREES

A potential drawback of mining low-entropy itemsets is that the sets do not have any structure that would help in interpretation. In this section we consider replacing itemsets by two kinds of tree patterns; the patterns impose a model on the attribute sets, and it is the entropy of the model that we seek to minimize. We call the tree patterns D- and U-trees; D-trees can be seen as tree-structured Bayesian networks where the edges are directed away from the root, i.e., down in the usual way of drawing trees in computer science. Correspondingly, in U-trees the edges are directed up, towards the root.

Definitions.

Both kinds of trees are formed by imposing a tree structure on some set of attributes, and the differences lie in the semantics of the structure, i.e., how the entropy of the tree is defined. Formally, a tree $T = \langle A, T_1, \ldots, T_k \rangle$ consists of a root attribute $A \in \mathcal{I}$ and $k \geq 0$ subtrees $T_1, \ldots, T_k$, each of which is a tree. We will only be interested in trees in which no attribute appears more than once.

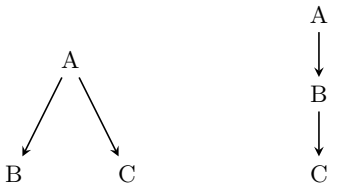

Figure 2: Two example D-trees

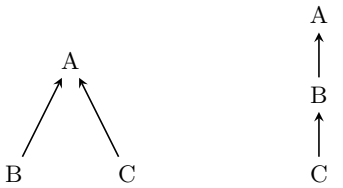

Figure 3: Two example U-trees

We define the entropy of a D-tree $T = \langle A, T_1, \ldots, T_k \rangle$ as

$$H_D(T) = H(A \mid \text{parent}(T)) + \sum_{j=1}^{k} H_D(T_j),$$

where parent(T) denotes the attribute at the root of the tree whose subtree T is. If T has no parent, we define $H(A \mid \text{parent}(T)) = H(A)$. Thus we start summing entropies from the root, and for child nodes always condition the entropy on the parent node, but not on further ancestors. This implies that when we seek low-entropy D-trees, we will prefer trees where each child node is as closely determined by its parent as possible. For example, consider Figure 2. The left-hand tree has entropy

$$H(A) + H(B \mid A) + H(C \mid A)$$

and the right hand tree has entropy

$$H(A) + H(B \mid A) + H(C \mid B).$$

Therefore the left-hand tree is preferred if knowing A reduces one's uncertainty about C more than does knowing B, and vice versa. A D-tree can be viewed as a Bayes network in which the edges are directed away from the root node.

DEFINITION 3. *Given an entropy threshold ϵ, a tree T is a low-entropy D-tree in $\mathcal{D}$ if $H_D(T) \leq \epsilon$. The set of low-entropy D-trees is denoted by $\mathcal{TP}_D(\mathcal{D}, \epsilon)$.*

For a U-tree $T = \langle A, T_1, \ldots, T_k \rangle$, we define the entropy as follows. Let $T_j = \langle A_j, \ldots \rangle$, i.e., denote by A_j the attribute at the root of T_j. Then

$$H_U(T) = H(A \mid A_1, \ldots, A_k) + \sum_{j=1}^{k} H_U(T_j).$$

That is, each node that has children contributes the entropy of the root attribute conditioned on (the joint distribution of) all the child attributes. Leaf nodes contribute the unconditional entropy (i.e., $H(A \mid \emptyset) = H(A)$).

As an example of U-trees, consider Figure 3. The left-hand tree has entropy $H(A \mid B, C) + H(B) + H(C)$ and the

right hand tree has entropy $H(A \mid B) + H(B \mid C) + H(C)$. The difference between the entropies,

$$(H(A \mid B, C) - H(A \mid B)) + (H(B) - H(B \mid C)),$$

is positive or negative depending on whether adding C as a condition in $H(A \mid B)$ or $H(B)$ gives more information.

In the same way that low-entropy itemsets give little information about the interconnections between items, U-trees where the nodes have many children pointing to them do not really explain the interconnections: they simply say that $H(X \mid A_1, \ldots, A_k)$ is small for a possibly large set of child attributes $A_1, \ldots, A_k$. Therefore we add a parameter β to control the branching of the tree.

DEFINITION 4. *Given an entropy threshold ϵ and a branching limit β, a tree T is a low-entropy U-tree in $\mathcal{D}$ if $H_U(T) \leq \epsilon$ and each node in T has at most β child nodes. The set of all low-entropy U-trees in $\mathcal{D}$ is denoted by $\mathcal{TP}_U(\mathcal{D}, \epsilon, \beta)$.*

Properties.

The following result shows that the tree-structured patterns are indeed less flexible than the arbitrary low-entropy itemsets of Section 2 in the sense that any low-entropy tree is necessarily a low-entropy itemset.

