The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Approximate and Exact Enumeration of Rule Models

Satoshi Hara,[1,2] **Masakazu Ishihata**[3]
1) Osaka University, Osaka, Japan
2) JST, ERATO, Kawarabayashi Large Graph Project
3) Hokkaido University, Hokkaido, Japan
satohara@ar.sanken.osaka-u.ac.jp, ishihata.masakazu@ist.hokudai.ac.jp

Abstract

In machine learning, rule models are one of the most popular choices when model interpretability is the primary concern. Ordinary, a single model is obtained by solving an optimization problem, and the resulting model is interpreted as the one that best explains the data. In this study, instead of finding a single rule model, we propose algorithms for enumerating multiple rule models. Model enumeration is useful in practice when (i) users want to choose a model that is particularly suited to their task knowledge, or (ii) users want to obtain several possible mechanisms that could be underlying the data to use as hypotheses for further scientific studies. To this end, we propose two enumeration algorithms: an approximate algorithm and an exact algorithm. We prove that these algorithms can enumerate models in a descending order of their objective function values approximately and exactly. We then confirm our theoretical results through experiments on real-world data. We also show that, by using the proposed enumeration algorithms, we can find several different models of almost equal quality.

1 Introduction

Background and Motivation Machine learning models are nowadays ubiquitous in society. They have a wide range of applications, in fields such as medical diagnosis (Ong, Wang, and Mu 2014), judicial decisions (Jordan and Freiburger 2015), and education (Lakkaraju et al. 2015), to name a few. While these models are expected to make accurate predictions, we often face another concern, namely the *interpretability* of the models. That is, we want humans to be able to easily understand the information contained in the models. The need for model interpretability is particularly strong in some practical situations, such as the following (Kim 2015; Doshi-Velez and Kim 2017).

- When machine learning models are used to support user decision making: If the model is a complete black-box, users have no way of verifying that the model is correct. Interpretable models allow humans to check whether or not the models are reliable. Moreover, with interpretable models, users can find and correct undesirable biases or bugs contained in the models based on task knowledge.
- When users are interested in finding interesting mechanisms underlying the data, as is often the case in data mining: Users want to obtain useful insights through fitting machine learning models to data. Highly interpretable models are essential if we are to obtain insights from them.

Of the various machine learning models, rule models, such as decision trees (Breiman et al. 1984; Quinlan 2014), rule lists (Rivest 1987; Letham et al. 2015; Angelino et al. 2017) and rule sets (Li, Shen, and Topor 2002; Lakkaraju, Bach, and Leskovec 2016), are preferred when model interpretability is the primary concern. These models describe their predictions using "if-then" rules (see Figure 1 for examples). Because of their simple structures, it is easy for humans to understand how these models behave.

In this study, instead of finding a single rule model, we enumerate several rule models with different structures that perform almost equally well. Enumerating rule models is helpful from both model reliability and data mining perspectives, for the following reasons.

- Model reliability: In many real-world tasks, we usually believe that the single model we have found is the one that best fits the data. This, however, is not always the case when we take the user's domain knowledge into account: sometimes, the model is counterintuitive, making it difficult to rely on. In such cases, enumerating a variety of models and then selecting the one that best fits the domain knowledge will help to improve the user's trust in the model.
- Data understanding: For many real-world datasets, there exist multiple rule models that can explain the data almost equally well. Thus, by considering several rule models, we can gain many more important insights into the data than would have been possible with just a single rule model.

Contributions In this study, we make the following contributions.

1. We formulate a rule model enumeration problem as enumerating rule models in descending order of their objective function values (Section 3).

Rule List	Rule Set
IF (sex = Male $\wedge$ juvenile-crimes > 0) **THEN** recidivate-within-two-years = Yes **ELSE IF** (priors > 3) **THEN** recidivate-within-two-years = Yes **ELSE** recidivate-within-two-years = No	**IF** (sex = Male $\wedge$ current-charge-degree = Felony) **OR** (priors > 3) **THEN** recidivate-within-two-years = Yes

Figure 1: Example of rule list (left) and rule set (right) learned from the same COMPAS dataset (see Section 6).

2. We propose an efficient approximate rule model enumeration algorithm (Section 4). The proposed algorithm gives a certain approximation guarantee when the targeting rule model can be learned with a guaranteed approximation ratio.
3. We propose an exact enumeration algorithm for rule models whose objective functions are submodular (Section 5). While the exact algorithm can be too slow for large problems, one of the advantages of having an exact algorithm is that it allows us to evaluate the quality of the models found by the approximate method.
4. We conduct experiments to confirm our theoretical results. We also show that, by using the proposed enumeration algorithms, we can find several different models of almost equal quality.

We note that model enumeration is an emerging topic in recent years. In our previous study (Hara and Maehara 2017), we proposed an algorithm for enumerating sparse linear models. An enumeration algorithm for decision trees was proposed by Ruggieri (2017).

Settings and Notations In this paper, for simplicity, we focus on binary classification of categorical data, but this can be naturally extended to multi-class classification. The algorithms described in this paper can also be applied to real-valued data by using discretization techniques (Srikant and Agrawal 1996; Fukuda et al. 1996; Rastogi and Shim 2002).

