Information-theoretic partition-based models for interpretable machine learning

Yang, L.

The more the data is compressed, the more is learned from it.

Information-theoretic Partition-based Models for Interpretable Machine Learning

Yang, Lincen
阳 林 岑

Information-theoretic Partition-based Models for Interpretable Machine Learning · Lincen Yang

Information-theoretic Partition-based Models for Interpretable Machine Learning

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op vrijdag 20 september 2024
klokke 11:30 uur

door

Yang, Lincen 阳林岑

geboren te Zigong 自贡, China

in 1993

Promotores:

Dr. M. van Leeuwen
Prof.dr. A. Plaat

Promotiecommissie:

Prof.dr. E. Fromont	Université de Rennes
Prof.dr. T. De Bie	University of Ghent
Dr. S. Yu	Vrije Universiteit Amsterdam
Prof.dr. M. Bonsangue	
Dr. A. Knobbe	

This PhD project was conducted in the Explanatory Data Analysis Group, Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands.

SIKS Dissertation Series No. 2024-27. The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Research presented in this thesis was funded by the Dutch Research Council (NWO), under the the research programme 'Human-Guided Data Science by Interactive Model Selection' with Project Number 612.001.804.

ISBN: 978-94-6506-377-5
Printed by: Ridderprint

To my grandfather.

Abstract

Data-driven modeling in applied research often requires both predictive modeling and understanding data. Predictive models in supervised learning are indispensable for decision-making, early warning systems, and forming robust associations; meanwhile, algorithms in data mining and unsupervised learning search for patterns that are crucial for understanding data, getting insights into the physical process behind it, and taking action in the application domain.

In this dissertation, we study partition-based models that can be used both for interpretable predictive modeling and for understanding data via interpretable patterns. Specifically, we study probabilistic rule-based models for multi-class classification and histogram models for discretization, explanatory data analysis, and conditional mutual information estimation.

For rule-based models, we address the long-unresolved problem that interpretability of rule-based models is harmed by the need for conflict-resolving schemes for "overlaps" among rules (i.e., instances covered by multiple rules). Based on the intuitions that 1) overlaps can be used for characterizing the uncertainty of a model, and 2) only rules with similar class probability estimates are "allowed" to overlap, we formally introduce a new probabilistic model based on probabilistic rules, which we name *Truly Unordered Rule Set (TURS)*. In a series of three research papers (Chapters 2–4) we showcase that our proposed method learns "independent" rules that are not "entangled" with each other, which significantly improves the comprehensibility of the induced rule sets, as (explicit and implicit) orders within rules are eliminated. Building upon our proposed TURS model, we conduct a pilot study to demonstrate how it facilitates *interactive rule learning*: rules can be updated after receiving feedback from domain experts regarding disliked variables.

Next, in the realm of histogram-based methods, we first consider two dimensional datasets (Chapter 5), motivated by the ubiquitous spatial datasets collected by GPS devices. While histograms are widely used to discretize and summarize data, how to incorporate dependency among variables in *multivariate unsupervised discretization* is understudied. Further, the lack of a principled way for parameter setting leads to ambiguity to data analysts, as different parameter settings for histograms lead to significantly different results.

We address these issues by introducing a two dimensional histogram method based on the minimum description length (MDL) principle, with enhanced interpretability and transparency in the following aspects. First, we formally define the optimal histogram under the MDL principle, and hence eliminate the need for setting bin sizes, which increases the transparency of how histogram-based data discretization/summarization is obtained. Second, we propose the problem of learning two dimensional histograms in an expressive and flexible model class, in which the data space can be partitioned into subsets consisting of unions of *disjoint* rectangles. Based on this, we increase the interpretability of the model by learning histograms in which neighboring "bins" must have density estimates that are "dissimilar enough" under the MDL framework.

Following this line, we lastly study multi-dimensional adaptive histograms for conditional mutual information (CMI) estimation (Chapter 6), which is a fundamental task in understanding (conditional) independence and dependence relationships among variables. Thus, CMI estimation has applications in feature selection, independence testing, probabilistic graphic models, and causal inference. We specifically consider discrete-continuous mixture data, which is common in application areas where data can be truncated or is collected in a way that numeric values (instead of discrete levels) are only recorded in specific (e.g., anomaly) situations. We introduce histograms that can handle such mixture data, and support it with theoretical underpinnings, including measure-theoretic entropy definitions and consistency proofs. Notably, histogram bins with large differences between the (empirical) joint entropy and the sum of marginal entropies can be regarded as interpretable patterns for explaining dependency.

In conclusion, this dissertation explores MDL-based partition-based models and advances the field of interpretable machine learning by introducing innovative methods for a variety of tasks.

Contents

Chapter 1

Introduction

1.1 Data-driven Models

Data may contain a large amount of information about the underlying process that generated it. The larger the dataset is, the more information it may contain, but meanwhile the more difficult it may be for humans to distinguish useful patterns and regularities from noise and randomness.

Algorithms can handle large datasets that are intractable for humans to process, either by summarizing datasets and extracting patterns that are comprehensible for humans, or by building predictive models that construct relationships between variables. Models and algorithms in *data mining* and *unsupervised machine learning* concern the former, and those in *supervised machine learning* and *statistical predictive modeling* concern the latter.

1.1.1 Interpretable patterns for knowledge discovery

Due to our curious nature, we explore, reflect and learn from past experiences for all kinds of tasks. This includes building new connections between phenomena, discovering new knowledge from observations, and getting deep understanding about fundamental and complicated matters.

We can now enhance such activities with the help of large amounts of collected data, with the following application scenarios as examples. First, for instance, with the help of customer purchase records in supermarkets, associative patterns such as customers who buy coffee tend to buy milk and cookies as well can be "mined" from the large amount of records. In addition, with data collected by sensors installed in different places in manufacturing factories, anomalies can be detected and insight of what perhaps causes the anomalies may be identified. Furthermore, with the records of patients' conditions in a hospital, knowledge of factors that may lead to dangerous situations may be discovered. Finally, with trajectories recorded by GPS devices, regions with different popularity can be detected and insight about urban planning and/or police force distribution may be obtained.

Various models and algorithms exist for different kinds of tasks in inducing patterns from data, including and not limited to clustering, anomaly detection, association rules, frequent pattern mining, and density estimation. However, with the increased complexity of such models and algorithms, it has become an im-

portant research problem to seek for *interpretability and transparency*, i.e., to ask how the model outputs are produced. For instance, given some tabular data with a large number of variables, a key task in understanding the data is to investigate the dependency structure among variables, as widely used in *graphical models and causal inference*. Both transparent models with interpretable patterns and black-box models like deep neural networks are commonly used for this task. However, only the former can provide insight into why the model outputs (conditionally) "independent" or "dependent" for a certain subset of variables.

1.1.2 Interpretable predictive models

Besides knowledge discovery, building predictive models for a given target variable from data with *machine learning algorithms* has become successful in many areas, in the sense that the accuracy of such models are (beyond) "reasonably good" nowadays.

Examples for areas where such models are applied include the following. First, streaming media like Netflix can predict whether a customer may like a new movie or not, based on their historic watch data. Second, banks can predict whether someone is likely to be capable of paying back their debts given their bank account transaction data. Third, physicians in hospitals may use data collected by monitoring patients' conditions to help them predict the risk of certain operations.

However, accuracy of a predictive model is not the only concern when we consider introducing such a model and putting it into real use in our society, especially in critical areas such as health care (in which decisions are to be made about diagnosis or medication use, for instance), the judicial system (in which decisions are to be made about whether someone is guilty), and financial services (in which decisions are to be made about whether someone can get a mortgage, or someone is conducting fraud).

Because of ethical reasons and/or because of the severity of the outcome after false predictions, decisions with major influence on people or the society cannot be made automatically merely based on predictions given by machine learning models. That is, only humans should be responsible for taking actions in such critical areas.

Consider a scenario where a physician needs to decide whether a patient in an intensive care unit (ICU) of a hospital is good enough to be discharged. While all

the data records on the conditions of the patient can be useful, physicians can use a machine learning model trained on historic patients dataset to make predictions *only* for decision support, instead of letting the model directly make a decision for the patient.

Another example may be the situation where analysts in an insurance company must decide whether a client is conducting fraud. Even if all the transactions related to this client contain a lot of information about their behavior, it still remains the job of the analyst to extract evidence from the data and the model outputs as we cannot let the model take the responsibility in judging whether someone is guilty of conducting fraud.

Thus, substantial research has been put into *obtaining interpretability* of predictive machine learning models, to accelerate the introduction of such models in critical areas. This includes both 1) explaining black-box models and 2) developing new intrinsically interpretable models, with the general goal of providing transparency for data-driven decisions and building trust between the model and the end user (e.g., data analysts and domain experts) (Doshi-Velez and Kim 2017; Rudin et al. 2022).

1.2 What is Interpretability?

Interpretability is an umbrella term that can have very different meanings in different contexts. Conceptually, interpretability may refer to *transparency, global interpretability*, and *local interpretability* (Molnar 2020).

First of all, transparency concerns the extent to which a human can understand the process of an algorithm "learning" the model from data—how an algorithm takes the data as input and then outputs the model (Molnar 2020). Further, local interpretability often refers to an explanation of how the model output of a single instance is obtained; in contrast, global interpretability refers to the explanation of the model as a whole.

Thus, different models are considered (intrinsically) interpretable for different reasons. For instance, decision tree models (Breiman et al. 1984; Quinlan 2014) and rule-based models (Clark and Boswell 1991; Cohen 1995) are considered interpretable often due to the fact that the decision logic of every single prediction can be directly read by humans. In addition, linear models are consid-

ered interpretable as the marginal effect of a unit change in some feature value on a predicted value is described by a linear function, which is assumed to be "easily understandable". Further, generalized additive models (GAMs) are sometimes also considered interpretable (Caruana et al. 2015) as the marginal effect of feature changes on the target variable can be described by some non-linear function that can be visualized (and hence examined by the end user).

Meanwhile, what "interpretable models" mean in *unsupervised learning* is a bit more vague. While we may consider the K-means method for clustering interpretable, as it can more or less be explained why two instances belong to the same cluster (or two different clusters), it may be difficult to justify a clustering method based on deep neural network to be interpretable. Similarly, we may consider (linear) principle component analysis (PCA) interpretable as the associated "importance" for each dimension after the "rotation" of the basis of a vector space can be directly calculated; however, an embedding method based on an auto-encoder can hardly be understood by a human.

Yet, the concept becomes much more intuitive when we talk about interpretability in a *comparative* manner. For instance, a model that can make predictions together with *feature importance* (i.e., how much each feature "contributes" to the given model output) seems *more* interpretable than a model without feature importance. For instance, this is widely used in the field of *computer vision*, i.e., to attribute the model output to each pixel and visualize it (Adebayo et al. 2018).

Thus, one may argue that by introducing an approach for obtaining feature importance, the interpretability of a machine learning model class is increased[1]. Besides obtaining feature importance, it is also common to increase interpretability by 1) providing (local) surrogate models that are much "simpler" than the model to be explained (Ribeiro et al. 2016), and 2) reducing model complexity while maintaining predictive performance (Wu et al. 2018).

[1]Nevertheless, this may bring another issue that the method used for obtaining the feature importance may be complicated and hence cannot be regarded as transparent.

1.3 Partition-based Models

In this dissertation, we focus on partition-based models. Specifically, we consider *rule-based models* for supervised learning and *(adaptive) histogram models* for unsupervised learning, for which we now provide a very gentle introduction.

1.3.1 Probabilistic rule sets

A probabilistic rule is in the form of **IF some condition is met, THEN** $P(Y)$ **is equal to a certain value**, where $P(Y)$ denotes the (estimated) probability distribution for the target variable Y. As an example, consider a dataset that contains information on all flights in an airport within a certain period; then, one rule that may be induced from this dataset looks like **"IF Weather = Fog AND Flight_time $\leq$ 9 a.m. THEN P(Delay) = 0.8"**.

Further, a probabilistic rule set is simply a set of probabilistic rules put together. Rule sets are often considered as intrinsically interpretability models, as such probabilistic rules can be directly read and comprehended by humans. In Table 1.1 we show an example rule set learned from a real dataset that we will elaborate on in Chapter 4.

Condition of Rules	**Probability of Readmission to ICU**
Ureum-max-all $\geq$ 12.1 Ademfrequentie-median-value-last24h $\geq$ 23.5	0.223
APTT-max-all $\geq$ 43.1 Ureum-mean-all $\geq$ 16.338	0.199
Leukocyten-mean-last $\geq$ 20.81	0.162
Kalium-count-first $\geq$ 6.0 specialty-Organization-value-sub-ICCTC = FALSE	0.131
Trombocyten-count-first $\geq$ 2.0 Ureum-last-last $<$ 9.2 specialty-Organization-value-sub-ICCTC = TRUE	0.019
None of the above	0.059

Table 1.1: A rule set describing readmission risk that is learned from patients admitted to the intensive care unit of a hospital (described in detail in Chapter 4).

We are particularly interested in rule-based models due to the following reasons. First, one appealing property of rule-based models is that it connects interpretable predictive modeling and knowledge discovery, in the sense that it on one hand can be used for making (probabilistic) predictions for the target variable, and on the other hand, each rule is a local pattern that summarizes a subset of

the data and hence can be used for understanding the data itself and obtaining insights.

Second, the interpretability of rule-based models concerns both global interpretability and local interpretability. That is, individual rules can be used for explaining why a single prediction is made; meanwhile, a human can comprehend the rule set as a whole to grasp the internal logic of the model. Thus, rule-based models do not rely on post-hoc, external, and potentially non-transparent methods for obtaining interpretability.

Third, as rules are readable by humans, rule-based models are very accessible to domain experts who are not experts on machine learning methods. Thus, rule-based models are suitable to be used as a foundation for developing *interactive machine learning* methods: to allow the domain expert to give feedback to rules and to let the model incorporate the feedback by means of self-updating.

1.3.2 Multi-dimensional adaptive histograms

Histograms are widely used as a tool for visualizing the distribution of one- or two-dimensional data. For one- and multi-dimensional datasets in general, histograms can also be used as a tool for density estimation, data summarization, and discretization.

As an unsupervised partition-based model, histograms partition data points into bins, and within each bin the probability density is estimated as one constant. Specifically, an *adaptive* histogram is a histogram with variable bin sizes. For multi-dimensional histograms, bins may refer to as (hyper-)boxes or even more flexible subsets from a certain data partitioning process.

A multi-dimensional adaptive histogram is a simple yet powerful model that can effectively capture dependency structures among different dimensions. Specifically, multi-dimensional bins can be regarded as interpretable patterns that highlight subsets of data points for which the empirical marginal and conditional distributions differ from each other. This makes multi-dimensional adaptive histograms suitable for 1) discretization that incorporates the dependencies among dimensions, and 2) learning dependency structures for probabilistic graphic models.

We illustrate an example of a two-dimensional adaptive histogram in Figure 1.1, which is obtained by our proposed method that will be discussed in Chapter 5 on a simulated Gaussian dataset.

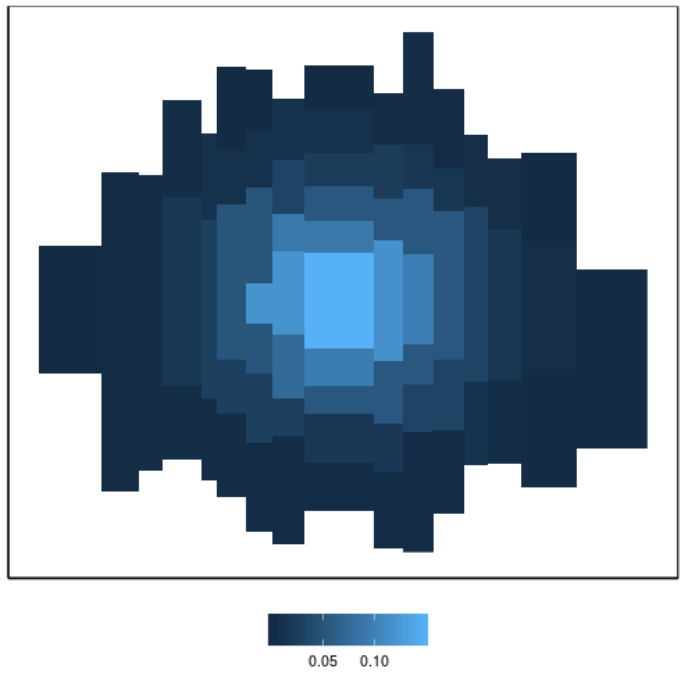

Figure 1.1: The histogram model for a simulated Gaussian dataset with density estimation (discussed in detail in Chapter 5).

1.4 A Gentle Introduction to the MDL Principle

We leverage information-theoretic tools and specifically the minimum description length (MDL) principle to formalize the problem of learning partition-based models from data as MDL-based model selection tasks.

The MDL principle has roots in information theory (Rissanen 1978). The core idea may be summarized as *learning by compression*. Specifically, the MDL principle states that the more we can compress the data in a *lossless* manner, the more structure and pattern we have found in the data. The degree of compression is measured by the code length, in bits, needed to encode data, together with the code length needed to encode the model that describes the regularities (structure and patterns) of the data.

Consider as an example learning regularities from the following two binary sequences: 1) a randomly generated binary sequence "100111101...", and 2) a binary sequence with the same length, which contains the regularity that a one is

always followed by a zero. Imagine we now need to communicate each sequence to a message receiver; for the first sequence, as there exists no regularity inside it, the only way is to enumerate each '1' and '0' in order. However, for the second sequence, we can first communicate the regularity itself to the message receiver, and then when we enumerate each '1' and '0' in order, we can skip the '0' after each '1' as the message receiver can add one '0' after receiving a '1' according to the regularity the receiver received. Thus, the number of bits needed to communicate the second sequence will be shorter than the length of the sequence itself. In this case, we say that the data is compressed with the help of the regularity. Reversely, regularity (instead of noise) is found if we find that it can be used to compress the data.

Thus, applying the MDL principle to certain tasks is about calculating the code length for the model and data together, which depends on the encoding scheme. Historically, choosing the encoding scheme was done in a crude and more or less arbitrary manner, and the earliest application of the MDL principle to partition-based models was to use the MDL principle in the well-known C4.5 (Quinlan 2014) and RIPPER (Cohen 1995) rule learning methods. In contrast, the modern version of the MDL principle (Grünwald and Roos 2019) exploits the connection between encoding and probabilistic modeling. Statistically, the length (in bits) of a given code[2] is connected to a corresponding probability distribution, as described by *Kraft's inequality* (Grünwald 2007).

The main motivation for adopting the MDL principle is that it removes the commonly used regularization parameter in the formalization of the learning problem, as the MDL principle automatically trades off between the goodness-of-fit and model complexity, which increases the transparency of how a learning method "creates" the model.

1.5 Research Questions

The overarching question we study in this dissertation is how to increase interpretability and transparency for partition-based models for supervised and unsupervised learning. This mainly concerns 1) how to make histograms more interpretable by having adaptive bins, as well as more transparent by reducing

[2]We assume all codes are prefix codes in this dissertation.

the number of user-defined parameters (e.g., the number of bins), and 2) how to increase the interpretability of rule-based models towards the level so that human-guided rule learning is possible. We next present our three main research questions in detail.

1.5.1 Towards rule sets for interactive rule learning

Although rule-based models carry significant interpretability because of the readability of the rules, our goal is to bring their interpretability to an even higher level so that domain experts can comprehend and potentially edit individual rules without considering the effect of/on other rules.

Consider a set of classification rules, each rule in the form of $\bigwedge\{X_i \in R_i\} \rightarrow Y \sim P(Y)$, in which X_i represents a single feature variable and R_i represents a set/range of values. For instance, a single rule could be denoted as **"Weather = Fog $\wedge$ Flight_time $\leq$ 9 a.m. $\rightarrow$ P(Delay) = 0.8"** .

Enhancing the interpretability of a set of such rules requires properly handling the "overlap" of rules, a long unresolved issue in learning rule-based models. Specifically, overlap refers to the case where one instance (e.g., one flight) satisfies the conditions of multiple rules, *potentially with different probabilistic predictions for the target variable* (e.g., flight delay).

As overlaps among rules make rules "entangled", we aim to enhance the interpretability of rule-based models by obtaining rules that are "independent" with regard to each other. Thus, we consider the following research question:

- **Research Question 1**: How can we formalize rule sets as probabilistic models such that the individual rules are independent from each other? Further, how can we learn such models from data?

Notably, due to the lack of a widely accepted general definition of interpretability, we consider interpretability in a comparative manner. Different from the common approach of seeking more interpretable rule-based models by making rules "simpler" (i.e., fewer and shorter rules), we instead consider making a rule-based model more interpretable by reducing the conflicts caused by overlaps among rules. We explain in detail why and how conflicts caused by overlaps affect interpretability in Chapters 2 and 3.

1.5.2 Adaptive histograms for discretization

Discretization is the task of summarizing continuous values and transforming them into a certain discrete representation form. It is a necessary pre-processing step if the following step in the modeling pipeline requires discrete values as input.

Intuitively, discretization methods need to strike a balance between the amount of preserved information and the complexity of the discretized representation (as a simple representation of data has benefits in terms of interpretability).

We specifically consider unsupervised discretization, i.e., discretization for a dataset without a target variable. Hence, the quality of the discretization cannot be evaluated by evaluating the prediction loss of the following step. Instead, it is crucial under such circumstances to *discretize the data in a way that makes sense to domain experts*, which concerns providing transparency regarding how the discretization is obtained.

Histogram-based models have the advantage of being very interpretable in discretization, data summarization, and density estimation (Kontkanen and Myllymäki 2007b; Scott 2015). However, while fixed histograms (histograms with equal bin sizes) are still widely used, they are often constructed with user-defined, more or less arbitrarily set parameters that control the number of bins (and hence the bin sizes). Thus, different patterns and, as a result, summarizations of a given dataset may exist, without any principled way of justifying which one represents the data more accurately. We argue that this may cause confusion to domain experts in practice, and hence negatively affects the trust in the model output by humans.

Further, while histograms as probabilistic models "approximate" the density of a given dataset by piece-wise constant values, existing methods lack a justification of whether the density inside each bin of a histogram is indeed (approximately) homogeneous, and at the same time, whether the density of neighboring bins are "very" different. Hence, the empirical distribution of data points within each bin is not transparent to domain experts in this case. Similarly, it often remains unclear whether "neighboring" bins have similar density estimates. For domain experts, merging such neighboring bins makes the model simpler and hence is beneficial for interpretability.

We specifically focus on two-dimensional datasets because spatial data is widely collected and analyzed, while a large number of existing algorithms for

mining spatial (or spatio-temporal) patterns require discrete values as input. This brings additional challenges as previous methods rarely considered the dependency of different dimensions, but applied a one-dimensional discretization method for each dimension separately.

To address these challenges, we propose our second research question:

- **Research Question 2:** how can we construct parameter-free two-dimensional histograms with transparent and informative patterns (bins)?

1.5.3 Histograms for learning dependency structure

We further exploit histogram-based models for the task of conditional mutual information estimation, which is useful in learning dependency structures among variables. That is, given three random variables denoted as X, Y, Z, the conditional mutual information (CMI) $I(X;Y|Z)$ characterizes whether X and Y are conditionally independent given Z. CMI estimation has wide applications in feature selection, conditional independence testing, and dependency structure learning (for graphic models).

We specifically consider CMI estimation for data with mixed types, of which each dimension can be continuous, discrete, and discrete-continuous mixtures. Although k-nearest neighbor (kNN) estimation is shown to work in such cases, we consider histogram-based models a more interpretable approach for such tasks, as each bin of the histogram can be regarded as an interpretable local pattern for explaining which subset of the data points contributes to the dependency among certain variables (and to what extent). This leads to our last research question:

- **Research Question 3:** How can we construct a multi-dimensional adaptive histogram-based model for interpretable CMI estimation?

Specifically, we extend our two-dimensional histogram-based models (discussed above) to multi-dimensional cases.

1.6 Contributions

This dissertation is composed of articles listed in the Table 1.2. The contributions for the paper corresponding to Chapter 6 were split half/half with Alexander Marx.

Article	Used in
Yang, L & van Leeuwen, M **Truly Unordered Probabilistic Rule Sets for Multi-class Classification**. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2022), 2022.	Chapter 2
Yang, L & van Leeuwen, M **Probabilistic Truly Unordered Rule Sets**. Under review, submitted to JMLR.	Chapter 3
Yang, L, van der Meijden, S, Arbous, M.S & van Leeuwen, M **ICU Readmission Risk Analysis with Probabilistic Rule Set Model**. In Preparation.	Chapter 4
Yang, L, Baratchi, M & van Leeuwen, M **Unsupervised Discretization by Two-dimensional MDL-based Histogram.** Machine Learning, Springer, 2023.	Chapter 5
Marx, A, Yang, L & van Leeuwen, M **Estimating Conditional Mutual Information for Discrete-Continuous Mixtures using Multi-Dimensional Adaptive Histograms**. In: Proceedings of the SIAM Conference on Data Mining 2021 (SDM'21), 2021.	Chapter 6

Table 1.2: List of papers.

We briefly summarize the contributions of these chapters as follows. In Chapters 2 and 3, we introduce the truly unordered rule set (TURS) model and present our method for learning TURS models from data, which substantially improves the comprehensibility of rule set models. Specifically, in Chapter 2 we address the challenge of how we can treat overlaps as uncertainty in order to eliminate the need for post-hoc conflict-resolving schemes for overlap. We propose our first algorithm to learn TURS models from data, and showcase that rule sets learned from data, with overlaps representing uncertainty, can have on-par predictive performance in comparison to rule-based methods with explicit or implicit orders among rules (which are hence less interpretable).

Subsequently, in Chapter 3, we formalize the probabilistic modeling and the learning problem for TURS in a more rigorous way. Further, we propose a refined algorithm and conduct extensive experiments to present the appealing properties of learned models from various perspectives.

In Chapter 4, we apply the TURS model to the problem of ICU readmission risk analysis, and demonstrate that our method can be used for interactive rule

learning.