PROPOSITION 5. *Let X be an itemset. If T is a D-tree for X, then $H(X) \leq H_D(T)$. If T is a U-tree for X, then $H(X) \leq H_U(T)$.*

PROOF. The tree T imposes a partial order on the variables in X: $A \prec B$ if there is a directed path from A to B in T. List the elements of X in any linearization of this partial order: $X = \{A_1, A_2, \ldots, A_n\}$, where $i < j$ whenever $A_i \prec A_j$. Denote by $\mathrm{pred}(A_j)$ the set of attributes in X from which there is an arc to A_j. Now we can write

$$\begin{aligned} H(X) = {} & H(A_1) + H(A_2 \mid A_1) + \ldots \\ & + H(A_n \mid A_1, A_2, \ldots, A_{n-1}) \end{aligned} \tag{1}$$

and

$$\begin{aligned} H_{D/U}(T) = {} & H(A_1) + H(A_2 \mid \mathrm{pred}(A_2)) + \ldots \\ & + H(A_n \mid \mathrm{pred}(A_n)). \end{aligned} \tag{2}$$

Because $\mathrm{pred}(A_j) \subseteq \{A_1, \ldots, A_{j-1}\}$, every term on the right-hand side of (1) is less than or equal to the corresponding term on the right-hand side of (2). □

This proposition implies also that one possible way to mine low-entropy trees is to mine first the low-entropy itemsets using e.g. the levelwise search and then fit tree structures into each itemset.

Notice that in the proof of Proposition 5 we have equality if $\mathrm{pred}(A_j) = \{A_1, \ldots, A_{j-1}\}$. This is not possible for trees larger than two nodes, but it requires that the Bayesian network is fully connected.

COROLLARY 6. *Let G be a fully connected directed acyclic graph and G is interpreted as a Bayesian network whose every node has the maximum-likelihood distribution from the data $\mathcal{D}$. Then we have $H(X) = H(G)$.*

The root of a D-tree T can be selected arbitrarily, because $H_D(T)$ can be written as a sum depending only on the entropies of the edges and the nodes, and the degrees of the nodes: $\sum_{(A,B)\in E} H(A, B) - \sum_{A \in V} (deg(A, T) - 1)H(A)$.

PROPOSITION 7. *For D-trees the choice of root does not matter.*

However, it is easy to see that for U-trees the topology of the tree can make a large difference. For example, consider the case where we have three binary variables: two independent variables with marginal probabilities $1/2$, and their exclusive or. Then any U-tree with a root with two children has entropy 2 whereas any stick-shaped U-tree has entropy 3: a single attribute does not tell anything about the other two, but any two of the attributes determine the third one completely.

The above example also shows that the entropy of the best D-tree for the attribute set X can be larger than the entropy of the best U-tree for X.

CONJECTURE 8. *If T is a D-tree for X, then there is a U-tree S for X such that $H_D(T) \geq H_U(S)$.*

Algorithms.

We next turn to the question of how to mine D-trees and U-trees. As noted above Proposition 5, the entropy of the set gives a lower bound for the entropy of the best D- and U-trees. However, the lower bound can be rather loose.

In the case of D-trees we look for the best tree for the given set of attributes. It turns out that the best D-tree for an itemset X can be obtained by adding an edge to the best D-tree for $X \setminus A$ for some $A \in X$.

PROPOSITION 9. *Let $D(X) = (X, E)$ denote the minimum entropy D-tree for the itemset X. Then we have*

$$H_D(D(X)) \geq H_D(D(X \setminus v))$$

and $D(X \setminus v)$ is a subtree of $D(X)$ for some $v \in X$ and all $X \neq \emptyset$.

PROOF. Observe that $D(X \setminus v)$ is a subtree of $D(X)$ for

$$v = \operatorname*{argmax}_{v \in X} \{H(v \mid u) : \{v, u\} \in E, deg(v, D(X)) = 1\}$$

and hence the claim holds. □

Hence the low-entropy D-trees can be found by e.g. breadth-first search by setting

$$D(X) = \operatorname*{argmin}_{A \in X} \min_{B \in X \setminus A} H_D(D(X \setminus A)) + H(A \mid B).$$

There are $|\mathcal{I} \setminus X||X|$ possible extensions of $D(X)$ and each can be evaluated in constant time as there is no need to look at the data after computing the pairwise entropies of the items. Furthermore, the approach can be adapted to mine top-k D-trees instead of the D-trees with entropies below the given threshold.

Also in the case of U-trees, the entropy is defined as a sum of local conditional entropies. We proceed by finding first *low U-trees* where the leaves are immediate children of the root, and then using the low trees as building blocks. Low U-trees correspond closely to sets, as demonstrated by the following result.

PROPOSITION 10. *The entropy of a low U-tree on the attribute set X is at least as large as the entropy of X.*

PROOF. Let $A \in X$ be the root of the tree and $X' = X \setminus A$ the set of the leaves. The entropy of the tree is by definition

$$\sum_{B \in X'} H(B) + H(A \mid X') \geq H(X') + H(X) - H(X') = H(X).$$

□

To mine low U-trees, we simply use the levelwise algorithm to find all low-entropy sets, let each attribute be the root in turn, and list the U-trees in the order of increasing entropy.