Suppose there are total of L items. We denote the set of all items by $[L] = \{1, 2, \ldots, L\}$ and its power set by $2^{[L]}$. Let $(x, y) \in 2^{[L]} \times \{0, 1\}$ be an input-output pair for the prediction problem. Here, $x \in 2^{[L]}$ is an itemset, and $y \in \{0, 1\}$ is the class category corresponding to x. We then consider finding a model $m \in \mathcal{F}$ that maximizes the objective function $f(m)$, where $\mathcal{F}$ is the set of feasible models. For the maximization problem $\max_{m \in \mathcal{F}} f(m)$, we say that $m \in \mathcal{F}$ is an α-approximate solution if $f(m) \geq \alpha f(m^*)$ where $\alpha \in [0, 1]$ and $m^* = \mathrm{argmax}_{m \in \mathcal{F}} f(m)$. We refer to α as an approximation ratio. For a proposition a, $\mathbb{I}(a)$ denotes the indicator of a, i.e., $\mathbb{I}(a) = 1$ if a is true, and $\mathbb{I}(a) = 0$ if a is false

2 Preliminaries

Three major types of rule models are used for building interpretable models: decision trees (Breiman et al. 1984; Quinlan 2014), rule lists (Rivest 1987; Letham et al. 2015; Angelino et al. 2017), and rule sets (Li, Shen, and Topor 2002; Lakkaraju, Bach, and Leskovec 2016). An enumeration method for decision trees has recently been studied by Ruggieri (2017), so we focus on enumerating rule lists and rule sets in this study. Here, we review some of the possible learning methods for rule models.

We first give a general formulation of rule model learning problems. Suppose there is a set of itemsets $T = \{t_j\}_{j=1}^{J} \subseteq 2^{[L]}$, where each itemset t_j indicates an association between data and its category, i.e., $t_j \subseteq x \Rightarrow y = 0$ or $t_j \subseteq x \Rightarrow y = 1$ for $j \in [J]$. Such a set T can be found, e.g., by using association rule mining (Agrawal, Imieliński, and Swami 1993; Agrawal and Srikant 1994) or emerging pattern mining (Dong and Li 1999; Dong and Bailey 2012; Komiyama et al. 2017). Then, learning the rule list or the rule set can generally be expressed as the following maximization problem:

$$\max_{m \in \mathcal{F}_T} f(m), \tag{1}$$

where $\mathcal{F}_T$ is the set of all rule models, such as rule lists and rule sets, that can be expressed using the set T. We later show examples of m and $\mathcal{F}_T$.

Rule List Rule lists are models that present their predictions in "if-then-else" format, as shown in Figure 1. A rule list of length I can be expressed using two parameters $Z = \{z_1, z_2, \ldots, z_I\}$ and $U = \{u_1, u_2, \ldots, u_I\}$ where $z_i \in \{0, 1\}$ and $u_i \in T$, as follows:

$$g(x|Z, U) = \sum_{i \in [I]} z_i \mathrm{cap}(x|U, i), \tag{2}$$

where

$$\mathrm{cap}(x|U, i) = \begin{cases} 1 & \text{if } (u_i \subseteq x) \wedge \left(\bigwedge_{i' \in [i-1]} (u_{i'} \not\subseteq x) \right), \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Here, $\mathrm{cap}(x|U, i) = 1$ means that the input x does not match the first $i - 1$ lines of the rule list, but is captured by the i-th line. The rule list model $g(x|Z, U)$ then returns the corresponding label z_i as output. We note that, usually, the last line of the rule list is just "else" without any other condition, which can be realized by setting $u_I = \emptyset$.

A recent study proposed formulating the task of learning a rule list as combinatorial problem (Angelino et al. 2017). While the original problem was formulated as a minimization problem, here we consider the following equivalent maximization problem so that the formulation to be consistent with (1).

$$\max_{Z \in \{0,1\}^I, U \in T^I} \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(y^{(n)} = g(x^{(n)}|Z, U)) - \rho I. \tag{4}$$

This is the problem of maximizing regularized classification accuracy, where $\{x^{(n)}, y^{(n)}\}_{n=1}^N$ is the training set and ρ is a regularization parameter that penalizes the length of the resulting rule list. Here, $m = (Z, U)$ and $\mathcal{F}_T = \{0,1\}^I \times T^I$ in the formulation of the problem (1).

An efficient optimization algorithm called CORELS (Angelino et al. 2017; Larus-Stone 2017) has been proposed for finding globally optimal rule lists by using several pruning techniques.

Rule Set In contrast with rule list, rule sets present their predictions in an "if-or-then" format, as shown in Figure 1. A recent study proposed formulating the task of learning a rule set as a non-monotone submodular function maximization problem (Lakkaraju, Bach, and Leskovec 2016), and presented a polynomial-time approximation algorithm for solving this maximization problem.

In this paper, for simplicity, we consider the following simplified version of the rule set learning method of Lakkaraju, Bach, and Leskovec (2016). Specifically, we focus on learning a rule set that describes the category $y = 1$. Such a rule set, of length I, can be described using $V = \{v_1, v_2, \ldots, v_I\}, v_i \in T$ as

$$g(x|V) = \begin{cases} 1 & \text{if } \exists i \in [I], v_i \subseteq x, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

This rule set can be learned by solving the following optimization problem:

$$\max_{V \subseteq T} \frac{1}{|\mathcal{N}_+|} \sum_{n \in \mathcal{N}_+} g(x^{(n)}|V), \text{ s.t. } |V| \leq I, \tag{6}$$

where $\mathcal{N}_+ := \{n; y^{(n)} = 1\}$. Here, $m = V$ and $\mathcal{F}_T = \bigcup_{i \in [I]} \binom{T}{i}$ in the formulation of the problem (1), where $\binom{T}{i} := \{V \subseteq T; |V| = i\}$.

