Finding Patterns in Features and Observations: New Machine Learning Models with Applications in Computational Criminology, Marketing, and Medicine

by

Tong Wang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2016

Author ..... Signature redacted .....
Department of Electrical Engineering and Computer Science
June 20, 2016

Certified by ... Signature redacted .....
Cynthia Rudin
Associate Professor
Thesis Supervisor

Accepted by ... Signature redacted .....
Leslie A. Kolodziejski
Professor
Chair, Department Committee on Graduate Theses

Finding Patterns in Features and Observations: New Machine Learning Models with Applications in Computational Criminology, Marketing, and Medicine

by

Tong Wang

Submitted to the Department of Electrical Engineering and Computer Science
on June 16, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

The revolution of "Big Data" has reached various fields like marketing, healthcare, and criminology, where domain experts wish to find and understand interesting patterns from data. This thesis studies patterns that are defined by subsets of observations or subsets of features.

The first part of the thesis studies patterns defined by subsets of observations. We look at a specific type of pattern, crime series (a set of crimes committed by the same individual or group) and develop two pattern detection algorithms. The first method is a sequential pattern building algorithm called *Series Finder*, which resembles how crime analysts process information instinctively and grows a crime series starting from a couple of seed crimes. The second method is a subspace clustering with cluster-specific feature selection, which is supervised when learning similarity graphs in order to reduce computation. Both methods we propose achieved promising results on a decade's worth of crime pattern data collected by the Crime Analysis Unit of the Cambridge Police Department.

The second part of the thesis studies patterns defined by subsets of features. We develop methods and theory for building Rule Set models with the hallmark of *interpretability*. Interpretability is inherent in using association rules to explain predicted results. We first design two methods for building rule sets for binary classification. The first method Bayesian Rule Set (BRS) uses a Bayesian framework with priors that favor small models. The Bayesian priors also bring significant computational benefits to MAP inferences by reducing the search space and restraining the sampling chain within appropriate regions. We apply BRS models to an in-vehicle recommender system data set we collected via Amazon Mechanical Turk to study the customers and contexts that would encourage acceptance of coupons. We develop another model Optimized Rule Set (ORS) using optimization methods to directly construct rule sets from data, without pre-mining rules or discretizing continuous attributes. As a main application of ORS, we build a diagnostic screening tool for obstructive sleep apnea trained on data provided by the Sleep Lab at Mass General Hospital. Our models achieve high accuracy with a substantial gain in interpretability over other

methods. Lastly, we build a Causal Rule Set (CRS) model for causal analysis, to identify subgroups that can benefit from a treatment. CRS combines BRS and Bayesian Logistic Regression. We take advantage of different strategies in inference algorithm to speed up computation. Simulations and experiments show that distributing treatment according to CRS models enhances average treatment effect.

Thesis Supervisor: Cynthia Rudin
Title: Associate Professor

Acknowledgments

First and foremost, I want to thank my research advisor, Professor Cynthia Rudin. She has been supportive, encouraging and patient since the first day I worked with her. I'm greatly influenced by her philosophy of doing useful research that can help people. This will always be the goal I pursue in my future career. Cynthia has been offering me tremendous help and guidance as an advisor and also life mentor who really cares and wants me to succeed. Her passion, integrity and dedication have been and will continue to be my sources of inspiration.

I would also like to thank my thesis committee, Professor Peter Szolovits and Professor Stefanie Jegelka for advices and suggestions on my thesis. I want to thank Pete especially for his help with my application and useful discussions on my research.

I want to thank my labmates, Berk Ustun, Siong Thye Goh, Hongyu Yang, Fulton Wang, Stefano Traca and Theja Tulabandhula. They are always the first audience of my presentations and generously offered help and suggestions to help me improve. I benefited a lot from discussions with them and enjoyed every interaction with them. I sincerely hope all the best with their future endeavors.

My research would not have been completed without my collaborators. I'm very grateful to Rich Sevieri and Lieutenant Daniel Wagner from the Cambridge Police Department. They are the domain experts for the crime project and offered their experience and advice to help me solve the problem. It was from many meetings with them and seeing their dedication to their work I realized the importance of the research we were doing. I'm also grateful to Professor Finale Doshi-Velez and collaborators from Ford Company for insightful discussions on the Ford project.

I would not have been enjoying my PhD life so much without many amazing friends I met at MIT. I learned from them, shared happy memories with them and grew together with them. Their support and company carried me through many dark moments, helped me overcome the obstacles and constantly reminded me that I am never alone. They made my six years at MIT unforgettable and beautiful.

Lastly, I would like to thank the most important people in my life, my family: my

parents for their unconditional love and unreserved support; my husband, Guanbo Chen for his understanding and encouragement. I am very fortunate to have Guanbo by my side through the end of my PhD to a new journey that awaits me, giving me faith and strength, and witnessing the milestones I have accomplished and to accomplish in the future.

This PhD thesis marks the end of a wonderful journey and the beginning of a new road to many dreams ahead that I will try my best to realize.

This doctoral thesis has been examined by a Committee of the Department of Electrial Engineering & Computer Science as follows:

Signature redacted

Professor Cynthia Rudin .....

Thesis Supervisor
Associate Professor of Statistics

Signature redacted

Professor Peter Szolovits .....

Thesis Committee
Professor of Computer Science and Engineering

Signature redacted

Professor Stefanie Jegelka .....

Thesis Committee
X-Consortium Career Development Assistant Professor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Applying machine learning models to different domains presents a number of challenges and opportunities. Traditional machine learning models that have complicated structures or provide a result without being able to make sense to humans do not suffice. The main theme of this thesis is that we try to build and study models that work with humans. We solve specific problems that humans currently have difficulty accomplishing, and we build general *interpretable* machine learning models that are easy for humans to understand and use.

1.1 Detecting Crime Series

In the first part of the thesis, we look at a type of patterns defined by subsets of "similar" observations. We pay our attention to a main application in crime data mining and develop algorithms for detecting a specific type of crime pattern, *crime series*, which refer to crimes that are committed by the same offender or group. Detecting crime series is one of the most challenging problems facing crime analysts every day due to the large volume of data and the difficulty of the task itself. It is also an important step in predictive policing, as police officers need to know about a crime series in order to take actions towards it. Currently, crime analysts detect patterns manually; our goal is to assist them by providing automated tools for discovering crime series from within a database of crimes.

In Chapter 2 and 3, we aim directly at identifying series in housebreaks from data

provided by the Crime Analysis Unit of the Cambridge Police Department. Housebreaks can be extremely difficult crimes to solve since there is often no suspect information, as the crimes take place when residents are not present. Nationwide only 14% of housebreaks are solved [96]. we proposed two methods, both of which have had promising results on a decade's worth of crime pattern data.

In Chapter 2, we propose a pattern detection algorithm called *Series Finder*, which resembles how crime analysts process information instinctively. It is a supervised machine learning method that grows a crime series starting from a couple of seed crimes. Series Finder measures the crime similarity in terms of the modus operandi (M.O.), using two types of coefficients: pattern-general coefficients that are learned from past crime series, and pattern-specific coefficients that are updated iteratively as new information arrives. As the series grows, the M.O. becomes more well-defined. After the supervised learning stage, which involves an optimization problem solved on past data, finding new crime series is computationally very efficient. This method can easily run in real time, so that when a new crime happens, it can be immediately detected if it is associated with an existing crime series and the rest of that series could be directly provided to a police officer at the scene. It is an example of a principled machine learning method developed for a new and important application domain.

Chapter 3 presents a different method for detecting crime series that can further identify the M.O. of a crime series. Our approach relies on a key hypothesis that each crime series possesses a *core* of crimes that are similar to each other, which can be used to characterize the M.O. of the criminal. We propose a subspace clustering method with three steps: we first construct a similarity graph to link crimes that are similar, second we find core sets of crime using an integer linear programming approach, and third we construct the rest of the crime series by merging core sets to form the full crime series. To judge whether a crime series is indeed a core, we consider both *pattern-general similarity*, which can be learned from past crime series, and *pattern-specific similarity*, which is specific to the M.O. of the series and cannot be learned. Our method can be used for general pattern detection beyond crime series detection, as core sets exists for patterns in many domains.

1.2 Interpretable Rule Sets

In the second part of the thesis, we learn patterns defined by subsets of features that are learned from data. This type of patterns are characterized by a set of rules that are conjunctions of conditions.

A main application of this form of patterns is in interpretable modeling. As nicely stated by the director of the U.S. National Institute of Justice [72], an interpretable model that is actually used is better than one that is more accurate that sits on a shelf. Interpretability of a model has been of interest to many domains especially medical diagnosis, customer behavior analysis, etc, where domain experts need to understand a model in order to trust and adopt it. Sometimes, the need for explainability of a model surpasses the need to make accurate predictions. Interpretability is a main quality we desire in the second part of the thesis and it is a hallmark of Rule Set models since they include two aspects of interpretability: logical forms, and sparsity (see, e.g. [22, 42, 59]). A rule set model might say:

if a customer (goes to coffee houses more than once per month AND has no urgent destination AND passenger is not kids)
OR (goes to coffee houses more than once per month AND the coupon expires in one day) **then**

the customer is predicted to accept the coupon for a coffee house.

else

the customer will not accept the coupon for a coffee house

A rule set model predicts that an observation is in the positive class when at least one of the "and" conditions is satisfied. Otherwise the observation is classified to be in the negative class.

Rule set formulae are particularly useful as *screening* models, where patients who do not meet the rule set criteria are not considered for further testing. In marketing, rule set is also called "disjunctions of conjunctions" or "non-compensatory decision rules." Marketing researchers strongly hypothesize that consumers use simple rules to screen products, and they would consider purchasing only the products in this *consideration set* to reduce

the cognitive load of considering all products. The consideration set may be precisely a rule set classifier [31,38].

In Chapter 4, 5 and 6, we study Rule Sets models, and show that they are powerful enough to achieve good predictive accuracy while maintaining the interpretability inherent in having rules as the fundamental unit of the model. A rule set model can be used as a binary classifier (discussed in Chapter 4 and 5), where the rules characterize one of the classes (usually the minor class); it can also be used to identify subgroups in causal analysis (discussed in Chapter 6), where the rules characterize subpopulation that benefit from the treatment.

In Chapter 4, we present a Bayesian approach for constructing rule sets for binary classification. We present two probabilistic models with prior parameters that the user can set to encourage the rule set to have a desired size and shape, to conform with a domain-specific definition of interpretability. We provide a scalable MAP inference approach and theoretical bounds to reduce computation. We apply the Bayesian Rule Set (*BRS*) model to predict user behavior with respect to in-vehicle context-aware personalized recommender systems. Our method has a major advantage over classical associative classification methods and decision trees in that it does not greedily grow the model.

In Chapter 5, we took alternative approaches for forming a rule set. We present two optimization methods, Optimized Rule Sets (ORS) and its faster version, Optimized Rule Sets with Approximations (ORSx). In the ORS model, instead of pre-mining rules from data with manually discretized continuous attributes, this approach relies on mixed integer programming to automatically find cut-offs on continuous attributes and directly choose conditions to form rules then a rule set. We build a diagnostic screening tool using ORS for obstructive sleep apnea that achieves high accuracy with a substantial gain in interpretability over other methods.

In Chapter 6, we turn our attention to causal analysis. We develop a method to build rule sets that discriminate between two subgroups, Effective Class, for which the treatment is effective, and Ineffective Class, for which the treatment is not effective. We present a Bayesian hierarchical structure with a prior that favors smaller models for interpretability purposes, and a Bayesian logistic regression that characterizes the relation between the

attributes, the treatment effect and the responses. The treatment effect is incorporated as an additional term in the regression model and assumed constant within the subgroup. We formulate the search as simulated annealing over joint solution space of pattern sets and regression parameters, with different strategies to improve the search efficiency by reducing solution space, branch-and-bounding, etc. We apply CRS to a bank marketing data set to identify subgroups of clients who will only make a deposit if they are contacted before the current campaign.

Chapter 2

Detecting Crime Series with Series Finder

2.1 Introduction

The foundation of *predictive policing* is that if we are able to predict crime, we can takes steps to prevent it. Empirical approaches to predictive policing have recently been adopted by many law enforcement agencies, and the National Institute of Justice has recently launched an initiative in support of predictive policing [68]. One of the most important problems in predictive policing is that of "crime series detection," or the detection of a set of crimes committed by the same individual or group. If a crime analyst can identify an ongoing crime series, it is possible that actions can be taken to stop this pattern, and possibly, to apprehend the suspect(s). There is convincing evidence that strategies where police target specific times and locations result in crime reduction and safer neighborhoods [78, 94, 95].

However, even with new data-driven approaches to crime prediction, the fundamental job of crime analysts still remains difficult and often manual; *specific* patterns of crime are not necessarily easy to find by way of automated tools, whereas larger-scale density-based trends comprised mainly of background crime levels are much easier for data-driven approaches and software to estimate. The most frequent (and most successful) method to identify specific crime patterns involves the review of crime reports each day and the comparison of those reports to past crimes [34], even though this process can be extraordinarily

time-consuming. In making these comparisons, an analyst looks for enough commonalities between a past crime and a present crime to suggest a pattern. Even though automated detection of specific crime patterns can be a much more difficult problem than estimating background crime levels, tools for solving this problem could be extremely valuable in assisting crime analysts, and could directly lead to actionable preventive measures. Locating these patterns automatically is a challenge that machine learning tools and data mining analysis may be able to handle in a way that directly complements the work of human crime analysts.

In this work, we take a machine learning approach to the problem of detecting specific patterns of crime that are committed by the same offender or group. Our learning algorithm processes information similarly to how crime analysts process information instinctively: the algorithm searches through the database looking for similarities between crimes in a growing pattern and in the rest of the database, and tries to identify the modus operandi (M.O.) of the particular offender. The M.O. is the set of habits that the offender follows, and is a type of motif used to characterize the pattern. As more crimes are added to the set, the M.O. becomes more well-defined. Our approach to pattern discovery captures several important aspects of patterns:

- *Each M.O. is different.* Criminals are somewhat self-consistent in the way they commit crimes. However, different criminals can have very different M.O.'s. Consider the problem of predicting housebreaks (break-ins): Some offenders operate during weekdays while the residents are at work; some operate stealthily at night, while the residents are sleeping. Some offenders favor large apartment buildings, where they can break into multiple units in one day; others favor single-family houses, where they might be able to steal more valuable items. Different combinations of crime attributes can be more important than others for characterizing different M.O's.

- *General commonalities in M.O. do exist.* Each pattern is different but, for instance, similarity in time and space are often important to any pattern and should generally by weighted highly. Our method incorporates both general trends in M.O. and also pattern-specific trends.

- *Patterns can be dynamic.* Sometimes the M.O. shifts during a pattern. For instance, a novice burglar might initially use bodily force to open a door. As he gains experience, he might bring a tool with him to pry the door open. Occasionally, offenders switch entirely from one neighborhood to another. Methods that consider an M.O. as stationary would not naturally be able to capture these dynamics.

2.2 Background and Related Work

In this work, we define a "pattern" as a series of crimes committed by the same offender or group of offenders. This is different from a "hotspot" which is a spatially localized area where many crimes occur, whether or not they are committed by the same offender. It is also different than a "near-repeat" effect which is localized in time and space, and does not require the crimes to be committed by the same offender. To identify true patterns, one would need to consider information beyond simply time and space, but also other features of the crimes, such as the type of premise and means of entry.

An example of a pattern of crime would be a series of housebreaks over the course of a season committed by the same person, around different parts of East Cambridge, in houses whose doors are left unlocked, between noon and 2pm on weekdays. For this pattern, sometimes the houses are ransacked and sometimes not, and sometimes the residents are inside and sometimes not. This pattern does not constitute a "hotspot" as it's not localized in space. These crimes are not "near-repeats" as they are not localized in time and space (see [71]).

Most predictive policing software has the capability to detect general background levels of crime density, which are much easier to predict than specific patterns of crime. Detailed information about each crime is not necessary to predict background levels, whereas it is necessarily required to determine the M.O. However, knowing the M.O. of a crime series can be actionable, whereas estimation of background levels may not be directly actionable. For instance, if suspect information is available for at least one crime in a series, then this suspect information can be linked to all crimes in the series. Further, if a current crime series has predictable times and locations (for instance, a pickpocket targeting a single cafe

at regular intervals throughout the week), actions can be taken to stop the pattern. Also, if a current crime series has the same M.O. as a past crime series for which the offender is known, then the suspect from the older series could be a potential offender for the current crime series.

We know of very few previous works aimed directly at detecting specific patterns of crime (e.g., [12, 21, 52, 64]). . One of these works is that of Dahbur and Muscarello [21], [1] who use a cascaded network of Kohonen neural networks followed by heuristic processing of the network outputs. However, feature grouping in the first step makes an implicit assumption that attributes manually selected to group together have the same importance, which is not necessarily the case: each crime series has a signature set of factors that are important for that specific series, which is one of the main points we highlighted in the introduction. Their method has serious flaws, for instance that crimes occurring before midnight and after midnight cannot be grouped together by the neural network regardless of how many similarities exists between them, hence the need for heuristics. Series Finder has no such serious modeling defect. Nath [64] uses a semi-supervised clustering algorithm to detect crime patterns. He developed a weighting scheme for attributes, but the weights are provided by detectives instead of learned from data, similar to the baseline comparison methods we use. Brown and Hagen [12] and Lin and Brown [52] use similarity metrics like we do, but do not learn parameters from past data.

Many classic data mining techniques have been successful for crime analysis generally, such as association rule mining [12, 13, 52, 67], classification [89], and clustering [64]. We refer to the general overview of Chen et al. [15], in which the authors present a general framework for crime data mining, where many of these standard tools are available as part of the COPLINK [37] software package. Much recent work has focused on locating and studying hotspots, which are localized high-crime-density areas (e.g., [60, 79, 80], and for a review, see [25]).

Algorithmic work on semi-supervised clustering methods (e.g., [8, 87]) is slightly related to our approach, in the sense that the set of patterns previously labeled by the police

[1] Also see http://en.wikipedia.org/wiki/Classification$_$System$_$for$_$Serial$_$Criminal$_$Patterns

can be used as constraints for learned clusters; on the other hand, each of our clusters has different properties corresponding to different M.O.'s, and most of the crimes in our database are not part of a pattern and do not belong to a cluster. Standard clustering methods that assume all points in a cluster are close to the cluster center would also not be appropriate for modeling dynamic patterns of crime. Also not suitable are clustering methods that use the same distance metric for different clusters, as this would ignore the pattern's M.O. Clustering is usually unsupervised, whereas our method is supervised. Work on (unsupervised) set expansion in information retrieval (e.g., [30,47]) is very relevant to ours. In set expansion, they (like us) start with a small seed of instances, possess a sea of unlabeled entities (webpages), most of which are not relevant, and attempt to grow members of the same set as the seed. The algorithms for set expansion do not adapt to the set as it develops, which is important for crime pattern detection. The baseline algorithms we compare with are similar to methods like Bayesian Sets applied in the context of Growing a List [30, 47] in that they use a type of inner product as the distance metric.

2.3 Series Finder for Pattern Detection

Series Finder is a supervised learning method for detecting patterns of crime. It has two different types of coefficients: pattern-specific coefficients $\{\eta_{\hat{\mathcal{P}},j}\}_j$, and pattern-general coefficients $\{\lambda_j\}_j$. The attributes of each crime (indexed by j) capture elements of the M.O. such as the means of entry, time of day, etc. Patterns of crime are grown sequentially, starting from candidate crimes (the seed). As the pattern grows, the method adapts the pattern-specific coefficients in order to better capture the M.O. The algorithm stops when there are no more crimes within the database that are closely related to the pattern.

The crime-general coefficients are able to capture common characteristics of all patterns (bullet 2 in the introduction), and the pattern-specific coefficients adjust to each pattern's M.O. (bullet 1 in the introduction). Dynamically changing patterns (bullet 3 in the introduction) are captured by a similarity S, possessing a parameter d which controls the "degree of dynamics" of a pattern. We discuss the details within this section.

Let us define the following:

- $\mathcal{V}$ – A set of all crimes.
- $\mathscr{P}$ – A set of all patterns.
- $\mathcal{P}$ – A single pattern, which is a set of crimes. $\mathcal{P} \subseteq \mathscr{P}$.
- $\hat{\mathcal{P}}$ – A pattern grown from a seed of pattern $\mathcal{P}$. Ideally, if $\mathcal{P}$ is a true pattern and $\hat{\mathcal{P}}$ is a discovered pattern, then $\hat{\mathcal{P}}$ should equal $\mathcal{P}$ when it has been completed. Crimes in $\hat{P}$ are represented by $v_1, v_2, ...v_{|\hat{\mathcal{P}}|}$.
- $V_{\hat{\mathcal{P}}}$ – The set of candidate crimes we will consider when starting from $\hat{\mathcal{P}}$ as the seed. These are potentially part of pattern $\mathcal{P}$. In practice, $V_{\hat{\mathcal{P}}}$ is usually a set of crimes occurring within a year of the seed of $\mathcal{P}$. $V_{\hat{\mathcal{P}}} \subseteq \mathcal{V}$.
- $s_j(v_i, v_k)$ – Similarity between crime i and k in attribute j. There are a total of J attributes. These similarities are calculated from raw data.
- $\gamma_{\hat{\mathcal{P}}}(v_i, v_k)$ – The overall similarity between crime i and k. It is a weighted sum of all J attributes, and is pattern-specific.

2.3.1 Crime-Crime Similarity

The pairwise similarity γ measures how similar crimes v_i and v_k are in a pattern set $\hat{\mathcal{P}}$. We model it in the following form:

$$\gamma_{\hat{\mathcal{P}}}(v_i, v_k) = \frac{1}{\Gamma_{\hat{\mathcal{P}}}} \sum_{j=1}^{J} \lambda_j \eta_{\hat{\mathcal{P}},j} s_j(v_i, v_k),$$

where two types of coefficients are introduced:

1. λ_j – pattern-general weights. These weights consider the general importance of each attribute. They are trained on past patterns of crime that were previously labeled by crime analysts as discussed in Section 2.3.4.
2. $\eta_{\hat{\mathcal{P}},j}$ – pattern-specific weights. These weights capture characteristics of a specific pattern. All crimes in pattern $\hat{\mathcal{P}}$ are used to decide $\eta_{\hat{\mathcal{P}},j}$, and further, the defining

characteristics of $\hat{\mathcal{P}}$ are assigned higher values. Specifically:

$$\eta_{\hat{\mathcal{P}},j} = \sum_{i=1}^{|\hat{\mathcal{P}}|} \sum_{k=1}^{|\hat{\mathcal{P}}|} s_j(v_i, v_k)$$

$\Gamma_{\hat{\mathcal{P}}}$ is the normalizing factor $\Gamma_{\hat{\mathcal{P}}} = \sum_{j=1}^{J} \lambda_j \eta_{\hat{\mathcal{P}},j}$. Two crimes have a high $\gamma_{\hat{\mathcal{P}}}$ if they are similar along attributes that are important specifically to that crime pattern, and generally to all patterns.

2.3.2 Pattern-Crime Similarity

Pattern-crime similarity S measures whether crime $\tilde{v}$ is similar enough to set $\hat{\mathcal{P}}$ that it should be potentially included in $\hat{\mathcal{P}}$. The pattern-crime similarity incorporates the dynamics in M.O. discussed in the introduction. The dynamic element is controlled by a parameter d, called the *degree of dynamics*. The pattern-crime similarity is defined as follows for pattern $\hat{\mathcal{P}}$ and crime $\tilde{v}$:

$$S(\hat{\mathcal{P}}, \tilde{v}) = \left(\frac{1}{|\hat{\mathcal{P}}|} \sum_{i=1}^{|\hat{\mathcal{P}}|} \gamma_{\hat{\mathcal{P}}}(\tilde{v}, v_i)^d \right)^{(1/d)}$$

where $d \geq 1$. This is a soft-max, that is, an ℓ_d norm over $i \in \hat{\mathcal{P}}$. Use of the soft-max allows the pattern $\hat{\mathcal{P}}$ to evolve: crime i needs only be very similar to a few crimes in $\hat{\mathcal{P}}$ to be considered for inclusion in $\hat{\mathcal{P}}$ when the degree of dynamics d is large. On the contrary, if d is 1, this forces patterns to be very stable and stationary, as new crimes would need to be similar to most or all of the crimes already in $\hat{\mathcal{P}}$ to be included. That is, if $d = 1$, the dynamics of the pattern are ignored. For our purpose, d is chosen appropriately to balance between including the dynamics (d large), and stability and compactness of the pattern (d small).

2.3.3 Sequential Pattern Building

Starting with the seed, crimes are added iteratively from $V_{\hat{\mathcal{P}}}$ to $\hat{\mathcal{P}}$. At each iteration, the candidate crime with the highest pattern-crime similarity to $\hat{\mathcal{P}}$ is tentatively added to $\hat{\mathcal{P}}$. Then $\hat{\mathcal{P}}$'s cohesion is evaluated, which measures the cohesiveness of $\hat{\mathcal{P}}$ as a pattern of

crime: Cohesion($\hat{\mathcal{P}}$) $= \frac{1}{|\hat{\mathcal{P}}|}\sum_{i\in\hat{\mathcal{P}}} S(\hat{\mathcal{P}} \setminus v_i, v_i)$. While the cohesion is large enough, we will proceed to grow $\hat{\mathcal{P}}$. If $\hat{\mathcal{P}}$'s cohesion is below a threshold, $\hat{\mathcal{P}}$ stops growing. Here is the formal algorithm:

1: **Initialization**: $\hat{\mathcal{P}} \leftarrow$ {Seed crimes}
2: **repeat**
3: $\quad v_{\text{tentative}} = \arg\max_{v\in(V_{\hat{\mathcal{P}}}\setminus\hat{\mathcal{P}})} S(\hat{\mathcal{P}}, v)$
4: $\quad \hat{\mathcal{P}} \leftarrow \hat{\mathcal{P}} \cup \{v_{\text{tentative}}\}$
5: $\quad$ Update: $\eta_{\hat{\mathcal{P}},j}$ for $j \in \{1, 2, \ldots J\}$, and Cohesion($\hat{\mathcal{P}}$)
6: **until** Cohesion($\hat{\mathcal{P}}$) $<$ cutoff
7: $\hat{\mathcal{P}}^{\text{final}} := \hat{\mathcal{P}} \setminus v_{\text{tentative}}$
8: **return** $\hat{\mathcal{P}}^{\text{final}}$

2.3.4 Learning the Pattern-General Weights $\boldsymbol{\lambda}$

The pattern-general weights are trained on past pattern data, by optimizing a performance measure that is close to the performance measures we will use to evaluate the quality of the results. Note that an alternative approach would be to simply ask crime analysts what the optimal weighting should be, which was the approach taken by Nath [64]. (This simpler method will also be used in Section 3.6 as a baseline for comparison.) We care fundamentally about optimizing the following measures of quality for our returned results:

- The fraction of the true pattern $\mathcal{P}$ returned by the algorithm:

$$\text{Recall}(\mathcal{P}, \hat{\mathcal{P}}) = \frac{\sum_{v\in\mathcal{P}} \mathbb{1}(v \in \hat{\mathcal{P}})}{|\mathcal{P}|}.$$

- The fraction of the discovered crimes that are within pattern $\mathcal{P}$:

$$\text{Precision}(\mathcal{P}, \hat{\mathcal{P}}) = \frac{\sum_{v\in\hat{\mathcal{P}}} \mathbb{1}(v \in \mathcal{P})}{|\hat{\mathcal{P}}|}.$$

The training set consists of true patterns $\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_\ell, \ldots \mathcal{P}_{|\mathscr{P}|}$. For each pattern $\mathcal{P}_\ell$ and its corresponding $\hat{\mathcal{P}}_\ell$, we define a gain function $g(\hat{\mathcal{P}}_\ell, \mathcal{P}_\ell, \boldsymbol{\lambda})$ containing both precision and

recall. The dependence on $\boldsymbol{\lambda} = \{\lambda_j\}_{j=1}^{J}$ is implicit, as it was used to construct $\hat{\mathcal{P}}_\ell$.

$$g(\hat{\mathcal{P}}_\ell, \mathcal{P}_\ell, \boldsymbol{\lambda}) = \text{Recall}(\mathcal{P}_\ell, \hat{\mathcal{P}}_\ell) + \beta \cdot \text{Precision}(\mathcal{P}_\ell, \hat{\mathcal{P}}_\ell)$$

where β is the trade-off coefficient between the two quality measures. We wish to choose $\boldsymbol{\lambda}$ to maximize the gain over all patterns in the training set.

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{maximize}} \quad & G(\boldsymbol{\lambda}) = \sum_{\ell} g(\hat{\mathcal{P}}_\ell, \mathcal{P}_\ell, \boldsymbol{\lambda}) \\ \text{subject to} \quad & \lambda_j \geq 0, \; j = 1, \ldots, J, \\ & \sum_{j=1}^{J} \lambda_j = 1. \end{aligned}$$

The optimization problem is non-convex and non-linear. However we hypothesize that it is reasonably smooth: small changes in $\boldsymbol{\lambda}$ translate to small changes in G. We use coordinate ascent to approximately optimize the objective, starting from different random initial conditions to avoid returning local minima. The procedure works as follows:

Initialize $\boldsymbol{\lambda}$ randomly, Converged=0
while Converged=0 **do**
    for $j = 1 \rightarrow J$ **do**
        $\lambda_j^{\text{new}} \leftarrow \text{argmax}_{\lambda_j} G(\boldsymbol{\lambda})$ (using a linesearch for the optima)
    end for
    if $\boldsymbol{\lambda}^{\text{new}} = \boldsymbol{\lambda}$ **then**
        Converged= 1
    else
        $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{\text{new}}$
return $\boldsymbol{\lambda}$

We now discuss the definition of each of the J similarity measures.

2.4 Attribute Similarity Measures

Each pairwise attribute similarity $s_j : \mathcal{V} \times \mathcal{V} \rightarrow [0, 1]$ compares two crimes along attribute j. Attributes are either categorical or numerical, and by the nature of our data, we are required to design similarity measures of both kinds.

2.4.1 Similarity for Categorical Attributes

In the Cambridge Police database for housebreaks, categorical attributes include "type of premise" (apartment, single-family house, etc.), "ransacked" (indicating whether the house was ransacked) and several others. We wanted a measure of agreement between crimes for each categorical attribute that includes (i) whether the two crimes agree on the attribute (ii) how common that attribute is. If the crimes do not agree, the similarity is zero. If the crimes do agree, and agreement on that attribute is unusual, the similarity should be given a higher weight. For example, in residential burglaries, it is unusual for the resident to be at home during the burglary. Two crimes committed while the resident was in the home are more similar to each other than two crimes where the resident was not at home. To do this, we weight the similarity by the probability of the match occurring, as follows, denoting v_{ij} as the jth attribute for crime v_i:

$$s_j(v_i, v_k) = \begin{cases} 1 - \sum_{q \in Q} p_j^2(x) & \text{if} \quad v_{ij} = v_{kj} = x \\ 0 & \text{if} \quad v_{ij} \neq v_{kj} \end{cases}$$

where $p_j^2(x) = \frac{n_x(n_x-1)}{N(N-1)}$, with n_x the number of times x is observed in the collection of N crimes. This is a simplified version of Goodall's measure [10].

2.4.2 Similarity for Numerical Attributes

Two formats of data exist for numerical attributes, either exact values, such as time 3:26pm, or a time window, e.g., 9:45am - 4:30pm. Unlike other types of crime such as assault and street robbery, housebreaks usually happen when the resident is not present, and thus time windows are typical. In this case, we need a similarity measure that can handle both exact

time information and range-of-time information. A simple way of dealing with a time window is to take the midpoint of it (e.g., [60]), which simplifies the problem but may introduce bias.

Time-of-day profiles. We divide data into two groups: exact data $(t_1, t_2, \ldots, t_{m_e})$, and time window data $(\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_{m_r})$ where each data point is a range, $\tilde{t}_i = [t_{i,1}, t_{i,2}]$, $i = 1, 2, \ldots m_r$. We first create a profile based only on crimes with exact time data using kernel density estimation: $\hat{p}_{\text{exact}}(t) \propto \frac{1}{m_e} \sum_{i=1}^{m_e} K(t - t_i)$ where the kernel $K(\cdot)$ is a symmetric function, in our case a gaussian with a chosen bandwidth (we chose one hour). Then we use this to obtain an approximate distribution incorporating the time window measurements, as follows:

$$\begin{aligned} p(t|\tilde{t}_1, \ldots, \tilde{t}_{m_r}) &\propto p(t) \cdot p(\tilde{t}_1, \ldots, \tilde{t}_{m_r}|t) \\ &\approx \hat{p}_{\text{exact}}(t) \cdot \hat{p}(\text{range includes } t|t). \end{aligned}$$

The function $\hat{p}(\text{range includes } t|t)$ is a smoothed version of the empirical probability that the window includes t:

$$\hat{p}(\text{range includes } t|t) \propto \frac{1}{m_r} \sum_{i=1}^{m_r} \tilde{K}(t, \tilde{t}_i)$$

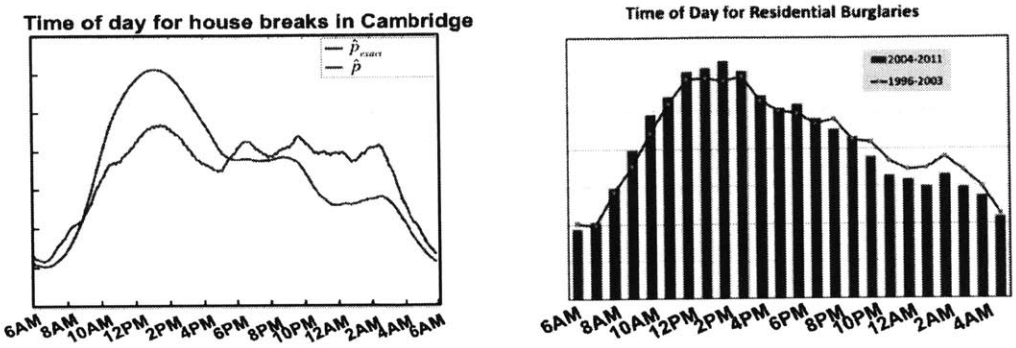
where $\tilde{t}_i = [t_{i,1}, t_{i,2}]$ and $\tilde{K}(t, \tilde{t}_i) := \int_\tau \mathbf{1}_{\tau \in [t_{i,1}, t_{i,2}]} K(t - \tau) \mathrm{d}\tau$. K is again a gaussian with a selected bandwidth. Thus, we define:

$$\hat{p}_{\text{range}}(t) \propto \hat{p}_{\text{exact}}(t) \cdot \frac{1}{m_r} \sum_{i=1}^{m_r} \tilde{K}(t, \tilde{t}_i).$$

We combine the exact and range estimates in a weighted linear combination, weighted according to the amount of data we have from each category:

$$\hat{p}(t) \propto \frac{m_e}{m_e + m_r} \hat{p}_{\text{exact}}(t) + \frac{m_r}{m_e + m_r} \hat{p}_{\text{range}}(t).$$

We used the approach above to construct a time-of-day profile for residential burglaries



(a) Cambridge, from year 1997 to 2012

(b) Portland, from year 1996 to 2011

Figure 2-1: Time of day profiling for house breaks in two different cities

in Cambridge, where $\hat{p}(t)$ and $\hat{p}_{\text{exact}}(t)$ are plotted in Figure 2-1(a). To independently verify the result, we compared it with residential burglaries in Portland between 1996 and 2011 (reproduced from [20]) shown in Figure 2-1(b).[2] The temporal pattern is similar, with a peak at around 1-2pm, a drop centered around 6-7am, and a smaller drop at around midnight, though the profile differs slightly in the evening between 6pm-2am.

A unified similarity measure for numeric attributes. We propose a similarity measure that is consistent for exact and range numerical data. The similarity decays exponentially with the distance between two data values, for either exact or range data. We use the expected average distance over the two ranges as the distance measure. For example, let crime i happen within $t_i := [t_{i,1}, t_{i,2}]$ and crime k happen within $t_k := [t_{k,1}, t_{k,2}]$. Then

$$\tilde{d}(t_i, t_k) = \int_{t_{i,1}}^{t_{i,2}} \int_{t_{k,1}}^{t_{k,2}} \hat{p}(\tau_i|t_i)\hat{p}(\tau_k|t_k)d(\tau_i, \tau_k)\mathrm{d}\tau_i\mathrm{d}\tau_k$$

where $\hat{p}$ was estimated in the previous subsection for times of the day, and $d(\tau_i, \tau_k)$ is the difference in time between τ_i and τ_k. The conditional probability is obtained by renormalizing $\hat{p}(\tau_i|t_i)$ to the interval (or exact value) t_i. The distance measure for exact numerical data can be viewed as a special case of this expected average distance where the conditional probability $\hat{p}(\tau_i|t_i)$ is 1.

[2]To design this plot for Portland, range-of-time information was incorporated by distributing the weight of each crime uniformly over its time window.

The general similarity measure is thus:

$$s_j(v_i, v_k) := \exp\left(-\tilde{d}(z_i, z_k)/\Upsilon_j\right)$$

where Υ_j is a scaling factor (e.g, we chose $\Upsilon_j = 120$ minutes in the experiment), and z_i, z_k are values of attribute j for crimes i and k, which could be either exact values or ranges of values. We applied this form of similarity measure for all numerical (non-categorical) crime attributes.

2.5 Experiments

We used data from 4855 housebreaks in Cambridge between 1997 and 2006 recorded by the Crime Analysis Unit of the Cambridge Police Department. Crime attributes include geographic location, date, day of the week, time frame, location of entry, means of entry, an indicator for "ransacked," type of premise, an indicator for whether residents were present, and suspect and victim information. We also have 51 patterns collected over the same period of time that were curated and hand-labeled by crime analysts.

2.5.1 Evaluation Metrics

The evaluation metrics used for the experimental results are *average precision* and *reciprocal rank*. Denoting $\hat{\mathcal{P}}^i$ as the first i crimes in the discovered pattern, and $\Delta\text{Recall}(\mathcal{P}, \hat{\mathcal{P}}^i)$ as the change in recall from $i-1$ to i:

$$\text{AveP}(\mathcal{P}, \hat{\mathcal{P}}) := \sum_{i=1}^{|\hat{\mathcal{P}}|} \text{Precision}(\mathcal{P}, \hat{\mathcal{P}}^i)\Delta\text{Recall}(\mathcal{P}, \hat{\mathcal{P}}^i).$$

To calculate reciprocal rank, again we index the crimes in $\hat{\mathcal{P}}$ by the order in which they were discovered, and compute

$$\text{RR}(\mathcal{P}, \hat{\mathcal{P}}) := \frac{1}{\left(\sum_{r=1}^{|\hat{\mathcal{P}}|} \frac{1}{r}\right)} \sum_{v_i \in \mathcal{P}} \frac{1}{\text{Rank}(v_i, \hat{\mathcal{P}})},$$

where Rank$(v_i, \hat{\mathcal{P}})$ is the order in which v_i was added to $\hat{\mathcal{P}}$. If v_i was never added to $\hat{\mathcal{P}}$, then Rank$(v_i, \hat{\mathcal{P}})$ is infinity and the term in the sum is zero.

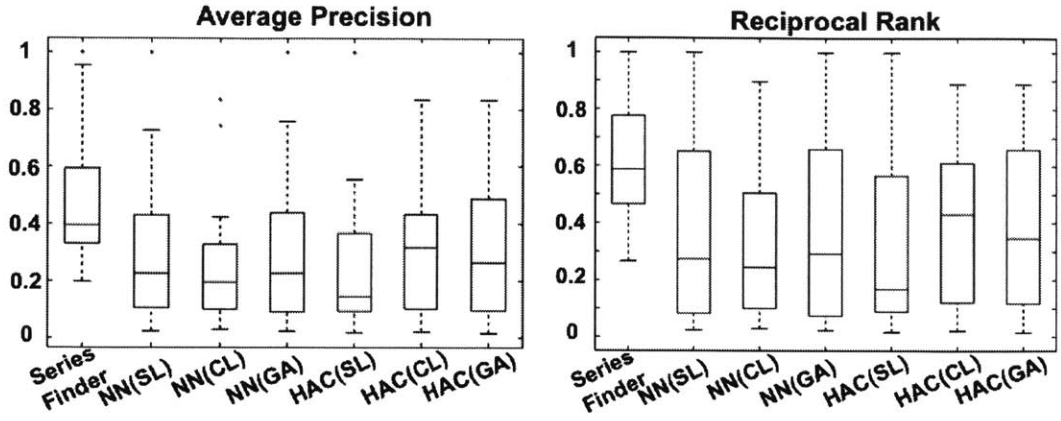
2.5.2 Competing Models and Baselines

We compare with hierarchical agglomerative clustering and an iterative nearest neighbor approach as competing baseline methods. For each method, we use several different schemes to iteratively add discovered crimes, starting from the same seed given to Series Finder. The pairwise similarity γ is a weighted sum of the attribute similarities:

$$\gamma(v_i, v_k) = \sum_{j=1}^{J} \hat{\lambda}_j s_j(v_i, v_k).$$

where the similarity metrics $s_j(v_i, v_k)$ are the same as Series Finder used. The weights $\hat{\boldsymbol{\lambda}}$ were provided by the Crime Analysis Unit of the Cambridge Police Department based on their experience. This will allow us to see the specific advantage of Series Finder, where the weights were learned from past data.

Hierarchical agglomerative clustering (HAC) begins with each crime as a singleton cluster. At each step, the most similar (according to the similarity criterion) two clusters are merged into a single cluster, producing one less cluster at the next level. *Iterative nearest neighbor classification* (NN) begins with the seed set. At each step, the nearest neighbor (according to the similarity criterion) of the set is added to the pattern, until the nearest neighbor is no longer sufficiently similar. HAC and NN were used with three different criteria for cluster-cluster or cluster-crime similarity: *Single Linkage* (SL), which considers the most similar pair of crimes; *Complete Linkage* (CL), which considers the most dissimilar pair of crimes, and *Group Average* (GA), which uses the averaged pairwise similarity [36]. The incremental nearest neighbor algorithm using the S_{GA} measure, with the weights provided by the crime analysts, becomes similar in spirit to the Bayesian Sets algorithm [30] and how it is used in information retrieval applications [47].



(a) (b)

Figure 2-2: Boxplot of evaluation metrics for out-of-sample patterns

$$S_{SL}(R,T) := \max_{v_i \in R, v_k \in T} \gamma(v_i, v_k)$$

$$S_{CL}(R,T) := \min_{v_i \in R, v_k \in T} \gamma(v_i, v_k)$$

$$S_{GA}(R,T) := \frac{1}{|R||T|} \sum_{v_i \in R} \sum_{v_k \in T} \gamma(v_i, v_k).$$

2.5.3 Testing

We trained our models on two-thirds of the patterns from the Cambridge Police Department and tested the results on the remaining third. For all methods, pattern $\hat{\mathcal{P}}_\ell$ was grown until all crimes in $\mathcal{P}_\ell$ were discovered. Boxplots of the distribution of average precision and reciprocal ranks over the test patterns for Series Finder and six baselines are shown in Figure 2-2(a) and Figure 2-2(b). We remark that Series Finder has several advantages over the competing models: (i) Hierarchical agglomerative clustering does not use the similarity between seed crimes. Each seed grows a pattern independently, with possibly no interaction between seeds. (ii) The competing models do not have pattern-specific weights. One set of weights, which is pattern-general, is used for all patterns. (iii) The weights used by the competing models are provided by detectives based on their experience, while the weights of Series Finder are learned from data.