We construct larger U-trees by connecting low-entropy low U-trees to already found low-entropy U-trees. In more detail, we maintain a priority queue of low-entropy U-trees. The queue is initialized by the low U-trees with sufficiently small entropy. Then the first tree is extended replacing a leaf by a low-entropy U-tree. A U-tree T and a low U-tree U may only be combined if the root of U is a leaf of T; then the entropy of the combined tree V is

$$H_U(V) = H_U(U) + H_U(T) - H(\text{root}(U)),$$

because in T the entropy of the subtree consisting of $\text{root}(U)$ is simply $H(\text{root}(U))$, but in V it is $H_U(U)$. If this entropy falls below the threshold, the combined tree is inserted in the list, and the process is continued until no more combinations have sufficiently low entropy.

PROPOSITION 11. *The U-tree mining algorithm described above finds all low-entropy U-trees.*

The algorithm lists the low-entropy U-trees in increasing order in entropy. So, adapting it for top-k mining of low-entropy U-trees is quite straightforward.

4. HIGH-ENTROPY SETS

So far we have considered itemsets and trees that have low entropy, i.e., whose values are highly concentrated on one or a few combinations. If, on the contrary, the values are spread out more than for most itemsets, this may also be an interesting pattern. Thus an obvious pattern class could be the itemsets whose entropies are highest.

Intuitively, a high-entropy set is a set of maximum diversity: a set of attributes whose value assignments shatter the data maximally. An example of such an itemset in the Course enrolment dataset is the set consisting of the courses Calculus I, Computer Uses in Education, Introduction to the Use of Computers, The metalanguage XML, and Unix Platform. The courses in the set are not excluding, but do not depend on each other either.

However, since entropy is increasing with respect to set inclusion, the itemset containing all items always has maximal entropy, and thus high-entropy sets do not form a local pattern class in the usual sense. Instead, we define versions of entropy scaled such that small itemsets have a chance of attaining a high score.

The discussion in Section 2 shows that the maximum possible value of $H(X)$ increases linearly with the number of items in X. In order to compare entropies of different-sized sets, we define the scaled entropy of X to be $H_s(X) = H(X)/|X|$. Another possibility is to consider the maximum possible value of $H(X)$ given the frequencies of all items $A \in X$. If these frequencies are fixed, it follows from the chain rule of entropy [8, Chapter 2] that the entropy $H(X)$ of the set X is maximized when the items constitute independent random variables, and in this case the entropy is $\sum_{A \in X} H(A)$. Thus we define the normalized entropy of X to be $H_n(X) = H(X)/\sum_{A \in X} H(A)$.

DEFINITION 12. *Given a threshold ϵ, an itemset X is a high-scaled-entropy (HSE) set in $\mathcal{D}$ if $H_s(X) \geq \epsilon$, and a high-normalized-entropy (HNE) set if $H_n(X) \geq \epsilon$.*

For normalized and scaled entropies, we do not have a monotonicity property like that of frequent itemsets. However, the following weaker properties can be proved. We prove that any HSE or HNE itemset X contains at least one HSE or HNE itemset $X' \subset X$ with $|X'| = |X| - 1$. This enables us to use a modified levelwise algorithm: we can prune those itemsets that have no HSE or HNE subsets.

PROPOSITION 13. *For all datasets $\mathcal{D}$ and for all sets $X \neq \emptyset$ of attributes there is an attribute $A \in X$ such that*

$$H_s(X) \leq H_s(X \setminus A).$$

PROOF. By definition, $H(X) = H(A \mid X \setminus A) + H(A)$. Hence, we can write

$$|X| H(X) = \sum_{A \in X} H(A \mid X \setminus A) + H(A).$$

Let $X = \{A_1, \ldots, A_k\}$. Then we can write

$$\begin{aligned} H(X) &= \sum_{i=1}^{k} H(A_i \mid \{A_{i+1}, \ldots, A_k\}) \\ &\geq \sum_{i=1}^{k} H(A_i \mid X \setminus A_i) = \sum_{A \in X} H(A \mid X \setminus A). \end{aligned}$$

Combining the observations above we get

$$\begin{aligned} (|X| - 1) H(X) &\leq \sum_{A \in X} H(X \setminus A) \\ &\leq |X| \min_{A \in X} H(X \setminus A) \end{aligned}$$

as claimed. □

PROPOSITION 14. *For all datasets $\mathcal{D}$ and for all sets $X \neq \emptyset$ of attributes there is an attribute $A \in X$ such that*

$$H_n(X) \leq H_n(X \setminus A).$$

PROOF. Observe that if $p/q \leq r/s$, we have

$$\frac{p}{q} \leq \frac{p+r}{q+s} \leq \frac{r}{s}.$$

The left-hand inequality follows from $p(q+s) = pq + ps \leq pq + qr = q(p+r)$, and the right-hand inequality is proven similarly. Using this inequality inductively on the fractions $f(A) = H(A \mid X \setminus A)/H(A)$ for all $A \in X$, we find that if A minimizes $f(A)$,

$$\frac{H(A \mid X \setminus A)}{H(A)} \leq \frac{\sum_{B \in X \setminus A} H(B \mid X \setminus B)}{\sum_{B \in X \setminus A} H(B)}. \tag{3}$$

