Appears in *Proc. of the Fifth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 145-154, 1999.

Mining the Most Interesting Rules

Roberto J. Bayardo Jr.
IBM Almaden Research Center
http://www.almaden.ibm.com/cs/people/bayardo/
bayardo@alum.mit.edu

Rakesh Agrawal
IBM Almaden Research Center
http://www.almaden.ibm.com/u/ragrawal/
ragrawal@acm.org

Abstract

Several algorithms have been proposed for finding the "best," "optimal," or "most interesting" rule(s) in a database according to a variety of metrics including confidence, support, gain, chi-squared value, gini, entropy gain, laplace, lift, and conviction. In this paper, we show that the best rule according to any of these metrics must reside along a support/confidence border. Further, in the case of conjunctive rule mining within categorical data, the number of rules along this border is conveniently small, and can be mined efficiently from a variety of real-world data-sets. We also show how this concept can be generalized to mine all rules that are best according to any of these criteria with respect to an arbitrary subset of the population of interest. We argue that by returning a broader set of rules than previous algorithms, our techniques allow for improved insight into the data and support more user-interaction in the optimized rule-mining process.

1. Introduction

There are numerous proposals for mining rules from data. Some are *constraint-based* in that they mine every rule satisfying a set of hard constraints such as minimum support or confidence (e.g. [1,2,6]). Others are *heuristic* in that they attempt to find rules that are predictive, but make no guarantees on the predictiveness or the completeness of the returned rule set (e.g. decision tree and covering algorithms [9,15]). A third class of rule mining algorithms, which are the subject of this paper, identify only the most interesting, or *optimal*, rules according to some interestingness metric [12,18,20,24]. Optimized rule miners are particularly useful in domains where a constraint-based rule miner produces too many rules or requires too much time.

It is difficult to come up with a single metric that quantifies the "interestingness" or "goodness" of a rule, and as a result, several different metrics have been proposed and used. Among them are confidence and support [1], gain [12], variance and chi-squared value [17,18], entropy gain [16,17], gini [16], laplace [9,24], lift [14] (a.k.a. interest [8] or strength [10]), and conviction [8]. Several algorithms are known to efficiently find the best rule (or a close approximation to the best rule [16]) according to a specific one of these metrics [12,18,20,24]. In this paper, we show that a single yet simple concept of rule goodness captures the best rules according to *any* of them. This concept involves a partial order on rules defined in terms of both rule support and confidence. We demonstrate that the set of rules that are optimal according to this partial order includes all rules that are best according to any of the above metrics, even given arbitrary minimums on support and/or confidence.

In the context of mining conjunctive association rules, we present an algorithm that can efficiently mine an optimal set according to this partial order from a variety of real-world data-sets. For example, for each of the categorical data-sets from the Irvine machine learning repository (excepting only connect-4), this algorithm requires less than 30 seconds on a 400Mhz Pentium-II class machine. Specifying constraints such as minimum support or confidence reduces execution time even further. While optimizing according to only a single interestingness metric could sometimes require less overhead, the approach we propose is likely to be advantageous since it supports an interactive phase in which the user can browse the optimal rule according to any of several interestingness metrics. It also allows the user to interactively tweak minimums on support and confidence. Witnessing such effects with a typical optimized rule miner requires repeated mining runs, which may be impractical when the database is large.

Another need for repeated invocations of an optimized rule miner arises when the user needs to gain insight into a broader population than what is already well-characterized by previously discovered rules. We show how our algorithm can be generalized to produce every rule that is optimal according to any of the previously mentioned interestingness metrics, and additionally, with respect to an arbitrary subset of the population of interest. Because data-mining is iterative and discovery-driven, identifying several good rules up-front in order to avoid repeatedly querying the database reduces total mining time when amortized over the entire process [13].

2. Preliminaries

2.1 Generic Problem Statement

A *data-set* is a finite set of *records*. For the purpose of this paper, a record is simply an element on which we apply boolean predicates called *conditions*. A *rule* consists of two conditions called the antecedent and consequent, and is denoted as $A \rightarrow C$ where A is the antecedent and C the consequent. A rule *constraint* is a boolean predicate on a rule. Given a set of constraints N, we say that a rule r satisfies the constraints in N if every constraint in N evaluates to true given r. Some common examples of constraints are item constraints [22] and minimums on support and confidence [1]. The input to the problem of mining optimized rules is a 5-tuple $\langle U, D, \leq, C, N \rangle$ where:

- U is a finite set of conditions;
- D is a data-set;
- $\leq$ is a total order on rules;
- C is a condition specifying the rule consequent;
- N is a set of constraints on rules.

When mining an optimal disjunction, we treat a set of conditions $A \subseteq U$ as a condition itself that evaluates to true if and only if one or more of the conditions within A evaluates to true on the given record. When mining an optimal conjunction, we treat A as a condition that evaluates to true if and only if every condition within A evaluates to true on the given record. For both cases, if A is empty then it always evaluates to true. Algorithms for mining optimal conjunctions and disjunctions differ significantly in their details, but the problem can be formally stated in an identical manner[1]:

PROBLEM (OPTIMIZED RULE MINING): Find a set $A_1 \subseteq U$ such that

(1) A_1 satisfies the input constraints, and

(2) there exists no set $A_2 \subseteq U$ such that A_2 satisfies the input constraints and $A_1 < A_2$.

Any rule $A \to C$ whose antecedent is a solution to an instance I of the optimized rule mining problem is said to be I*-optimal* (or just *optimal* if the instance is clear from the context). For simplicity, we sometimes treat rule antecedents (denoted with A and possibly some subscript) and rules (denoted with r and possibly some subscript) interchangeably since the consequent is always fixed and clear from the context.

We now define the support and confidence values of rules. These values are often used to define rule constraints by bounding them above a pre-specified value known as *minsup* and *minconf* respectively [1], and also to define total orders for optimization [12,20]. The *support* of a condition A is equal to the number of records in the data-set for which A evaluates to true, and this value is denoted as $\sup(A)$. The support of a rule $A \to C$, denoted similarly as $\sup(A \to C)$, is equal to the number of records in the data-set for which both A and C evaluate to true.[2] The *antecedent support* of a rule is the support of its antecedent alone. The *confidence* of a rule is the probability with which the consequent evaluates to true given that the antecedent evaluates to true in the input data-set, computed as follows:

$$\mathrm{conf}(A \to C) = \frac{\sup(A \to C)}{\sup(A)}$$

2.2 Previous Algorithms for the Optimized Rule Mining Problem

Many previously proposed algorithms for optimized rule mining solve specific restrictions of the optimized rule mining problem. For example, Webb [24] provides an algorithm for mining an optimized conjunction under the following restrictions:

- U contains an existence test for each attribute/value pair appearing in a categorical data-set outside a designated class column;
- $\leq$ orders rules according to their laplace value (defined later);
- N is empty.

Fukuda et al. [12] provide algorithms for mining an optimized disjunction where:

- U contains a membership test for each square of a grid formed by discretizing two pre-specified numerical attributes of a data-set (a record is a member of a square if its attribute values fall within the respective ranges);
- $\leq$ orders rules according to either confidence, antecedent support, or a notion they call gain (also defined later);
- N includes minimums on support or confidence, and includes one of several possible “geometry constraints” that restrict the allowed shape formed by the represented set of grid squares;

Rastogi and Shim [20] look at the problem of mining an optimized disjunction where:

- U includes a membership test for every possible hypercube defined by a pre-specified set of record attributes with either ordered or categorical domains;
- $\leq$ orders rules according to antecedent support or confidence;
- N includes minimums on antecedent support or confidence, a maximum k on the number of conditions allowed in the antecedent of a rule, and a requirement that the hypercubes corresponding to the conditions of a rule are non-overlapping.