Since Series Finder's performance depends on pattern-specific weights that are cal-

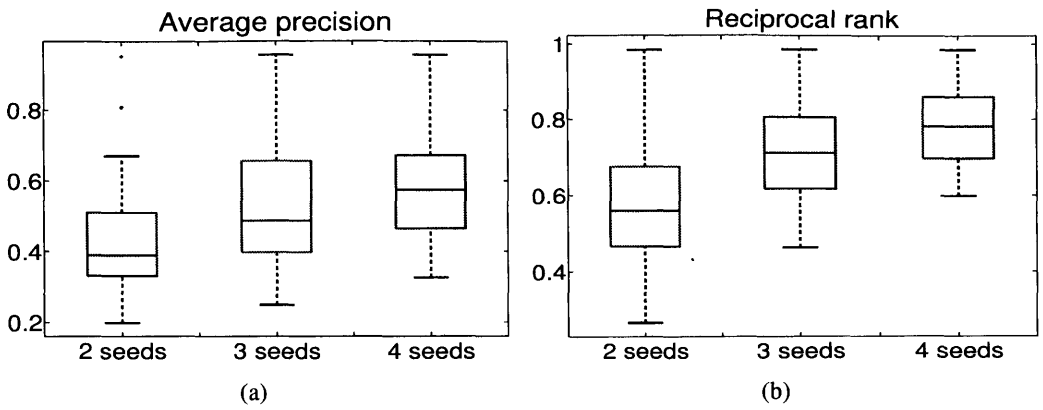


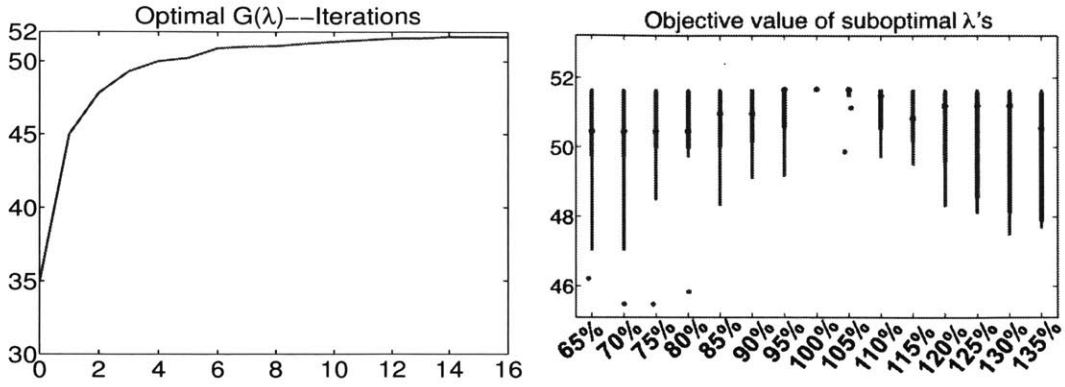
Figure 2-3: Performance of Series Finder with 2, 3 and 4 seeds.

culated from seed crimes, we would like to understand how much each additional crime within the seed generally contributes to performance. The average precision and reciprocal rank for the 16 testing patterns grown from 2, 3 and 4 seeds are plotted in Figure 2-3(a) and Figure 2-3(b). For both performance measures, the quality of the predictions increases consistently with the number of seed crimes. The additional crimes in the seed help to clarify the M.O.

2.5.4 Model Convergence and Sensitivity Analysis

In Section 2.3, when discussing the optimization procedure for learning the weights, we hypothesized that small changes in $\boldsymbol{\lambda}$ generally translate to small changes in the objective $G(\boldsymbol{\lambda})$. Our observations about convergence have been consistent with this hypothesis, in that the objective seems to change smoothly over the course of the optimization procedure. Figure 2-4(a) shows the optimal objective value at each iteration of training the algorithm on patterns collected by the Cambridge Police Department. In this run, convergence was achieved after 14 coordinate ascent iterations. This was the fastest converging run over the randomly chosen initial conditions used for the optimization procedure.

We also performed a sensitivity analysis for the optimum. We varied each of the J coefficients λ_j from 65% to 135%, of its value at the optimum. As each coefficient was varied, the others were kept fixed. We recorded the value of $G(\boldsymbol{\lambda})$ at several points along this spectrum of percentages between 65% and 135%, for each of the λ_j's. This allows



(a) Convergence of $G(\lambda)$

(b) Sensitivity analysis

Figure 2-4: Performance analysis

us to understand the sensitivity of $G(\boldsymbol{\lambda})$ to movement along any one of the axes of the J-dimensional space. We created box plots of $G(\boldsymbol{\lambda})$ at every 5th percentage between between 65% and 135%, shown in Figure 2-4(b). The number of elements in each box plot is the number of dimensions J. These plots provide additional evidence that the objective $G(\boldsymbol{\lambda})$ is somewhat smooth in $\boldsymbol{\lambda}$; for instance the objective value varies by a maximum of approximately 5-6% when one of the λ_j's changes by 10-15%.

2.6 Expert Validation and Case Study

We wanted to see whether our data mining efforts could help crime analysts identify crimes within a pattern that they did not yet know about, or exclude crimes that were misidentified as part of a pattern. To do this, Series Finder was trained on all existing crime patterns from the database to get the pattern-general weights $\boldsymbol{\lambda}$. Next, using two crimes in each pattern as a seed, Series Finder iteratively added candidate crimes to the pattern until the pattern cohesion dropped below 0.8 of the seed cohesion. Crime analysts then provided feedback on Series Finder's results for nine patterns.

There are now three versions of each pattern: $\mathcal{P}$ which is the original pattern in the database, $\hat{\mathcal{P}}$ which was discovered using Series Finder from two crimes in the pattern, and $\mathcal{P}_{\text{verified}}$ which came from crime experts after they viewed the union of $\hat{\mathcal{P}}$ and $\mathcal{P}$. Based on these, we counted different types of successes and failures for the 9 patterns, shown in

Table 2.1: Expert validation study results.

Type of crimes	$\mathcal{P}$	$\hat{\mathcal{P}}$	$\mathcal{P}_{\text{verified}}$	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_5$	$\mathcal{P}_6$	$\mathcal{P}_7$	$\mathcal{P}_8$	$\mathcal{P}_9$
Correct hits	$\subseteq$	$\subseteq$	$\subseteq$	6	5	6	3	8	5	7	2	10
Correct finds	$\not\subseteq$	$\subseteq$	$\subseteq$	2	1	0	1	0	1	2	2	0
Correct exclusions	$\subseteq$	$\not\subseteq$	$\not\subseteq$	0	0	4	1	0	2	1	0	0
Incorrect exclusions	$\subseteq$	$\not\subseteq$	$\subseteq$	0	0	1	0	1	0	0	0	1
False hits	$\not\subseteq$	$\subseteq$	$\not\subseteq$	2	0	0	0	2	2	0	0	0

Table 2.1. The mathematical definition of them is represented by the first 4 columns. For example, *correct finds* refer to crimes that are not in $\mathcal{P}$, but that are in $\hat{\mathcal{P}}$, and were verified by experts as belonging to the pattern, in $\mathcal{P}_{\text{verified}}$.

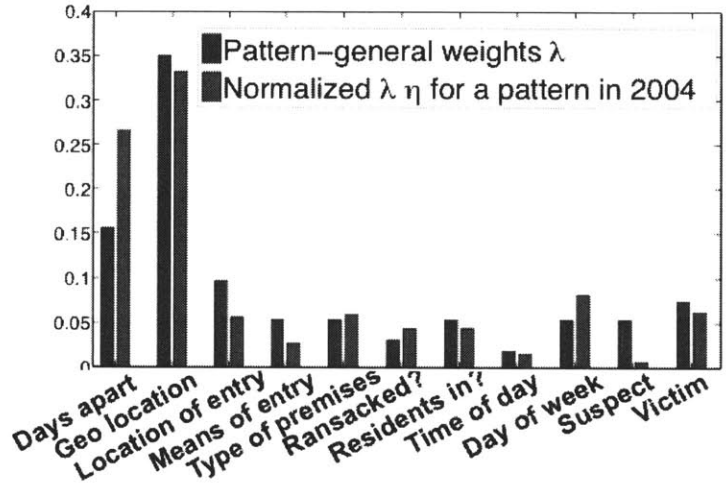
Correct hits, *correct finds* and *correct exclusions* count successes for Series Finder. Specifically, correct finds and correct exclusions capture Series Finder's improvements over the original database. Series Finder was able to discover 9 crimes that analysts had not previously matched to a pattern (the sum of the correct finds) and exclude 8 crimes that analysts agreed should be excluded (the sum of correct exclusions). *Incorrect exclusions* and *false hits* are not successes. On the other hand, false hits that are similar to the crimes within the pattern may still be useful for crime analysts to consider when determining the M.O.

Table 2.2: Example: A 2004 Series

No.	Crime type	Date	No.	Crime type	Type of premises	If ransacked	If resident in	Time of day	Day	Suspect	Victim
1	Seed	1/7/04	Front door	Pried	Aptment	No	Not in	8:45	Wed	null	White F
2	Corr hit	1/18/04	Rear door	Pried	Aptment	Yes	Not in	12:00	Sun	White M	White F
3	Corr hit	1/26/04	Grd window	Removed	Res Unk	No	Not in	7:30-12:15	Mon	null	Hisp F
4	Seed	1/27/04	Rear door	Popped Lock	Aptment	No	Not in	8:30-18:00	Tues	null	null
5	Corr exclu	1/31/04	Grd window	Pried	Res Unk	No	Not in	13:21	Sat	Black M	null
6	Corr hit	2/11/04	Front door	Pried	Aptment	No	Not in	8:30-12:30	Wed	null	Asian M
7	Corr hit	2/11/04	Front door	Pried	Aptment	No	Not in	8:00-14:10	Wed	null	null
8	Corr hit	2/17/04	Grd window	Unknown	Aptment	No	Not in	0:35	Tues	null	null
9	Corr find	2/19/04	Door: unkn	Pried	Aptment	No	Not in	10:00-16:10	Thur	null	White M
10	Corr find	2/19/04	Door: unkn	Pried	Aptment	No	Not in	7:30-16:10	Thur	null	White M
11	Corr hit	2/20/04	Front door	Broke	Aptment	No	Not in	8:00-17:55	Fri	null	null
12	Corr hit	2/25/04	Front door	Pried	Aptment	Yes	Not in	14:00	Wed	null	null

We now discuss a pattern in detail to demonstrate the type of result that Series Finder is producing. The example provided is Pattern 7 in Table 2.1, which is a series from 2004 in Mid-Cambridge covering a time range of two months. Crimes were usually committed on

(a) Locations of crimes



(b) λ and $\frac{1}{\Gamma_{\vec{p}}}\lambda \cdot \eta$ for a pattern in 2004

Figure 2-5: An example pattern in 2004

weekdays during working hours. The premises are all apartments (except two unknowns). Figure 2-5(a) shows geographically where these crimes were located. In Figure 2-5(a), four categories of crime within the 2004 pattern are marked with different colored dots: seed crimes are represented with blue dots, correct hits are represented with orange dots, the correct exclusion is represented with a red dot and the two correct finds are represented with green dots. Table 2.2 provides some details about the crimes within the series.

We visualize the M.O. of the pattern by displaying the weights in Figure 2-5(b). The red bars represent the pattern-general weights $\boldsymbol{\lambda}$ and the blue bars represent the total normalized weights obtained from the product of pattern-general weights and pattern-specific weights for this 2004 pattern. Notable observations about this pattern are that: the time be-

tween crimes is a (relatively) more important characteristic for this pattern than for general patterns, as the crimes in the pattern happen almost every week; the means and location of entry are relatively less important as they are not consistent; and the suspect information is also relatively less important. The suspect information is only present in one of the crimes found by Series Finder (a white male). Geographic closeness is less important for this series, as the crimes in the series are spread over a relatively large geographic distance.

Series Finder made a contribution to this pattern, in the sense that it detected two crimes that analysts had not previously considered as belonging to this pattern. It also correctly excluded one crime from the series. In this case, the correct exclusion is valuable since it had suspect information, which in this case could be very misleading. This exclusion of this crime indicates that the offender is a white male, rather than a black male.

2.7 Conclusion

Series Finder is designed to detect patterns of crime committed by the same individual(s). In Cambridge, it has been able to correctly match several crimes to patterns that were originally missed by analysts. The designer of the near-repeat calculator, Ratcliffe, has stated that the near-repeat calculator is not a "silver bullet" [65]. Series Finder also is not a magic bullet. On the other hand, Series Finder can be a useful tool: by using very detailed information about the crimes, and by tailoring the weights of the attributes to the specific M.O. of the pattern, we are able to correctly pinpoint patterns more accurately than similar methods. As we have shown through examples, the extensive data processing and learning that goes into characterizing the M.O. of each pattern leads to richer insights that were not available previously. Some analysts spend hours each day searching for crime series manually. By replicating the cumbersome process that analysts currently use to find patterns, Series Finder could have enormous implications for time management, and may allow analysts to find patterns that they would not otherwise be able to find.

Chapter 3

Finding Patterns with a Rotten Core: Data Mining for Crime Series with Core Sets

3.1 Introduction

Criminals follow a modus operandi (M.O.) that characterizes their crime series; for instance, some criminals operate exclusively during the day, others work at night, some criminals target apartments for housebreaks, while others target single family houses. In this chapter, we aim to develop a method that automatically identify the M.O. while detecting crime series [34]. Meanwhile, the method we are proposing does not rely on "seed" crimes provided by domain experts. It automatically detects the entire series from a data base.

In this chapter, crime series detection is viewed as a type of subspace clustering problem where the M.O. defines the set of relevant features for each cluster. We cannot determine in advance what the exact M.O. of an undetected crime series will be. Generally speaking, we need to reveal groups of objects that are similar on an unknown subset of their features. We say that such sets exhibit a *pattern-specific* similarity. The pattern-specific similarity cannot be learned since it may be true for only a small number of crimes (perhaps less than 10).

Though we cannot characterize exact M.O.'s before we see them, we can characterize the type of M.O.'s we expect generally - for instance, crimes in a series are often close in time and space. We define a *pattern-general* similarity that encodes which factors are generally common to most crime series. It is learned from past crime series; for instance, time and space have high pattern-general weights. The pattern-general similarity induces a *similarity graph* over the set of crimes. We use this graph to examine whether sets of crimes are generally similar to historical crime series.

The main hypothesis in this work is that most crime patterns have a *core* of crimes that exemplify the M.O. of the series. If we can locate all small cores of crime, we should thus have located parts of most crime series'. This hypothesis is based on the intuition of analysts, and has the dual purpose of assisting with computation: we can indeed consider all possible small subsets to calculate whether they are plausible core sets. We further reduce computation by placing a similarity graph structure on the crimes, so that if the crimes are too far apart in pattern-general similarity, we would never consider them to form a core of a pattern. The core sets are found using an integer linear programming (ILP) formulation, which specifies that core sets must have pattern-specific and pattern-general similarity. Once the core sets are found, we construct the full crime series by merging overlapping core sets. We prove that merging core sets preserves desired properties.

Our method is a general subspace clustering method. It is novel in that it considers both "pattern-general" and "pattern-specific" aspects of patterns, where "pattern-general" means that it is common to many crime series (supervised from past clusters), and "pattern-specific" meaning aspects of a particular crime series (unsupervised). Our method does not force all examples to be part of a cluster, and can thus accommodate "background" (non-series) crime. Our method also can find subspace clusters whose subspace morphs dynamically over the cluster. We note that the three pieces of the method – learn similarity graph, mine core sets, and merge core sets – can each be used independently or paired with other algorithms. The first part is supervised, whereas the other two parts are unsupervised.

Our three methodology sections follow the three main components of our method: *learn a similarity graph*, *mine core sets* and *merge core sets*. We then show experiments where our method was tested on the full housebreak database from the Cambridge Police Depart-

ment containing information from thousands of crimes from over a decade. This method has been able to provide new insights into true patterns of crime committed in Cambridge.

Related Work

Our work relates to various subfields of clustering (see for instance [44]), including pattern-based clustering [69,90] which is a semi-supervised approach (unlike ours - we do not use test data at training time), subspace clustering (e.g., [24,86]) which detects all clusters in all subspaces, and specifically, work at the intersection of dense subgraph mining and pattern mining in feature graphs [33,62].

Other work on space-time event detection is relevant [66], though the goal is to detect patterns where the frequency of records is higher than an expected frequency, whereas here we cannot accurately estimate the expected frequency due to high dimensionality and sparseness.

Most previous work on crime series detection [52, 64] have pattern-general weights that come from experts (like our baselines) and are not learned from data (with [12] as an exception), do not consider pattern-specific aspects, and thus cannot capture specific M.O.'s of crimes. Some past approaches have flaws that require heuristic post-processing to handle [21]. Our previous attempt at solving this was an iterative approach that did not consider core sets [93].

3.2 Model Formulation

Our data consist of entities (crimes) $\mathcal{V}$ each of which has a vector of J features. A similarity metric $s_j(v_l, v_k)$ is computed in Chapter 2 to represent the simialrity between crime l and crime k in the j-th attribute. The average similarity of a set of crimes in feature j is defined as the cohesion of the set:

Definition 1 ***(Cohesion$_j$)*** *For a set of crimes V, the cohesion in the jth feature is the mean*

of pairwise similarities,

$$Cohesion_j(V) = \frac{1}{|V|(|V|-1)} \sum_{v_l, v_k \in V} s_j(v_l, v_k).$$

The defining features of a pattern are those with sufficiently high cohesion.

Definition 2 ***(Defining feature)*** *Defining features for V are those that satisfy $Cohesion_j(V) \geq \theta_j$. The set of defining features for set V is denoted by $\Lambda(V)$.*

The defining features characterize the M.O. of the crime series. If several housebreaks happen in the same neighborhood, around the same time of day, within the same month, and the location of entry is always a window, regardless of the differences in other features, these similarities would indicate that the crimes could have been committed by the same offender. The features "geographic location," "time of day," "time between crimes," and "location of entry" characterize the M.O. for this particular crime series. When we later define core sets, the pattern-specific statistic of interest is the number of defining features of V.

Let us switch from pattern-specific definitions to pattern-general definitions. We will learn a pattern-general similarity function from past crime series'. Pattern-general similarity allows us to weight important features highly; for instance, crimes that are spread very far apart in time and space are unlikely to be a pattern, and thus we will learn from past crime series' that time and space are important features. We will learn a set of weights $[\lambda_1, ..., \lambda_j, ...\lambda_J]$ from past crime data that will provide the importance of each feature within a linear combination. We will search for core sets that have high pattern-general cohesion, defined as follows:

Definition 3 ***(Pattern-general cohesion)*** *The pattern-general cohesion of V is the weighted sum of cohesions over the features: $\sum_{j=1}^{J} \lambda_j Cohesion_j(V)$.*

Having high pattern-general cohesion is not sufficient; for V to be a core of a series, we require additional constraints to ensure the core is connected in the graph theoretic sense (and thus is not composed of two separate clusters for instance), and that it has sufficient connectivity. To define these constraints we thus need to define the similarity graph. To

construct edges for the graph, we first define a metric to measure the similarity between two crimes, as follows:

Definition 4 ***(Pattern-general similarity)*** *The pattern-general similarity is a weighted sum of similarity measures for each feature, using the pattern-general weights* $[\lambda_1, \lambda_2, ...\lambda_J]$.

$$\gamma(v_\ell, v_k) = \sum_{j=1}^{J} \lambda_j s_j(v_\ell, v_k). \tag{3.1}$$

We define the similarity graph to contain edges between crimes that are sufficiently close in the pattern-general sense.

Definition 5 ***(Similarity graph)*** *A similarity graph is an undirected graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, *where* $\mathcal{V} = \{v_1, v_2, ...v_n\}$ *is the node set,* $\mathcal{E}$ *is the edge set,* $\mathcal{E} = \{\{v_\ell, v_k\} | \gamma(v_\ell, v_k) \geq \Delta, v_\ell, v_k \in V, v_\ell \neq v_k\}$.

We learn both the weights $\{\lambda_j\}_j$ and cut-off threshold Δ in Section 3.3. Note that we could eliminate the pattern-general similarity all together by setting the threshold Δ to be very low, and that way we consider only pattern-specific similarity; this, however, would not ease computation or restrict the core sets to be similar to those from other patterns.

Imposing a graph structure on the crimes eases computation, in the sense that we are now only looking for connected sets of the graph. Note that we cannot simply maximize the number of defining features and/or the pattern-general cohesion over all subsets of crime, as it would favor choosing very small sets of crime. To alleviate this problem, we specify the size of the core set $|V|$ and the number of defining features d, and maximize the pattern-general cohesion. Once we find the core sets, we merge or expand them to find the rest of the pattern. We call patterns formed by merging other patterns together *series-like* patterns. An illustration of a series-like pattern is shown in Figure 3-1.

3.3 Learning the Similarity Graph

A similarity graph is constructed by connecting pairs of nodes with pattern-general cohesion above a threshold Δ. The pattern-general cohesion γ is defined in (3.1) as a weighted

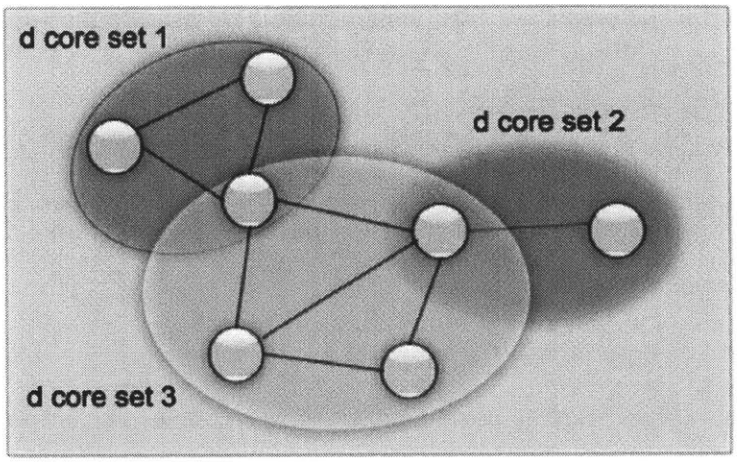


Figure 3-1: A d series-like pattern consisting of three d core sets of various sizes

sum of pairwise similarities in different features, with pattern-general weights $\boldsymbol{\lambda} \in \mathcal{R}^J$. The set of coefficients $\boldsymbol{\lambda}$ and Δ are parameters that we learn from data. These data consist of 51 historical crime series' that have been identified as true crime series by crime analysts.

To learn the weights, we optimize over the historical training patterns to make them as close as possible to being connected subgraphs. This means we want each crime in a historical pattern to be close to at least one other crime in the same pattern. At the same time, we want crimes within a historical pattern to be distant from crimes not in the same pattern. The condition for crime ℓ to be close to at least one other crime within its pattern (with some slack) is:

$$\max_{\{k:k\in\text{pattern}(\ell)\}} \sum_{j=1}^{J} \lambda_j s_j(\ell, k) \geq \Delta - \epsilon_\ell. \tag{3.2}$$

Conversely, crimes that do not belong to the same pattern should not be very similar:

$$\sum_{j=1}^{J} \lambda_j s_j(\ell, k) \leq \Delta + \xi_{\ell k}, \tag{3.3}$$

true for all ℓ and k s.t. $k \notin$ pattern(ℓ). Our goal is to minimize the total weighted slack.

Loosely, we compute:

$$\min_{\boldsymbol{\lambda},\Delta} \quad \sum_{\substack{\text{crimes in the} \\ \text{same pattern}}} \text{slack} + C_1 \sum_{\substack{\text{crimes not in the} \\ \text{same pattern}}} \text{slack} + C_0 \|\boldsymbol{\lambda}\|_0,$$

where $\|\boldsymbol{\lambda}\|_0$ is the ℓ_0 semi-norm of $\boldsymbol{\lambda}$, which encourages sparsity in $\boldsymbol{\lambda}$. The constant C_1 is set very small, so most edges that should be present are present, and we consider removal of unnecessary edges as a secondary goal.

We propose a mixed-integer programming (MIP) formulation for solving this. In what follows, decision variables $y_{\ell k}$ are binary, and they select a crime k that is most similar to a given crime ℓ within the same pattern. That is, they encode the max from constraint (3.2). The formulation is:

$$\min_{\boldsymbol{\lambda},\Delta,\{y_{\ell,k}\}_{\ell,k},\{\beta_j\}_j} \sum_{\ell \in \{1,2...m\}} \epsilon_\ell + C_1 \sum_{\substack{\ell \in \{1,2...m\} \\ \{k: k \notin \text{pattern}(\ell)\}}} \xi_{\ell k} + C_0 \sum_{j=1}^{J} \beta_j$$

such that

$$\sum_{j=1}^{J} \lambda_j s_j(\ell,k) \geq (\Delta - \epsilon_\ell) + M(y_{\ell k} - 1), \qquad \forall \ell,\ \forall k \text{ s.t. } k \in \text{pattern}(\ell) \tag{3.4}$$

$$\sum_{k:\ k \in \text{pattern}(\ell)} y_{\ell k} = 1 \qquad \forall \ell \tag{3.5}$$

$$y_{\ell k} \in \{0,1\}, \qquad \forall \ell, k \tag{3.6}$$

$$\sum_{j=1}^{J} \lambda_j s_j(\ell,k) \leq \Delta + \xi_{\ell k}, \qquad \forall \ell,\ \forall k \text{ s.t. } k \notin \text{pattern}(\ell) \tag{3.7}$$

$$\sum_{j=1}^{J} \lambda_j = 1, \tag{3.8}$$

$$\lambda_j \geq 0 \qquad \forall j \tag{3.9}$$

$$\lambda_j \leq \beta_j \qquad \forall j \tag{3.10}$$

$$\beta_j \in \{0,1\} \qquad \forall j. \tag{3.11}$$

Constraint (3.4) comes from (3.2). It forces $y_{\ell,k}$ to be chosen correctly so that if k is the

crime closest to ℓ, then $y_{\ell,k}$ will be 1. This is because ϵ_ℓ is minimized within the objective, so $y_{\ell,k}$ is necessarily going to correspond to the index where ϵ_ℓ is minimized, and where the similarity is maximized within (3.4). Constraints (3.5) and (3.6) further define $y_{\ell,k}$'s by stating that a crime in the pattern needs only to be connected to one closest neighbor ℓ in the pattern (this is the requirement of connectivity), and its entries are binary. Constraint (3.7) comes from (3.3). The value $\sum_j \beta_j$ is the ℓ_0 norm of $\boldsymbol{\lambda}$. The β_j's are decision variables where if $\beta_j = 1$, λ_j is non-zero. This formulation is linear, and thus using MIP technology there is a guarantee on the optimality of the solution.

3.4 Core Sets

A *d-core set* is a set of crimes that exhibit similarity in a feature subspace of d dimensions. A d-core set has d defining features that are not predetermined. Further, crimes in a d-core set need to be well connected in the similarity graph. Formally, the definition is as follows.

Definition 6 *(**d-core set**) A similarity graph $G = (V, E)$ with density threshold α is called a d-core set if it satisfies the core set constraints:*

- *Pattern specific constraint: the size of the defining feature set of the graph is equal to d, $|\Lambda(G)| = d$.*
- *Pattern general constraints: G is connected, and G is dense, $\frac{|E|}{|V|(|V|-1)} \geq \alpha$. That is, the fraction of possible edges in the graph exceeds α.*

Two parameters, d and α, control the property of the core set in a pattern-specific and pattern-general way, respectively.

The pattern-general constraints should be thought of as being much looser than the pattern-specific constraints, as we often include unnecessary edges in the similarity graph. For the set of crimes to be feasible in the pattern-specific sense is much more difficult, as crimes in the pattern need to be similar to each other in d separate ways.

We find core sets $G = (V, E)$ using an optimization method to maximize the pattern-

general cohesion while satisfying the core set constraints.

$$\begin{aligned} &\underset{G}{\text{maximize}} && \sum_{j \in J} \lambda_j \text{Cohesion}_j(G) \\ &\text{subject to} && |V| = n \\ &\text{Core set constraints:} && \begin{cases} |\Lambda(G)| = d, \\ G \text{ is connected}, \\ \frac{|E|}{|V|(|V|-1)} \geq \alpha. \end{cases} \end{aligned} \tag{3.12}$$

We propose a binary integer linear formulation for the optimization problem (3.12). Let $m = |\mathcal{V}|$, the number of crimes in the database. Let n be the size of the pattern we want to discover, and we loop through possible values of n, re-solving each time. We define an $m \times m$ similarity matrix for each feature $\mathbf{S}_1, \ldots, \mathbf{S}_J$ with elements $\mathbf{S}_j(\ell, k) = s_j(v_\ell, v_k)$. Let $\mathbf{X}$ be the matrix of binary decision variables defining the core set. $\mathbf{X}(\ell, k)$ is 1 if a pair of crimes ℓ and k are in core set G, which is the same as $\mathbf{X}(k, \ell)$, so matrix $\mathbf{X}$ is symmetric. On the diagonal, $\mathbf{X}(\ell, \ell)$ represents whether crime ℓ is in core set G. We use $d_1, \ldots, d_J \in \{0, 1\}$ to indicate whether feature j is a defining feature, or equivalently, whether $\text{Cohesion}_j(G) \geq \theta_j$. Let $\mathbf{E}$ be the adjacency matrix for the (pattern-general) similarity graph, where $\mathbf{E}(\ell, k) = 1$ if $\{v_\ell, v_k\} \in E$, and $\mathbf{E}(\ell, k) = 0$ otherwise. Since the graph is undirected, $\mathbf{E}$ is symmetric. We set $\mathbf{E}(\ell, \ell) = 1$ for computational simplicity. $(\mathbf{E} \circ \mathbf{X})$ is the Hadamard product of matrix $\mathbf{E}$ and $\mathbf{X}$, where $(\mathbf{E} \circ \mathbf{X})(\ell, k) = \mathbf{E}(\ell, k) \cdot \mathbf{X}(\ell, k)$. M is a large auxiliary parameter for formulating the problem with a big-M formulation, and ϵ is small. With this notation, the optimization problem (3.12) can be reformulated:

$$\max_{\mathbf{X}, \{\mathbf{d_j}\}_\mathbf{j}} \quad \sum_j \lambda_j \sum_{\ell,k} (\mathbf{S}_j \circ \mathbf{X})(\ell, k) \tag{3.13}$$

$$\text{s.t.} \quad \sum_\ell \mathbf{X}(\ell, \ell) = n \tag{3.14}$$

$$\frac{1}{n(n-1)} \sum_{\ell,k} (\mathbf{S}_j \circ \mathbf{X})(\ell, k) - M d_j \leq \theta_j - \epsilon \quad \forall j \tag{3.15}$$

$$\frac{1}{n(n-1)} \sum_{\ell,k} (\mathbf{S}_j \circ \mathbf{X})(\ell, k) - M d_j \geq \theta_j - M \quad \forall j \tag{3.16}$$

$$\sum_{j \in J} d_j = d \tag{3.17}$$

$$\mathbf{X}(\ell, k) = \mathbf{X}(k, \ell) \quad \forall \ell, k \tag{3.18}$$

$$\mathbf{X}(\ell, k) \leq \mathbf{X}(k, k) \quad \forall \ell, k \tag{3.19}$$

$$\mathbf{X}(\ell, \ell) + \mathbf{X}(k, k) \leq \mathbf{X}(\ell, k) + 1 \; \forall \ell, k \tag{3.20}$$

$$(\hat{\mathbf{E}}^{n-1} \circ \mathbf{X})(\ell, k) \geq \mathbf{X}(\ell, k) \quad \forall \ell, k \tag{3.21}$$

$$\mathbf{X}(\ell, k), d_j \in \{0, 1\} \quad \forall \ell, k, j, \tag{3.22}$$

where matrix $\hat{\mathbf{E}}$ is defined just below.

Let us derive the objective. Since $\mathbf{X}(\ell, k) = 1$ if and only if both crimes ℓ and k are in the core (that is they are in graph G that we discover), we have the following:

$$\frac{1}{n(n-1)} \sum_{\ell, k} (\mathbf{S}_j \circ \mathbf{X})(\ell, k) = \text{Cohesion}_j(G). \tag{3.23}$$

The objective is the pattern-general cohesion. Equation (3.14) ensures that the core sets we discover are of size n. Constraints (3.15), (3.16) and (3.17) are the pattern-specific constraints. Constraint (3.15) forces $d_j = 1$ when $\text{Cohesion}_j(G) \geq \theta_j$, where the strict inequality is enforced by ϵ. Constraint (3.16) forces $d_j = 0$ when $\text{Cohesion}_j(G) < \theta_j$. Constraint (3.17) specifies the number of defining features as d. The symmetry of $\mathbf{X}$ is enforced by (3.18). Constraints (3.19) and (3.20) imply $\mathbf{X}(\ell, k) = 1$ iff both $\mathbf{X}(\ell, \ell)$ and $\mathbf{X}(k, k)$ are 1 and 0 otherwise.

Expression (3.21) is a pattern-general constraint. Our formulation does not enforce the core set to be connected, but it does enforce something weaker, namely that each node in a core set of size n is at most $n - 1$ steps along the similarity graph from any other node in the core set. We handle the connectivity afterwards, by examining the result to ensure that it is connected, and labeling it as infeasible if not. To handle the constraint that each node in the core set is at most distance $n - 1$ from every other node, we recall that in graph theory, if node v_k is reachable from node v_ℓ in exactly q steps, $\mathbf{E}^q(k, \ell) > 0$. If node v_k is reachable from node v_ℓ in at most $n - 1$ steps, it means that at least one of $\mathbf{E}^q(k, \ell) > 0$ for $q \leq n - 1$. We define the following matrix $\hat{\mathbf{E}}^{n-1}$, where an element $\hat{\mathbf{E}}^{n-1}(k, \ell)$ indicates if

node k and node ℓ are distance at most $n-1$ steps along the graph.

$$\hat{\mathbf{E}}^{n-1}(\ell,k) = \begin{cases} 1 & (\mathbf{E}+\mathbf{E}^2+\cdots+\mathbf{E}^{n-1})(\ell,k) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Thus, (3.21) forces that if v_ℓ and v_k are both in the pattern, they must be at most distance $n-1$ along the graph.

The pattern-general density constraint is handled similarly to the connectivity constraint, where feasibility is checked for each solution, and infeasible solutions are removed.

The integer program finds one solution at a time. In order to avoid finding a solution that was found previously, we introduce a constraint for each previous solution found. Suppose in the t-th run, the crimes in the solution are $\mathcal{Q}_t$, i.e. $\mathbf{X}(k,k) = 1$ if $k \in \mathcal{Q}_t$, $\mathbf{X}(k,k) = 0$ otherwise. The constraint we add before running the $t+1$-th time is

$$\sum_{k \in \mathcal{Q}_t} \mathbf{X}(k,k) \leq n-1. \tag{3.24}$$

This constraint will exclude the current solution from the feasible region and we will obtain a different solution in the next run if any feasible solutions remain.

Since all of the matrices are symmetric, in practice we keep only the upper (or lower) triangle, including the diagonal, to compute all of the sums.

3.5 Merging Core Sets

By our main hypothesis, the vast majority of crime series contain a core set. This means that by finding all core sets, we would have located the vast majority of all crime series. We now grow the rest of the crime series from the core sets by merging them together. One advantage of merging is that it allows the pattern to dynamically change, as the defining features from the merged core sets are not always equal to each other. Consider a burglar's M.O. with a shifting means of entry. At first he enters through unlocked doors, then he starts to use bodily force to open doors, and later he learns to use a screwdriver to pry the

door open. His full pattern thus consists of several smaller d-core sets. This suggests a more flexible definition of pattern than a simple core set. We provide such a definition below.

Definition 7 *(d-**series-like pattern**) A graph $G = (V, E)$ is called a d-series-like pattern with defining feature set $\Pi(G)$ of size d if it satisfies:*

- *Pattern general constraint: G is connected.*
- *Pattern specific constraint: Each node u in G is contained in at least one subgraph $G' \subseteq G$ that is a d'-core set with defining features that include set $\Pi(G)$, that is $\Pi(G) \subseteq \Lambda(G')$, $d \leq d'$.*

Note that if a graph is a d-series-like pattern, it is also a $(d-1)$-series-like pattern, a $(d-2)$-series-like pattern, and so on. The pattern specific constraints for d-series-like patterns are looser than those for d-core sets. A d-series-like pattern may not be a d-core set. On the other hand, a d-core set is a special case of a d-series-like pattern. Before we proceed, we must ensure that merging is justified.

Theorem 1 ***(The set of series-like patterns is closed under merging.)*** *Suppose G_1 is a d_1-series-like pattern and G_2 is a d_2-series-like pattern. If $G_1 \cap G_2 \neq \emptyset$, then $\hat{G} = G_1 \cup G_2$ is a d-series-like pattern, with defining features $\Pi(G_1) \cap \Pi(G_2)$, $d = |\Pi(G_1) \cap \Pi(G_2)|$.*

Proof 1 *First, since G_1 and G_2 are connected and $G_1 \cap G_2 \neq \emptyset$, the union of them is also connected, i.e, $\hat{G}$ satisfies the pattern general constraint. Then for all nodes $u \in G_1$, $\exists$ a d_1-core set G_u^1 such that $u \in G_u^1$ and $\Pi(G_1) \cap \Pi(G_2) \subseteq \Pi(G_1) \subseteq \Lambda(G_u^1)$, and for all nodes $u \in G_2$, $\exists$ a d_2-core set G_u^2 such that $u \in G_u^2$ and $\Pi(G_1) \cap \Pi(G_2) \subseteq \Pi(G_2) \subseteq \Lambda(G_u^2)$. So, either way, the defining feature set includes $\Pi(G_1) \cap \Pi(G_2)$. This means that for any node $\hat{u} \in \hat{G}$, $\exists$ a core set $\hat{G}_u$ such that $\Pi(G_1) \cap \Pi(G_2) \subseteq \Lambda(\hat{G}_u)$.*

This leads directly to the following:

Corollary 1 *Suppose G_1 is a d_1-series-like pattern, G_2 is a d_2-series-like pattern, ..., G_n is a d_n-series-like pattern, and $G_1 \cup ... \cup G_n$ is connected,*

$$|\Pi(G_1) \cap \Pi(G_2) \cap ... \cap \Pi(G_n)| = d.$$

Then $G_1 \cup ... \cup G_n$ is a d-series-like pattern.

These properties lead to the following breadth first search algorithm for mining d-series-like patterns. We start with the core sets that we found using the integer program. These core sets are *candidates* for merging. We also maintain an *active pattern set* that contains the d-series-like patterns that we are not done constructing. We keep the d-series-like patterns that we are done constructing in a *maximal pattern set.* To start, the active pattern set contains all of the d-core sets we found. For each active pattern, we iterate through the candidates to see if they meet the merging criteria provided just below. If a merge is possible, and if the merged set had not been previously created, we append the merged pattern to the active pattern set, and continue iterating through the candidates. If there are no candidates that can be merged with the active pattern at all, then the active pattern is maximal, and it is placed in the maximal pattern list. The *merging criteria* for $G_1 \cup G_2$ to form a d-series-like pattern $\hat{G}$ is

- $G_1 \cap G_2 \neq \emptyset$
- $|\Pi(G_1) \cap \Pi(G_1)| \geq d.$

The merging algorithm is formulated in Algorithm 1.

3.6 Experiments

Our data set was provided by the Crime Analysis Unit of the Cambridge Police Department in MA, USA. It has 7,067 housebreaks that happened in Cambridge between 1997 and 2011, containing 51 hand-curated patterns contained within the 4,864 crimes between 1997 and 2006. (Patterns from 2007 to 2012 were not assembled at the time of writing.) Crime attributes include geographic location, date, day of week, time frame, location of entry, means of entry, an indicator for "ransacked," type of premise, an indicator for whether residents were present, and suspect and victim information. Data were processed using the similarity functions $\{s_j\}_j$ discussed in our previous work [93] where each pairwise feature is mapped into a number between 0 and 1. These similarity measures are p-values, and they

Algorithm 1 Merging core sets

INPUT: d, core sets, each with $\geq d$ defining features
candidate list $\leftarrow$ core sets
active set $\leftarrow$ core sets, each with defining features
maximal set $\leftarrow \emptyset$
while active set $\neq \emptyset$ **do**
  $G_{\text{current}} \leftarrow$ any element in active set
  $\Pi_{\text{current}} \leftarrow$ defining features from G_{current}
  isMaximal $\leftarrow$ TRUE;
  for $G_j \in$ candidates, $G_j \not\subset G_{\text{current}}$ **do**
    if $G_{\text{current}} \cap G_j \neq \emptyset$, $|\Pi(G_{\text{current}}) \cap \Lambda(G_j)| \geq d$ **then**
      $\hat{G} \leftarrow G_{\text{current}} \cup G_j$
      $\Pi(\hat{G}) \leftarrow \Pi(G_{\text{current}}) \cap \Lambda(G_j)$
      if $\hat{G}$ does not exist in active set or maximal set **then**
        isMaximal $\leftarrow$ FALSE;
        append $\hat{G}$ to active set
  end for
  if isMaximal==TRUE **then**
    remove G_{current} from active set, put into maximal set
OUTPUT: maximal set

consider the baseline frequency of each possible outcome for the categorical variables. We took the 51 hand-curated patterns, and divided them randomly into four subsets (folds) with sizes 12 or 13 patterns each. We used 3 of the 4 folds to learn the pattern-general weights and tested on the remaining fold for the experiments discussed below.

Baselines

As this problem is fundamentally a clustering problem, we compare with several varieties of hierarchical agglomerative clustering and incremental nearest neighbor approaches. For these baselines, we use several different schemes to iteratively add discovered crimes, starting from pairs of nodes with high similarity γ, which is a weighted sum of the attribute similarities:

$$\gamma(C_i, C_k) = \sum_{j=1}^{J} \hat{\lambda}_j s_j(C_i, C_k).$$

Unlike our method where the weights are learned, the weights $\hat{\boldsymbol{\lambda}}$ for the baselines were provided by crime analysts based on domain expertise, similar to several other works [52,

64].

Hierarchical agglomerative clustering (HAC) begins with each crime as a singleton cluster, and iteratively merges the clusters based on the similarity measure between clusters. *Nearest neighbor classification* (NN) first selects pairs of crimes with high similarity and then iteratively grows a cluster by adding the nearest neighbor crime to the cluster.

HAC and NN were used with three different criteria for cluster-cluster or cluster-crime similarity: *Single Linkage* (SL), which considers the most similar pair of crimes, *Complete Linkage* (CL), which considers the most dissimilar pair of crimes, and *Group Average* (GA), which uses the averaged pairwise similarity [36]. When the nearest neighbor algorithm is used with the S_{GA} measure defined below with weights provided by crime analysts, it is similar to the Bayesian Sets algorithm and how it is used for set expansion [30, 47].

$$\begin{aligned} S_{SL}(G_1, G_2) &:= \max_{v_k \in G_1, v_\ell \in G_2} \gamma(v_k, v_\ell) \\ S_{CL}(G_1, G_2) &:= \min_{v_k \in G_1, v_\ell \in G_2} \gamma(v_k, v_\ell) \\ S_{GA}(G_1, G_2) &:= \frac{1}{|G_1||G_2|} \sum_{v_k \in G_1} \sum_{v_\ell \in G_2} \gamma(v_k, v_\ell). \end{aligned} \tag{3.25}$$

Evaluation metrics

There are two levels of performance we evaluate - *pattern-level* and *object-level.*

Pattern level precision and recall

We evaluate the quality of the core set detector using "pattern-level" precision and recall. The d-core sets are smaller as d becomes larger. The core sets are used for discovering larger merged patterns. Thus we evaluate the accuracy of the core set finder in its detection ability; if a real pattern is missed completely by our core set detector, there is no way to recover from this in order to detect it. If a core set covers more than one pattern, this is also a bad seed for further mining, since it would generate misleading defining features that do not characterize any real patterns. Thus, we call core sets that cover one and only one real pattern *good* core sets. The pattern level precision and recall are both defined using good

core sets. N denotes the number of core sets we discover.

$$\text{P-Precision (core sets)} = \frac{\sum_i^N \mathbf{1}(\text{core set } i \text{ is } \textit{good})}{N} \tag{3.26}$$

$$\text{P-Recall (core sets)} = \frac{\sum_i^N \mathbf{1}(\text{core set } i \text{ is } \textit{good})}{|\mathcal{P}|}. \tag{3.27}$$

Note that pattern level precision should be large, as each real pattern should contain many core sets, inflating the reported precision values.