The problem (6) is equivalent to the well-known maximum coverage problem (Feige 1998) under a cardinality constraint: $\frac{1}{|\mathcal{N}_+|} \sum_{n \in \mathcal{N}_+} g(x^{(n)}|V)$ is the fraction of positively-categorized training data that is covered by the set V. This function is known to be a monotone submodular function that can be approximated, with an approximation ratio of $\alpha = 1 - 1/e$, using a greedy algorithm (Nemhauser, Wolsey, and Fisher 1978; Hochbaum 1996). One can also use the filtered search method (Chen, Chen, and Weinberger 2015) to solve the problem (6) with an arbitrary approximation ratio α.

3 Problem Formulation

In this section, we formulate our rule model enumeration problem.

An important observation about the problem (1) is that, usually, the model m derived by solving the optimization problem uses only a fraction of the set T for the model description. For example, the problem (4) has a regularization term that penalizes the length of the rule list, so the resulting rule lists will tend to use a small subset of T. Similarly, the problem (6) imposes an explicit cardinality constraint on the model size that means the length of the resulting rule set will be at most I. To describe which itemsets in T are used for the model m, we define the "support" of the model m as $\mathrm{supp}(m) := \{t_j; t_j \text{ appears in } m\} \subseteq T$. Specifically, $\mathrm{supp}(m) = U$ if $m = (Z, U)$ is a rule list and $\mathrm{supp}(m) = V$ if $m = V$ is a rule set.

For a given subset $S \subseteq T$, we consider the problem (1) with an additional constraint $\mathrm{supp}(m) \subseteq S$:

$$\max_{m \in \mathcal{F}_T} f(m), \text{s.t. } \mathrm{supp}(m) \subseteq S. \tag{7}$$

We note that this problem is equivalent to the problem (1) with its model class restricted to $\mathcal{F}_S$. In this study, we assume that we are given a solution algorithm $\texttt{Alg}_\alpha$ to solve the problem (1), which can also be used to solve the problem (7) by replacing T with S: for a given set S, $\texttt{Alg}_\alpha$ returns one of solution models $m \in \mathcal{F}_S$ for Problem (1), which we write as $m = \texttt{Alg}_\alpha(S)$. In this study, we assume that a solution algorithm $\texttt{Alg}_\alpha$ satisfies the following property.

Assumption 1. The solution algorithm $\texttt{Alg}_\alpha : 2^T \to \mathcal{F}_T$ has the following properties:

(i) It returns an α-approximate solution.
(ii) $\forall S, S' \in 2^T$, $\texttt{Alg}_\alpha(S) = \texttt{Alg}_\alpha(S')$ if $\mathrm{supp}(\texttt{Alg}_\alpha(S)) \subseteq S' \subseteq S$.

We note that Assumption (ii) requires the output of $\texttt{Alg}_\alpha$ to be independent of redundant elements $S \setminus \mathrm{supp}(\texttt{Alg}_\alpha(S))$. Assumption 1 holds both on rule list and rule set learning algorithms presented in Section 2. If the model is a rule list, one can use CORELS (Angelino et al. 2017) as $\texttt{Alg}_\alpha$ with $\alpha = 1$. If the model is a rule set, one can use a greedy method as $\texttt{Alg}_\alpha$ with $\alpha = 1 - 1/e$ for the formulation (6), and the filtered search method (Chen, Chen, and Weinberger 2015) as $\texttt{Alg}_\alpha$ with an arbitrary α.

We now formulate our enumeration problem. Here, we let $\mathcal{M}_{\text{all}} = \{\texttt{Alg}_\alpha(S); S \in 2^T\}$ be the set of all models that can be obtained by using $\texttt{Alg}_\alpha$. The problem is then formulated as enumerating models in $\mathcal{M}_{\text{all}}$ in descending order of their objective function values.

Problem 2 (Model Enumeration)**.** Find a set of models $\mathcal{M} = \{m^{(1)}, m^{(2)}, \ldots\} \subseteq \mathcal{M}_{\text{all}}$ such that $\mathrm{supp}(m^{(k)}) \neq \mathrm{supp}(m^{(\ell)})$ and $f(m^{(k)}) \geq f(m^{(\ell)})$ for $1 \leq k < \ell$.

In the next two sections, we propose two enumeration algorithms: an approximate algorithm and an exact algorithm. We also prove the correctness of the proposed algorithms defined below, which is essential for a valid enumeration.

Definition 3 (Correctness of Enumeration)**.** The set of enumerated models $\mathcal{M} = \{m^{(1)}, m^{(2)}, \ldots\}$ is α-correct when the following two conditions hold, where $\alpha \in [0, 1]$:

- Necessity: $\forall k, \ell \in [|\mathcal{M}|]$, $k < \ell \Rightarrow f(m^{(k)}) \geq \alpha f(m^{(\ell)})$.
- Sufficiency: $\forall m \in \mathcal{M}_{\text{all}}$, $m \in \mathcal{M}$ if $\exists m' \in \mathcal{M}$ such that $\alpha f(m) > f(m')$.

The necessity is a guarantee that the models are enumerated in a descending order of their objective function values approximately ($\alpha < 1$) or exactly ($\alpha = 1$). The sufficiency is a guarantee that no models are missed while enumeration

Table 1: Correctness of the proposed enumeration algorithms, where α is an approximation ratio of $\texttt{Alg}_\alpha$.