In Chapter 5, we study two-dimensional MDL-based histograms for unsupervised discretization. The main contributions are two fold. First, regarding the MDL theory, we show that the *parametric complexity* does *not* depend on the dimensionality of the data, which is defined as the *regret* term in the formula that calculates the code length (in bits). Second, we propose a novel method that can learn very flexible and expressive histograms for simulated and real-world datasets.

In Chapter 6, we extend the MDL histograms to multi-dimensional cases for the task of CMI estimation. Our main contributions include the following. First, we develop a series of theoretic results to construct our CMI estimator: 1) we define measure-theoretic entropy and prove the formula for calculating CMI based on entropy also holds for discrete-continuos mixtures, 2) we formally define histogram-based models for discrete-continuous mixture data, and 3) we prove the consistency of the proposed CMI estimator. Second, we propose an alternating algorithm to learn multi-dimensional adaptive histograms that are shown to be highly competitive when we benchmark against several widely used competitor methods.

Chapter 2

Rule Sets with Overlaps that Represent Uncertainty

This chapter has been published as Yang, L and van Leeuwen, M **Truly Unordered Probabilistic Rule Sets for Multi-class Classification.** In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2022), Springer,* 2022.

Chapter Abstract

Rule set learning has long been studied and has recently been frequently revisited due to the need for interpretable models. Still, existing methods have several shortcomings: 1) most recent methods require a binary feature matrix as input, while learning rules directly from numeric variables is understudied; 2) existing methods impose orders among rules, either explicitly or implicitly, which harms interpretability; and 3) currently no method exists for learning probabilistic rule sets for multi-class target variables (there is only one for probabilistic rule lists).

We propose TURS, for Truly Unordered Rule Sets, which addresses these shortcomings. We first formalize the problem of learning truly unordered rule sets. To resolve conflicts caused by overlapping rules, i.e., instances covered by multiple rules, we propose a novel approach that exploits the probabilistic properties of our rule sets. We next develop a two-phase heuristic algorithm that learns rule sets by carefully growing rules. An important innovation is that we use a surrogate score to take the global potential of the rule set into account when learning a local rule.

Finally, we empirically demonstrate that, compared to non-probabilistic and (explicitly or implicitly) ordered state-of-the-art methods, our method learns rule sets that not only have better interpretability but also better predictive performance.

2.1 Introduction

When using predictive models in sensitive real-world scenarios, such as in health care, analysts seek for intelligible and reliable explanations for predictions. Classification rules have considerable advantages here, as they are directly readable by humans. While rules all seem alike, however, some are more interpretable than others. The reason lies in the subtle differences of how rules form a model. Specifically, rules can form an unordered *rule set*, or an explicitly ordered *rule list*; further, they can be categorized as probabilistic or non-probabilistic.

In practice, probabilistic rules should be preferred because they provide information about the uncertainty of the predicted outcomes, and thus are useful when a human is responsible to make the final decision, as the expected "utility" can be calculated. Meanwhile, unordered rule sets should also be preferred, as they have better properties regarding interpretability than ordered rule lists. While no agreement has been reached on the precise definition of interpretability of machine learning models (Molnar 2020; Murdoch et al. 2019), we specifically treat interpretability with domain experts in mind. From this perspective, a model's interpretability intuitively depends on two aspects: the degree of difficulty for a human to comprehend the model itself, and to understand a single prediction. Unordered probabilistic rule sets are favorable with respect to both aspects, for the following reasons. First, comprehending ordered rule lists requires comprehending not only each individual rule, but also the relationship among the rules, while comprehending unordered rule sets requires only the former. Second, the explanation for a single prediction of an ordered rule list must contain the rule that the instance satisfies, together with all of its preceding rules, which becomes incomprehensible when the number of preceding rules is large.

Further, crucially, existing methods for rule set learning claim to learn unordered rule sets, but most of them are not truly unordered. The problem is caused by *overlap*, i.e., a single instance satisfying multiple rules. Ad-hoc schemes are widely used to resolve prediction conflicts caused by overlaps, typically by ranking the involved rules with certain criteria and always selecting the highest ranked rule (Lakkaraju et al. 2016; Zhang and Gionis 2020) (e.g., the most accurate one). This, however, imposes implicit orders among rules, making them entangled instead of truly unordered.

This can badly harm interpretability: to explain a single prediction for an instance, it is now insufficient to only provide the rules the instance satisfies, because other higher-ranked rules that the instance does *not* satisfy are also part of the explanation. For instance, imagine a patient is predicted to have *Flu* because they have *Fever*. If the model also contains the higher-ranked rule *"Blood in stool* $\rightarrow$ *Dysentery"*, the explanation should include the fact that *"Blood in stool"* is not true, because otherwise the prediction would change to *Dysentery*. If the model contains many rules, it becomes impractical to have to go over all higher-ranked rules for each prediction.

Learning truly unordered probabilistic rule sets is a very challenging problem though. Classical rule set learning methods usually adopt a separate-and-conquer strategy, often sequential covering: they iteratively find the next rule and remove instances satisfying this rule. This includes 1) binary classifiers that learn rules only for the "positive" class (Fürnkranz et al. 2012), and 2) its extension to multi-class targets by the one-versus-rest paradigm, i.e., learning rules for each class one by one (Clark and Boswell 1991; Cohen 1995). Importantly, by iteratively removing instances the *probabilistic predictive conflicts* caused by overlaps, i.e., rules having different probability estimates for the target, are ignored. Recently proposed rule learning methods go beyond separate-and-conquer by leveraging discrete optimization techniques (Dash et al. 2018; Lakkaraju et al. 2016; Wang et al. 2017; Yang et al. 2021; Zhang and Gionis 2020), but this comes at the cost of requiring a binary feature matrix as input. Moreover, these methods are neither probabilistic nor truly unordered, as they still use ad-hoc schemes to resolve predictive conflicts caused by overlaps.

Approach and contributions. To tackle these challenges and learn truly unordered probabilistic rules, we first formalize rule sets as probabilistic models. We adopt a probabilistic model selection approach for rule set learning, for which we design a criterion based on the minimum description length (MDL) principle (Grünwald and Roos 2019). Second, we propose a novel surrogate score based on decision trees that we use to evaluate the potential of incomplete rule sets. Third, we are the first to design a rule learning algorithm that deals with probabilistic conflicts caused by overlaps already during the rule learning process. We point out that rules that have been added to the rule set may become obstacles for new rules, and hence carefully design a two-phase heuristic algorithm, for which we

adopt diverse beam search (Van Leeuwen and Knobbe 2012). Last, we benchmark our method, named TURS, for Truly Unordered Rule Sets, against a wide range of methods. We show that the rule sets learned by TURS, apart from being probabilistic and truly unordered, have better predictive performance than existing rule list and rule set methods.

2.2 Related Work

Rule lists. Rules in a rule list are connected by IF-THEN-ELSE statements. Existing methods include CBA (Liu et al. 1998), ordered CN2 (Clark and Niblett 1989), PART (Frank and Witten 1998), and the recently proposed CLASSY (Proença and Leeuwen 2020) and Bayesian rule list (Yang et al. 2017). We argue that rule lists are more difficult to interpret than rule sets because of their explicit orders.

One-versus-rest learning. This category focuses on only learning rules for a single class label, i.e., the "positive" class, which is already sufficient for binary classification (Dash et al. 2018; Wang et al. 2017; Yang et al. 2021). For multi-class classification, two approaches exist. The first, taken by RIPPER (Cohen 1995) and C4.5 (Quinlan 2014), is to learn each class in a certain order. After all rules for a single class have been learned, all covered instances are removed (or those with this class label). The resulting model is essentially an ordered list of rule sets, and hence is more difficult to interpret than rule set.

The second approach does not impose an order among the classes; instead, it learns a set of rules for each class against all other classes. The most well-known are unordered-CN2 and FURIA (Clark and Boswell 1991; Hühn and Hüllermeier 2009). FURIA avoids dealing with conflicts of overlaps by using all rules for predicting unseen instances; as a result, it cannot provide a single rule to explain its prediction. Unordered-CN2, on the other hand, handles overlaps by "combining" all overlapping rules into a "hypothetical" rule, which sums up all instances in all overlapping rules and hence ignoring probabilistic conflicts for constructing rules. In Section 2.6, we show that our method learns smaller rule sets with better predictive performance than unordered-CN2.

Multi-class rule sets. Very few methods exist for directly learning rules for multi-class targets, which is algorithmically more challenging than the one-versus-rest paradigm, as the separate-and-conquer strategy is not applicable. To the best

of our knowledge, the only existing methods are IDS (Lakkaraju et al. 2016) and DRS (Zhang and Gionis 2020). Both are neither probabilistic nor truly unordered. To handle conflicts of overlaps, IDS follows the rule with the highest F1-score, and DRS uses the most accurate rule.

Last, different but related approaches include 1) decision tree based methods such as CART (Breiman et al. 1984), which produce rules that are forced to share many "attributes" and hence are longer than necessary, as we will empirically demonstrate in Section 2.6, and 2) a Bayesian rule mining (Gay and Boullé 2012) method, which adopts naive bayes with the mined rules for prediction, and hence does not produce a rule set model in the end. The 'lazy learning' approach for rule-based models can also avoid the conflicts of overlaps (Veloso et al. 2006), but no global rule set model describing the whole dataset is constructed in this case.

2.3 Rule Sets as Probabilistic Models

We first formalize individual rules as *local* probabilistic models, and then define rule sets as *global* probabilistic models. The key challenge lies in how to define $P(Y = y|X = x)$ for an instance (x, y) that is covered by multiple rules.

2.3.1 Probabilistic Rules

Denote the input random variables by $X = (X_1, \ldots, X_d)$, where each X_i is a one-dimensional random variable representing one dimension of X, and denote the categorical target variable by $Y \in \mathscr{Y}$. Further, denote the dataset from which the rule set can be induced as $D = \{(x_i, y_i)\}_{i\in[n]}$, or (x^n, y^n) for short. Each (x_i, y_i) is an instance. Then, a probabilistic rule S is written as

$$(X_1 \in R_1 \wedge X_2 \in R_2 \wedge \ldots) \rightarrow P_S(Y), \tag{2.1}$$

where each $X_i \in R_i$ is called a *literal* of the *condition* of the rule. Specifically, each R_i is an interval (for a quantitative variable) or a set of categorical levels (for a categorical variable).

A probabilistic rule of this form describes a subset S of the full sample space of X, such that for any $x \in S$, the conditional distribution $P(Y|X = x)$ is approximated by the probability distribution of Y conditioned on the event $\{X \in$

$S\}$, denoted as $P(Y|X \in S)$. Since in classification Y is a discrete variable, we can parametrize $P(Y|X \in S)$ by a parameter vector $\boldsymbol{\beta}$, in which the jth element β_j represents $P(Y = j|X \in S)$, for all $j \in \mathscr{Y}$. We therefore denote $P(Y|X \in S)$ as $P_{S,\boldsymbol{\beta}}(Y)$, or $P_S(Y)$ for short. To estimate $\boldsymbol{\beta}$ from data, we adopt the maximum likelihood estimator, denoted as $P_{S,\hat{\boldsymbol{\beta}}}(Y)$, or $\hat{P}_S(Y)$ for short.

Further, if an instance (x, y) satisfies the condition of rule S, we say that (x, y) is *covered* by S. Reversely, the *cover* of S denotes the instances it covers. When clear from the context, we use S to both represent the rule itself and/or its cover, and define the number of covered instances $|S|$ as its *coverage*.

2.3.2 Truly Unordered Rule Sets as Probabilistic Models

While a rule set is simply a set of rules, the challenge lies in how to define rule sets as probabilistic models while keeping the rules truly unordered. That is, how do we define $P(Y|X = x)$ given a rule set M, i.e., a model, and its parameters? We first explain how to do this for a single instance of the training data, using a simplified setting where at most two rules cover the instance. We then discuss—potentially unseen—test instances and extend to more than two rules covering an instance. Finally, we define a rule set as a probabilistic model.

Class probabilities for a single training instance. Given a rule set M with K individual rules, denoted $\{S_i\}_{i\in[K]}$, any instance (x, y) falls into one of four cases: 1) exactly one rule covers x; 2) at least two rules cover x and no rule's cover is the subset of another rule's cover (*multiple non-nested*); 3) at least two rules cover x and one rule's cover is the subset of another rule's cover (*multiple nested*); and 4) no rule in M covers x.

To simplify the notation, we here consider at most two rules covering an instance—we later describe how we can trivially extend to more than two rules.

Covered by one rule. When exactly one rule $S \in M$ covers x, we use $P_S(Y)$ to "approximate" the conditional probability $P(Y|X = x)$. To estimate $P_S(Y)$ from data, we adopt the maximum likelihood (ML) estimator $\hat{P}_S(Y)$, i.e.,

$$\hat{P}_S(Y = j) = \frac{|\{(x, y) : x \in S, y = j\}|}{|S|}, \forall j \in \mathscr{Y}. \tag{2.2}$$

Note that we do not exclude instances in S that are also covered by other rules

(i.e., in overlaps) for estimating $P_S(Y)$. Hence, the probability estimation for each rule is independent of other rules; as a result, each rule is *self-standing*, which forms the foundation of a truly unordered rule set.

Covered by two non-nested rules. Next, we consider the case when x is covered by S_i and S_j, and neither $S_i \subseteq S_j$ nor $S_j \subseteq S_i$, i.e., the rules are non-nested.

When an instance is covered by two non-nested, partially overlapping rules, we interpret this as probabilistic *uncertainty*: we cannot tell whether the instance belongs to one rule or the other, and therefore approximate its conditional probability by the *union* of the two rules. That is, in this case we approximate $P(Y|X = x)$ by $P(Y|X \in S_i \cup S_j)$, and we estimate this with its ML estimator $\hat{P}(Y|X \in S_i \cup S_j)$, using all instances in $S_i \cup S_j$.

This approach is particularly useful when the estimator of $P(Y|X \in S_i \cap S_j)$, i.e., conditioned on the event $\{X \in S_i \cap S_j\}$, is indistinguishable from $\hat{P}(Y|X \in S_i)$ and $\hat{P}(Y|X \in S_j)$. Intuitively, this can be caused by two reasons: 1) $S_i \cap S_j$ consists of very few instances, so the variance of the estimator for $P(Y|X \in S_i \cap S_j)$ is large; 2) $P(Y|X \in S_i \cap S_j)$ is just very similar to $P(Y|X \in S_i)$ and $P(Y|X \in S_i)$, which makes it undesirable to create a separate rule for $S_i \cap S_j$. Our model selection approach, explained in Section 2.4, will ensure that a rule set with non-nested rules has high goodness-of-fit only if this 'uncertainty' is indeed the case.

Covered by two nested rules. When x is covered by both S_i and S_j, and S_i is a subset of S_j, i.e., $x \in S_i \subseteq S_j$, the rules are nested[1]. In this case, we approximate $P(Y|X = x)$ by $P(Y|X \in S_i)$ and interpret S_i as an *exception* of S_j. Having such nested rules to model such exceptions is intuitively desirable, as it allows to have general rules covering large parts of the data while being able to model smaller, deviating parts. In order to preserve the self-standing property of individual rules, for $x \in S_j \setminus S_i$ we still use $P(Y|X \in S_j)$ rather than $P(Y|X \in S_j \setminus S_i)$. Although this might seem counter-intuitive at first glance, using $P(Y|X \in S_j \setminus S_i)$ would implicitly impose an order between S_j and S_i, or—equivalently—implicitly change S_j to another rule that only covers instances in $S_j \wedge \neg S_i$.

Not covered by any rule. When no rule in M covers x, we say that x belongs to the so-called "else rule" that is part of every rule set and equivalent to $x \notin \bigcup_i S_i$. Thus, we approximate $P(Y|X = x)$ by $P(Y|X \notin \bigcup_i S_i)$. We denote the else rule

[1]Note that "nestedness" is based on the rules' covers rather than on their conditions. For instance, if S_i is $X_1 <= 1$ and S_j is $X_2 <= 1$, S_i and S_j could still be nested.

by S_0 and write $S_0 \in M$ for the else rule in M. Observe that the else rule is the only rule in every rule set that depends on the other rules and is therefore not self-standing; however, it will also have no overlap with other rules by definition.

Predicting for a new instance. When an unseen instance x' comes in, we predict $P(Y|X = x')$ depending on which of the aforementioned four cases it satisfies. An important question is whether we always need access to the training data, i.e., whether the probability estimates we obtain from the training data points are sufficient for predicting $P(Y|X = x')$. Specifically, if x' is covered by non-nested S_i and S_j, $P(Y|X = x')$ is predicted as $\hat{P}(Y|X \in S_i \cup S_j)$. However, if there are no training data points covered both by S_i and S_j, then we would not obtain $\hat{P}(Y|X \in S_i \cup S_j)$ in the training phase. Nevertheless, in this case we have $|S_i \cup S_j| = |S_i| + |S_j|$, and hence

$$\hat{P}(Y|X \in S_i \cup S_j) = \frac{|S_i|\hat{P}(Y|X \in S_i) + |S_j|\hat{P}(Y|X \in S_j)}{|S_i| + |S_j|}. \tag{2.3}$$

Thus, if x' is covered by one rule, two nested rules, or no rule in M, the corresponding probability estimates are already obtained during training. Thus, we conclude that access to the training data is not necessary for prediction.

Extension to overlaps of multiple rules. Whenever an instance x is covered by multiple rules, denoted $J = \{S_i, S_j, S_k, ...\}$, three cases may happen. The first case is all rules in J are nested. Without loss of generality, assume that $S_i \subseteq S_j \subseteq S_k \subseteq ...$; then, following the rationale for case of two nested rules, $P(Y|X = x)$ should be approximated by $P_{S_i}(Y)$. Therefore, when x is covered by multiple nested rules, only the "smallest" rule matters and we can act as if x is only covered by that single rule.

The second case is that all rules in J are non-nested with each other. Following the solution for modeling two non-nested rules, we use $P(Y|X \in \bigcup_{S \in J} S)$.

The third case is a mix of the previous two cases, i.e., rules in J are partially nested. In this case, we iteratively go over all $S \in J$: if there exists an $S' \in J$ satisfying $S' \subseteq S$ we remove S from J, and continue iterating until no nested overlap in J remains. If one single rule is left, we act as if x is covered by that single rule; otherwise, we follow the paradigm of modeling the non-nested overlaps with the rules left in J.

Probabilistic rule sets. We can now build upon the previous to define rule sets as probabilistic models. Formally, the probabilistic model corresponding to a rule set M is a family of probability distributions, denoted $P_{M,\theta}(Y|X)$ and parametrized by θ. Specifically, θ is a parameter vector representing all necessary probabilities of Y conditioned on events $\{X \in G\}$, where G is either a single rule or the union of multiple rules. θ is estimated from data by estimating each $P(Y|X \in G)$ by its maximum likelihood estimator. The resulting estimated vector is denoted as $\hat{\theta}$ and contains $\hat{P}(Y|X \in G)$ for all $G \in \mathscr{G}$, where $\mathscr{G}$ consists of all individual rules and the unions of overlapping rules in M.

Finally, we assume the dataset $D = (x^n, y^n)$ to be i.i.d. Specifically, let us define $(x, y) \vdash G$ for the following two cases: 1) when G is a single rule (including the else rule), then $(x, y) \vdash G \iff x \in G$; and 2) when G is a union of multiple rules, e.g., $G = \bigcup S_i$, then $(x, y) \vdash G \iff x \in \bigcap S_i$. We then have

$$P_{M,\theta}(y^n|x^n) = \prod_{G \in \mathscr{G}} \prod_{(x,y) \vdash G} P(Y = y|X \in G). \tag{2.4}$$

2.4 Rule Set Learning as Probabilistic Model Selection

Exploiting the formulation of rule sets as probabilistic models, we define the task of learning a rule set as a probabilistic model selection problem. Specifically, we use the minimum description length (MDL) principle for model selection.

2.4.1 Normalized Maximum Likelihood Distributions for Rule Sets

The MDL principle is one of the best off-the-shelf model selection methods and has been widely used in machine learning and data mining (Grünwald and Roos 2019). Although rooted in information theory, it has been recently shown that MDL-based model selection can be regarded as an extension of Bayesian model selection (Grünwald and Roos 2019).

The core idea of MDL-based model selection is to assign a single probability distribution to the data given a rule set M, the so-called *universal distribution* denoted by $P_M(Y^n|X^n = x^n)$. Informally, $P_M(Y^n|X^n = x^n)$ should be a

representative of the rule set model—as a family of probability distributions—$\{P_{M,\theta}(y^n|x^n)\}_\theta$. The theoretically optimal "representative" is defined to be the one that has minimax regret, i.e.,

$$\arg\min_{P_M} \max_{z^n \in \mathscr{Y}^n} \Bigg[-\log_2 P_M(Y^n = z^n | X^n = x^n) \\ - \Big(-\log_2 P_{\hat{\theta}(x^n, z^n)}(Y^n = z^n | X^n = x^n) \Big) \Bigg]. \tag{2.5}$$

We write the parameter estimator as $\hat{\theta}(x^n, z^n)$ to emphasize that it depends on the values of the target variables Y^n. The unique solution to P_M of Equation 2.5 is the so-called normalized maximum likelihood (NML) distribution:

$$P_M^{NML}(Y^n = y^n | X^n = x^n) = \frac{P_{M,\hat{\theta}(x^n,y^n)}(Y^n = y^n | X^n = x^n)}{\sum_{z^n \in \mathscr{Y}^n} P_{M,\hat{\theta}(x^n,z^n)}(Y^n = z^n | X^n = x^n)}. \tag{2.6}$$

That is, we "normalize" the distribution $P_{M,\hat{\theta}}(.)$ to make it a proper probability distribution, which requires the sum of all possible values of Y^n to be 1. Hence, we have $\sum_{z^n \in \mathscr{Y}^n} P_M^{NML}(Y^n = z^n | X^n = x^n) = 1$ (Grünwald and Roos 2019).

2.4.2 Approximating the NML Distribution

A crucial difficulty in using the NML distribution in practice is the computation of the normalizing term $\sum_{z^n} P_{\hat{\theta}(x^n,z^n)}(Y^n = z^n | X^n = x^n)$. Efficient algorithms almost only exist for exponential family models (Grünwald and Roos 2019), hence we approximate the term by the product of the normalizing terms for the individual rules.

NML distribution for a single rule. For an individual rule $S \in M$, we write all instances covered by S as (x^S, y^S), in which y^S can be regarded as a realization of the random vector $Y^S = (Y, ..., Y)$, and Y^S takes values in $\mathscr{Y}^{|S|}$, the $|S|$-ary Cartesian power of $\mathscr{Y}$. Then, the NML distribution for $P_S(Y)$ equals

$$P_S^{NML}(Y^S = y^S | X^S = x^S) = \frac{\hat{P}_S(Y^S = y^S | X^S = x^S)}{\sum_{z^S \in \mathscr{Y}^S} \hat{P}_S(Y^S = z^S | X^S = x^S)}. \tag{2.7}$$

Note that $\hat{P}_S$ depends on the values of z^S. As $\hat{P}_S(Y)$ is a categorical distribution, the normalizing term can be written as $\mathcal{R}(|S|, |\mathscr{Y}|)$, a function of $|S|$—the rule's coverage—and $|\mathscr{Y}|$—the number of unique values that Y can take (Mononen and Myllymäki 2008):

$$\mathcal{R}(|S|, |\mathscr{Y}|) = \sum_{z^S \in \mathscr{Y}^S} \hat{P}_S(Y^S = z^S | X^S = x^S), \tag{2.8}$$

which can be efficiently calculated in sub-linear time (Mononen and Myllymäki 2008).

The approximate NML distribution. We propose to approximate the normalizing term of P_M^{NML} as the product of the normalizing terms of P_S^{NML} for all $S \in M$, and propose the approximate-NML distribution as our model selection criterion:

$$P_M^{apprNML}(Y^n = y^n | X^n = x^n) = \frac{P_{M,\hat{\theta}(x^n,y^n)}(Y^n = y^n | X^n = x^n)}{\prod_{S \in M} \mathcal{R}(|S|, |\mathscr{Y}|)}. \tag{2.9}$$

Note that the sum over all $S \in M$ *does* include the "else rule" S_0. Finally, we can formally define the optimal rule set M^* as

$$M^* = \arg\max_M P_M^{apprNML}(Y^n = y^n | X^n = x^n). \tag{2.10}$$

The rationale of using the approximate-NML distribution is as follows. First, it is equal to the NML distribution for a rule set without any overlap, as follows.

Proposition 1. *Given a rule set M in which for any $S_i, S_j \in M$, $S_i \cap S_j = \emptyset$, then $P_M^{NML}(Y^n = y^n | X^n = x^n) = P_M^{apprNML}(Y^n = y^n | X^n = x^n)$.*

Second, when overlaps exist in M, approximate-NML puts a small extra penalty on overlaps, which is desirable to trade-off overlap with goodness-of-fit: when we sum over all instances in each rule $S \in M$, the instances in overlaps are "repeatedly counted". Third, approximate-NML behaves like the Bayesian information criterion (BIC) asymptotically, which follows from the next proposition.

Proposition 2. *Assume M contains K rules in total, including the else rule, and we have n instances. Under the mild assumption that $|S|$ grows linearly as the sample size n for all $S \in M$, then $\log\left(\prod_{S \in M} \mathcal{R}(|S|, |\mathscr{Y}|)\right) = \frac{K(|\mathscr{Y}|-1)}{2} \log n + \mathcal{O}(1)$, where $\mathcal{O}(1)$ is bounded by a constant w.r.t. to n.*

We defer the proofs of the two propositions to the Appendix of this chapter.

2.5 Learning Truly Unordered Rule Sets from Data

As our MDL-based model selection criterion unfortunately does not enable efficient search for the optimal model, we resort to heuristics. We first address the challenge of evaluating incomplete rule sets, after which we explain how to grow individual rules in two phases and implement this with beam search. Finally, we show how everything comes together to iteratively learn rule sets from data.

2.5.1 Evaluating Incomplete Rule Sets with a Surrogate Score

When iteratively searching for the next "best" rule, defining "best" is far from trivial: rule coverage and precision are contradicting factors and typical scores therefore combine those two factors in some—more or less—arbitrary way.