Object-level precision and recall

We evaluate the full pipeline for generating series-like patterns using "object-level" precision and recall. To do this, for each pattern discovered, we determine how close it is to one of the real patterns. If the discovered pattern overlaps only one real pattern, then we call this the *dominating pattern* and evaluate precision and recall with respect to crimes in that pattern. If the series-like pattern overlaps more than one real pattern, we assign the dominating pattern to be the real pattern possessing the most crimes that overlap with our discovered pattern. Note that it is possible for the recall not to grow with the size of the discovered pattern, as the dominating real pattern could change as the discovered pattern grows larger. The definitions of object-level precision and recall for a d-series-like pattern $G = (V, E)$ are as follows:

$$\text{O-Precision(G)} = \frac{\sum_{\ell=1}^{|V|} \mathbf{1}(\ell \in \text{dominating pattern})}{|V|} \tag{3.28}$$

$$\text{O-Recall(G)} = \frac{\sum_{\ell=1}^{|V|} \mathbf{1}(\ell \in \text{dominating pattern})}{|V_{\text{dominating pattern}}|} \tag{3.29}$$

where $|V_{\text{dominating pattern}}|$ is the number of crimes in the dominating real pattern.

Computational gain from similarity graph

The first step in our method is to learn the similarity graph. The similarity graph provides a computational gain in that it creates constraints on possible core sets, reducing the feasibility region of the ILP. For this similarity graph, recall that we desire crimes in the same real

pattern to be connected to each other. We call edges connecting crimes that belong to the same pattern *good edges*. If we have constructed the similarity graph well, the similarity graph should have a higher percentage of good edges than if we had simply used the full graph consisting of all possible edges. If we remove a few good edges in the process, this is not problematic as long as the true patterns are still connected in the similarity graph - this will be assessed when we assess the quality of the core sets and the full pipeline next.

Table 3.1 shows the percentages of good edges in both the similarity graph and the full graph, for four test folds (the data were divided into four folds, and each was used in turn as the test fold). The learning method tends to reduce the number of unnecessary edges by a factor of 7 or 8 in each of the test folds, as shown in the third column (which is the first column divided by the second column). This reduction substantially reduces computation for the core set finder.

Table 3.1: Reduction of complexity by constructing similarity graphs on test data

	Good edges% in similarity graphs	Good edges % in complete graphs	Reduction factor	Training time
1	1.84	0.26	7.1	3.64×10^3 s
2	2.42	0.33	7.3	9.58×10^3 s
3	3.34	0.42	8.0	6.20×10^3 s
4	2.95	0.34	8.7	5.41×10^3 s

Pattern-general weights

The pattern-general weights come from the learning step for the similarity graph. In Figure 3-2 we report the mean over the test folds of the pattern-general weights we discovered. The highest weights are similarity in distance, number of days apart, suspect information, whether residents are present, and means of entry (e.g., pried, forced, cut screen).

Time windowing

Consider solving the ILP for finding core sets on data from 4,864 housebreaks. Note that if we were to search for patterns of size 10 among 1,000 crimes, this would mean investigating

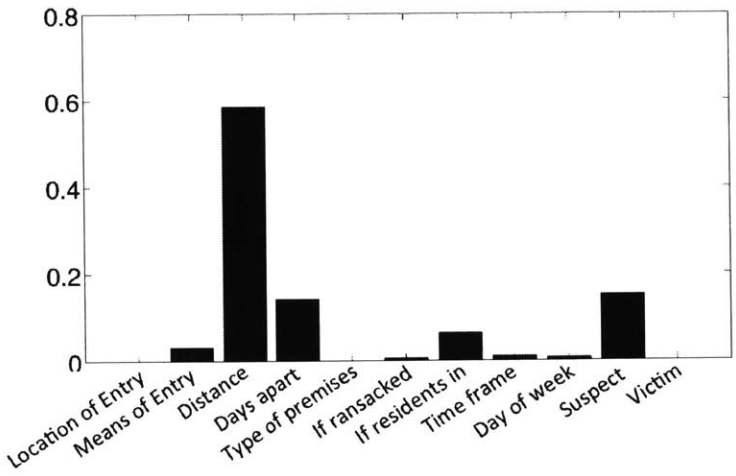


Figure 3-2: Means of pattern general weights $\{\lambda_j\}_j$

$\binom{1000}{10} \approx 2.634 \times 10^{23}$ possible subsets. Further, CPLEX would need to handle $1{,}000^2$ constraints (3.19) and (3.20) of the optimization problem. Luckily it is unlikely that a crime series would possess a core of size 10, but still we need to find ways to reduce computation.

Because the pattern-general weight on closeness in time is so high, we determined that we would be unlikely to miss true core sets if we considered windows of time that include at least 200 crimes. We thus indexed the housebreak records in chronological order and created overlapping windowed blocks of 200 crimes each, where neighboring blocks have an overlap of 100 crimes. Therefore we solve ILPs among crime subsets $\{1, \cdots, 200\}$, $\{100, \cdots, 300\}$, $\cdots$, $\{4700, \cdots, 4864\}$. In each subset, we input the number of defining features d and core set size n, and then iteratively run the ILP to get all feasible solutions, by adding the constraint (3.24) after each iteration to avoid returning repeated solutions.

3.6.1 Evaluation of Mining Core Sets

We chose performance evaluation metrics from information retrieval, and for some of these metrics, we need to rank the discovered core sets by a scoring function. This scoring function represents how certain we are that these core sets are real. We use a scoring function that is a weighted version of pattern-general cohesion and the (pattern-specific) number of defining features, as we desire core sets that are both tight in the pattern-general

sense and in the pattern-specific sense. Here is the score function:

$$\text{Score}(G) = \sum_{j=1}^{J} \lambda_j \text{Cohesion}_j(G) + \frac{1}{6} \cdot d. \tag{3.30}$$

We ordered the discovered core sets in decreasing order of the scores. Note that the first evaluation below does not require these scores, but the second and third do.

1) Core Sets with different d We expect that discovered patterns with more defining features are more likely to be true crime series. Figure 3-3 shows how the precision increases with the number of defining features d. In this figure we consider only core sets of the same size (3 crimes). There are no overlapping core sets between the three bars, as each core set is used once with its exact number of defining features d, which is 6, 7, or 8. The number of discovered core sets for $d = 6$ is 1072, $d = 7$ is 215 and for $d = 8$ is 36. There were too few core sets with more than 8 defining features to reliably calculate precision. The reported numbers of core sets are totals from all test folds.

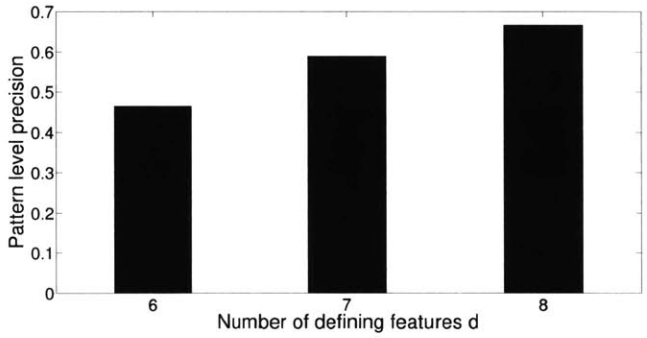


Figure 3-3: Pattern-level average precisions of core sets with different number of defining features d.

2) Core Sets with different size n We used the scoring function (3.30) as a filter to pick the best 1,000 core sets, from each of sizes 3, 4 and 5 and discarded the other discovered score sets. Larger core sets have higher chances of hitting a pattern, since there are more crimes in the core set; however, they also have higher chances of hitting more than one real pattern. As shown in Figure 3-4, core sets of size 4 have a much higher pattern-level

precision than core sets of size 3, but core sets of size 5 do not have noticeable gains over size 4. This is because the increased probability of hitting a real pattern cancels with the increased probability of hitting more than one real pattern.

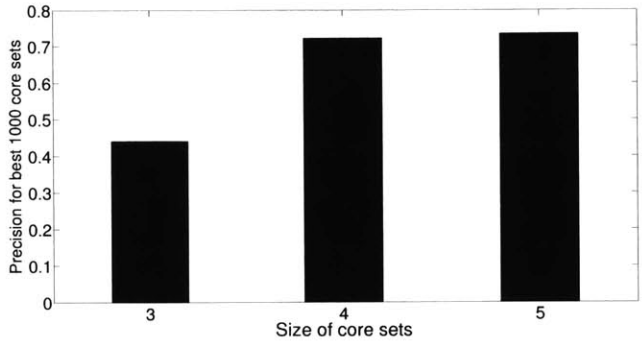


Figure 3-4: Pattern-level average precisions of core sets with different sizes

3) P-Precision - P-Recall curve We generated a full list of core sets of size 3 with d between 6 and 8, and ranked the core sets according to their scores. As we moved down the list, we evaluated pattern-level precision and recall at each step. We also did the same procedure with the baseline iterative nearest neighbor method used for generating core sets of size 3, using all of the similarity measures in (3.25). (Note that for HAC we cannot control the size of core sets for evaluation.) The precision-recall curves for all four methods averaged over the test folds are plotted in Figure 3-5. It is clear that our core set finder is substantially better than the baselines, though that is not surprising given that it searches globally for the best core sets.

3.6.2 Evaluation for Mining Series-Like Patterns

We evaluated the quality of our full pipeline and the baseline methods as follows. After all the series-like patterns were discovered, we evaluated the average object-level precision and recall for all the patterns and over all the test folds, plotted as a point on Figure 3-6. For HAC, we simply iterated it, stopping at a threshold where recall was approximately equivalent to our method, and again reported average object-level precision and recall on Figure 3-6. For the nearest neighbor method, after each element was added to a growing

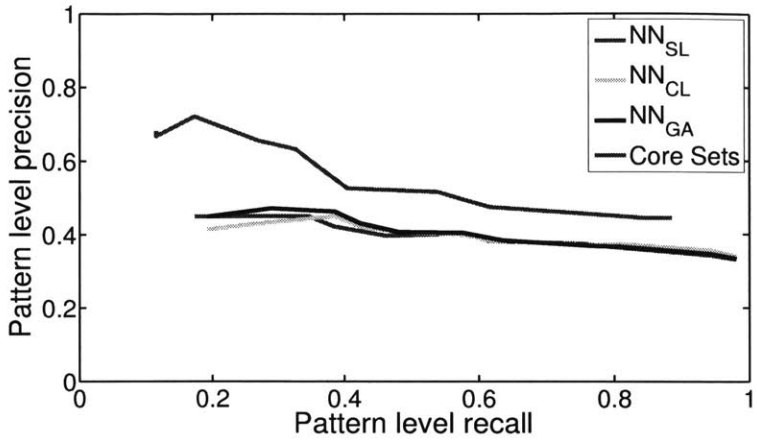


Figure 3-5: Average pattern-level precision vs. recall for core sets and nearest neighbor algorithms

pattern, we evaluated precision and recall to trace out a precision-recall curve. All three metrics in (3.25) were used for HAC and nearest neighbors. We note that for the same level of recall, the precision attained by our method was quite a bit higher than that of other methods.

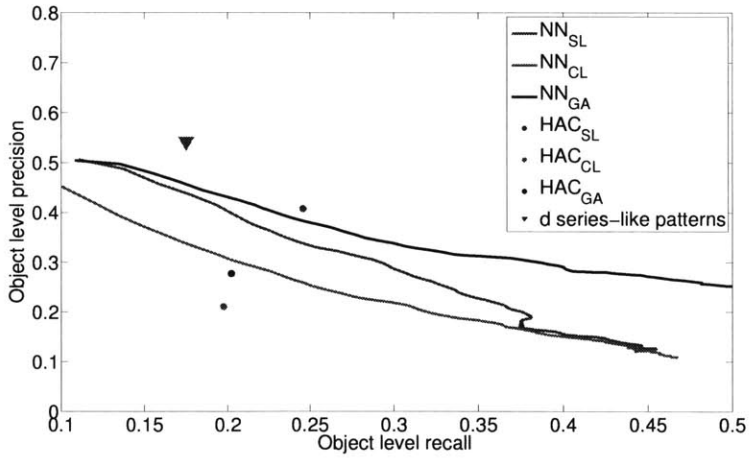


Figure 3-6: Object-level precision and recall for core sets, nearest neighbor algorithms and hierarchical agglomerate clustering algorithms

There is still a lot of room for improvement, as with precision on the order of 53%, we capture approximately 18% of the crimes identified by analysts. This might be improved

by making the core set finder less conservative, finding a way to incorporate crimes that are not in cores, or possibly by working with domain experts to improve the database used for evaluation. We note that the baseline methods work surprisingly well, and are themselves reasonable options in practice.

3.7 Case Studies

We performed a blind test, where we aimed to detect crime patterns between 2007 to 2012 for which we do not have pattern data. The results were analyzed by hand by crime analysts.

Case Study 1

One particularly interesting crime series includes 10 crimes from November 2006 to March 2007. Figure 3-7 shows geographically where these crimes were located. Table 3.2 provides some of the details about the crimes within the series.

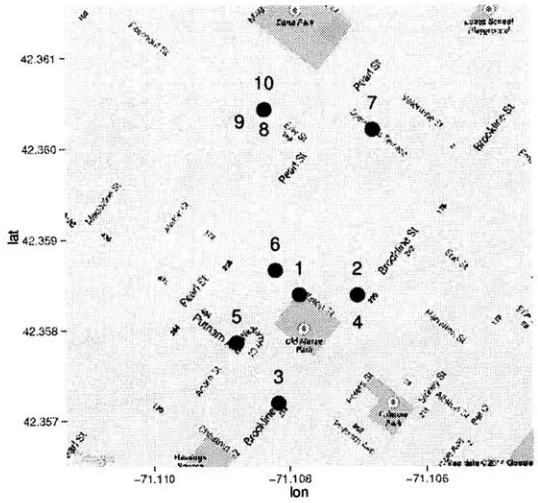


Figure 3-7: The locations of crimes in first series.

From the (pattern-general) similarity graph, we isolated the subgraph containing the 10 crimes, which is diagrammed in Figure 3-8. Crimes 1 to 5 are well connected as a subset, and crimes 6 to 10 are well connected as another subset. From only the similarity graph,

Table 3.2: Attributes for the first crime series.

No.	Date	Location of entry	Means of entry	Type of premises	If ransacked	If resident in	Time of day	Day	Suspect	Victim
1	11/8/06	Basement Door	Unknown	Unknown	No	Not in	10:45-15:00	Wed	null	1 F
2	11/8/06	Front door	Pried	Unknown	No	Not in	8:00-18:30	Wed	null	1 M
3	11/16/06	Front door	Shoved/Forced	Unknown	No	Not in	9:00-17:00	Thur	null	1 M
4	12/7/06	Front door	Pried	Unknown	No	Not in	9:00-17:00	Thur	null	1 F
5	12/22/06	Front door	Pried	Unknown	No	In	11:48	Fri	null	1 M
6	2/1/07	Front door	Shoved/Forced	Unknown	No	In	14:45	Thur	3 Males	1 F
7	2/15/07	Front door	Unknown	Aptment	No	In	12:00-13:30	Thur	null	2 F
8	3/5/07	Front door	Shoved/Forced	Aptment	No	Not in	12:22-14:56	Mon	null	White M
9	3/5/07	Front door	Broke	Aptment	No	Not in	12:22-14:56	Mon	null	1 F & 1 M
10	3/8/07	Front door	Pried	Aptment	No	Not in	12:50-13:30	Thur	null	1 M

the two subsets do not seem very related except for a single edge between crimes $\{5, 6\}$ connecting them; however, this is only the pattern-general part of the story.

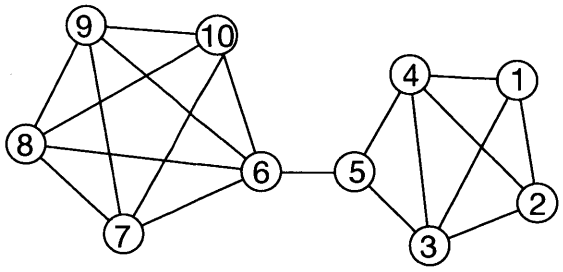


Figure 3-8: Similarity graph for the first crime series.

We used the integer linear program (3.12) to discover cores of size 3 with at least 6 defining features. Table 3.3 lists the cores and their defining features in this series, where "1" means the feature is a defining feature and "0" means it is not. These cores show how the crimes are similar to each other in a pattern-specific way. The next step is merging the cores. The defining feature set $\Pi(G)$ was chosen to include 6 features, which are geographic location, days apart, location of entry, the ransacked indicator, time of day, and day of the week. One of the cores, core 16 in Table 3.3, was not included as geographic location is not a defining feature for that core, and the rest of the cores were merged. (The same set of crimes is in the merged pattern regardless of whether core 16 was used for the merge.)

Table 3.3: Cores and their defining features for the first crime series

Cores	Crimes	Geographical location	Days apart	Location of entry	Means of entry	Type of premises	If ransacked	If resident in	Time of day	Day	Suspect	Victim
1	1 2 3	✓	✓	✓	o	o	✓	o	✓	✓	o	o
2	1 2 4	✓	✓	✓	o	o	✓	o	✓	✓	o	o
3	1 2 5	✓	✓	✓	o	o	✓	o	✓	✓	o	o
4	1 3 4	✓	✓	✓	o	o	✓	o	✓	✓	o	o
5	1 3 5	✓	✓	✓	o	o	✓	o	✓	✓	o	o
6	1 4 5	✓	✓	✓	o	o	✓	o	✓	✓	o	o
7	1 5 6	✓	✓	✓	o	o	✓	o	✓	✓	o	o
8	2 3 4	✓	✓	✓	o	o	✓	o	✓	✓	o	o
9	2 3 5	✓	✓	✓	o	o	✓	o	✓	✓	o	✓
10	2 4 5	✓	✓	✓	✓	o	✓	o	✓	✓	o	o
11	2 4 6	✓	✓	✓	o	o	✓	o	✓	✓	o	o
12	2 5 6	✓	✓	✓	o	o	✓	o	✓	✓	o	o
13	3 4 5	✓	✓	✓	o	o	✓	o	✓	✓	o	o
14	3 5 6	✓	✓	✓	o	o	✓	o	✓	✓	o	o
15	4 5 6	✓	✓	✓	o	o	✓	o	✓	✓	o	o
16	5 6 7	o	✓	✓	o	o	✓	✓	✓	✓	o	o
17	6 8 9	✓	✓	✓	o	o	✓	o	✓	✓	o	o
18	6 8 10	✓	✓	✓	o	o	✓	o	✓	✓	o	o
19	6 9 10	✓	✓	✓	o	o	✓	o	✓	✓	o	o
20	7 8 9	✓	✓	✓	o	✓	✓	o	✓	✓	o	o
21	7 8 10	✓	✓	✓	o	✓	✓	o	✓	✓	o	o
22	7 9 10	✓	✓	✓	o	✓	✓	o	✓	✓	o	o
23	8 9 10	✓	✓	✓	o	✓	✓	o	✓	✓	o	o

As these data were reconsidered by crime analysts, we found out that when these crimes were analyzed back in 2006-2007, they were viewed as two unrelated patterns, one at the end of 2006, crimes 1 to 5, and one at the beginning of 2007, crimes 6 to 10. The connection between these two subsets of crime is very subtle and there is over a month gap between the two patterns, so it did not occur to the crime analysts to link them. Their intuition agrees completely with the similarity graph, as the two subsets are weakly connected only by one edge; however, recall that this only describes the pattern-general similarity - what one would expect from generic pattern without considering a specific M.O. On examination of the cores, not only are they correlated in 6 features, but five of the cores (core indices 11, 12, 14, 15, 16) contain crimes from both of the subsets, which is strong evidence that the two subsets should be merged together. It is particularly interesting that the core consisting of crimes 5, 6, and 7 spanned the two subsets, where these crimes share the unusual feature that residents were present during the break-in.

We wondered why the offenders changed their M.O. to move a few blocks north since they committed crimes 1-6 in the same area. What may have happened is that the criminals left near the end of December (just before the holidays), and returned in February to commit housebreak number 6 in the same area as 1-5; however they were witnessed committing crime 6 (suspect information in crime 6 reads "3 males"). This may have spooked the offenders, causing them to alter their M.O. by moving north to commit crimes 7-10. Analysts now believe that these two series were actually a single series, and that the suspect information from crime 6 can be carried through to all the crimes in the discovered series.

This is a good example to show how crime patterns can be composed of cores, and exhibit similarity both in a pattern-general way and pattern-specific way. It shows how we can use both aspects to mine patterns. This is a pattern that would be very difficult for a crime analyst to find: the M.O. changes over time, there was a long break in the middle of the series, and there was nothing deterministic (e.g., fingerprints) linking these crimes.

Table 3.4: Attributes for the second crime series

No.	Date	Location of entry	Means of entry	Type of premises	If ransacked	If resident in	Time of day	Day	Suspect	Victim
1	1/25/07	Front Door	Punched/Popped	Apartment	No	Not in	10:20-12:00	Thur	null	1 F
2	1/25/07	Unknown	Cut Screen	Apartment	No	Not in	8:45-14:30	Thur	null	1 M
3	1/25/07	Unknown	Pried	Apartment	No	Not in	9:10-21:00	Thur	null	1 F
4	1/25/07	Front door	Unlocked	Apartment	No	In	13:00-13:30	Mon	null	1 F
5	1/29/07	Front door	Key	Apartment	No	Not in	14:52-14:52	Mon	null	2 M & 3 F
6	1/29/07	Unknown	Unknown	Apartment	No	Not in	12:00-12:00	Mon	1 M	1 M
7	1/29/07	Unknown	Unknown	Apartment	No	Not in	15:00-15:00	Mon	null	2 M

Case Study 2

Table 3.4 and Figure 3-9 shows a more typical pattern in 2007 discovered by our method. The crimes were committed on two dates in late January of 2007, most of them in the same building. According to the Cambridge police, they arrested a suspect while he was committing the last crime in the series and confirmed that he did commit crimes 2-7. Note that the record for the fourth crime records the means of entry as "key." This is based only on the claim of a witness that the offender possesses a master key. If the offender did possess a master key to the apartments in the building, it would explain why the means of entry was unknown for other crimes in the series - the means of entry would thus have been very difficult to determine for earlier crimes where residents were not present.

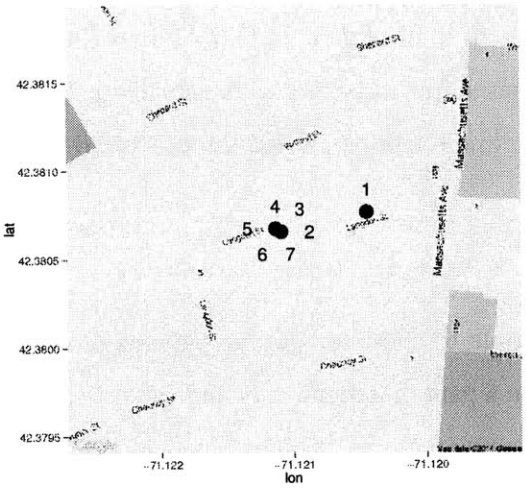


Figure 3-9: The locations of crimes in second series.

Table 3.5: Attributes for the third crime series

No.	Date	Location of entry	Means of entry	Type of premises	If ransacked	If resident in	Time of day	Day	Suspect	Victim
1	2/9/05	Ground window	Shoved/Forced	Apartment	No	In	1:47	Wed	1 white M	1 F
2	2/9/05	Rear door	Shoved/Forced	Single-Family House	No	In	9:50	Wed	1 black M	1 M & 1 F
3	2/15/05	Ground window	Broke	Apartment	Yes	Not in	7:00-13:30	Tue	null	2 M
4	2/21/05	Front door	Key	Unknown	No	Not in	7:10-10:00	Mon	null	1 F
5	2/23/05	Front door	Pried	Apartment	No	Not in	7:10-16:00	Wed	null	2 M
6	2/23/05	Front door	Pried	Apartment	No	Not in	7:00-14:00	Wed	null	2 M
7	2/23/05	Front door	Pried	Apartment	No	Not in	7:45-17:25	Wed	null	2 F
8	2/28/05	Rear door	Unknown	Apartment	No	Not in	20:55	Mon	1 white M	1 F

Table 3.6: Cores and their defining features for the third crime series

Cores	Crimes	Geographical location	Days apart	Location of entry	Means of entry	Type of premises	If ransacked	If resident in	Time of day	Day	Suspect	Victim
1	1 2 8	o	✓	✓	o	✓	✓	o	o	✓	o	o
2	2 5 6	o	✓	✓	o	✓	✓	o	✓	✓	o	o
3	2 5 7	o	✓	✓	o	✓	✓	o	✓	✓	o	o
4	2 5 8	o	✓	✓	o	✓	✓	o	o	✓	o	o
5	2 6 7	o	✓	✓	o	✓	✓	o	✓	✓	o	o
6	2 6 8	o	✓	✓	o	✓	✓	o	o	✓	o	o
7	2 7 8	o	✓	✓	o	✓	✓	o	o	✓	o	o
8	5 6 7	✓	✓	✓	✓	✓	✓	o	✓	✓	o	✓
9	5 6 8	✓	✓	✓	o	✓	✓	o	o	✓	o	o
10	5 7 8	✓	✓	✓	o	✓	✓	o	o	✓	o	o
11	6 7 8	✓	✓	✓	o	✓	✓	o	o	✓	o	o

Case Study 3

Our method has also been used to help ensure the fidelity of the historical database of past crimes. We ran our method using data collected prior to 2006 to see whether we would be able to make discoveries.

Table 3.5 shows details from crimes within a 2005 pattern, discovered by both the police and our algorithm. Our algorithm and the crime analysts agreed on six out of the eight crimes in the series, but disagreed on two crimes: crime 3 and crime 4. The crime analysts identified these crimes as part of the pattern, but our algorithms did not identify these crimes as being part of the pattern. Our algorithm provides reasons why these crimes should be excluded from the pattern: they are not close to other crimes in the pattern-general sense, and are not connected to the other crimes in the series within the similarity graph, as depicted in Figure 3-11. Neither of the crimes are contained in any cores. In particular, the map in Figure 3-10 shows that these two crimes are geographically far away

from the other crimes. Since geographic closeness has a large contribution to pattern-general similarity, crimes 3 and 4 are already not likely to be part of the same series. Besides that, we also notice that other aspects of crimes 3 and 4 differ from the rest of the pattern (see Table 3.5). In crime 4, the means of entry is by key which is different from that of other crimes where entries were forced or pried. According to police narratives, the fourth crime had a 91 year old victim, who reported that $35 was stolen from her purse by someone who used a key to enter the apartment; this indicates the crime was not part of the series, and further indicates the possibility that this crime never actually occurred. In crime 3, the apartment was ransacked while none of the other locations were. The narrative for the third crime is very generic: someone broke in through the living room window, stole jewelry and loose change and exited through the front door. It is not clear that this crime possesses any distinguishing characteristics to identify it as being part of the series.

When examining the crime series retrospectively, the police agreed that crimes 3 and 4 likely should be excluded from the pattern (note that this pattern has not been officially verified through an arrest).

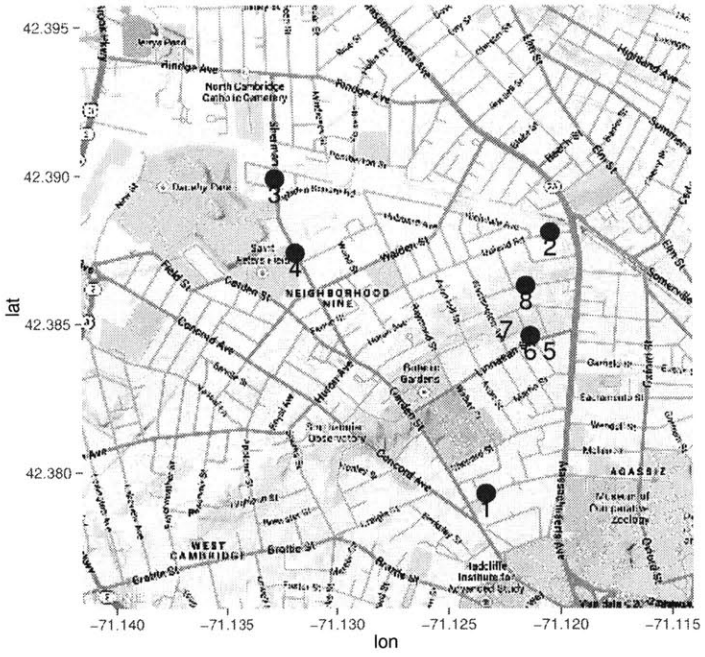


Figure 3-10: The locations of crimes in third series.

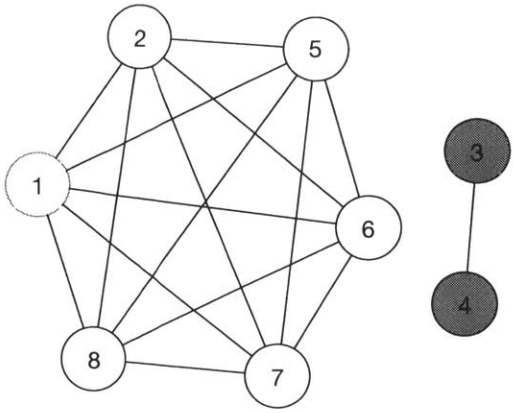


Figure 3-11: Similarity graph for third crime series

3.8 Conclusions

The problem of detecting crime series is both subtle and difficult. In cases where crime series detection is trivial, meaning that there is identifiable information about the criminal (e.g., fingerprints or DNA) then sophisticated modeling techniques are not needed - the solution is direct in those cases. The more subtle cases where crime series might otherwise become hidden in a sea of data are where data mining methods can shine.

Compared with Series Finder that depends on "seed" crimes to initialize a set, the second method is fully automated that it does not rely on human input but can complete the entire task on its own. Series finder produces a set of weights to represent the importance of different attributes and the clustering method directly selects a set of important features as the defining feature, both of which are informative and clear to crime analysts.

Our automated series detection methods are able to detect crime patterns within big data sets where a human could not. Detecting patterns among crime data is critical to effective policing. When a pattern is identified, police can implement effective strategies to prevent future crimes, solve past crimes, identify and capture offenders, and ensure offenders are investigated and prosecuted fully for the crimes for which they are responsible. Our methods could be used very broadly for crime series detection, and would be particularly useful across regions or districts, where human crime analysts would fail to be able to manually handle vast amounts of data from several different regions or data sources in order to detect

patterns.

We envision that out pattern detection methods could potentially have broader application to not just crime data mining, but other pattern detection problems where a pattern is defined by subsets of "similar" observations and the subsets are task-oriented. Such application include disease outbreak detection where a pattern refers to a set of patients who are infected and display similar symptoms, or detecting patterns in consumers with similar interest/taste in their transaction history.

Chapter 4

Bayesian Rule Set

4.1 Introduction

Our goal is to construct an interpretable predictive model, Rule Set, that not only classifies but also "explains" data. This classification model consists of a *small* set of rules connected by *or*s, and each rule is a *small* set of conditions connected by *and*s. The model predicts *positive* if at least one rule is satisfied and predicts *negative* if none of the rules is satisfied. An example of rule set model for predicting if a customer will accept a coupon for a coffee house recommended via a mobile recommender system is as follows:

if a customer (goes to coffee houses more than once per month AND has no urgent destination AND passenger is NOT kids)
OR (goes to coffee houses more than once per month AND the coupon expires in one day) **then**
the customer will accept the coupon for a coffee house

This model is interpretable to humans for using simple "if ... then ..." rule and a symbolic formulation of rules via 'and's and 'or's to classify data in a way that is consistent with how humans would think and reason. [38] shows that this form of model is natural for modeling consumer behavior. In particular, it has been hypothesized that people make purchasing decisions using simple *or*'s of *and*'s models (e.g., "I would only purchase product X if it has options 1 and 2 or options 3 and 4.").

Compared with other interpretable models, rule set models have two major advantages, accuracy and user-controlled interpretability. Some models such as decision trees (e.g., C4.5, CART), inductive rule learners (e.g., RIPPER [18]) and associative classification methods [14, 16, 50, 98] use heuristics to grow a model greedily, which often compromises the predictive accuracy. Besides, they do no include interpretability as part of the objective or heuristics, and therefore often generate models with heavy cognitive load to understand. However, sparsity is a major necessity for interpretability, as shown in [59] that humans can handle about 7 ± 2 cognitive entities at once. It is well-known that for many domains, the space of good predictive models is often large enough to include very simple models [39]. Rule Set models aim to include both aspects in the overall objective to produce models that are both accurate and cognitively simple.

Note that logical models are generally robust to outliers and naturally handle missing data, with no imputation needed for missing attribute values. These methods can perform comparably with traditional convex optimization-based methods such as support vector machines or lasso (though linear models are not always considered to be interpretable in many domains).

We propose generative approaches for learning Rule Set models. These methods strike a nice balance between accuracy and interpretability through user-adjustable prior parameters. To control interpretability, we introduce two types of priors on Rule Set models, a model with beta-binomial priors, called BRS-BetaBinomial, and a model with Poisson priors, called BRS-Poisson. Both can be adjusted to suit a domain-specific notion of interpretability; it is well-known that interpretability comes in different forms for different domains [4, 29, 41, 56, 57, 76].

We provide an approximate inference method that uses association rule mining and simulated annealing to find optimal BRS maximum a posteriori (MAP) models. This approach is motivated by a theoretical bound that allows us to reduce the size of the computationally hard problem of finding a MAP solution. This bound states that we need only mine rules that are sufficiently frequent in the database, which greatly reduces computation complexity, and allows the problem to be solved in practical settings. The theoretical result takes advantage of the strength of the Bayesian prior.

Our applied interest is to understand user responses to personalized advertisements that are chosen based on the user, the advertisement, and the context. Such systems are called *context-aware recommender systems* (see survey [1, 2, 7, 85] and references therein.) One major challenge in the design of recommender systems, reviewed in [85], is the *interaction challenge*: users typically wish to know *why* certain recommendations were made. Our work addresses precisely this challenge: our models provide rules in data that describe conditions on which a recommendation will be accepted.

The main contribution of our chapter are as follows:

- We propose a Bayesian approach for learning Rule Set classifiers. This approach incorporates two important performance metrics, accuracy and interpretability, and uses a Bayesian structure to balance between them via user-defined parameters.

- We derive bounds on the support of rules in a MAP solution. These bounds are useful in practice because we can safely reduce the solution space to a much smaller subset thus greatly improving computation efficiency.

- We derive bounds on the number of the rules in a MAP solution. These bounds guarantees a sparse solution if appropriate parameters are chosen for the model. In addition, we present generalization bounds that relate the size of the coefficient set to a uniform guarantee on testing accuracy.

- We ran experiments on Amazon Mechanical Turk to collect data in a mobile recommendation system. Rule Set models are used to understand and analyze consumers' behavior and predict their response to different coupons recommended in different contexts. We generate interpretable results that can help domain experts gain insights from data and understand their customers.

- We present results from experiments that compare Rule Set models to interpretable and uninterpretable models from other popular classification methods on publicly available datasets. Our results suggest that BRS models can achieve competitive performance and they are most favored when data is generated from a set of underlying rules.

The remainder of this chapter is structured as follows. In Section 2, we discuss related work. In Section 3, we present Bayesian Rule Set modeling. In Section 4, we introduce approximate MAP inference method using associative rule mining and simulated annealing. In Section 5, we present theoretical bounds on the support and the number of rules in an optimal MAP solution and generalization bound of BRS models. In Section 6, we show the simulation studies to justify the inference methods. In Section 7, we report experimental results to show that BRS models are accurate and interpretable.

4.2 Related Work

The models we are studying have different names in different fields: "disjunctions of conjunctions" in marketing, "classification rules" in data mining, and "disjunctive normal forms" (rule set) in artificial intelligence. Learning logical models of this form has an extensive history. [84] showed that rule sets could be learned in polynomial time in the PAC (probably approximately correct) setting, and recent work has improved those bounds via polynomial threshold functions [43] and Fourier analysis [28]. Other work studies the hardness of learning rule set [27]. None of these theoretical approaches considered the practical aspects of building predictive models using sparse rule set. In parallel, the data-mining literature has developed approaches to building logical conjunctive models. Associative classification methods, e.g. [14, 16, 50, 55, 58, 75, 98] mine for frequent rules in the data and combine them in a heuristic way, where rules are ranked by an interestingness criteria and the top several rules are used.

Despite the efforts that theoretical communities have placed on learning rule set (e.g., [26, 43, 53]), the algorithms designed for inductive logic programming that essentially produce rule set [45,63], the associative classification algorithms [14,16,35,50,55,98], and rule induction methods (e.g. [18],) there has been little in the way of algorithms designed for applications where cognitive simplicity is first and foremost with some exceptions, like [38] which we discuss later. For instance, [98] reported that the average number of rules used by CMAR was 305 and CPAR had 244 rules on average, on 26 datasets from UCI ML Repository [51]; these are not cognitively simple models. Besides, all of the algorithms

discussed above use greedy approximations, which hurts accuracy and sparsity. For instance, RIPPER employs local greedy splitting, meaning that a mistake at the beginning is difficult to undo. Inductive logic programming starts with a collection of rules and locally combines them. The associative classification methods also follow separate-and-conquer or covering strategies. Unlike these methods, our methods aim to produce cognitive simple models, and do not use greedy approximations or similar heuristics. There are few recent techniques that do aim to fully learn rule set models [32, 38], which present integer programming approaches for solving the full problems, and also present relaxations for computational efficiency. These are very different from our work in that the method of [38] is not generative and also does not have the advantage of reduction to a smaller problem that we have.

4.3 Bayesian Rule Set

We work with standard classification data. The data set S consists of $\{\mathbf{x}_n, y_n\}_{n=1,\ldots N}$, where $y_n \in \{0, 1\}$ and $\{\mathbf{x}_n\}_{n=1,\ldots N}$ has N observations and J attributes. S^+ is the class of observations with positive labels, and the observations with negative labels are S^-. Let a represent a rule and we use $a(\cdot)$ to represent a boolean function

$$a(\cdot) : \mathcal{X} \rightarrow [0, 1] \tag{4.1}$$

$a(\mathbf{x})$ indicates if x satisfies rule a. We call the number of observations that satisfy rule a the *support* of a, denoted as

$$\text{supp}(a) = |\{i|a(X_i) = 1\}| \tag{4.2}$$

Let A denote a set of rules, we can define similar concepts following (4.1) to (4.2). We build a classifier from A,

$$A(\mathbf{x}) = \begin{cases} 1 & \exists a \in A, a(\mathbf{x}) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

As long as a data point satisfies at least one of the rules in A, it is classified as positive.

Figure 4-1 shows an example of a rule set. Each rule is a yellow patch that covers a particular area, and the rule applies to the area it covers. In Figure 4-1, the white oval in the middle indicates the positive class. Our goal is to find a set of rules A that covers mostly the positive class, but little of the negative class, while keeping A a small set of short rules.

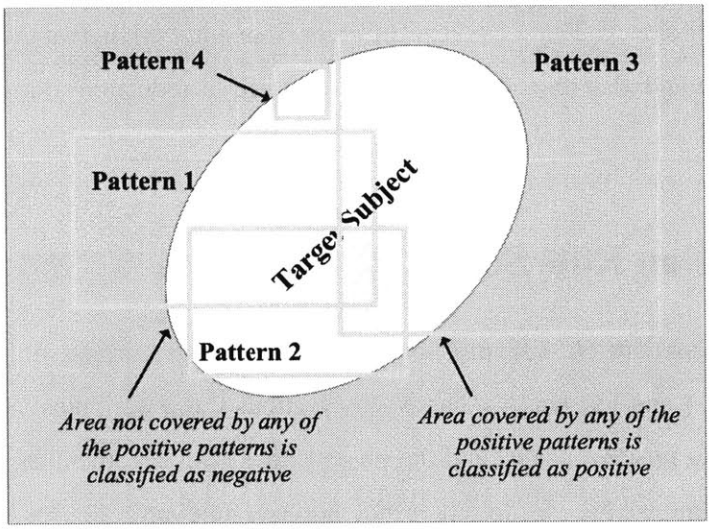


Figure 4-1: Illustration of Rule Set models for binary classification. The target subject represents a minor class.

We present a probabilistic model for selecting rules. Taking a Bayesian approach allows us to flexibly incorporate users' expectations on the "shape" of a rule set through the prior. The user can guide the model toward more domain-interpretable solutions by specifying a desired balance between the size and lengths of rules without committing to any particular value for these parameters. We propose two models for the prior, a Beta-Binomial prior and a Poisson prior, which characterize the interpretability from different perspectives. The likelihood ensures that the model explains the data well, and ensures good classification performance. We detail the priors and likelihood below.

4.3.1 Prior

We propose two models for the prior. In the BRS-BetaBinomial model, the maximum length L of rules is pre-determined by a user. Rules of the same length are placed into the same rule *pool*. The model uses L beta priors to control the probabilities of selecting rules from different pools. In a BRS-Poisson model, the "shape" of a rule set, which includes the number of rules and lengths of rules, is decided by drawing from Poisson distributions parameterized with user-defined values. Then the generative process fills it in with conditions by first randomly selecting attributes and then randomly selecting values corresponding to each attribute. We present the two prior models in detail.

Beta-Binomial prior

Notice that the rule set we want to find should include rules that describe only the positive class and also discriminate it from the negative class. Thus we need only rules from the positive data S^+ and not from S^-. We use $\mathcal{A}$ to represent a complete set of rules mined from S^+. We assume the interpretability of a rule is associated simply with the length of a rule (number of conditions in a rule). Therefore, we divide the rule space $\mathcal{A}$ into L pools indexed by the lengths of rules, L being the maximum length the user allow.

$$\mathcal{A} = \cup_{l=1}^{L} \mathcal{A}^{[l]} \tag{4.4}$$

rules are selected independently in each pool and we assume in $\mathcal{A}^{[l]}$ each rule has probability p_l to be selected, which we place a beta prior on. For $l \in \{1, ..., L\}$,

$$\text{Select a rule from } \mathcal{A}^{[l]} \sim \text{Bernoulli}(p_l) \tag{4.5}$$

$$p_l \sim \text{Beta}(\alpha_l, \beta_l). \tag{4.6}$$

Let $M^{[l]}$ notate the number of rules drawn from $\mathcal{A}^{[l]}$. Then the probability of selecting a BRS A is

$$p(A) = p(\{M^{[l]}\}_l; \{\alpha_l, \beta_l\}_l)$$

$$
\begin{aligned}
&= \prod_{l=1}^{L} p(M^{[l]}; \alpha_l, \beta_l) \\
&= \prod_{l=1}^{L} \int_{p_l} p_l^{M^{[l]}} (1-p_l)^{|\mathcal{A}^{[l]}|-M^{[l]}} \text{Beta}(p_l; \alpha_l, \beta_l) d(p_l) \\
&= \prod_{l=1}^{L} \theta_l h_l(M^{[l]}; \alpha_l, \beta_l, A^{[l]}) \qquad (4.7)
\end{aligned}
$$

where for future convenience we denote $\theta_l(\alpha_l, \beta_l, A^{[l]}) = \frac{\Gamma(\alpha_l+\beta_l)}{\Gamma(\alpha_l)\Gamma(\beta_l)\Gamma(|\mathcal{A}^{[l]}|+\alpha_l+\beta_l)}$, which is a constant once the prior parameters and rule space are given, and $h_l(M^{[l]}; \alpha_l, \beta_l, A^{[l]}) = \Gamma(M^{[l]}+\alpha_l)\Gamma(|\mathcal{A}^{[l]}| - M^{[l]} + \beta_l)$. We write $\theta_l(\alpha_l, \beta_l, \mathcal{A}^{[l]})$ as θ_l, and $h_l(M^{[l]}; \alpha_l, \beta_l, \mathcal{A}^{[l]})$ as $h_l(M^{[l]})$ for convenience, ommiting dependence on $\alpha_l, \beta_l, \mathcal{A}^{[l]}$ when appropriate.

The parameters $\{\alpha_l, \beta_l > 0\}_{l=1,...,L}$ on the priors control the expected number of rules of each length in the rule set. We usually choose $\alpha_l \ll \beta_l$ so that the model tends to choose a smaller set from each pool.