Denote $X \setminus A$ by X', and enumerate the attributes in X' arbitrarily so that $X \setminus A = \{B_1, B_2, \ldots, B_n\}$. Consider the numerator of the fraction on the right-hand side. By

weakening the conditions of the conditional entropies we can only increase the entropies, and thus

$$\sum_{B \in X'} H(B \mid X \setminus B) \leq \sum_{j=1}^{n} H(B_j \mid B_1, \ldots, B_{j-1}) = H(X').$$

Thus we obtain from (3)

$$\frac{H(A \mid X')}{H(A)} \leq \frac{H(X')}{\sum_{B \in X'} H(B)}$$

and using again the inequality at the beginning of the proof,

$$\frac{H(A \mid X') + H(X')}{\sum_{B \in X} H(B)} \leq \frac{H(X')}{\sum_{B \in X'} H(B)}$$

Since the numerator on the left-hand side is equal to $H(X)$, we have shown that $H_n(X) \leq H_n(X') = H_n(X \setminus A)$. □

As the weak monotonicity results in a larger number of candidates, optimizations in candidate counting are of interest. In practice many candidates can be eliminated by utilizing the fact that $H(X) \leq H(X \setminus Y) + H(Y)$ for any $Y \subseteq X$.

An interesting combinatorial variant of high-entropy sets is obtained by looking, for an itemset X, at the number of different value combinations for X that occur in the data. This corresponds to projecting the data to the columns of X and discarding duplicate rows. If there is a set X such that all the $2^{|X|}$ combinations occur, then the Vapnik-Chervonenkis dimension [4] of the data is at least $|X|$.

The property of all $2^{|X|}$ combinations occurring is downwards closed, so the levelwise algorithm can be used. Such sets X are in a way maximally diverse in a combinatorial sense, and listing them could be useful in some situations.

5. EXPERIMENTS

5.1 Generated data

In this section we briefly discuss our experiments on generated data. We used a simple procedures to plant low-entropy itemsets into generated data. First, for each row u the set of attributes of an itemset X were all set to one with probability 0.5, or all to zero with 0.5 probability respectively. Notice that, this will results to an itemset pattern with an entropy score of 1. Second, some noise was added to the rows by independently flipping each bit with probability r.

Using this procedure we generated data sets with low-entropy patterns of four attributes ranging the noise parameter r from 0.0 to 0.3. For each data set the generated number of rows was 100000. The results for this experiment showed that using either D-trees, U-trees or low-entropy itemsets the best 4 size patterns always correctly corresponded to the attributes of the planted patterns. The number of planted patterns used in the experiments were 2 and 4.

5.2 Real data

Next we describe experiments on two real data sets, one about courses completed by computer science students at the University of Helsinki, and another about terms used in computer science bibliographies. We show that it is feasible to run our algorithms on these data sets, and give examples of interesting patterns found in them. We also show that the patterns found are significant in the sense that when the data sets are permuted, the number of patterns found at a given entropy threshold decreases by a large factor.

σ	ϵ	trees	cands	items	max	time
0.25	3.0	532	5778	27	3	2
0.25	3.5	2763	21856	27	4	4
0.25	4.0	5290	49432	27	5	9
0.2	3.0	1146	17525	34	3	3
0.2	3.5	6090	59623	34	4	11
0.2	4.0	13854	180499	34	5	31
0.15	3.0	3980	75063	43	4	11
0.15	3.5	15567	226530	43	5	40
0.15	4.0	44156	784541	43	6	140

Table 2: D-tree results for the course data. The number of D-trees in the collection $\mathcal{TP}_D(D, \epsilon)$ with various values of the preprocessing frequency threshold σ and entropy threshold ϵ. Legend: trees = number of trees found; cands = number of candidates examined; items = number of attributes in the data after preprocessing; max = size of the largest tree; time = running time in seconds.

In each case, we preprocess the data by removing attributes whose frequencies are outside a range. In both data sets there are many rarely occurring attributes – the course data includes seminars that were only given once, and the bibliography data has typographical errors, author names, and other rare words. Such attributes have low entropy but cannot be considered interesting, since the reasons for their rarity are known. We include the frequency threshold used as a parameter of the experiments. From the bibliography data we also remove very frequently occurring stop words.

Course data.

The course data describes courses taken by students at the Department of Computer Science of the University of Helsinki. The data has 2405 observations corresponding to students and 5021 attributes corresponding to courses. On average, each student has taken 26.9 courses.