This issue is further aggravated by the iterative rule learning process, in which the intermediate rule set is evaluated as an *incomplete rule set* in each step. Evaluating incomplete rule sets is a challenging task (Fürnkranz and Flach 2005), mainly because any good score needs to simultaneously consider two aspects: 1) how well do all the rules currently in the rule set describe the already covered instances; and 2) what is the "potential" for the uncovered instances, in the sense that how well can those uncovered instances be described by rules that might be added later?

Without knowing the rules that will be added later, we cannot compute the NML-based criterion for the complete rule set. Yet, we should take into account the potential of the uncovered instances. We propose to approximate the latter using a *surrogate score*, which we obtain by fitting a decision tree on the uncovered instances and using the leafs of the resulting tree as a surrogate for "future" rules. Formally, we define the tree-based surrogate score as

$$L_{\mathcal{T}}(M) = P_{M \oplus \mathcal{T}}^{apprNML}(Y^n = y^n | X^n = x^n), \tag{2.11}$$

where $M \oplus \mathcal{T}$ denotes the surrogate rule set obtained by converting the branches of $\mathcal{T}$ to rules and appending those to M (parameters are estimated as usual).

Although the branches of the decision tree learned from the currently uncovered instances may be different from the rules that will later be added to the rule set, using the tree-based surrogate score will make it easier to gradually grow good rule sets. We use decision trees because they are quick to learn and use, and the correspondence of branches to rules makes using them straightforward. We will empirically study the effects of the surrogate score on the predictive performance of rule sets in Section 2.6.

2.5.2 Two-phase Rule Growth

To avoid having to traverse all possible rules when searching for the rule to add to an incomplete rule set, we resort to a common heuristic: we start with an empty rule and gradually refine it by adding literals—also referred to as *growing* a rule (Fürnkranz et al. 2012). In contrast to existing methods, we propose a two-phase method.

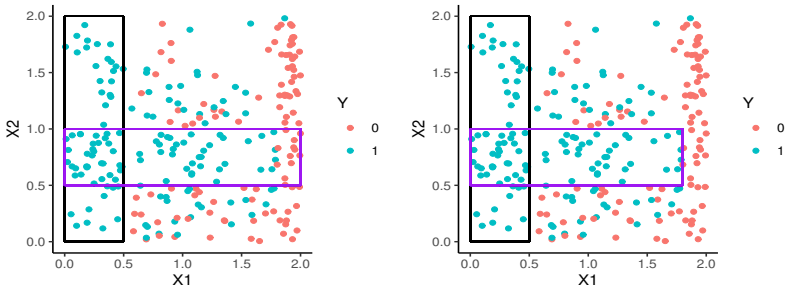

Figure 2.1: (Left) Simulated data with two overlapping rules: $S_1 : X_1 < 0.5$ (outlined in black) and $S_2 : 0.5 < X_2 < 1$ (purple). (Right) S_2 has grown to $0.5 < X_2 < 1 \wedge X_1 < 1.8$, which changes $P(Y|X \in S_2)$ and resolves the problematic overlap.

Motivation. A rule can only improve the surrogate score—and thus be added to the rule set—if it achieves two goals: 1) it should improve the likelihood of currently uncovered instances (penalized by the approximate-NML normalizing term); and 2) it should not deteriorate the goodness-of-fit of the rule set by creating "bad" overlaps. These goals can be conflicting though, for two reasons.

First, it is not necessarily bad to have overlaps between a rule being grown and the current rule set, because the rule and its probability estimates for the target variable may still change. For example, consider the left plot of Figure 2.1. If the current rule set consists of S_1 (indicated in black), then adding S_2 (in

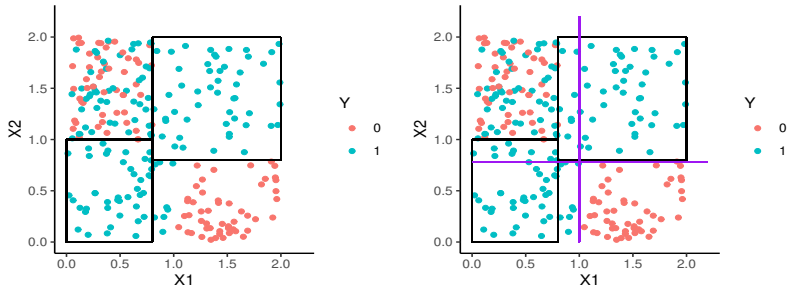

Figure 2.2: (Left) Simulated data with a rule set containing two rules (black outlines). (Right) Growing a rule to describe the bottom-right instances will create conflicts with existing rules. I.e., adding either $X_1 > 1$ (vertical purple line) or $X_2 < 0.8$ (horizontal purple line) would create a huge overlap that deteriorates the surrogate score (Eq. 2.11).

purple) would be problematic: this would strongly deteriorate the likelihood of the instances covered by both rules. However, as we further grow S_2, as shown in the right plot, we get $P(Y|S_1) = P(Y|S_2)$ and the problem is solved.

Second, rules already in the rule set may become obstacles to growing a new rule. For example, consider the data and rule set with two rules (in black) in Figure 2.2. If we want to grow a rule that covers the bottom-right instances, the existing rules form a blockade: the right plot shows how adding either $X_1 > 1$ or $X_2 < 0.8$ to the empty rule (in purple) would create a large overlap with the existing rules, with significantly different probability estimates.

Therefore, instead of navigating towards the two goals simultaneously, we propose to grow the next rule in two phases: 1) grow the rule as if the instances covered by the (incomplete) rule set are excluded; 2) further grow the rule to eliminate potentially "bad" overlaps, to further optimize the tree-based score.

Method. Given a rule S, define S_{unc} as its uncovered "counterpart", which covers all instances in S not covered by M, i.e., $S_{unc} = S \setminus \cup\{S_i \in M\}$. Then, given M, the search for the next best rule that optimizes the surrogate tree-based score is divided into two phases. First, we aim to find the m rules for which the uncovered counterparts have the highest surrogate scores, defined as

$$L_{\mathcal{T}}(M \oplus S_{unc}) = P^{apprNML}_{M \oplus S_{unc} \oplus \mathcal{T}}(Y^n = y^n | X^n = x^n), \tag{2.12}$$

where $M \oplus S_{unc} \oplus \mathcal{T}$ denotes M appended with S_{unc} and all branches of $\mathcal{T}$. Here, m is a user-specified hyperparameter that controls the number of candidate rules

Algorithm 1: Find Next Rule Ignoring Overlaps

Input: rule set M, data (x^n, y^n)
Output: A beam that contains the w best rules

RULE $\leftarrow \emptyset$; Beam $\leftarrow$ [RULE] `// Initialize the empty rule and beam`
BeamList $\leftarrow$ Beam `// Record all the beams in the beam search`
while length(*Beam*) $\neq 0$ **do**
  candidates $\leftarrow$ [ ] `// initialized to store all possible refinements`
  for *RULE* $\in$ *Beam* **do**
    Rs $\leftarrow$ [Append L to RULE for L $\in$ all possible literals]
    candidates.extend(Rs)
  Beam $\leftarrow$ the w rules in candidates that have 1) the highest positive $g_{unc}()$, and 2) coverage diversity $> \alpha$ `// w is the beam width`
  if length*(Beam)* $\neq 0$ **then**
    BeamList.extend(Beam) `// extend the BeamList as an array`
for *Rule* $\in$ *BeamList* **do**
  Beam $\leftarrow$ w rules in BeamList with best $L_{\mathcal{T}}(M \oplus S_{unc})$
return Beam

that are selected for further refinement in the second phase. In the second phase, we further grow each of these m rules to search for the best one rule that optimizes

$$L_{\mathcal{T}}(M \oplus S) = P^{apprNML}_{M \oplus S \oplus \mathcal{T}}(Y^n = y^n | X^n = x^n). \tag{2.13}$$

Given a rule S and its counterpart S_{unc}, the score of S_{unc} is an upper-bound on the score of S: if S can be further refined to cover exactly what S_{unc} covers, we can obtain $L_{\mathcal{T}}(M \oplus S_{unc}) = L_{\mathcal{T}}(M \oplus S_{unc})$. This is often not possible in practice though, and we therefore generate m candidates in the first phase (instead of 1).

2.5.3 Beam Search for Two-phase Rule Growth

In both phases we aim for growing a rule that optimizes the tree-based score (Equation 2.11); the difference is that we ignore the already covered instances in the first phase. To avoid growing rules too greedily, i.e., adding literals that quickly reduce the coverage of the rule, we use a heuristic that is based on the

NML distribution of a single rule and motivated by Foil's information gain (Cohen 1995).

Phase 1: rule growth ignoring covered instances. We propose the NML-gain to optimize $L_T(M \oplus S_{unc})$: given two rules S and Q, where we obtain S by adding one literal to Q, we define the NML-gain as $g_{unc}(S, Q)$:

$$g_{unc}(S,Q) = \left(\frac{P^{NML}_{S_{unc}}(y^{S_{unc}}|x^{S_{unc}})}{|S_{unc}|} - \frac{P^{NML}_{Q_{unc}}(y^{Q_{unc}}|x^{Q_{unc}})}{|Q_{unc}|} \right) |S_{unc}| \tag{2.14}$$

$$= \left(\frac{\hat{P}_{S_{unc}}(y^{S_{unc}}|x^{S_{unc}})}{\mathcal{R}(|S_{unc}|, |\mathscr{Y}|) \ \ |S_{unc}|} - \frac{\hat{P}_{Q_{unc}}(y^{Q_{unc}}|x^{Q_{unc}})}{\mathcal{R}(|Q_{unc}|, |\mathscr{Y}|) \ \ |Q_{unc}|} \right) |S_{unc}|, \tag{2.15}$$

which we use as the navigation heuristic.

The advantage of having a tree-based score to evaluate rules, besides the navigation heuristic (local score), is that we can adopt beam search, as outlined in Algorithm 1. We start by initializing 1) the rule as an *empty rule* (a rule without any condition), 2) the Beam containing that empty rule, and 3) the BeamRecord to record the rules in the beam search process (Line 1-2). Then, for each rule in the beam, we generate refined candidate rules by adding one literal to it (Ln 5-7). Among all candidates, we select at most w rules with the highest NML-based gain g_{unc}, satisfying two constraints: 1) $g_{unc} > 0$; and 2) for each pair of these (at most) w rules, e.g., S and Q, their "coverage diversity" $\frac{|S_{unc} \cap Q_{unc}|}{|S_{unc} \cup Q_{unc}|} > \alpha$, where α is a user-specified parameter that controls the diversity of the beam search (Van Leeuwen and Knobbe 2012). We update the Beam with these (at most) w rules (Ln 8-10). We repeat the process until we can no longer grow any rule with positive g_{unc} based on all rules in Beam (Ln 3). Last, among the record of all Beams we obtained during the process, we return the best w rules with the highest tree-based score $L(S_{unc} \cup M)$ (Ln 11-13).

Phase 2: rule growth including covered instances. We now optimize $L(M \oplus S)$ and select a rule based on the candidates obtained in the previous step. We first define a navigation heuristic: given two rules S and Q, where S is obtained by adding one literal to Q, we define the NML-gain $g(S, Q)$ as

$$g(S,Q) = \left(\frac{\hat{P}_S(y^{S_{unc}}|x^{S_{unc}})}{\mathcal{R}(|S_{unc}|, |\mathscr{Y}|) \ \ |S_{unc}|} - \frac{\hat{P}_Q(y^{Q_{unc}}|x^{Q_{unc}})}{\mathcal{R}(|Q_{unc}|, |\mathscr{Y}|) \ \ |Q_{unc}|} \right) |S_{unc}|. \tag{2.16}$$

Algorithm 2: Find Rule Set

Input: training data (x^n, y^n)
Output: rule set M

1 $M \leftarrow \emptyset$; $M\_record \leftarrow [M]$ scores $\leftarrow [P_M^{apprNML}(y^n|x^n)]$ `// Record` $P_M^{apprNML}$ `while growing`
2 **while** *True* **do**
3 $S^* \leftarrow$ FindNextRule$(M, (x^n, y^n))$ `// find the next best rule` S^*
4 **if** $S^* = \emptyset$ *or* $L_{\mathcal{T}}(M \oplus S) = P_{M \oplus S^*}^{apprNML}(y^n|x^n)$ **then**
5 **Break**
6 **else**
7 $M \leftarrow M \oplus S^*$; $M\_record.append(M)$ `// update and record` M
8 scores.append$(P_M^{apprNML}(y^n|x^n))$
9 **return** the rule set with the maximum score in M_record

Note that the difference between $g(S, Q)$ and $g_{unc}(S, Q)$ is that they use a different maximum likelihood estimator: $\hat{P}_Q$ is the ML estimator based on all instances in Q, while $\hat{P}_{Q_{unc}}$ is based on all instances in Q_{unc}.

The algorithm is almost identical to Algorithm 1, with four small modifications: 1) the navigation heuristic is replaced by $g(S, Q)$; 2) $L_{\mathcal{T}}(M \oplus S)$ is used to select the best rule from the BeamRecord instead of $L_{\mathcal{T}}(M \oplus S_{unc})$; and 3) the coverage diversity is based on the rules itself instead of the counterparts; 4) only the best rule is returned.

2.5.4 Iterative search for the rule set

Algorithm 2 outlines the proposed rule set learner. We start with an empty rule set (Ln 1-2), then iteratively add the next best rule (Ln 3–9) until the stopping criterion is met (Ln 5–6). That is, it stops when 1) the surrogate score equals the 'real' model selection criterion (i.e., the model's NML distribution), or 2) no more rules with positive NML-gain can be found. We record the 'real' criterion when adding each rule to the set, and pick the one maximizing it (Ln 10).

2.6 Experiments

We demonstrate that TURS learns rule sets with competitive predictive performance, and that using the surrogate score substantially improves the AUC scores. Further, we demonstrate that TURS achieves model complexities comparable to other rule set methods for multi-class targets.

We here discuss the most important parts of the experiment setup; for completeness, additional information can be found in the Appendix[2].

Decision trees for surrogate score. We use CART (Breiman et al. 1984) to learn the trees for the surrogate score. For efficiency and robustness, we do not use any post-pruning for the decision trees but only set a minimum sample size for the leafs.

Beam width and coverage diversity. We set the coverage diversity $\alpha = 0.05$, and beam width $w = 5$. With the coverage diversity as a constraint, we found that $w \in \{5, 10, 20\}$ all give similar results. Due to the limited space, we leave a formal sensitivity analysis of α as future work.

Benchmark datasets and competitor algorithms. We test on 13 UCI benchmark datasets (shown in Table 1), and compare against the following methods: 1) unordered CN2 (Clark and Boswell 1991), the one-versus-rest rule sets method without implicit order among rules; 2) DRS (Zhang and Gionis 2020), a representative multi-class rule set learning method; 3) BRS (Wang et al. 2017), the Bayesian rule set method for binary classification; 4) RIPPER (Cohen 1995), the widely used one-versus-rest method with orders among class labels; 5) CLASSY (Proença and Leeuwen 2020), the probabilistic rule list methods using MDL-based model selection; and 6) CART (Breiman et al. 1984), the well-known decision tree method, *with* post-pruning by cross-validation.

2.6.1 Results

Predictive performance. We report the ROC-AUC scores in Table 2.1. For multi-class classification, we report the weighted one-versus-rest AUC scores, as was also used for evaluating the recently proposed CLASSY method (Proença and Leeuwen 2020).

[2]The source code is available at `https://github.com/ylincen/TURS`

data	TURS	CN2	DRS	BRS	CLASSY	RIPPER	CART	TURS %overlap
anuran	0.998	1.000	0.858	—	0.983	0.999	0.996	0.395
avila	0.968	0.978	0.530	—	0.954	0.997	0.988	0.286
backnote	**0.991**	0.969	0.945	0.957	0.987	0.979	0.984	0.297
car	**0.978**	0.633	0.924	—	0.945	0.980	0.971	0.063
chess	**0.995**	0.536	0.823	0.945	0.991	0.995	0.994	0.264
contracept	**0.667**	0.597	0.544	—	0.630	0.626	0.600	0.074
diabetes	**0.766**	0.677	0.628	0.683	0.761	0.735	0.661	0.155
ionosphere	0.914	0.912	0.663	0.837	0.909	0.901	0.845	0.310
iris	0.964	0.985	0.935	—	0.960	0.973	0.965	0.018
magic	**0.886**	0.590	0.695	0.794	0.895	0.818	0.800	0.500
tic-tac-toe	0.972	0.826	0.971	0.976	0.983	0.954	0.847	0.231
waveform	**0.902**	0.775	0.588	—	0.833	0.884	0.803	0.528
wine	0.954	0.962	0.810	—	0.961	0.945	0.932	0.031
Avg Rank	2.231	4.077	5.846	5.462	3.154	3.000	4.231	/

Table 2.1: ROC-AUC scores, averaged over 10 cross-validated folds. The rank (smaller means better) is further averaged over all datasets. Among the four *rule set* methods, TURS is substantially better on 7 out 13 datasets (AUC scores in bold).

Compared to non-probabilistic rule set methods—i.e., CN2, DRS, and BRS (only for binary targets)—TURS is much better in terms of the mean rank of its AUC scores. Specifically, it performs substantially better on about half of the datasets (shown in bold). Besides, it is ranked better than rule list methods, which produce explicitly ordered rules that may be difficult for domain experts to comprehend and digest in practice. Next, CART attains AUCs generally inferior to TURS, although it helps TURS to get a higher AUC as part of the surrogate score.

Last, we report the percentage of instances covered by more than one rule for TURS in Table 2.1, and we show that overlaps are common in the rule sets obtained for different datasets. This empirically confirms that our way of formalizing rule sets as probabilistic models, i.e., treating overlaps as uncertainty and exception, can indeed lead to improved predictive performance, as the overlapping rules are a non-negligible part of the model learned from data and hence indeed play a role.

Effects of the surrogate score. Figure 2.3 shows the difference in AUC obtained by our method with and without using the surrogate score (i.e., without surrogate score means replacing it with the final model selection criterion). We conclude that the surrogate score has a substantial effect on learning better rule sets, except for three "simple" datasets, of which the sample sizes and the number

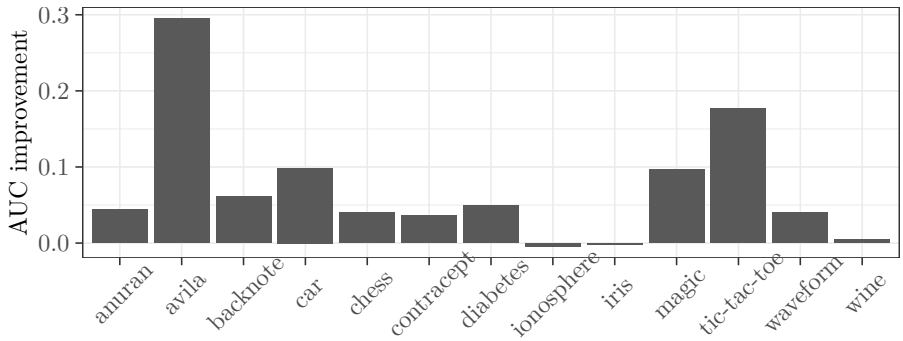

Figure 2.3: Improvement in AUC by enabling the surrogate score for TURS.

#instances	#features	data	TURS	CN2	DRS	BRS	CLASSY	RIPPER	CART
1372	5	backnote	42	41	55	22	22	16	94
1473	10	contracept	75	275	73	—	14	14	6241
768	9	diabetes	55	152	131	10	10	6	827
150	5	iris	7	9	23	—	3	3	9
958	10	tic-tac-toe	86	90	108	24	27	60	816
178	14	wine	10	6	134	—	6	5	15
1728	7	car	211	163	325	—	92	111	718
7195	24	anuran	74	37	407	—	49	7	96
3196	37	chess	299	316	482	21	37	44	355
351	35	ionosphere	50	30	261	14	6	5	101
5000	22	waveform	707	802	60	—	139	115	3928
20867	11	avila	890	1296	179	—	988	574	8145
19020	11	magic	1321	2238	48	23	256	69	22566
		Avg Rank	2.15	2.46	2.77	1.00	—	—	—

Table 2.2: Left: The sample sizes and number of features of datasets. Right: total number of literals, i.e., average rule lengths × number of rules in the set, averaged over 10-fold cross-validation. The rank is averaged over all datasets, for rule sets methods only.

of variables are small, as shown in Table 2.2 (Left).

Model complexity. Finally, we compare the 'model complexity' of the rule sets for all methods. As this is hard to quantify in a unified manner, as a proxy we report the *total number of literals in all rules in a rule set*, averaged over 10-fold cross-validation (the same as used for the results reported in Table 2.1).

We show that among all rule set methods (TURS, CN2, DRS, BRS), TURS has better average ranks than both CN2 and DRS. Although BRS learns very small rule sets, it is only applicable to binary targets and its low model complexity also brings worse AUC scores than TURS. Further, although rule list methods (CLASSY, RIPPER) generally have fewer literals than rule sets methods, this

does not make rule lists easy to interpret, as every rule depends on all previous rules. Last, we empirically confirm that tree-based method CART produces much larger rule sets.

2.7 Conclusion

We formalized the problem of learning truly unordered probabilistic rule sets as a model selection task. We also proposed a novel, tree-based surrogate score for evaluating incomplete rule sets. Building upon this, we developed a two-phase heuristic algorithm that learns rule set models that were empirically shown to be accurate in comparison to competing methods.

For future work, we will study the practical use of our method with a case study in the health care domain. This involves investigating how well our method scales to larger datasets. Furthermore, a user study will be performed to investigate whether, and in what degree, the domain experts find the truly unordered property of rule sets obtained by our method helps them comprehend the rules better in practice, in comparison to rule lists/sets with explicit or implicit orders.

2.8 Appendix I: Reproducibility for Experiments

Decision trees for surrogate score. We use a CART decision tree (Breiman et al. 1984) to get the tree-based surrogate score. For efficiency and robustness, we do not use any post-pruning for the decision tree but only set the minimum sample size on leafs, denoted as s. Specifically, we try $s \in \{10, 30, 50, 70, 90\}$ and hence calculate five surrogate scores accordingly, among which we pick the smallest as the final surrogate score.

Beam width and coverage diversity. We set the coverage diversity $\alpha = 0.05$, and beam width $w = 5$. With the coverage diversity as a constraint, we found that $w \in \{5, 10, 20\}$ gives similar results. Due to the limited space, we leave formal sensitivity analysis of α as future work.

Number of cut points for numeric features. To generate literals for numeric features, we need to decide the number of cut points for these features. In practice, it should depend on how the analysts want to interpret the resulting rules: given a specific task, is it useful to be more precise than the granularity of the 10- or 20-quantiles? Intuitively, we believe it is seldom necessary to be more precise than 100-quantiles, and hence we set the number of cut points as 100.

Benchmark datasets and competitor algorithms. For reproducibility, we use the implementation of CN2 from Orange3 (Demšar et al. 2013), RIPPER from RWeka (Hornik et al. 2009), CART from Sklearn (Pedregosa et al. 2011), and BRS and DRS from the authors' original implementation. Most parameters are set as "default" based on the implementation. For BRS and DRS, this means that we use what the author suggested in the original papers. Specifically, for CART, we use the post-pruning for trees with the regularization parameter chosen from cross-validation.

2.9 Appendix II: Proof of Proposition 1

Proposition 1: *Given a rule set M in which for any $S_i, S_j \in M$, $S_i \cap S_j = \emptyset$, then $P_M^{NML}(Y^n = y^n | X^n = x^n) = P_M^{apprNML}(Y^n = y^n | X^n = x^n)$.*

Proof. The numerators are the same, and hence we only need to show that the denominators are the same. Assume there are K rules in M in total,

$$\begin{aligned}
&\sum_{z^n \in \mathscr{Y}^n} P_{M,\hat{\theta}(x^n,z^n)}(z^n|x^n) = \sum_{z^n} \prod_{S\in M} \hat{P}_S(y^S|X^S) \\
&= \sum_{z^n} \hat{P}_{S_1}(z^{S_1}|x^{S_1}) \dots \hat{P}_{S_K}(z^{S_K}|x^{S_K}) \\
&= \sum_{z^{S_1}} \dots \sum_{z^{S_K}} \left(\hat{P}_{S_1}(z^{S_1}|x^{S_1}) \dots \hat{P}_{S_K}(z^{S_K}|x^{S_K}) \right) \\
&= \left(\sum_{z^{S_1}} \dots \sum_{z^{S_{K-1}}} \hat{P}_{S_1}(z^{S_1}|x^{S_1}) \dots \hat{P}_{S_{K-1}}(z^{S_{K-1}}|x^{S_{K-1}}) \right) \left(\sum_{z^{S_K}} \hat{P}_{S_K}(z^{S_K}|x^{S_K}) \right) \\
&\dots \\
&= \left(\sum_{z^{S_1}} \hat{P}_{S_1}(z^{S_1}|x^{S_1}) \right) \dots \left(\sum_{z^{S_K}} \hat{P}_{S_K}(z^{S_K}|x^{S_K}) \right) \\
&= \prod_{S\in M} \sum_{z^S} \hat{P}_S(z^S|x^S) \\
&= \prod_{S\in M} \mathcal{R}(|S|, |\mathscr{Y}|),
\end{aligned} \tag{2.17}$$

which completes the proof. □

2.10 Appendix III: Proof of Proposition 2

Proposition 2: *Assume M contains K rules in total, including the else rule, and we have n instances. Under the mild assumption that $|S|$ grows linearly as the sample size n for all $S \in M$, then* $\log\left(\prod_{S\in M} \mathcal{R}(|S|, |\mathscr{Y}|)\right) = \frac{K(|\mathscr{Y}|-1)}{2} \log n + \mathcal{O}(1)$, *where $\mathcal{O}(1)$ is bounded by a constant w.r.t. to n.*

Proof. The proof directly follows from Theorem 3 of (Silander et al. 2008). Firstly, it has been proven that $\log \mathcal{R}(|S|, |\mathscr{Y}|) = \frac{|\mathscr{Y}|-1}{2} \log |S| + \mathcal{O}(1)$ (Rissanen 1996). Next, under the mild assumption that $|S|$ grows linearly as the full sample size n, we have $\log |S| = \log((\gamma + o(1))n) = \log n + \mathcal{O}(1)$. Hence, $\log \prod_{S\in M} \mathcal{R}(|S|, |\mathscr{Y}|) = \sum_S \log \mathcal{R}(|S|, |\mathscr{Y}|) = \frac{K(|\mathscr{Y}|-1)}{2} \log n + \mathcal{O}(1)$, which completes the proof. □

Chapter 3

Probabilistic Truly Unordered Rule Sets

This chapter consists of a paper titled *Probabilistic Truly Unordered Rule Sets* (submitted to JMLR), which describes a refined version of the TURS model (in comparison to Chapter 2).
For being self-contained, Chapter 3 inevitably contains some repeated content from Chapter 2, including notation descriptions, basic definitions, and some identical related work discussions. However, the definition, model selection criterion, and algorithm for learning a TURS model are all refined based on Chapter 2. The differences between Chapter 2 and 3 are briefly discussed at the end of Section 3.1, and more thoroughly in the Appendix of this chapter.