Poisson prior

We introduce a different prior for the BRS model. Let M denote the total number of rules in A and L_m denote the lengths of rules for $m \in \{1, ...M\}$. There can be at least 0 and at most $|\mathcal{A}|$ rules in a set, and for each rule, the length needs to be at least 1 and at most the number of all attributes J. We first draw M and L_m from truncated Poisson distributions to decide the "shape" of a rule set, then we fill in the rules with conditions. To generate a condition, we first randomly select the attributes then randomly select values corresponding to each attribute. We use $v_{m,k}$ to represent the attribute index for the k-th condition in the m-th rule, $v_{m,k} \in \{1, ...J\}$. $K_{v_{m,k}}$ is the total number of values for attribute $v_{m,k}$. The rule set is constructed as follows:

1: Draw the number of rules: $M \sim$ Truncated-Poisson(λ_M), $M \in \{0, ...|\mathcal{A}|\}$
2: **for** $m \in \{1, ...M\}$ **do**
3: Draw the number of conditions in m-th rule: $L_m \sim$ Truncated-Poisson(λ_L), $L_m \in \{1, ...J\}$
4: Randomly select L_m attributes from J attributes without replacement

5: **for** $k \in \{1, ...L_m\}$ **do**

6: Uniformly at random select a value from $K_{v_{m,k}}$ values corresponding to attribute $v_{m,k}$

7: **end for**

8: **end for**

We define $\theta_{\text{prior}} = \{\lambda_M, \lambda_L\}$, and normalization constant $\omega(\lambda_M, \lambda_L)$, thus the probability of generating a rule set A is

$$p(A) = \omega(\lambda_M, \lambda_L)\text{Poisson}(M; \lambda_M) \prod_m^M \text{Poisson}(L_m; \lambda_L) \frac{1}{\binom{J}{L_m}} \prod_k^{L_m} \frac{1}{K_{v_{m,k}}}. \quad (4.8)$$

4.3.2 Likelihood

Let $A(\mathbf{x}_n)$ denote the classification outcome for $\mathbf{x}_n$, and let y_n denote the observed outcome. Recall that $A(\mathbf{x}_n) = 1$ if $\mathbf{x}_n$ obeys any of the rules $a \in A$. We introduce likelihood parameter ρ_+ to govern the probability that $y_n = 1$ when $\mathbf{x}_n$ satisfies the rule set, and ρ_- as the probability that $y_n = 0$ when $\mathbf{x}_n$ does not satisfy the set.

The likelihood of data $S = \{\mathbf{x}_n, y_n\}_n$ given a rule set A and parameters ρ_+, ρ_- is thus:

$$p(S|A, \rho_+, \rho_-) = \prod_n \rho_+^{A(\mathbf{x}_n)y_n}(1-\rho_+)^{A(\mathbf{x}_n)(1-y_n)}\rho_-^{[1-A(\mathbf{x}_n)](1-y_n)}(1-\rho_-)^{[1-A(\mathbf{x}_n)]y_n}, \quad (4.9)$$

where the four components in formula (4.9) represent four classification outcomes: true positive, false positive, true negative, and false negative. We place beta priors over ρ_+ and ρ_-:

$$\rho_+ \sim \text{Beta}(\alpha_+, \beta_+), \quad (4.10)$$

$$\rho_- \sim \text{Beta}(\alpha_-, \beta_-). \quad (4.11)$$

Here, $\alpha_+, \beta_+, \alpha_-, \beta_-$ should be chosen such that $E[\rho_+]$ and $E[\rho_-]$ are close to 1, indicating the classification outcomes agree with the observed outcomes. Integrating out ρ_+ and ρ_-

from the likelihood in (4.9), we get

$$p(S|A;\alpha_+,\beta_+,\alpha_-,\beta_-) = \int_0^1 P(S|A,\rho_+,\rho_-)\text{Beta}(\alpha_+,\beta_+)\text{Beta}(\alpha_-,\beta_-)d\rho_+ d\rho_-$$
$$= C_{\text{B}} \cdot \frac{\Gamma\left(\sum_n A(\mathbf{x}_n)y_n + \alpha_+\right)\Gamma\left(\sum_n A(\mathbf{x}_n)(1-y_n) + \beta_+\right)}{\Gamma\left(\sum_n A(\mathbf{x}_n) + \alpha_+ + \beta_+\right)} \cdot$$
$$\frac{\Gamma\left(\sum_n (1-A(\mathbf{x}_n))\,y_n + \beta_-\right)\Gamma\left(\sum_n (1-A(\mathbf{x}_n))\,(1-y_n) + \alpha_-\right)}{\Gamma\left(\sum_n (1-A(\mathbf{x}_n)) + \alpha_- + \beta_-\right)}, \tag{4.12}$$

where $C_B = \frac{\Gamma(\alpha_+ + \beta_+)}{\Gamma(\alpha_+)\Gamma(\beta_+)}\frac{\Gamma(\alpha_- + \beta_-)}{\Gamma(\alpha_-)\Gamma(\beta_-)}$. We write the likelihood as $p(S|A)$, omitting dependence on parameters when appropriate.

The training data can be divided into four cases: true positives (TP = $\sum_n A(\mathbf{x}_n)y_n$), false positives (FP = $\sum_n A(\mathbf{x}_n)(1-y_n)$), true negatives (TN = $\sum_n (1-A(\mathbf{x}_n))\,(1-y_n)$) and false negatives (FN = $\sum_n (1-A(\mathbf{x}_n))\,y_n$). The above likelihood can be rewritten as:

$$p(S|A) = C_B \cdot \frac{\Gamma(\text{TP}+\alpha_+)\Gamma(\text{FP}+\beta_+)}{\Gamma(\text{TP}+\text{FP}+\alpha_+ + \beta_+)}\frac{\Gamma(\text{TN}+\alpha_-)\Gamma(\text{FN}+\beta_-)}{\Gamma(\text{TN}+\text{FN}+\alpha_- + \beta_-)}. \tag{4.13}$$

We notate

$$\text{Likelihood}(S) = \log p(S|A) \tag{4.14}$$
$$\text{Prior}(A) = \log p(A). \tag{4.15}$$

Solving for a MAP model is equivalent to maximizing

$$F(A;S,H) = \text{Likelihood}(S) + \text{Prior}(A) \tag{4.16}$$

where we write the objective as $F(A)$, omitting dependence on the data and hyper-parameters θ when appropriate.

4.3.3 Remarks on Prior Parameters

For the BRS-BetaBinomial model, parameters $\{\alpha_l, \beta_l\}_l$ govern the prior probability of selecting a rule set. To favor smaller models, we would choose α_l, β_l such that $\frac{\alpha_l}{\alpha_l+\beta_l}$ is close

to 0. In practice, we simply let $\alpha_l = 1$ for all l.

According to (4.10) and (4.11), $\alpha_+, \beta_+, \alpha_-, \beta_-$ govern the probability that a prediction is correct on training data, which determines the likelihood. To make sure the model achieves the maximum likelihood when all data points are classified correctly, i.e., TP $= |S^+|$, FP $= 0$, TN$= |S^-|$ and FN $= 0$, we choose $\alpha_+, \beta_+, \alpha_-, \beta_-$ such that $\frac{\alpha_+}{\alpha_+ + \beta_+} > 0.5$ and $\frac{\alpha_-}{\alpha_- + \beta_-} > 0.5$.

4.4 Approximate MAP Inference

We describe a procedure combining association rule mining and simulated annealing, for approximately solving for the maximum *a posteriori* (MAP) solution to a BRS model.

We first use association rule mining to construct a rule space, assuming all rules are drawn from this rule space. To reduce computation, we derive bounds on the minimum support of rules in a MAP model, which can safely remove a majority of rules from the rule space without affecting the output. This reduces the original problem to a much more manageable size in practice. To deal with large rule space, we use a heuristic to rank the rules which proved to be effective in selecting good rules. Lastly, we apply simulated annealing based stochastic search algorithms to search within reduced rule space for a MAP model.

We will first detail the search space and discuss theoretical bounds , and then present the simulated annealing search algorithm with two proposing strategies .

4.4.1 Reduce Search Space

Inference in the BRS model is challenging because finding the best model involves a search over exponentially many possible sets of rules: since each rule is a conjunction of conditions, the number of rules increases exponentially with the number of conditions, and the number of sets of rules increases exponentially with the number of rules. This is the reason learning a rule set has always been a difficult problem.

For a BRS model, however, the problem becomes easier. Since the Bayesian prior penalizes large models, a BRS model tends to contain a small number of rules. Therefore,

given the positive class to "cover", each rule in a MAP model needs to satisfy a non-negligible support. Now we detail the first strategy of reducing the rule space

Minimum support filtering

Let's for now assume m_l is the upper bound on the number of rules of length l in a MAP model (which we will obtain later), and we prove that rules with a support lower than a certain threshold do not exist in a MAP model.

Theorem 2 *Take a BRS-BetaBinomial model with parameters*

$$H = \left\{L, \alpha_+, \beta_+, \alpha_-, \beta_-, \{\mathcal{A}^{[l]}, \alpha_l, \beta_l\}_{l=1,..L}\right\},$$

where $L, \alpha_+, \beta_+, \alpha_-, \beta_-, \{\alpha_l, \beta_l\}_l \in \mathbb{N}^+$. *When* $\frac{|S^+|+\alpha_+-1}{|S^+|+\alpha_++\beta_+-1}\frac{|S^-|+\alpha_-+\beta_-}{\beta_-} \geq 1$, $\alpha_l < \beta_l$ *for* $l = 1, 2...L$ *and* $A^* \in \arg\min_A F(A; S, H)$, *then*

$$\forall a \in A^*, supp(a) \geq \frac{\log \max_l \left(\frac{|\mathcal{A}^{[l]}|-m_l+\beta_l}{m_l-1+\alpha_l}\right)}{\log \frac{|S^+|+\alpha_+-1}{|S^+|+\alpha_++\beta_+-1}\frac{|S^-|+\alpha_-+\beta_-}{\beta_-}}.$$

Proof 2 *For a rule set A, we will show that if any rule z has support* $supp(z) < C$ *on data S, then* $A \notin \arg\max_{A' \in \Lambda_S} F(A')$. *Assume rule z has support less than C and define* $A_{\setminus z}$ *which has the z-th rule removed from A:*

$$A_{\setminus z} = \{a | a \in A, a \neq z\}.$$

Our goal is to find conditions on $sup(z)$ *such that*

$$F(A) \geq F(A_{\setminus z}) \tag{4.17}$$

Assume A consists of M rules, among which M_l *come from pool* $\mathcal{A}^{[l]}$ *of rules with length* l, $l \in \{1, ...L\}$, *and the z-th rule has length* l' *so it is removed from* $\mathcal{A}^{[l']}$. *Define TP, FP, TN and FN to be the number of true positives, false positives, true negatives and false negatives in S given A. We now compute the likelihood for model* $A_{\setminus z}$. *The most extreme*

case is when rule a_z is an accurate rule that applies only to real positive data points and those data points satisfy only a_z. Therefore once removing it, the number of true positives decreases by supp(z) and the number of false negatives increases by supp(z).

Step 1: Relate $P(S|A_{\setminus z})$ ***to*** $P(S|A)$.

$$\begin{aligned} P(S|A_{\setminus z}) \geq & C_B \cdot \frac{\Gamma(TP+\alpha_+ - supp(z))\Gamma(FP+\beta_+)}{\Gamma(TP+FP+\alpha_+ +\beta_+ - supp(z))} \frac{\Gamma(\alpha_- +\beta_-)}{\Gamma(\alpha_-)\Gamma(\beta_-)} \frac{\Gamma(TN+\alpha_-)\Gamma(FN+\beta_- + supp(z))}{\Gamma(TN+FN+\alpha_- +\beta_- + supp(z))} \\ = & P(S|A) \cdot g_3(TP, FP, TN, FN, supp(z)), \end{aligned} \tag{4.18}$$

where

$$\begin{aligned} g_3(TP, FP, TN, FN, supp(z)) = & \frac{\Gamma(TP+\alpha_+ - supp(z))}{\Gamma(TP+\alpha_+)} \frac{\Gamma(TP+FP+\alpha_+ +\beta_+)}{\Gamma(TP+FP+\alpha_+ +\beta_+ - supp(z))} \cdot \\ & \frac{\Gamma(FN+\beta_- + supp(z))}{\Gamma(FN+\beta_-)} \frac{\Gamma(TN+FN+\alpha_- +\beta_-)}{\Gamma(TN+FN+\alpha_- +\beta_- + supp(z))}. \end{aligned} \tag{4.19}$$

Now we break down $g_3(TP, FP, TN, FN, supp(z))$ to find a lower bound for it. The first two terms in (4.19) become

$$\begin{aligned} & \frac{\Gamma(TP+\alpha_+ - supp(z))}{\Gamma(TP+\alpha_+)} \frac{\Gamma(TP+FP+\alpha_+ +\beta_+)}{\Gamma(TP+FP+\alpha_+ +\beta_+ - supp(z))} \\ = & \frac{(TP+FP+\alpha_+ +\beta_+ - supp(z)) \ldots (TP+FP+\alpha_+ +\beta_+ - 1)}{(TP+\alpha_+ - supp(z)) \ldots (TP+\alpha_+ - 1)} \\ \geq & \left(\frac{TP+FP+\alpha_+ +\beta_+ - 1}{TP+\alpha_+ - 1} \right)^{supp(z)} \\ \geq & \left(\frac{|S^+| + \alpha_+ +\beta_+ - 1}{|S^+| + \alpha_+ - 1} \right)^{supp(z)}. \end{aligned} \tag{4.20}$$

Equality holds in (4.20) when $TP = |S^+|, FP = 0$. The last two terms in (4.19) become

$$\begin{aligned} & \frac{\Gamma(FN+\beta_- + supp(z))}{\Gamma(FN+\beta_-)} \frac{\Gamma(TN+FN+\alpha_- +\beta_-)}{\Gamma(TN+FN+\alpha_- +\beta_- + supp(z))} \\ = & \frac{(FN+\beta_-) \ldots (FN+\beta_- + supp(z) - 1)}{(TN+FN+\alpha_- +\beta_-) \ldots (TN+FN+\alpha_- +\beta_- + supp(z) - 1)} \\ \geq & \left(\frac{FN+\beta_-}{FN+TN+\alpha_- +\beta_-} \right)^{supp(z)} \end{aligned}$$

$$\geq \left(\frac{\beta_-}{|S^-| + \alpha_- + \beta_-}\right)^{supp(z)}. \tag{4.21}$$

Equality in (4.21) holds when $TN = |S^-|, FN = 0$. *Combining (4.19), (4.20) and (4.21), we obtain a lower bound for* $g_3(TP, FP, TN, FN; \alpha_+, \beta_+, \alpha_-, \beta_-, supp(z))$ *as*

$$g_3(TP, FP, TN, FN, \alpha_+, \beta_+, \alpha_-, \beta_-, supp(z)) \geq \left(\frac{|S^+| + \alpha_+ + \beta_+ - 1}{|S^+| + \alpha_+ - 1} \frac{\beta_-}{|S^-| + \alpha_- + \beta_-}\right)^{supp(z)}. \tag{4.22}$$

Following (4.18),

$$P(S|A_{\setminus z}) \geq \left(\frac{|S^+| + \alpha_+ + \beta_+ - 1}{|S^+| + \alpha_+ - 1} \frac{\beta_-}{|S^-| + \alpha_- + \beta_-}\right)^{supp(z)} \cdot P(S|A). \tag{4.23}$$

Step 2: Relate $p(A_{\setminus z})$ ***to*** $p(A)$. *Since* $A_{\setminus z}$ *consists of the same rules as* A *except missing one rule from* $\mathcal{A}^{[l']}$, *we multiply* $p(A_{\setminus z})$ *with 1 in disguise to relate it to* $p(A)$.

$$\begin{aligned}
p(A_{\setminus z}) &= \theta_{l'} h_{l'}(M_{l'} - 1) \prod_{l \neq l'}^{L} \theta_l h_l(M_l) \\
&= p(A) \frac{h_{l'}(M_{l'} - 1)}{h_{l'}(M_{l'})} \\
&= p(A) \frac{\Gamma(M_{l'} - 1 + \alpha_{l'}) \Gamma(|\mathcal{A}^{[l']}| - M_{l'} + 1 + \beta_{l'})}{\Gamma(M_{l'} + \alpha_{l'}) \Gamma(|\mathcal{A}^{[l']}| - M_{l'} + \beta_{l'})} \\
&= \frac{|\mathcal{A}^{[l']}| - M_{l'} + \beta_{l'}}{M_{l'} - 1 + \alpha_{l'}} \cdot p(A).
\end{aligned}$$

$\frac{|\mathcal{A}^{[l']}| - M_{l'} + \beta_{l'}}{M_{l'} - 1 + \alpha_{l'}}$ *decreases monotonically as* $M_{l'}$ *increases, therefore it is lower bounded at the upper bound on* $M_{l'}$, $m_{l'}$. *so*

$$\frac{|\mathcal{A}^{[l']}| - M_{l'} + \beta_{l'}}{M_{l'} - 1 + \alpha_{l'}} \geq \frac{|\mathcal{A}^{[l']}| - m_{l'} + \beta_{l'}}{m_{l'} - 1 + \alpha_{l'}} \text{ for } l' \in \{1, ...L\}.$$

Therefore

$$p(A_{\setminus z}) \geq \max_l \left(\frac{|\mathcal{A}^{[l]}| - m_l + \beta_l}{m_l - 1 + \alpha_l}\right) p(A). \tag{4.24}$$

Step 3: Combine Step 1 and Step 2. *Combining (4.23) and (4.24), the joint probability of*

S and $A_{\setminus z}$ is bounded by

$$P(S, A_{\setminus z}) = p(A_{\setminus z})P(S|A_{\setminus z})$$
$$\geq \max_l \left(\frac{|\mathcal{A}^{[l]}| - m_l + \beta_l}{m_l - 1 + \alpha_l} \right) \left(\frac{|S^+| + \alpha_+ + \beta_+ - 1}{|S^+| + \alpha_+ - 1} \frac{\beta_-}{|S^-| + \alpha_- + \beta_-} \right)^{supp(z)} \cdot P(S, A).$$

In order to get $P(S, A_{\setminus z}) \geq P(S, A)$, we need

$$\max_l \left(\frac{|\mathcal{A}^{[l]}| - m_l + \beta_l}{m_l - 1 + \alpha_l} \right) \left(\frac{|S^+| + \alpha_+ + \beta_+ - 1}{|S^+| + \alpha_+ - 1} \frac{\beta_-}{|S^-| + \alpha_- + \beta_-} \right)^{supp(z)} \geq 1.$$

We have $\frac{|S^+|+\alpha_+ +\beta_+ -1}{|S^+|+\alpha_+ -1} \frac{\beta_-}{|S^-|+\alpha_- +\beta_-} \leq 1$ from the assumption in the theorem's statement, thus

$$supp(z) \leq \frac{\log \max_l \left(\frac{|\mathcal{A}^{[l]}| - m_l + \beta_l}{m_l - 1 + \alpha_l} \right)}{\log \frac{|S^+|+\alpha_+ -1}{|S^+|+\alpha_+ +\beta_+ -1} \frac{|S^-|+\alpha_- +\beta_-}{\beta_-}}.$$

We proof a similar bound for BRS-Poisson models. We start with the following lemma that we will need later.

Lemma 1 *Define a function $g(l; \lambda, J) = \left(\frac{\lambda}{2}\right)^l \Gamma(J - l + 1), \lambda, J \in \mathbb{N}^+$. If $1 \leq l \leq J$, $g(l; \lambda, J) \leq g_{max}(\lambda, J)$ where $g_{max}(\lambda, J) = \max\left\{ \left(\frac{\lambda}{2}\right) \Gamma(J), \left(\frac{\lambda}{2}\right)^J \right\}$.*

Proof 3 *In order to bound $g(l; \lambda, J)$, we will show that $g(l; \lambda, J)$ is convex, which means its maximum value occurs at the endpoints of the interval we are considering. The second derivative of $g(l; \lambda, J)$ respect to l is*

$$g''(l; \lambda, J) = g(l; \lambda, J) \left[\left(\ln \frac{\lambda}{2} + \gamma - \sum_{k=1}^{\infty} \frac{1}{k} - \frac{1}{k + J - L_m} \right)^2 + \sum_{k=1}^{\infty} \frac{1}{(k + J - l)^2} \right] > 0 \quad (4.25)$$

since at least one of the terms $\frac{1}{(k+J-l)^2} > 0$. Thus $g(l; \lambda, J)$ is strictly convex. Therefore

the maximum of $g(l;\lambda,J)$ is achieved at the boundary of L_m, namely 1 or J. So we have

$$\begin{aligned} g(l;\lambda,J) &\leq \max\{g(1;\lambda,J), g(J;\lambda,J)\} \\ &= \max\left\{\left(\frac{\lambda}{2}\right)\Gamma(J), \left(\frac{\lambda}{2}\right)^J\right\} \\ &= g_{max}(\lambda,J). \end{aligned} \tag{4.26}$$

Then we claim

Theorem 3 *Take a BRS-Poisson model with parameters*

$$H = \{\alpha_+, \beta_+\alpha_-, \beta_-, \lambda_M, \lambda_L\},$$

where α_+, β_+, α_-,β_-, $\lambda_M, \lambda_L \in \mathbb{N}^+$. When $\frac{|S^+|+\alpha_++\beta_+-1}{|S^+|+\alpha_+-1}\frac{\beta_-}{|S^-|+\alpha_-+\beta_-} \leq 1$ and $A^* \in \arg\min_A F(A;S,H)$, then

$$\forall a \in A^*, supp(a) \geq \frac{\log\left(\frac{\Gamma(J+1)}{\lambda_M e^{-\lambda_L}\max\left\{\left(\frac{\lambda}{2}\right)\Gamma(J),\left(\frac{\lambda}{2}\right)^J\right\}}\right)}{\log\frac{|S^+|+\alpha_+-1}{|S^+|+\alpha_++\beta_+-1}\frac{|S^-|+\alpha_-+\beta_-}{\beta_-}}.$$

Proof 4 *Similar to the proof for Theorem 2, we will show that for a rule set A, if any rule a_z has support $supp(z) < C$ on data S, then $A \notin \arg\max_{A'\in\Lambda_S} F(A')$. Assume rule a_z has support less than C and $A_{\setminus z}$ has the k-th rule removed from A. Assume A consists of M rules, and the z-th rule has length l'.*

Step 1 is the same as in the proof for Theorem 2, we relate $p(A_{\setminus z})$ with $p(A)$. We multiply $p(A_{\setminus z})$ with 1 in disguise to relate it to $p(A)$:

$$\begin{aligned} p(A_{\setminus z}) &= \omega(\lambda_M,\lambda_L)Poisson(M-1;\lambda_M)\prod_{m\neq z}^{M} Poisson(L_m;\lambda_L)\frac{1}{\binom{J}{L_m}}\prod_{k}^{L_m}\frac{1}{K_{v_{m,k}}} \\ &= \frac{\omega(\lambda_M,\lambda_L)Poisson(M-1;\lambda_M)\prod_{m\neq z}^{M} Poisson(L_m;\lambda_L)\frac{1}{\binom{J}{L_m}}\prod_{k}^{L_m}\frac{1}{K_{v_{m,k}}}}{\omega(\lambda_M,\lambda_L)Poisson(M;\lambda_M)\prod_{m}^{M} Poisson(L_m;\lambda_L)\frac{1}{\binom{J}{L_m}}\prod_{k}^{L_m}\frac{1}{K_{v_{m,k}}}}p(A) \\ &= \frac{M\Gamma(J+1)}{\lambda_M e^{-\lambda_L}\lambda_L^{L_z}\Gamma(J-L_z+1)\prod_k^{L_z}\frac{1}{K_{v_{z,k}}}}p(A) \end{aligned}$$

$$\geq \frac{M\Gamma(J+1)}{\lambda_M e^{-\lambda_L}\left(\frac{\lambda_L}{2}\right)^{L_z}\Gamma(J-L_z+1)}p(A) \tag{4.27}$$

$$= \frac{M\Gamma(J+1)}{\lambda_M e^{-\lambda_L} g(L_z;\lambda_L,J)}p(A) \tag{4.28}$$

$$\geq \frac{\Gamma(J+1)}{\lambda_M e^{-\lambda_L} g_{max}(\lambda_L,J)}p(A), \tag{4.29}$$

where (4.27) follows that $K_{v_{m,k}} \geq 2$ since all attributes have at least two values, (4.28) follows the definition of $g(l;\lambda,J)$ in Lemma 1, and (4.29) uses the upper bound in Lemma 1 and $M \geq 1$. Then combining (4.23) with (4.29), the joint probability of S and $A_{\setminus z}$ is lower bounded by

$$p(S,A_{\setminus z}) = p(A_{\setminus z})p(S|A_{\setminus z})$$
$$\geq \frac{\Gamma(J+1)}{\lambda_M e^{-\lambda_L} g_{max}(\lambda_L,J)}\left(\frac{|S^+|+\alpha_+ +\beta_+ -1}{|S^+|+\alpha_+ -1}\frac{\beta_-}{|S^-|+\alpha_- +\beta_-}\right)^{supp(z)}\cdot p(S,A).$$

In order to get $p(S,A_{\setminus z}) \geq p(S,A)$, we need

$$\frac{\Gamma(J+1)}{\lambda_M e^{-\lambda_L} g_{max}(\lambda_L,J)}\left(\frac{|S^+|+\alpha_+ +\beta_+ -1}{|S^+|+\alpha_+ -1}\frac{\beta_-}{|S^-|+\alpha_- +\beta_-}\right)^{supp(z)} \geq 1$$

We have $\frac{|S^+|+\alpha_+ +\beta_+ -1}{|S^+|+\alpha_+ -1}\frac{\beta_-}{|S^-|+\alpha_- +\beta_-} \leq 1$ and $g_{max}(\lambda_L,J) = \max\left\{\left(\frac{\lambda}{2}\right)\Gamma(J), \left(\frac{\lambda}{2}\right)^J\right\}$, thus

$$supp(z) \leq \frac{\log\left(\frac{\Gamma(J+1)}{\lambda_M e^{-\lambda_L}\max\left\{\left(\frac{\lambda}{2}\right)\Gamma(J),\left(\frac{\lambda}{2}\right)^J\right\}}\right)}{\log\frac{|S^+|+\alpha_+ -1}{|S^+|+\alpha_+ +\beta_+ -1}\frac{|S^-|+\alpha_- +\beta_-}{\beta_-}}. \tag{4.30}$$

The theorems state that the quantity $\arg\max_{A\in\Lambda} F(A)$ on the left (which is computationally impossible to compute in practice) has the same set of minimizers as the quantity $\arg\max_{A\in\Lambda^C} F(A)$ on the right (which is what we would compute in practice). It states that we need only to mine rules of support of at least C. If we solve for a maximizer of F on these mined rules, it is a global maximizer of F across all rule sets, since there are now weak conditions under which the "approximation" of using pre-mined rules is not an approximation at all. These bounds yield a major computational benefit, as it tells us that we need only mine and optimize over rules of a certain minimum support in order to find the MAP

solution. This eliminates an enormous number of rules and decreases the search space substantially. The stronger the prior, the more tractable the computation for this model.

Given the minimum support C computed from the model, we pre-mine rules to filter out those with small support. To make the rule mining process more tractable and also for interpretability purposes, we specify the maximum length of rules we allow L. In reality, long rules can will almost always be discarded since they haver lower probabilities of satisfying a minimum support than shorter rules.

When mining rules, we consider both positive associations (e.g., x_j='blue') and negative associations (x_j='not green') as conditions. The importance of negative conditions is stressed, for instance, by [11, 81, 97]. We then mine for frequent rules within the set of positive observations S^+. To do this, we use an established frequent rule mining algorithm FP-growth algorithm [9], which can in practice be replaced with any desired frequent rule-mining method.

A minimum support helps reduce the rule space. However, for large data sets, there are still more than thousands of rules generated and the search space remains too large for simulated annealing to explore. Therefore, we propose the second screening method to further reduce the rule space.

Individual performance filtering

We note that any reasonably accurate sparse classifier should contain largely accurate rules. Rather than considering all rules, we use rules that have individually good performance.

Therefore, we wish to use a second criterion to screen for the most potentially useful M rules. M is defined by the user. In practice, we usually choose M to be a few thousand. We first filter out rules on the lower right plane of ROC space, i.e., their false positive rate is greater than true positive rate. Then we use *information gain* to screen rules, similarly to other works [14, 16]. For a rule a, the information gain is

$$\text{InfoGain}(S|a) = H(S) - H(S|a), \tag{4.31}$$

where $H(S)$ is the entropy of the data and $H(S|a)$ is the conditional entropy of data that

split on rule a. Given a dataset S, entropy $H(S)$ is constant; therefore our screening technique chooses the M rules that have the smallest $H(S|a)$.

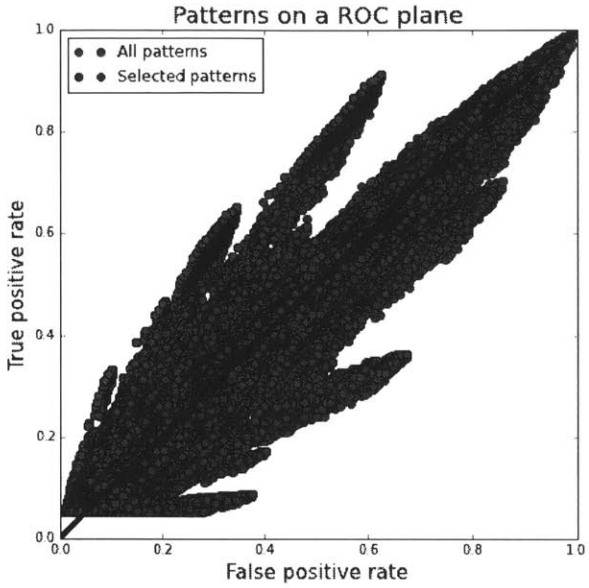


Figure 4-2: All rules and rules with highest information gains on a ROC plane

We illustrate the effect of screening on one of our advertisement data sets. We mined all rules with minimum support 5% and maximum length 3. For each rule, we computed its true positive rate and false positive rate on the training data, and plotted it as a dot in Figure 4-2. The top 5000 rules with highest information gains are colored in red, and the rest are in blue. As shown in the figure, information gain indeed selected good rules as they are closer to the upper left corner in ROC space. For many applications where there are tens of thousands of rules after the minimum support filtering step, this secondary screening technique is not necessary since the search algorithm is able to handle them, as shown in simulations studies, so we can simply use the entire set of pre-mined rules.

4.4.2 Simulated Annealing

Given pre-mined rules $\mathcal{A}$, the next step is to search within $\mathcal{A}$ for the optimal A^* via simulated annealing. Our simulated annealing steps are similar to the Gibbs sampling steps used by [48, 88] for rule-list models. Given an objective function $F(A) = \log P(S, A)$ over dis-

crete search space of different sets of rules and a temperature schedule function over time steps till the maximum allowed steps N_{iter}, $T(t) = T_0^{1-\frac{t}{N_{\text{iter}}}}$, simulated annealing generates a discrete time, discrete state Markov Chain where at time t, given the current state A^t, the next state A^{t+1} is chosen from the neighbors of A^t. "Neighboring" solutions are rule sets that are "one-rule-different" from the current model and can be generated by adding or removing a rule from A^t. The proposed new model is then accepted with probability $\min\left\{1, \exp\left(\frac{F(A^{t+1})-F(A^t)}{T(t)}\right)\right\}$.

In proposing a neighbor solution, we introduce two different strategies, one with random search, referred to as Stochastic Random Search (SRS), and the other with more strategic search, referred to as Stochastic Smart Search (SSS). Both methods can converge to the MAP solution with high probability and within a relatively short time. We show that the smarter strategies applied by SSS improves the efficiency.

We discuss the two algorithms in detail as below.

4.4.3 Stochastic Random Search

At time 0, SRS starts with a set of rules randomly selected from $\mathcal{A}$, denoted as A^0. Then at each iteration, the algorithm proposes a neighbor solution by choosing from the following actions with equal probability.

- $\text{ADD}_{\text{random}}$: Choose one rule z from $\mathcal{A} \backslash A^t$ uniformly at random and add z to A^t. $A^{t+1} \leftarrow A^t \cup z$.
- $\text{CUT}_{\text{random}}$: Choose one rule z at random from A^t, and remove it. $A^{t+1} \leftarrow A^t \backslash z$.

The algorithm is displayed below.

Note that this approach optimizes over the full set of rule sets from itemsets, and does not rely on greedy splitting. Even Bayesian tree methods that aim to traverse a wider search space use greedy splitting and local solutions (e.g. [17]).

This algorithm aims to fully explore the solution space and chooses actions and rules randomly without learning about previous models. We now introduce a second algorithm that applies smarter strategies that finds the MAP model much more quickly.

Algorithm 2 Stochastic Random Search

procedure SRS(*maxSteps*)
  $A^0 \leftarrow$ a randomly generated rule set,
  $step \leftarrow 0$,
  while *step<maxSteps* **do**
    $step \leftarrow step + 1$
    $A^{t+1} \leftarrow \begin{cases} \text{ADD}_{\text{random}}(A^t), & \text{with probability } 0.5 \\ \text{CUT}_{\text{random}}(A^t), & \text{with probability } 0.5 \end{cases}$
    $A_{\max} \leftarrow \arg\max(F(A_{\max}), F(A^{t+1}))$
    $\alpha = \min\left\{1, \exp\left(\frac{F(A^{t+1};H)-F(A^t;H)}{T(t)}\right)\right\}$
    $A^{t+1} \leftarrow \begin{cases} A^{t+1}, & \text{with probability } \alpha \\ A^t, & \text{with probability } 1-\alpha \end{cases}$
  return $A_{\max}$
end procedure

4.4.4 Stochastic Smart Search

The second algorithm, called Stochastic Smart Search, applies smarter strategies to improve the search efficiency. Instead of randomly choosing between adding and cutting, the algorithm learns from previous models and chooses an action that aims to do better than the current model. It then evaluates the objective for all neighbor solutions given an action, and chooses rules based on these potential outcomes. In this step, instead of simply choosing the best neighbor, SSS trades off between exploration and exploitation by incorporating randomness in selection. SSS chooses between the best and a random neighbor with certain probability. To further assist a smart proposing, we derive bounds on the size of MAP models, to inform the algorithm where the sampling chain needs to go in order to find the solution more quickly.

Before we continue, we introduce a few notations which we will be using repeatedly in the following sections. Given data S, the maximum likelihood is achieved when all data are classified correctly (we assume that appropriate hyper-parameters are chosen to make sure this holds), i.e. TP $= |S^+|$, FP $= 0$, TN $= |S^-|$, and FN $= 0$, giving

$$\mathcal{L}_* := \max_A p(S|A) = C_B \frac{\Gamma(|S^+| + \alpha_+)\Gamma(\beta_+)}{\Gamma(|S^+| + \alpha_+ + \beta_+)} \frac{\Gamma(|S^-| + \alpha_-)\Gamma(\beta_-)}{\Gamma(|S^-| + \alpha_- + \beta_-)}. \tag{4.32}$$

Now we compute the likelihood for the empty set BRS model. The empty set classifies all data points as negative, so TP $= 0$, FP $= 0$, FN $= |S^+|$, TN $= |S^-|$. The likelihood is

$$\mathcal{L}_\emptyset := p(S|\emptyset) = C_B \frac{\Gamma(\alpha_+)\Gamma(\beta_+)}{\Gamma(\alpha_+ + \beta_+)} \frac{\Gamma(|S^-| + \alpha_-)\Gamma(|S^+| + \beta_-)}{\Gamma(|S| + \alpha_- + \beta_-)}. \tag{4.33}$$

We also compute the prior probability of an empty set in a BRS-BetaBinomial model and a BRS-Poisson model.

$$\text{Prior}_\text{B}(\emptyset) = \sum_{l=1}^{L} \log\left(\theta_l h(0)\right) \tag{4.34}$$

$$\text{Prior}_\text{P}(\emptyset) = \log p(\emptyset) = \log \omega(\lambda_M, \lambda_L) e^{-\lambda_M}. \tag{4.35}$$

Now we detail the strategies and algorithm as below.

Branch-and-Bounding

We showed in Section 4.4.1 that the inclusion of the prior (which was designed to help with interpretability) provably assists with computation by reducing the rule space. In this part, we continue to take advantage of the Bayesian prior and derive a bounding strategy that guides the sampling chain towards MAP models.

We show that depending on the prior parameters and on the size of the data, we have deterministic upper bounds on the sizes of MAP BRS models. Knowing the range of the true size helps the search algorithm to find the MAP models more quickly, as we could decrease the probability of adding rules as the size of the current model is beyond the upper bound. Furthermore, this upper bound is updated as the search goes and becomes tighter as better models are discovered.

Let us start with BRS-BetaBinomial models. For BRS-BetaBinomial models, rules are selected based on their lengths, so we get separate bounds for rules of different lengths, and we notate the upper bounds at iteration t as $\{m_l(t)\}_{l=1,\ldots,L}$, where $m_l(t)$ represents the upper bound for the number of rules of length l. Let v_t^* denote the best solution found till iteration t, i.e.,

$$v_t^* = max_{\tau \leq t} F(A^\tau).$$

We claim,

Theorem 4 *Take a BRS-BetaBinomial model with parameters*

$$H = \left\{L, \alpha_+, \beta_+, \alpha_-, \beta_-, \{\mathcal{A}^{[l]}, \alpha_l, \beta_l\}_{l=1,..L}\right\},$$

where $L, \alpha_+, \beta_+, \alpha_-, \beta_-, \{\alpha_l, \beta_l\}_{l=1,...L} \in \mathbb{N}^+$ *and* $\alpha_l < \beta_l$. *Define* $A^* \in \arg\max_A F(A)$, *and* $M^* := |A^*|$. *Let* $v_t^* = max_{\tau \leq t} F(A^\tau)$, *and* $m_l(t)$ *indicate the upper bound on the number of rules of length* l *computed at time* t. *We have:*

$$M^* \leq \sum_{l=1}^{L} m_l(t),$$

where

$$m_l(t) = \frac{\log \mathcal{L}_* + \textit{Prior}_B(\emptyset) - v_t^*}{\log \frac{|\mathcal{A}^{[l]}|+\beta_l-1}{\alpha_l+m_l(t-1)-1}}.$$

Proof 5 *Since* $A^* \in \arg\max_A F(A)$, $F(A^*) \geq v_t^*$, *i.e.,*

$$\textit{Likelihood}(S|A^*) + \textit{Prior}(A^*) \leq v_t^* \tag{4.36}$$

We first upper bound the left-hand-side. Let M_l^* *denote the number of rules of length* l *in* A^*. *The prior probability of selecting* A^* *from* $\mathcal{A}$ *is*

$$p(A^*) = \prod_{l-1}^{L} \theta_l h_l(M_l^*) = p(\emptyset) \prod_{l=1}^{L} \frac{h_l(M_l^*)}{h(0)}$$

We observe that when for $0 \leq M_l^* \leq |A^{[l]}|$, $h_l''(M_l^*) \geq 0$, *therefore* $h_l(M_l^*)$ *is convex. Since* $h_l(|A^{[l]}|) \leq h_l(0)$ *and , we have* $h_l(M_l^*) \leq h_l(0)$.

Now we upper bound $h_l(M_l^*)$ *with* $h_l(0)$ *for each* $l \in \{1, ..., L\}$ *except* l', *giving*

$$\begin{aligned} p(A^*) &\leq p(\emptyset) \frac{h_{l'}(M_{l'}^*)}{h_{l'}(0)} \\ &= p(\emptyset) \frac{\Gamma(M_{l'}^* + \alpha + l')\Gamma(|\mathcal{A}^{[l']}| - M_{l'}^* + \beta_{l'})}{\Gamma(\alpha_{l'})\Gamma(|\mathcal{A}^{[l']}| + \beta_{l'})} \end{aligned}$$

$$= p(\emptyset)\frac{\alpha_{l'} \cdot (\alpha_{l'}+1)\cdots(M^*_{l'}+\alpha_{l'}-1)}{(|\mathcal{A}^{[l']}| - M^*_{l'}+\beta_{l'})\cdots(|\mathcal{A}^{[l']}|+\beta_{l'}-1)}$$
$$\leq p(\emptyset)\left(\frac{M^*_{l'}+\alpha_{l'}-1}{|\mathcal{A}^{[l']}|+\beta_{l'}-1}\right)^{M^*_{l'}} \tag{4.37}$$

The likelihood is upper bounded by

$$p(S|A^*) \leq \mathcal{L}_* \tag{4.38}$$

Plug (4.37) and (4.38) in (4.36) we get

$$\log \mathcal{L}_* + \textit{Prior}(\emptyset) + M^*_{l'} \log\left(\frac{M^*_{l'}+\alpha_{l'}-1}{|\mathcal{A}^{[l']}|+\beta_{l'}-1}\right) \geq v^*_t, \tag{4.39}$$

which gives

$$\begin{aligned} M^*_{l'} &\leq \frac{\log \mathcal{L}_* + \textit{Prior}(\emptyset) - v^*_t}{\log\left(\frac{|\mathcal{A}^{[l']}|+\beta_{l'}-1}{M^*_{l'}+\alpha_{l'}-1}\right)} \\ &\leq \frac{\log \mathcal{L}_* + \textit{Prior}(\emptyset) - v^*_t}{\log\left(\frac{|\mathcal{A}^{[l']}|+\beta_{l'}-1}{m_{l'}(t-1)+\alpha_{l'}-1}\right)} \\ &:= m_{l'}(t) \textit{ for all } l' \in \{1,...,L\}. \end{aligned} \tag{4.40}$$

(4.40) follows because $M^*_{l'} \leq m_{l'}(t-1)$. *Thus*

$$M^* \leq \sum_{l=1}^{L} m_l(t) \tag{4.41}$$

$\log \mathcal{L}_*$ and $\text{Prior}_\text{B}(\emptyset)$ represent the best possible likelihood and prior probability, respectively. The sum of them upper bounds the maximum objective value $F(A^*)$. The numerator is an upper bound for the room for improvement from current solution v^*_t. The bigger the difference, the larger the $m_l(t)$. The rate of convergence depends on the previous bound $m_l(t-1)$. As $m_l(t)$ becomes smaller, the curve of $m_l(t)$ gradually becomes flat. Additionally, when α_l is set to be small and β_l is set to be large, the bound is smaller so we are guaranteed to choose a smaller number of rules overall. This is consistent with users'

expectation when they set $\frac{\alpha_l}{\alpha_l+\beta_l}$ to be small, as explained in Section 4.3.3.

The above bound depends on previous solutions. At the beginning of the search where there is no v_t^* or $m_l(t-1)$, we obtain an upper bound using an empty set as a solution, i.e. $v_0^* = \log \mathcal{L}_\emptyset + \text{Prior}_\text{B}(\emptyset)$, and replacing $m_l(t-1)$ with the size of the rule set $|A^{[l]}|$, giving

Corollary 1 *Take a BRS-BetaBinomial model with parameters*

$$H = \left\{L, \alpha_+, \beta_+, \alpha_-, \beta_-, \{\mathcal{A}^{[l]}, \alpha_l, \beta_l\}_{l=1,..L}\right\},$$

where $L, \alpha_+, \beta_+, \alpha_-, \beta_-, \{\alpha_l, \beta_l\}_{l=1,...L} \in \mathbb{N}^+$. *Define* $A^* \in \arg\max_A F(A)$, *and* $M^* := |A^*|$. *Whenever* $\alpha_l < \beta_l$, *we have:*

$$M^* \leq \sum_l^L \frac{\log \mathcal{L}_* - \log \mathcal{L}_\emptyset}{\log \frac{|\mathcal{A}^{[l]}|+\beta_l-1}{\alpha_l+|\mathcal{A}^{[l]}|-1}}.$$

As $\mathcal{L}_\emptyset$ increases, the bound becomes smaller. Intuitively, if an empty set already achieves high likelihood, adding rules will often hurt the prior term and achieve little gain on the likelihood. This often happens for unbalanced data sets and the output model A is to cover the minor class. The more unbalanced the data set is, the tighter the bound will be.

We derive similar bounds for BRS-Poisson models.