In general, the optimized rule mining problem, whether conjunctive or disjunctive, is NP-hard [17]. However, features of a specific instance of this problem can often be exploited to achieve tractability. For example, in [12], the geometry constraints are used to develop low-order polynomial time algorithms. Even in cases where tractability is not guaranteed, efficient mining in practice has been demonstrated [18,20,24]. The theoretical contributions in this paper are conjunction/disjunction neutral. However, we focus on the conjunctive case in validating the practicality of these results through empirical evaluation.

2.3 Mining Optimized Rules under Partial Orders

We have carefully phrased the optimized rule mining problem so that it may accommodate a partial order in place of a total order. With a partial order, because some rules may be incomparable, there can be several equivalence classes containing optimal rules. The previous problem statement requires an algorithm to identify only a single rule from one of these equivalence classes. However, in our application, we wish to mine at least one representative from each equivalence class that contains an optimal rule. To do so, we could simply modify the previous problem statement to find all optimal rules instead of just one. However, in practice, the equivalence classes of rules can be large, so this would be unnecessarily inefficient. The next problem statement enforces our requirements specifically:

PROBLEM (PARTIAL-ORDER OPTIMIZED RULE MINING): Find a set O of subsets of U such that:

(1) every set A in O is optimal as defined by the optimized rule mining problem.

(2) for every equivalence class of rules as defined by the partial order, if the equivalence class contains an optimal rule, then exactly one member of this equivalence class is within O.

We call a set of rules whose antecedents comprise a solution to an instance I of this problem an I*-optimal* set. An I-optimal rule is one that may appear in an I-optimal set.

2.4 Monotonicity

Throughout this paper, we exploit (anti-)monotonicity properties of functions. A function $f(x)$ is said to be monotone (resp. anti-monotone) in x if $x_1 < x_2$ implies that $f(x_1) \leq f(x_2)$ (resp. $f(x_1) \geq f(x_2)$). For example, the confidence function, which is defined in terms of rule support and antecedent support, is anti-monotone in antecedent support when rule support is held fixed.

3. SC-Optimality

3.1 Definition

Consider the following partial order $\leq_{sc}$ on rules. Given rules r_1 and r_2, $r_1 <_{sc} r_2$ if and only if:

- $\sup(r_1) \leq \sup(r_2) \wedge \mathrm{conf}(r_1) < \mathrm{conf}(r_2)$, or
- $\sup(r_1) < \sup(r_2) \wedge \mathrm{conf}(r_1) \leq \mathrm{conf}(r_2)$.

Additionally, $r_1 =_{sc} r_2$ if and only if $\sup(r_1) = \sup(r_2)$ and $\mathrm{conf}(r_1) = \mathrm{conf}(r_2)$.

An I-optimal set where I contains this partial order is depicted in Figure 1. Intuitively, such a set of rules defines a *support-confidence border* above which no rule that satisfies the input constraints can fall.

1 Algorithms for mining optimal disjunctions typically allow a single fixed conjunctive condition without complications, e.g. see [20]. We ignore this issue for simplicity of presentation.

2 This follows the original definition of support as defined in [1]. The reader is warned that in the work of Fukuda et. al. [14] and Rastogi and Shim [20] (who are careful to note the same discrepancy), the definition of support corresponds to our notion of antecedent support.

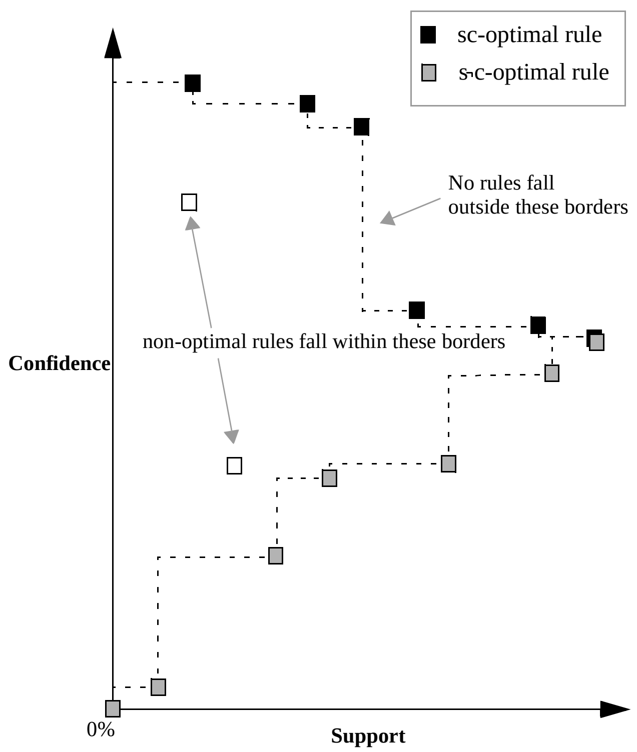

Figure 1. Upper and lower support-confidence borders.

Consider also the similar partial order $\leq_{s\neg c}$ such that $r_1 <_{s\neg c} r_2$ if and only if:

- $\text{sup}(r_1) \leq \text{sup}(r_2) \wedge \text{conf}(r_1) > \text{conf}(r_2)$, or
- $\text{sup}(r_1) < \text{sup}(r_2) \wedge \text{conf}(r_1) \geq \text{conf}(r_2)$.

The equivalence condition is the same as before. An optimal set of rules according to this partial order forms a lower border.

3.2 Theoretical Implications

In this section, we show that for many total orders $\leq_t$ intended to rank rules in order of interestingness, we have that $r_1 <_{sc} r_2 \Rightarrow r_1 \leq_t r_2$, and $r_1 =_{sc} r_2 \Rightarrow r_1 =_t r_2$. We say that any such total order $\leq_t$ is *implied* by $\leq_{sc}$. This property of a total order is useful due to the following fact:

LEMMA 3.1: Given the problem instance $I = \langle U, D, \leq_t, C, N\rangle$ such that $\leq_t$ is implied by $\leq_{sc}$, an I-optimal rule is contained within any I_{sc}-optimal set where $I_{sc} = \langle U, D, \leq_{sc}, C, N\rangle$.

Proof: Consider any rule r_1 that is not I_{sc}-optimal (for simplicity we will ignore the presence of constraints in N). Because r_1 is non-optimal, there must exist some rule r_2 that is optimal such that $r_1 <_{sc} r_2$. But then we also have that $r_1 \leq_t r_2$ since $\leq_t$ is implied by $\leq_{sc}$. This implies that any non-I_{sc}-optimal rule is either non-I-optimal, or it is equivalent to some I-optimal rule which resides in an I_{sc}-optimal equivalence class. At least one I_{sc}-optimal equivalence class must therefore contain an I-optimal rule. Further, because $=_t$ is implied by $=_{sc}$, every rule in this equivalence class must be I-optimal. By definition, an I_{sc}-optimal set will contain one of these rules, and the claim follows. □

Put simply, mining the upper support/confidence border identifies optimal rules according to several different interestingness metrics. We will show that these metrics include support, confidence, conviction, lift, laplace, gain, and an unnamed interest measure proposed by Piatetsky-Shapiro [19]. If we also mine the lower border, metrics such as entropy gain, gini, and chi-squared value are also included. But first, consider the following additional property of total orders implied by $\leq_{sc}$:

OBSERVATION 3.2: Given instance $I = \langle U, D, \leq_t, C, N\rangle$ such that $\leq_t$ is implied by $\leq_{sc}$, and N contains a minimum support constraint n_s and/or a minimum confidence constraint n_c, an I-optimal rule is contained within any I_{sc}-optimal set where $I_{sc} = \langle U, D, \leq_{sc}, C, N - \{n_s, n_c\}\rangle$.

The implication of this fact is that we can mine without minimum support and confidence constraints, and the optimal rule given any setting of these constraints will remain in the mined rule set. This allows the user to witness the effects of modifying these constraints without further mining of the data -- a useful fact since the user often cannot determine accurate settings of minimum support or confidence apriori. Minimum support and confidence constraints are quite critical in some applications of optimized rule mining, particularly when the optimization metric is itself support or confidence as in [12] and [20].