Chapter Abstract

Rule set learning has been frequently revisited because of its interpretability. Existing methods have several shortcomings though. First, most existing methods impose orders among rules, either explicitly or implicitly, which makes the models less comprehensible. Second, due to the difficulty of handling conflicts caused by overlaps (i.e., instances covered by multiple rules), existing methods often do not consider *probabilistic* rules. Third, learning classification rules for multi-class target is understudied, as most existing methods focus on binary classification or multi-class classification via the "one-versus-rest" approach.

To address these shortcomings, we propose TURS[1], for Truly Unordered Rule Sets. To resolve conflicts caused by overlapping rules, we propose a novel model that exploits the probabilistic properties of our rule sets, with the intuition of only allowing rules to overlap if they have similar probabilistic outputs. We next formalize the problem of learning a TURS model based on the MDL principle and develop a carefully designed heuristic algorithm. We benchmark against a wide range of rule-based methods and demonstrate that our method learns rule sets that have lower model complexity and highly competitive predictive performance. In addition, we empirically show that rules in our model are empirically "independent" and hence truly unordered.

[1] A refined version based on Chapter 2.

3.1 Introduction

Despite the great success of black-box models in a wide range of tasks, intrinsically interpretable machine learning models have also received a lot of attention due to their transparency and hence their applicability to sensitive real-world scenarios, such as health care and judicial systems (Rudin 2019). We particularly focus on modelling and learning probabilistic rule sets for multi-class classification.

A probabilistic rule is in the form of **IF** X **meets certain conditions, THEN** $P(Y) = \hat{P}(Y)$, in which X represents the feature variables, Y the target variable, and $\hat{P}$ the associated class probability estimator.

Rule-based methods have the unique advantage that they are not only accessible and interpretable to statisticians and data scientists but also to domain experts, since rules can be directly read. While a single rule summarizes a local pattern from the data and hence only describes a subset of the instances, existing rule-based methods adopt various approaches to put individual rules together to form a global predictive model.

For instance, rule lists (or decision lists) (Fürnkranz et al. 2012) connect all individual rules by the "**IF ... (possibly multiple) ELSE IF ... ELSE ...**" statement, which is equivalent to specifying an explicit order for each rule. This approach is compatible with the very efficient divide-and-conquer algorithms: when a rule is induced from data, the covered instances (i.e., instances satisfying the condition of the rule) can be removed and hence iteratively simplify the search space. While this approach is very efficient, it comes at the cost of interpretability. As the condition of each rule depends on all preceding rules, comprehending a single rule requires going over (the negations of) all preceding rules' conditions, which is impractical when the rule list becomes large.

On the other hand, rule set models put rules together without specifying explicit orders. In this case, an instance can be covered by one single rule or simultaneously by multiple rules. When the instances covered by two or more rules have intersections, we say that these rules *overlap*. Although existing rule set methods claim that individual rules in rule sets are unordered (Clark and Boswell 1991; Kotsiantis 2013; Van Leeuwen and Knobbe 2012), we argue that they are not truly unordered. In fact, when one instance is covered by multiple rules at the same time, different rules may give conflicting (probabilistic) predictions. As a

result, ad-hoc schemes are widely used to resolve the conflicts, typically by ranking the involved rules with certain criteria (e.g., accuracy) and always selecting the highest ranked rule (Lakkaraju et al. 2016; Zhang and Gionis 2020). This approach, however, imposes implicit orders among rules, making rules entangled instead of truly unordered.

Implicit orders in rule sets severely harm interpretability, especially from the perspective of comprehensibility. While no agreement has been reached on the precise definition of interpretability of machine learning models (Molnar 2020; Murdoch et al. 2019), we specifically treat interpretability with domain experts in mind. In particular, to explain a single prediction for an instance to domain experts when implicit orders exist, it is insufficient to only provide the rules that the instance satisfies, because other *higher-ranked* rules that the instance does *not* satisfy are also a necessary part of the explanation. For example, imagine a patient is predicted to have *Flu* because they have *Fever*. If the model also contains the higher-ranked rule *"Blood in stool* $\rightarrow$ *Dysentery"*, the explanation should include the fact that *"Blood in stool"* is not true, because otherwise the prediction would change to *Dysentery*. If the model contains many rules, however, it becomes impractical to go over all higher-ranked rules for each prediction.

Additionally, decision trees, which can broadly be viewed as a rule-based approach, often have rules (root-to-leaf paths) that share multiple attributes due to their inherent structure. This can result in overly lengthy rules (as we will also empirically demonstrate in the Experiment section), since some internal nodes may not contribute to the classification itself but only serve to maintain the tree structure. Thus, decision trees are often less compact than decision rules, and consequently it is more difficult for domain experts to grasp the internal decision logic, and hence also the explanations for single predictions.

Given these shortcomings of existing rule-based models, we introduce *truly unordered rule sets (TURS)*, with the following properties. First, unlike most recently proposed rule sets/lists methods that only predict labels as outputs (Dash et al. 2018; Wang et al. 2017; Yang et al. 2021; Yang et al. 2017), we aim for formalizing a set of rules as a probabilistic model in a principled way. Since rule-based methods are potentially most applicable in sensitive areas, probabilistic predictions are much more useful for decision making and knowledge discovery, especially when domain experts are responsible for taking actions, such as in

health care. Probabilistic rules also allow us to directly apply our model for multi-class classification tasks, without leveraging the commonly used "one-versus-rest" paradigm (Clark and Boswell 1991; Hühn and Hüllermeier 2009). Second, we aim to develop a method to learn a set of probabilistic rules without implicit orders: to achieve this, we "allow" rules to overlap only if they have similar probabilistic outputs. In this case, when one instance is covered by multiple rules, it does not matter much even if we randomly pick one of these rules for prediction, since the differences of the prediction given by each individual rule is controlled. Thus, each rule becomes "self-standing" and can be used for explaining the predictions alone.

Particularly, we formally define a *truly unordered rule set* (TURS) as a probabilistic model, i.e., given a TURS model denoted as M and a dataset D, the likelihood of the target values conditioned on the feature values is defined. Notably, without putting implicit orders among rules, instances covered by multiple rules are modelled in a subtle manner such that the resulting likelihood is "penalized" if these overlapping rules have very different probabilistic outputs. Thus, we leverage our formal definition of TURS model and incorporate the probabilistic output differences into the goodness-of-fit of our probabilistic model, *without* introducing any hyper-parameter to control the probabilistic output differences of overlapping rules. Further, we treat the problem of learning a TURS model from data as a probabilistic model selection task, and hence further design a model selection criterion based on the minimum description length (MDL) principle (Grünwald 2007; Grünwald and Roos 2019), which does not require a regularization parameter to be tuned.

We resort to heuristics for optimization as the search space combined with the model selection criterion do not allow efficient search. Yet, we carefully and extensively extend the common heuristic approach for learning decision rules from data, in the following aspects. First, we consider a "learning speed score" heuristic, i.e., the decrease of our optimization score (to be minimized) *per extra covered instance* as the quality measure for searching the next "best" rule. Second, we take a novel beam search approach, such that 1) the degree of "patience" is considered by using a diverse beam search approach, and 2) an auxiliary beam together with a "complementary" score is proposed, in order to resolve the challenge that rules that have been added to the rule set may become obstacles for new rules. This challenge comes along with the fact that, unlike existing rule set methods, we

do consider overlaps of rules in the process of learning rules from data. Third, we propose an *MDL-based local testing method* in order to characterize whether the "left out" instances during the process of refining a rule can be well covered by rules we search for later. That is, while existing heuristics in rule learning only characterize the "quality" of the individual rules in different ways, our local testing criterion can be regarded as a look-ahead strategy.

In summary, our main contributions in this chapter are as follows:

1. In contrast to most recently proposed rule lists/sets methods that focus on non-probabilistic modelling and binary classification, we propose a principled way to formalize rule sets as probabilistic models that arguably provides more transparency and uncertainty information to domain experts in sensitive areas such as health care. It can also handle multi-class classification naturally, without resorting to the one-versus-rest scheme.

2. While existing rule sets methods adopt an ad-hoc approach to deal with conflicts caused by overlaps, often by always following the rule that scores the best according to a pre-defined criterion (e.g., accuracy or F-score), we identify that this approach puts implicit orders among rules that can severely harm interpretability. To tackle this issue, we propose to only "allow" overlaps that are formed by rules with similar probabilistic outputs. We formally define the TURS model, for Truly Unordered Rule Sets, in a way such that the probabilistic output difference among overlapping rules is incorporated in the goodness-of-fit as measured by the likelihood.

3. We formalize the problem of learning a TURS model from data as a probabilistic model selection task. We further propose an MDL-based model selection criterion that automatically handles the trade-off between the goodness-of-fit and model complexity, without any hyper-parameters to be tuned by the time-consuming cross-validation.

4. We develop a heuristic optimization algorithm with considerable algorithmic innovations. We benchmark our model TURS together with the proposed algorithm with extensive empirical comparisons against a wide range of rule-based methods. We show that TURS has superior performance in the following aspects: 1) it has very competitive predictive performance (measured by ROC-AUC); 2) it can *empirically* learn truly unordered rules: the

probabilistic conflicts caused by overlaps are negligible, in the sense that the influence is little even if we predict for instances covered by multiple rules by randomly picking one single rule from these rules; 3) TURS learns a set of rules with class probability estimates that can generalize well to unseen data; and 4) it produces simpler models in comparison to competitor algorithms.

Comparison with our previous work. This chapter is based on the previous chapter (Chapter 2), with vast extensions and modifications in all components, including probabilistic modelling, model selection criterion, algorithmic approach, and experiments. We summarize the key difference points between this chapter and the previous chapter as follows. First of all, we developed a completely new algorithm, with 1) a learning-speed-score heuristic motivated by the "normalized gain" used in the CLASSY algorithm for rule lists (Proença and Leeuwen 2020); 2) a diverse beam search approach with diverse "patience", in which the concept of patience is taken from the PRIM method (Friedman and Fisher 1999) for regression rules; 3) an innovative extension to the normal beam search approach, in the sense that we propose to use an auxiliary beam together with the "main" beam (and hence we simultaneously keep two beams); and 4) an MDL-local-test that serves as a look-ahead strategy for instances that are not covered for now. Further, we substantially extend the experiments in various aspects, and we now demonstrate that we can empirically treat the rule sets induced from data as truly unordered, in the sense that if a instance is covered by multiple rules we can now randomly pick one single rule for prediction, with negligible influence on the predictive performance (measured by ROC-AUC). Lastly, we also make a moderate modification to our optimization score. We discuss all these differences more in detail in the Appendix.

Organization of the chapter. The remainder of the chapter is structured as follows. In Section 3.2 we review related work. In Section 3.3 we present how to formalize a rule set as a probabilistic model, with the key component of how to model the instances covered by overlaps, i.e., by multiple rules at the same time. In Section 3.4, we discuss our model selection approach for learning a the truly unordered rule set, and formally define the model selection criterion based on the minimum description length (MDL) principle. In Section 3.5, we motivate and discuss our heuristics for learning the rule sets, and next present our proposed algorithm. Finally, we discuss our experiment setup and demonstrate

our experiment results in Section 3.6.

Algorithm	Model type	Rule learning strategy	Probabilistic	Handle overlap conflicts
CBA	ordered rule list	divide and conquer	✓	explicit order
CN2-ordered	ordered rule list	divide and conquer	✓	explicit order
PART	ordered rule list	divide and conquer	✓	explicit order
CLASSY	ordered rule list	divide and conquer	✓	explicit order
RIPPER	ordered list of rule sets	divide and conquer	×	explicit order
C4.5 rules	ordered list of rule sets	one-versus-rest	×	explicit order
BRS	rule set (binary target)	rules for positive class only	×	no conflict
CG	rule set (binary target)	rules for positive class only	×	no conflict
Submodular	rule set (binary target)	rules for positive class only	×	no conflict
CN2-unordered	rule set	one-versus-rest	✓	ad-hoc (weighted average)
FURIA	fuzzy rule set	one-versus-rest	✓	fuzzy (weighted average)
CMAR	rule set	association rule mining	×	ad-hoc (implicit orders, χ^2)
CPAR	rule set	association rule mining	×	ad-hoc (implicit orders, accuracy)
IDS	rule set	optimization for multi-class target	×	ad-hoc (implicit orders, F1-score)
DRS	rule set	optimization for multi-class target	×	ad-hoc (implicit orders, accuracy)
TURS (ours)	**truly unordered rule set**	**optimization for multi-class target**	✓	**Not needed**

Table 3.1: Summary of the algorithms' key properties.

3.2 Related Work

We next review the related work and we categorize them as follows. First, we discuss rule list methods, in which no overlap among rules exists by definition. Second, we review previous methods that learn rules for a single class labels, and based on it, the one-versus-rest rule learning methods. Last, we discuss rule sets methods for multi-class targets, as well as several different but related methods such as association rule mining. We summarize the key properties of closely related methods in Table 3.1.

Rule lists. Rules in a rule list are connected by IF-THEN-ELSE statements, and hence are with explicit orders. When classifying an instance, rules in the rule list are checked sequentially: once a rule is found of which the condition is satisfied by the instance, that single rule is used for prediction. Existing methods include CBA (Liu et al. 1998), ordered CN2 (Clark and Niblett 1989), PART (Frank and Witten 1998), and the more recently proposed CLASSY (Proença and Leeuwen 2020) and Bayesian rule list (Yang et al. 2017). Although these methods are often efficient by leveraging the divide-and-conquer (i.e., sequential covering) approach, rule lists are more difficult to interpret than rule sets because of their explicit orders. To comprehend the conditions of each rule, conditions in all preceding rules must also be taken into account; thus, the condition of each individual rule may not be meaningful when domain experts examine it separately (except for the first one).

One-versus-rest rule learning. This category focuses on only learning rules for a single class label, i.e., the "positive" class, which is already sufficient for binary classification (Dash et al. 2018; Quinlan 1990; Wang et al. 2017; Yang et al. 2021). That is, if an instance satisfies at least one of the induced rules, it can be classified as "positive", and otherwise negative. As all rules output the "positive" class, no prediction conflicts exist by definition. Recently, this line of research focuses on adopting discrete optimization techniques with provably better theoretical properties than heuristic algorithms; however, they suffer from the following drawbacks. First, these methods are non-probabilistic by definition, and hence it is not clear how to estimate the class probability for the instances covered by multiple rules (i.e., in the overlap). Second, no explicit explanation exists for those instances that are classified into the "negative" class; instead, the explanations for the negative class depend on the negation of all rules for the positive class, which can be overly complicated to comprehend when the number of rules is large. Third, these methods require discretizing and binarizing the feature matrix, and hence can only afford rather coarse search granularity for continuous-valued features, due to the high memory cost.

Further, learning rules for a single class label can be extended to multi-class classification, through the one-versus-rest paradigm. Existing methods mostly take the following two approaches to achieve this. The first, taken by RIPPER (Cohen 1995) and the C4.5 decision rule method (Quinlan 2014), is to learn each class in a certain order. After all rules for a single class have been learned, all covered instances are removed (or those with this class label). The resulting model is essentially an ordered list of rule sets, and hence is more difficult to interpret than a rule set.

The second approach does no impose an order among the classes; instead, it learns a set of rules for each class against all other classes. The most well-known are unordered-CN2 and FURIA (Clark and Boswell 1991; Hühn and Hüllermeier 2009). FURIA avoids dealing with conflicts of overlaps by essentially using all (fuzzy) rules for predicting unseen instances; i.e., the rules' outputs are weighted by the so-called "certainty factor". As a result, it cannot provide a single rule to explain its prediction. Unordered-CN2, on the other hand, handles overlaps by estimating the class probability as the weighted average of the class probability estimates for all individual rules involved in the overlap. That is, unlike our

method, CN2 adopts a post-hoc conflict resolving scheme, and as a result the issue of probabilistic conflicts is ignored during the training phase of CN2.

Multi-class rule sets. Very few methods exist for formalizing learning rules for multi-class classification as an optimization task directly (like our method), which leads to algorithmically more challenging tasks than the one-versus-rest paradigm, as the separate-and-conquer strategy is not applicable. To the best of our knowledge, the only existing methods are IDS (Lakkaraju et al. 2016) and DRS (Zhang and Gionis 2020). Both are neither probabilistic nor truly unordered. To handle conflicts of overlaps, IDS follows the rule with the highest F1-score, and DRS uses the most accurate rule. As we elaborated in Section 3.1, this approach imposes implicit orders and thus harms the comprehensibility of the model.

Decision trees and association rules. Other related approaches include the following. To begin with, decision tree based methods such as CART (Breiman et al. 1984) and C4.5 (Quinlan 2014) produce rules that are forced to share many "attributes" and hence are longer than necessary, as we will empirically demonstrate in Section 3.6.

Besides, a large category of methods is associative rule classification, which is to build rule-based classifiers based on existing association rule mining algorithms (Abdelhamid and Thabtah 2014). Association rule mining is known to have the problem of inducing redundant rules (Chen et al. 2006), hence a single instance can be easily covered by potentially many rules at the same time. As a result, various ad-hoc schemes have been proposed for handling the prediction conflicts of rules.

For instance, CMAR (Li et al. 2001) first groups rules based on their (different) predicted class labels for a given instance, and next constructs a contingency table for the whole dataset based on 1) whether an instance is covered by the group of rules and 2) the class label of each instance. Then the group of rules (and hence the conflicting class labels) is ranked with the χ^2 statistic. Moreover, CPAR (Yin and Han 2003) extends the sequential covering approach taken by FOIL (Quinlan 1990): instead of removing covered instances, the weights of covered instances are downgraded, in order to guide the search algorithm to focus on uncovered instances, and then resolves the prediction conflicts based on ranking the rules with the expected accuracy.

Lastly, the 'lazy learning' approach for associative rule classification, which

focuses on learning a single rule for every test (unseen) instance separately with a given training set of instances, can also avoid the conflicts of overlaps (Veloso et al. 2006). As a result, the lazy learning approach will not construct a global rule set model that describes the whole dataset, and hence provide less transparency for domain experts than our method.

3.3 Truly Unordered Rule Sets

We first formalize individual rules as *local* probabilistic models, and then define rule sets as *global* probabilistic models. The key challenge lies in how to define $P(Y = y|X = x)$ for an instance (x, y) that is covered by multiple rules.

3.3.1 Probabilistic rules

Denote the input random variables by $X = (X_1, \ldots, X_d)$, where each X_i is a one-dimensional random variable representing one dimension of X, and denote the categorical target variable by Y together with its domain $\mathscr{Y}$ that contains all unique class labels. Further, denote the dataset from which the rule set can be induced as $D = \{(x_i, y_i)\}_{i\in[n]}$, or (x^n, y^n) for short. Each (x_i, y_i) is an instance. Then, a probabilistic rule S is written as

$$(X_1 \in R_1 \wedge X_2 \in R_2 \wedge \ldots) \rightarrow P_S(Y), \tag{3.1}$$

where each $X_i \in R_i$ is called a *literal* of the *condition* of the rule. Specifically, each R_i is an interval (for a quantitative variable) or a set of categorical levels (for a categorical variable).

A probabilistic rule of this form describes a subset S of the full sample space of X, such that for any $x \in S$, the conditional distribution $P(Y|X = x)$ is approximated by the probability distribution of Y conditioned on the event $\{X \in S\}$, denoted as $P(Y|X \in S)$. Since in classification Y is a discrete variable, we can parametrize $P(Y|X \in S)$ by a parameter vector $\vec{\beta}$, in which the jth element β_j represents $P(Y = j|X \in S)$, for all $j \in \mathscr{Y}$. We therefore denote $P(Y|X \in S)$ as $P_{S,\vec{\beta}}(Y)$, or $P_S(Y)$ for short. To estimate $\vec{\beta}$ from data, we adopt the maximum likelihood estimator, denoted as $P_{S,\hat{\vec{\beta}}}(Y)$, or $\hat{P}_S(Y)$ for short.

Further, if an instance (x, y) satisfies the condition of rule S, we say that

(x, y) is *covered* by S. Reversely, the *cover* of S denotes the instances it covers. When clear from the context, we use S to *both represent the rule itself and/or its cover*, and define the number of covered instances $|S|$ as its *coverage*.

3.3.2 The TURS model

We aim for defining a rule set model with the following properties. First, each individual rule can be regarded as a reliable local pattern and generalizable probabilistic model that can serve as an explanation for the model's predictions. Second, if certain rules overlap with each other, i.e., some instances are covered by multiple rules simultaneously, then the probabilistic outputs of these rules "must be similar enough", in the sense that the likelihood of a TURS model given a fixed dataset incorporates (and penalizes) the differences of probabilistic outputs of overlapping rules.

Given a rule set with K individual rules, denoted as $M = \{S_i\}_{i \in [K]}$, any instance (x, y) falls into one of three cases: 1) exactly one rule covers x; 2) at least two rules cover x; and 3) no rule in M covers x. We formally define the model M as follows.

Covered by one rule only. Given a single rule denoted as S, when $x \in S, S \in M$ and $x \notin M \setminus S$, we define

$$P(Y|X = x) = P(Y|X \in S) = P_S(Y), \ \ \forall x \in S, x \notin M \setminus S \tag{3.2}$$

in which $P_S(Y)$ can be estimated from data. That is, we use $P_S(Y)$ to "approximate" the conditional probability $P(Y|X = x)$. To estimate $P_S(Y)$ we adopt the maximum likelihood (ML) estimator based on all instances covered by S. We define the ML estimator as $\hat{P}_S(Y)$, and let

$$\hat{P}_S(Y = j) = \frac{|\{(x, y) : x \in S, y = j\}|}{|S|}, \forall j \in \mathscr{Y}. \tag{3.3}$$

Note that we intentionally *do not exclude* instances in S that are also covered by other rules (i.e., in overlaps) for estimating $P_S(Y)$. Hence, the probability estimation for each rule is independent of other rules; as a result, each rule is *self-standing*, which forms the foundation of a truly unordered rule set.

Covered by multiple rules. For the second case when $x \in \bigcap_{i \in I} S_i, I \subseteq [K]$, we

define

$$P(Y|X = x) = P(Y|X \in \bigcup_{i \in I} S_i), \quad \forall x \in \bigcap_{i \in I} S_i, I \subseteq [K] \tag{3.4}$$

in which $P(Y|X \in \bigcup_{i \in I} S_i)$ is to be estimated from data with the ML estimator, defined and denoted as

$$\hat{P}(Y = j|X \in \bigcup_{i \in I} S_i) = \frac{|\{(x, y) : x \in \bigcup_{i \in I} S_i, y = j\}|}{|\bigcup_{i \in I} S_i|}, \forall j \in \mathscr{Y}. \tag{3.5}$$

Note that we take the union $\bigcup_{i \in I} S_i$ for the instances covered by the overlap (i.e., intersection) $\bigcap_{i \in I} S_i$. As counter-intuitive as it may seem at first glance, this subtle definition plays a key role in our modelling: with this novel definition, the likelihood of the data given the model—as the measure of the model's goodness-of-fit—automatically incorporates the differences between the rules' probabilistic outputs if they form an overlap.

Without loss of generalization, consider two rules denoted as S_i and S_j. When $P_{S_i}(Y)$ and $P_{S_j}(Y)$ are very similar, the conditional probability conditioned on the event $\{S_i \cup S_j\}$, denoted as $P(Y|S_i \cup S_j)$, will also be similar to both $P_{S_i}(Y)$ and $P_{S_j}(Y)$. In this case, it does not matter which of these three (i.e., $P_{S_i}(Y)$, $P_{S_j}(Y)$, or $P(Y|S_i \cup S_j)$) we use to model $P(Y|X = x), \forall x \in S_i \cap S_j$, in the sense that the "goodness-of-fit" measured by the likelihood of all instances covered by the overlap $S_i \cap S_j$ would be all similar.

On the other hand, when $P_{S_i}(Y)$ and $P_{S_j}(Y)$ are very different, the goodness-of-fit would be poor when using $P(Y|S_i \cup S_j)$ for estimating $P(Y|X = x)$ for $x \in S_i \cap S_j$. Thus, we leverage this property to penalize "bad" overlaps by incorporating the probabilistic goodness-of-fit in our model selection criterion that will be discussed in detail in Section 3.4.

Covered by no rule. When no rule in M covers x, we say that x belongs to the so-called "else rule" that is part of every rule set and equivalent to $x \notin \bigcup_i S_i$. Thus, we approximate $P(Y|X = x)$ by $P(Y|X \notin \bigcup_i S_i)$. We denote the else rule by S_0, which is the only rule in every rule set that depends on the other rules and is therefore not self-standing; however, it will also have no overlap with other rules by definition.

TURS as a probabilistic model. Building upon our definition for modelling the conditional class probability and the maximum likelihood estimators, we can

now formally define truly unordered rule sets as probabilistic models. Formally, a rule set M as a probabilistic model is a family of probability distributions, denoted $P_{M,\theta}(Y|X)$ and parametrized by θ. Specifically, θ is a parameter vector representing all necessary probabilities of Y conditioned on events $\{X \in G\}$, where G is either a single rule (including the else-rule) or the union of multiple rules. θ is estimated from data by estimating each $P(Y|X \in G)$.