Tables 2, 3 and 4 show the number of D-trees, U-trees, and low-entropy itemsets with different parameter values computed from the course data. For all three pattern classes the number of elements in the answer set increases rapidly with decreasing frequency threshold σ and increasing entropy threshold ϵ. This is similar to the behavior of frequent itemsets, whose number increases rapidly when the frequency threshold is decreased, and as in the case of frequent itemsets, for our pattern classes it is easy to iteratively find thresholds that produce outputs of desired size, and even for larger output sizes the running times stay feasible.

Some example U-trees drawn from the results are shown in Figures 1, 4, and 5. The tree in Figure 1, already discussed in the introduction, is the lowest-entropy 5-node tree in the course enrolment data for a preprocessing frequency threshold of 0.15. The trees in Figures 4 and 5 are the best two 5-node U-trees left when we eliminate the most frequently taken courses (frequency > 0.18). Both trees clearly have an AI component and a software engineering component, corresponding to two of the specialization areas within the department.

Similarly, the lowest-entropy 7-node D-tree (Figure 6) has clear AI and distributed systems components. The central

σ	ϵ	trees	low	max	cands	time
0.25	2.0	411	364	3	3276	2.55
0.2	2.0	699	598	3	6546	3.61
0.15	2.0	1836	1326	4	13535	6.81
0.25	2.4	1873	904	4	4528	3.56
0.2	2.4	5070	2506	4	13078	8.67
0.15	2.4	16802	8167	4	53823	122
0.25	2.8	8027	3270	5	20040	16.8
0.2	2.8	19383	6774	5	51841	80.8
0.15	2.8	91774	21835	5	135196	600

Table 3: U-tree results for the course data. The number of U-trees in the collection $\mathcal{TP}_U(D, \epsilon, \beta)$ with the maximum branching factor of $\beta = 3$ for various values of the preprocessing frequency threshold σ and entropy threshold ϵ. Legend: trees = number of trees found; low = number of low trees found; max = size of the largest tree; cands = number of candidates examined; time = running time in seconds.

σ	ϵ	sets	cands	max	time	
0.25	2.0	434	4262	27	3	1
0.25	2.4	1569	15879	27	4	3
0.25	2.8	4837	38807	27	5	8
0.20	2.0	739	9953	34	3	2
0.20	2.4	3807	46565	34	4	8
0.20	2.8	11682	129475	34	5	25
0.15	2.0	1650	33138	43	4	5
0.15	2.4	10274	144997	43	4	25
0.15	2.8	38266	567253	43	5	105

Table 4: Low-entropy itemset results for the course data. The number of low-entropy sets for various values of the preprocessing frequency threshold σ and entropy threshold ϵ. Legend: sets = number of itemsets found; cands = number of candidates examined; max = size of the largest set found; time = running time in seconds.

node of the tree is Theory of Computation. On one hand, the students taking Artificial Intelligence and AI Languages usually consider Theory of Computation as a relevant course. On the other hand, the test of the branches correspond to students very close to graduation, which means that most of them has also passed Theory of Computation.

As already discussed, low-entropy itemsets are not as easily interpretable as the tree patterns. One example of a low-entropy set found in the course data is shown in Table 5.

Bibliography data.

The bibliography data consists of terms used in bibliographies on theory and foundations of computer science.[2] The data set contains 67043 bibliography entries. For the experiment we reduced the data set by taking a random 10% sample of the data resulting in a set of 6695 bibliography entries. In addition to this. the frequent stop words – `an`, `to`, `with`, `in`, `on`, `a`, `for`, `and`, `the`, `by` `some`, `is`, `from`, and `of` – were removed. On average the resulting data set has 8.28 words per bibliography entry. Tables 6, 7 and 8 show the number of D-trees, U-trees, and low-entropy itemsets with different parameter values computed from the bibliography data. The best D-tree of size 6 is shown as Figure 7; since the root of a D-tree can be selected arbitrarily, the edges in the figure are not directed.

[2]http://liinwww.ira.uka.de/bibliography/Theory/Seiferas/

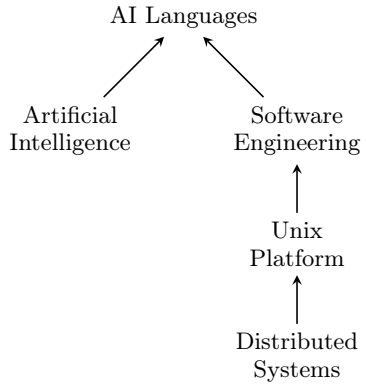

Figure 4: Example U-tree from course data.

	CS	SW	DB	MT	ToC	UI	MSc
1378			(none of the courses)				
114	CS	SW	DB	MT			
102	CS	SW	DB	MT	ToC		MSc
88		SW		MT			
86	CS	SW	DB	MT	ToC	UI	MSc
82	CS	SW	DB	MT	ToC		
65	CS						
63	CS		DB				
55			DB				
54	CS	SW	DB	MT		UI	
54	CS	SW	DB	MT	ToC	UI	
33	CS	SW		MT			
23						UI	
19	CS	SW	DB	MT		UI	MSc

Table 5: Joint distribution of an example low-entropy itemset in the course data. Legend: CS = Concurrent Systems; SW = Scientific Writing; DB = Database Systems 1; MT = Maturity Test; ToC = Theory of Computation; UI = User Interfaces; MSc = Master's Thesis.