To identify the interestingness metrics that are implied by $\leq_{sc}$, we use the following lemma.

LEMMA 3.3: The following conditions are sufficient for establishing that a total order $\leq_t$ defined over a rule value function $f(r)$ is implied by partial order $\leq_{sc}$:
(1) $f(r)$ is monotone in support over rules with the same confidence, and
(2) $f(r)$ is monotone in confidence over rules with the same support.

Proof: Suppose $r_1 <_{sc} r_2$, then consider a rule r where $\text{sup}(r_1) = \text{sup}(r)$ and $\text{conf}(r_2) = \text{conf}(r)$. Note that by definition $r_1 \leq_{sc} r$ and $r \leq_{sc} r_2$. Now, if a total order has the monotonicity properties from above, then $r_1 \leq_t r$ and $r \leq_t r_2$. Since total orders are transitive, we then have that $r_1 \leq_t r_2$, which establishes the claim. □

These conditions trivially hold when the rule value function is $\text{sup}(r)$ or $\text{conf}(r)$. So consider next the Laplace function which is commonly used to rank rules for classification purposes [9,24].

$$\text{laplace}(A \to C) = \frac{\text{sup}(A \to C) + 1}{\text{sup}(A) + k}$$

The constant k is an integer greater than 1 (usually set to the number of classes when building a classification model). Note that if confidence is held fixed to some value c, then we can rewrite the Laplace value as below.

$$\text{laplace}(r) = \frac{\text{sup}(r) + 1}{\text{sup}(r)/c + k}$$

It is straightforward to show that this expression is monotone in rule support since $k > 1$ and $c \geq 0$. The Laplace function is also monotone in confidence among rules with equivalent support. To see why, note that if support is held constant, in order to raise the function value, we need to decrease the value of the denominator. This decrease can only be achieved by reducing antecedent support, which implies a larger confidence. Note that an optimal set contains the optimized Laplace rule for any valid setting of k. This means the user can witness the effect of varying k on the optimal rule without additional mining of the database.

$$\text{gain}(A \to C) = \text{sup}(A \to C) - \theta \times \text{sup}(A)$$

The gain function of Fukuda et al. [12] is given above, where θ is a fractional constant between 0 and 1. If confidence is held fixed at c, then this function is equal to $\text{sup}(r)(1 - \Theta/c)$, which is trivially monotone in support as long as $c \geq \Theta$. We can ignore the case where $c < \Theta$ if we assume there exists any rule r satisfying the input constraints such that $\mathrm{conf}(r) \geq \Theta$. This is because for any pair of rules r_1 and r_2 such that $\mathrm{conf}(r_1) \geq \Theta$ and $\mathrm{conf}(r_2) < \Theta$, we know that $\mathrm{gain}(r_1) \geq \mathrm{gain}(r_2)$ irrespective of their support. Should this assumption be false, the gain criteria is optimized by any rule with zero support should one exist (e.g. in the conjunctive case, one can simply add conditions to a rule until its support drops to zero).

The gain function is monotone in confidence among rules with equivalent support for reasons similar to the case of the Laplace function. If support is held constant, then an increase in gain implies a decrease in the subtractive term. The subtractive term can be decreased only by reducing antecedent support, which implies a larger confidence. Note that, like k from the Laplace function, after identifying the optimal set of rules, the user can vary θ and view its effect on the optimal rule without additional mining. Another interestingness metric that is identical to gain for a fixed value of $\Theta = \mathrm{sup}(C) / |D|$ was introduced by Piatetsky-Shapiro [19]:

$$\text{p-s}(A \rightarrow C) = \mathrm{sup}(A \rightarrow C) - \frac{\mathrm{sup}(A)\mathrm{sup}(C)}{|D|}$$

Consider next conviction [8], which was framed in [6] as a function of confidence:

$$\mathrm{conviction}(A \rightarrow C) = \frac{|D| - \mathrm{sup}(C)}{|D|(1 - \mathrm{conf}(A \rightarrow C))}$$

Conviction is obviously monotone in confidence since confidence appears in a subtractive term within the denominator. It is also unaffected by variations in rule support if confidence is held constant, which implies monotonicity.

Lift, a well-known statistical measure that can be used to rank rules in IBM's Intelligent Miner [14] (it is also known as interest [8] and strength [10]), can also be framed as a function of confidence [6]:

$$\mathrm{lift}(A \rightarrow C) = \frac{|D|\mathrm{conf}(A \rightarrow C)}{\mathrm{sup}(C)}$$

Like conviction, lift is obviously monotone in confidence and unaffected by rule support when confidence is held fixed.

The remaining interestingness metrics, entropy gain, gini, and chi-squared value, are not implied by $\leq_{sc}$. However, we show that the space of rules can be partitioned into two sets according to confidence such that when restricted to rules in one set, each metric is implied by $\leq_{sc}$, and when restricted to rules in the other set, each metric is implied by $\leq_{s\neg c}$. As a consequence, the optimal rules with respect to entropy gain, gini, and chi-squared value must reside on either the upper or lower support confidence border. This idea is formally stated by the observation below.

OBSERVATION 3.4: Given instance $I = \langle U, D, \leq_t, C, N \rangle$, if $\leq_{sc}$ implies $\leq_t$ over the set of rules whose confidence is greater than equal to some value γ, and $\leq_{s\neg c}$ implies $\leq_t$ over the set of rules whose confidence is less than or equal to γ, then an I-optimal rule appears in either (a) any I_{sc} optimal set where $I_{sc} = \langle U, D, \leq_{sc}, C, N \rangle$, or (b) any $I_{s\neg c}$-optimal set where $I_{s\neg c} = \langle U, D, \leq_{s\neg c}, C, N \rangle$.

To demonstrate that the entropy gain, gini, and chi-squared values satisfy the requirements put forth by this observation, we need to know when the total order defined by a rule value function is implied by $\leq_{s\neg c}$. We use an analog of the Lemma 3.3 for this purpose:

LEMMA 3.5: The following conditions are sufficient for establishing that a total order $\leq_t$ defined over a rule value function $f(r)$ is implied by partial order $\leq_{s\neg c}$:
(1) $f(r)$ is monotone in support over rules with the same confidence, and
(2) $f(r)$ is *anti*-monotone in confidence over rules with the same support.

In [16], the chi-squared, entropy gain, and gini values of a rule are each defined in terms of a function $f(x, y)$ where $x = \mathrm{sup}(A) - \mathrm{sup}(A \cup C)$ and $y = \mathrm{sup}(A \cup C)$ given the rule $A \rightarrow C$ to which it is applied[3] (definitions appear in Appendix A). These functions are further proven to be *convex*. Another important property of each of these functions is that they reach their minimum at any rule whose confidence is equal to the "expected" confidence [17]. More formally, $f(x, y)$ is minimum when $\mathrm{conf}(x, y) = c$ where $\mathrm{conf}(x, y) = y / (x + y)$ and $c = \mathrm{sup}(C) / |D|$. To prove our claims, we exploit two properties of convex functions from [11]:

(1) Convexity of a function $f(x, y)$ implies that for an arbitrary dividing point x_3, y_3 of the line segment between two points x_1, y_1 and x_2, y_2, we have $\max(f(x_1, y_1), f(x_2, y_2)) \geq f(x_3, y_3)$.[4]

(2) A convex function over a given region must be continuous at every point of the region's relative interior.

The next two lemmas show that a convex function $f(x, y)$ which is minimum at $\mathrm{conf}(x, y) = c$ has the properties required by Observation 3.4, where γ from the observation is set to the expected confidence value.

LEMMA 3.6: For a convex function $f(x, y)$ which is minimum at $\mathrm{conf}(x, y) = c$, $f(x, y)$ is (1) monotone in $\mathrm{conf}(x, y)$ for fixed y, so long as $\mathrm{conf}(x, y) \geq c$, and (2) monotone in y when $\mathrm{conf}(x, y) = A$ for any constant $A \geq c$.