Theorem 5 *Take a BRS-Poisson model with parameters* $H = \{\alpha_+, \beta_+\alpha_-, \beta_-, \lambda_M, \lambda_L\}$, *where* α_+, β_+, α_-,β_-, $\lambda_M, \lambda_L \in \mathbb{N}^+$. *The dataset* S *has* J *attributes, where the* j*-th attribute has* K_j *values for* $j \in \{1, ..., J\}$. *Define* $A^* \in \arg\max_A F(A)$ *and* $M^* := |A^*|$. *If* $\frac{e^{-\lambda_L}\left(\frac{\lambda_L}{2}\right)^J}{\Gamma(J+1)} \leq 1$, *Then*

$$M^* \leq \lambda_M + \frac{\mathit{Prior}_P(\emptyset) + \log \mathcal{L}_* + \log \frac{x^{\lambda_M}}{\lambda_M!} - v_t^*}{\log \frac{\lambda_M+1}{x}},$$

where $x = \frac{e^{-\lambda_L}\lambda_M \max\left\{\left(\frac{\lambda}{2}\right)\Gamma(J), \left(\frac{\lambda}{2}\right)^J\right\}}{\Gamma(J+1)}$.

Proof 6 *Similar to the steps in the proof of Theorem 4, we upper bound* $F(A^*)$. *Let* $L_m, m \in \{1, ..., M^*\}$ *denote the lengths of rules in* A^*. *The prior probability of select-*

ing A^* is

$$p(A^*) = \omega(\lambda_M, \lambda_L)\textit{Poisson}(M^*; \lambda_M) \prod_m^{M^*} \textit{Poisson}(L_m; \lambda_L) \frac{1}{\binom{J}{L_m}} \prod_k^{L_m} \frac{1}{K_{v_{m,k}}}$$
$$= p(\emptyset) \frac{\lambda_M^{M^*}}{M^*!} \prod_m^{M^*} \frac{e^{-\lambda_L} \lambda_L^{L_m} \Gamma(J - L_m + 1)}{\Gamma(J+1)} \prod_k^{L_m} \frac{1}{K_{v_{m,k}}}$$
$$\leq \frac{p(\emptyset)}{M^*!} \left(\frac{e^{-\lambda_L} \lambda_M}{\Gamma(J+1)} \right)^{M^*} \prod_m^{M^*} \left(\frac{\lambda_L}{2} \right)^{L_m} \Gamma(J - L_m + 1). \tag{4.42}$$

(4.42) follows from $K_{v_{m,k}} \geq 2$ since all attributes have at least two values. Using Lemma 1 we have that

$$g(L_m; \lambda_L, J) = \left(\frac{\lambda_L}{2} \right)^{L_m} \Gamma(J - L_m + 1) \leq g_{max}(\lambda, J). \tag{4.43}$$

Combining (4.42) and (4.43) we have

$$p(A^*) \leq \frac{p(\emptyset)}{M^*!} \left(\frac{e^{-\lambda_L} \lambda_M g_{max}(\lambda, J)}{\Gamma(J+1)} \right)^{M^*}. \tag{4.44}$$

The maximum likelihood of data is achieved when all points in the training set are classified correctly,

$$p(S|A^*) \leq \mathcal{L}_*. \tag{4.45}$$

Since $F(A^*) \geq v_t^*$,

$$\textit{Prior}_P(\emptyset) + \log \frac{x^{M^*}}{M^*!} + \log \mathcal{L}_* \geq v_t^* \tag{4.46}$$

where $x = \frac{e^{-\lambda_L} g_{max}(\lambda, J) \lambda_M}{\Gamma(J+1)}$. If $M^* \leq \lambda_M$, the statement of the theorem holds trivially. For the remainder of the proof we consider, if $M > \lambda_M$,

$$\frac{x^{M^*}}{M^*!} \leq \frac{x^{\lambda_M}}{\lambda_M!} \frac{x^{(M^* - \lambda_M)}}{(\lambda_M + 1)^{(M^* - \lambda_M)}},$$

where in the denominator we used $M^*! = \lambda_M(\lambda_M + 1) \cdots M^* \geq \lambda_M!(\lambda_M + 1)^{(M^* - \lambda_M)}$.

So we have

$$\log \frac{x^{\lambda_M}}{\lambda_M!} \left(\frac{x}{\lambda_M + 1} \right)^{(M^* - \lambda_M)} \geq v_t^* - \mathit{Prior}_P(\emptyset) - \log \mathcal{L}_* \tag{4.47}$$

By the theorem's assumption, we have $\frac{e^{-\lambda_L} \left(\frac{\lambda_L}{2}\right)^J}{\Gamma(J+1)} \leq 1$, *so* $\frac{e^{-\lambda_L} \left(\frac{\lambda_L}{2}\right)^J \lambda_M}{\Gamma(J+1)(\lambda_M+1)} < 1$. *We also have* $\frac{e^{-\lambda_L} \left(\frac{\lambda_L}{2}\right) \Gamma(J) \lambda_M}{\Gamma(J+1)(\lambda_M+1)} \leq 1$. *To see this, note* $\frac{\Gamma(J)}{\Gamma(J+1)} < 1$, $\frac{\lambda_M}{\lambda_M+1} < 1$ *and* $e^{-\lambda_L} \left(\frac{\lambda_L}{2}\right) < 1$ *for every* λ_L. *Therefore* $\frac{x}{\lambda_M+1} = \frac{e^{-\lambda_L} g_{max}(\lambda, J) \lambda_M}{\Gamma(J+1)(\lambda_M+1)} < 1$. *Solving for* M^* *in (4.47), using* $\frac{x}{\lambda_M+1} < 1$ *to determine the direction of the inequality yields:*

$$M^* \leq \lambda_M + \frac{\mathit{Prior}_P(\emptyset) + \log \mathcal{L}_* + \log \frac{x^{\lambda_M}}{\lambda_M!} - v_t^*}{\log \frac{\lambda_M + 1}{x}},$$

Using an emptyset as a bench mark model, and setting $v^*(t) = \text{Prior}_\text{P}(\emptyset) + \mathcal{L}_\emptyset$, we derive the following corollary from the above theorem.

Corollary 2 *Take a BRS-Poisson model with parameters* $H = \{\alpha_+, \beta_+ \alpha_-, \beta_-, \lambda_M, \lambda_L\}$, *where* α_+, β_+, α_-,β_-, $\lambda_M, \lambda_L \in \mathbb{N}^+$. *The dataset* S *has* J *attributes, where the* j*-th attribute has* K_j *values for* $j \in \{1, ..., J\}$. *Define* $A^* \in \arg\max_A F(A)$ *and* $M^* := |A^*|$. *If* $\frac{e^{-\lambda_L} \left(\frac{\lambda_L}{2}\right)^J}{\Gamma(J+1)} \leq 1$, *Then*

$$M^* \leq \lambda_M + \frac{\log \mathcal{L}_* - \log \mathcal{L}_\emptyset + \log \frac{x^{\lambda_M}}{\lambda_M!}}{\log \frac{\lambda_M + 1}{x}},$$

where $x = \frac{e^{-\lambda_L} \lambda_M \max\left\{ \left(\frac{\lambda}{2}\right) \Gamma(J), \left(\frac{\lambda}{2}\right)^J \right\}}{\Gamma(J+1)}$.

The upper bounds on $|A^*|$ are meaningful when the algorithm decides which action to choose. When the current model A^t has rules beyond the upper bound, the probability of adding a rule when proposing a new solution needs to be decreased.

Besides an upper bound, the SSS also learns from the current model and tries to improve the objective. To do that, at each iteration $t+1$, an example k is drawn from the data points misclassified by A^t. If example k is positive, it means the current rule set fails to "cover" it and we need to find a new model from the neighbors that covers more data than A^t. If the example is negative, it means the current rule set covers the wrong data and we need to find a neighbor rule set that covers less.

Since we consider both the bounds and the misclassified examples when proposing a neighbor solution, there are situations when the two strategies conflict each other, when A^t has more rules than the upper bound, indicating a CUT action, while the example k is negative, indicating an ADD action. Therefore, we introduce another action, REPLACE, which is simply concatenating CUT and ADD. Then SSS proposes by choosing from ADD, CUT and REPLACE, based on the size of A^t, and the drawn example.

The algorithm is displayed below.

Algorithm 3 Stochastic Smart Search

procedure SSS(*maxSteps*)
  $\mathcal{A}_S^C \leftarrow$ FP-Growth(S, C)
  $A^0 \leftarrow$ a randomly generated rule set,
  $t \leftarrow 0$,
  while *step*<*maxSteps* and $S_\epsilon \neq \emptyset$ **do**
    $S_\epsilon \leftarrow$ misclassified examples by A^t
    $k \leftarrow$ a random example drawn from S_ϵ
    if $Y_k = 1$ **then**

$$A^{t+1} \leftarrow \begin{cases} \text{ADD}(A^t)\text{w.p. } \frac{1}{2}e^{-\max\{0,|A^t|-M^*\}} \\ \text{REPLACE}(A^t)\text{w.p. } 1-\frac{1}{2}e^{-\max\{0,|A^t|-M^*\}} \end{cases}$$

    else

$$A^{t+1} \leftarrow \begin{cases} \text{CUT}(A^t)\text{ w.p. } \frac{1}{2} \\ \text{REPLACE}(A^t)\text{ w.p. } \frac{1}{2} \end{cases}$$

    $A_{\max} \leftarrow \arg\max(F(A_{\max}), F(A^{t+1}))$

$$\alpha = \min\left\{1, \exp\left(\frac{F(A^{t+1})-F(A^t)}{T(t)}\right)\right\}$$

$$A^{t+1} \leftarrow \begin{cases} A^{t+1}, \text{ with probability } \alpha \\ A^t, \text{ with probability } 1-\alpha \end{cases}$$

    $t \leftarrow t+1$
  return $A_{\max}$
end procedure

Now we detail the three actions.

Exploring-and-exploiting

Given an action to perform, SSS needs to choose a rule. Instead of randomly drawing a rule, we evaluate the objective on all possible neighbors obtained via the selected action. Given this list of $F(\cdot)$, SSS makes a choice between exploration, choosing a random rule, and exploitation, choosing the best rule. This randomness helps avoid local minima and

greatly improves search efficiency. We assume the randomness has probability p in our search algorithm. Then the three actions are performed following

- ADD:
  - With probability p, $\text{ADD}_{\text{random}}$.
  - With probability $1-p$, draw a rule z,

$$z = \arg\max_{a \in \mathcal{A} \setminus A^t} F(A^t \cup a).$$

    Then $A^{t+1} \leftarrow A^t \cup z$.

- CUT:
  - With probability p, $\text{CUT}_{\text{random}}$.
  - With probability $1-p$, draw a rule z,

$$z = \arg\max_{a \in A^t} F(A^t \backslash a).$$

    Then $A^{t+1} \leftarrow A^t \backslash z$.

- REPLACE: first CUT, then ADD.

4.5 Generalization Bound

So far our model has been focusing on properties of the *a posteriori* model. We now turn to generalization performance. The following result is algorithm independent. The true risk of a rule set A is defined as:

$$R^{\text{true}}(A) = \mathbb{E}_{(X,Y)\sim D}(A(X) \neq Y),$$

where classifier $A(X)$ equals one when X obeys one of the rules in A, and $A(X)$ equals zero otherwise. The true risk is the standard expected misclassification error.

Theorem 6 *Consider $\Lambda^{\mathcal{F}}$ which is the set of BRS models parameterized by $\{\rho_+, \rho_-\}$, ρ_+, $\rho_- \geq 1/2$, where the number of rules of $A \in \Lambda^{\mathcal{F}}$ obeys $|A| \leq M_{upper}$. With probability $1-\delta$, for all $A \in \Lambda^{\mathcal{F}}$,*

$$R^{true}(A) \leq \frac{\log P(S|A;\rho_+,\rho_-)}{N \log \frac{1}{2}} + \sqrt{\frac{\sum_{m=1}^{M_{upper}} \binom{\prod_j^J (K_j+1)}{m} + \log \frac{1}{\delta}}{2N}}.$$

Proof 7 *Consider the empirical risk on data S given ρ_+, ρ_-:*

$$\begin{aligned}
R^{emp}(A) =& \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}_{y_n \neq f_A(\mathbf{x}_n)} \\
=& \frac{1}{N} \left(\sum_{f(\mathbf{x}_n)=1, y_n=1} 0 + \sum_{f(\mathbf{x}_n)=0, y_n=0} 0 + \sum_{f(\mathbf{x}_n)=1, y_n=0} 1 + \sum_{f(\mathbf{x}_n)=0, y_n=1} 1 \right) \\
\leq& \frac{1}{N} \left(\sum_{f(\mathbf{x}_n)=1, y_n=1} \frac{\log \rho_+}{\log \frac{1}{2}} + \sum_{f(\mathbf{x}_n)=0, y_n=0} \frac{\log \rho_-}{\log \frac{1}{2}} \right. \\
& \left. + \sum_{f(\mathbf{x}_n)=1, y_n=0} \frac{\log(1-\rho_+)}{\log \frac{1}{2}} + \sum_{f(\mathbf{x}_n)=0, y_n=1} \frac{\log(1-\rho_-)}{\log \frac{1}{2}} \right) \qquad (4.48) \\
\leq& \frac{\log P(S|\rho_+,\rho_-)}{N \log \frac{1}{2}}. \qquad (4.49)
\end{aligned}$$

Since $\rho_+, \rho_- > \frac{1}{2}$, (4.48) follows

$$\begin{aligned}
0 \leq \left\{ \frac{\log \rho_+}{\log \frac{1}{2}}, \frac{\log \rho_-}{\log \frac{1}{2}} \right\} \leq 1 \\
\left\{ \frac{\log(1-\rho_+)}{\log \frac{1}{2}}, \frac{\log(1-\rho_-)}{\log \frac{1}{2}} \right\} \geq 1
\end{aligned}$$

Using Hoeffding's Inequality and the union bound, we can get, with probability $1-\delta$, for all $A \in \Lambda^{\mathcal{F}}$,

$$R^{true}(A) \leq R^{emp}(A) + \sqrt{\frac{|\Lambda^{\mathcal{F}}| + \log \frac{1}{\delta}}{2N}}, \qquad (4.50)$$

where $\Lambda^{\mathcal{F}}$ denotes the class of BRS models. $|\Lambda^{\mathcal{F}}|$ can be computed by counting the number of rules sets of different sizes up to M_{upper}, which is in Theorem 1 and Theorem 2 for the

two BRS models.

$$|\Lambda^{\mathcal{F}}| = \sum_{m=1}^{M_{upper}} \binom{|\mathcal{A}|}{m}.$$

$\mathcal{A}$ contains all rules.

$$|\mathcal{A}| = \prod_{j}^{J}(K_j + 1).$$

Therefore

$$|\Lambda^{\mathcal{F}}| \leq \sum_{m=1}^{M_{upper}} \binom{\prod_j^J (K_j + 1)}{m}. \tag{4.51}$$

Combining (4.49), (4.50) and (4.51), we have

$$R^{true}(A) \leq \frac{\log P(S|A; \rho_+, \rho_-)}{N \log \frac{1}{2}} + \sqrt{\frac{\sum_{m=1}^{M_{upper}} \binom{\prod_j^J (K_j+1)}{m} + \log \frac{1}{\delta}}{2N}}. \tag{4.52}$$

This theorem states that the true risk is upper bounded by the empirical risk of data S, in particular the likelihood, and a complexity term. The complexity term increases with $\prod_j^J (K_j + 1)$, which is the number of all rules. The value of M_{upper} can come from either Corollary 1 or Corollary 2.

4.6 Simulation Studies

In this section, we present simulation studies to show that if data are generated from a fixed rule set, both SRS and SSS can recover it with high probability. We also provide convergence analysis on simulated data sets to show that our model can achieve the MAP solution in a relatively short time. We run the two search algorithms with the same data sets and compare their performance.

Given observations $\{\mathbf{x}_n\}_{n=1,\ldots N}$, we assume there exists a true rule set comprised of 5 rules that generate the outcome y_n ($y_n = 1$ if x_n satisfies at least one of the 5 rules). We want to show that our search procedure can find this rule set efficiently. Let there be a collection of rules $\{a_j\}_{j=1,\ldots M}$ mined from the observations, and we can construct a binary matrix where the entry on n-th row and j-th column is the Boolean function $a_j(x_n)$, which represents if the n-th observation satisfies the j-th rules. We need only to simulate

this binary matrix to represent the observations without losing generality. We vary the following variables to study how the convergence and run time change with them.

- M: the number of candidate rules
- N: the number of observations in a data set

We first apply SRS as described in Section 4.4.3. The experiments are indexed as below.

1. Experiments A_1: $N = 10,000, M = 2,000$, srs
2. Experiments A_2: $N = 100,000, M = 2,000$, srs
3. Experiments A_3: $N = 10,000, M = 20,000$, srs

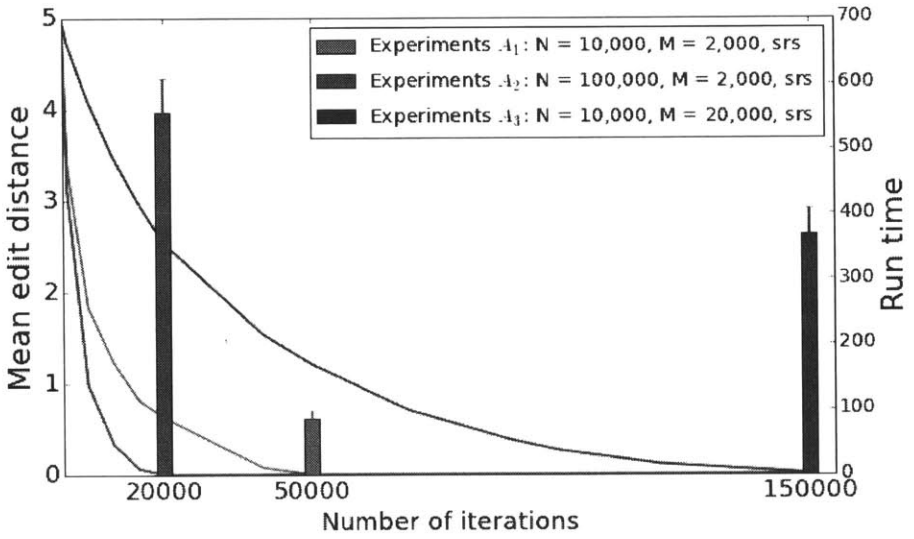


Figure 4-3: Convergence of mean edit distance and running time analysis for stochastic random search.

In each experiment, we first choose (N, M) and then generate a binary matrix of size $N \times M$ representing the data set, by setting each entry to 1 independently with probability 0.1. We then assign the N rules with lengths uniformly selected from 1 to 3. A true rule set is generated by randomly selecting 5 rules. Each experiment is repeated 100 times.

We record the run time till convergence and edit distance between the true rule set and a recovered rule set. We report the mean performance in Figure 4-3 for stochastic random

search. The y-axis on the left represents the edit distance between the true rule set and the recovered rule set. The y-axis on the right represents the run time till convergence.

Experiments A_1 took 50,000 iterations (84.9 seconds) on average to converge. When the size of the data set N is increased to 10 times of the original, the algorithm took fewer iterations (20,000) to converge since the algorithm now has more evidence from observations to find out the true rule. However, experiments A_2 still took longer time (554.0 seconds) than A_1 due to the size of the binary matrix. Experiments A_3 took substantially more iterations to converge when the solution space is increased by 10 times.

For the same data sets, now we apply the SSS described in Section 4.4.4. The experiments are indexed by

1. Experiments B_1: $N = 10,000, M = 2,000$, sss
2. Experiments B_2: $N = 100,000, M = 2,000$, sss
3. Experiments B_3: $N = 10,000, M = 20,000$, sss

Each experiment is repeated 100 times and the mean performances are shown in Figure 4-4.

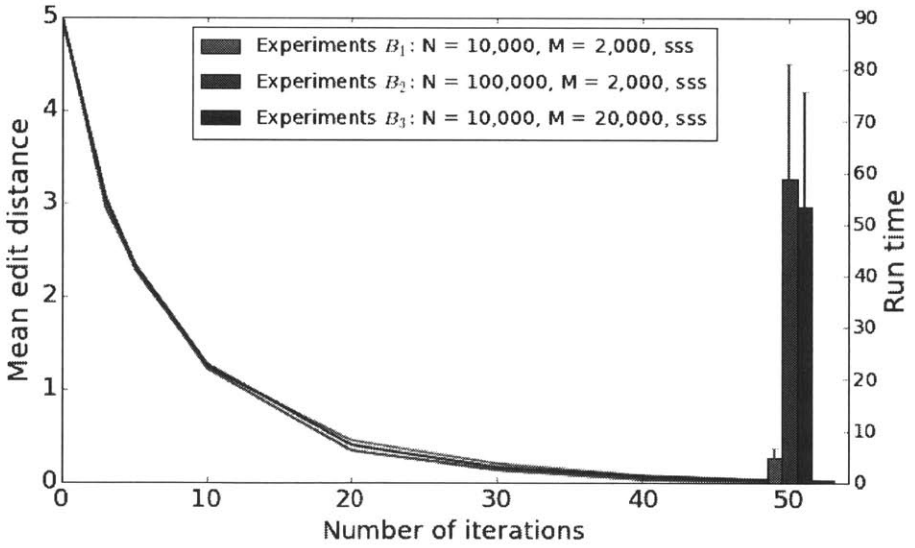


Figure 4-4: Convergence of mean edit distance and running time analysis for stochastic smart search.

Comparing the three sets of experiments, we notice that while it took different time for them to converge due to different sizes of the binary matrix, the three edit distance curves in

Figure 4-4 almost overlap completely despite of the different values of M and N. Neither does the size of the data set nor the size of the rule space affect the search. This is because the proposing strategy based on the potential outcomes of all neighbors is able to pick the best solution efficiently, regardless of the size of data or the rule space.

4.7 Experiments

We test our model on as well as other publicly available datasets. We show that our model can generate interpretable results that are easy for human to understand, without losing too much (if any) predictive accuracy. In situations where the ground truth consists of deterministic rules (similarly to the simulation study), our method tends to perform better than other popular machine learning techniques.

As a main application of BRS, we apply the model to mobile advertisement datasets that we collected and study the rules that characterize contexts and customers that would lead to an acceptance of coupons.

4.7.1 Demonstration with Tic-Tac-Toe Data

For this demonstration, we run the BRS-BetaBinomial model. We show the benefit of applying the two strategies proposed in Section 4.4.4 by running an example data set and show the reduced rule size and convergence performance. We choose tic-tac-toe data set since the data can be classified exactly by 8 rules, shown in Figure 4-5.

Reducing rule space

First, we apply FP-Growth [9] to generate candidate rules. Since we do have no prior knowledge about the support and lengths of the true rules, we set the minimum support to 1 (the rule has to appear at least once in the data set), and the maximum length to 9. FP-growth generates a total of 84429 rules, which are divided by length into 9 pools $\{A^{[l]}\}_{l=1}^{9}$ displayed in Table 4.1

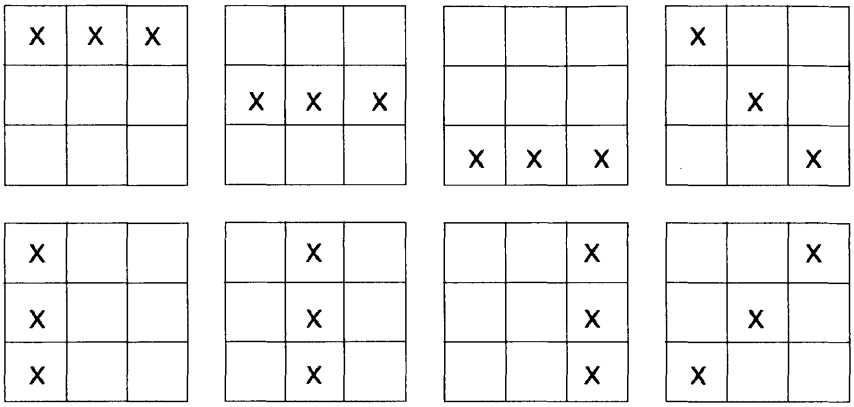


Figure 4-5: Eight rules that classify the tic-tac-toe data set

Table 4.1: Number of rules of different lengths mined from tic-tac-toe data set, with minimum support of 1

l	1	2	3	4	5	6	7	8	9
$\lvert A^{[l]}\rvert$	27	324	2246	9382	21782	26864	17544	5634	626

We then set the hyper-parameters as below,

$$\alpha_l = 1 \text{ for } l = 1, ..., 9$$
$$\beta_l = |\mathcal{A}^{[l]}| \text{ for } l = 1, ..., 9$$
$$\alpha_+ = |S^+| + 1$$
$$\beta_+ = |S^+|$$
$$\alpha_- = |S^-| + 1$$
$$\beta_- = |S^-|,$$

for which we compute the minimum threshold from Theorem 2: $C = 9$.

If however, we would like to get a stronger prior for small models by increasing β_l to 10 times of $|A^{[l]}|$ for all l and keeping all the other parameters unchanged, then a larger threshold for support is obtained: C = 13. This is consistent with the intuition that if the prior penalizes large models more heavily, the output tends to have a smaller size, therefore each rule will need to "cover" more data.

Placing a bound on minimum support is very critical in helping computation since the number of rules decays exponentially with the support. As the minimum support is increased, more rules are removed from the original rule space. We plot in Figure 4-6 the percentage of rules left in the solution space as the minimum threshold is increased from 1 to 9 and 13. At the minimum support of 9, the rule space is reduced to 9.0%; at the minimum support of 13, the rule space is reduced 4.7% to its original size.

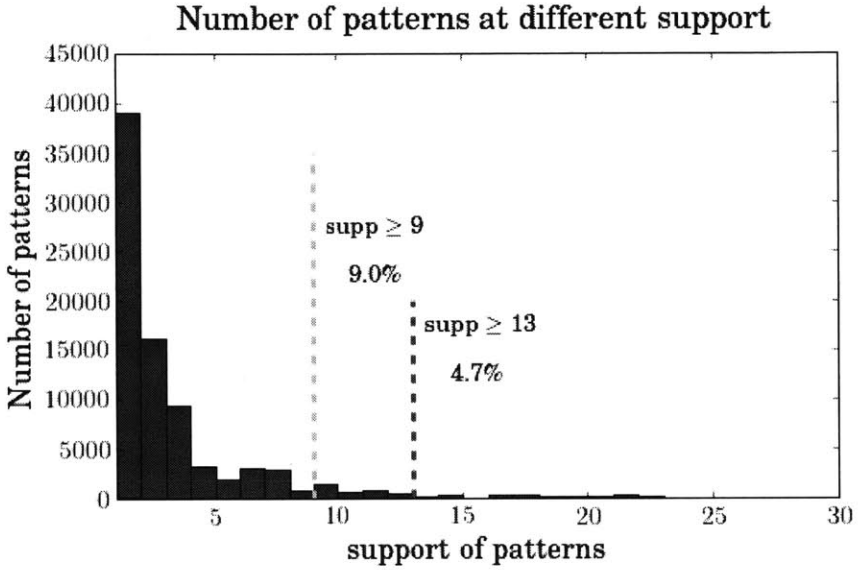


Figure 4-6: Reduced rule space as the minimum support increases

Although rules are filtered out based on their support, we observe that longer rules are more heavily reduced than shorter rules. This is because the support of a rule is negatively correlated with its length, as the more conditions in a rule, the fewer observations can satisfy it. Figure 4-7 shows distributions of rules across different lengths, when mined at different minimum support. Before any filtering (i.e., C = 1), more than half of the rules have lengths greater than 5. As C is increased to 9 and 13, these long rules are almost completely removed from the rule space.

Branch-and-Bounding

We continue with $C = 9$ and 7659 rules, and run the Stochastic Smart Search. We keep a list of the best solution found so far and add to the list when a higher v_t^* is found. We then

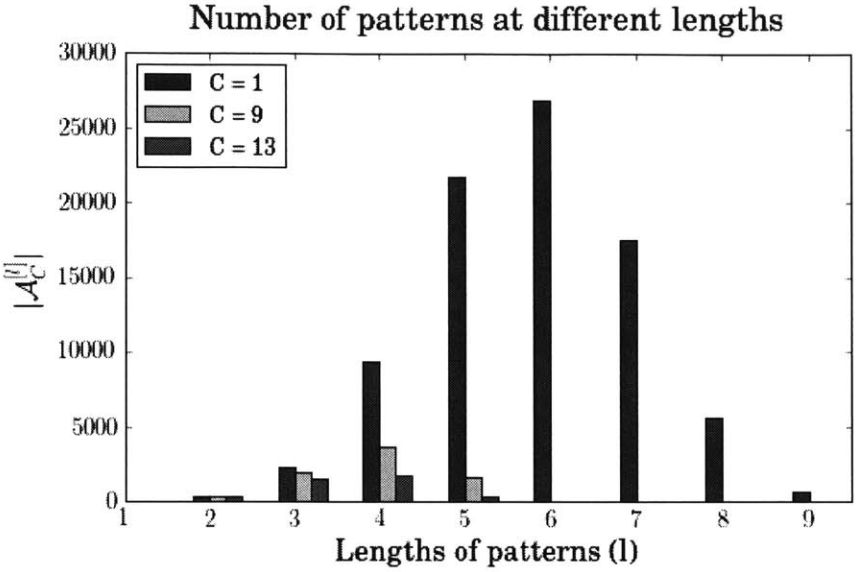


Figure 4-7: Number of rules of different lengths mined at different minimum support C.

update $m_l(t-1)$ with the new v_t^*. We plot in Figure 4-8 the sizes of rules of length 1,2 and 3. The sizes become smaller as v_t^* increases. After 10 iterations, $m_l(t)$ is decreased to below 30. This tells the search algorithm to form small rules sets from a pool of thousands of rules.

4.7.2 Experiments with UCI Data Sets

We tested BRS on nine data sets from the UCI machine learning repository [6] with different types of data; 4 of them are numerical, 4 of them are categorical data and one of them is mixed. We compared with baseline interpretable algorithms Lasso (without interaction terms to preserve interpretability), decision trees, and inductive rule learner RIPPER. We chose random forests and SVM with RBF kernels to represent the set of uninterpretable machine learning methods. Each data set was divided into 5 folds, where models were trained on 4 folds and tested on the fifth fold.

Table 4.2 displays the means and standard deviations of the test accuracy. BRS's performance was on par with the uninterpretable methods, and surpassed that of the interpretable methods. BRS tended perform better on categorical data than numerical or mixed data. (This is because BRS does not need to slice data if it is already categorical.) Two of the

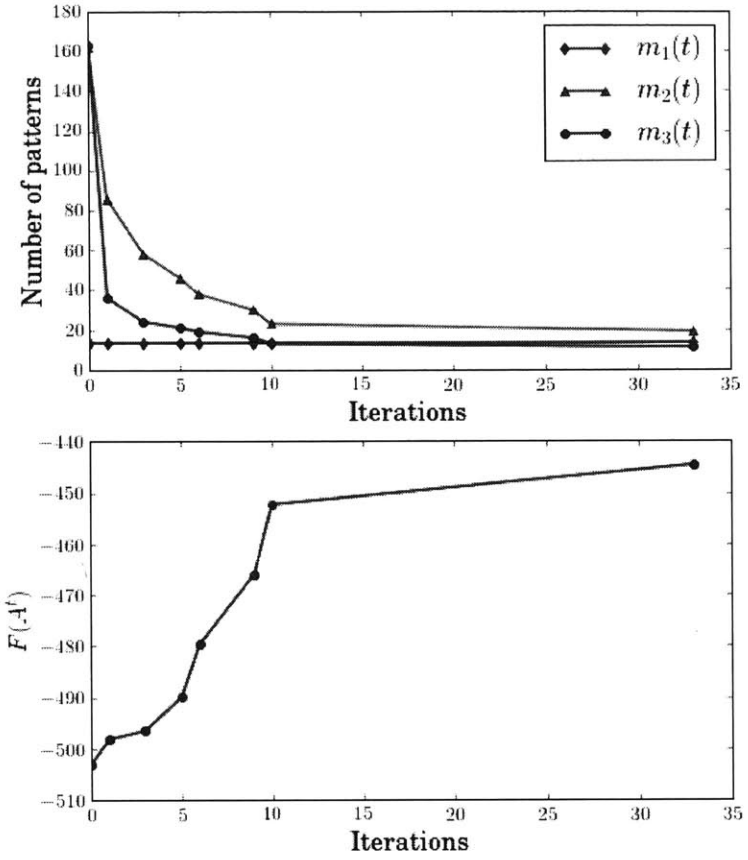


Figure 4-8: Upper bounds $m_l(t)$ updated with v_t^*.

data sets had an underlying structure, in that data were generated from small true rules. BRS achieved perfect accuracy on both of these datasets (tic-tac-toe and monks1), finding the smallest set of conditions that yields perfect accuracy. For instance, the positive class of the tic-tac-toe data set can be identified using exactly 8 conditions. BRS has the capability to exactly learn these conditions, whereas the greedy splitting and pruning methods that are pervasive throughout the data mining literature (e.g., CART, C4.5 and RIPPER) and convexified approximate methods (e.g., SVM) have difficulty with this. Even when these methods are extensively tuned, they still have trouble finding these rules.

Table 4.2: Accuracy comparison for BRS models and baselines on UCI data sets.

	Data	*Interpretable Models*					*Uninterpretable Models*	
	Type	BRS	Lasso	C4.5	CART	RIPPER	random forest	SVM
blogger	Categorical	.85(.05)	.85(.05)	.77(.05)	.78(.07)	.76(.06)	.82(.07)	.82(.10)
votes		.98(.00)	.96(.02)	.96(.02)	.96(.03)	.96(.01)	.95(.02)	.97(.02)
tic-tac-toe		1(.00)	.71(.02)	.92(.03)	.93(.02)	.98(.01)	.99(.00)	.99(.00)
monks1		1(.00)	.76(.02)	.90(.06)	.88(.07)	.94(.12)	1(.00)	1(.00)
bupa	Numerical	.74(.01)	.68(.04)	.63(.04)	.68(.03)	.65(.05)	.70(.03)	.73(.04)
transfusion		.79(.01)	.77(.02)	.76(.02)	.78(.02)	.78(.02)	.78(.02)	.80(.02)
banknote		1(.00)	1(.00)	.90(.01)	.90(.02)	.91(.01)	.91(.01)	1(.00)
indian-diabetes		.76(.02)	.67(.01)	.66(.03)	.67(.01)	.67(.02)	.76(.02)	.69(.01)
heart	mixed	.83(.03)	.85(.04)	.76(.06)	.77(.06)	.78(.04)	.81(.06)	.86(.06)

Parameter Tuning

A MAP solution minimizes the sum of negative logs of prior and likelihood. The scale of the prior and likelihood directly determines how the model trades off between fitting the data and achieving the desired sparsity level. We study how sensitive the results are to different parameters and how to tune the parameters in order to get the best performing model. The Bayesian model uses parameters $\{\alpha_l, \beta_l\}_{l=1}^{L}$ to govern the prior for selecting rules and parameters $\alpha_+, \beta_+, \alpha_-, \beta_-$ to govern the likelihood of data. We fix the prior parameters $\{\alpha_l, \beta_l\}_{l=1}^{L}$ and only vary likelihood parameters $\alpha_+, \beta_+, \alpha_-, \beta_-$ to analyze how the performance changes as the magnitude and ratio of the parameters change. To simplify the study, we assume $\alpha_+ = \alpha_- = \alpha$, and $\beta_+ = \beta_- = \beta$ for all experiments and vary α and β. Let $s = \alpha + \beta$ and s is chosen from $\{100, 1000, 10000\}$. Let $p = \frac{\alpha}{\alpha+\beta}$, which is the mean of ρ_+ and ρ_- according to Section 4.3.2. Value p varies within $[0, 1]$. The closer p is to 1, the better the model fits the data. Here s and p uniquely define α and β.

We report the out-of-sample accuracy as s and p change and plot in Figure 4-9 the 5-fold experiments for all data sets used in Table 4.2. The accuracy increases as p increases, which is consistent with the intuition of ρ_+ and ρ_-. The highest accuracy is always achieved at the right half of the curve. The performance is less sensitive to the magnitude s, especially when $p > 0.5$. We notice that, for some data sets, the accuracy became flat once p becomes

greater than a certain threshold and the performance is not sensitive to p either after that point, which makes the tuning very easy in practice. Generally speaking, taking p close to 1 would usually lead to a satisfactory output. In the experiments, we chose s and p via nested cross-validation.

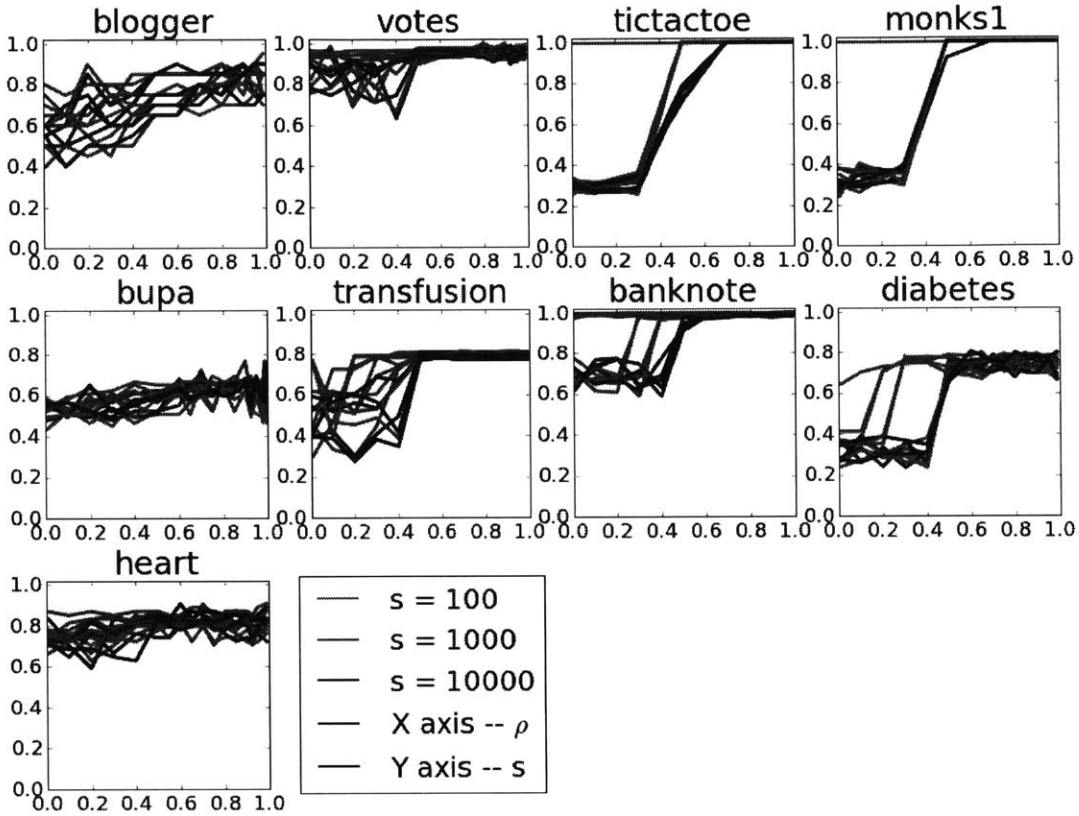


Figure 4-9: Parameter tuning experiments. Accuracy vs. ρ for all datasets.

Examples of BRS models

We illustrated BRS's output on the heart data set in the introduction. BRS took less than one minutes on a laptop to generate that model, which achieves an out-of-sample accuracy of 0.85. For another data set (pima-indian-diabetes), the BRS model consists of only one rule '$95 \leq$ plasma ≤ 125 AND age ≤ 81 AND BMI ≥ 31' and achieves an out-of-sample accuracy of 0.78.

Rule sets models could potentially be useful for medical applications, since they could

characterize simple sets of conditions that would place a patient in a high risk category. This may be more useful in some cases than the typical (linear) scoring systems used in medical calculators.

4.8 Application to In-vehicle Recommender Systems

For this experiment, our goal was to understand customers' response to recommendations made in an in-vehicle recommender system that provide coupons for local businesses. The coupons would be targeted to the user in his/her particular context. Our data were collected on Amazon Mechanical Turk via a survey that we will describe shortly. We used Turkers with high ratings (95% or above) and used two random questions with easy answers to reject surveys submitted by workers who were not paying attention. Out of 752 surveys, 652 were accepted, which generated 12684 data cases (after removing rows containing missing attributes). The attributes of this data set include:

1. User attributes

    - Gender: male, female
    - Age: below 21, 21 to 25, 26 to 30, etc.
    - Marital Status: single, married partner, unmarried partner, or widowed
    - Number of children: 0, 1, or more than 1
    - Education: high school, bachelors degree, associates degree, or graduate degree
    - Occupation: architecture & engineering, business & financial, etc.
    - Annual income: less than $12500, $12500 - $24999, $25000 - $37499, etc.
    - Number of times that he/she goes to a bar: 0, less than 1, 1 to 3, 4 to 8 or greater than 8
    - Number of times that he/she buys takeaway food: 0, less than 1, 1 to 3, 4 to 8 or greater than 8
    - Number of times that he/she goes to a coffee house: 0, less than 1, 1 to 3, 4 to 8 or greater than 8

- Number of times that he/she eats at a restaurant with average expense less than $20 per person: 0, less than 1, 1 to 3, 4 to 8 or greater than 8
- Number of times that he/she goes to a bar: 0, less than 1, 1 to 3, 4 to 8 or greater than 8

2. Contextual attributes

   - Driving destination: home, work, or no urgent destination
   - Location of user, coupon and destination: we provide a map to show the geographical location of the user, destination, and the venue, and we mark the distance between each two places with time of driving. The user can see whether the venue is in the same direction as the destination.
   - Weather: sunny, rainy, or snowy
   - Temperature: 30F°, 55F°, or 80F°
   - Time: 10AM, 2PM, or 6PM
   - Passenger: alone, partner, kid(s), or friend(s)

3. Coupon attributes

   - time before it expires: 2 hours or one day

All coupons provide a 20% discount. The survey was divided into different parts, so that Turkers without children would never see a scenario where their "kids" were in the vehicle. Figure 4-10 and 4-11 show two examples of scenarios in the survey.

The prediction problem is to predict if a customer is going to accept a coupon for a particular venue, considering demographic and contextual attributes. Answers that the user will drive there 'right away' or 'later before the coupon expires' are labeled as '$Y = 1$' and answers 'no, I do not want the coupon' are labeled as '$Y = 0$'. We are interested in investigating 5 types of coupons: bars, takeaway food restaurants, coffee houses, cheap restaurants (average expense below $20 per person), expensive restaurants (average expense between $20 to $50 per person). In the first part of the survey, we asked users to provide their demographic information and preferences. In the second part, we described

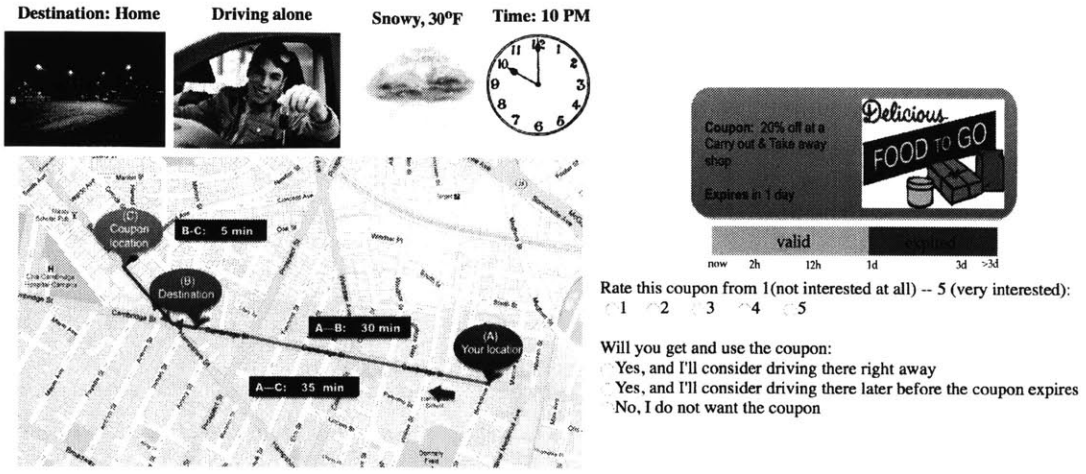


Figure 4-10: Example 1 of scenario in the survey

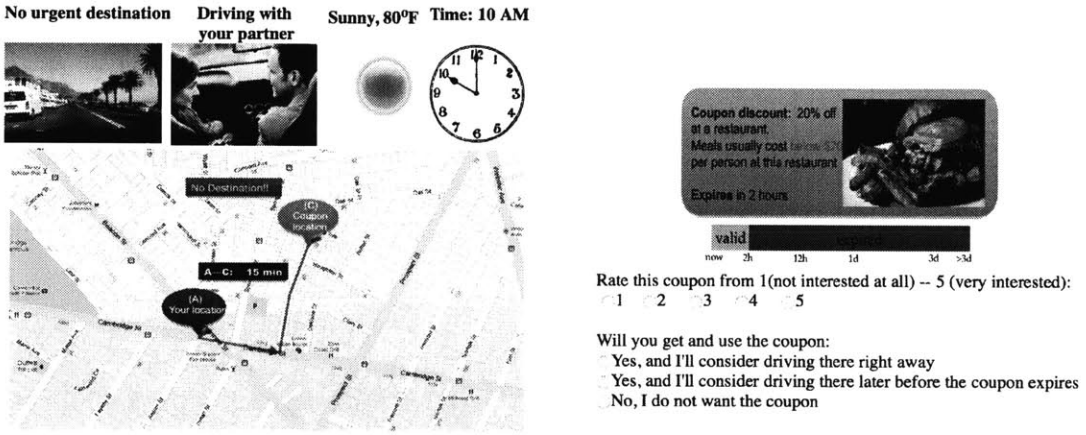


Figure 4-11: Example 2 of scenario in the survey

20 different driving scenarios (see examples in Appendix) to each user along with additional context information and coupon information (see Appendix for a full description of attributes) and asked the user if s/he will use the coupon.