Result validation with data randomization.

To evaluate how well the D-trees, U-trees and low-entropy itemsets find significant structure in data we compare the scores of the patterns found from the course data set against the scores of patterns obtained form 10 randomized course data set instances. For this we use the swap randomization method described in [11]. The method preserves the row and column margins of the given dataset, but obscures the internal dependencies of the data. The idea is that if the

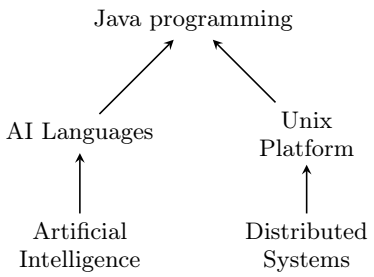

Figure 5: Example U-tree from course data.

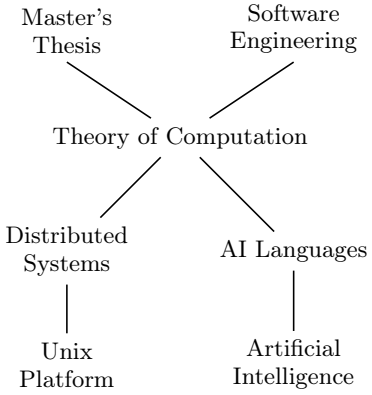

Figure 6: Example D-tree from course data.

σ	ϵ	trees	cands	items	max	time
0.025	0.6	269	2244	19	3	1
0.025	0.7	545	3795	19	4	1
0.025	0.8	1096	6852	19	4	1
0.020	0.6	3222	44048	33	4	7
0.020	0.7	12566	154186	33	4	28
0.020	0.8	32246	371959	33	5	75
0.015	0.6	83640	1680737	52	5	323
0.015	0.7	374097	7310174	52	6	1650

Table 6: D-tree results for the bibliography data. The number of D-trees in the collection $\mathcal{TP}_D(D, \epsilon)$ with various values of the preprocessing frequency threshold σ and entropy threshold ϵ. Legend: trees = number of trees found; cands = number of candidates examined; items = number of attributes in the data after preprocessing; max = size of the largest tree; time = running time in seconds.

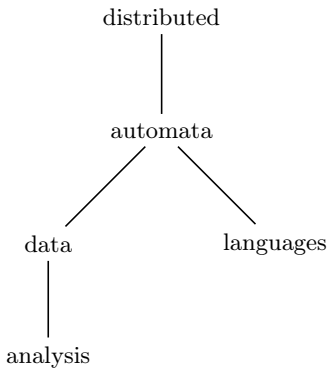

Figure 7: Example D-tree from bibliography data.

σ	ϵ	β	trees	low	max	cands	time
0.025	0.6	2	353	249	3	892	1.96
0.025	0.6	3	353	249	3	1011	2.40
0.025	0.7	2	889	517	4	1123	2.01
0.025	0.7	3	894	522	4	1792	3.30
0.02	0.6	2	8515	3272	4	6036	7.57
0.02	0.6	3	9019	3776	4	20542	24.5
0.02	0.7	2	58234	5231	4	6513	18.3
0.02	0.7	3	67904	14901	4	35545	152
0.015	0.6	2	369755	17147	5	22615	617
0.015	0.6	3	438827	82399	5	180058	5220
0.015	0.7	2	4710983	21555	6	23376	13900

Table 7: U-tree results for the bibliography data. The number of U-trees in the collection $\mathcal{TP}_U(D, \epsilon, \beta)$ with the maximum branching factor of $\beta = 3$ and $\beta = 2$ for various values of the preprocessing frequency threshold σ and entropy frequency ϵ. Legend: trees = number of trees found; low = number of low trees found; max = size of the largest tree; cands = number of candidates examined; time = running time in seconds.

true structure in the data is captured by the patterns, the number and scores of the found patterns should be better in the original data than in the randomized data sets instances.

The results of the experiment are depicted in Figure 8, 9, and 10. For each figure the upper image is the histogram of the scores of patterns found from the original course data and the lower image depicts the respective histogram for the aggregate patterns found in the 10 swap randomized course data sets. The mining was done using courses with a frequency of 0.2 or higher in the data and an entropy parameter of $\epsilon = 2.8$. For U-trees the maximum branching factor was limited to 4.