Proof: For case (1), when y is fixed to a constant Y, $\mathrm{conf}(x, Y)$ for $\mathrm{conf}(x, Y) \geq c$ represents a horizontal line segment that extends from point $\mathrm{conf}(x, Y) = c$ leftward. The value of $\mathrm{conf}(x, Y)$ is clearly anti-monotone in x. Because f is also anti-monotone in x in this region as a consequence of its convexity, it follows that $f(x, Y)$ is monotone in $\mathrm{conf}(x, Y)$[5].

For case (2), assume the claim is false. This implies there exists a vector v defined by $\mathrm{conf}(x, y) = A$ for some constant $A \geq c$ along which f is not monotone in y. That is, there exist two points (x_1, y_1) and (x_2, y_2) along v where $f(x_1, y_1) > f(x_2, y_2)$ yet $y_1 < y_2$ (see Figure 2). If f were defined to be minimum at $(0, 0)$, then this would contradict the fact that f is convex. But since f as well as v are undefined at this point, another argument is required. Consider then some sufficiently small non-zero value δ such that $x_2 - \delta \geq 0$ and $f(x_1, y_1) > f(x_2 - \delta, y_2)$. Because f is convex and continuous in

[3] We are making some simplifications: these functions are actually defined in terms of a vector defining the class distribution after a binary split. We are restricting attention to the case where there are only two classes ($\neg C$ and C which correspond to x and y respectively). The binary split in our case is the segmentation of data-set D made by testing the antecedent condition A of the rule.

[4] This well-known property of convex functions is sometimes given as the definition of a convex function, e.g. [16]. While this property is necessary for convexity, it is not sufficient. The proofs of convexity for gini, entropy, and chi-squared value in [16] are nevertheless valid for the actual definition of convexity since they show that the second derivatives of these functions are always non-negative, which is necessary and sufficient [11].

[5] We are not being completely rigorous due to the bounded nature of the convex region over which f is defined. For example, the point $\mathrm{conf}(x, Y) = c$ may not be within this bounded region since x can be no greater than $|D|$. Verifying that these boundary conditions do not affect the validity of our claims is left as an exercise.

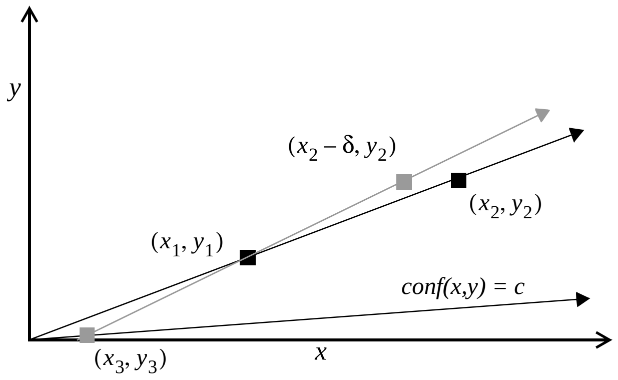

Figure 2. Illustration of case (2) from Lemma 3.6.

its interior region, such a value of δ is guaranteed to exist unless $x_2 = 0$, which is a trivial boundary case. Now, consider the line $[(x_1, y_1), (x_2 - \delta, y_2)]$. This line must contain a point (x_3, y_3) such that x_3 and y_3 are non-negative, and one or both of x_3 or y_3 is non-zero. But because f is convex and minimum at (x_3, y_3), we have that $f(x_1, y_1) \le f(x_2 - \delta, y_2)$, which is a contradiction. □

LEMMA 3.7: For a convex function $f(x, y)$ which is minimum at $\mathrm{conf}(x, y) = c$, $f(x, y)$ is (1) anti-monotone in $\mathrm{conf}(x, y)$ for fixed y, so long as $\mathrm{conf}(x, y) \le c$, and (2) monotone in y when $\mathrm{conf}(x, y) = A$ for any constant $A \le c$.

Proof: Similar to the previous. □

The previous two lemmas and Observation 3.4 lead immediately to the following theorem, which formalizes the fact that mining the upper and lower support-confidence borders identifies the optimal rules according to metrics such as entropy gain, gini, and chi-squared value. Conveniently, an algorithm specifically optimized for mining the upper border can be used without modification to mine the lower border by simply negating the consequent of the given instance, as stated by the subsequent lemma.

THEOREM 3.8: Given instance $I = \langle U, D, \le_t, C, N \rangle$, if $\le_t$ is defined over the values given by a convex function $f(x, y)$ over rules $A \to C$ where:
(1) $x = \sup(A) - \sup(A \cup C)$ and $y = \sup(A \cup C)$, and
(2) $f(x, y)$ is minimum at $\mathrm{conf}(x, y) = \sup(C) / |D|$,
then an I-optimal rule appears in either (a) any I_{sc} optimal set where $I_{sc} = \langle U, D, \le_{sc}, C, N \rangle$, or (b) any $I_{s\neg c}$-optimal set where $I_{s\neg c} = \langle U, D, \le_{s\neg c}, C, N \rangle$.

LEMMA 3.9: Given an instance $I_{s\neg c} = \langle U, D, \le_{s\neg c}, C, N \rangle$, any I_{sc}-optimal set for $I_{sc} = \langle U, D, \le_{sc}, \neg C, N \rangle$ (where $\neg C$ evaluates to true only when C evaluates to false) is also an $I_{s\neg c}$-optimal set.

Proof Idea: Note that $\mathrm{conf}(A \to \neg C) = 1 - \mathrm{conf}(A \to C)$. Thus, maximizing the confidence of $A \to \neg C$ minimizes the confidence of $A \to C$. □

Before ending this section we consider one practical issue -- that of result visualization. Note that the support-confidence borders as displayed in Figure 1 provide an excellent means by which optimal sets of rules may be visualized. Each border clearly illustrates the trade-off between the support and confidence. Additionally, one can imagine the result visualizer color-coding points along these borders that are optimal according to the various interestingness metrics, e.g. blue for Laplace value, red for chi-squared value, green for entropy gain, and so on. The result of modifying minimum support or confidence on the optimal rules could be displayed in real-time as the user drags a marker along either axis in order to specify a minimum bound.

3.3 Practical Implications for Mining Optimal Conjunctions

In this section we present and evaluate an algorithm that efficiently mines an optimal set of conjunctions according to $\le_{sc}$ (and $\le_{s\neg c}$ due to Lemma 3.9) from many real-world categorical data-sets, without requiring any constraints to be specified by the user. We also demonstrate that the number of rules produced by this algorithm for a given instance is typically quite manageable -- on the order of a few hundred at most. We address the specific problem of mining optimal conjunctions within categorically valued data, where each condition in U is simply a test of whether the given input record contains a particular attribute/value pair, excluding values from a designated class column. Values from the designated class column are used as consequents. While our algorithm requires no minimums on support or confidence, if they are specified, they can be exploited for better performance. Space constraints prohibit a full explanation of the workings of this algorithm, so we highlight only the most important features here. A complete description appears in an extended draft [7].

The algorithm we use is a variant of Dense-Miner from [6], which is a constraint-based rule miner suitable for use on large and dense data-sets. In Dense-Miner, the rule mining problem is framed as a set-enumeration tree search problem [21] where each node of the tree enumerates a unique element of 2^U. Dense-Miner returns every rule that satisfies the input constraints, which include minimum support and confidence. We modified Dense-Miner to instead maintain only the set of rules R that are potentially optimal at any given point during its execution. Whenever a rule r is enumerated by a node and found to satisfy the input constraints, it is compared against every rule presently in R. If r is better than or incomparable to every rule already in R according to the partial order, then rule r is added to R. Also, any rule in R that is worse than r is removed. Given this policy, assuming the tree enumerates every subset of U, upon termination, R is an optimal set.