The resulting estimated vector is denoted as $\hat{\theta}$ and contains $\hat{P}(Y|X \in G)$ for all $G \in \mathscr{G}$, where $\mathscr{G}$ consists of all individual rules and the unions of overlapping rules in M. To simplify the notation, we denote $(x, y) \vdash G$, for the following two cases: 1) when G is a single rule (including the else rule), then $(x, y) \vdash G \iff x \in G$; and 2) when G is a union of multiple rules, $G = \bigcup S_i$, then $(x, y) \vdash G \iff x \in \bigcap S_i$. By assuming the dataset $D = (x^n, y^n)$ to be i.i.d., we have

$$P_{M,\theta}(y^n|x^n) = \prod_{G \in \mathscr{G}} \prod_{(x,y) \vdash G} P(Y = y|X \in G). \tag{3.6}$$

3.3.3 Predicting for a new instance

When an unseen instance x' comes in, we predict $P(Y|X = x')$ depending on whether x' is covered by one rule, multiple rules, or no rule. An important question is whether we always need access to the training data, i.e., whether the probability estimates we obtain from the training data points are sufficient for predicting $P(Y|X = x')$, especially when x' is covered by multiple rules by which no instance in the training data is covered simultaneously.

For instance, if x' is covered by two rules S_i and S_j, $P(Y|X = x')$ is then predicted as $\hat{P}(Y|X \in S_i \cup S_j)$. However, if there are no training data points covered both by S_i and S_j, then we would not obtain $\hat{P}(Y|X \in S_i \cup S_j)$ in the training phase. Nevertheless, in this case we have $|S_i \cup S_j| = |S_i| + |S_j|$, and hence

$$\hat{P}(Y|X \in S_i \cup S_j) = \frac{|S_i|\hat{P}(Y|X \in S_i) + |S_j|\hat{P}(Y|X \in S_j)}{|S_i| + |S_j|}. \tag{3.7}$$

By contrast, when x' is covered by one rule only or no rule, the corresponding class probability estimation is already obtained during the training phase. Thus, we conclude that access to the training data is not necessary for prediction.

3.4 Rule Set Learning as Probabilistic Model Selection

Exploiting the formulation of rule sets as probabilistic models, we define the task of learning a rule set as a probabilistic model selection problem. Specifically, we use the minimum description length (MDL) principle for model selection.

The MDL principle is one of the best off-the-shelf model selection methods and has been widely used in machine learning and data mining (Galbrun 2022; Grünwald and Roos 2019). Although rooted in information theory, it has been recently shown that MDL-based model selection can be regarded as an extension of Bayesian model selection (Grünwald and Roos 2019).

The principle of MDL-based model selection is to pick the model, such that the code length (in bits) needed to encode the data given the model, together with the model itself, is minimized. We begin with discussing Normalized Maximum Likelihood (NML) distributions for calculating the bits for encoding the data given the model, followed by the calculation of the code length for encoding the model itself.

3.4.1 Normalized Maximum Likelihood Distributions for Rule Sets

As the Kraft inequality connects code length and probability[2], the core idea of the (modern) MDL principle is to assign a single probability distribution to the data given a rule set M (Grünwald and Roos 2019), the so-called *universal distribution* denoted by $P_M(Y^n|X^n = x^n)$. Informally, $P_M(Y^n|X^n = x^n)$ should be a representative of the rule set model—as a family of probability distributions—$\{P_{M,\theta}(y^n|x^n)\}_\theta$. The theoretically optimal "representative" is defined to be the one that has minimax regret, i.e.,

[2]Note that Section 3.4.1 — 3.4.2 describe the definitions of NML distributions and our proposed approximation for it, which were already introduced in Chapter 2 (Section 2.4.1 — 2.4.2). We deliberately keep the repeated content so that Chapter 3 is self-contained in describing the refined method for learning TURS models.

$$\arg\min_{P_M} \max_{z^n \in \mathscr{Y}^n} \Bigg[-\log_2 P_M(Y^n = z^n | X^n = x^n) \\ - \Big(-\log_2 P_{\hat{\theta}(x^n, z^n)}(Y^n = z^n | X^n = x^n) \Big) \Bigg]. \tag{3.8}$$

We write the parameter estimator as $\hat{\theta}(x^n, z^n)$ to emphasize that it depends on the values of (X^n, Y^n). The unique solution to P_M of Equation (3.8) is the so-called normalized maximum likelihood (NML) distribution (Grünwald 2007),:

$$P_M^{NML}(Y^n = y^n | X^n = x^n) = \frac{P_{M,\hat{\theta}(x^n, y^n)}(Y^n = y^n | X^n = x^n)}{\sum_{z^n \in \mathscr{Y}^n} P_{M,\hat{\theta}(x^n, z^n)}(Y^n = z^n | X^n = x^n)}. \tag{3.9}$$

That is, we "normalize" the distribution $P_{M,\hat{\theta}}(.)$ to make it a proper probability distribution, which requires the sum of all possible values of Y^n to be 1. Hence, we have $\sum_{z^n \in \mathscr{Y}^n} P_M^{NML}(Y^n = z^n | X^n = x^n) = 1$ (Grünwald and Roos 2019).

3.4.2 Approximating the NML Distribution

A crucial difficulty in using the NML distribution in practice is the computation of the normalizing term $\sum_{z^n} P_{\hat{\theta}(x^n, z^n)}(Y^n = z^n | X^n = x^n)$. Efficient algorithms almost only exist for exponential family models (Grünwald and Roos 2019), hence we approximate the term by the product of the normalizing terms for the individual rules.

NML distribution for a single rule. For an individual rule $S \in M$, we write all instances covered by S as (x^S, y^S), in which y^S can be regarded as a realization of the random vector of length $|S|$, denoted as Y^S. Y^S takes values in $\mathscr{Y}^{|S|}$, the $|S|$-ary Cartesian power of $\mathscr{Y}$. Consequently, following the definition of the NML distribution in Equation (3.9), the NML distribution for $P_S(Y)$ equals

$$P_S^{NML}(Y^S = y^S | X^S = x^S) = \frac{\hat{P}_S(Y^S = y^S | X^S = x^S)}{\sum_{z^S \in \mathscr{Y}^{|S|}} \hat{P}_S(Y^S = z^S | X^S = x^S)}. \tag{3.10}$$

Note that $\hat{P}_S$ depends on the values of z^S. As $\hat{P}_S(Y)$ is a categorical distribution,

it has been shown (Mononen and Myllymäki 2008) that the normalizing term can be written as $\mathcal{R}(|S|, |\mathscr{Y}|)$, a function of $|S|$—the rule's coverage—and $|\mathscr{Y}|$—the number of unique values that Y can take:

$$\mathcal{R}(|S|, |\mathscr{Y}|) = \sum_{z^S \in \mathscr{Y}^{|S|}} \hat{P}_S(Y^S = z^S | X^S = x^S), \tag{3.11}$$

and it can be efficiently calculated in sub-linear time (Mononen and Myllymäki 2008).

The approximate NML distribution. We propose to approximate the normalizing term of the NML distribution for rule set model P_M^{NML} as the product of the normalizing terms of P_S^{NML} for all $S \in M$:

$$P_M^{apprNML}(Y^n = y^n | X^n = x^n) = \frac{P_{M,\hat{\theta}(x^n,y^n)}(Y^n = y^n | X^n = x^n)}{\prod_{S \in M} \mathcal{R}(|S|, |\mathscr{Y}|)}. \tag{3.12}$$

Note that the sum over all $S \in M$ *does* include the "else rule" S_0. The rationale of using the approximate-NML distribution is as follows. First, it is equal to the NML distribution for a rule set without any overlap, as follows.

Proposition 1. *Given a rule set M in which for any $S_i, S_j \in M$, $S_i \cap S_j = \emptyset$, then $P_M^{NML}(Y^n = y^n | X^n = x^n) = P_M^{apprNML}(Y^n = y^n | X^n = x^n)$.*

Second, when overlaps exist in M, approximate-NML puts a small extra penalty on overlaps, which is desirable to trade-off overlap with goodness-of-fit: when we sum over all instances in each rule $S \in M$, the instances in overlaps are "repeatedly counted". Third, approximate-NML behaves like the Bayesian information criterion (BIC) asymptotically, which follows from the next proposition.

Proposition 2. *Assume M contains K rules in total, including the else rule. Under the mild assumption that $|S|$ grows linearly as the sample size n for all $S \in M$, then $\log\left(\prod_{S \in M} \mathcal{R}(|S|, |\mathscr{Y}|)\right) = \frac{K(|\mathscr{Y}|-1)}{2} \log n + \mathcal{O}(1)$, where $\mathcal{O}(1)$ is bounded by a constant w.r.t. to n.*

The proofs of these two propositions are shown in the Appendices of Chapter 2.

3.4.3 Code length of model

To obtain the final MDL based score, we next describe how to calculate the code length of the model, denoted as $L(M)$. The code length needed to encode the model depends on the encoding scheme we choose. Given the Kraft's inequality (Grünwald 2007), this can be practically treated as putting prior distributions on the model class. We describe the encoding scheme in a hierarchical manner due to the complexity of the model class.

Integer code for the number of rules. First, we encode the number of rules in the rule set, for which we use the standard Rissanen's integer universal code (Rissanen 1983). The code length needed for encoding an integer K is equal to

$$L_{rissanen}(K) = c + \log_2(K) + \log_2(\log_2(K)) + \log_2(\log_2(\log_2(K))) + \ldots;$$

the summation continues until a certain precision is reached (which we set as 10^{-5} in our implementation), and $c \approx 2.865$ is a constant.

Encoding individual rules. Next, we encode the each individual rule separately. For a given rule with k literals, we first encode k, the number of literals, by a *uniform code*: as k's range is bounded by the number of columns of the dataset, denoted by K_{col}, the code length needed to encode k is equal to

$$L_{num\_literal} = \log_2 K_{col}. \tag{3.13}$$

As each literal contains one unique variable, given the number of literals k, we further specify which are these k variables among all K_{col} variables, again with a uniform code. Thus, the code length needed to specify which these k variables are is equal to

$$L_{which\_vars} = \log_2 \binom{K_{col}}{k}. \tag{3.14}$$

Further, we sequentially encode the *operator* (i.e., '$\geq$' and/or '$<$') and the *value* of each literal. Specifically, for numeric variables, the literal is in either of the two forms: 1) $X \geq$ (or $<$) v, and 2) $v_1 \leq X < v_2$. As a result, we first need to encode the which form the literal is, which cost $L_{form} = 1$ bit. Next, to encode the values v [or $(v_1, v2)$], we need to know in advance the search space of v [or $(v_1, v2)$], which are chosen as quantiles in our algorithm implementation.

The number of candidate values (quantiles) for each numeric feature variable is a hyper-parameter, which we argue should be chosen based on the task at hand: it should be large enough without loss too much information for the prediction, while at the same time the computational budget and the prior knowledge on what is useful for interpreting the rules should also be taken into account in practice.

Denote the number candidate cut points *after excluding those that result in a rule with coverage equal to 0* as K_{value}. Depending on whether the literal contains one or two splits, we can further calculate the code length needed to encode the operator and value(s) in the literal, denoted as L_{value_op}, as

$$L_{value\_op} = L_{operator} + L_{form} + \log_2 K_{value}, \text{or} \tag{3.15}$$

$$L_{value\_op} = \log_2 \binom{K_{value}}{2} + L_{form}, \tag{3.16}$$

since for the former case we also need to encode the operator in the literal, i.e., "$\geq$" or "$<$", which cost $L_{operator} = 1$ bit. In contrast, the latter case has only one possibility for the operators, and hence requires 0 bit to encode it.

Next, for categorical variables with $\mathcal{L}$ levels, encoding a subset of l levels requires $L_{value\_op} = \log_2 \mathcal{L} + \log_2 \binom{\mathcal{L}}{l}$ bits; the former term, $\log_2 \mathcal{L}$, is needed for encoding the number l itself, and the latter one is code length needed to specify these l levels from $\mathcal{L}$ in total. For simplicity, in our implementation we assume all categorical features are one-hot encoded, and hence $L_{value\_op} = 1$.

To sum up, the number of bits needed for encoding an individual rule S, denoted as $L(S)$, is equal to

$$L(S) = L_{num\_literal} + L_{which\_vars} + \sum L_{value\_op}, \tag{3.17}$$

in which the term $\sum L_{value\_op}$ denotes the summation of the code length needed to encode the operator and value for each single literal.

Note that $2^{-L(S)}$ can be interpreted as a prior probability mass for S among all possible individual rules (Grünwald 2007). Moreover, because of the way we determine K_{value} (i.e., by excluding those candidate cut points that lead to rules with coverage equal to 0), the code length needed to encode a single rule does depend on the order of encoding each literal in the condition of the rule. This turns out to be desirable because of our algorithmic approach, which will be

described in Section 3.5.

Encoding the rule set. Based on the code length needed for single rules, we can now define the code length needed to encode the whole rule set. Given a rule set M with K rules, the total bits needed to encode M is

$$L(M) = L_{rissanen}(K) + \sum_{i=1}^{K} L(S) - \log_2(K!), \tag{3.18}$$

in which the last term is to eliminate the redundancy caused by the fact that the order of the rules in a rule set does not matter.

To see the rationale of introducing the term $(-\log_2(K!))$, consider the prior probability of each rule denoted as $P(S_i) = 2^{-L(S_i)}$. Then, the prior probability of the set of rules $\{S_1, ..., S_K\}$, conditioned on the fixed K, can be defined as

$$P(\{S_1, ..., S_K\}) = \sum \prod_{i=1}^{K} P(S_i) = (K!) \prod_{i=1}^{K} P(S_i), \tag{3.19}$$

in which the sum goes over all permutations of $\{S_1, ..., S_K\}$. Thus, we have $L(M) = L_{rissanen}(K) - \log_2 P(\{S_1, ..., S_K\})$, which connects the definition of $L(M)$ to the prior probability of M and hence justifies the introduction of the term $(-\log_2(K!))$ in Equation (3.18).

3.4.4 MDL-based model selection

After the describing the approximate normalized maximum likelihood distributions and the code length (number of bits) needed to specifying a model in the model class, we can now formulate the task of learning truly unordered rule sets as a model selection problem. That is, our goal is to search for the rule set, denoted as M^*, among all possible rule sets $\mathcal{M}$, such that

$$M^* = \arg\min_M L\left((x^n, y^n), M\right) := \arg\min_M \left[-\log_2 P_M^{apprNML}(Y^n = y^n | X^n = x^n) + L(M)\right], \tag{3.20}$$

in which $P_M^{apprNML}(Y^n = y^n | X^n = x^n)$ is defined in Equation (3.12) and $L(M)$ in Equation (3.18).

We refer to the proposed optimization function $L((x^n, y^n), M)$ as our model selection criterion; for a fixed model M and a (training) dataset (x^n, y^n), we refer to the value of $L((x^n, y^n), M)$ as the *MDL-based score for the rule set model M*.

3.5 Learning Truly Unordered Rules from Data

Given the combinatorial nature of the search space, learning rule sets from data is an extremely difficult task. Notably, although recently proposed algorithms can obtain provably optimal rule lists (Angelino et al. 2017) and decision trees (Hu et al. 2019), their branch-and-bound approaches are not applicable to learning TURS models due to the following reasons. First, our model class (and hence also search space) is different than that of rule lists and decision trees, since our TURS model allows for overlaps of rules. Second, the output of the TURS model is probabilistic while the optimal trees/lists algorithms learn rule-based models with non-probabilistic (or just binary) output. Third, our model selection criterion, although requiring no hyper-parameter for regularization, does not allow efficient search for the global optimum, as like most existing MDL-based approaches (Galbrun 2022). Hence, we cannot easily apply the branch-and-bound approaches as employed by the optimal tree/list algorithms.

As for rule set methods, traditional algorithms focus on defining heuristics that try to characterize the "quality" of individual rules in different ways, often without a global optimization score (Fürnkranz and Flach 2005; Fürnkranz et al. 2012). In addition, recently proposed ones mostly rely on randomized techniques: DRS (Zhang and Gionis 2020) is based on heuristic-based randomized algorithm, IDS (Lakkaraju et al. 2016) on stochastic local search, BRS (Wang et al. 2017) on simulated annealing, and CG (Dash et al. 2018) on (randomized) integer programming. However, BRS and CG are only suitable for binary target and non-probabilistic rules, while DRS and IDS turn out to have unsatisfactory predictive performance as shown in Section 3.6.

Therefore, we develop a heuristic-based algorithm for iteratively learning single rules with extensive innovations in comparison to traditional heuristic algorithms.

3.5.1 Learning a rule set

In the following, we start by describing the process of iteratively learning a rule set, followed by discussing the heuristic of defining the "best" single rule given the current status of the rule set. Then, we discuss how to learn a single rule in Section 3.5.2, in which we introduce a *diverse-patience dual-beam search*

algorithm, together with a novel look-ahead strategy that we propose based on the analogy between the MDL principle and hypothesis testing (Grünwald 2007, Chapter 14.3), which we hence name "MDL-based local testing".

Iteratively learning a rule set

Algorithm 3: Iteratively Learning a Rule Set

Data: dataset $D = (x^n, y^n)$
Result: rule set M

```
1 Initialize M                                          // Empty rule set.
2 while TRUE do
3     S ← Learn-Single-Rule(M, D)        // Described in Algorithm 4
4     if L(D, M ∪ {S}) < L(D, M) then
5         M ← M ∪ {S}     // The ``else-rule" updates accordingly
6     else
7         return rule set M
```

The process of learning a rule set iteratively, rule by rule, is shown in Algorithm 3. The algorithm starts with an empty rule set (in which all instances are covered by the "else-rule") [Line 1]. Then, the "best" single rule, defined as the one that maximizes what we call the *learning-speed-score* heuristic that is discussed in detail next, is learned from data [Line 3]. This single rule is added to the rule set if adding it to the rule set decreases the MDL-based model selection criterion defined Equation (3.20) [Lines 4-5]. This process is repeated until no new rule can be found that further optimizes our model selection criterion [Lines 2-9].

Heuristic score for a single rule

Consider the search space of all possible rule sets, adding one single rule to the rule set can be considered as one single "step" towards another "point" in the search space. As it is obviously meaningless to add a new rule that does not cover any previously uncovered instance, such a step always leads to a monotonic increase for the *coverage* of the rule set (excluding the else rule).

Therefore, we propose a heuristic that leads to the next rule (step) with the

steepest descent with respect to the increase in the coverage of rule set; that is, the next "best" single rule (step) is defined as the one that maximizes *the decrease of the MDL-based score per extra covered instance*. We hence name this heuristic as the *learning speed score*. Formally, given a rule set denoted as M, the learning speed score for a single rule S to be added to M is defined as

$$r(S) = \frac{L\left((x^n, y^n), M\right) - L\left((x^n, y^n), M \cup \{S\}\right)}{|M \cup \{S\}| - |M|}, \tag{3.21}$$

in which $M \cup \{S\}$ denotes the rule set obtained by adding the single rule S to M. Further, $|M|$ and $|M \cup \{S\}|$ respectively denotes the coverage before and after adding S to the rule set M (excluding the else-rule).

We next discuss how to search for the next best rule that optimizes $r(S)$.

3.5.2 Learning a single rule

For describing our algorithm for learning a single rule, we start with describing the general paradigm of applying beam search in learning a single rule, and then move forward to describe our three algorithmic innovations. Last, we put everything together and describe our proposed algorithm in detail.

Preliminary: Beam Search for Learning a single rule

Recall that the condition of a rule S can be written as the *conjunction* of literals, in which each literal takes the form of $\{X_i \in R_i\}$, with R_i representing an interval if X_i is a quantitative variable and a set of categorical levels if X_i is a categorical variable.

When applying a beam search in learning a single rule, we start with an *empty rule containing no literal that hence covers all instances*. Next, we enumerate all feature variables X_i to construct the search space of all possible single literals: for continuous-valued X_i, we pick quantiles as *splits points* and combine it with the operator ('$\geq$' or '$<$') to construct a literal, in which the "search granularity" (i.e., the number of quantiles) is a hyper-parameter that depends on the task at hand, as previously discussed in Section 3.4.3; for categorical variables, we assume they are all one-hot encoded for simplicity, and hence the possible literals are just $(X_i = 1)$ or $(X_i = 0)$. After enumerating all possible single literals, given a beam

width W, we rank these literals with a predetermined criterion, and then pick the top-W literals to be the W candidate rules of length one.

Next, for each of these W candidate rule of length one, we repeat the process of enumerating all possible single literals to append to this rule. We refer to these possible rules obtained by adding one more literal to a given rule as the *rule growth results*. Among all *rule growth results* of these W length-one candidate rules, we again pick the top-W length-two candidate rules, according to the predetermined criterion.

We can repeat this process until some stopping criterion is met, e.g., no rule growth result that can further optimize the model selection criterion can be found (or this has happened consecutively for a number of times). Lastly, among all these candidate rules with different lengths, we return the rule based on the heuristic that defines the "best" next rule (i.e., the learning speed score $r(.)$ in our case).

Note that we build our *diverse-patience dual-beam search* algorithm upon this general paradigm of applying beam search to learning a single rule with significant algorithmic innovations, as follows: 1) instead of using one single heuristic for searching for the next "best" rule, we introduce a look ahead strategy in the rule growth process; 2) instead of simply keeping the top-W rule growth results in the beam, we also monitor the diversity of "patience"; and 3) instead of a single beam, we introduce another auxiliary beam with a complementary score and we simultaneously keep two beams. The complementary score is proposed as we observe that allowing overlaps in rule sets leads to the algorithmic challenge that existing rules in the rule set may become obstacles to searching for new rules to be added to the rule set. We next describe these three heuristics in depth.

MDL-based local testing

When growing a rule S by adding a literal and obtaining its growth result S', we essentially leave out the instances covered by S *but not* S' to be covered potentially by rules we may obtain later. Existing rule learning heuristics often neglect this left-out part but focus only on characterizing the quality of the rule growth result S' itself. In contrast, we introduce a local test that can be viewed as a way of assessing whether it is better to model the instances in $\{S \setminus S'\}$ by the rule S (and hence discard S' and stop growing S), or to leave out the instances

in $\{S \setminus S'\}$ for "future" rules that we may obtain later.

Formally, consider a rule S, its growth result S', and the potentially left-out part, defined and denoted as $S_l = S \setminus S'$. We only proceed to consider S' as an appropriate rule growth candidate if

$$-\log_2 P_S^{NML}(y^S|x^S) > -\log_2 P_{S'}^{NML}(y^{S'}|x^{S'}) - \log_2 P_{S_l}^{NML}(y^{S_l}|x^{S_l}) + L_{split}, \tag{3.22}$$

in which $P_S^{NML}(y^S|x^S)$ is the NML-distribution when viewing a single rule as a local probabilistic model, defined in Equation (3.10). Further, L_{split} denotes the code length needed to encode the condition that splits S into S' and S_l. This requires specifying 1) the variable of the literal and 2) the numeric threshold or the categorical levels (which depends on the variable type), both with the uniform code as described in Section 3.4.3. That is, we only allow rule growth that satisfies the local test defined in Equation (3.22).

Intuitively, this is equivalent to building a depth-one decision tree for instances covered by S only, in which the left and right nodes are S' and S_l respectively. We then compare whether S on itself or S' *together with* S_l is a better local model, according to the MDL principle (Grünwald 2007). Recall that MDL-based model selection picks the model that minimizes the code length needed to encode the data together with the model; thus, if the local test is satisfied, we prefer the depth-one decision tree with nodes S' and S_l over the single-node tree with the only node S, and vice versa.

The rationale of the local test is that, by explicitly considering the local model for the left out part S_l, we incorporate the potential carried by the instances in S_l. That is, the local test we introduce can exclude those rule growth results of S that may leave out a subset of instances that are hard to model later. We empirically show in Section 3.5.2 that without MDL-based local testing, the learning speed score can be too greedy and hence the algorithm fails to reveal the ground-truth rule set model even in a simple simulated case.

Beam search with "patience" diversity

We now present the beam search with the patience diversity. For the simplicity of presentation, we now focus on describing the beam search with the "main" beam that adopts the learning speed score $r(S)$ as the heuristic, after which the

description of the complementary-score-assisted auxiliary beam immediately follows in Section 3.5.2.

Assuming the beam width is W, we start with a rule with empty condition which all instances satisfy. Next, we go over all possible rule growth results by adding one single literal. Furthermore, we keep the top-W rule growth results, with the following properties: 1) they satisfy the MDL-based local testing, defined previously in Section 3.5.2; 2) they have the highest learning speed score defined by $r(S)$ in Equation (3.21); and 3) they satisfy the patience diversity constraint, which we discuss below.

Motivation for "patience" diversity. While we aim to iteratively search for the rule with the best learning speed score $r(S)$ (Equation 3.21), it may be too greedy to directly use $r(S)$ to search for the next best literal (as a rule can contain multiple literals). Denote a rule as S and its growth result as S', we empirically observe that the coverage of S' can shrink drastically in comparison to that of S when directly using $r(S)$ for learning the next literal. However, a more "patient" search procedure with a moderate change in the coverage may be desirable in some cases, as a moderate decrease in coverage leaves many possibilities for adding more literals later. This concept of "patience" was first introduced in PRIM (Friedman and Fisher 1999), and we are the first to combine it with a beam search approach.

Specifically, we propose to use the beam search approach to keep the diversity of the patience, i.e., to have a variety of rule growth results, with diverse coverage *relative to the rule from which the rule growth result is obtained*.

Beam search with patience diversity. Given a potentially incomplete rule S, we search all candidate rules $\{S'\}$ that can be obtained by adding a literal to S (excluding those not satisfying the MDL-based local test).

Given a beam width W, we categorize all candidate rules, denoted as $\{S'\}$, into W clusters according to their coverage: the wth cluster is defined as:

$$\{S'\}_w = \{S' \in \{S'\} : \frac{|S'|}{|S|} \in \left[\frac{w-1}{W}, \frac{w}{W}\right)\}, \quad w \in \{1, ..., W\}; \tag{3.23}$$

i.e., all candidate rule growth results in $\{S'\}_w$ must satisfy the condition that its coverage divided by the coverage of S is in the interval $[(w-1)/W, w/W)$.