The results show that the amount of structure in terms of the number of patterns found from the original data compared to the randomized data instances is larger for all three proposed patterns classes. With the given parameters the number of generated D-trees is 1.8 times larger in the original data set compared to the randomized data set instances on the average and 1.6 time larger for the U-trees and 2.1 for the low-entropy sets. Moreover, the average score within

σ	ϵ	sets	cands	items	max	time
0.025	0.5	108	460	19	2	2
0.025	0.6	269	952	19	3	2
0.025	0.7	546	1811	19	4	4
0.0225	0.5	312	1473	25	3	5
0.0225	0.6	924	3932	25	3	15
0.0225	0.7	2267	8233	25	4	34
0.020	0.5	1414	6998	33	3	41
0.020	0.6	3230	17552	33	4	110
0.020	0.7	12608	53370	33	4	339

Table 8: Low-entropy itemset results for the bibliography data. The number of low-entropy sets for various values of the preprocessing frequency threshold σ and entropy frequency ϵ. Legend: sets = number of itemsets found; cands = number of candidates examined; max = size of the largest set found; time = running time in seconds.

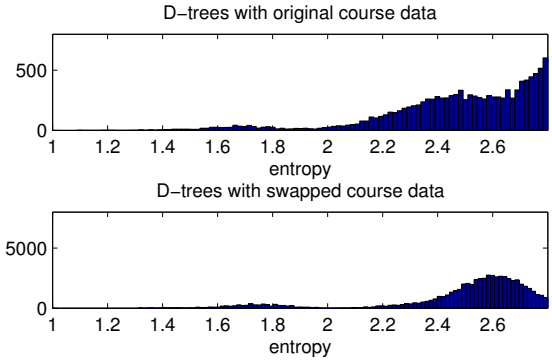

Figure 8: Validation of D-tree results in course data. Upper pane shows the distribution of entropies for 11648 D-trees mined from the course data with frequency threshold 0.2 and entropy threshold 2.8. Lower pane shows the respective distribution from 10 aggregated swap-randomized course data set instances (6297 trees on average).

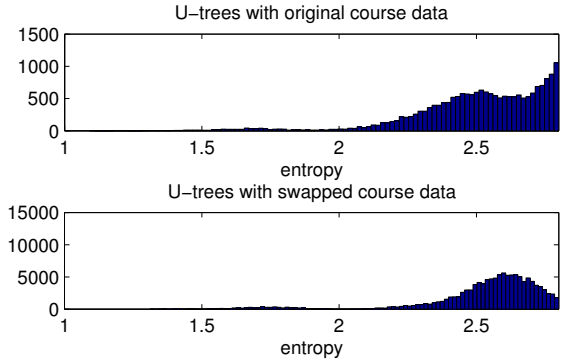

Figure 9: Validation of U-tree results in course data. Upper pane shows the distribution of entropies of 19380 U-trees mined from the course data, lower pane shows the corresponding distribution in 10 aggregated randomizations of the data (11967 trees on average).

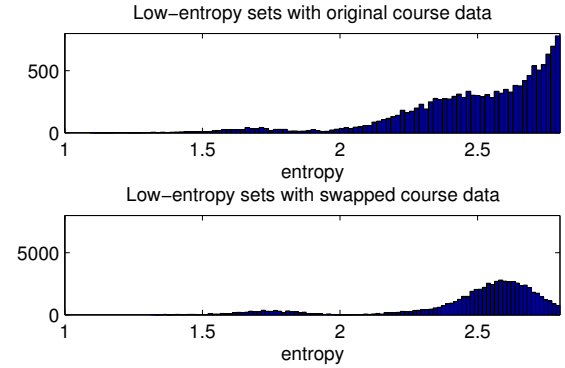

Figure 10: Validation of low-entropy itemset results in course data. Upper pane shows the distribution of entropies of 13575 itemsets mined from the course data, lower pane shows the corresponding distribution in 10 aggregated randomizations of the data (6365 itemsets on average).

patterns of the same size is smaller and the average pattern size is larger in the original data for all pattern classes.

6. RELATED WORK

Decision trees [12, Chapter 9.2] are similar to D-trees in that low classification error implies low entropy. There are two crucial differences. First, D-trees do not split the data into groups corresponding to different values of an attribute, but all data are considered in each node. In other words, $H(A \mid B)$ is used instead of e.g. $H(A \mid B = b)$. Second, decision tree algorithms seek greedily one tree, but the D-tree algorithm finds exhaustively all D-trees satisfying the criteria.

The work in [14] is a simple special case of the current setting. The trees of the earlier paper were defined to be present in a row of data if the attributes of a rooted subtree were present simultaneously: e.g. for the tree of Figure 1, this means that Scientific Writing and Maturity Test must be present whenever anything lower in the tree is present, and if Probability Theory 1 is present, then the whole branch from Scientific Writing to Probability Theory 1 must be present. Such a tree constitutes a low-entropy U-tree, but U-trees are more general by allowing the distribution to be concentrated on other combinations than positive conjunctions.

The task of finding interesting itemsets has been addressed mainly in the context of frequent itemset mining. Morishita and Sese have presented a branch-and-bound method for finding association rules like $X \Rightarrow Y$ where the itemsets X and Y are not required to have high support but high correlation [19]. They also count those data rows where X and Y appear completely, whereas our entropy-based method counts arbitrary combinations. Similar strategies have been employed by Zimmermann *et al.* [24, 5]. In [5] they describe a sequence of graph pattern classes that is to some extent paralleled by our sequence of itemset pattern classes.