	Approximate Algorithm (Section 4)	Exact Algorithm (Section 5)
Rule List (Problem (4))	α-correct	(Not Applicable)
Rule Set (Problem (6))	α-correct	1-correct

approximately ($\alpha < 1$) or exactly ($\alpha = 1$). Table 1 summarizes the correctness of the proposed enumeration algorithms.

4 Approximate Enumeration Algorithm

In this section, we propose an approximate algorithm for solving Problem 2. The proposed algorithm is applicable to any algorithm $\texttt{Alg}_\alpha$ with any $\alpha \in [0, 1]$ as long as it satisfies Assumption 1. This setting is practical because the problem (7) is usually a combinatorial optimization which tends to require exponential time to solve exactly, while the approximate solutions can be obtained in a polynomial time in many situations.

Before we describe the algorithm, we summarize the pros and cons of the approximate enumeration algorithm:

Pros. It is computationally efficient because it requires $\texttt{Alg}_\alpha$ to return only an approximate solution.

Cons. The enumeration is α-correct, i.e., it is correct only approximately unless $\alpha = 1$.

We now turn to details of the algorithm. The algorithm is based on Lawler's framework (Lawler 1972), which was previously used for enumerating feature selection results (Hara and Maehara 2017). In Lawler's framework, we successively computes the solution and then constructs subproblems that exclude the obtained solution.

The approximate enumeration algorithm is summarized in Algorithm 1, which is an application of our previous algorithm for sparse linear models (Hara and Maehara 2017) to rule models. In this algorithm, we maintain a heap data structure that stores pairs of a tuple and its priority value: a tuple consists of one model and two subsets of T, $(m, S, F) \in \mathcal{F}_T \times 2^T \times 2^T$, where $m = \texttt{Alg}_\alpha(S)$. The priority value of the tuple (m, S, F) is $f(m)$ and the heap pops a tuple with the maximum priority value. The set F is used to avoid inserting the same candidate set S into to the data structure twice.

The algorithm starts by computing a model without any support constraint $m = \texttt{Alg}_\alpha(T)$, and inserting the tuple $(m, T, \emptyset)$ into the heap. The algorithm then repeats the following steps until a given termination condition is met.

1. Extract a tuple (m, S, F) with the maximum priority value $f(m)$ from the heap (line 5).
2. Output m as the k-th model if it has not already been output (lines 7–8).
3. "Branch" the search space: compute $m' = \texttt{Alg}_\alpha(S \setminus \{t_j\})$ and insert the result into the heap for $t_j \in \mathrm{supp}(m)$ (lines 15–19).

Algorithm 1 Approximate Enumeration Algorithm

```
1: Compute m = Alg_α(T).
2: Insert (m, T, ∅) into the heap.
3: M ← ∅
4: for k = 1, 2, ... do
5:     Extract (m, S, F) from the heap.
6:     /* output m as the k-th solution m^(k) */
7:     if m ∉ M then
8:         M ← M ∪ {m}
9:     end if
10:    /* terminate when a certain condition is met */
11:    if Terminate(M) is True then
12:        break
13:    end if
14:    /* branch the search space */
15:    for t_j ∈ supp(m) and t_j ∉ F do
16:        Compute m' = Alg_α(S \ {t_j}).
17:        Insert (m', S \ {t_j}, F) into the heap.
18:        F ← F ∪ {t_j}
19:    end for
20: end for
```

The most important step in the algorithm is the third step, where we branch the search space. From Assumption 1(ii), if a set S' satisfies $\mathrm{supp}(m) \subseteq S' \subseteq S$, m is also the solution to $\texttt{Alg}_\alpha(S')$. We therefore branch the search space in shrinking directions $S \setminus \{t_j\}$. The set F is used to avoid enumerating the same set multiple times, which is achieved by skipping branching on any $t_j \in F$. Enumeration stops when a certain condition is met, e.g., when a certain number of models has been enumerated ($\mathrm{Terminate}(\mathcal{M}) =$ "$|\mathcal{M}| > K$"), or the quality of the solution is a factor of θ worse than the best solution found so far ($\mathrm{Terminate}(\mathcal{M}) =$ "$\min_{m \in \mathcal{M}} f(m) \leq \theta \max_{m \in \mathcal{M}} f(m)$"). We note that Algorithm 1 is *complete*: it enumerates all models $m \in \mathcal{M}_{\mathrm{all}}$ if it is not terminated (Hara and Maehara 2017).

Correctness We prove the correctness of Algorithm 1.

Theorem 4. Algorithm 1 is α-correct, where α is an approximation ratio of $\texttt{Alg}_\alpha$.

Proof. Necessity: Consider the step at which $m^{(k)}$ is extracted from the heap. Then, there are two possible situations: either $m^{(\ell)}$ is in the heap or it is not.

In the first case, $f(m^{(k)}) \geq f(m^{(\ell)})$ holds by the definition of the heap. Hence, $f(m^{(k)}) \geq \alpha f(m^{(\ell)})$ follows from $\alpha \in [0, 1]$.

In the second case, there exists a tuple (m', S', F') in the heap such that $m^{(\ell)}$ is obtained from its branches, i.e., (m', S', F') is a descent of $(m^{(\ell)}, S^{(\ell)}, F^{(\ell)})$. Here, we note that $f(m^{(k)}) \geq f(m')$ holds by the definition of the heap. Let $m^*(S')$ and $m^*(S^{(\ell)})$ be the exact maximum solutions to the problem (7) with S' and $S^{(\ell)}$, respectively. Then, $f(m^*(S')) \geq f(m^*(S^{(\ell)}))$ holds because $S^{(\ell)}$ is a branch of S', i.e., $S^{(\ell)} \subset S'$. Moreover, $f(m') \geq \alpha f(m^*(S'))$ holds from Assumption 1(i). From these results, we have

$f(m^{(k)}) \geq f(m') \geq \alpha f(m^*(S')) \geq \alpha f(m^*(S^{(\ell)})) \geq \alpha f(m^{(\ell)})$, which proves the necessity.