For each cluster, we search for the best growth result by optimizing the learning speed score $r(S')$. In this way, our beam search is diverse with regard to

the degree of "patience": when the coverage decreases by a small ratio only, the optimization is "patient" (by leaving a lot of possibilities for adding more literals); on the other hand, when the coverage decreases by a large ratio, the optimization is greedy (by leaving out little room for further refinement). We empirically show that adopting patience diversity improves the prediction performance of our method in Section 3.6.6.

Auxiliary beam with a complementary score

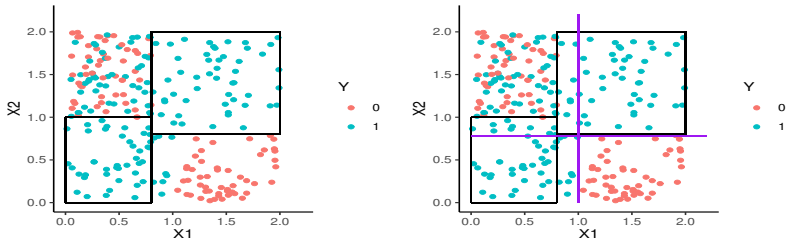

Figure 3.1: (Left) Simulated data with a rule set containing two rules (black outlines). (Right) Growing a rule to describe the bottom-right instances will create conflicts with existing rules. E.g., adding either $X_1 > 1$ (vertical purple line) or $X_2 < 0.8$ (horizontal purple line) would create a huge overlap that deteriorates the likelihood.

We now describe the auxiliary beam in our dual-beam approach. We start with the motivation for having an auxiliary beam, and next describe in detail the complementary score, as well as how we incorporate the auxiliary beam in the beam search algorithm.

Motivation for auxiliary beam. Recall that the learning speed score $r(S)$ evaluates the decrease of the MDL-based optimization score per extra covered instance when S is added to the rule set; thus, to maximize $r(S)$ we aim for obtaining a rule S that 1) improves the likelihood of the instances not covered by the rule set so far, and 2) has similar class probability estimates to those rules in the rule set that overlap with S. However, when iteratively searching for the next literal, the single literals we consider may not be able to contribute to both aims simultaneously.

Consider an illustrative example with data and a rule set with two rules (in black) in Figure 3.1 (left). If we want to grow a rule that covers the *bottom-right* instances, the existing rules form a blockade: the right plot shows how adding

either $X_1 > 1$ or $X_2 < 0.8$ to the empty rule (shown in purple) would create a large overlap with the existing rules, with significantly different probability estimates.

Our auxiliary beam is useful in cases like this, to keep literals like $X_1 > 1$ or $X_2 < 0.8$, which solely contributes to the first goal we discussed above, i.e., it improves the likelihood of the instances not covered by the rule set so far but creates a "bad" overlap with a large class probability difference. As a rule's class probability estimation is still up to change during the growing process, we can potentially "correct" bad overlaps by adding more literals later.

Thus, we propose an auxiliary beam together with a complementary score, informally defined as the *learning speed score calculated by ignoring the overlap created by the new rule that is being grown.* We next formally define the complementary score.

Complementary score. Formally, given a rule set M and a new rule S (i.e., $S \notin M$), the complementary learning speed after adding S to M, denoted as $R(S)$, is defined as

$$R(S) = \frac{L\left((x^n, y^n), M\right) - L\left((x^n, y^n), M \cup \{S \setminus M\}\right)}{|M \cup \{S \setminus M\}| - |M|} \tag{3.24}$$

in which $S \setminus M$ can be regarded as a "hypothetical" rule with the cover equal to the instances covered by rule S excluding the instances covered by rules already in M, and hence $L\left((x^n, y^n), M \cup \{S \setminus M\}\right)$ denotes the MDL-based score (as defined in Equation 3.20) after adding $S \setminus M$ to the rule set M.

Complementary-score-assisted beam search. We simultaneously keep two beams, both with a beam width W. Apart from the beam that keeps the top-W literals according to the learning speed score $r(.)$, we additionally keep an auxiliary beam that keeps the top-W literals according to the complementary score $R(.)$.

Further, the auxiliary beam must also satisfy the MDL-based local testing defined in Section 3.5.2, with the NML distribution calculated based on instances *excluding* those covered by the rule set. That is, consider a rule set M, a rule S with its growth result S', and the left-out part $S \setminus S' := S_l$, the local test for the

auxiliary beam is defined as

$$\begin{aligned} &- \log_2 P_S^{NML}(y^{S\setminus M}|x^{S\setminus M}) > \\ &\quad - \log_2 P_{S'\setminus M}^{NML}(y^{S'\setminus M}|x^{S'\setminus M}) - \log_2 P_{S_l\setminus M}^{NML}(y^{S_l\setminus M}|x^{S_l\setminus M}) + L_{split}. \end{aligned} \tag{3.25}$$

Additionally, the auxiliary beam must satisfy the patience diversity, as described in Section 3.5.2. The only difference is that the coverage of each rule is calculated based on $S \setminus M$ instead of S; i.e., instances that are already covered by the rule set M are ignored.

Algorithm description

We now put all heuristics together and describe in full our algorithm for finding the next rule, of which the pseudo code is provided in Algorithm 4.

With a rule set M that either contains no rule or some existing rules, we always start with an *Empty Rule* that contains no literals for its condition, and we initialize the "rules-for-next-iter" **[Line 2]** as an array containing the empty rule only.

For each iteration, we initialize a new beam an a new auxiliary beam **[Line 4-5]** with beam width W. The beam keeps the top-W rule growth results using the learning speed score defined in Equation (3.21); in contrast, the auxiliary beam keeps the W best rule growth results using the complementary score by ignoring the rule's overlap with M, as discussed in detail in Section 3.5.2.

Next, we use every rule in the "rules-for-next-iter" array as a "base" for growing **[Line 6-18]**. Specifically, given a rule, we first generate its candidate growth *by adding one literal only* **[Line 7]**. That is, we go over all feature variables in the dataset, and for each variable, we generate candidate literals with numeric thresholds (quantiles) or with categorical levels, based on the variable type.

Further, we cluster the generated candidates by their coverage (for the beam), as well as their coverage excluding the instances already covered by M (for the auxiliary beam) **[Line 8 & 12]**. We next filter out the candidates in "categorized-candidates" and "categorized-candidates-auxiliary" with the MDL-based local test defined in Section 3.5.2 **[Line 9 & 13]**. Further, we search for the best candidate in each cluster of the beam using $r(.)$, and each cluster of the auxiliary beam using $R(.)$ **[Line 10-11 & 14-15]**.

To check whether the growing process should be stopped after this iteration, we take a budget denoted as K_{stop}: we stop the beam search when this is the K_{stop}-th time in a row that both beams (the beam and the auxiliary beam) produce rules with worse scores ($r(.)$ for the "beam" and $R(.)$ for the "auxiliary-beam") than the previous beams [**Line 19**].

If the stopping criterion is not met, we first filter the beam and auxiliary beam to reduce the number of rules in each beam to be equal to the beam width W, as both of them now contain ($W * \text{length(rules-for-next-iter)}$) rules [**Line 22-23**]. Specifically, we sort all rules in the beam based on their coverage and categorize them into W clusters; next, for each cluster, we keep the top-W rules with the highest $r(.)$ for the "beam" and highest $R(.)$ for the "auxiliary-beam", as the base for rule growing for the next iteration. Last, we update "all-candidate-rules" and "rules-for-next-iter" [**Line 24-25**], and continue to the next iteration [**Line 3**]. The former is the pool we use for finally selecting the next best rule to be potentially added to M, and the latter contains all "base rules" for the next rule growth iteration, which contains all rules in the beam and the auxiliary beam.

Finally, if the stopping criterion is met, we return the rule S among "all-candidate-rules" with the best (largest) learning speed score $r(.)$ [**Line 20**].

3.6 Experiments

We extensively benchmark our diverse-patience dual-beam algorithm and we study the truly unordered rule sets (TURS) model learned from data in the following aspects:

1. Does the TURS model learned from data achieve on-par or better classification performance in comparison to other rule-based methods, especially rule set methods that allow (implicit) orders among rules?

2. Can rules in the TURS model learned from data be empirically treated as *truly unordered*?

3. Do the class probability estimates from rules in the induced TURS model generalize well to unseen (test) instances, such that these probability estimates are reliable to serve as part of the explanations for the (probabilistic) predictions?

4. Is the model complexity of the TURS model learned from data smaller than that of the rule-based models learned by competitor methods?
5. What are the effects of our proposed heuristics, including the beam search with patience diversity and the MDL-based local test?

3.6.1 Setup

Datasets. We conduct an extensive experiments with 31 datasets, summarized in Table 3.2. Our multi-class datasets are from the UCI repository, while the binary-class datasets are from both the UCI repository (Dua and Graff 2017) and the ADBench Github repository (Han et al. 2022). ADBench is a benchmark toolbox for anomaly detection (including imbalanced classification), and all datasets from it are marked in *italics* in Table 3.2.

Competitors. We compare against a wide ranges of methods, summarized as follows. First, we compare with *unordered* CN2 (Clark and Boswell 1991), which adopts the one-versus-rest strategy. As CN2 does not impose an implicit order among rules, it is conceptually the closest competitor to our method. Second, we compare with DRS (Zhang and Gionis 2020) and IDS (Lakkaraju et al. 2016), as they are the only two multi-class rule set methods without first learning rules for individual class labels and then leveraging the one-versus-rest strategy, to the best of our knowledge. Further, similar to us, they also incorporate the properties of overlaps in their optimization scores: DRS aims to minimize the size of overlaps, while IDS optimizes a linear combination of seven scores, one of which explicitly penalizes the size of overlaps. Third, we compare with CLASSY, a recently proposed ordered rule list method, as it uses a similar model selection approach based on the MDL principle. Fourth, since the MDL principle is conceptually related to Bayesian modelling, we also compare with BRS (Wang et al. 2017) as a representative method under the Bayesian framework, which also adopts an non-heuristic simulated annealing approach. Last, we include RIPPER (Cohen 1995), CART (Breiman et al. 1984), and C4.5 decision trees (Quinlan 2014), due to their wide use in practice.

Implementation details. For TURS, we set the beam width as 10, and the number of candidate cut points for numeric features as 20[3]. For competitor al-

[3]We observe that further increasing the number of candidate cut points for numeric features to 100, as well as the beam width to 20, makes no big difference on the predictive performance in general.

gorithms, we use CN2 from Orange (Demšar et al. 2013), IDS from a third-party implementation with proven scalability (Filip and Kliegr 2019), RIPPER and C4.5 from Weka (Hall et al. 2009) and its R wrapper, CART from Python's Scikit-Learn package (Pedregosa et al. 2011), and finally, DRS, BRS, CLASSY from the original authors' implementations. Competitors algorithms' configurations are set to be the same as the default as in the paper and/or in original authors' implementations. We make the code public for reproducibility[4].

All reported results in this section are based on five-fold stratified cross-validation, unless mentioned otherwise.

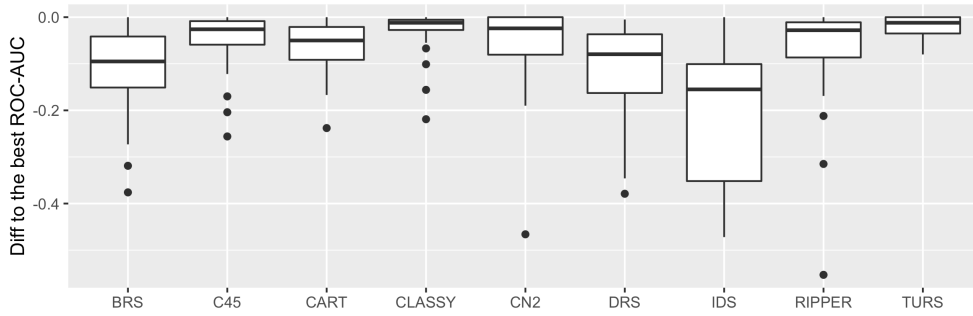

Figure 3.2: For each algorithm, we calculate for every individual dataset the difference between its ROC-AUC score and the best ROC-AUC scores. The differences to the best ROC-AUC scores for each algorithm is illustrated by a box-plot.

3.6.2 Classification performance

To investigate the classification performance for the TURS model learned from data, we report in Table 3.3 average ROC-AUC scores on the test sets obtained using five-fold *stratified* cross-validation. For multi-class classification, we report the "macro" one-versus-rest AUC scores, as "macro" AUC treats all class labels equally and hence can characterize how well the classifiers predict for the minority classes.

Note that BRS (Wang et al. 2017) can only be applied to binary datasets. Further, we fail to obtain the results of DRS on three datasets because the implementation of DRS makes it incapable of handling datasets with very large number

[4] `https://github.com/ylincen/TURS2`.

of columns[5]. We also fail to obtain the result of IDS on one dataset as it exceeds the predetermined time limit: 10 hours for one single fold of one dataset.

We show that TURS is very competitive in comparison to its competitors in the following aspects. First, TURS performs the best in 11 out of the total 31 datasets, and performs the best in 6 out of 11 multi-class datasets. We denote the best ROC-AUC for each dataset in bold. Second, we report the difference between TURS's ROC-AUC scores and the best ROC-AUC scores for each individual dataset, in the bracket in the table. This shows the gap between TURS and the best competitor for each individual dataset.

We further calculate the ROC-AUC scores of each competitor algorithm for each dataset, minus the best ROC-AUC score for each individual dataset, which measures the "gaps to the best" for each competitor algorithm. We compare these gaps-to-best scores for all competitor algorithms in Figure 3.2. The boxplots demonstrate that TURS is very stable for all 31 datasets we have tested, and in comparison to its competitors the gaps-to-best scores are much smaller.

Third, among all rule set methods (CN2, DRS, IDS, TURS), TURS shows substantially superior performance against DRS and IDS. As DRS and IDS both aim to reduce the size of overlaps, our results indicate that simply minimizing the sizes of overlaps may impose a too restricted constraint and hence lead to sub-optimal classification performance. On the other hand, CN2 is competitive in terms of obtaining the best AUCs, especially for binary datasets, as shown in Table 3.3. However, as shown in Figure 3.2, CN2 has in general larger gaps to the best AUCs than TURS does. Further, more comparison between TURS and CN2 will be presented in the following paragraphs.

3.6.3 Prediction with 'random picking' for overlaps

Recall that in our definition of the truly unordered rule set (TURS) model, we estimate the class probabilities for overlaps by considering the "union" of the covers of all involved rules. Thus, the next question we study empirically is whether our formalization of rule sets as probabilistic models can indeed lead to overlaps only formed by rules with similar probabilistic estimates.

Therefore, we compare the probabilistic predictions of our TURS models

[5]The key issue is that their implementation involves transforming a binary vector to an integer, and they use the "numpy" package for this, which does not support "arbitrarily large integers".

against the probabilistic predictions by what we call "random picking" for overlaps: when an unseen instance is covered by multiple rules, we randomly pick one of these rules, and use its estimated class probabilities (estimated from the training set) as the probabilistic prediction for this instance.

Intuitively, if the overlaps are formed only by rules with similar probabilistic output, we expect the probabilistic prediction performance by TURS and by "TURS with random-picking" (abbreviated as TURS-RP) to be very close. We report the ROC-AUC of TURS and TURS-RP in Table 3.4, together with the percentage of instances covered by more than one rules (the "%overlaps" column). The ROC-AUC scores are obtained using five-fold cross-validation, and specifically, for each fold, the "random picking" ROC-AUC is obtained by averaging the ROC-AUC scores obtained by 10 random picking probabilistic predictions.

We benchmark the ROC-AUC scores against those of CN2 (IDS and DRS are excluded due to their sub-optimal performance in general). We have shown that the differences between the ROC-AUC of TURS and TURS-RP are all negligible up to the second decimal (i.e., smaller than 0.01), while the differences between the ROC-AUC of CN2 and CN2-RP are mostly larger than 0.01 (shown in bold), among which eight are larger than 0.05.

We can hence conclude that, while CN2 relies heavily on its conflict resolving schemes for overlaps, TURS produces overlaps only formed by probabilistic rules with very similar probability estimates. This indicates our probabilistic rules can be viewed as *truly unordered in the sense that, when an instance is covered by multiple rules, the rule chosen to predict class probabilities has little effect on the prediction performance.*

3.6.4 Generalizability of local probabilistic estimates

While rule-based models are commonly considered to be intrinsically explainable models, we argue that only rules with probability estimates that generalize well can serve as trustworthy explanations. Thus, we next examine the difference between individual rules' probability estimates on the train and test sets.

Specifically, given a rule set induced from a specific dataset, we look at each individual rule's probability estimates, estimated from the training and test set respectively, by the maximum likelihood estimator. Finally, we report the weighted averages of the probability estimates differences for all rules, weighted by the

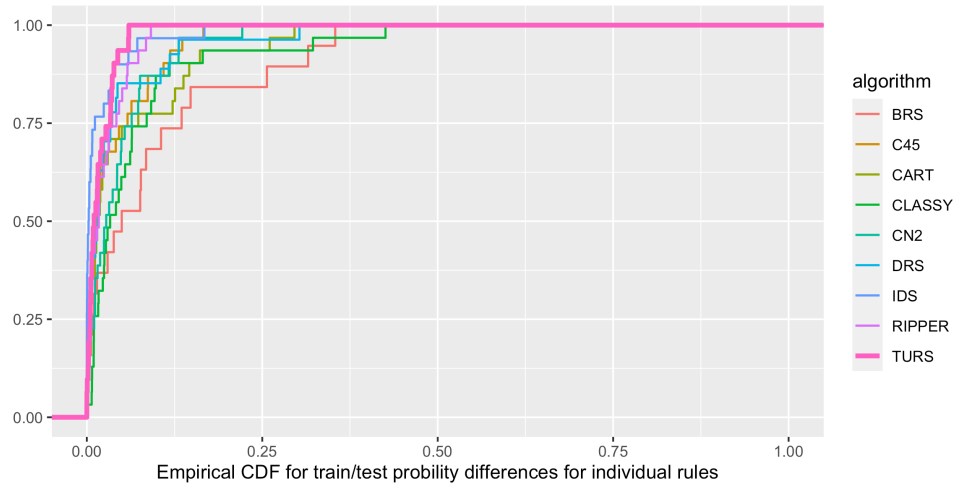

Figure 3.3: The weighted average of the differences between the class probability estimates of every individual rule for training and test sets, shown as the empirical cumulative distribution function, in which the weight is defined as the coverage of each rule for the training set.

coverage of each rule on the training set.

Formally, given a rule set with K rules, $M = \{S_1, ..., S_K\}$, denote the probability estimates of all rules by $(\mathbf{p}_1, ..., \mathbf{p}_K)$ and $(\mathbf{q}_1, ..., \mathbf{q}_K)$, respectively estimated from the training and test set. Assume each probability estimate has length C (i.e., $C = 2$ for binary target and $C > 2$ for multi-class target), we measure how well the individual rules generalize by

$$g = \frac{1}{K} \sum_j |S_j| \left(\sum_c \frac{1}{C} |p_{jc} - q_{jc}| \right) \tag{3.26}$$

in which p_{jc} (q_{jc}) is the c-th element of vector $\mathbf{p_j}$ ($\mathbf{q_j}$). Note that each individual rule is treated separately in calculating the g-score above, and hence the overlaps do not play a role here.

We calculate this score for all algorithms and all datasets, averaged using the five-fold stratified cross-validation, and we present the results with empirical cumulative density functions (ECDF) in Figure 3.3. Since the position of the curve towards the upper-left shows that the corresponding algorithm has small probability estimate differences between training and test sets, we observe that

TURS (the bold curve) dominates rule sets learned by the rest of the algorithms, with IDS the only close competitor.

For some datasets, IDS learns rule sets that have smaller probability estimation differences than the TURS model (shown by the fact that part of the corresponding blue curve is above the curve of TURS in bold). However, this indicates that IDS has serious "under-fitting" if we take into consideration IDS's suboptimal predictive performance as discussed in Section 3.6.2. That is, IDS produces rules with too large coverage, and hence is not specific and refined enough for classification, although rules with large coverage have probability estimates that generalize well.

Thus, in conclusion, rules in the TURS model learned by our algorithm are equipped with more reliable and trustworthy class probability estimates, in comparison to the other eight tree- and rule-based models.

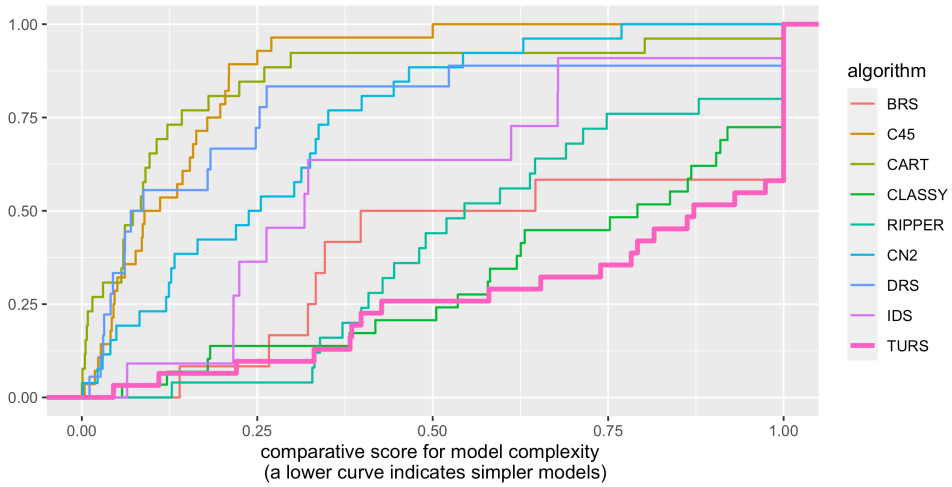

Figure 3.4: Empirical cumulative distribution function for the comparative score for model complexity. Curves towards the bottom-right indicate larger comparative scores and simpler models.

3.6.5 Model complexity

We study next whether TURS empirically leads to more complex rule sets given that it allows overlaps formed by rules with similar probabilistic outputs only.

We measure the model complexity by the number of total literals for each model: i.e., summing up the lengths of rules in a rule set, rule list, or decision tree (by treating each tree path as rule), which directly indicates the workload for a domain expert if they read the rules. We report this measure in Table 3.5, and specifically, we mark the results from the models with substantially worse ROC-AUC scores than those of TURS by denoting them in *smaller* font sizes. Precisely, for a given dataset, all competitor models with more than 0.1 smaller ROC-AUC scores than that of TURS are marked. Excluding the results from these models, we observe that TURS produces the simplest model for 13 out 31 datasets. The model complexity of all simplest models are denoted in bold in Table 3.5.

Further, to illustrate the differences between the number of literals across all algorithms, we calculate a comparative score as follow: for each individual dataset, we divide the minimum total number of literals by the total number of literals of each algorithm. This score show that, for each pair of dataset and algorithm, what is the ratio of the minimum number of literals for this dataset, over the number of literals for the algorithm-dataset pair, i.e., *larger scores indicate simpler models as the minimum number of literals is the numerator*. We plot the ECDF of these comparative scores in Figure 3.4, excluding the comparative scores obtained from models with substantially worse ROC-AUC scores than that of TURS, same as above. We observe that TURS lies at the most bottom-right, dominating the other competitors, since curves towards the bottom-right indicate larger comparative scores and hence simpler models.

3.6.6 Ablation study 1: diverse patience beam search

We study the effect of using the beam search with the "diverse patience", by replacing it with a "normal" (non-diverse) beam search. Suppose the beam width is W, we then pick the top-W rule growth candidates *without categorizing rule growth candidates by their coverage*. That is, we "turn off" the diverse coverage constraints both for updating the beam and the auxiliary beam.

As shown in Figure 3.5, when using the diverse coverage heuristic, the ROC-AUC on the test sets (points and curve in orange) becomes better on 25 out of 31 datasets, demonstrating the benefits for predictive performance.

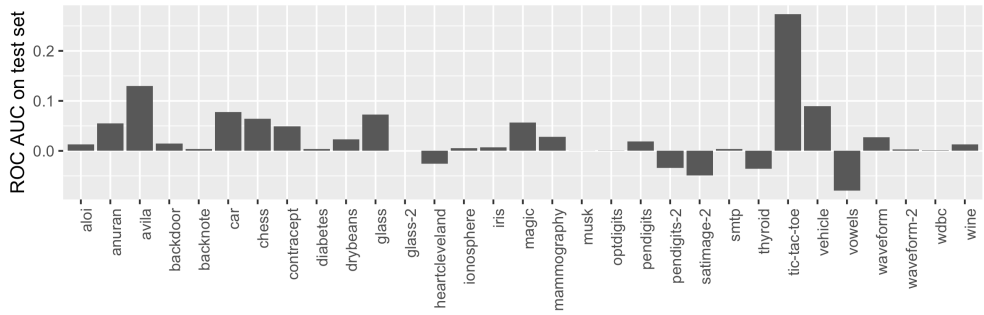

Figure 3.5: The differences between the ROC-AUC scores on the test sets with and without the diverse patience.

3.6.7 Ablation study 2: MDL-based local testing

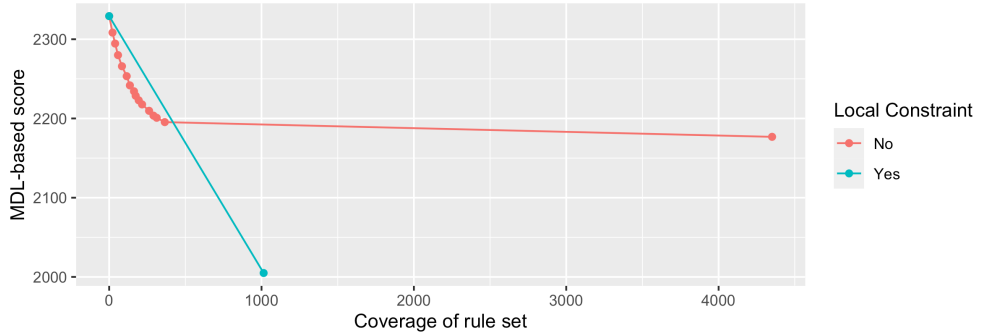

Figure 3.6: The process of adding rules to the rule set, with and without the local testing heuristics, using the first dataset among the 100 simulated datasets. Each point represents the status after a single rule is added, with the x-axis representing the coverage of the (potentially incomplete) rule set after adding this rule, and the y-axis representing the MDL-based score.