Closer to our approach are Knobbe and Ho's maximally informative k-itemsets, i.e., itemsets with as high entropy

as possible [16]. The difference to our high-entropy itemsets is that Knobbe and Ho avoid the trivial result of the full itemset by restricting their itemsets to have a fixed number k of elements, whereas we define scaled versions of entropy and thus avoid having to specify an extra parameter. Another information-theoretically motivated approach is to select itemsets that can be used to compress the data, recently introduced by Siebes *et al.* [22, 23]. They, however, consider only all-1s itemsets.

In real-valued data, the task of finding interesting subsets has been addressed by research areas such as subspace clustering [20, 1] and projection pursuit [9, 15]. Another method somewhat related to our task is the problem of learning the structure of a Bayesian network. This problem is computationally challenging, and the main methods are probabilistic [7, 13]; the best current exact algorithms are of order $O(n2^n)$ [17]. The key difference between Bayesian network structure learning and our approach is that we seek interesting subsets of the data, not complete models; also, of course, we investigate only fully connected or tree-structured networks and not arbitrary graphs, and we do not try to incorporate any prior knowledge into the patterns.

7. CONCLUDING REMARKS

We have considered the problem of finding low-entropy sets and trees from binary data. The approach we have chosen is a natural generalization of the discovery of frequent sets and association rules: a frequent set is a particular form of a low-entropy set.

We defined the concepts of low and high-entropy sets and two types of trees, and gave efficient algorithms for finding such sets. The experiments show that the methods are able to discover interesting sets and trees and that they are feasible to use also on large datasets. We also considered the search for high-entropy sets of variables.

Our approach searches for local structure: small subsets of variables for which the data can be modeled well, in the sense of having a small entropy or being amenable to be described by a tree. Such techniques are especially useful when dealing with datasets of large dimension, and when many of the dimensions are assumed to be relatively unimportant.

8. REFERENCES

[1] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD Conference* (1998), pp. 94–105.

[2] Agrawal, R., Imielinski, T., and Swami, A. N. Mining association rules between sets of items in large databases. In *SIGMOD Conference* (1993), pp. 207–216.

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

[4] Anthony, M., and Biggs, N. *Computational Learning Theory: An Introduction*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.

[5] Bringmann, B., Zimmermann, A., De Raedt, L., and Nijssen, S. Don't be afraid of simpler patterns. In *PKDD* (2006), pp. 55–66.

[6] Chow, C. K., and Liu, C. N. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Info. Theory 14*, 3 (1968), 462–467.

[7] Cooper, G. F., and Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning 9* (1992), 309–347.

[8] Cover, T. M., and Thomas, J. A. *Elements of Information Theory*. Wiley Series in Telecommunications. Wiley Interscience, 1991.

[9] Friedman, J. H., and Tukey, J. W. A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput. 23* (1974), 881–890.

[10] Gionis, A., Kujala, T., and Mannila, H. Fragments of order. In *KDD* (2003), pp. 129–136.

[11] Gionis, A., Mannila, H., Mielikäinen, T., and Tsaparas, P. Assessing data mining results via swap randomization. In *KDD* (2006), pp. 167–176.

[12] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*. Springer, 2001.

[13] Heckerman, D., Geiger, D., and Chickering, D. M. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning 20*, 3 (1995), 197–243.

[14] Heikinheimo, H., Mannila, H., and Seppänen, J. K. Finding trees from unordered 0-1 data. In *PKDD* (2006), pp. 175–186.

[15] Huber, P. J. Projection pursuit. *The Annals of Statistics 13*, 2 (June 1985), 435–475.

[16] Knobbe, A. J., and Ho, E. K. Y. Maximally informative k-itemsets and their efficient discovery. In *KDD* (2006), pp. 237–244.

[17] Koivisto, M. Advances in exact Bayesian structure discovery in Bayesian networks. In *UAI* (2006), pp. 241–248.

[18] Meilă, M., and Jordan, M. I. Learning mixtures of trees. *Journal of Machine Learning Research 1* (2000), 1–48.

[19] Morishita, S., and Sese, J. Traversing itemset lattice with statistical metric pruning. In *PODS* (2000), ACM, pp. 226–236.

[20] Parsons, L., Haque, E., and Liu, H. Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl. 6*, 1 (2004), 90–105.

[21] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[22] Siebes, A., Vreeken, J., and van Leeuwen, M. Item sets that compress. In *SDM* (2006), J. Ghosh, D. Lambert, D. B. Skillicorn, and J. Srivastava, Eds., SIAM.

[23] van Leeuwen, M., Vreeken, J., and Siebes, A. Compression picks item sets that matter. In *PKDD* (2006), pp. 585–592.

[24] Zimmermann, A., and De Raedt, L. CorClass: Correlated association rule mining for classification. In *Discovery Science* (2004), E. Suzuki and S. Arikawa, Eds., vol. 3245 of *Lecture Notes in Computer Science*, Springer, pp. 60–72.