For this problem, we want to generate simple BRS models that are easy to understand. So we restricted the lengths of rules and the number of rules to yield very sparse models. Before mining rules, we convert each row of data into an item which is an attribute and value pair. For categorical attributes, each attribute-value pair was directly coded into a condition. Using marital status as an example, 'marital status is single' is converted into (MaritalStatus: Single), (MaritalStatus: Not Married partner), and (MaritalStatus: Not Un-

married partner), (MaritalStatus: Not Widowed). For discretized numerical attributes, the levels are ordered, such as: age is '20 to 25', or '26 to 30', etc; each attribute-value pair was converted into two conditions, each using one side of the range. For example, age is '20 to 25' was converted into (Age:>=20) and (Age:<=25). Then each condition is a half-space defined by threshold values. For the rule mining step, we set the minimum support to be 5% and set the maximum length of rules to be 4. We used information gain in Equation (4.31) in Section 4 to select the best 5000 rules to use for BRS. We ran simulated annealing for 50000 iterations to obtain a rule set.

We compared with interpretable classification algorithms that span the space of widely used methods that are known for interpretability and accuracy, C4.5, CART, Lasso, Ripper [18], and a naive baseline using top K rules, referred to as Top$K$. For C4.5, CART, Lasso and Ripper, we used the RWeka package in R and tuned the hyperparameters to generate different models on a ROC plane. For the Top$K$ method, we varied K from 1 to 10 to produce 10 models using best K pre-mined rules ranked by the information gain in equation (4.31). For BRS, We varied the hyperparameters $\alpha_+, \beta_+, \alpha_-, \beta_-$ to obtain different sets of rules. For all methods, we picked models on their ROC frontiers and reported their performance below.

Performance in accuracy To compare their predictive performance, we measured out-of-sample AUC (the Area Under The ROC Curve) from 5-fold testing for all methods, reported in Table 4.3. In the table, BRS1 represents BRS-BetaBinomial and BRS2 represents BRS-Poisson. The BRS classifiers, while restricted to produce sparse disjunctions of conjunctions, had better performance than decision trees and Ripper, which use greedy splitting and pruning methods, and do not aim to globally optimize. Top$K$'s performance was substantially below that of other methods. BRS models are also comparable to Lasso, but the form of the model is different. BRS models do not require users to cognitively process coefficients.

Performance in complexity For the same experiment, we would also like to know the complexity of all methods at different accuracy levels. Since the methods we compare

	Bar	Takeaway Food	Coffee House	Cheap Restaurant	Expensive Restaurant
BRS1	0.773 (0.013)	0.682 (0.005)	0.760 (0.010)	0.736 (0.022)	0.705 (0.025)
BRS2	0.776 (0.011)	0.667 (0.023)	0.762 (0.007)	0.736 (0.019)	0.707 (0.030)
C4.5	0.757 (0.015)	0.602 (0.051)	0.751 (0.018)	0.692 (0.033)	0.639 (0.027)
CART	0.772 (0.019)	0.615 (0.035)	0.758 (0.013)	0.732 (0.018)	0.657 (0.010)
Lasso	0.795 (0.014)	0.673 (0.042)	0.786 (0.011)	0.769 (0.024)	0.706 (0.017)
Ripper	0.762 (0.015)	0.623 (0.048)	0.762(0.012)	0.705(0.023)	0.689 (0.034)
Top$K$	0.562 (0.015)	0.523 (0.024)	0.502(0.012)	0.582(0.023)	0.508 (0.011)

Table 4.3: AUC comparison for mobile advertisement data set, means and standard deviations over folds are reported.

have different structures, there is not a straightforward way to directly compare complexity. However, decision trees can be converted into equivalent *or*'s of *and*'s models. For a decision tree, an example is classified as positive if it falls into any positive leaves. Therefore, we can generate equivalent models in *or*'s of *and*'s form for each decision tree, by simply collecting branches with positive leaves. Therefore, the four algorithms we compare, C4.5, CART, Ripper and BRS have the same form. To measure the complexity, we count the total number of conditions in the model, which is the sum of lengths for all rules. This loosely represents the cognitive load needed to understand a model. For each method, we take the models that are used to compute the AUC in Table 4.3 and plot their accuracy and complexity in Figure 4-12. BRS models achieved the highest accuracy at the same level of complexity. This is not surprising given that BRS performs substantially more optimization than other methods.

To show that the benefits of BRS did not come from rule mining or screening using heuristics, we compared the BRS models with Top$K$ models that rely solely on rule mining and ranking with heuristics. Figure 4-13 shows there is a substantial gain in the accuracy of BRS models compared to Top$K$ models.

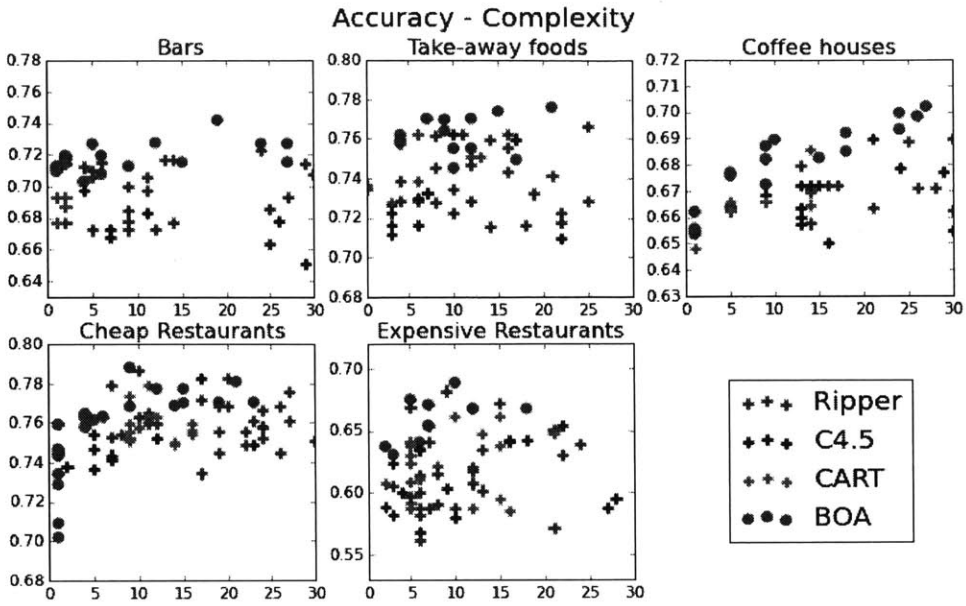


Figure 4-12: Accuracy and complexity comparison for BRS and other interpretable models on mobile advertisement data sets for different coupons

Examples of BRS models In practice, for this particular application, the benefits of interpretability far outweigh small improvements in accuracy. An interpretable model can be useful to a vender when choosing whether to provide a coupon and what type of coupon to provide, it can be useful to users of the recommender system, and it can be useful to the designers of the recommender system to understand the population of users and correlations with successful use of the system. As discussed in the introduction, *or*'s of *and*'s classifiers are particularly natural for representing consumer behavior, particularly consideration sets, as modeled here.

We show several classifiers produced by BRS in Figure 4-14, where the curves were produced by models from previous experiments. Example rule sets are listed in each box along the curve. For instance, the classifier near the middle of the curve in Figure 4-14 (a) has one rule, and reads "If a person visits a bar at least once per month, is not traveling with kids, and their occupation is not farming/fishing/forestry, then predict the person will use the coupon for a bar before it expires." In these examples (and generally), we see that a user's general interest in a coupon's venue (bar, coffee shop, etc.) is the most relevant attribute to the classification outcome; it appears in every rule in the two figures.

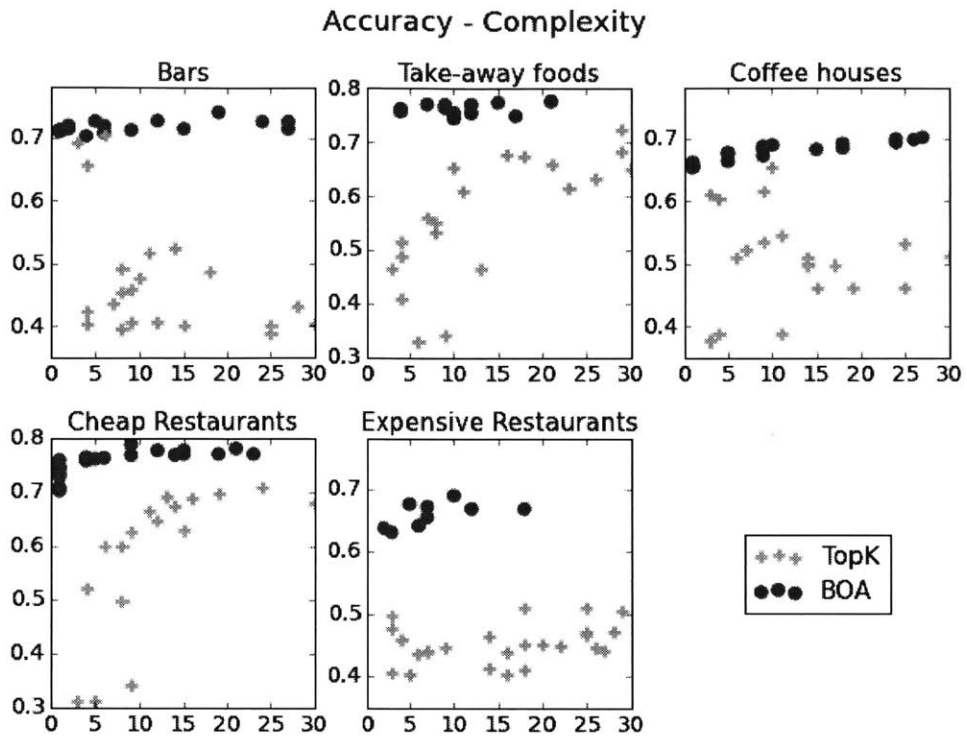
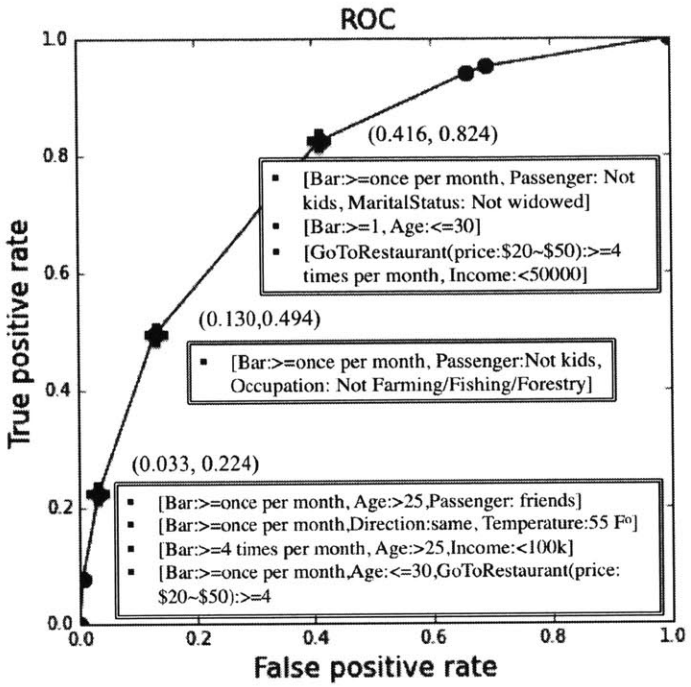


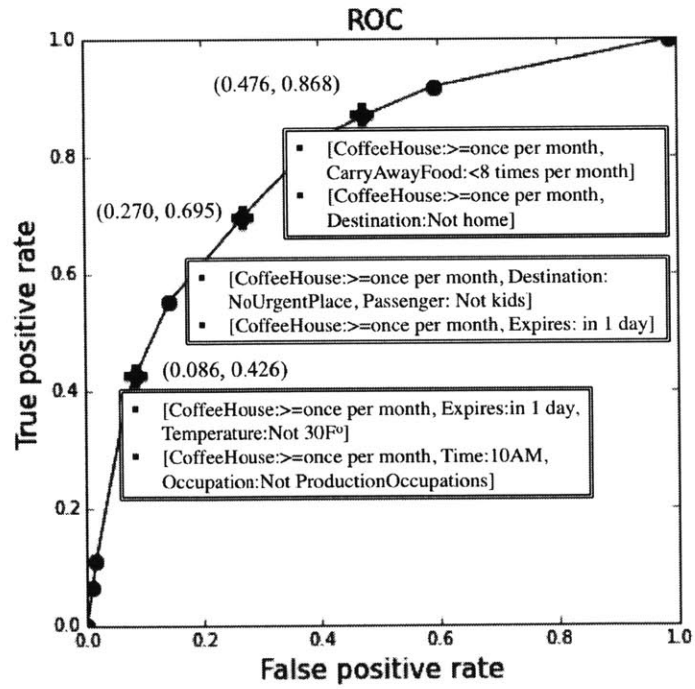
Figure 4-13: Accuracy and complexity comparison for BRS and Top$K$ on mobile advertisement data sets for different coupons

4.9 Conclusion

We presented a method that produces Rule Set models, where the shape of the model can be controlled by the user through Bayesian priors. In some applications, such as those arising in customer behavior modeling, the form of these models may be more useful than traditional linear models. Since finding sparse models is computationally hard, most approaches take severe heuristic approximations (such as greedy splitting and pruning in the case of decision trees, or convexification in the case of linear models). These approximations can severely hurt performance, as is easily shown experimentally, using datasets whose ground truth formulas are not difficult to find. We chose a different type of approximation, where we make an up-front statistical assumption in building our models out of pre-mined rules, and find the globally optimal solution in the reduced space of rules. We then find theoretical conditions under which using pre-mined rules provably does not change the set of MAP optimal solutions. These conditions relate the size of the dataset to the strength of the prior.



(a) Coupons for bars



(b) Coupons for coffee houses

Figure 4-14: ROC for dataset of coupons for bars and coffee houses

If the prior is sufficiently strong and the dataset is not too large, the set of pre-mined rules is provably sufficient. We showed the benefits of this approach on a consumer behavior modeling application of current interest to "connected vehicle" projects. Our results, using data from an extensive survey taken by several hundred individuals, show that simple rules based on a user's context can be directly useful in predicting the user's response.

Chapter 5

Optimized Rule Set

5.1 Introduction

Bayesian Rule Set has the advantage of a Bayesian interpretation of the prior parameters, but a disadvantage in that the analytical bounds on the maximum a posteriori solution of the Bayesian method are weaker than those we present in this chapter for the minimizer of the optimization problem. In this chapter, we present mathematical programming formulations for producing Rule Sets. Our goal is again, cognitive simplicity, as well as predictive accuracy. We choose mathematical programming (mixed-integer and integer linear programming – MIP and ILP) and rule mining to form our models. Using these tools have several benefits, namely flexibility on the user's side on the objective and constraints, fast solvers that have been improving exponentially over recent years, and a guarantee on the optimality of the solution. We improve computation also using statistical approximations. In one of our algorithms, ORSx, we first mine rules and design an ILP to choose the subset of rules to form the rule set model. This is a statistical assumption that dramatically speeds up computation, but we can show (in Theorem 8) that as long as we mine all rules with sufficiently high support, the optimal solution will be attained anyway. We also present various bounding conditions on the optimal solution.

5.2 Optimized Rule Sets

Let us discuss the first framework for learning Rule Sets, called Optimized Rule Sets (ORS). We work with a data set $S = \{(X_n, Y_n)\}_n^N$, consisting of N examples with J attributes of mixed type. $Y_n \in \{1, -1\}$ represents the labels. Numerical attributes are indexed by index set $\mathcal{J}_\text{n}$ and categorical attributes are indexed by $\mathcal{J}_\text{c}$. The j-th attribute of the n-th example is denoted as X_{nj}.

An rule set classifier consists of a set of rules that characterize a single class, here, the positive class. Each rule is a conjunction of conditions (literals), and the number of conditions is called the *length* of a rule. For example. the length of rule "age $\geq$ 30 *AND* has hypertension *AND* is female" is 3. Let z denote a rule, and $z(X)$ indicate if X satisfies rule z. A represents a set of rules. An ORS classifier built on A is denoted as:

$$A(X) = \begin{cases} 1 & \exists z \in A, \text{s.t.} z(X) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

5.2.1 MIP Formulation

We formulate a mixed integer program to generate a rule set containing numerical and categorical attributes. The MIP uses the following objective $L(A)$ to minimize the training error while maintaining sparseness of the model.

$$L(A) = \frac{\#\text{error}(A)}{N} + C_1 \#\text{literals}(A) + C_2 \#\text{rules}(A). \tag{5.2}$$

The first term in the objective is the loss function, which counts the number of classification errors. The regularization terms include 1) the total number of literals in A, denoted as $\#\text{literals}(A)$, which is the sum of the length of each rule in A, and 2) the total number of rules in A, denoted as $\#\text{rules}(A)$. The two terms are scaled by parameters C_1 and C_2 to penalize the complexity of the model. C_1 represents the percentage of training errors the user is willing to trade in order to reduce a rule by one literal. Similarly, C_2 represents the percentage of training errors a user needs to trade to reduce one rule. A user can tune C_1 and C_2 to influence the shape of the output.

Now we explain how the constraints work. A challenging part is to deal with both numerical and categorical attributes in the MIP. For numerical attributes, the MIP needs to select the upper and lower boundary to form a range; for categorical attributes, it needs to select a category for a literal. Simultaneously, the MIP needs to decide for each example if it satisfies the literals and rules. All of the constraints are linear in the decision variable, to ensure a duality gap or proof of optimality on the solution we obtain.

Literals for Numerical Attributes

For a numerical attribute, a literal j (for simplicity, when we refer to literal j, we mean a literal containing attribute j) has the form "$l_{kj} \leq X_{.j} \leq u_{kj}$", where u_{kj} and l_{kj} represent the upper and lower boundary of the range in literal j. k is a rule index. For each example X_n, let $\hat{u}_{nkj} \in \{0,1\}$ indicate if X_{nj} satisfies the upper bound u_{kj} and $\hat{l}_{nkj} \in \{0,1\}$ indicate if X_{nj} satisfies the lower bound l_{kj}. That is, $\hat{u}_{nkj} = 1$ if $X_{nj} \leq u_{kj}$, and $\hat{l}_{nkj} = 1$ if $X_{nj} \geq l_{kj}$. Using a big M formulation, we obtain the following constraints, that for $\forall n, k, \forall j \in \mathcal{J}_n$,

$$u_{kj} - X_{nj} \leq M\hat{u}_{nkj}, \tag{5.3}$$

$$u_{kj} - X_{nj} \geq M(\hat{u}_{nkj} - 1) + \epsilon, \tag{5.4}$$

$$X_{nj} - l_{kj} \leq M\hat{l}_{nkj}, \tag{5.5}$$

$$X_{nj} - l_{kj} \geq M(\hat{l}_{nkj} - 1) + \epsilon, \tag{5.6}$$

$$u_{kj} \leq U_j \tag{5.7}$$

$$l_{kj} \geq L_j \tag{5.8}$$

A small number ϵ is used to force $\hat{u}_{nkj} = 1$ when $X_{nj} = u_{kj}$, and force $\hat{l}_{nkj} = 1$ when $X_{nj} = l_{kj}$. U_j and L_j denote the maximum and minimum value of attribute j. Constraints (5.7) and (5.8) bound on u_{kj} and l_{kj} to ensure a bounded M for computation efficiency.

It is possible that the upper and lower bounds apply to all examples, when both constraints (5.7) and (5.8) are binding. In that case, the literal does not have any classification power. We need numerical literals that are meaningful, or what we call *substantive*, that

their ranges only apply to a subset of training examples. This means at most one of constraints (5.7) and (5.8) can be binding. Let a binary variable δ_{kj} indicates if literal j in rule k is substantive. We use a big M formulation to construct the constraints. Therefore $\forall k, \forall j \in \mathcal{J}_{\text{n}}$,

$$M\delta_{kj} \geq U_j - u_{kj}, \tag{5.9}$$

$$M\delta_{kj} \geq l_{kj} - L_j, \tag{5.10}$$

$$M(1 - \delta_{kj}) \geq (u_{kj} - l_{kj}) - (U_j - L_j) + \epsilon, \tag{5.11}$$

Constraint (5.9) means $\delta_{kj} = 1$ if $u_{kj} < U_j$. Constraint (5.10) means $\delta_{kj} = 1$ if $l_{kj} > L_j$. (5.11) forces $\delta_{kj} = 0$ when $u_{kj} = U_j$ and $l_{kj} = L_j$, i.e., literal j is non-substantive.

Literals for Categorical Attributes

For a categorical attribute, a literal j has the form "$X_{.j}$ = the v-th category", where $v \in \{1, ...V_j\}$ is an index for categories of attribute j and V_j is the total number of categories. We use $o_{kjv} \in \{0, 1\}$ to indicate whether the v-th category of attribute j is present in literal j of rule k. To determine if X_n satisfies the condition in literal j, let $\hat{o}_{nkj} \in \{0, 1\}$ indicate if X_{nj} equals the category contained in this literal. We binary code X_{nj} into X_{njv} such that $X_{njv} = 1$ if X_{nj} takes the v-th category of attribute j. Therefore $\hat{o}_{nkj} = 1$ if and only if there exists $v \in \{1, ...V_j\}$ such that $o_{kjv} = 1$ and $X_{njv} = 1$. We formulate it as the following. For $\forall n, k, \forall j \in \mathcal{J}_{\text{c}}$,

$$\hat{o}_{nkj} \leq \sum_v X_{njv} o_{kjv}, \tag{5.12}$$

$$V_j \hat{o}_{nkj} \geq \sum_v X_{njv} o_{kjv}, \tag{5.13}$$

$$\sum_{v \in \{1,...,V_j\}} o_{kjv} \leq 1. \tag{5.14}$$

(5.14) ensures each categorical literal contains at most one value. This constraint forces a rule to have a fixed form. If we remove the constraint and allow a literal to take multiple values, a rule could have the following form: "$5 \leq X_1 \leq 20$ AND X_2 = red or blue." It

depends on the application and users' preference as to whether leave this constraint in the MIP. The model will work the same without changing the rest of the formulation.

We define that a categorical literal j is *substantive*, if there exists some $v \in \{1, ..., V_j\}$ such that $o_{kjv} = 1$, indicated by $\delta_{kj} \in \{0, 1\}$. For $\forall k, \forall j \in \mathcal{J}_c$,

$$\delta_{kj} \leq \sum_{v \in \mathcal{V}_j} o_{kjv}, \tag{5.15}$$

$$V_j \delta_{kj} \geq \sum_{v \in \mathcal{V}_j} o_{kjv}. \tag{5.16}$$

Counting Classification Errors

Given the literals, X_n satisfies rule k if and only if it satisfies every literal in the rule. For categorical attributes, we consider both cases where a literal is substantive, i.e., $\delta_{kj} = 1$, and we need $\hat{o}_{nkj} = 1$; or non-substantive, i.e. $\delta_{kj} = 0$, and $\hat{o}_{nkj}$ is always 0 for all v. For numerical attributes, when the literal is substantive, the MIP needs to check if a data point satisfies both upper and lower bounds of the range, indicated by $\hat{u}_{nkj}$ and $\hat{l}_{nkj}$; when the literal is non-substantive, i.e., $u_{kj} = U_j$ and $l_{kj} = L_j$, then $\hat{u}_{nkj} = 1$ and $\hat{l}_{nkj} = 1$ for all X_n. Using $\omega_{nk} \in \{0, 1\}$ to indicate if X_n satisfies rule k, the above conditions can be formulated below. $\forall n, k$,

$$\zeta_k + \sum_{j \in \mathcal{J}_n} \left(\hat{u}_{nkj} + \hat{l}_{nkj} \right) + \sum_{j \in \mathcal{J}_c} (\hat{o}_{nkj} + 1 - \delta_{kj}) - (2|\mathcal{J}_n| + |\mathcal{J}_c|) \leq \omega_{nk}, \tag{5.17}$$

$$(2|\mathcal{J}_n| + |\mathcal{J}_c| + 1)\omega_{nk} \leq \zeta_k + \sum_{j \in \mathcal{J}_n} \left(\hat{u}_{nkj} + \hat{l}_{nkj} \right) + \sum_{j \in \mathcal{J}_c} (\hat{o}_{nkj} + 1 - \delta_{kj}). \tag{5.18}$$

Let $\xi_n \in \{0, 1\}$ indicate if a classification error is made, which means either a positive data point does not satisfy any rule, or a negative data point satisfies at least one rule. In both cases $\xi_n = 1$. These two situations are captured by constraints (5.19) and (5.20).

$$\xi_n + \sum_{k}^{K} \omega_{nk} \geq 1, \forall n \in \mathcal{I}^+, \tag{5.19}$$

$$K\xi_n \geq \sum_{k}^{K} \omega_{nk}, \forall n \in \mathcal{I}^-, \tag{5.20}$$

where $\mathcal{I}^+$ denotes the set of indices for positive examples and $\mathcal{I}^-$ denotes the set of indices for negative examples. K is the upper bound on the number of rules that we allow the solution to have. MIP creates this K "boxes" that will be filled up as it searches in the solution space.

When the MIP is formulated, we do not know how many out of the K "boxes" the MIP will use. Therefore we introduce binary variables $\zeta_k \in \{0, 1\}$ to indicate if rule k is non-empty in the final rule set, which means it contains at least one substantive literal. For $\forall k$,

$$J\zeta_k \geq \sum_j \delta_{kj}. \tag{5.21}$$

Since we are minimizing the total number of rules in the objective, the constraint will always be binding.

The Objective

Now we reprsent the objective using decision variables introduced before. The MIP minimizes

$$\frac{1}{N}\sum_{n=1}^{N}\xi_n + C_1\sum_{k}^{K}\sum_{j}^{J}\delta_{kj} + C_2\sum_{k=1}^{K}\zeta_k$$

over variables ω_{nk},u_{kj}, l_{kj}, $\hat{u}_{nkj}$, $\hat{l}_{nkj}$, o_{kjv}, $\hat{o}_{nkj}$, δ_{kj}, δ_j, ξ_n, and ζ_k, such that they satisfy constraints (5.3) to (5.21).

The complexity of the MIP comes from three aspects, 1) choosing the upper and lower boundaries for ranges in numerical literals, and picking categories for categorical literals, 2) deciding for each example if it satisfies every literal and every rule, and 3) deciding how many rules are constructed from the K "boxes." There are in total $\mathcal{O}(NKJ)$ constraints and $\mathcal{O}(NKJ)$ decision variables for this MIP, though the full matrix of variables corresponding to the mixed integer programming formulation is sparse since most literals operate only on a small subset of the data. This formulation can be solved efficiently for small to medium sized datasets. As the size of the dataset grows, the computation gets complicated. We might need a faster method that operates in an approximate way on a much larger scale,

presented below.

5.3 Optimized Rule Sets with Approximations

To speed up the learning process, we propose **Optimized Rule Sets with Approximations** (ORSx), that separates from the optimization process, the first two previously mentioned aspects of complexity. ORSx uses a *pre-mining then selecting* approach. It takes advantage of mature rule mining techniques to generate a set of rules. Then a secondary criteria is applied to further screen the rules to form a candidate rule set. Finally, an integer linear program (ILP) searches within these rules set for an optimal set. This method consists of following three steps, *rule mining*, *rule screening* and *rule selecting*.

5.3.1 Rule Mining

There are many frequently used rule mining methods such as FP-growth [35], Apriori [3], Eclat [99], etc. In our implementation, we use FP-growth in python [9] that takes the binary coded data, and user specified minimum support and maximum length, to generate rules that satisfy the two requirements. The algorithm runs sufficiently fast (usually less than a second for thousands of observations). Since the FP-growth algorithm handles binarized data, we discretize the numerical attributes by manually selecting thresholds for bins. For instance, $X = 3.5$ can be transformed into $2 \leq X \leq 4$, etc. Note that there are other rule mining techniques that handle real-valued variables.

5.3.2 Rule Screening

In the rule mining step, the number of generated rules is usually overwhelming for even a medium size data set. For instance, for the sleep apnea data set (which we will discuss in detail in the experiment sections) of size 1192 patients and 112 binary coded attributes, if the maximum length is 3 and the minimum support is 5%, millions of rules are generated. Ideally, we would like the candidate rule set to contain thousands of rules for computational

convenience. Therefore, we use a secondary criteria to further screen the rules.

$$\text{Score}(z) = \text{InfoGain}(S|z) - \gamma l_z. \tag{5.22}$$

This criteria considers the classification power of a rule, measured by information gain $\text{InfoGain}(S|z)$, and the sparsity, measured by the length of the rule l_z. Information gain of rule z on data S is $\text{InfoGain}(S|z) = H(S) - H(S|z)$, where $H(S)$ is the entropy of S, written as $H(S) = -\sum_i P_i \log P_i$. $H(S|z)$ is the conditional entropy of S. Using this criteria, we select a set of candidate rules $\mathcal{P}$ of size $K_{\mathcal{P}}$.

To represent the sparseness of each rule, we create a binary matrix $\mathbf{P}$ of size $K_{\mathcal{P}} \times J$, where each row represents which attribute is present in a rule. For instance, $P_{kj} = 1$ indicates that literal j is substantive in rule k, and $P_{kj} = 0$ otherwise. We also need to determine for each example, which of the $K_{\mathcal{P}}$ rules it satisfies. For a data set with N examples, we create a matrix $\mathbf{W}$ of size $N \times K_{\mathcal{P}}$, where the k-th element in the n-th row, ω_{nk}, indicates if the n-th observation satisfies rule k. Both matrices are pre-computed before the final step.

5.3.3 Rule Selecting

The previous two steps greatly reduce the computational load by feeding the final step with a set of high quality candidate rules. Now our goal is only to select an optimal set A from the candidate set $\mathcal{P}$. We formulate an ILP using the same objective (5.2), and present it below.

$$\min_{\xi_n, \zeta_k} \frac{1}{N} \sum_{n=1}^{N} \xi_n + C_1 \sum_{k}^{K_{\mathcal{P}}} \zeta_k l_k + C_2 \sum_{k}^{K_{\mathcal{P}}} \zeta_k$$

such that

$$\xi_n + \sum_{k=1}^{K_{\mathcal{P}}} \omega_{nk} \zeta_k \geq 1, \forall n \in \mathcal{I}^+ \tag{5.23}$$

$$K\xi_n \geq \sum_{k=1}^{K_{\mathcal{P}}} \omega_{nk} \zeta_k, \forall n \in \mathcal{I}^- \tag{5.24}$$

$$\xi_n, \zeta_k \in \{0, 1\}. \tag{5.25}$$

The length of rule k, l_k, can be pre-computed by $l_k = \sum_{j=1}^{J} P_{kj}$. Constraint (5.23) means that an error occurs for a positive example if it does not satisfy any rules that are selected. Constraint (5.24) means that an error occurs for a negative example if it satisfies at least one rule that is selected. This ILP only involves $\mathcal{O}(N)$ constraints and $\mathcal{O}(N) + \mathcal{O}(K_{\mathcal{P}})$ variables, which is much simpler than the MIP in an ORS framework.

The difference between ORS and ORSx method is that the latter avoids forming rules in the optimization process, by handling it to other efficient off-the-shelf algorithms. Separating the mining step from the optimization problem renders more control to users over the quality and size of desired rules. Users can manually modify the pre-mining and screening process by applying domain-specific minimum support, maximum length and secondary selection criteria.

5.4 Analysis on rules and ORS Models

In this section, we discuss the quality of rules in an rule set classifier. Certain properties of the rules improve computation complexity. We also show the VC dimension of rule set models and compare rule set classifiers with other discrete classifiers (decision trees and random forests). Due to the page limit, some proofs are provided in the supplementary material.

5.4.1 Bounds on rules

Define the *support set* of rule z over data set S as

$$\mathcal{I}^S(z) = \{X | z(X) = 1, X \in S\}, \tag{5.26}$$

and the *support* of rule z over S as

$$\text{supp}^S(z) = |\mathcal{I}^S(z)|. \tag{5.27}$$

$\text{supp}^{S^+}(z)$ is called the *positive support* of z, which is the number of positive examples in $\mathcal{I}^S(z)$, and $\text{supp}^{S^-}(z)$ is called the *negative support* of z, which is the number of negative

examples in $\mathcal{I}^S(z)$.

An rule set classifier is essentially an ensemble of weaker classifiers, rules. Including rules with a low quality is expensive, and as we will prove, unnecessary. First we show in Theorem 7 that the optimal solution never includes a rule with a high negative support.

Theorem 7 *Take an rule set model with regularization parameters C_1 and C_2. The rule set model is trained on a data set S, consisting of N examples, N^+ of which are positive examples. If $A^* \in \arg\min_A L(A)$, then for any $z \in A^*$, $supp^{S^-}(z) \leq N^+ - N(C_1 + C_2)$.*

Proof 8 *The objective function of the optimal solution A^* is*

$$\begin{aligned} L(A^*) &= \frac{N^+ - supp^{S^+}(A^*) + supp^{S^-}(A^*)}{N} + \\ & C_1 \#literals(A^*) + C_2 \#rules(A^*) \\ &\geq \frac{N^+ - supp^{S^+}(A^*) + supp^{S^-}(A^*)}{N} + C_1 + C_2 \\ &\geq \frac{supp^{S^-}(A^*)}{N} + C_1 + C_2. \end{aligned}$$

These inequalities become tight when A^* was one rule with one literal covers the whole positive class and some of the negative class. Let $\emptyset$ denote an empty set where there are no rules and all the data points are classified as negative, so the total number of errors are the number of positive data, denoted as N^+. The objective function given $\emptyset$ is

$$L(\emptyset) = \frac{N^+}{N}.$$

Since $A^* \in \arg\min_A L(A)$, $L(A^*) \leq L(\emptyset)$, then

$$supp^{S^-}(A) \leq N^+ - N(C_1 + C_2)$$

Since $I^{S^-}(A) = \cup_{a \in A} I_a^{S^-}$, then $supp^{S^-}(z) \leq supp^{S^-}(A^*)$, thus

$$supp^{S^-}(z) \leq N^+ - N(C_1 + C_2)$$

This means after rule mining, we can safely reduce the rule space by disregarding rules

with a negative support above $N^+ - N(C_1 + C_2)$. Similarly, we can also prove that if a rule has a low positive support, removing it always achieves a better objective. Let $A_{\setminus z}$ denote the rule set with rule z removed from A.

Theorem 8 *Take an rule set model with regularization parameters C_1 and C_2. The rule set model is trained on a data set S, consisting N examples. If $supp^{S^+}(z) \leq (C_1 + C_2)N$, then $L(A_{\setminus z}) \leq L(A)$.*

Proof 9 *The worst case when rule z is removed is when z is an accurate rule with confidence equal to 100%, i.e., all data points that satisfy rule z are positive; and the points covered by z are not covered by any other rule. Therefore once removing it, the number of errors increased by the positive support of z. On the other hand, removing z benefits the regularization terms, by decreasing the sum of rule lengths by at least 1, and the number of rules by 1. Then the objective function given $A_{\setminus z}$ obeys*

$$\begin{aligned} L(A_{\setminus z}) =& L(A) + \frac{\#error(A_{\setminus z}) - \#error(A)}{N} + \\ & C_1 \left(\#literals(A_{\setminus z}) - \#literals(A)\right) + \\ & C_2 \left(\#rules(A_{\setminus z}) - \#rules(A)\right) \\ \leq & L(A) + \frac{supp^{S^+}(z)}{N} - C_1 - C_2. \end{aligned}$$

In order to prove $L(A_{\setminus z}) \leq L(A)$, we need

$$L(A) + \frac{supp^{S^+}(z)}{N} - C_1 - C_2 \leq L(A),$$

i.e.,

$$supp^{S^+}(z) \leq (C_1 + C_2)\, N.$$

Proof 10 *Let $M^* = |A^*|$. A^* contains M^* rules where each rule has at least one literal. Therefore, the objective function given A^* is lower bounded by*

$$\begin{aligned} L(A^*) &\geq \frac{\#error(A^*)}{N} + C_1 M^* + C_2 M^* \\ &\geq M^* (C_1 + C_2). \end{aligned}$$

Since $A^* \in \arg\min_A L(A)$, $L(A^*) \leq L(\emptyset)$. *That is*

$$M^* (C_1 + C_2) \leq \frac{N^+}{N}.$$

Therefore

$$M^* \leq \frac{N^+/N}{C_1 + C_2}.$$

It means we need not bother mining rules of low positive support. Theorem 8 is a stronger statement than Theorem 7 since it provides a lower bound on positive support for rules in all rule sets and saying that removing a low supported rule always improves the performance; while Theorem 2 only applies to optimal solutions.

With the above theoretical guarantees, we know it is safe to reduce the rule space, by setting the minimum positive support to be $(C_1 + C_2) N$ when we pre-mine the rules and throwing away rules with negative support higher than $N^+ - N (C_1 + C_2)$ in the screening stage. This does not benefit an ORS framework as it directly forms rules. But it provides strong computational motivation for pre-mining rules in an ORSx framework.

The sparseness of a model is also associated with the number of rules in an rule set model. We prove in Theorem 9 that the number of rules in an optimal rule set is upper bounded.

Theorem 9 *Take an rule set model with regularization parameters* C_1 *and* C_2. *The rule set model is trained on a set of* N *examples,* N^+ *of which are positive examples. If* $A^* \in \arg\min_A L(A)$, *then* $|A^*| \leq \frac{N^+/N}{C_1+C_2}$.

This theorem is meaningful not only for showing the simplicity of the output, but also gives us a suggestion for K when we use the MIP in an ORS framework. Knowing that the optimal set can never be larger than $\frac{N^+/N}{C_1+C_2}$, we can safely set K to be $\frac{N^+/N}{C_1+C_2}$. The smaller K can be set, the better it is computationally for the MIP.

5.4.2 VC Dimension of an ORS classifier

Let us consider the VC dimension of hypothesis classes representing rule sets selected from a pre-mined set $\mathcal{P}$. There are some results for k-rule set [26] and monotone functions (that

is, Boolean functions that can be represented without negated literals) [70]. [53] has shown that the class of k-term monotone l-rule set formulas (i.e., with monomials containing at most l variables) has VC dimension at least $lk\lfloor\log(\frac{n}{m})\rfloor$, where $l \leq m \leq n$, and $k \leq \binom{m}{l}$. However, his theorem does not have the constraint that the rules come from a fixed rule set.

Let $\mathcal{S} = \mathbb{R}^J$ represent the complete set of all possible data that could be constructed from J attributes. To compute the VC dimension, we introduce the following definition.

Definition 8 *An **Efficient Set** of $\mathcal{P}$ is a set of rules where the support set of each rule is not a subset of the rest of the efficient set, i.e.,*

$$\mathcal{P}^E = \{z | z \in \mathcal{P}, \mathcal{I}^{\mathcal{S}}(z) \not\subset \mathcal{I}^{\mathcal{S}}(\mathcal{P}^E_{\setminus z})\}.$$

This means for any rule z in $\mathcal{P}^E$, there exists data points that satisfy only z and none of the rest of the rules in $\mathcal{P}^E$. We call the efficient set with the maximum number of rules the **Maximum Efficient Set** of $\mathcal{P}$, denoted as $\mathcal{P}^E_{\text{max}}$. We claim that the VC dimension of ORSx learned from $\mathcal{P}$ depends on the size of $\mathcal{P}^E_{\text{max}}$, stated as the following.

Theorem 10 *The VC dimension of an rule set classifier f built from $\mathcal{P}$ equals the size of the maximum efficient set of $\mathcal{P}$:*

$$VCdim(f) := |\mathcal{P}^E_{max}|.$$

Proof 11 First we prove that VCdim$(f) \geq |\mathcal{P}^E_{\text{max}}|$, which means there exists a set of $|\mathcal{P}^E_{\text{max}}|$ examples $X_1, ...X_{|\mathcal{P}^E_{\text{max}}|}$ that any labels $Y_1, ..., Y_{|\mathcal{P}^E_{\text{max}}|}$ can be realized by a classifier f built from $\mathcal{P}$. To construct this example set, we use the maximum efficient set $\mathcal{P}^E_{\text{max}}$. For any rule z_i in $\mathcal{P}^E_{\text{max}}$, since $\mathcal{I}^{\mathcal{S}}(z_i) \not\subset \mathcal{I}^{\mathcal{S}}(\mathcal{P}^E_{\text{max}} \setminus z_i)$, there always exists a data point $X_i \in \mathcal{S}$ that satisfies only z_i, i.e.,

$$X_i \in \mathcal{I}^{\mathcal{S}}(\mathcal{P}^E_{\text{max}} \setminus z) - \mathcal{I}^{\mathcal{S}}(z),$$

for $i \in \{1, ..., |\mathcal{P}^E_{\text{max}}|\}$. Each X_i is covered by exactly one rule in $\mathcal{P}^E_{\text{max}}$. These points can always be shattered since for any labels, we can from a rule set $A = \{z_i | z_i \in \mathcal{P}^E_{\text{max}}, \text{s.t. } \mathbb{1}_{z_i}(X_i) = 1, Y_i = 1\}$. Therefore, all possible labels of $Y_1, ..., Y_N$ can be realized, which means that VCdim$(f) \geq |\mathcal{P}^E_{\text{max}}|$.

Then we show VCdim$(f) \leq |\mathcal{P}^E_{\max}|$. We prove this by contradiction. Let $\mathcal{P}^E_{\max}$ be the maximum efficient set of $\mathcal{P}$. Assume there exists a set of h examples $X_1, ...X_h$ where $h > |\mathcal{P}^E_{\max}|$, and their labels $Y_1, ...Y_h$ can always be realized. Let $\mathbf{0}_{\setminus i}$ denote an all-zero vector of size h except a one at the i-th position. For $\mathbf{0}_{\setminus i}$ to be a realizable set of labels, there must exist a rule z_i that satisfies $\mathbb{1}_{z_i}(X_i) = 1$ and $\mathbb{1}_{z_i}(X_j) = 0$ for $j \neq i$. This should be true for all $i \in \{1, ..., h\}$. Therefore, there must exist h such rules that each of them covers a data point that only satisfies this rule. According to definition 8, this is equivalent to declaring that these h rules is an efficient set, and the size of the set is h, which is greater than $|\mathcal{P}^E_{\max}|$. This contradicts the assumption that $\mathcal{P}^E_{\max}$ is the maximum efficient set and should contain the largest number of rules. Therefore, VCdim$(f) \leq |\mathcal{P}^E_{\max}|$.