Because an algorithm which enumerates every subset would be unacceptably inefficient, we use pruning strategies that greatly reduce the search space without compromising completeness. These strategies use Dense-Miner's pruning functions (appearing in Appendix B), which bound the confidence and support of any rule that can be enumerated by a descendent of a given node. To see how these functions are applied in our variant of the algorithm, consider a node g with support bound s and confidence bound c. To see if g can be pruned, the algorithm determines if there exists a rule r in R such that $r_i \le_{sc} r$, where r_i is some imaginary rule with support s and confidence c. Given such a rule, if any descendent of g enumerates an optimal rule, then it must be equivalent to r. This equivalence class is already represented in R, so there is no need to enumerate these descendents, and g can be pruned.

This algorithm differs from Dense-Miner in only two additional ways. First, we allow the algorithm to perform a set-oriented best-first search of the tree instead of a purely breadth-first search. Dense-miner uses a breadth-first search since this limits the number of database passes required to the height of the search tree. In the context of optimized rule mining, a breadth-first strategy can be inefficient because pruning improves as better rules are found, and good rules sometimes arise only at the deeper levels. A pure best-first search requires a database pass for each node in the tree, which would be unacceptable for large data-sets. Instead, we process several of the best nodes (at most 5000 in our implementation) with each database pass in order to reduce the number of database passes while still substantially reducing the search space. For this purpose, a node is better than another if the rule it enumerates has a higher confidence value.

The remaining modification is the incorporation of inclusive pruning as proposed by Webb [23]. This pruning strategy avoids enumerating a rule when it can be determined that its antecedent can be extended with an additional condition without affecting the support of the rule. In the absence of item constraints, this optimization prunes many rules that are either non-optimal or equivalent to some other optimal rule to be enumerated. Unfortunately, when there are item constraints in N (e.g. "rules must contain fewer than k conditions"), this pruning strategy cannot be trivially applied without compromising completeness, so it must be disabled. Full details of this pruning strategy are provided in Appendix B.

We evaluated our algorithm on the larger of the categorical data-sets from the Irvine machine learning database repository,[6] including chess, mushroom, letter, connect-4, and dna. We also used the pums data-set from [6] which is compiled from census data (a similar data-set was used in [8]). For the Irvine data-sets, we used each value of the designated class column as the consequents. For the pums data-set, we used the values of the RELAT1 column (13 in all)[7]. Each of these data-sets is known to be difficult for constraint-based rule mining algorithms such as Apriori, even when specifying a strong minimum support constraint [4,6,8].

Experiments were performed on an IBM IntelliStation with 400 MHZ Intel Pentium-II processor and 128 MBytes of main memory. Execution time and the number of rules returned by the algorithm appear in Table 1; characteristics of each data-set appear in Table 2. For the Irvine data-sets, with the exception of connect-4, our algorithm identified an optimal set of rules within 30 seconds in every case, with many runs requiring less than 1 second. Connect-4 was the most difficult of the data-sets for two reasons. First, it has substantially more records and more columns than many of the other data-sets. But a stronger contributor to this discrepancy was the fact that rules with high confidence within the connect-4 data-set have very low support. For example, with the "tie" class as the consequent, rules with 100% confidence have a support of at most 14 records. This property greatly reduces pruning effectiveness, resulting in almost one hour of execution time given this consequent. In cases like these, modest settings of the minimum support or confidence constraint can be used to improve runtime considerably. For example, a minimum support of 676 records (1% of the data-set) reduces execution time to 6 minutes.

The number of rules in each optimal set was on the order of a few hundred at most. Of the Irvine data-sets, connect-4 contained the most optimal rules, with 216 for win, 171 for tie, and 465 for lose. We plot the upper support-confidence border for each of these consequents in Figure 3. Rule support is normalized according to consequent support so that each border ranges from 0 to 100% along the x axis.

Data-set	**Consequent**	**Time (sec)**	**# of Rules**
chess	win	<1	60
	nowin	<1	41
connect-4	win	642	216
	draw	3066	171
	loss	1108	465
letter	A-Z	18	322
dna	EI	20	9
	IE	23	15
	N	<1	9
mushroom	poisonous	<1	12
	edible	<1	7
pums	1	740	324
	2	204	702
	3	509	267
	4	174	152
	5	46	91
	6	19	81
	7	50	183
	8	270	210
	9	843	383
	10	572	424
	11	88	165
	12	12	11
	13	22	102

Table 1. Execution time and number of rules returned.

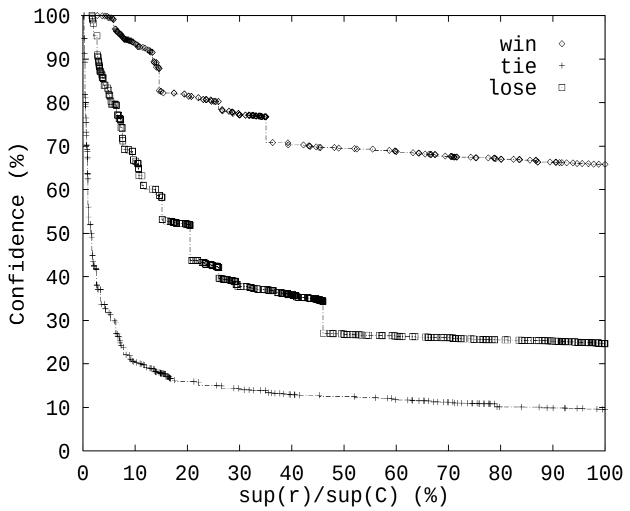

Figure 3. Upper support/confidence borders for Connect-4.

4. PC-Optimality

4.1 Definition

While sc-optimality is a useful concept, it tends to produce rules that primarily characterize only a specific subset of the population of interest (by *population of interest*, we mean the set of records for which condition C evaluates to true). In this section, we propose another partial order with the goal of remedying this deficiency. First, the *population* of a rule $A \rightarrow C$ is simply the set of records from data-set D for which both A and C evaluate to true. We denote the population of a rule r as $\mathrm{pop}(r)$. Clearly then, $|\mathrm{pop}(r)| = \mathrm{sup}(r)$. A rule which contains some record t within its population is said to *characterize* t.

[6] http://www.ics.uci.edu/~mlearn/MLRepository.html

[7] This data-set is available in the form used in these experiments through: http://www.almaden.ibm.com/cs/quest
The values 1-13 for the RELAT1 column correspond to items 1-13 in the apriori binary format of this data.

Data-set	# of Rows	# of Columns
chess	3,196	37
connect-4	67,557	43
dna	3,124	61
letter	20,000	17
mushroom	8,124	23
pums	49,046	74

Table 2. Number of rows and columns in each data-set.

Consider now the partial order $\leq_{pc}$ on rules where $r_1 <_{pc} r_2$ if and only if:

- $\text{pop}(r_1) \subseteq \text{pop}(r_2) \wedge \text{conf}(r_1) < \text{conf}(r_2)$, or
- $\text{pop}(r_1) \subset \text{pop}(r_2) \wedge \text{conf}(r_1) \leq \text{conf}(r_2)$.

Two rules are equivalent according to this partial order if their population sets are identical and their confidence values are equal. One can analogously define the partial order $\leq_{p\neg c}$ where $r_1 <_{p\neg c} r_2$ if and only if:

- $\text{pop}(r_1) \subseteq \text{pop}(r_2) \wedge \text{conf}(r_1) > \text{conf}(r_2)$ or
- $\text{pop}(r_1) \subset \text{pop}(r_2) \wedge \text{conf}(r_1) \geq \text{conf}(r_2)$.

4.2 Theoretical Implications

It is easy to see that $\leq_{sc}$ is implied by $\leq_{pc}$ (and $\leq_{s\neg c}$ by $\leq_{p\neg c}$), so all the claims from Section 3 also hold with respect to pc-optimality. Note that $\leq_{pc}$ results in more incomparable rule pairs than $\leq_{sc}$ due to the population subsumption requirement. This implies there will be many more rules in a pc-optimal set compared to an sc-optimal set. The consequence of these additional rules, as formalized by the observation below, is that a pc-optimal set always contains a rule that is optimal with respect to any of the previous interestingness metrics, and further with respect to any constraint that requires the rules to characterize a given subset of the population of interest.