Recall that the MDL-based local test is used for evaluating the "potential" in the left out instances when growing a rule. Thus, from the perspective of optimization, it is used for looking ahead to prevent ending up in a local minimum when optimizing our MDL-based model selection criterion as defined in Equation (3.20).

We next illustrate that, without the local test, our algorithm would fail to reveal the ground-truth rule set model even for a very simple simulated dataset. Instead, it would learn a much more complicated model, and consequently, our optimization algorithm would end up at an inferior minimum.

Consider a simulated dataset generated by a known ground-truth rule set model with one rule only as follows. The feature variables are denoted as $X = (X_1, ..., X_{50})$, which are assumed to be all binary. We sample $X_1 \sim Ber(0.2)$, $X_i \sim Ber(0.5)(i = 2, ..., 50)$, in which $Ber(.)$ denotes the Bernoulli distribution. Further, we consider binary target variable Y and sample $Y|X_1 = 1 \sim Ber(0.7)$ and $Y|X_1 = 0 \sim Ber(0.95)$. That is, $X_1 = 1$ (or $X_1 = 0$) is the only "true rule" in this simulated dataset.

We simulate the dataset with sample size 5 000 for 100 times, and run TURS with and without local testing. As shown in Table 3.6, without local testing we achieve a worse (larger) score for our optimization function (i.e., the MDL-based score).

Notably, although the ROC-AUC scores are similar for using and not using the local testing, the ground truth model is only found when local testing is used. When the local testing is disabled, the number of rules and the rule lengths are both not consistent with the "true" model, as irrelevant variables are picked when growing the rules. We have two perspectives to explain the inconsistency.

To begin with, as shown in Table 3.6, when the local testing is *not* used, the difference between the class probabilities estimated from the training and test dataset is larger than the difference when the local testing is imposed, which indicates that the rules as local probabilistic models generalize worse when the local test heuristic is turned off. In other words, we observe overfitting locally.

Further, as we wrote as motivation in Section 3.5.2, the local testing heuristic is designed to prevent leaving out instances that are difficult to cover for 'future' rules, and we do notice this phenomenon empirically. Specifically, for a single run of TURS on the simulated dataset, we plot in Figure 3.6 the procedure of iteratively searching for the next best rule: each point represents the status of the rule set after a single rule is added, with the x-axis representing the coverage of the rule set (i.e., the number of instances covered by at least one of the rules excluding the else-rule), and the y-axis representing the MDL-based score for the rule set as a whole model. Thus, our learning speed score, defined in Section 3.5.1, basically tries to iteratively find the next point (i.e., the next rule) in Figure 3.6 with the steepest slope.

However, without the local test heuristic, although the learning speed scores (shown by the red curve in Figure 3.6) are in the first place larger than that of

the blue curve (for the case when the local testing is used), the red curve achieves an inferior optimization result in the end. That is, without local testing, the instances that are left out are simply ignored during the process of rule growing, which leads to a worse optimization result.

3.6.8 Runtime

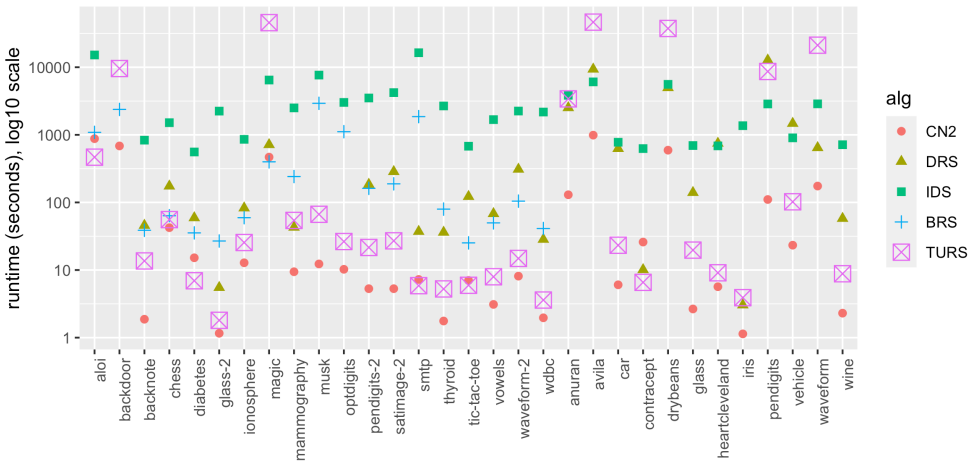

Figure 3.7: Average runtime for five rule set methods. The y-axis is scaled by $log_{10}(\cdot)$.

Last, we report the runtime of TURS, together with rule set competitor methods only, as decision trees/lists methods from mature software (Weka and Python Scikit-Learn) are highly optimized in speed and are known to be very fast.

We illustrate average the runtime (in seconds) obtained using cross-validation in Figure 3.7. In general, the runtime of TURS is competitive among all rule set methods except for CN2. CN2 seems faster in general and scales better to larger datasets, which can be caused both by a more efficient implementation (from the software "Orange3"), and by its algorithmic properties (a greedy and separate-and-conquer approach). However, as we saw in Section 3.6.3, rule sets learned from data by CN2 cannot be empirically treated as truly unordered.

3.7 Conclusion

We studied the problem of learning truly unordered rule sets from data. While existing rule set methods adopt post-hoc schemes to resolve conflicts caused by overlapping rules, we proposed the intuitive idea of only "allowing" rules to overlap if they have similar probabilistic output. Building upon this, we formally defined a truly unordered rule set (TURS) model: given a set of rules and a dataset (assumed i.i.d.), the TURS model defines the likelihood of the class labels given the feature values.

Further, we formalized the problem of learning such TURS model from data as a probabilistic model selection problem, by leveraging the minimum description length (MDL) principle. Our MDL-based model selection criterion can strike a balance between the goodness-of-fit and the model complexity without any regularization parameter.

We further proposed a carefully designed dual-beam diverse-patience algorithm to learn the TURS model from data. We showed that our algorithm can induce rules with competitive performance with respect to the following aspects. First, we benchmarked our algorithm using a large number of datasets and showed that the learned TURS model has very competitive predictive performance measured by the ROC-AUC. Specifically, in comparison to other multi-class rule set methods (CN2, DRS, IDS), the TURS model learned by our algorithm shows clear superiority with respect to the ROC-AUC scores. Second, uniquely, we showed that the TURS model learned by our algorithm is empirically truly ordered, in the sense that the predictive performance is hardly affected when predicting instances covered by multiple rules through a randomly picked rule among these multiple rules. Third, the learned TURS model contains single rules with reliable and trustworthy class probability estimates that can generalize well to the unseen instances. Fourth, the model complexity of the learned TURS model is also competitive in comparison to other rule-based methods.

For future work, we consider using TURS as a building block towards designing interactive rule learning algorithms with humans in the loop, since rules being truly unordered instead of entangled are more comprehensible and easier to edit. That is, comprehending and editing single rules in the TURS model does not require domain experts or data analysts to consider other (potentially many,

higher-ranked) rules. In sensitive area like health care, this may help build trust between the data-driven models and domain experts.

In addition, extending truly unordered rule sets to other machine learning tasks such as feature construction, subgroup discovery, regression with uncertainty, and explaining black-box models are all promising directions.

3.8 Appendix: Comparison to the Previous Work

As this chapter is based on the previous chapter, we hereby summarize the main changes and additions as follows. First, while working on the follow-up real-world case study in health care, we noticed an unsatisfactory prediction performance of our previous method. After careful investigation, we realized it was the algorithmic heuristics that could be further improved. Specifically, the previous method used a heuristic motivated by the FOIL's information gain (Fürnkranz et al. 2012), i.e., an MDL-based Foil-like compression gain; however, we later noticed extending the FOIL's information gain to multi-class situations will cause problem when using it to guide the search for rule growth, since it can be proven that the FOIL's information gain will only lead to rules with lower empirical entropy than the rules in the previous step. This specifically can cause problems when the dataset is noisy (in the sense that the Bayes-optimal classifier cannot achieve a perfect or near-perfect classification) and/or imbalanced. Therefore, we now implement a *learning-speed* heuristic, motivated by the "normalized gain" used in the CLASSY algorithm for rule lists (Proença and Leeuwen 2020); however, as we observe "normalized gain" often shrinks of the rule's coverage (the number of instances covered by the rule) too fast, we further introduced a diverse beam search with diverse "patience", in which the concept of patience is motivated from the PRIM method (Friedman and Fisher 1999), one of the first pioneer works for regression rules.

Second, one unique challenge of learning truly unordered rules is to both evaluate the quality of individual rules and the quality of the overlaps (i.e., whether the rules that form the overlap do not have similar enough outputs). However, this makes existing rules obstacles for the following search for more rules, as we elaborate in Section 3.5.2. In our previous work, we adopted a "two-stage" algorithm: in the first stage, the existing rules are ignored when calculating the heuristics, and next we use the results of the first stage as "seeds" for the second stage, in which the existing rules are considered in order to calculate the MDL-based score for evaulating the rules. However, we noticed that the first stage can output rules of which the number of covered instances is too small to be further refined when incorporating its overlaps with existing rules in the second stage. Therefore, we now combine these two stages by always keeping two "beams" in

the beam search, with one beam using the heuristic score that ignores the existing rules and the other incorporating the existing rules.

Third, given the necessity to evaluate the "potential" for the instances that are not covered by any rule so far during the rule set learning process, which is closely related to the claim made in the previous work (Fürnkranz and Flach 2005) that evaluating incomplete rule sets are a challenging and unresolved issue in rule learning in general, in our previous work we proposed to use a surrogate CART decision tree model to assess the potential for the uncovered instances. However, this approach turned out to be not very stable for this purpose in general, as we cannot afford the computational time for tuning the regularization parameter for the post-pruning for CART; in addition, when the dataset is very imbalanced, the performance of CART is sub-optimal and hence does not provide a satisfactory assessment. To resolve this issue, in this chapter we introduce a local constraint based on the local MDL compression gain, as discussed in Section 3.5.2.

Besides the algorithmic improvements, we substantially extended the experiments for the purpose of studying the truly unordered rules in detail. That is, the purpose of the experiments in the previous chapter was to show that, with the (soft) constraints of only allowing rules with similar outputs to overlap, truly unordered rule sets can achieve on-par predictive performance in comparison to rule sets methods that adopt ad-hoc schemes for conflicts caused by overlaps. However, in this chapter, we aim for studying 1) the predictive performance on a large scale of datasets, 2) whether the induced rules from data can be empirically regarded as truly unordered, in the sense that how large is the effect if we randomly pick one rule for predicting an instance covered by multiple rules, 3) whether the probabilistic estimates of individual rules can generalize to unseen (test) instances, such that the individual rules can be used as reliable and trustworthy explanations to the predictions, and 4) whether our rule sets need to sacrifice model complexity for being "truly unordered", given that our search space is essentially much more complicated in comparison to i) non-probabilistic rules, ii) rules for binary targets only, and iii) methods with the separate-and-conquer strategy that simplifies the search space by iteratively removing covered instances.

Moreover, we also made a moderate modification to our optimization score. If we simply regard the (vanilla definition of) MDL-based model selection criterion as a score based on the penalized maximum likelihood, the penalty consists of two

terms: 1) the code length of model and 2) the regret. However, it is well-known that, firstly by the implementation of C4.5 rules (Quinlan 2014), the code length of the model (the first term in the penalty) does not consider the redundancy in the model class of all possible rule sets, which can cause under-fitting. Specifically, during the implementation of our previous work, we simply exclude this "code length of the model" term, since we noticed that when not including this term, the predictive performance is in general better (at the cost of higher model complexity though). However, with the improved algorithm we propose in this chapter, we can now include the code length of model term for obtaining simpler models without sacrificing predictive performance.

Finally, we now formally defined TURS as a probabilistic model, while the previous chapter was not very precise in this regard. Also, we unified the nested overlap (i.e., one rule fully cover the other rule) and non-nested overlap of rules in the previous chapter, without using separate modelling schemes for the two cases respectively. Empirically, checking whether an overlap is a nested overlap is computationally expensive, while the empirical results show that the final model learned from the data rarely contains such nested overlap.

Algorithm 4: Learn a single rule

Input: Rule set M, dataset (x^n, y^n), beam width W
Output: The next rule S

all_candidate_rules ← [ ]
rules_for_next_iter ← [$\emptyset$] `// Initialize the rule with an ``empty" condition`
while *TRUE* **do**
  beam ← [ ] `// Initialize the beam for the beam search`
  auxiliary_beam ← [ ] `// Initialize the auxiliary beam (Section 3.5.2)`
  for *rule in rules_for_next_iter* **do**
    rule_candidates ← generate_candidates(rule) `// Enumerate literals and append to rule`
    categorized_candidates ← categorize(rule_candidates)
     `// Categorize into clusters by coverage (Section 3.5.2)`
    categorized_candidates ←
     MDL_local_testing(categorized_candidates) `// Defined in Section 3.5.2`
    top_W_candidates ← The candidate with the highest $r(.)$
                in each category of categorized_candidates
    categorized_candidates_auxiliary ← categorize(rule_candidates)
     `// Categorize into clusters by coverage excluding the instances covered by` M `(Section 3.5.2)`
    categorized_candidates_auxiliary ←
     MDL_local_testing(categorized_candidates_auxiliary)
    top_W_auxiliary ← The best candidate with the highest $R(.)$
              in each category of categorized_candidates_auxiliary
    beam.append(top_W_candidates)
    auxiliary_beam.append(top_W_auxiliary)
  if *stopping_criterion_is_met* **then**
    return the rule with the highest $r(.)$ in all_candidate_rules
  else
    beam ← top-W candidates in beam with the highest $r(.)$
     `// Reduce the number of rules to` W
    auxiliary_beam ←
     top-W candidates in auxiliary_beam with the highest $r(.)$
    all_candidate_rules.append(beam)
    rules_for_next_iter ← beam ∪ auxiliary_beam

Data	# rows	# columns	# classes	max. class prob.	min. class prob.
aloi	49534	28	2	0.970	0.030
backdoor	95329	197	2	0.976	0.024
backnote	1372	5	2	0.555	0.445
chess	3196	37	2	0.522	0.478
diabetes	768	9	2	0.651	0.349
glass-2	214	8	2	0.958	0.042
ionosphere	351	35	2	0.641	0.359
magic	19020	11	2	0.648	0.352
mammography	11183	7	2	0.977	0.023
musk	3062	167	2	0.968	0.032
optdigits	5216	65	2	0.971	0.029
pendigits-2	6870	17	2	0.977	0.023
satimage-2	5803	37	2	0.988	0.012
smtp	95156	4	2	1.000	0.000
thyroid	3772	7	2	0.975	0.025
tic-tac-toe	958	10	2	0.653	0.347
vowels	1456	13	2	0.966	0.034
waveform-2	3443	22	2	0.971	0.029
wdbc	367	31	2	0.973	0.027
anuran	7195	24	4	0.614	0.009
avila	20867	11	12	0.411	0.001
car	1728	7	4	0.700	0.038
contracept	1473	10	3	0.427	0.226
drybeans	13611	17	7	0.261	0.038
glass	214	11	6	0.355	0.042
heartcleveland	303	14	5	0.541	0.043
iris	150	5	3	0.333	0.333
pendigits	7494	17	10	0.104	0.096
vehicle	846	19	4	0.258	0.235
waveform	5000	22	3	0.339	0.329
wine	178	14	3	0.399	0.270

Table 3.2: Datasets for binary (top) and multi-class classification (bottom), publicly available on the UCI repository and the ADBench Python package; datasets from the latter are marked in *Italic*. We use the maximum and minimum of the marginal class probabilities to indicate the degree of class imbalance.

Appendix: Comparison to the Previous Work

Data	BRS	C45	CART	CLASSY	Ripper	CN2	DRS	IDS	TURS(diff to best)
aloi	0.519	0.398	0.621	**0.654**	0.485	0.569	0.500	0.509	0.619 (-0.035)
backdoor	0.917	0.990	0.979	0.996	0.976	**0.997**	—	—	0.995 (-0.002)
backnote	0.957	0.987	0.983	0.990	0.982	**0.993**	0.988	0.765	0.981 (-0.012)
chess	0.957	**0.998**	0.995	0.992	0.995	0.532	0.809	0.677	0.994 (-0.004)
diabetes	0.725	0.710	0.667	0.737	0.641	0.709	0.727	0.595	**0.750** (0)
glass-2	0.676	0.890	0.790	0.730	0.793	0.941	0.926	0.912	**0.949** (0)
ionosphere	0.802	0.882	0.851	0.886	0.911	**0.941**	0.712	0.786	0.904 (-0.037)
magic	0.767	0.869	0.799	**0.888**	0.819	0.698	0.774	0.507	0.887 (-0.001)
mammography	0.644	0.817	0.730	0.890	0.582	0.891	0.857	0.535	**0.897** (0)
musk	**1.000**	0.995	**1.000**	**1.000**	**1.000**	**1.000**	—	**1.000**	**1.000** (0)
optdigits	0.897	0.959	0.942	0.986	0.966	**0.992**	—	0.960	0.977 (-0.015)
pendigits-2	0.938	0.986	0.964	0.974	0.973	**0.996**	0.948	0.914	0.955 (-0.041)
satimage-2	0.922	0.914	0.915	0.929	**0.964**	**0.964**	0.699	0.867	0.909 (-0.055)
smtp	0.596	0.930	0.965	0.905	0.950	0.853	0.889	0.879	**0.972** (0)
thyroid	0.897	0.972	0.950	0.983	0.989	**0.998**	0.921	0.960	0.961 (-0.037)
tic-tac-toe	**1.000**	0.878	0.918	0.978	0.972	0.932	0.992	0.599	0.965 (-0.035)
vowels	0.854	0.693	0.773	0.796	0.758	**0.897**	0.813	0.748	0.817 (-0.08)
waveform-2	0.567	0.716	0.648	0.847	0.333	**0.886**	0.540	0.774	0.832 (-0.054)
wdbc	0.836	**0.999**	0.896	0.843	0.899	0.836	0.620	0.942	0.947 (-0.052)
anuran	—	0.995	0.944	0.968	**0.996**	0.962	0.945	0.602	0.973 (-0.023)
avila	—	**0.999**	0.977	0.987	0.993	0.920	0.729	0.617	0.990 (-0.009)
car	—	0.956	0.939	0.978	0.931	0.885	0.935	0.831	**0.980** (0)
contracept	—	**0.680**	0.597	0.653	0.607	0.598	0.598	0.549	0.658 (-0.022)
drybeans	—	0.970	0.943	0.977	0.979	0.929	0.975	0.591	**0.989** (0)
glass	—	0.970	**0.984**	0.975	0.940	0.937	0.926	0.793	0.967 (-0.017)
heartcleveland	—	0.603	0.572	**0.721**	0.509	0.694	0.611	0.513	0.695 (-0.026)
iris	—	0.960	0.975	0.970	0.962	0.977	0.954	0.810	**0.981** (0)
pendigits	—	0.982	0.974	0.986	0.983	0.991	0.967	0.522	**0.994** (0)
vehicle	—	0.856	0.789	0.870	0.859	0.858	0.764	0.579	**0.882** (0)
waveform	—	0.842	0.814	0.910	0.880	0.803	0.654	0.517	**0.915** (0)
wine	—	0.937	0.906	0.960	0.937	**0.973**	0.909	0.854	0.952 (-0.021)

Table 3.3: Average ROC-AUC scores obtained using cross-validation. BRS can only be applied to binary datasets, and DRS and IDS fail to get results on a few datasets, denoted as "—" in the table. Best ROC-AUC for each dataset is shown in bold and. The difference between the best ROC-AUC for each dataset and the ROC-AUC of TURS for the same dataset is shown in bracket.

data	TURS	TURS-RP	Diff.	%overlap	CN2	CN2-RP	Diff.	%overlap
aloi	0.619	0.62	-0.001	4%	0.569	0.578	-0.009	97%
anuran	0.973	0.969	0.004	28%	0.962	0.913	**0.048**	90%
avila	0.99	0.989	0.001	17%	0.92	0.915	0.004	45%
backdoor	0.995	0.995	0	0%	0.997	0.976	**0.021**	96%
backnote	0.981	0.98	0.001	20%	0.993	0.973	**0.019**	60%
drybeans	0.989	0.986	0.004	34%	0.929	0.908	**0.021**	94%
glass-2	0.949	0.949	0	0%	0.941	0.839	**0.102**	33%
heartcleveland	0.695	0.687	0.008	8%	0.694	0.663	**0.031**	61%
mammography	0.897	0.897	0	5%	0.891	0.806	**0.084**	86%
musk	1	1	0	0%	1	1	0	0%
optdigits	0.977	0.977	0	0%	0.992	0.972	**0.02**	92%
pendigits-2	0.955	0.955	0	0%	0.996	0.972	**0.024**	88%
satimage-2	0.909	0.909	0	0%	0.964	0.909	**0.055**	89%
smtp	0.972	0.972	0	0%	0.853	0.795	**0.058**	51%
thyroid	0.961	0.961	0	0%	0.998	0.941	**0.056**	87%
vehicle	0.882	0.878	0.004	15%	0.858	0.826	**0.033**	77%
vowels	0.817	0.817	0	1%	0.897	0.838	**0.059**	71%
waveform-2	0.832	0.832	0	9%	0.886	0.754	**0.132**	92%
wdbc	0.947	0.947	0	0%	0.836	0.596	**0.241**	69%
car	0.98	0.98	0.001	22%	0.885	0.794	**0.091**	91%
chess	0.994	0.994	0	23%	0.532	0.551	**-0.019**	95%
contracept	0.658	0.657	0.001	3%	0.598	0.572	**0.026**	100%
diabetes	0.75	0.748	0.002	11%	0.709	0.676	**0.033**	82%
glass	0.967	0.965	0.002	2%	0.937	0.937	0	0%
ionosphere	0.904	0.904	0	15%	0.941	0.895	**0.046**	55%
iris	0.981	0.98	0.001	5%	0.977	0.977	0	0%
magic	0.887	0.887	0	38%	0.698	0.738	**-0.04**	92%
pendigits	0.994	0.991	0.003	40%	0.991	0.982	0.009	76%
tic-tac-toe	0.965	0.965	0	7%	0.932	0.925	0.007	49%
waveform	0.915	0.905	0.009	51%	0.803	0.84	**-0.037**	77%
wine	0.952	0.952	0	0%	0.973	0.971	0.002	2%

Table 3.4: Average ROC-AUC for the predictions with and without "random picking", both for TURS and CN2. The difference between the two ROC-AUC scores are shown in bold if the difference is larger than 0.01. We further show the percentage of instances covered by more than one rule, denoted as %*overlap*.

Appendix: Comparison to the Previous Work

Data	BRS	C45	CART	CLASSY	RIPPER	CN2	DRS	IDS	TURS
aloi	**3**	2659.1	26952.8	52.3	26.2	2116	0	14	66.5
backdoor	**13**	701.5	2460.5	72.6	101.5	259.6	—	—	59.1
backnote	35.8	79.6	116.7	22.7	22.3	39.5	54	12.5	**14.2**
chess	**19.2**	250.2	340.5	33	57.9	297	54.2	14.5	58
diabetes	15.6	107.6	700.5	**5**	6.6	165.4	82.5	13.2	6.8
glass-2	10.8	19.7	6.7	3	5.9	1.8	37.2	15.5	**1**
ionosphere	31.2	59	86.6	5.6	12.5	25.1	440.7	12	**5.1**
magic	43.5	3211.8	20053.1	228.4	**87.1**	3351.9	44.7	18.5	227.8
mammography	3	273.7	1126.8	38.7	30.5	199.2	**24.4**	14	37.4
musk	6	4	**2**	**2**	**2**	**2**	—	9.3	**2**
optdigits	54.5	46.6	78.7	9.6	15.8	16.7	—	11.2	**7.6**
pendigits-2	14.8	48.3	66.9	**9.5**	12.7	19.9	136.5	14.1	12
satimage-2	15	14.8	17.8	7.9	9	9.9	524.8	12.4	**4**
smtp	3	20.9	34.4	5.6	7	19.2	16.7	13.9	**3**
thyroid	**3**	18.9	28.1	16.4	4.2	9.7	73.1	13.4	7.8
tic-tac-toe	**25.2**	411.9	410.8	29.2	42.3	81.1	101.7	13.5	28.9
vowels	25.2	58.9	147.8	14.1	13.5	22.6	141.9	14.3	**8.7**
waveform-2	4.2	165.6	406.3	19.3	20.5	79.4	522.2	**11.2**	12
wdbc	9	8	2.5	3.2	**2**	4.4	337.6	7.6	**2**
anuran	—	80.9	1284.3	90.7	**11**	264.8	346.1	12.4	100.2
avila	—	3460.4	7795.9	939.8	**707.2**	1297.4	181.3	14	726
car	—	659.8	776.1	**56.4**	171.7	72	307.6	13.5	132
contracept	—	1268.1	5536.2	12	14.7	221.8	**5**	16.8	12.6
drybeans	—	2481.6	7039.4	**105.8**	153.3	1280.6	202.3	13.2	182.4
glass	—	24.8	20	**5.2**	14	7.8	171	11.3	6
heartcleveland	—	245.7	410.2	6.6	5.5	42.1	522.2	14	**5.7**
iris	—	15.2	13	2.8	5.9	8.4	26.7	10.4	**2.3**
pendigits	—	1274.9	1689.7	**142.3**	260.9	430.4	561.7	15.6	174.5
vehicle	—	560.6	760.3	25.4	46.9	175.5	741.1	11.6	**23**
waveform	—	2782.9	3520	**125.7**	143	970.5	75.6	14.4	160.6
wine	—	22	15.5	5	8.9	7.5	103.8	14.4	**4.6**

Table 3.5: Total number of literals in the rule set, rule list, or decision tree. Smaller fonts indicate that the model learned by a certain algorithm gives the ROC-AUC score substantially worse than TURS does, and the results in bold indicate the smallest total number of literals (excluding those models with substantially worse ROC-AUC scores).