Sufficiency: Thanks to the completeness of Algorithm 1, $\mathcal{M} = \mathcal{M}_{\mathrm{all}}$ holds if we don't terminate the algorithm. We denote $\mathcal{M} = \{m^{(1)}, \dots, m^{(K)}\}$ and $\mathcal{M}_{\mathrm{all}} = \{m^{(1)}, \dots, m^{(W)}\}$: $\mathcal{M}$ is the first K models of $\mathcal{M}_{\mathrm{all}}$. We also denote the model with the minimum objective function value as $m^{(k)}$ where $k = \mathrm{argmin}_{k \in [K]} f(m^{(k)})$. We here assume that there exists $\ell \in [W]$ such that $\ell > k$ and $\alpha f(m^{(\ell)}) > f(m^{(k)})$. From the necessity and the fact $\ell > k$, we have $f(m^{(k)}) \geq \alpha f(m^{(\ell)})$; however, it is in apparent conflict with the assumption $\alpha f(m^{(\ell)}) > f(m^{(k)})$. Consequently, there exits no $\ell \in [W]$ such that $\ell > k$ and $\alpha f(m^{(\ell)}) > f(m^{(k)})$, which proves the sufficiency. □

Applications As a special case of Algorithm 1, it can be 1-correct if we use an exact algorithm as $\mathtt{Alg}_\alpha$ with $\alpha = 1$. For example, one can use CORELS (Angelino et al. 2017) as an exact algorithm for rule lists and the filtered search method (Chen, Chen, and Weinberger 2015) with $\alpha = 1$ for rule sets. For rule sets, Algorithm 1 is also $(1 - 1/e)$-correct if we use a greedy method as $\mathtt{Alg}_\alpha$ to solve the problem (6).

5 Exact Enumeration Algorithm

In this section, we propose an exact enumeration algorithm for solving Problem 2. We note that, although Algorithm 1 can be 1-correct if we use an exact algorithm as $\mathtt{Alg}_\alpha$, it can be computationally inefficient for rule sets when we use the filtered search method (Chen, Chen, and Weinberger 2015) as $\mathtt{Alg}_\alpha$. This is because the search space of the filtered search highly overlaps between independent runs of $\mathtt{Alg}_\alpha$ in Algorithm 1, i.e., it searches for same solution candidates many times. To overcome this computational inefficiency, we propose an exact enumeration algorithm for rule sets by extending the filtered search so that we can avoid searching same solution candidates multiple times. Here, we note that the exact enumeration is still computationally challenging even if we use the proposed algorithm because it involves solving the problem (1) exactly. To summarize, the pros and cons of the exact enumeration algorithm is as follows:

Pros. The enumeration is exact, i.e., it is 1-correct.

Cons. It is computationally expensive.

First, we make the following two assumptions. Here, for notational convenience, for a model m, we denote a model whose support is $\mathrm{supp}(m) \cup P$ by m_P for $P \subseteq T$. If $P = \{t_j\}$, we write as $m_P = m_j$.

A1. The function $f(m)$ is submodular, i.e., $f(t_j|m) \geq f(t_j|m')$ holds whenever $\mathrm{supp}(m) \subseteq \mathrm{supp}(m')$ and $t_j \in T \setminus \mathrm{supp}(m')$, where $f(t_j|m) := f(m_j) - f(m)$.

A2. If $m \in \mathcal{F}_T$, $m' \in \mathcal{F}_T$ holds whenever $\mathrm{supp}(m') \subseteq \mathrm{supp}(m) \subseteq T$.

A1 requires the objective function to be submodular, which is essential for correct filtered search. A2 requires the model space $\mathcal{F}_T$ to be down-monotone, which is true, e.g., when the model space obeys cardinality, matroid, or knapsack constraints. We note that these assumptions are all valid for the formulation of Lakkaraju, Bach, and Leskovec (2016) and for the problem (6).

In the proposed exact enumeration algorithm, which is based on the best-first search, we use the *heuristic function* $h(m) : \mathcal{F}_T \to [0, +\infty)$. Here, the heuristic function $h(m)$ is the amount how much we overestimate the quality of the model m during enumeration. In the filtered search, we assume that $h(m)$ satisfies the following *critical admissibility condition* (Chen, Chen, and Weinberger 2015).

Definition 5 (Critical Admissibility)**.** The heuristic function $h(m)$ is critically admissible if and only if, for all $m \in \mathcal{F}_T$, it satisfies

CA1. $h(m) = 0$, if $f(t_j|m) \leq 0$ or $m_j \notin \mathcal{F}_T, \forall t_j \in T$,

CA2. $h(m) \geq \max_{P \subseteq T : m_P \in \mathcal{F}_T} \sum_{t_j \in P} f(t_j|m)$, otherwise.

If the value of the heuristic function $h(m)$ is zero, this means that the model m cannot be improved by the addition of an itemset t_j (CA1), which implies that the model m is locally optimal. On the other hand, a positive heuristic function $h(m)$ implies that the model m can be improved without violating any constraints (CA2). Examples of critically admissible heuristic functions can be found in Chen, Chen, and Weinberger (2015).