OBSERVATION 4.1: Given an instance $I = \langle U, D, \leq_t, C, N \rangle$ where
(1) $\leq_t$ is implied by $\leq_{pc}$, and
(2) N has a constraint n on rules r stating that $P \subseteq \text{pop}(r)$ for a given subset P of the population of interest,
an I-optimal rule appears in any I_{pc}-optimal set where $I_{pc} = \langle U, D, \leq_{pc}, C, N - \{n\} \rangle$.

We note that this generalization of sc-optimality is quite broad. Given the large number of rules that can appear in a pc-optimal set (see next subsection), a more controlled generalization might be desirable. One idea is to have the rule set support any constraint that requires a rule to characterize a given member t (in place of a subset P) of the population of interest. This would still guarantee a broad characterization, but potentially with much fewer rules. This notion designates a rule r as uninteresting if there exists some set of rules R such that (1) each rule in R satisfies the input constraints, (2) every member of $\text{pop}(r)$ is characterized by at least one rule in R and (3) each rule in R is equal to or better than r according to $\leq_{sc}$. (Note that pc-optimality designates a rule as uninteresting only if there exists such a set R containing exactly one rule.) This notion of uninterestingness cannot, to our knowledge, be expressed using a partial order on rules. Instead, we are examining how the concept of pc-optimality combined with constraints can successfully express (and exploit) this notion.

4.3 Practical Implications for Mining Optimal Conjunctions

Producing an algorithm that can efficiently mine an optimal set of conjunctions with respect to $\leq_{pc}$ may seem an impossible proposition in large data-sets due to the need to verify subsumption between rule populations over many rules. However, we will show that we can verify the relationships induced by the partial order "syntactically" in many cases; that is, by checking the conditions of the rules along with support and confidence, instead of examining the set of database records that comprise their populations. Keep in mind that in this subsection we are restricting attention to mining optimal conjunctions. In typical formulations of mining optimized disjunctions (e.g. [12,20]), syntactic checks for population subsumption simply require geometrically comparing the regions represented by the base conditions.

DEFINITION (A-MAXIMAL): A rule $A_1 \rightarrow C$ is *a-maximal* if there is no rule $A_2 \rightarrow C$ such that $A_1 \subset A_2$ and $\text{sup}(A_1 \rightarrow C) = \text{sup}(A_2 \rightarrow C)$.

Note that the definition of an a-maximal rule is such that it need not satisfy the input constraints. The intuition behind a-maximality ("antecedent" maximality) is that an a-maximal rule cannot have its antecedent enlarged without strictly reducing its population.

Given a rule r, we denote an a-maximal rule that has the same population of r as $\text{amax}(r)$. Lemma 4.2a implies there is only one such rule, which means that $\text{amax}()$ is in fact a function. The purpose of this function is to provide a concise, canonical description of a rule's population that allows for efficiently checking the subsumption condition (Lemma 4.2). In turn, this provides a syntactic method for comparing rules with $\leq_{pc}$ (Theorem 4.3).

LEMMA 4.2 A: $\text{pop}(r_1) = \text{pop}(r_2) \Leftrightarrow \text{amax}(r_1) = \text{amax}(r_2)$
B: $\text{pop}(r_1) \subset \text{pop}(r_2) \Leftrightarrow \text{amax}(r_1) \supset \text{amax}(r_2)$

Proof: The $\Leftarrow$ direction is immediate for both cases, so we prove the $\Rightarrow$ direction. Suppose the claims are false and consider the rule r_3 which is formed by taking the union of the antecedents from $\text{amax}(r_1)$ and $\text{amax}(r_2)$. We establish each claim by contradiction.

For claim A, if $\text{pop}(r_1) = \text{pop}(r_2)$, then clearly r_3 has the same population as r_1 and r_2. Since we assume the claim is false, we have that $\text{amax}(r_1)$ is different than $\text{amax}(r_2)$. Given this, either $\text{amax}(r_1) \subset \text{amax}(r_3)$ or $\text{amax}(r_2) \subset \text{amax}(r_3)$. But since all three rules have the same population, this leads to the contradiction that either $\text{amax}(r_1)$ or $\text{amax}(r_2)$ is not a-maximal.

For claim B, if $\text{pop}(r_1) \subset \text{pop}(r_2)$, then $\text{pop}(r_3) = \text{pop}(r_1)$. As a consequence, we must have that $\text{amax}(r_1) = \text{amax}(r_3)$ by claim A from above. Since we assume the claim is false, we must in addition have that $\text{amax}(r_1) \subset \text{amax}(r_2)$ or $\text{amax}(r_1) = \text{amax}(r_2)$. The case where they are equal is an immediate contradiction due to claim A. For the case where $\text{amax}(r_1) \subset \text{amax}(r_2)$, note that $\text{amax}(r_3) \subset \text{amax}(r_2)$, which contradicts the fact that $r_3 = \text{amax}(r_1) \cup \text{amax}(r_2)$. $\square$

THEOREM 4.3 A: $r_1 <_{pc} r_2$ if and only if:
(1) $\text{conf}(r_1) < \text{conf}(r_2)$ and $\text{amax}(r_1) \supseteq \text{amax}(r_2)$, or
(2) $\text{conf}(r_1) \leq \text{conf}(r_2)$ and $\text{amax}(r_1) \supset \text{amax}(r_2)$.
B: $r_1 =_{pc} r_2$ if and only if
$\text{conf}(r_1) = \text{conf}(r_2)$ and $\text{amax}(r_1) = \text{amax}(r_2)$.

These facts cannot be trivially applied in an efficient manner since computing $\text{amax}(r)$ requires scanning the data-set (an algorithm for this purpose appears in Figure 4). Instead, we describe pruning optimizations which can be incorporated into a rule miner to avoid generating many rules that are not pc-optimal. A post-processing phase then computes $\text{amax}(r)$ for every rule identified in one final pass over the database, and this information is used to efficiently identify any remaining non-optimal rules.

```
INPUT: a rule $r$ and a data-set $D$
OUTPUT: $\text{amax}(r)$
1. Find the first record in $D$ that is a member of the population of $r$. Initialize a set $A$ to contain those conditions of $U$ that evaluate to true on this record, but are not already in $r$.
2. For each remaining record in $D$ which is in the population of $r$, remove any condition in $A$ which evaluates to false on this record.
3. Add the conditions in $A$ to the antecedent of $r$ and return the result.
```

Figure 4. Algorithm for computing $\text{amax}(r)$.

```
INPUT: $I_{pc} = \langle U, D, \leq_{pc}, C, N \rangle$
OUTPUT: An $I_{pc}$-optimal set.
1. Find all rules with positive improvement among those satisfying the input constraints. Call this set of rules $R$.
2. For each rule $r$ in $R$, associate $r$ with $\text{amax}(r)$ and put $\text{amax}(r)$ into a set $R_a$.
3. Remove any rule $r_1$ from $R$ if there exists some $r_2 \in R$ such that such that $r_1 <_{pc} r_2$ according to Theorem 4.3a.
4. For each rule $r_a \in R_a$, if there is more than one rule $r$ in $R$ such that $\text{amax}(r) = r_a$, then remove all but one of them from $R$.
5. Return $R$.
```

Figure 5. Algorithm for mining a pc-optimal set of conjunctions.

OBSERVATION 4.4: Given an instance $I_{pc} = \langle U, D, \leq_{pc}, C, N \rangle$, a rule $A_1 \rightarrow C$ cannot be I_{pc}-optimal if there exists some rule $A_2 \rightarrow C$ where $A_2 \subset A_1$, A_2 satisfies the input constraints, and $\text{conf}(A_2) > \text{conf}(A_1)$.