Thus, we conclude that the VC-dimension of a classifier f built from $\mathcal{P}$ is $|\mathcal{P}^E_{\max}|$. (Learning an efficient set will be another topic that we do not discuss in this chapter.)

5.4.3 Comparing with Other Discrete Classifiers

Like rule set classifiers, decision trees and random forests also discretize the input space and assign each subspace with a label. We prove that for these models, there always exist equivalent rule set classifiers. These theorems are simple, but may not be obvious to those who have not thought about it. We present the definition of two classifiers being *equivalent* below.

Definition 9 *Two classifiers f_1, f_2 are equivalent if for any input X, $f_1(X) = f_2(X)$.*

In a decision tree, the leaves divide up the input space into areas with different labels, which will be the predicted outcome for any data that ends up in that area. A path from the root to a leaf is a conjunction of literals, i.e., a rule. See Figure 5-1 as an example. The decision tree ends up with 4 leaves, and therefore 4 rules. To convert the tree into an equivalent rule set classifier, we simply collect the rules that are associated with positive leaves, in this case, leaf 4 and 6, shown in grey boxes.

Theorem 11 *For any decision tree f_1, there exists an equivalent rule set classifier f_2, where the number of rules in f_2 equals to the number of positive Y labels in f_1.*

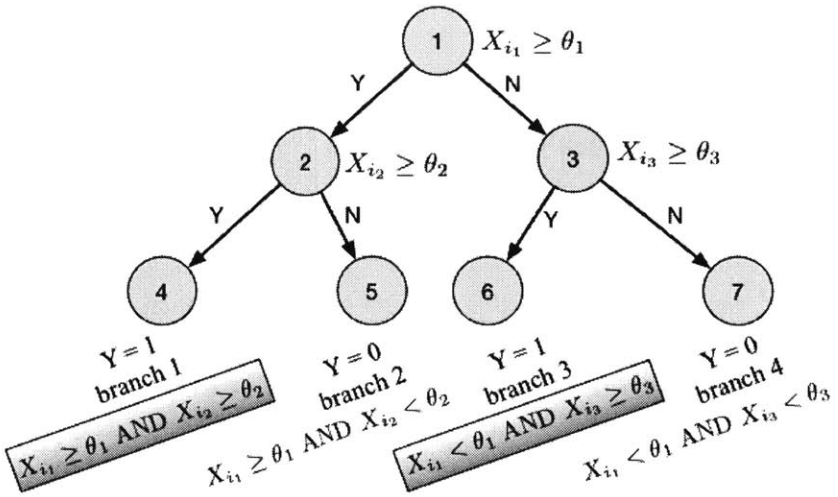


Figure 5-1: A decision tree and the corresponding rules.

Similar inductions hold for random forests. Random forest is an ensemble method based on decision trees. If an input data point falls into a positive leaf in at least half of all the trees, then it is labeled as positive. Therefore, the equivalent rule set classifier consists of rules that are conjunctions of positive rules from at least half of all the trees. We summarize the above statements into Theorem 12.

Theorem 12 *For any random forest f_1, there exists an equivalent rule set classifier f_2. If f_1 consists of K_{rf} decision trees, and the k-th tree has n_k positive leaves for $k \in \{1, ..., K_{rf}\}$, then the size of the rule set in f_2 is upper bounded by $\sum_{\pi \in \Pi} \prod_{k \in \pi} n_k$, where Π is a collection of all possible combinations of $\lfloor \frac{K_{rf}}{2} \rfloor + 1$ elements selected from $\{1, ..., K\}$.*

The size is upper bounded by instead of exactly equal to $\sum_{\pi \in \Pi} \prod_{k \in \pi} n_k$ because some rules could be equivalent, or contained in others. Note that we only need conjunction of $\lfloor \frac{K_{rf}}{2} \rfloor + 1$ rules because conjunctions of more than that are contained in conjunctions of exactly $\lfloor \frac{K_{rf}}{2} \rfloor + 1$ positive rules.

The above theorems provide theoretical guarantees that rule set classifiers can be as good as decision trees and random forests, in terms of predictive performance, although it might not be desired to create complex rule set models, since the whole purpose of designing an rule set model is to favor its interpretability over other models.

5.5 Experiments

Our experiments include applying rule set models to diagnose obstructive sleep apnea (OSA) and experimenting on 9 public datasets from UCI Machine Learning repository [51].

To construct simple rule set models for interpretability purposes, we set the maximum number of rules to be 5 in all experiments. (In an ORS framework, we set $K = 5$; in a ORSx framework, we add a constraint that the sum of ζ_k's is less than or equal to 5.) Since we placed strong restrictions on the characteristics of rule set models, we expect to lose predictive accuracy over unrestricted baseline methods. In many of the experiments we did, we found that rule set models do not lose in performance, and most of the time are the top performing models, while achieving a substantial gain in interpretability.

5.5.1 Diagnosing Obstructive Sleep Apnea

The main experimental result is an application of rule set models to build a diagnostic screening tool based on routinely available medical information. We analyzed polysomnography and self-reported clinical information from 1922 patients tested in the clinical sleep laboratory of Massachusetts General Hospital, first analyzed in 2015 [82, 83]. The goal is to classify which patients who enter into the MGH Sleep Lab have OSA, based on a survey filled out upon admission. We produce predictive models for OSA screening using attributes from self-reported symptoms and self-reported medical information. The attributes include detailed information such as age, sex, BMI, sleepiness, if the patient snores, if the patient wakes up during sleep, if the patient falls back to sleep easily, the level of tiredness, etc. The data set was binary coded into 112 attributes.

Due to the size of this dataset, we chose ORSx for faster computation. We mined rules with minimum support of 5% and maximum length of 3. We tuned parameters C_1 and C_2 using nested cross-validation to obtain the best performance, under the constraint that the rule size cannot exceed 5. We measured out-of-sample performance using accuracy from 5-fold cross validation for rule set models and 5 other methods that adhere to a certain level of interpretability, BRS [92], Lasso, C4.5, CART and RIPPER. For all baseline methods, we tuned the hyperparameters with grid search in nested cross validation. The results are

displayed in Table 5.1.

Table 5.1: Accuracy comparison for rule set models and baselines on obstructive sleep apnea dataset.

	Accuracy	Complexity
ORSx	.80(.01)	total number of rules = 2.8 average length of rules = 1.67 total number of literals = **4.7**
BRS	.80(.01)	total number of rules = 4 average length of rules = 1.75 total number of literals =**7**
RIPPER	.79(.01)	total number of rules = 4.6 average length of rules = 4 total number of literals = **18.4**
C4.5	.79(.01)	depth = 5 total number of nodes = **14.4**
CART	.79(.01)	depth = 5 total number of nodes = **11**
Lasso	.80(.01)	non-negative coefficients = **5**

To compare interpretability, we reported the complexity of each model averaged across 5 folds. For ORS and BRS models, we reported the total number of rules, average length of rules and the total number of literals. rule set models achieve the same performance as BRS models but with higher interpretability. This is due to a more flexible control over the size and shape of the rule set compared to BRS models. RIPPER models are decision lists, having a different form than Rule Sets. We reported the total number of rules, average length of rules and total number of literals. For decision trees C4.5 and CART, we reported the depth of a tree, and the total number of nodes in a tree. For lasso, we reported the number of non-negative coefficients. Since baseline models have different logical forms than rule set models, we compare one universal metric, the number of literals/nodes used in each model, marked in bold in Table 5.1. We find that rule set models used substantially

Table 5.2: Accuracy comparison for rule set models and baselines on UCI datasets.

	Data	*Interpretable Models*							*Uninterpretable Models*	
	Type	ORS	ORSx	BRS	Lasso	C4.5	CART	RIPPER	random forest	SVM
blogger	Categorical	.85(.11)	**.86(.10)**	.80(.04)	.81(.08)	.77(.05)	.78(.07)	.76(.06)	.82(.07)	.82(.10)
votes		**.98(.02)**	**.98(.02)**	.95(.02)	.96(.02)	.96(.02)	.96(.03)	.96(.01)	.95(.02)	.97(.02)
tic-tac-toe		**1(.00)**	**1(.00)**	**1(.00)**	.71(.02)	.92(.03)	.93(.02)	.98(.01)	.99(.00)	.99(.00)
monks1		**1(.00)**	**1(.00)**	**1(.00)**	.76(.02)	.90(.06)	.88(.07)	.94(.12)	**1(.00)**	**1(.00)**
bupa	Numerical	.65(.02)	.65(.03)	.66(.02)	.68(.04)	.63(.04)	.68(.03)	.65(.05)	.70(.03)	**.73(.04)**
transfusion		.78(.02)	**.80(.01)**	.77(.01)	.77(.02)	.76(.02)	.78(.02)	.78(.02)	.78(.02)	.80(.02)
banknote		.98(.01)	.97(.01)	.96(.01)	1(.00)	.90(.01)	.90(.02)	.91(.01)	.91(.01)	**1(.00)**
indian-diabetes		.73(.02)	**.77(.03)**	.74(.02)	.67(.01)	.66(.03)	.67(.01)	.67(.02)	.76(.02)	.69(.01)
heart	mixed	.80(.04)	.84(.05)	.83(.07)	.85(.04)	.76(.06)	.77(.06)	.78(.04)	.81(.06)	**.86(.06)**

fewer literals than all other models while achieving a competitive accuracy to all models.

An example of an rule set model is shown below.

if a patient satisfies (age $\geq$ 30 *AND* patient checked snoring as a potential symptom in the questionnaire),

OR (age $\geq$ 30 *AND* patient checked snoring as a reason for "why are you here" in the questionnaire),

OR (age $\geq$ 30 *AND* has hypertension),

OR (BMI $\geq$ 25) **then**

predict the patient has sleep apnea,

else

predict the patient does not have sleep apnea.

The model lists four rules to characterize patients that has sleep apnea. It is a sparse model with only a few attributes and a simple structure, and can potentially be used by people without a machine learning background.

5.5.2 Performance on UCI Datasets

We applied ORS and ORSx to several UCI datasets and compared with 5 previously mentioned interpretable models and 2 black box models, random forest and SVM. In the experimental set up, we set a time limit for the MIP in the ORS framework to ensure that it

returns a solution in a reasonable amount of time. Table 5.2 displays the mean and standard deviation of out-of-sample accuracy across 5 folds.

We observed that even with the severe restrictions, rule set classifiers achieve very competitive performance. For the four categorical datasets in Table 5.2, rule set classifiers always do better than other models. Especially for tic-tac-toe and monks, where there are correct models that correctly classify all examples, rule set models are able to discover the correct rules and achieve 100% accuracy. For numerical and mixed datasets, rule set models' performance levels are on par with those of other methods, sometimes slightly dominated by uninterpretable machine learning models.

We show an example of an rule set classifier learned from dataset "votes" using ORS framework. This data set includes votes for each of the U.S. House of Representatives Congressmen on 16 key votes on water project cost sharing, duty free exports, immigration, education spending, anti-satellite test ban and etc. The objective is to predict if the voter is democratic or republican.

if a voter (votes for eduction spending *AND* for physician fee freeze *AND* against water project cost sharing),

OR (votes for export administration act of South Africa *AND* for physician fee freeze *AND* agains synfuels corporation cutback),

OR (votes against aid to Nicaraguan Contrast *AND* against adoption of the budget resolution *AND* against handicapped infants and toddlers act *AND* against superfund right to sue),

OR (votes for adoption of the budget resolution *AND* for physician fee freeze *AND* agains synfuels corporation cutback),

OR (votes against adoption of the budget resolution *AND* for El Salvador aid *AND* for physician fee freeze),

OR (votes for aid to Nicaraguan Contras *AND* against adoption of the budget resolution *AND* against duty free exports *AND* against synfuels corporation cutback), **then**

predict the voter is republican,

else

predict the voter is democratic.

5.6 Conclusion

OA models have a long history. They are particularly useful as either (i) interpretable screening mechanisms, where they reduce much of the data from consideration from a further round of modeling, and (ii) consideration sets from marketing, which are rules that humans create to reduce cognitive load in order to make a decision.

We presented two optimization-based frameworks for learning Rule Sets. The first framework, ORS, uses a MIP to directly form rules from data. It can deal with both categorical and numerical data without pre-processing. The second framework ORSx reduces computation through pre-mining rules. We provided bounds on the support of rules that guarantee that the rule space can be safely reduced. Both methods can produce high quality rule set classifiers, as demonstrated through experiments. They achieve competitive performance compared to other classifiers, with a substantial gain in sparsity and interpretability.

One of the main benefits not discussed extensively earlier is the benefit of customizability. Because we use MIP/ILP technology, constraints of almost any kind are very easy to include, and we do not need to derive a new algorithm; this benefit does not come with any other technology that we know of. Customizability is an important component of interpretability.

Chapter 6

Causal Rule Set for Subgroup Identification

6.1 Introduction

In estimating causal effects, one important end goal is to identify a subgroup for which a treatment is effective, those that will have a good outcome only if they receive a treatment. We refer to such a group as *Effective class*. Identifying this subgroup is critical for better allocating a treatment at a budget, so that treatment is not distributed to those that have a good outcome anyways without a treatment or have a poor outcome given a treatment, which we call *Ineffective class*. For example, during a presidental election, given limited time and resource, it's more beneficial to run election campaigns in regions where the voters are mostly likely to be swayed instead of where the candidate will win the votes for sure or will lose the votes anyways with or without campaigns.

Subgroup studies have been of most interest in domains such as online advertising [49] and medical studies [23, 40, 54, 74]. However, subgroup analyses have often been used to compare treatment effectiveness across subgroups pre-determined by domain experts, especially in clinical research [5, 19]. Methods that identify naturally occurring subgroups who benefit most from the treatment have rarely been studied and we found only very recent work on this topic [46, 77]. These previous models, however, do not consider interpretability, which is of paramount importance in domains such as medical diagnosis and public

policy, where human, the decision makers, need to understand a model in order to trust and adopt it.

We want to develop a data-driven method to identify such subgroups and be able to characterize it in a human interpretable way. We introduce *Causal Rule Set* that i) classifies Effective/unenhanceble classes and ii) is interpretable for using Boolean expressions. An example of CRS is shown below.

if a customer (is older than 40 AND has children AND annual income <$50,000)
OR (is female AND married AND used the coupon before) **then**
the customer will purchase the product if and only if s/he receives the coupon.

The model works for scenarios such as those there is cost for sending coupons. Then decision makers would want to send to customers that meet the above conditions to enhance the treatment effect. They would not send to those who will buy the product anyways without a coupon, or those who would not buy the product in any circumstances; in either case, they lose money.

As shown in the above example, a CRS consists of *rules*, each rule being a conjunction of *conditions*, and claims the Effective class satisfies at least one of the rules. This structure of CRS makes it very easy to understand and use once the model is trained.

The CRS model is inspired by the previous work on Bayesian Rule Sets [92] in Chapter 4. BRS formulates the search for the "best" rule set as a MAP inference problem and uses simulated annealing which directly optimizes the posterior of models in the search space. A Bayesian Rule Set divides data into four subgroups defined by the predicted and observed outcomes (true positive, false positive, true negative and false negative) and the likelihood of data is modeled as being constant within each subgroup. This likelihood model works well for binary classification, but does not suffice for causal analysis however, since the outcomes need to be associated with the confounders, the treatment and the classifier we aim to learn. We apply Bayesian Logistic Regression to model the likelihood of the data where we assume the treatment effect remains constant and non-negative within the Effective class, and none for the ineffective class. We believe a Bayesian framework is appealing because it provides in a single unified framework a way to choose from different models and provide uncertainty estimates conditional on the choice of a single model.

To learn a CRS model, we apply simulated annealing for MAP inference over the search space. To make the search process tractable, we make an approximation that the CRS models consist of rules drawn from a pre-mined rule set that satisfy a minimum support and a maximum length (number of conditions in a rule). Simulated annealing searches over discrete state defined by a rule set and its corresponding optimal regression parameters.

We apply simulated annealing for MAP inference over the joint space of rule sets and regression parameters. To make the search process tractable, we make an approximation that the CRS models consist of rules drawn from a pre-mined rule set that satisfy a minimum support and a maximum length (number of conditions in a rule). Simulated annealing searches over discrete state defined by a rule set and its corresponding optimal regression parameters. Branch-and-bounding strategies are proposed to reduce the search space and guiding the sampling chain towards the MAP model, which improves the efficiency of the search algorithm.

The contributions of CRS are listed below.

- *Interpretability* Causality is of most interest to domain experts who want to understand the relations between different covariates and how they affect each other, as well as the mechanism that a desired outcome is achieved. Therefore, a model needs to serve the purpose of being easy to understand and being able explain the data or a mechanism.

- *Classifier for subgroups* CRS is a two-class classifier in a causal setting. The general framework allows various other forms of classifiers to replace CRS. If interpretability is not an concern, black-box classifiers such as SVM and random forests can be easily used in this framework, with minor modifications to the model.

The remainder of the chapter is organized as follows. In Section 6.2 we prepare readers with notations and preliminaries about causal analysis and Rule Sets models. In Section 6.3, we present the Bayesian framework for constructing CRS models. In Section 6.4 we present an inference algorithm that searches the joint rule set and parameter space for MAP models. In Section 6.5 we run simulations to prove the convergence of the solutions and show that the CRS model is able to identify the real positive subgroup under noise.

Finally in Section 6.6, we apply CRS to a realistic bank marketing data set to identify and characterize the subgroup of clients who are most likely to make a deposit after multiple advertising campaigns.

6.2 Notations and Preliminaries

We work with data $S = \{(X_i, Y_i, T_i)\}_{i=1,\ldots,n}$ where each observation consists of a J-dimensional covariate vector X_i, an observed outcome $Y_i \in \{0, 1\}$ and a treatment assignment indicator $T \in \{0, 1\}$. S is the union of the treatment group, S_{T}, and the control group, S_{C}.

We use the Rubin Causal Model framework [73] with the potential outcome $Y_i(0), Y_i(1) \in \{0, 1\}$ under treatment and control, respectively. Because each potential outcome can take on only two values, the unit-level causal effect the comparison of these two outcomes for the same unit involves one of four (two by two) possibilities:

1. $Y_i(1) = 1, Y_i(0) = 0$ - Effective Class: In this case, the outcome becomes positive only when it receives treatment.

2. $Y_i(1) = 0, Y_i(0) = 0$ - Ineffective Class: In this case, the outcome is always negative, which means the data point does not benefit from the treatment.

3. $Y_i(1) = 1, Y_i(0) = 1$ - Ineffective Class: In this case, the outcome is positive with and without a treatment, which means the treatment is not necessary and will not further enhance the outcome.

4. $Y_i(1) = 0, Y_i(0) = 1$: This case does not exist in our model since we assume the treatment always has a non-negative effect, which means it will not hurt the outcome. This is a not unrealistic assumption in general since treatments are usually selected by domain experts and they are designed to improve the outcome.

We define treatment effect as the difference in potential outcomes.

Definition 10 *The treatment effect on X_i is*

$$\tau(i) = Y_i(1) - Y_i(0) \tag{6.1}$$

Definition 11 *The average treatment effect (ATE) on a set of data indexed by I is*

$$\bar{\tau}(I) = \frac{1}{|\{i \in I : T_i = 1\}|} \sum_{\{i \in I : T_i = 1\}} Y_i - \frac{1}{|\{i \in I : T_i = 0\}|} \sum_{\{i \in I : T_i = 0\}} Y_i \tag{6.2}$$

According to their treatment effect, each data point can be classified into one of the two classes: *effective class* if $\tau(i) = 1$ and *ineffective class* if $\tau(i) = 0$. The two classes contain the following four situations base on their potential outcomes

Our goal is to find a rule set that classifies the two classes.

Our model also works for continuous outcomes if the user pre-determine a threshold to suit a domain-specific definition of "positive" and then binarize the outcomes. For example, Y_i could represent if a patient is cured given a medicine, or if a customer buys a product given a advertisements, etc. Then the treatment defined above has a slightly different meaning. It is rather an indicator of if the treatment is effective enough to bring the outcome to positive from negative, instead of absolute amount of change in the outcome.

Our model relies on three key assumptions.

Assumption 1 *Strong ignorability, which includes the following two assumptions*

- *(Unconfoundedness)* $(Y_i(0), Y_i(1)) \perp T_i | X_i$
- *(Overlap) For some* $\epsilon, \epsilon < P(T_i = 1 | X_i) < 1 - \epsilon$

Assumption 2 *The treatment effect is non-negative.*

Assumption 3 *The treatment effect is constant within each class.*

For the binary classifier in the model, we choose rule sets models for their interpretability. A rule set model consists of a set of rules that describe and discriminate a class from the rest of the data. It has been studied in [91, 92] and showed promising performance in predictive accuracy as well as the interpretability.

Now we introduce concepts and notations related with Srule set models. Let a represent a rule, and we use the same notation as a boolean function $a(X) \in \{0, 1\}$ to indicate if a data point X satisfies rule a. Indices of those that satisfy rule a are in the *support set* of rule a, denoted as $I(a)$.

$$I(a) = \{i | i \in \{1, ..., n\}, a(X_i) = 1\} \tag{6.3}$$

The *support* of a rule a in data S is the number of observations ins S that satisfy a, which is equal to the size of $I(a)$. We call observations that satisfy rule a and are also *treated* and *positive* the positive treated support set of a.

$$I_{\mathrm{T}}^{+}(a) = \{i | i \in I(a), Y_i = 1, T_i = 1\} \tag{6.4}$$

Given a set of rules in A, we can build a classifier denoted as $A(\cdot)$, which classifies an instance into positive (or negative, depending on different situations) when it satisfies at least one of the rules in A.

$$A(X) = \begin{cases} 1 & \exists a \in A, a(X) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

Here we will abuse the notation to allow A to represent both a rule set and a function evaluating if an instance satisfies this rule set. In the causal setting, A characterizes the minor class of effective and ineffective classes. We always choose A to represent the effective class in this chapter without loss of generality (it is true for most real applications).

6.3 Model

We use a Bayesian approach to characterize the posterior over CRS given training data S. The Bayesian framework contains three components, a prior for selecting a rule set A with parameters H, Prior(A), a prior for regression coefficients with parameters Prior($\mathbf{w}; \sigma$), and the likelihood of data.

6.3.1 Bayesian Logistic Regression

A Bayesian Logistic Regression model controls for confounding covariates, and models the impact of receiving the treatment and being in the effective class, giving the conditional likelihood:

$$p(Y_i = 1|X_i, T_i) = \text{Logistic}(w_0 + \sum_{j=1}^{J} w_j X_{ij} + \gamma T_i A(X_i) \tag{6.6}$$

depending on unknown parameters

$$A \sim \text{Prior}(A) \qquad \text{(a sparse rule set for the effective class)} \tag{6.7}$$

$$w_j \sim \mathcal{N}(0, \sigma_j), j = 0, ..., J \qquad \text{(regression coefficients)} \tag{6.8}$$

$$\gamma \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma) \qquad \text{(treatment effect rate, } \gamma \geq 0) \tag{6.9}$$

We will detail Prior(A) in the next section. The exponent in formula (6.6) models the synergy between the attributes X_i, treatment T_i and the rule set A. w_0 is the intercept, w_j represents the weight for each attribute contributing to the outcome and γ measures treatment effects for the effective class compared to the ineffective class. $w_0 + \sum_j^J w_j X_{ij}$ captures the base intensity of a patient to be cured, which is independent of if the patient receives a treatment. $\gamma T A(X)$ adds the effect of the treatment if X_i satisfies A. $\gamma T_i A(X_i)$ implies that the additive treatment effect exists only when the instance receives the treatment (i.e., $T_i = 1$), and it belongs to the effective class (i.e., $A(X_i) = 1$).

We write the Bayesian logistic regression in the following form,

$$\mathcal{B}(S; A, \mathbf{w}) = \text{Likelihood}(S; A, \mathbf{w}) + \text{Prior}(\mathbf{w}), \tag{6.10}$$

where $\mathbf{w} = \{\{w_j\}_{j=0}^J, \gamma\}$ and

$$\text{Likelihood}(S; A, \mathbf{w}) = \log \prod_{i=1}^{n} p(Y_i|X_i, T_i) \tag{6.11}$$

$$\text{Prior}(\mathbf{w}) = \sum_{j=0}^{J} \log p(w_j) + \log p(\gamma) \tag{6.12}$$

6.3.2 Generative Process for CRS

The generative process for rule set considers the interpretability of a rule, which is associated with the length of a rule (number of conditions in a rule). Therefore, we divide the rule space $\mathcal{A}$ into L pools, L being the maximum length the user allow.

$$\mathcal{A} = \cup_{l=1}^{L} \mathcal{A}^{[l]} \tag{6.13}$$

rules in the pool $\mathcal{A}^{[l]}$ have length l. rules are selected independently in each pool and we assume rules in $\mathcal{A}^{[l]}$ has probability p_l to be selected, which we place a beta prior on.

$$\text{Select rule } a_l \sim \text{Bernoulli}(p_l) \tag{6.14}$$

$$p_l \sim \text{Beta}(\alpha_l, \beta_l). \tag{6.15}$$

Let $M^{[l]}$ notate the number of rules in A that have length l. Then the probability of selecting a rule set A is

$$\begin{aligned}
\text{Prior}(A) &= \log p(A; \{\alpha_l, \beta_l\}_l) \\
&= \sum_{l=1}^{L} \log \int_{p_l} p_l^{M^{[l]}} (1-p_l)^{|\mathcal{A}^{[l]}| - M^{[l]}} \text{Beta}(p_l; \alpha_l, \beta_l) d(p_l) \\
&= \sum_{l=1}^{L} \log \left(\frac{\Gamma(\alpha_l + \beta_l)}{\Gamma(\alpha_l)\Gamma(\beta_l)\Gamma(|\mathcal{A}^{[l]}| + \alpha_l + \beta_l)} \cdot \Gamma(M^{[l]} + \alpha_l)\Gamma(|\mathcal{A}^{[l]}| - M^{[l]} + \beta_l) \right) \\
&= \sum_{l=1}^{L} \log \left(\theta_l \cdot h(M^{[l]}) \right),
\end{aligned} \tag{6.16}$$

where for future convenience we denote $\theta_l = \frac{\Gamma(\alpha_l+\beta_l)}{\Gamma(\alpha_l)\Gamma(\beta_l)\Gamma(|\mathcal{A}^{[l]}|+\alpha_l+\beta_l)}$, which is a constant once the prior parameters and rule space are given, and $h(M^{[l]}) = \Gamma(M^{[l]} + \alpha_l)\Gamma(|\mathcal{A}^{[l]}| - M^{[l]} + \beta_l)$ which is a function of $M^{[l]}$.

The parameters $\{\alpha_l, \beta_l > 0 | l \in \{1, ..., L\}\}$ on the priors control the expected number of rules of each length in the rule set. We usually choose $\alpha_l \ll \beta_l$ so that BOA tends to choose a smaller $M^{[l]}$ for each l.

6.3.3 Posterior

Notate $H = \{\mathcal{A}, L, \{\alpha_l, \beta_l\}_{l=1}^{L}, \{\sigma_j\}_{j=0}^{J}, \mu_\gamma, \sigma_\gamma\}$. Our goal is to find the optimal (A, w) that maximize the posterior $p(A, \mathbf{w}|H, S)$ which is equivalent to maximizing the following:

$$F(A, \mathbf{w}; S, H) = \mathcal{B}(S; A, \mathbf{w}) + \text{Prior}(A), \tag{6.17}$$

We write the objective as $F(A, \mathbf{w})$, omitting dependence on hyperparameters and data when appropriate.

6.4 Model Fitting

We now detail the solution space over which we maximize the objective $F(A, \mathbf{w})$, and the search algorithm for doing so. The search algorithm is developed from simulated annealing over joint solution space of rule sets and parameters. We propose strategies to improve the efficiency by i) finding upper bounds on the size of a MAP model and restrain the sampling chain within appropriate region; ii) early detecting bad rule sets and cutting bad branches from the search space; iii) finding lower bounds on the positive support of rules that can safely remove useless rules from the rule space.

We first describe the basic search procedure, and then introduce the three strategies we apply.

6.4.1 Search Procedure

For computational feasibility and interpretability purposes, the rule set space contains models where each rule is assumed to come from a pre-mined set of rules $\mathcal{A}$ that satisfy a minimum support proved in Theorem 15. There are many mature rule mining techniques in literature that can mine rules from a data set sufficiently fast. For this particular work, we used FP-Growth [9] that generated rules within a second for all the datasets we use in the experiment section. FP-Growth takes binary coded data where each column represents if the attribute in a data point satisfies a condition. which means if a continuous attribute is

within a range (for example, age is between 10 to 20), and if a categorical attribute equals a specific category (for example, the patient has hypertension).

Meanwhile, for a given model A, we need to find the optimal parameters that fit the data to A. Let $\mathbf{w}_A$ represent the optimal parameters that maximize the posterior. $\mathbf{w}_A$ is defined as

$$\mathbf{w}_A = \arg\max_{\mathbf{w}} F(A, \mathbf{w}). \tag{6.18}$$

Now we detail the search algorithm over the joint rule set space and parameter space for the optimal solution $\{A^*, \mathbf{w}_{A^*}\}$. The main procedure follows the steps of simulated annealing which generates a Markov Chain that starts from a random state, proposes the next state by randomly selecting from the neighbors, and converges to the optimal solution as the temperature cools down. In the context of our model, each state of the Markov Chain is defined as $s_t = (A_t, \mathbf{w}_A)$. A candidate for next state $(A_{t+1}, \mathbf{w}_{A_{t+1}})$ is proposed by first selecting from the neighbors of A_t to get A_{t+1}, and then maximizing the objective function to obtain $\mathbf{w}_{A_{t+1}}$. Given a temperature schedule function over time steps, $T(t) = T_0^{1-\frac{t}{N_{\text{iter}}}}$, the proposed state $(A_{t+1}, \mathbf{w}_{A_{t+1}})$ is accepted and assigned to s_{t+1} with probability $\min(1, \exp\{\frac{F(A_{t+1}, \mathbf{w}_{A_{t+1}}) - F(s_t)}{T(t)}\})$.

In summary, each proposal stage performs two steps: *proposing* and *fitting*.

1. Proposing: Given a rule set A_t, the neighbors of A_t are defined as rule sets that are "one-rule-different" from A_t. Therefore, a neighbor A_{t+1} is generated by either adding or removing a rule from A_t.

2. Fitting: Given the proposed rule set A_{t+1}, $\mathbf{w}_{A_{t+1}}$ is obtained via equation (6.18), setting $A = A_{t+1}$.

Thanks to the well-developed packages, the fitting step can be done very efficiently in practice.

Note that this search procedure involves exploring a solution space that increases exponentially with the number of rules, and the number of rules increases exponentially with the number of attributes. The complexity is too high to manage in practice. Therefore, we introduce the following strategies to apply to the algorithm, that reduces the problem

to a manageable size by reducing useless rules, and leading the sampling chain to more promising regions by deriving tighter bounds on the size of a MAP model as the search goes longer.

6.4.2 Branch-and-Bound Strategies

Before we continue, we prove two lemmas which will be used later. Following the definition of positive treated support set in (6.19), let $I_{\mathrm{T}}^{+}(S)$ notate the set of positive treated observations in S.

$$I_{\mathrm{T}}^{+} = \{i | i \in \{1, ..., |S|\}, T_i = 1, Y_i = 1\}. \tag{6.19}$$

The optimal parameter γ^* is bounded as below.

Lemma 2 *On data S, apply a causal OA model with parameters*

$$H = \{\mathcal{A}, L, \{\alpha_l, \beta_l\}_{l=1}^{L}, \{\sigma_j\}_{j=0}^{J}, \mu_\gamma, \sigma_\gamma\},$$

where $L, \{\alpha_l, \beta_l\}_l \in \mathbb{N}^+$, $\beta_l - \alpha_l > |\mathcal{A}^{[l]}| + 1$. If $(A^*, \mathbf{w}^*) \in \arg\min_{A,\mathbf{w}} F(A, \mathbf{w})$ and $\mathbf{w}^* = \{\{w_j^*\}_{j=0}^{J}, \gamma^*\}$, then

$$\frac{\log \max_l \frac{\beta_l}{|\mathcal{A}^{[l]}|+\alpha_l-1}}{|I_T^+|} < \gamma^* < \frac{\sigma_\gamma^2 |I_T^+|}{2}.$$

Proof 12 *The optimal function is written as*

$$F(A^*, \mathbf{w}^*) = \mathit{Likelihood}(S; A^*, \mathbf{w}^*) + \mathit{Prior}(\mathbf{w}^*) + \mathit{Prior}(A^*) \tag{6.20}$$

where the conditional likelihood $\mathit{Likelihood}(S; A^*, \mathbf{w}^*)$ is defined in equation (6.11), and reduced to

$$\mathit{Likelihood}(S; A^*, \mathbf{w}^*) = \sum_{i=1}^{n} (Y_i - 1) g(\mathbf{X}_n, T_i; A^*, \mathbf{w}^*) - \log\left(1 + e^{-g(\mathbf{X}_n, T_i; A^*, \mathbf{w}^*)}\right). \tag{6.21}$$

We compare $F(A^*, \mathbf{w}^*)$ with a suboptimal solution to derive the bound. Since $(A^*, \mathbf{w}^*)$ is the optimal solution, $F(A^*, \mathbf{w}^*)$ is always larger than or equal to the suboptimal objective

values.

Step 1: Upper bound γ^* *We first fix the optimal rule set A^*, but change the parameter $\mathbf{w}^*$ by a little bit and compare the new solution with the optimal solution. We define a new set of variables which take the coefficients from $\mathbf{w}^*$ except γ^*.*

$$\tilde{\mathbf{w}}^* = \{\{w_j^*\}_{j=0}^J, \gamma^* - \Delta_\gamma\}, \tag{6.22}$$

where we assume $0 < \Delta_\gamma < \gamma^*$. Since $(A^*, \tilde{\mathbf{w}}^*)$ is ssuboptimal, $F(A^*, \tilde{\mathbf{w}}^*) \leq F(A^*, \mathbf{w}^*)$. Combining with equation (6.20), we get

$$\begin{aligned} \textit{Likelihood}(S; A^*, \mathbf{w}^*) - \textit{Likelihood}(S; A^*, \tilde{\mathbf{w}}^*) &\geq \textit{Prior}(\mathbf{w}^*) - \textit{Prior}(\mathbf{w}) \\ &= \frac{2\gamma^*\Delta_\gamma - \Delta_\gamma^2}{\sigma_\gamma^2}. \end{aligned} \tag{6.23}$$

Now we upper bound the left-hand-side of (6.23).

$$\begin{aligned} &\textit{Likelihood}(S; A^*, \mathbf{w}^*) - \textit{Likelihood}(S; A^*, \tilde{\mathbf{w}}^*) \\ =& \sum_{i=1}^n (Y_i - 1)\left(g(\mathbf{X}_n, T_i; A^*, \mathbf{w}^*) - g(\mathbf{X}_n, T_i; A^*, \tilde{\mathbf{w}}^*)\right) - \log\left(\frac{1 + e^{-g(\mathbf{X}_n, T_i; A^*, \mathbf{w}^*)}}{1 + e^{-g(\mathbf{X}_n, T_i; A^*, \tilde{\mathbf{w}}^*)}}\right) \\ =& \sum_{i=1}^n (Y_i - 1)\Delta_\gamma T_i f_{A^*}(X_i) - \log\left(\frac{1 + e^{-g(\mathbf{X}_n, T_i; A^*, \tilde{\mathbf{w}}^*)} e^{-\Delta_\gamma T_i f_{A^*}(X_i)}}{1 + e^{-g(\mathbf{X}_n, T_i; A^*, \tilde{\mathbf{w}}^*)}}\right) \\ <& \sum_{i=1}^n Y_i \Delta_\gamma T_i f_{A^*}(X_i) \qquad (6.24) \\ \leq& \Delta_\gamma |I_T^+| \qquad (6.25) \end{aligned}$$

(6.24) follows because $\log\left(\frac{1+e^{-g(\mathbf{X}_n, T_i; A^*, \tilde{\mathbf{w}}^*)} e^{-\gamma^* T_i f_{A^*}(X_i)}}{1+e^{-g(\mathbf{X}_n, T_i; A^*, \tilde{\mathbf{w}}^*)}}\right)$ *achieves the minimum value $-\gamma^* T_i f_{A^*}(X_i)$ when $e^{-g(\mathbf{X}_n, T_i; A^*, \tilde{\mathbf{w}}^*)} \to \infty$. Joining (6.25) and (6.23), we have*

$$\Delta_\gamma |I_T^+| > \frac{2\gamma^*\Delta_\gamma - \Delta_\gamma^2}{\sigma_\gamma^2} \tag{6.26}$$

Since $\gamma^*, \Delta_\gamma \geq 0$,

$$\gamma^* < \frac{\sigma_\gamma^2 |I_T^+| + \Delta_\gamma}{2}. \tag{6.27}$$

In order for the above inequality to hold for any Δ_γ, *including when* $\Delta_\gamma \to 0$, *we need*

$$\gamma^* < \frac{\sigma_\gamma^2 |I_T^+|}{2}.$$

Step 2: Lower bound γ^* *Then we fix the optimal parameter* $\mathbf{w}^*$ *and change the rule set by removing a rule from it. We define* $A^*_{\setminus z}$ *as:*

$$A^*_{\setminus z} = \{a | a \in A^*, a \neq z\}.$$

Since $(A^*_{\setminus z}, \mathbf{w}^*)$ *is suboptimal to* $(A^*, \mathbf{w}^*)$, *then* $F(A^*_{\setminus z}, \mathbf{w}^*) \geq F(A^*, \mathbf{w}^*)$, *i.e.,*

$$\textit{Likelihood}(S; A^*, \mathbf{w}^*) - \textit{Likelihood}(S; A^*_{\setminus z}, \mathbf{w}^*) \geq \textit{Prior}(A^*_{\setminus z}) - \textit{Prior}(A^*; \mathbf{w}^*). \tag{6.28}$$

Now we upper bound the left-hand-side. Before that, we remind the readers of the definition of support set in (6.3) and introduce the following notation.

$$I(z \backslash A^*) = \{i | i \in \{1, ..., n\}, f_{A^*_{\setminus z}}(X_i) = 0, f_{A^*}(X_i) = 1\}, \tag{6.29}$$

which refers to indices of observations that do that satisfies rule z *but none of the rules in* A^*. *Then we decompose data* S *into* $\mathcal{I}^{z \backslash A^*}$ *and the rest.*

$$\begin{aligned}
&\textit{Likelihood}(S; A^*, \mathbf{w}^*) - \textit{Likelihood}(S; A^*_{\setminus z}, \mathbf{w}^*) \\
&= \sum_{i=1}^{n} (Y_i - 1) \left(g(\mathbf{X}_n, T_i; A^*, \mathbf{w}^*) - g(\mathbf{X}_n, T_i; A^*_{\setminus z}, \mathbf{w}^*) \right) - \log \left(\frac{1 + e^{-g(\mathbf{X}_n, T_i; A^*, \mathbf{w}^*)}}{1 + e^{-g(\mathbf{X}_n, T_i; A^*_{\setminus z}, \mathbf{w}^*)}} \right) \\
&= \sum_{i=1}^{n} (Y_i - 1) \gamma^* T_i \left(f_{A^*}(X_i) - f_{A^*_{\setminus z}}(X_i) \right) - \log \left(\frac{1 + e^{-g(\mathbf{X}_n, T_i; A^*, \mathbf{w}^*)}}{1 + e^{-g(\mathbf{X}_n, T_i; A^*_{\setminus z}, \mathbf{w}^*)}} \right) \\
&= \sum_{i \in I(z \backslash A^*)} (Y_i - 1) \gamma^* T_i - \log \left(\frac{1 + e^{-g(\mathbf{X}_n, T_i; A^*_{\setminus z}, \mathbf{w}^*)} e^{-\gamma^* T_i}}{1 + e^{-g(\mathbf{X}_n, T_i; A^*_{\setminus z}, \mathbf{w}^*)}} \right) + \sum_{i \notin I(z \backslash A^*)} 0
\end{aligned}$$

$$< \gamma^* \sum_{i \in I(z \backslash A^*)} Y_i T_i \tag{6.30}$$

$$\leq \gamma^* |I_T^+| \tag{6.31}$$

(6.30) follows because $\log\left(\frac{1+e^{-g(\mathbf{X}_n,T_i;A^*_{\backslash z},\mathbf{w}^*)}e^{-\gamma^* T_i}}{1+e^{-g(\mathbf{X}_n,T_i;A^*_{\backslash z},\mathbf{w}^*)}}\right)$ *achieves its minimum* $-\gamma^* T_i$ *when* $e^{-g(\mathbf{X}_n,T_i;A^*,\mathbf{w}^*)} \to \infty$.

Then we lower bound the right-hand-side of (6.28). Assume in A^*, $M^{[l]}$ *rules have length* l, *for* $l \in \{1, ..., L\}$. $A^*_{\backslash z}$ *consists of the same rules as* A, *missing one rule which we assume has length* l' *and therefore selected from* $\mathcal{A}^{[l']}$. *So for* $A^*_{\backslash z}$, $M_{l'} - 1$ *rules have length* l'. *From (4.24) in Chapter 4, we have*

$$\textit{Prior}(A^*_{\backslash z}) - \textit{Prior}(A^*) = \log \frac{|\mathcal{A}^{[l']}| - M_{l'} + \beta_{l'}}{M_{l'} + \alpha_{l'} - 1}. \tag{6.32}$$

$\frac{|\mathcal{A}^{[l']}| - M_{l'} + \beta_{l'}}{M_{l'} + \alpha_{l'} - 1}$ *decreases monotonically as* $M_{l'}$ *increases, therefore it is lower bounded at* $M_{l'} = |\mathcal{A}^{[l']}|$, *which leads to*

$$\textit{Prior}(A^*_{\backslash z}) - \textit{Prior}(A^*) \geq \log \frac{\beta_{l'}}{|\mathcal{A}^{[l']}| + \alpha_{l'} - 1} \textit{ for all } l' \in \{1, ..., L\}.$$

Therefore

$$\textit{Prior}(A^*_{\backslash z}) - \textit{Prior}(A^*) \geq \log \max_l \frac{\beta_l}{|\mathcal{A}^{[l]}| + \alpha_l - 1} \tag{6.33}$$

Combining (6.28) with (6.31) and (6.33) we get

$$\gamma^* > \frac{\log \max_l \frac{\beta_l}{|\mathcal{A}^{[l]}| + \alpha_l - 1}}{|I_T^+|}. \tag{6.34}$$

Joining inequalities (6.27) and (6.34), we proved

$$\frac{\log \max_l \frac{\beta_l}{|\mathcal{A}^{[l]}| + \alpha_l - 1}}{|I_T^+|} < \gamma^* < \frac{\sigma_\gamma^2 |I_T^+|}{2} \tag{6.35}$$

Given data S, we define

$$\bar{\mathcal{B}}(S) := \max_{\mathbf{w}} \mathcal{B}(S; \bar{A}, \mathbf{w}), \tag{6.36}$$

where $\bar{A}(\cdot)$ represents a rule set that only covers positive data, i.e., $\bar{A}(X_i) = Y_i$. We claim

Lemma 3

$$\mathcal{B}(S; A, \mathbf{w}) \leq \bar{\mathcal{B}}(S)$$

Proof 13 *To upper bound the Bayesian Logistic Regression on data S given A and $\mathbf{w}$, we divide S into control group S_C and treatment group S_T.*

$$\begin{aligned}\mathcal{B}(S; A, \mathbf{w}) &= \textit{Likelihood}(S; A, \mathbf{w}) + \textit{Prior}(\mathbf{w}) \\ &= \textit{Likelihood}(S_T; A, \mathbf{w}) + \textit{Likelihood}(S_C; A, \mathbf{w}) + \textit{Prior}(\mathbf{w}) \end{aligned} \tag{6.37}$$

First we upper bound Likelihood$(S_T; A, \mathbf{w})$ and since $T_i = 1, \forall i \in I_T$, the likelihood is upper bounded as following, assuming $\gamma \geq 0$,

$$\begin{aligned}&\textit{Likelihood}(S_T; A, \mathbf{w}) \\ =&\sum_{i \in I_T} \log P(Y_i | X_i, T_i; A, \mathbf{w}) \\ =&\sum_{i \in I_T} Y_i \log \frac{1}{1 + e^{-\left(w_0 + \sum_j^J w_j X_{ij} + \gamma A(X_i)\right)}} + (1 - Y_i) \log \frac{1}{1 + e^{\left(w_0 + \sum_j^J w_j X_{ij} + \gamma A(X_i)\right)}} \\ \leq&\sum_{i \in I_T} Y_i \log \frac{1}{1 + e^{-\left(w_0 + \sum_j^J w_j X_{ij} + \gamma\right)}} + (1 - Y_i) \log \frac{1}{1 + e^{\left(w_0 + \sum_j^J w_j X_{ij}\right)}} \\ =&\sum_{i \in I_T} Y_i \log \frac{1}{1 + e^{-\left(w_0 + \sum_j^J w_j X_{ij} + \gamma Y_i\right)}} + (1 - Y_i) \log \frac{1}{1 + e^{\left(w_0 + \sum_j^J w_j X_{ij} + \gamma Y_i\right)}} \\ =&\textit{Likelihood}(S_T; \bar{A}, \mathbf{w}) \end{aligned} \tag{6.38}$$

Substituting (6.38) into (6.37) gives

$$\begin{aligned}\mathcal{B}(S; A, \mathbf{w}) &\leq \textit{Likelihood}(S_T; \bar{A}, \mathbf{w}) + \textit{Likelihood}(S_C; \bar{A}, \mathbf{w}) + \textit{Prior}(\mathbf{w}) \\ &\leq \max_{\mathbf{w}} \mathcal{B}(S; \bar{A}, \mathbf{w}) \end{aligned} \tag{6.39}$$

Notice that for the control group, since $T_i = 0$, the additive term $\gamma T A(X_i)$ in Likelihood$(S_C; \emptyset, \mathbf{w})$ is always 0. Therefore it does not matter what rule set A it is, Likelihood$(S_C; \emptyset, \mathbf{w})$'s value depends only on the data S and parameters $\mathbf{w}$.