The proposed exact enumeration algorithm is summarized in Algorithm 2. Intuitively, we search for local optima using the heuristic function, and output them in descending order of their objective function values. Throughout the algorithm, we maintain a heap data structure that stores pairs of a model $m \in \mathcal{F}_T$ and its priority value. The priority value of the model m is $f(m) + h(m)$ and the heap pops a model with the maximum priority value.

The algorithm starts by inserting an empty set to the heap as the initial model. Then, the algorithm repeats the following steps until a given termination condition is met.

1. Extract a model m with the maximum priority value $f(m) + h(m)$ from the heap (line 4).
2. Output m as the k-th model if $h(m) = 0$ (lines 5–7).
3. "Branch" the search space: insert m_j into the heap for $j > \max\{j; t_j \in \mathrm{supp}(m)\}$ (lines 14–16).

The most important step in the algorithm is the third step, where we branch the search space. The heuristic function $h(m)$ is positive as long as the model m can be improved by adding some itemset to the model. We therefore add each itemset t_j to the current model m in turn and insert the resulting model m_j into the heap.

Correctness We prove the correctness of Algorithm 2. The correctness follows from the fact that a subumodular function is bounded above by its modular approximation.

Lemma 6 (Nemhauser and Wolsey 1978)**.** For any submodular function f and for any model $m \in \mathcal{F}_T$, we have

$$f(m) \leq f(m') + \sum_{t_j \in \mathrm{supp}(m) \setminus \mathrm{supp}(m')} f(t_j|m'), \quad (8)$$

Algorithm 2 Exact Enumeration Algorithm

Insert the initial model $m = \emptyset$ into the heap.
$\mathcal{M} \leftarrow \emptyset$
while the heap is not empty **do**
  Extract m from the heap.
  if $h(m) = 0$ **then**
    /* *output m as the k-th solution $m^{(k)}$* */
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
    /* *terminate when a certain condition is met* */
    if $\mathrm{Terminate}(\mathcal{M})$ is True **then**
      break
    end if
  else
    /* *branch the search space* */
    for $j = \max\{j; t_j \in \mathrm{supp}(m)\} + 1, \ldots, J$ **do**
      Insert m_j into the heap.
    end for
  end if
end while

whenever $\mathrm{supp}(m') \subseteq \mathrm{supp}(m) \subseteq T$.

Theorem 7. Algorithm 2 is 1-correct.

Proof. Necessity: Recall that $h(m^{(k)}) = h(m^{(\ell)}) = 0$ holds for enumerated models $m^{(k)}$ and $m^{(\ell)}$. Consider the step at which $m^{(k)}$ is output. Then, there are two possible situations: either $m^{(\ell)}$ is in the heap or it is not.

In the first case, $f(m^{(k)}) \geq f(m^{(\ell)})$ holds by the definition of the heap.

In the second case, there exists an m' in the heap such that $m^{(\ell)}$ is obtained from branches of m', which is assured by Assumption A2. Here, we note that $f(m^{(k)}) \geq f(m') + h(m')$ holds by the definition of the heap. Moreover, because $m^{(\ell)}$ is branched from m', $\mathrm{supp}(m') \subset \mathrm{supp}(m^{(\ell)})$ holds and we have that

$$\begin{aligned} f(m^{(\ell)}) &\leq f(m') + \sum_{t_j \in \mathrm{supp}(m^{(\ell)}) \setminus \mathrm{supp}(m')} f(t_j|m') \\ &\leq f(m') + \max_{P \subseteq T : m_P \in \mathcal{F}_T} \sum_{t_j \in P} f(t_j|m') \\ &\leq f(m') + h(m'), \end{aligned} \tag{9}$$

which implies $f(m^{(k)}) \geq f(m^{(\ell)})$. Here, we have used Lemma 6 for the first inequality and the critical admissibility of the heuristic function for the last inequality.

Sufficiency: Let $k = \mathrm{argmin}_{k \in [|\mathcal{M}|]} f(m^{(k)})$. Consider the step at which $m^{(k)}$ is extracted from the heap. If $f(m) > f(m^{(k)})$, the search must reach the model m before step k because of the necessity and the fact that Algorithm 2 is complete, i.e., the search graph of Algorithm 2 is strongly connected. Hence, $m \in \mathcal{M}$. □

6 Experiments

In this section, we conduct experiments to evaluate the proposed algorithms. Specifically, we confirm the correctness of the proposed algorithms, and present some advantages of model enumeration. In the experiments, we considered the following three settings: (S1) Rule List (1-correct) that uses Algorithm 1 with CORELS as $\texttt{Alg}_\alpha$; (S2) Rule Set ($(1 - 1/e)$-correct) that uses Algorithm 1 with a greedy method as $\texttt{Alg}_\alpha$; (S3) Rule Set (1-correct) that uses Algorithm 2. In the experiments, Algorithm 1 was implemented in Python 3.5, while Algorithm 2 was implemented in C. To evaluate the algorithms, we used two classification datasets of categorical data: COMPAS (Larson et al. 2016) and Mushroom (Lichman 2013).