Using the terminology from [6], this observation simply states that a pc-optimal rule must have a non-negative *improvement* value. We can in fact require that improvement be positive since we only need one member of each equivalence class. The Dense-Miner algorithm already provides pruning functions that bound the improvement value of any rule derivable from a given node. We thus modify Dense-Miner to prune nodes whose improvement bound is 0 (see Appendix B for the simplified pruning functions that can be used for this special case). We also again exploit Webb's inclusive pruning strategy for the case where there are no item constraints in N.

We can now fully describe an algorithm for mining an I_{pc}-optimal set of rules without performing any explicit subsumption checks between rule populations (Figure 5). We use the above-described variant of Dense-Miner to implement step 1 of this algorithm. Step 3 applies Theorem 4.3a to identify non-optimal rules for removal. This step requires we first associate each rule with its a-maximal rule (step 2). To find the a-maximal rules, we apply the algorithm in Figure 4 for each rule in R, using a single shared database scan. For a data-set such as connect-4, our implementation of this step requires less than 3 seconds for a set with up to 10,000 rules. Finally, step 4 applies Theorem 4.3b in order to identify equivalent rules so that there is only one representative from each equivalence class in the returned result.

This algorithm could be simplified slightly when the input constraints N have the property that $\text{amax}(r)$ satisfies N whenever r does. In this case, the set R_a could be returned immediately following step 2 (since R_a is a set, we assume it contains no duplicates). However, some common rule constraints (e.g. "a rule must have fewer than k conditions") do not have this property.

Data-set	Consequent	Time (sec)	# of Rules
chess	win	2,821	236,735
	nowin	504	42,187
connect-4	win	19,992	178,678
	draw	18,123	119,984
	loss	34,377	460,444
letter	A-Z	65	37,850
dna	EI	64	55,347
	IE	46	49,505
	N	53	9,071
mushroom	poisonous	1	217
	edible	3	389
pumsb*	1	1,058	84,594
	2	829	33,443
	3	305	14,927
	4	770	28,553
	5	308	21,244
	6	59	5,717
	7	412	15,474
	8	428	22,992
	9	3,079	160,908
	10	3,857	175,061
	11	118	5,701
	12	12	991
	13	482	59,088

Table 3. Execution time and number of rules returned.

In practice, we find that the number of rules returned by this algorithm is considerably larger than that of the algorithm for mining sc-optimal sets. On most data-sets, rule constraints such as minimum support or confidence must be specified to control the size of the output as well as execution time. For the execution times reported in Table 3, the minimum support constraint was set to ensure that each rule's population is at least 5% of the population of interest. For the pums data-set, this constraint was not sufficiently strong to control combinatorial explosion. We therefore simplified the data-set by removing values which appear in 80% or more of the records (the resulting data-set is known as pumsb*, and was used in [5]). We conclude that pc-optimal sets of rules are useful primarily on data-sets where the density is somewhat subdued, or when the user is capable of specifying strong rule constraints prior to mining.

5. Conclusions

We have defined a new optimized rule mining problem that allows a partial order in place of the typical total order on rules. We have also shown that solving this optimized rule mining problem with respect to a particular partial order $\leq_{sc}$ (and in some cases its analog $\leq_{s\neg c}$) is guaranteed to identify a most-interesting rule according to several interestingness metrics including support, confidence, gain, laplace value, conviction, lift, entropy gain, gini, and chi-squared value. In practice, identifying such an optimal set of conjunctive rules can be done efficiently, and the number of rules in such a set is typically small enough to be easily browsed by an end-user. We lastly generalized this concept using another partial order $\leq_{pc}$ in order to ensure that the entire population of interest is well-characterized. This generalization defines a set of rules that contains the most interesting rule according to any of the above metrics, even if one requires this rule to characterize a specific subset of the population of interest.

These techniques can be used to facilitate interactivity in the process of mining most-interesting rules. After mining an optimal set of rules according to the first partial order, the user can examine the most-interesting rule according to any of the supported interestingness metrics without additional querying or mining of the database. Minimum support and confidence can also be modified and the effects immediately witnessed. After mining an optimal set of rules according to the second partial order, in addition to the above, the user can quickly find the most-interesting rule that characterizes any given subset of the population of interest. This extension overcomes the deficiency of most optimized rule miners where much of the population of interest may be poorly characterized or completely uncharacterized by the mined rule(s).

Acknowledgments

We are indebted to Ramakrishnan Srikant and Dimitrios Gunopulos for their helpful suggestions and assistance.

References

[1] Agrawal, R.; Imielinski, T.; and Swami, A. 1993. Mining Associations between Sets of Items in Massive Databases. In *Proc. of the 1993 ACM-SIGMOD Int'l Conf. on Management of Data*, 207-216.

[2] Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, A. I. 1996. Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 307-328.

[3] Ali, K.; Manganaris, S.; and Srikant, R. 1997. Partial Classification using Association Rules. In *Proc. of the 3rd Int'l Conf. on Knowledge Discovery in Databases and Data Mining*, 115-118.

[4] Bayardo, R. J. 1997. Brute-Force Mining of High-Confidence Classification Rules. In *Proc. of the Third Int'l Conf. on Knowledge Discovery and Data Mining*, 123-126.

[5] Bayardo, R. J. 1998. Efficiently Mining Long Patterns from Databases. In *Proc. of the 1998 ACM-SIGMOD Int'l Conf. on Management of Data*, 85-93.

[6] Bayardo, R. J.; Agrawal, R.; and Gunopulos, D. 1999. Constraint-Based Rule Mining in Large, Dense Databases. In *Proc. of the 15th Int'l Conf. on Data Engineering*, 188-197.

[7] Bayardo, R. J. and Agrawal, R. 1999. *Mining the Most Interesting Rules*. IBM Research Report. Available from: `http://www.almaden.ibm.com/cs/quest`

[8] Brin, S.; Motwani, R.; Ullman, J.; and Tsur, S. 1997. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *Proc. of the 1997 ACM-SIGMOD Int'l Conf. on the Management of Data*, 255-264.

[9] Clark, P. and Boswell, P. 1991. Rule Induction with CN2: Some Recent Improvements. In *Machine Learning: Proc. of the Fifth European Conference*, 151-163.

[10] Dhar, V. and Tuzhilin, A. 1993. Abstract-driven pattern discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6).

[11] Eggleston, H. G. 1963. *Convexity.* Cambridge Tracts in Mathematics and Mathematical Physics, no. 47. Smithies, F. and Todd, J. A. (eds.). Cambridge University Press.

[12] Fukuda, T.; Morimoto, Y.; Morishita, S.; and Tokuyama, T. 1996. Data Mining using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization. In *Proc. of the 1996 ACM-SIGMOD Int'l Conf. on the Management of Data*, 13-23.

[13] Goethals, B. and Van den Bussche, J. 1999. A Priori Versus A Posteriori Filtering of Association Rules. In *Proc. of the 1999 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, paper 3.

[14] International Business Machines, 1996. *IBM Intelligent Miner User's Guide*, Version 1, Release 1.

[15] Mitchell, T. M. 1997. *Machine Learning*. McGraw-Hill, Inc.

[16] Morimoto, Y.; Fukuda, T.; Matsuzawa, H.; Tokuyama, T.; and Yoda, K. 1998. Algorithms for Mining Association Rules for Binary Segmentations of Huge Categorical Databases. In *Proc. of the 24th Very Large Data Bases Conf.*, 380-391.

[17] Morishita, S. 1998. On Classification and Regression. In *Proc. of the First Int'l Conf. on Discovery Science -- Lecture Notes in Artificial Intelligence* 1532:40-57.

[18] Nakaya, A. and Morishita, S. 1999. *Fast Parallel Search for Correlated Association Rules*. Unpublished manuscript.

[19] Piatetsky-Shapiro, G. 1991. Discovery, Analysis, and Presentation of Strong Rules. Chapter 13 of *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991.