This lemma states that for any unknown rule set and the corresponding regression parameters, we can place an upper bound on $\mathcal{B}(S; A, \mathbf{w}_A)$.

Now we present the strategies we propose for boosting the search efficiency.

Smart Proposing

As shown in Section 4.4.3, randomly selecting a rule very inefficient search that it will take too many iterations to converge. For CRS model where each iteration includes a Bayesian Logistic Regression fitting that brings extra computation cost, the complexity will be too high for the algorithm to be practical. Therefore, we need a proposing strategy that finds good rules to perform adding or cutting and will converge faster without running Bayesian Logistic Regression many times. In Section 4.4.3, we do this by evaluate the objective value on all neighbors and then choose the best one. We do not want to use the same strategy here since evaluating a neighbor involves running a costly Bayesian Logistic Regression. However, in Lemma 3 show that the maximum likelihood is achieved when the rule set is $\bar{A}$ on the treated group. Therefore our goal is to create a rule set that covers the positive observations in the treated group. The control group will be omitted from the evaluation since any rule set does not make any difference.

For a neighbor A_{t+1}, we evaluate its precision on treated group, defined as

$$Q(A_{t+1}) = \frac{\sum_i \mathbb{1}(A_{t+1}(X_i) = 1, Y_i = 1, T_i = 1)}{\sum_i \mathbb{1}(Y_i = 1, T_i = 1)} \tag{6.40}$$

Then the algorithm proposes a neighbor by

- ADD:
  - With probability p, $\text{ADD}_{\text{random}}$.
  - With probability $1 - p$, draw a rule z,

$$z = \arg \max_{a \in \mathcal{A} \setminus A^t} Q(A^t \cup a).$$

  Then $A^{t+1} \leftarrow A^t \cup z$.

- CUT:
  - With probability p, $\text{CUT}_{\text{random}}$.
  - With probability $1-p$, draw a rule z,

$$z = \arg\max_{a \in A^t} Q(A^t \backslash a).$$

Then $A^{t+1} \leftarrow A^t \backslash z$.

Gravity towards Smaller Models

The Bayesian prior places a preferences on particular sizes of models (we choose the parameters so that smaller models are preferred over large models), which means the MAP solution is more likely to appear in certain regions than others. If we locate the more promising regions, we can restrain the search within a much smaller space and will therefore find the MAP models more quickly.

We wish to derive an upper bound on size of a MAP model; further more, as we obtain better solutions with the search, we would like the bound to become tighter so that the "appropriate" region becomes more defined as we get close to the MAP model.

Let v_t^* denote the maximum objective value till time t,

$$v_t^* = \max_{\tau \leq t} F(A_\tau, \mathbf{w}_{A_\tau}).$$

Let M_l^* denote the number of rules of length l in the MAP model A^*, $m_l(t)$ represent the upper bound on M_l^* till time t. Since the upper bound becomes tighter as the search goes, we have

$$m_l(0) \geq m_l(1) \cdots \geq m_l(t) \geq |M_l^*|.$$

$m_l(t)$ is updated via

Theorem 13

$$m_l(t) = \frac{\bar{\mathcal{B}}(S) + \textit{Prior}(\emptyset) - v_t^*}{\log\left(\frac{|\mathcal{A}^{[l]}|+\beta_l-1}{m_l(t-1)+\alpha_l-1}\right)}.$$

Proof 14 *Let $(A_t^*, \mathbf{w}_{A_t^*})$ represent the best solution till time t. Since $F(A^*, \mathbf{w}_{A^*}) \geq v_t^*$,*

$$\mathcal{B}(S; A^*, \mathbf{w}_{A^*}) + Prior(A^*) \geq v_t^* \tag{6.41}$$

where we proved in Lemma 3 that

$$\mathcal{B}(S; A^*, \mathbf{w}_{A^*}) \leq \bar{\mathcal{B}}(S).$$

and from (4.37) in the proof in Section 4, we have

$$Prior(A^*) \leq Prior(\emptyset) + M_l^* \log\left(\frac{m_l(t-1) + \alpha_l - 1}{|A^{[l]}| + \beta_l - 1}\right).$$

Combine the above inequalities together, we obtain

$$M_l^* \leq \frac{\bar{\mathcal{B}}(S) + Prior(\emptyset) - v_t^*}{\log\left(\frac{|\mathcal{A}^{[l]}| + \beta_l - 1}{m_l(t-1) + \alpha_l - 1}\right)} := m_l(t). \tag{6.42}$$

Prior($\emptyset$) is the log-probability of selecting an empty rule set. We observe that the smaller $\frac{\alpha_l}{\beta_l}$, the tighter the bound, which is consistent with the intuition of selecting rules from $A^{[l]}$. As simulated continues, v_t^* becomes smaller and smaller and the upper bound on M^* decreases accordingly.

At time 0, there is no prior information about the bound. We use an empty rule set as a benchmark, i.e.

$$\begin{aligned} v_t^* &= \mathcal{B}(S; \emptyset, \mathbf{w}_\emptyset) + \text{Prior}(\emptyset), \\ M_l^* &\leq |\mathcal{A}^{[l]}|, \end{aligned}$$

giving the Corollary,

Corollary 3 *On data S, apply a causal OA model with parameters*

$$H = \{\mathcal{A}, L, \{\alpha_l, \beta_l\}_{l=1}^L, \{\sigma_j\}_{j=0}^J, \mu_\gamma, \sigma_\gamma\},$$

where $L, \{\alpha_l, \beta_l\}_{l=1,\ldots L} \in \mathbb{N}^+$. *Define* $\{A^*, \mathbf{w}^*\} \in \arg\min_{A,\mathbf{w}} F(A, \mathbf{w})$, *and* $M^* := |A^*|$. *If* $\alpha_l < \beta_l$, *we have:*

$$M^* \leq \sum_{l=1}^{L} \sum_{l=1}^{L} \frac{\bar{\mathcal{B}}(S) - \mathcal{B}(S; \emptyset, \mathbf{w}_\emptyset)}{\log\left(\frac{|\mathcal{A}^{[l]}|+\beta_l-1}{|\mathcal{A}^{[l]}|+\alpha_l-1}\right)}.$$

This Corollary shows that if an empty set is a good approximation, i.e., $\mathcal{B}(S; \emptyset, \mathbf{w}_\emptyset)$ is close to $\bar{\mathcal{B}}(S)$, then M^* must be small. Corollary 3 is quite intuitive; if an empty set already achieves a good performance, then adding more rules will likely hurt the outcome.

This upper bounds help search procedure decide which action to take: the closer to the upper bound, the more often a CUT action should be selected instead of an ADD action, in order to reduce the model size. Now the proposing strategy becomes

- With probability $\frac{1}{2}e^{-\max(0, \frac{|A_t|-m^*}{s})}$, choose the ADD action
- With probability $1 - \frac{1}{2}e^{-\max(0, \frac{|A_t|-m^*}{s})}$, choose the CUT action

The bigger the size of the current model A_t, the smaller the probability of choosing an ADD action. This bound drags the sampling chain towards smaller models and helps find the MAP model much quicker in practice.

Early bounding

We apply the second strategy that detects if a rule set is not a subset of a MAP model, which tells the algorithm to stop exploring with this rule set. To do this, we keep a blacklist which contains sets of rules that optimal solutions would never contain, therefore once we reach a model that is on the list, we know we should not continue with this branch and should take the action "CUT".

Consider a set of rules A'. If the best possible rule set containing A' cannot beat the best rule set we have found so far, then we know that any rule set containing A' is suboptimal. In that case, we should stop exploring any models containing A'. To compete the best possible objective value, we divide S into two subsets, one that satisfies A', denoted as $S_{I(A')}$, one does not $S_{\neg I(A')}$, and define $\Upsilon(A')$ as:

$$\Upsilon(A') = \log \text{Likelihood}(S_{I(A')}; A', \mathbf{w}) + \bar{\mathcal{B}}(S_{\neg I(A')}) + \text{Prior}(A').$$

We claim

Theorem 14 *For a rule set A', if $\Upsilon(A') < v_t^*$, $A' \subset A_x$, then $A_x \notin \arg\max F(A, \mathbf{w})$.*

Proof 15 *In order to prove $A_x \notin \arg\min F(A, \mathbf{w})$, we need to show that $\exists A^*, \mathbf{w}_{A^*}$, such that $F(A_x, \mathbf{w}_{A_x}) < F(A^*, \mathbf{w}_{A^*})$. Now we lower bound $F(A_x, \mathbf{w}_{A_x})$.*

$$
\begin{aligned}
& F(A_x, \mathbf{w}_{A_x}) = \mathcal{B}(S; A_x, \mathbf{w}_x) + \textit{Prior}(A_x) \\
&= \log \textit{Likelihood}(I^S_{\neg A'}; A_x, \mathbf{w}_{A_x}) + \Big(\log \textit{Likelihood}(I^S_{A'}; A_x, \mathbf{w}_{A_x}) + \textit{Prior}(\mathbf{w}_{A_x}) \Big) + \textit{Prior}(A_x) \\
&\leq \max_{\mathbf{w}} \log \textit{Likelihood}(I^S_{\neg A'}; A_x, \mathbf{w}) + \max_{\mathbf{w}} \mathcal{B}(I^S_{A'}; A_x, \mathbf{w}) + \textit{Prior}(A') \\
&= \Upsilon(A') \qquad (6.43)
\end{aligned}
$$

Since $\Upsilon(A') \leq v_t^*$ and we also have $v_t^* \leq v^*$, then we get

$$F(A_x, \mathbf{w}_{A_x}) < v^*,$$

which means $A_x \notin \arg\min F(A, \mathbf{w}_A)$.

Theorem 14 states that for a rule set containing A', if the upper bound on the objective value, $\Upsilon(A')$, is smaller than the objective value of the best rule set we have seen so far, then A' cannot lead to a MAP solution and we should not continue searching with A'. This theorem is implemented in our code in the following way: in each iteration, when the action "ADD" is selected (which indicates generating a new rule set containing the current model A_t) and A_t is not on the blacklist, then A_t is checked against the bound of Theorem 14. If the condition $\Upsilon(A_t) < v_t^*$ holds, then the action "ADD" is not pursued with A_t and A' is added to the blacklist. Theorem 14 provides a substantial computational speedup in finding high quality or optimal solutions.

Reduce Solution Space

The previous two strategies have focused on bounding the sampling chain within an appropriate region. Now we present the third strategy that removes bad rules from the rule space and therefore directly reduces the number of solutions to explore.

Due to the prior we construct to favor smaller models, each rule needs to "cover" enough positive treated instances in order to be selected into a MAP solution. This eliminates an enormous number of rules that do not satisfy the minimum threshold and can be excluded from the rule space, thus decreasing the search space substantially. We must then locate the threshold and prove that, under weak conditions, the MAP model found from the reduced rule space is equivalent to a globally MAP model considering all rules in $\mathcal{A}$.

We prove that for any rule in a MAP model, it has to satisfy a minimum threshold on the positive treated support.

Theorem 15 *On data S, apply a causal OA model with parameters*

$$H = \{\mathcal{A}, L, \{\alpha_l, \beta_l\}_{l=1}^{L}, \{\sigma_j\}_{j=0}^{J}, \mu_\gamma, \sigma_\gamma\},$$

where $L, \{\alpha_l, \beta_l\}_l \in \mathbb{N}^+$, $\alpha_l < \beta_l$. Define $\{A^*, \mathbf{w}^*\} \in \arg\min_{A,\mathbf{w}} F(A, \mathbf{w})$. Then $\forall z \in A^*$,

$$|I_T^+(a)| > \frac{|I_T^+| \max_l \log\left(\frac{|\mathcal{A}^{[l]}|-m_l+\beta_l}{m_l-1+\alpha_l}\right)}{\max_l \log \frac{\beta_l}{|\mathcal{A}^{[l]}|+\alpha_l-1}},$$

where m_l is the upper bound for the number of rules of length l.

Proof 16 *We prove that under weak conditions, an optimal rule set A^* does not contain rules that have the positive treated support below a threshold. This is equivalent of proving that removing such a rule always improves the objective value. We will show that if any rule z, $z \in A^*$, has positive treated support $supp^{I_T^+}(z) < C$ on data S, then removing it will obtain a new rule set $A^*_{\setminus z}$ such that*

$$F(A^*_{\setminus z}, \mathbf{w}_{A^*_{\setminus z}}) > F(A^*, \mathbf{w}_{A^*}). \tag{6.44}$$

In order to do this, we prove a stricter statement

$$F(A^*_{\setminus z}, \mathbf{w}_{A^*}) > F(A^*, \mathbf{w}_{A^*}),$$

which is

$$\textit{Likelihood}(A^*_{\setminus z}; \mathbf{w}_{A^*}) - \textit{Likelihood}(A^*; \mathbf{w}_{A^*}) > \textit{Prior}(A^*) - \textit{Prior}(A^*_{\setminus z}). \tag{6.45}$$

Our goal is to find conditions that the above inequality holds. To do that we lower bound the left-hand-side and upper bound the right-hand-side, and then force the inequality hold. We now do the proceeding three steps.

Step 1: Lower bound the left-hand-side *From (6.31), we have*

$$\textit{Likelihood}(A^*_{\setminus z}; \mathbf{w}_{A^*}) - \textit{Likelihood}(A^*; \mathbf{w}_{A^*}) > -\gamma^* \cdot |I^+_T(z)|. \tag{6.46}$$

Step 2: Upper bound the right-hand-side *Assume A^* consists of M rules, among which $M^{[l]}$ come from pool $\mathcal{A}^{[l]}$ of rules with length l, $l \in \{1, ...L\}$, and the z-th rule has length l' so it would be removed from $\mathcal{A}^{[l']}$. From (6.32), we have*

$$\textit{Prior}(A^*) - \textit{Prior}(A^*_{\setminus z}) = \log\left(\frac{M_{l'} - 1 + \alpha_l}{|\mathcal{A}^{[l]}| - M_{l'} + \beta_l}\right). \tag{6.47}$$

(6.47) follows because $\frac{M_{l'}-1+\alpha_l}{|\mathcal{A}^{[l]}|-M_{l'}+\beta_l}$ increases monotonically as $M_{l'}$ increases, and achieves the maximum at $M_{l'} = m_{l'}$ which is the upper bound on $M_{l'}$ in Lemma 2.

Step 3: Combine Step 1 and Step 2. *In order to get inequality (6.45), we need*

$$-\gamma^* \cdot |I^+_T(z)| \geq \log\left(\frac{M_{l'} - 1 + \alpha_l}{|\mathcal{A}^{[l]}| - M_{l'} + \beta_l}\right),$$

which means

$$|I^+_T(z)| \leq \frac{\log\left(\frac{|\mathcal{A}^{[l]}|-M_{l'}+\beta_l}{M_{l'}-1+\alpha_l}\right)}{\gamma^*}.$$

This inequality must hold for any $l' \in \{1, ..., L\}$, therefore it holds for $\max_l \log\left(\frac{|\mathcal{A}^{[l]}|-M_l+\beta_l}{M_l-1+\alpha_l}\right)$.

It also must be true for $\gamma^* \in \left(\frac{\log \max_l \frac{\beta_l}{|\mathcal{A}^{[l]}|+\alpha_l-1}}{|I_T^+|}, \frac{\sigma_\gamma^2 |I_T^+|}{2} \right)$ *from Lemma 2. Therefore*

$$|I_T^+(z)| \leq \frac{|I_T^+| \max_l \log \left(\frac{|\mathcal{A}^{[l]}|-m_l+\beta_l}{m_l-1+\alpha_l} \right)}{\max_l \log \frac{\beta_l}{|\mathcal{A}^{[l]}|+\alpha_l-1}}. \tag{6.48}$$

The bound depends explicitly on the prior parameters and the number of positive treated data points in S. The reason that the size of a MAP solution is bounded is that the Bayesian structure places a prior that favors smaller models ($\alpha_l < \beta_l$). This means we need to search within rules that satisfy this minimum threshold, which makes the computation much more tractable in practice.

Depending on when we would like to apply this theorem, m_l could take on values from either Corollary 3, so that we can remove rules once before the search algorithm, or in Theorem 13 so that we keep reducing the rule space as the search goes.

Depending on the number of rules, sometimes at the beginning of the search, it is necessary to apply a secondary criteria, as Section 4 did, to further select a smaller set of rules. A good criteria we use in practice is the empirical average treatment effect for each rule.

Now we present the full algorithm as below.

6.5 Simulation Studies

We show that for simulated data generated by a known CRS model, our search algorithm recovers the model with high probability. If distributing the treatment according to the recovered rule set, we gain higher average treatment effect.

Given observations with arbitrary features $\{X_i\}_{i=1}^n$, and a collection of rules $\mathcal{A}$ on those attributes, we generate the treatment $\{T_i\}_{i=1}^n$ and labels $\{Y_i\}_{i=1}^n$ by the following steps.

1. Generate $Y_i(0)$ with a logistic regression with randomly generated parameters $\{w_j\}_{j=0,\ldots,J}$

$$Y_i(0) = \begin{cases} 1 & \text{Logistic}(w_0 + \sum_{j=1}^J w_j X_{ij}) \geq 0.5 \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 4 Search Algorithm for CRS

procedure CRS(S,H)
  $\mathcal{A} \leftarrow$ FP-Growth
  $A_0 \leftarrow$ a set of rules randomly selected from $\mathcal{A}$
  $\mathbf{w}_{A_0} = \arg\min_{\mathbf{w}} F(A_0, \mathbf{w})$
  $v^* = F(A_0, \mathbf{w}_{A_0})$
  blacklist $= \emptyset$
  for $t = 0, ..., \tau$ **do**
    if $A_t \in$blacklist **then**
      $A_{t+1} = \text{CUT}(A_t)$
    else
      $$A_{t+1} = \begin{cases} \text{CUT}(A_t) \text{ with probability } 1 - \frac{1}{2}e^{-\max(0,|A_t|-M_t^*)} \\ \text{ADD}(A_t) \text{ with probability } \frac{1}{2}e^{-\max(0,|A_t|-M_t^*)} \end{cases}$$
    if $\Upsilon(A_{t+1}) \geq v^*$ **then**
      blacklist = blacklist $\cup A_{t+1}$
    else
      $\mathbf{w}_{A_{t+1}} = \arg\min_{\mathbf{w}} F(A_{t+1}, \mathbf{w})$
      if $F(A_{t+1}, \mathbf{w}_{A_{t+1}}) \geq v^*$ **then**
        $v^* = F(A_{t+1}, \mathbf{w}_{A_{t+1}})$
        $A^* = A_{t+1}$
      $(A_{t+1}, \mathbf{w}_{A_{t+1}}) = (A_t, \mathbf{w}_{A_t})$ with probability $\exp\left(\frac{F(A_{t+1},\mathbf{w}_{A_{t+1}})-F(A_t,\mathbf{w}_{A_t})}{T(t)}\right)$
  end for
  return A^*
end procedure

2. For each observation X_i, set $T_i = 1$ with probability $\frac{1}{2}$.

3. Select 5 rules A from $\mathcal{A}$. Set $Y_i(1) = 1$ if $A(X_i) = 1$, otherwise $Y_i(1) = 0$.

4. For each observation X_i, set the label

$$Y_i = (1 - T_i)Y_i(0) + T_iY_i(1)$$

.

We follow the above procedure and create simulated data by setting the number of observations $n = 100,000$, and number of attributes, $J = 15$. We randomly split the data into 50% of training data and 50% of test data. On the training data, we first mine rules with a minimum support of 5% and $L = 3$, then use average treatment effect as a secondary criteria to select the best 1000 rules as our search space $\mathcal{A}$. Finally run simulated annealing till

50 steps. The experiments are repeated 100 times. We record the training error at specified steps, obtained by comparing predicted Effective and Ineffective classes with the observed class labels and plot the mean of error in Figure 6-1 with the red curve.

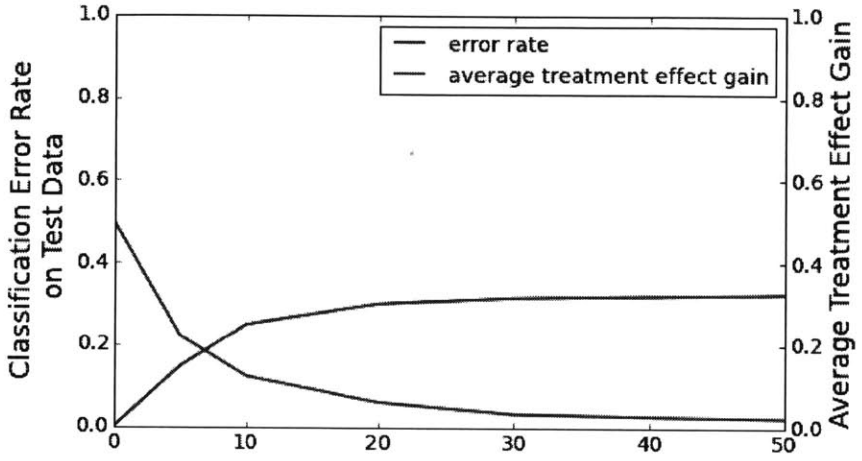


Figure 6-1: Classification error and average treatment effect gain.

We are also interested to know if we distribute the treatment according to the recovered rule set, how much treatment effect gain we would get.

Let $A_{\text{recovered}}$ denote the rule set discovered by our search procedure. Then we locate a set of observations from the test data that satisfy $A_{\text{recovered}}$, denoted by $I(A_{\text{recovered}})$. The average treatment effect on $I(A_{\text{recovered}})$ is denoted as τ_{selected}, computed by randomly assigning half of the observations in $I(A_{\text{recovered}})$ with treatment, the other half with control, then applying Definition (6.2). We follow the similar steps for evaluating average treatment, on a randomly selected set of observations of the same size as $I(A_{\text{recovered}})$ and compute τ_{random} The average treatment effect gain is computed by taking the difference of the two.

$$\Delta\tau = \tau_{\text{selected}} - \tau_{\text{random}}$$

We record the average treatment effect gain and plot the mean with a red blue in Figure 6-1.

6.6 Experiments

We apply CRS to a bank marketing data set [61] related with direct marketing campaigns (phone calls) of a Portuguese banking institution. Companies use direct marketing when targeting segments of customers by contacting them to meet a specific goal. This experiment studies real data collected from a Portuguese retail bank, from May 2008 to June 2013, in a total of 52,944 phone contacts. The dataset is unbalanced, as only 6557 (12.38%) records are related with successes. Each record contains 17 input features that include 9 numeric attributes (e.g. age, number of contacts performed before this campaign for this client, and last contact duration,) and 10 categorical attributes (e.g. if the person have personal loan, contact communication type, and outcome of previous marketing campaign.) The prediction target is a binary outcome that if the client subscribed a term deposit. Among the features, we consider "if the client was contacted before this campaign" as a treatment and our goal is to find a set of rules that characterize a subgroup of clients that will only make a deposit if s/he is contacted multiple times.

Because the rule miner we use accepts binary features only, we first normalize the numeric features and then discretize each of them into 5 quantiles. Together with categorical attributes, we obtain $J = 89$ binary features, so that each $X_i \in \{0,1\}^{89}$. We then mine all rules with a minimum support and at most 4 conditions, to obtain a few thousands of rules in each fold. The minimum support is selected from $\{5, 10, 15, 20\}$ to obtain different rule sets.

Regarding priors, we set priors on selecting rules as $\alpha_l = 1, \beta_l = |A_S^{[l]}|$ for all l, so that small models are favored. We set the variance of the coefficients in the logistic regression $\sigma = 1$. With this prior, we ran simulated annealing for 50 steps, with a temperature starting $T_0 = 1000$ and decays exponentially with each step. The search initializes with a random rule set of a size uniformly drawn from 1 to 8.

To evaluate the performance of CRS, we compute the treatment effect for clients that satisfy the rule set generated by our model using (6.2). We compare the average treatment effect on clients selected by the optimal rule set, with the average treatment effect on the entire test data data, which is the default state where no selection is conducted before

Table 6.1: MAP CRS models on 5 folds for bank marketing data set

fold	MAP model	τ_{selected}	$\bar{\tau}$
1	call duration $\geq$ 146 AND previous outcome = successful AND number of employees $<$ 5099 AND day of week $\neq$ Monday	.31	.16
2	call duration $\geq$ 146 AND number of employees $<$ 5099 OR call duration $\geq$ 220 AND default = No	.29	.19
3	call duration $\geq$ 146 AND number of employees $<$ 5191 AND job $\neq$ blue-collar AND previous outcome = successful	.43	.20
4	call duration $\geq$ 222 AND euribor 3 month rate $<$ 1.9 AND consumer price index $\geq$ 93.9 AND previous outcome = successful	.25	.17
5	call duration $\geq$ 222 AND number of employees $<$ 5099 AND previous outcome = successful OR day of week $\neq$ Monday AND call duration $\geq$ 146 AND number of employees $<$ 5099 AND previous outcome = successful OR consumer price index $\geq$ 93.44 AND call duration $\geq$ 222 AND month = July AND euribor 3 month rate $<$ 1.3	.29	.17
Mean		.31(.06)	.18(.01)

assigning the treatment.

We display in Table 6.1 a rule set with the highest posterior in an experiment. We report the average treatment effect (ATE) on the support set of the rule set and the ATE on the entire test data. The average treatment effect on test data without selecting a subgroup is 0.18 while distributing to observations captured by CRS achieves mean ATE of 0.31 over 5 folds.

6.7 Conclusion

We present a new class of interpretable models for identifying real beneficiaries from a treatment that could potentially have a major advantage in resource-constrained environments. The specific form of CRS can directly inform decision makers the characteristics of

subgroups of their interest, which is critical in their decision-making process.

We use a Bayesian framework to learn CRS models so that the user can influence the interpretability of the output model by choosing appropriate parameters. When searching for the MAP model, the inference algorithm applies different strategies to improve search efficiency and reduce computation complexity.

We envision models produced by CRS can be used in making medical decision, running drug trials, experimenting new policies, studying customers, etc.

Chapter 7

Conclusions and Outlook

This thesis develops methods and theory for detecting patterns that are defined by subsets of observations or subsets of features.

The first type of patterns is defined by subsets of observations. To detect the patterns, an algorithm needs to first discover why the observations are "similar" by learning from all patterns the pattern-general similarity. It then learns from individual patterns the pattern-specific similarity to distinguish one pattern from another.

The two algorithms we proposed for detecting crime series can be directly applied to cybercrimes where again, like the housebreaks, there is usually missing suspect information. Besides crime data mining, the algorithms could potentially be applied to various other fields as well. One possible application is in detecting disease outbreaks: patients infected with the same disease display similar symptoms and they will, like serial crimes, concentrate in time and space. Another interesting application is to detecting patterns of consumers based on their transaction history. Similar to detecting crime series where we do not know which attributes link the crimes together, we do not know which items or purchases are important "defining features", that link the consumers to form a pattern and also distinguish them from other patterns.

The second type of patterns is defined by subsets of features. We study a specific type of models, Rule Sets, and their application in binary classification and subgroup identification in causal analysis. Rule sets models have the advantage of being interpretable while having high predictive power. For future work, we want to apply rule set models for interpretable

clustering, so that each cluster is labeled with a set of rules that characterize the main features. We also want to extend the Causal Rule Set model to handle more general cases with continuous outcomes and/or multiple treatments.

We envision that interpretable machine learning models including Rule Sets models, will have broader application in social sciences where domain experts care about the transparency and understandability as well as, if not more than, the predictive power of models. Maintaining both the interpretability and predictive power, or at least providing the user the option to trade-off between them to suite domain-specific needs, will become a more and more desirable feature of machine learning models.

Bibliography

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Proc of the 2008 ACM Conf on Rec Systems*, RecSys '08, pages 335–336, New York, NY, USA, 2008. ACM.

[3] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, volume 1215, pages 487–499, 1994.

[4] Hiva Allahyari and Niklas Lavesson. User-oriented assessment of classification model understandability. In *SCAI*, pages 11–19, 2011.

[5] Susan F Assmann, Stuart J Pocock, Laura E Enos, and Linda E Kasten. Subgroup analysis and other (mis) uses of baseline data in clinical trials. *The Lancet*, 355(9209):1064–1069, 2000.

[6] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[7] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[8] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *International Conference on Machine Learning*, pages 19–26, 2002.

[9] Christian Borgelt. An implementation of the fp-growth algorithm. In *Proc of the 1st interl workshop on open source data mining: frequent pattern mining implementations*, pages 1–5. ACM, 2005.

[10] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the SIAM Conference on Data Mining (SDM)*, pages 243–254, 2008.

[11] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *ACM SIGMOD Record*, volume 26 (2), pages 265–276. ACM, 1997.

[12] Donald E. Brown and Stephen Hagen. Data association methods with applications to law enforcement. *Decision Support Systems*, 34(4):369–378, 2003.

[13] Anna L Buczak and Christopher M Gifford. Fuzzy association rule mining for community crime pattern discovery. In *ACM SIGKDD Workshop on Intelligence and Security Informatics*, 2010.

[14] Guoqing Chen, Hongyan Liu, Lan Yu, Qiang Wei, and Xing Zhang. A new approach to classification based on association rule mining. *Decision Support Systems*, 42(2):674–689, 2006.

[15] Hsinchun Chen, W. Chung, J.J. Xu, G. Wang, Y. Qin, and M. Chau. Crime data mining: a general framework and some examples. *IEEE Computer*, 37(4):50–56, 2004.

[16] Hong Cheng, Xifeng Yan, Jiawei Han, and Chih-Wei Hsu. Discriminative frequent pattern analysis for effective classification. In *ICDE*, pages 716–725. IEEE, 2007.

[17] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian cart model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

[18] William Cohen. Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning*, pages 115–123, 1995.

[19] David I Cook, Val J Gebski, and Anthony C Keech. Subgroup analysis in clinical trials. *Medical Journal of Australia*, 180(6):289, 2004.

[20] Criminal Justice Policy Research Institute. Residential burglary in Portland, Oregon. Hatfield School of Government, Criminal Justice Policy Research Institute, `http://www.pdx.edu/cjpri/time-of-dayday-of-week-0`.

[21] Kamal Dahbur and Thomas Muscarello. Classification system for serial criminal patterns. *Artificial Intelligence and Law*, 11(4):251–269, 2003.

[22] Robyn M Dawes. The robust beauty of improper linear models in decision making. *American psychologist*, 34(7):571–582, 1979.

[23] Marie Diener-West, Thomas W Dobbins, Theodore L Phillips, and Diana F Nelson. Identification of an optimal subgroup for treatment evaluation of patients with brain metastases using rtog study 7916. *International Journal of Radiation Oncology* Biology* Physics*, 16(3):669–673, 1989.

[24] Carlotta Domeniconi, Dimitris Papadopoulos, Dimitrios Gunopulos, and Sheng Ma. Subspace clustering of high dimensional data. In *Proceedings of the SIAM Conference on Data Mining (SDM)*, pages 517–521, 2004.

[25] John Eck, Spencer Chainey, James Cameron, and R Wilson. Mapping crime: Understanding hotspots. Technical report, National Institute of Justice, NIJ Special Report, August 2005.

[26] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–261, 1989.

[27] Vitaly Feldman. Hardness of approximate two-level logic minimization and pac learning with membership queries. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 363–372. ACM, 2006.

[28] Vitaly Feldman. Learning DNF expressions from fourier spectrum. *CoRR*, abs/1203.0594, 2012.

[29] Alex A Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10, March 2014.

[30] Zoubin Ghahramani and Katherine Heller. Bayesian sets. In *Proc of Neural Information Proc Sys (NIPS)*, 2005.

[31] Timothy J Gilbride and Greg M Allenby. A choice model with conjunctive, disjunctive, and compensatory screening rules. *Marketing Science*, 23(3):391–406, 2004.

[32] Siong Thye Goh and Cynthia Rudin. Box drawings for learning with imbalanced data. In *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2014.

[33] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. DB-CSC: a density-based approach for subspace clustering in graphs with feature vectors. In *Proceedings of the European conference on Machine Learning and principles and practice of knowledge discovery in databases (ECML-PKDD)*, pages 565–580. Springer, 2011.

[34] Samantha L. Gwinn, Christopher Bruce, Julie P. Cooper, and Steven Hick. Exploring crime analysis. Readings on essential skills, Second edition. Published by BookSurge, LLC, 2008.

[35] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.

[36] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction.* Springer, 2009.

[37] Roslin V Hauck, H Atabakhsb, Pichai Ongvasith, Harsh Gupta, and Hsinchun Chen. Using COPLINK to analyze criminal-justice data. *Computer*, 35(3):30–37, 2002.

[38] John R Hauser, Olivier Toubia, Theodoros Evgeniou, Rene Befurt, and Daria Dzyabura. Disjunctions of conjunctions, cognitive simplicity, and consideration sets. *Jour of Marketing Research*, 47(3):485–496, 2010.

[39] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–91, 1993.

[40] Jue Hou, Chamindi Seneviratne, Xiaogang Su, Jeremy Taylor, Bankole Johnson, Xin-Qun Wang, Heping Zhang, Henry R Kranzler, Joseph Kang, and Lei Liu. Subgroup identification in personalized treatment of alcohol dependence. *Alcoholism: Clinical and Experimental Research*, 39(7):1253–1259, 2015.

[41] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

[42] PN Johnson-Laird, Sangeet S Khemlani, and Geoffrey P Goodwin. Logic, probability, and human reasoning. *Trends in cognitive sciences*, 19(4):201–214, 2015.

[43] Adam R. Klivans and Rocco Servedio. Learning dnf in time $2^{O(n^{1/3})}$. In *Proc of the 33rd Annual ACM Symp on Theory of Computing*, STOC '01, pages 258–265, New York, NY, USA, 2001. ACM.

[44] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans on KD from Data*, 3(1):1–58, March 2009.

[45] Nada Lavrac and Saso Dzeroski. Inductive logic programming. In *WLP*, pages 146–160. Springer, 1994.

[46] Beom S Lee, Pranab K Sen, Nan S Park, Roger A Boothroyd, Roger H Peters, and David A Chiriboga. A clustering method to identify who benefits most from the treatment group in clinical trials. *Health Psychology and Behavioral Medicine: an Open Access Journal*, 2(1):723–734, 2014.

[47] Benjamin Letham, Cynthia Rudin, and Katherine A. Heller. Growing a list. *Data Mining and Know Disc*, 27(3):372–395, 2013.

[48] Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 2015. accepted with minor revision.

[49] Randall A Lewis and David H Reiley. Online ads and offline sales: measuring the effect of retail advertising via a controlled experiment on yahoo! *Quantitative Marketing and Economics*, 12(3):235–266, 2014.

[50] Wenmin Li, Jiawei Han, and Jian Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *ICDM 2001*, pages 369–376. IEEE, 2001.

[51] M. Lichman. UCI machine learning repository, 2013.

[52] Song Lin and Donald E. Brown. An outlier-based data association method for linking criminal incidents. *Decision Support Systems*, 41(3):604–615, March 2006.

[53] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

[54] Tzu-Pin Lu and James J Chen. Subgroup identification for treatment selection in biomarker adaptive design. *BMC medical research methodology*, 15(1):1, 2015.

[55] Bing Ma, Wynne Liu, and Yiming Hsu. Integrating classification and association rule mining. In *Proceedings of the 4th International Conf on Knowledge Discovery and Data Mining (KDD)*, 1998.

[56] David Martens and Bart Baesens. Building acceptable classification models. In *Data Mining*, pages 53–74. Springer, 2010.

[57] David Martens, Jan Vanthienen, Wouter Verbeke, and Bart Baesens. Performance of classification models from a user perspective. *Decision Support Systems*, 51(4):782–793, 2011.

[58] Tyler McCormick, Cynthia Rudin, and David Madigan. A hierarchical model for association rule mining of sequential events: An approach to automated medical symptom prediction. *Annals of Applied Statistics*, 2012.

[59] George Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

[60] George O Mohler, Martin B Short, P Jeffrey Brantingham, Frederic Paik Schoenberg, and George E Tita. Self-exciting point process modeling of crime. *Journal of the American Statistical Association*, 106(493), 2011.

[61] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[62] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining cohesive patterns from graphs with feature vectors. In *Proceedings of the SIAM Conference on Data Mining (SDM)*, volume 9, pages 593–604, 2009.

[63] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

[64] Shyam Varan Nath. Crime pattern detection using data mining. In *Proc. Web Intelligence and Intelligent Agent Technology Workshops*, pages 41–44, 2006.

[65] National Law Enforcement and Corrections Technology Center. 'Calculate' repeat crime. *TechBeat*, Fall 2008.

[66] Daniel B Neill and Gregory F Cooper. A multivariate bayesian scan statistic for early event detection and characterization. *Machine learning*, 79(3):261–282, 2010.

[67] Vincent Ng, Stephen Chan, Derek Lau, and Cheung Man Ying. Incremental mining for temporal association rules for crime pattern discoveries. In *Proc. 18th Australasian Database Conference*, volume 63, pages 123–132, 2007.

[68] Beth Pearsall. Predictive policing: The future of law enforcement? *National Institute of Justice Journal*, 266:16–19, 2010.

[69] Jian Pei, Xiaoling Zhang, Moonjung Cho, Haixun Wang, and Philip S Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 259–266. IEEE, 2003.

[70] Ariel D Procaccia and Jeffrey S Rosenschein. Exact vc-dimension of monotone formulas. *Neural Information ProcessingâĂŤLetters and Reviews*, 10(7):165–168, 2006.

[71] Jerry H Ratcliffe and George F Rengert. Near-repeat patterns in Philadelphia shootings. *Security Journal*, 21(1):58–76, 2008.

[72] Greg Ridgeway. The pitfalls of prediction. *NIJ Journal*, 271:34–40, 2013.

[73] Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

[74] G Rubio, G Ponce, R Rodriguez-Jimenez, MA Jimenez-Arriero, J Hoenicka, and T Palomo. Clinical predictors of response to naltrexone in alcoholic patients: who benefits most from treatment with naltrexone? *Alcohol and Alcoholism*, 40(3):227–233, 2005.

[75] Cynthia Rudin, Benjamin Letham, and David Madigan. Learning theory analysis for association rules and sequential event prediction. *Journal of Machine Learning Research*, 14:3384–3436, 2013.

[76] Stefan Rüping. *Learning interpretable models*. PhD thesis, Universität Dortmund, 2006.

[77] Patrick M Schnell, Qi Tang, Walter W Offen, and Bradley P Carlin. A bayesian credible subgroups approach to identifying patient subgroups with positive treatment effects. 2015.

[78] Lawrence W. Sherman, Patrick R. Garlin, and Micheal E. Buerger. Hot spots predatory crime: routine activities and the criminology of place. *Criminology*, 27(1):27–56, 1989.

[79] Martin B Short, Maria R D'Orsogna, Virginia B Pasour, George E Tita, Paul J Brantingham, Andrea L Bertozzi, and Lincoln B Chayes. A statistical model of criminal behavior. *Mathematical Models and Methods in Applied Sciences*, 18:1249–1267, 2008.

[80] Martin B Short, MR D'Orsogna, PJ Brantingham, and GE Tita. Measuring and modeling repeat and near-repeat burglary effects. *Journal of Quantitative Criminology*, 25(3):325–339, 2009.

[81] Wei-Guang Teng, Ming-Jyh Hsieh, and Ming-Syan Chen. On the mining of substitution rules for statistically dependent items. In *ICDM 2003*, pages 442–449. IEEE, 2002.

[82] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 2015. Accepted.

[83] Berk Ustun, M. Brandon Westover, Cynthia Rudin, and Matt T. Bianchi. Clinical prediction models for sleep apnea: The importance of medical history over symptoms. *Journal of Clinical Sleep Medicine*, 2015. Accepted.

[84] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[85] Katrien Verbert, Nikos Manouselis, Xavier Ochoa, Martin Wolpers, Hendrik Drachsler, Ivana Bosnic, and Erik Duval. Context-aware recommender systems for learning: a survey and future challenges. *IEEE Transactions on Learning Technologies*, 5(4):318–335, 2012.

[86] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

[87] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Int'l Conf on Machine Learning*, pages 577–584, 2001.

[88] Fulton Wang and Cynthia Rudin. Falling rule lists. In *Proceedings of Artificial Intelligence and Statistics (AISTATS)*, 2015.

[89] Gang Wang, Hsinchun Chen, and Homa Atabakhsh. Automatically detecting deceptive criminal identities. *Communications of the ACM*, 47(3):70–76, 2004.

[90] Haixun Wang, Wei Wang, Jiong Yang, and Philip S Yu. Clustering by pattern similarity in large data sets. In *Proceedings of ACM SIGMOD*, pages 394–405, 2002.

[91] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, and Erica Klampfl. Learning optimized or's of and's. *arXiv preprint arXiv:1511.02210, 2015*, 2015.

[92] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. Or's of and's for interpretable classification, with application to context-aware recommender systems. *arXiv preprint arXiv:1504.07614*, 2015.

[93] Tong Wang, Cynthia Rudin, Daniel Wagner, and Rich Sevieri. Detecting patterns of crime with series finder. In *AAAI (Late-Breaking Developments)*, 2013.

[94] David Weisburd. Bringing social context back into the equation. *Criminology and Public Policy*, 11(2):317–326, 2012.

[95] David Weisburd and Lorraine Green Mazerolle. Crime and disorder in drug hot spots: Implications for theory and practice in policing. *Police Quarterly*, 3(3):331–349, 2000.

[96] Deborah Lamm Weisel. Burglary of single-family houses. Problem-Oriented Guides for Police Series, No.18, 2002.

[97] Xindong Wu, Chengqi Zhang, and Shichao Zhang. Mining both positive and negative association rules. In *Proceedings of the Nineteenth ICML*, pages 658–665. Morgan Kaufmann Publishers Inc., 2002.

[98] Xiaoxin Yin and Jiawei Han. Cpar: Classification based on predictive association rules. In *SDM*, volume 3, pages 369–376. SIAM, 2003.

[99] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, et al. New algorithms for fast discovery of association rules. In *KDD*, volume 97, pages 283–286, 1997.