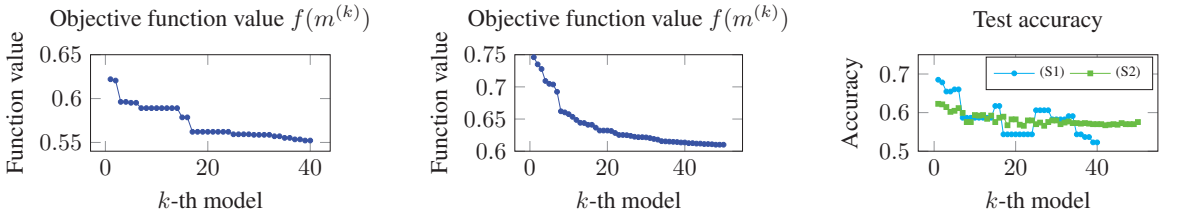
COMPAS Dataset We used COMPAS dataset distributed at the github repository (Larus-Stone 2017). It comprises 19 categorical attributes of individual people, relating their criminal history, with a total of 6,489 training samples and 721 test samples. The task is binary classification, where the positive category $y = 1$ indicates that the individual recidivate within two years.

For learning rule lists with CORELS, we used the predefined set of itemsets T in the github repository which consists of 155 itemsets. For learning rule sets, we prepared another set $T_{\mathrm{P}} \subseteq T$ where if $t \in T_{\mathrm{P}}$ then $t \subseteq x \wedge y = 1$ is held for at least 50% of the samples satisfying $t \subseteq x$ in the training set, i.e., $t \subseteq x$ indicated $y = 1$ with 50% confidence. We used T_{P} instead of T because we were interested in characterizing the positive category in the formulation (6), and none of the itemsets other than T_{P} were useful for this.

Figure 2 shows the results on the two settings (S1) and (S2). The result on (S3) is omitted because it was identical with (S2). Examples of learned rule models are shown in Figure 1. For CORELS, we used the configurations recommended in the github repository with regularization parameter $\rho = 0.015$. We set the length of the rule sets to be $I = 2$. Rule list enumeration stopped after 40 models because at that point the heap became empty, i.e., $|\mathcal{M}_{\mathrm{all}}| = 40$. For the rule sets, we enumerated the top-50 models. The figures show that the models were enumerated in descending order of their objective function values both on (S1) and (S2). We note that, although only $(1 - 1/e)$-correctness was expected to the setting (S2), it attained the exact enumeration in this experiment (Figure 2(b)), and hence its result got identical with (S3). These results confirm the correctness of the proposed algorithms. Figure 2(c) shows that, for both (S1) and (S2), first few models attained comparative test accuracies with the optimal models. That is, there were several different models of almost equal quality.

Mushroom Dataset The Mushroom dataset comprises 22 categorical attributes of 8,124 different mushrooms. The task is binary classification into the categories *poisonous* (positive, $y = 1$) and *edible* (negative, $y = 0$). For the purpose of evaluation, we randomly split the samples into 6,499 (80%) training samples and 1,625 (20%) test samples.

To prepare the set of itemsets T, we used emerging pattern mining (Komiyama 2017) with winning rate $= 0.9$. By applying this technique to the positive category, we obtained a set T_{P} with $|T_{\mathrm{P}}| = 69$. Here, if $t \in T_{\mathrm{P}}$ then $t \subseteq x \wedge y = 1$

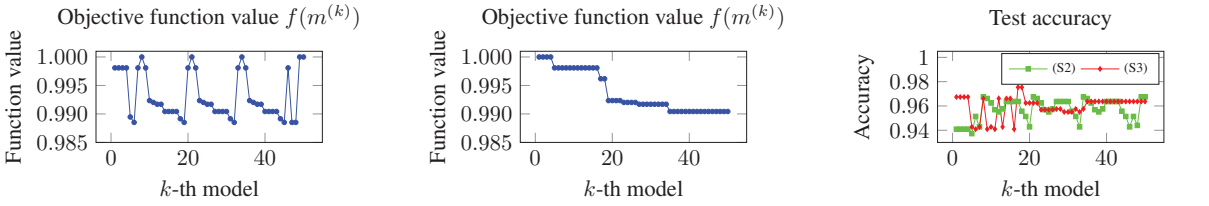


(a) Setting (S1): Rule List (1-correct)

(b) Setting (S2): Rule Set ($(1-1/e)$-correct)

(c) Test accuracies on (S1) and (S2)

Figure 2: [Results for COMPAS dataset] Figures (a) and (b) are the results on the two setting (S1) Rule List (1-correct) that uses Algorithm 1 with CORELS as $\texttt{Alg}_\alpha$, and (S2) Rule Set ($(1-1/e)$-correct) that uses Algorithm 1 with a greedy method as $\texttt{Alg}_\alpha$, respectively. Figure (c) is the test accuracy of enumerated models.



(a) Setting (S2): Rule Set ($(1-1/e)$-correct)

(b) Setting (S3): Rule Set (1-correct)

(c) Test accuracies on (S2) and (S3)

Figure 3: [Results for Mushroom dataset] Figures (a) and (b) are the results on the two settings (S2) Rule Set ($(1-1/e)$-correct) that uses Algorithm 1 with a greedy method as $\texttt{Alg}_\alpha$, and (S3) Rule Set (1-correct) that uses Algorithm 2, respectively. Figure (c) is the test accuracy of the enumerated models using the exact algorithm.

is held for more than 90% of the samples satisfying $t \subseteq x$ in the training set. We then used T_{P} to learn rule lists.

Figure 3 shows the results on (S2) and (S3). The result on (S1) is omitted because there was a memory issue in CORELS: too much memory was required and the optimal rule list was not found. In the experiment, we set the length of the rule sets to be $I = 4$. Figure 3(a) shows that the output of (S2) were not enumerated in an exact order. However, it is important to note that the $(1-1/e)$-correctness was still kept as expected. By contrast, in (S3), the models were enumerated in an exact order, as shown in Figure 3(b). From the results, we observed three important findings.