[20] Rastogi, R. and Shim, K. 1998. Mining Optimized Association Rules with Categorical and Numeric Attributes. In *Proc. of the 14th Int'l Conf. on Data Engineering*, 503-512.

[21] Rymon, R. 1992. Search through Systematic Set Enumeration. In *Proc. of Third Int'l Conf. on Principles of Knowledge Representation and Reasoning*, 539-550.

[22] Srikant, R.; Vu, Q.; and Agrawal, R. 1997. Mining Association Rules with Item Constraints. In *Proc. of the Third Int'l Conf. on Knowledge Discovery in Databases and Data Mining*, 67-73.

[23] Webb, G. I. 1996. Inclusive Pruning: A New Class of Pruning Axiom for Unordered Search and its Application to Classification Learning. In *Proc. of the 1996 Australian Computer Science Conference*, 1-10.

[24] Webb, G. I. 1995. OPUS: An Efficient Admissible Algorithm for Unordered Search. *Journal of Artificial Intelligence Research*, 3:431-465.

Appendix A

Here we provide definitions for the gini, entropy gain, and chi-squared rule value functions. For a given condition A, we denote the fraction of records that satisfy the consequent condition C among those that satisfy A as $p(A)$, and the fraction of records that do not satisfy the consequent condition among those that satisfy A as $\bar{p}(A)$. Note then that:

$$p(A) = \frac{\sup(A \to C)}{\sup(A)} \text{ and } \bar{p}(A) = \frac{\sup(A) - \sup(A \to C)}{\sup(A)}$$

$$\begin{aligned}\text{gini}(A \to C) &= 1 - [p(\emptyset)^2 + \bar{p}(\emptyset)^2] \\ &\quad - \frac{\sup(A)}{|D|}(1 - [p(A)^2 + \bar{p}(A)^2]) \\ &\quad - \frac{\sup(\neg A)}{|D|}(1 - [p(\neg A)^2 + \bar{p}(\neg A)^2])\end{aligned}$$

$$\begin{aligned}\text{ent}(A \to C) &= -[p(\emptyset)\log(p(\emptyset)) + \bar{p}(\emptyset)\log(\bar{p}(\emptyset))] \\ &\quad - \frac{\sup(A)}{|D|}[p(A)\log(p(A)) + \bar{p}(A)\log(\bar{p}(A))] \\ &\quad - \frac{\sup(\neg A)}{|D|}[p(\neg A)\log(p(\neg A)) + \bar{p}(\neg A)\log(\bar{p}(\neg A))]\end{aligned}$$

$$\begin{aligned}\text{chi}^2(A \to C) &= \\ &\frac{\sup(A)[p(A) - p(\emptyset)]^2 - \sup(\neg A)[p(\neg A) - p(\emptyset)]^2}{p(\emptyset)} \\ &+ \frac{\sup(A)[\bar{p}(A) - \bar{p}(\emptyset)]^2 - \sup(\neg A)[\bar{p}(\neg A) - \bar{p}(\emptyset)]^2}{\bar{p}(\emptyset)}\end{aligned}$$

Since C and D are constant for a given instance of the problem, terms such as $p(\emptyset) = \sup(C)/|D|$ are constants. Any of the above variable terms can be expressed as functions of $x = \sup(A) - \sup(A \to C)$ and $y = \sup(A \to C)$, and hence so can the functions themselves. For example, $\sup(A) = y + x$, $\sup(\neg A) = |D| - (y + x)$, $p(A) = y/(y + x)$, $\bar{p}(A) = x/(y + x)$, and so on.

Appendix B

Set-Enumeration Tree Search.

The set-enumeration tree search framework is a systematic and complete tree expansion procedure for searching through the power set of a given set U. The idea is to first impose a total order on the elements of U. The root node of the tree will enumerate the empty set. The children of a node N will enumerate those sets that can be formed by appending a single element of U to N, with the restriction that this single element must follow every element already in N according to the total order. For example, a fully-expanded set-enumeration tree over a set of four elements (where each element of the set is denoted by its position in the ordering) appears in Figure 6.

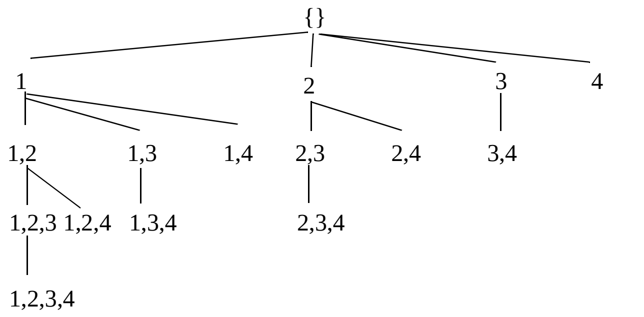

Figure 6. A complete set-enumeration tree over a 4 item set.

To facilitate pruning, we use a node representation called a group where the set enumerated by a node g is called the head and denoted $h(g)$. The set of viable elements of U which can be appended to $h(g)$ in order to form a child is called the tail and denoted $t(g)$. Making the tail elements explicit enables optimizations such as element reordering (for dynamic tree rearrangement) and tail pruning. Element reordering involves locally changing the total order on U at each node in the tree to maximize pruning effectiveness. Tail pruning involves removing elements from the tail if they cannot possibly be part of any solution in the sub-tree rooted at the node. This directly reduces the search space, and indirectly reduces the search space by improving the bounds we compute on values such as the confidence of any rule that can be enumerated by a descendent of the node.

Pruning with Confidence, Support, and Improvement

We say a rule r (which we represent using only its antecedent since the consequent it fixed) is *derivable* from a group g if $h(g) \subset r$, and $r \subseteq h(g) \cup t(g)$. By definition, any rule that can be enumerated by a descendent of g in the set-enumeration tree is also derivable from g. From [6] we have that:

- The function $f_c(x, y) = x/(x + y)$ provides an upper-bound on the confidence of any conjunctive rule derivable from a given group g, where x and y are non-negative integers such that $y \le \sup(h(g) \cup t(g) \to \neg C)$ and $x \ge \sup(h(g) \to C)$.
- The value of x from above provides an upper-bound on support.
- If the confidence bound given by f_c from above for some group g is less than or equal to the confidence of any rule r enumerated thus far such that r is a subset of $h(g)$ and r satisfies the input constraints, then the maximum improvement of any derivable rule is zero.
- If the following value is equal to zero, then the maximum improvement of any derivable rule is zero:

$$\beta = \min(\forall u \in h(g), \sup((h(g) - \{u\}) \cup \{\neg u\} \to \neg C))$$

We refer the reader to [6] for details on how to compute the above values economically given that the data-set is large, and how to heuristically reorder the elements of U in order to ensure these functions have plenty of pruning opportunities.

Inclusive Pruning

Webb's inclusive pruning strategy [23] moves a subset T of the tail of a group g into its head whenever the following fact can be established: if some solution is derivable from g, then at least one of the solutions derivable from g is a superset of T. For example, in the case of mining optimized conjunctions according to the Laplace function (and many of the other metrics we have examined including our partial orders), suppose $\sup(h(g) \to C) = \sup(h(g) \cup \{u\} \to C)$ for some element u in $t(g)$. Ignoring the effects of rule constraints, if some rule r derivable from g is optimal, then so is the rule $r \cup \{u\}$, which is also derivable from g. If one or more such elements are found in the tail of some node, instead of expanding its children, these elements can all be moved into the head to form a new node that replaces it.

Some constraints may unfortunately prohibit straightforward application of this particular inclusive pruning strategy. For example, an item constraint may disallow u from participating in a solution when combined with some other items from $h(g)$ and $t(g)$. Another problematic constraint is one which bounds the size of a rule's antecedent. Luckily, work-arounds are typically possible. For example, in the case where a bound k is specified on the number of base conditions that may appear in an antecedent, the strategy can be applied safely whenever $|h(g) \cup t(g)| \le k$.