1. There were four models whose objective function values were equally one. That is, these models could explain the training data equally well. This result implies that, by focusing only on a single optimal model, we may overlook other possible explanations of data, which is not favorable from the data mining perspective. The result shows that we can aid this risk by enumerating models.
2. 40 models out of 50 enumerated models coincided between the outputs of (S2) and (S3). That is, the approximate enumeration (S2) could goodly mimic the output of the exact enumeration (S3). This result indicates that the approximate enumeration can be a pragmatic alternative of the exact enumeration when the problem size is large and the exact enumeration takes too much time.
3. The enumerated models sometimes attained better test accuracy than the optimal models, as shown in Figure 3(c).

We note that the observations 1 and 3 show advantages of model enumeration that we can obtain several different models of almost equal quality. Obtaining these models can be useful when (i) users want to choose a model that is particularly suited to their task knowledge, or (ii) users want to obtain several possible mechanisms that could be underlying the data to use as hypotheses for further scientific studies.

7 Conclusion

In this paper, we have proposed algorithms for enumerating rule lists and rule sets with different supports. In addition, we have proved that these algorithms can enumerate the models in descending order of their objective function values, both approximately and exactly. The experimental results then confirmed the correctness of the enumeration algorithms. In the experiments, we also found that, by using the enumeration algorithms, we were able to find several different models of almost equal quality.

One important problem remains open. That is, how one can find a good model out of enumerated ones. Naively checking all the models would be practically too exhaustive for users. It is therefore important to design an efficient strategy for identifying a single good model.

Acknowledgments

This work was partly supported by JST ERATO Grant Number JPMJER1201, Japan, and JSPS KAKENHI(S) 15H05711.

References

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, 487–499.

Agrawal, R.; Imieliński, T.; and Swami, A. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 207–216.

Angelino, E.; Larus-Stone, N.; Alabi, D.; Seltzer, M.; and Rudin, C. 2017. Learning certifiably optimal rule lists for categorical data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 35–44.

Breiman, L.; Friedman, J.; Stone, C. J.; and Olshen, R. A. 1984. *Classification and Regression Trees*. CRC press.

Chen, W.; Chen, Y.; and Weinberger, K. 2015. Filtered search for submodular maximization with controllable approximation bounds. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 156–164.

Dong, G., and Bailey, J. 2012. *Contrast Data Mining: Concepts, Algorithms*. Chapman & Hall/CRC.

Dong, G., and Li, J. 1999. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 43–52.

Doshi-Velez, F., and Kim, B. 2017. Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*.

Feige, U. 1998. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)* 45(4):634–652.

Fukuda, T.; Morimoto, Y.; Morishita, S.; and Tokuyama, T. 1996. Mining optimized association rules for numeric attributes. In *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 182–191.

Hara, S., and Maehara, T. 2017. Enumerate lasso solutions for feature selection. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 1985–1991.

Hochbaum, D. S. 1996. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation Algorithms for NP-Hard Problems*, 94–143.

Jordan, K. L., and Freiburger, T. L. 2015. The effect of race/ethnicity on sentencing: Examining sentence type, jail length, and prison length. *Journal of Ethnicity in Criminal Justice* 13(3):179–196.

Kim, B. 2015. *Interactive and interpretable machine learning models for human machine collaboration*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Komiyama, J.; Ishihata, M.; Arimura, H.; Nishibayashi, T.; and Minato, S.-I. 2017. Statistical emerging pattern mining with multiple testing correction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 897–906.

Komiyama, J. 2017. https://github.com/jkomiyama/qtlamp.

Lakkaraju, H.; Bach, S. H.; and Leskovec, J. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1675–1684.

Lakkaraju, H.; Aguiar, E.; Shan, C.; Miller, D.; Bhanpuri, N.; Ghani, R.; and Addison, K. L. 2015. A machine learning framework to identify students at risk of adverse academic outcomes. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1909–1918.

Larson, J.; Mattu, S.; Kirchner, L.; and Angwin, J. 2016. How we analyzed the compas recidivism algorithm. *ProPublica*.

Larus-Stone, N. 2017. https://github.com/nlarusstone/corels.

Lawler, E. L. 1972. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science* 18(7):401–405.

Letham, B.; Rudin, C.; McCormick, T. H.; Madigan, D.; et al. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics* 9(3):1350–1371.

Li, J.; Shen, H.; and Topor, R. 2002. Mining the optimal class association rule set. *Knowledge-Based Systems* 15(7):399–405.

Lichman, M. 2013. UCI machine learning repository.

Nemhauser, G. L., and Wolsey, L. A. 1978. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research* 3(3):177–188.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functionsI. *Mathematical Programming* 14(1):265–294.

Ong, H. Y.; Wang, D.; and Mu, X. S. 2014. Diabetes prediction with incomplete patient data. *Technical Report*.

Quinlan, J. R. 2014. *C4.5: Programs for Machine Learning*. Elsevier.

Rastogi, R., and Shim, K. 2002. Mining optimized association rules with categorical and numeric attributes. *IEEE Transactions on Knowledge and Data Engineering* 14(1):29–50.

Rivest, R. L. 1987. Learning decision lists. *Machine Learning* 2(3):229–246.

Ruggieri, S. 2017. Enumerating distinct decision trees. In *Proceedings of the 34th International Conference on Machine Learning*, 2960–2968.

Srikant, R., and Agrawal, R. 1996. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 1–12.