Local testing	# rules	rule length	ROC-AUC	MDL-based score	train/test prob. diff.
No	12.48(±1.56)	5.597(±0.42)	0.722(±0.02)	2191.189(±65.91)	0.049(±0.01)
Yes	1(±0)	1(±0)	0.724(±0.01)	2050.087(±68.88)	0.007(±0)

Table 3.6: Results of ablation study on local testing. We report the mean (± standard deviation) over 100 repetitions.

Chapter 4

Case Study: Towards Interactive Rule Learning for ICU Readmission Analysis

Chapter Abstract

Interactive machine learning systems that can incorporate human feedback for automatic model updating have great potential use in critical areas such as health care, as such systems can combine the strength of data-driven modeling and the prior knowledge from domain experts. Designing such a system is a challenging task as it must enable mutual understanding between humans and computers, which hence relies on interpretable and specifically easily comprehensible models. Specifically, we consider the problem of incorporating human feedback for model updating in rule set learning for the task of predicting readmission risks for ICU patients. Building upon the TURS model described in the previous chapters, we further propose a certain format for feedback for rules, together with an automatic model updating scheme. We conduct a pilot study and demonstrate that the rules obtained by updating the TURS model learned from the ICU patients' data can empirically incorporate human feedback without sacrificing predictive performance. Notably, the updated model can exclude conditions of rules that ICU physicians consider clinically irrelevant, and thus enhance the trust of physicians.

4.1 Introduction

In critical areas such as health care, developing machine learning models that domain experts can comprehend and trust potentially has great societal impact. Specifically, in intensive care units (ICU) where patients are monitored intensively, conditions of patients are to a large extent recorded digitally, which provides the foundations for building decision support systems with data-driven models (Hond et al. 2023).

We consider the problem of predicting the probability of readmission to the ICU within a short period (7 days) after a patient is discharged from the ICU and moved to a normal ward. Such readmission risk for patients is clinically relevant, as it is observed that patients who are readmitted often become much worse in comparison to their condition when they were in the ICU previously (Kramer et al. 2013; Woldhek et al. 2017). Thus, the readmission itself is a key factor that is highly correlated with the patient's condition; as a result, predicting the readmission risk can both facilitate efficient ICU resource management and prevent discharging patients improperly. In practice, beds in the ICU are a very scarce and costly resource; thus, discharging patients from the ICU smartly can help distribute the resource to patients who need it more.

As physicians are responsible for estimating the risk of discharging a patient from the ICU, data-driven models only brings benefits if physicians trust the model and are willing to use it in practice. To build trust, the data-driven model needs to have interpretability for domain experts to comprehend what is going on (Li et al. 2023). Further, beyond interpretability, the situation when physicians and machine learning models disagree must be properly handled (Holzinger 2016; Mosqueira-Rey et al. 2023; Teso and Kersting 2019). That is, when the model gives a probabilistic prediction together with explanations, what if the physician disagrees with the prediction and/or the explanation? For instance, the model could identify a factor that is known to be irrelevant clinically as important for predicting readmission risk for a single patient. In this situation, it would be ideal if the physician would give this feedback to the machine learning model; further, if the model can be automatically updated when receiving the feedback from human, the physician could trust the model next time when the model gives the same explanation and prediction for a similar patient in the future.

Thus, interaction between humans (i.e., physicians in the ICU in this case) and the machine learning model is crucial in such a scenario, which requires the human to understand the machine, and at the same time, the machine to understand the human. While rule-based models, and especially truly unordered rule set (TURS) models, are in principle comprehensible to domain experts, the challenges remain unresolved that *1) how and in what formats feedback from domain experts can be incorporated, and 2) how rule-based models can be updated according to human feedback*.

To tackle these challenges, we introduce a human-guided rule updating scheme based on the TURS model. The TURS model paves the way towards an interactive rule learning process with the following two advantages over existing methods for learning rule lists (in which rules are explicit ordered) and rule sets (in which external and ad-hoc methods are mostly used to handle the conflicts caused by overlaps).

The first advantage is that rules in the TURS model can be empirically regarded as truly unordered and hence independent from each other. Thus, deleting and/or editing one rule (that a domain expert dislikes) has little influence on other, potentially overlapping rules. In contrast, for rules with (implicit) orders obtained by other existing methods, editing or deleting one rule may cause "a chain of effects" on how instances covered by other rules are modeled. Secondly, the TURS model reduces the workload for domain experts to find out which rules need to be edited. Specifically, when comprehending a single rule, there is no need to go over all other rules that are ranked (explicitly or implicitly) higher, as unlike other existing methods, our TURS model does not impose any order among rules.

In the following, we conduct an empirical pilot study by applying the TURS model to a dataset collected at the ICU of Leiden University Medical Center (LUMC) in the year 2020. To this end, we ask a domain expert from LUMC to identify rules with clinically irrelevant variables, and we also propose an updating scheme for the TURS model.

4.2 Updating Rule Sets with Human Feedback

We now describe in what format we allow ICU physicians to give feedback, and how the TURS model can be updated automatically with the feedback.

4.2.1 Human feedback format

Although it seems tempting to allow feedback in flexible formats (and the most flexible format would be in natural language), we argue that it is desirable to constrain human feedback to have certain formats, in order to transform the feedback into *transparent human guidance* to the algorithm for updating the model. In other words, we aim to propose certain human feedback formats so that the consequence of such human feedback can be easily explained to domain experts.

However, such feedback format should also allow domain experts to express clearly and sufficiently why they dislike the current model. This requires a deep understanding about what might cause dissatisfaction from domain experts. Hence, how to design such feedback formats may depend on the application task at hand, and may require collaboration between computer scientists and domain experts.

Focusing on the task of ICU readmission risk analysis, we constrain ourselves to a simple yet fundamental feedback format and leave as future work incorporating other feedback formats. Formally, given a truly unordered rule set model with K rules denoted as $M = \{S_1, ..., S_K\}$, we consider feedback from domain experts in the following form: *remove rule* S_j *due to irrelevant variables* $\{X_i\}_{i \in I}$, in which S_j denotes a single rule and I denotes an index set. Notably, feedback in this format contains not only information regarding whether a rule is disliked, but also the reason why a rule is disliked.

4.2.2 Updating a rule set

We now present how we can equip the TURS model with an "self-updating" scheme after receiving feedback from a domain expert.

Removing a rule. Given the rule set $M = \{S_1, ..., S_K\}$, assume that a domain expert gives the feedback that rule S_i does not make sense as it contains irrelevant variable X_j. Then, removing S_i from M is straightforward as there exist no implicit or explicit orders among rules. That is, following the procedure of formalizing a rule set as a TURS model, we simply have a new rule set $M' = M \setminus \{S_i\}$, for which the likelihood can be calculated according to how the TURS model is defined.

We next analyze for which instances the empirical class probabilities are affected. First of all, when S_i is eliminated from model M, it has an effect on the estimated probabilities of both 1) instances covered by S_i, and 2) instances not covered by any rule (i.e., covered by the else rule). Specifically, instances previously covered by S_i only (i.e., before removing S_i) are now combined with instances originally covered by the else-rule, which are now used for obtaining new class probability estimates for the new else rule after eliminating S_i from M. Meanwhile, for instances covered by the overlap of S_i and some other rule(s), the class probability estimates will be updated accordingly.

Learn a new rule with constraint. Building upon the new TURS model M', we next consider learning a new rule that can be added to M' as the replacement for the removed rule, for which we leverage the dual-beam diverse-patience algorithm for learning the next "best" rule given the current status of a rule set, as proposed in Chapter 3.

As the conditional likelihood of class labels can be calculated given the dataset and the rule set M' given the definition of the TURS model, the MDL-based model selection score for the rule set M' can be calculated accordingly. Further, when adding a rule S' to M', the model selection score can be calculated for $M' \cup \{S'\}$ as well.

Thus, the algorithm can search for the next best rule S' such that when adding S' to M' the *learning speed score* $r(S')$ is optimized (as defined in Chapter 3), in which $r(S')$ measures how much the MDL-based model selection score decreases per extra covered instance when adding S' to M'.

4.3 An Empirical Pilot Study

We conduct a pilot study in collaboration with Leiden University Medical Center (LUMC) using the real-world patient dataset to showcase how the TURS model together with the model updating scheme can be used for interactive rule learning with humans in the loop. We next describe the experiment setup and present our results.

4.3.1 Experiment setup

Dataset description. We specifically considered the dataset collected at the ICU of LUMC in the year 2020, in which the patients who are readmitted within 7 days are labelled as "positive".

The original dataset is multi-modal and contains information in different forms, including time series measurements (e.g., cardiology monitor records), lab results over time (e.g., blood tests), medication use records, as well as static information for each patient (e.g., age, gender, etc). This dataset was described and pre-processed into a tabular dataset by an external expert in previous research (Van der Meijden 2021). The resulting processed dataset was further split randomly for training and test, which contains 9737 and 2435 patients respectively (approximately 80%/20% splitting), with 550 feature variables. The dataset is very imbalanced, as the overall probability of readmission is roughly 0.07.

Human feedback collection. We ask one domain expert from LUMC to give feedback to the rules, with the procedure as follows. First, a TURS model is learned on the training set, with beam width set as 5 and the number of candidate cut points (for continuous-valued features) set as 20, which is the "default" setting that we also used in Chapter 3.

Second, the rule set is shown to the domain expert; specifically, the condition of each rule together with the class probability estimates (obtained using the training set) are shown to the domain expert. Moreover, the algorithm configuration (e.g., the beam width) is revealed to the domain expert as well.

Next, we ask the domain expert to go through each of all rules, and to give feedback to the ruleset in the format as we described in Section 4.2. Subsequently, the feedback is used to update the TURS model, and we use the test set of the ICU dataset for assessing the predictive performance of the TURS model before and after the human feedback. We refer to the latter as the human-guided model. Lastly, note that the test set of the whole dataset is only used for this final assessment step, and therefore the domain expert has no access to it during the procedure of giving feedback to rules.

4.3.2 Rule set for the ICU dataset

Learning a TURS model using our proposed method in Chapter 3, we obtain a surprisingly simple rule set with 5 rules only, which has average rule length of 2. The obtained rule set is shown in Table 4.1:

Rule Conditions	Prob. of Readmission	# Patients
Ureum-max-all $\geq$ 12.1 Ademfrequentie-median-value-last24h $\geq$ 23.5	0.223	494
APTT-max-all $\geq$ 43.1 Ureum-mean-all $\geq$ 16.338	0.199	548
Leukocyten-mean-last $\geq$ 20.81	0.162	464
Kalium-count-first $\geq$ 6.0 specialty-Organization-value-sub-ICCTC = FALSE	0.131	1979
Trombocyten-count-first $\geq$ 2.0 Ureum-last-last $<$ 9.2 specialty-Organization-value-sub-ICCTC = TRUE	0.019	3922
None of the above	0.059	3220

Table 4.1: Rule sets describing the probability of readmission for LUMC ICU patients.

The literals contain feature names that are mostly consisting of three parts, with the first part indicating the basic meaning of this feature variable (in Dutch). For instance, "Ureum" indicates the "Urea" in blood. The second part of feature names indicates how the results are aggregated, among which "count", "mean", "median", and "max" are commonly used. Last, the third part of feature names indicates the time window for which the aggregated values are obtained, in which "first" represents the first 24 hours, "last" represents the last 24 hours, and "all" represents the whole period in ICU. A detailed explanation of the feature names can be found in previous work (Van der Meijden 2021).

4.3.3 Rule-based competitor methods

To benchmark the performance of the TURS model induced from the training dataset, we apply several commonly used probabilistic rule-based models to the ICU dataset. The motivation for such benchmark is to show that the TURS model has competitive predictive performance and thus implicitly describes the data relatively well, which is the foundation for involving humans in the loop.

The comparative predictive performance is summarized in Table 4.2. Notably, the TURS model shows advantages over competitor methods in several aspects. First, the results with respect to ROC-AUC and PR-AUC show that the ICU

dataset is difficult to model using widely used rule-based models (as listed in the table), since the ROC-AUC of C4.5 and RIPPER are roughly equal to 0.5. Further, the TURS model shows its robustness in achieving the best ROC-AUC and PR-AUC, and notably with significantly simpler rules (except when compared to RIPPER, which seriously "underfits" the data).

Moreover, rules in the TURS model generalize best to the unseen instances in the test set (excluding RIPPER for its low ROC-AUC scores). Specifically, we calculate the difference between the class probability estimates obtained using the training and test dataset, as also reported in the table. We hence conclude that the probability estimate for each single rule of the TURS model shown to physicians are most reliable and trustworthy.

Algorithm	CN2	CART	RIPPER	C4.5	TURS
ROC-AUC	0.641	0.690	0.514	0.539	0.705
PR-AUC	0.114	0.137	0.084	0.076	0.164
Train/test prob. diff.	0.041	0.031	0.001	0.054	0.006
# rules	851	25	1	249	5
Avg. rule length	2.5	4.2	5.0	16.8	2.0

Table 4.2: Rule-based model results on ICU dataset.

4.3.4 Human-AI collaboration

We now showcase that our TURS model can be equipped with the model updating scheme to generate human-guided rule sets. Notably, our approach is very different than existing model editing approaches (Wang et al. 2022), as the end user is not allowed to directly edit the model in our model updating scheme; instead, we only allow user to provide feedback, and the updated model is still learned in a data-driven manner. That is, we let the data always take the leading role, in order to avoid arbitrary (or adversarial) model editing.

Specifically, we consider the rule set obtained in Section 4.3.2, and we collected two pieces of feedback from the domain expert: 1) the domain expert dislikes the 5th rule due to the first variable, and 2) the domain expert dislikes the 3rd rule which contains only one literal.

We thus discard the 5th rule from the rule set, and we next search for a new rule to be added to the rule set, with the constraint that the first variable in the 5th

rule must not be included. We present the new human-guided rule together with the original rule in Table 4.3. We show that our TURS model indeed makes such an interactive process possible, and specifically that it can handle feedback that can be transformed into constraints with respect to excluding certain variables. Further, we demonstrate that for the rule set induced from the ICU patients' dataset, editing a rule based on the human feedback (without the necessity to modify other 'overlapping' rules), can indeed discard certain variables but at the same time keep the predictive performance at the same level.

Note that the updated rule and the original rule are coincidentally very similar; that is, the feedback to the TURS model is only about discarding the first literal of the 5th rule, without asking it to keep the other literals and/or variables in the original rule.

Human-guided	No	Yes
Rule	If Trombocyten-count-first $\geq$ 2.0; Ureum-last-last $<$ 9.2; specialty-Organization-value-sub-ICCTC = TRUE $\rightarrow$ Probability of Readmission: 0.019, Number of patients 3922	If Leukocyten-count-first $\geq$ 2.0; Ureum-last-last $<$ 9.2; specialty-Organization-value-sub-ICCTC = TRUE $\rightarrow$ Probability of Readmission: 0.019, Number of patients 3958
ROC-AUC (rule set)	0.705	0.706
PR-AUC (rule set)	0.164	0.164

Table 4.3: Comparison between the rule before and after a domain expert feedback, together with the ROC-AUC and PR-AUC of the resulting new rule set. Changes in rules conditions before and after human feedback are shown in red and blue respectively.

Next, for examining the effect of the second feedback, we remove the 3rd rule from the original purely data-driven rule set, and search for another rule by excluding the variable "Leukocyten-mean-last" from the search space. We present the results in Table 4.4, which shows that the new rule covers 375 more patients than the original rule. Again, without the need for further modifying other rules, editing the 3rd rule in the original rule set with the updated rule keeps the ROC-AUC and PR-AUC at the same level.

Human-guided	No	Yes
Rule	Leukocyten-mean-last $\geq$ 20.8 $\rightarrow$ Probability of Readmission: 0.162, Number of patients 464	CRP-mean-last-missing $=$ 1 $\rightarrow$ Probability of Readmission: 0.030, Number of patients 839
ROC-AUC (rule set)	0.705	0.704
PR-AUC (rule set)	0.164	0.172

Table 4.4: Another comparison between the rule before and after a domain expert feedback.

4.4 Conclusion and Discussion

We studied the problem of estimating readmission risk for patients in ICU as an applied machine learning task. In order to resolve the difficult situation when domain experts (physicians) dislike certain rules, which can result in the lack of trust for such data-driven models, we aimed for developing a human-guided rule learning scheme based on our method for learning truly unordered rule set (TURS) models.

We presented a pilot empirical study using the patients data collected at Leiden University Medical Center (LUMC) in the year 2020. Specifically, we firstly presented the learned rule set from the ICU dataset, and compared the predictive performance against other widely used rule-based competitor models, which demonstrated the superiority of the TURS model in terms of both predictive performance and model complexity. This result set the foundation for using the TURS model as a basis for interactive rule learning.

Next, we asked a domain expert from LUMC to give feedback to the purely data-driven rules, and we proposed a simple model updating scheme to incorporate the feedback to induce human-guided rules. We showcased that such a process can lead to new rules as replacements for rules that the domain expert disliked, without sacrificing the predictive performance of the whole model. Notably, the properties of the TURS model enables straightforward, transparent, and efficient model editing, without the need for re-training other rules in the model. We next discuss potential future research directions.

4.4.1 Discussion for future work

We have shown that the truly unordered rule set (TURS) model is "ready" for interactive rule learning, i.e., in a straightforward way it can be equipped with a model updating scheme that incorporates human feedback in certain formats. Following this research line, it may be with great potential to explore the following research questions.

User feedback formats. One natural but crucial question is in what formats we allow domain experts to give feedback to the data-driven model, and further how to inspire and elicit feedback with tools that allow an end user to investigate the data and the rule-based models.

For instance, it may be beneficial to allow domain experts to "zoom in" for each single rule, and examine values of other features for each corresponding subset of patients. While all instances in each rule share the same class probability estimate, domain experts may find one single "typical" patient who should have a different probability estimate than the rest. This may induce feedback in the form of "modifying a given rule by excluding a certain instance from its cover".

Further, we could allow domain experts to name risky factors within each rule; i.e., to allow the domain experts to suggest informative feature to be included in a single rule. Thus, we may allow feedback in the form of "for all patients covered by this rule, those patients whose feature value for variable X_i is larger than a certain threshold may have a higher risk of readmission". Such feedback is useful for 1) obtaining single rules with variables that are congruent with the domain knowledge, and 2) more interestingly, understanding the limits of the data (since the "best" rule with the suggested variables may result in a "worse" score according to the model selection criterion).

Transparent model updating. Introducing the human in the loop extends the meaning of transparency of a machine learning method. Previously, transparency roughly referred to whether the process of obtaining a model based on a given dataset is comprehensible to humans; in contrast, we argue that transparency is also applicable to describing whether the process of model updating based on human feedback is comprehensible to humans. Thus, it is a natural question to ask whether the trust between domain experts and data-driven models depends not only on the interpretability and transparency of the model but also on that

of the model updating scheme.

Further, while it is very transparent to incorporate human feedback as constraints like those we proposed, other ways of processing human feedback are to be explored. For instance, except for considering human feedback in certain formats as constraints, we may also translate human feedback to "prior" preferences.

User study for trust. Trust between domain experts like ICU physicians and data-driven models is a fundamental requirement for deploying a decision-support system in critical areas like health care, because, for instance, if physicians do not trust the data-driven model, they tend to simply ignore the data-driven predictions.

While the goal of involving humans in the loop to obtain human-guided rules to increase the trust by obtaining rules that are (more) congruent with the domain knowledge, whether trust is indeed increased can only be evaluated empirically via user studies. Thus, an interesting research question is how to formally define trust in the task of predicting readmission risk, inevitably with subjectiveness. As a result, it remains a challenge to design questionnaires for evaluating the trust between domain experts and data-driven models.

Chapter 5

Summarizing Two-dimensional Data with MDL-based Discretization by Histograms

This chapter has been published as Yang, L, Baratchi, M, and van Leeuwen, M **Unsupervised discretization by two-dimensional mdl-based histogram**. *Machine Learning, 2023: 1-35.*

Chapter Abstract

Unsupervised discretization is a crucial step in many knowledge discovery tasks. The state-of-the-art method for one-dimensional data infers locally adaptive histograms using the minimum description length (MDL) principle, but the multi-dimensional case is far less studied: current methods consider the dimensions one at a time (if not independently), which result in discretizations based on rectangular cells of adaptive size. Unfortunately, this approach is unable to adequately characterize dependencies among dimensions and/or results in discretizations consisting of more cells (or bins) than is desirable.

To address this problem, we propose an expressive model class that allows for far more flexible partitions of two-dimensional data. We extend the state of the art for the one-dimensional case to obtain a model selection problem based on the normalized maximum likelihood, a form of refined MDL. As the flexibility of our model class comes at the cost of a vast search space, we introduce a heuristic algorithm, named PALM, which partitions each dimension alternately and then merges neighboring regions, all using the MDL principle. Experiments on synthetic data show that PALM 1) accurately reveals ground truth partitions that are within the model class (i.e., the search space), given a large enough sample size; 2) approximates well a wide range of partitions outside the model class; 3) converges, in contrast to the state-of-the-art multivariate discretization method IPD. Finally, we apply our algorithm to three spatial datasets, and we demonstrate that, compared to kernel density estimation (KDE), our algorithm not only reveals more detailed density changes, but also fits unseen data better, as measured by the log-likelihood.

5.1 Introduction

Discretization, i.e., the transformation of continuous variables into discrete ones, is part of numerous data analysis workflows, making it a crucial step for a wide variety of applications in knowledge discovery and predictive modeling. However, many different discretization methods exist and it is often not easy to determine which method should be used. As a result, naïve methods such as equal-length and equal-frequency binning are still widely used, often with the number of bins chosen more or less arbitrarily, which can lead to suboptimal discretization.

A good discretization strikes a balance between the amount of preserved information and the complexity of the representation of the discretized data, so as to avoid discretizations that are either too coarse—resulting in severe loss of information—or too fine-grained—resulting in a bin per data point in the extreme case.

Achieving an optimal balance has been thoroughly studied for supervised discretization, i.e., discretization using additional information from a target variable. Optimal discretizations have been formalized using 1) statistical quantities, e.g., Pearson's chi-square (Boullé 2004), 2) information-theoretic scores based on entropy or the minimum description length (MDL) principle (Fayyad and Irani 1993; Jin et al. 2009), and 3) Bayesian approaches (Boullé 2006).

In contrast, unsupervised discretization, which does not assume a target variable, has long been understudied (Kotsiantis and Kanellopoulos 2006). It serves a different purpose: supervised discretization aims to reduce the loss of information about the distribution of the target variable conditioned on the features (Boullé 2004; Fayyad and Irani 1993; Kerber 1992), whereas unsupervised discretization aims to preserve information about the probability distribution of the variable to be discretized (Biba et al. 2007; Schmidberger and Frank 2005).

This makes histograms well-suited to unsupervised discretization, and particularly adaptive histograms. An adaptive histogram is a probabilistic model that approximates probability density by piecewise constant densities, partitioning the data into bins such that 1) the probability density within each bin is approximately uniform (otherwise finer bins are needed), and 2) probability densities of neighboring bins are significantly different (otherwise they should be merged). Kontkanen and Myllymäki (2007b) formalized this goal for one-dimensional adaptive

histograms based on the minimum description length (MDL) principle (Rissanen 1978), which is now considered to be the state-of-the-art univariate discretization method (Kameya 2011; Marx et al. 2021; Nguyen et al. 2014).

The MDL principle (Grünwald and Roos 2019; Rissanen 1978) is arguably one of the best off-the-shelf approaches for model selection tasks such as selecting a histogram model for given data, as it provides a means to naturally trade-off goodness-of-fit with model complexity. It achieves this by defining the "best" probabilistic model for given data as the model that results in the best *compression* of data and model together, which has been widely used in data mining and machine learning tasks (Galbrun 2020).

Flexible multi-dimensional discretization. Traditional discretization methods are defined for one-dimensional (or univariate) data, and multi-dimensional (or multivariate) data is typically discretized by separately and independently discretizing each dimension, which ignores any dependencies between the dimensions. Multivariate discretization methods aim to take such dependencies into account, but they suffer from two problems. First, most methods focus on supervised discretization (Bay 2001; Ferrandiz and Boullé 2005; Kurgan and Cios 2004; Kwedlo and Kretowski 1999). Second, existing methods produce an adaptive grid based on the Cartesian product of the discretization results of individual dimensions. This approach ignores that the density of one dimension may change more drastically for certain values of another dimension; hence, appropriate binning of one dimension may depend on the values of the other dimensions.

For instance, consider a two-dimensional synthetic dataset sampled from a mixture of Gaussians as shown in Figure 5.1 (leftmost)[1]. To adequately discretize data from this distribution, the binning of the x-axis should be different depending on whether y is above or below the black dashed line, in order to capture the different density changes for the Gaussian distribution (above) and the Gaussian mixture (below). Similarly, the binning of the y-axis should be different depending on whether x is left or right to the red dashed line. This motivates us to consider partitions that are more flexible than adaptive grids: we consider all partitions that can be obtained by clustering the "cells" of a fine-grained fixed-grid. The remaining three plots in Figure 5.1 show the density plots obtained by

[1]For reproducibility, the data is generated by the mixture of $N[\binom{0}{1}, \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)]$, $N[\binom{1.5}{4}, \left(\begin{smallmatrix}2 & 0\\ 0 & 1\end{smallmatrix}\right)]$, $N[\binom{3}{0}, \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)]$, $N[\binom{7}{2}, \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)]$ with all mixing coefficients 0.25; sample size is 40 000.

1) IPD (Nguyen et al. 2014), the state-of-the-art multivariate unsupervised discretization method, 2) the one-dimensional MDL-based histogram method (Kontkanen et al. 1997) applied independently on each dimension, and 3) our method. Our method produces the density estimation that most resembles the shape of the original contour, as we allow the bins of one dimension to depend on the value of another dimension.

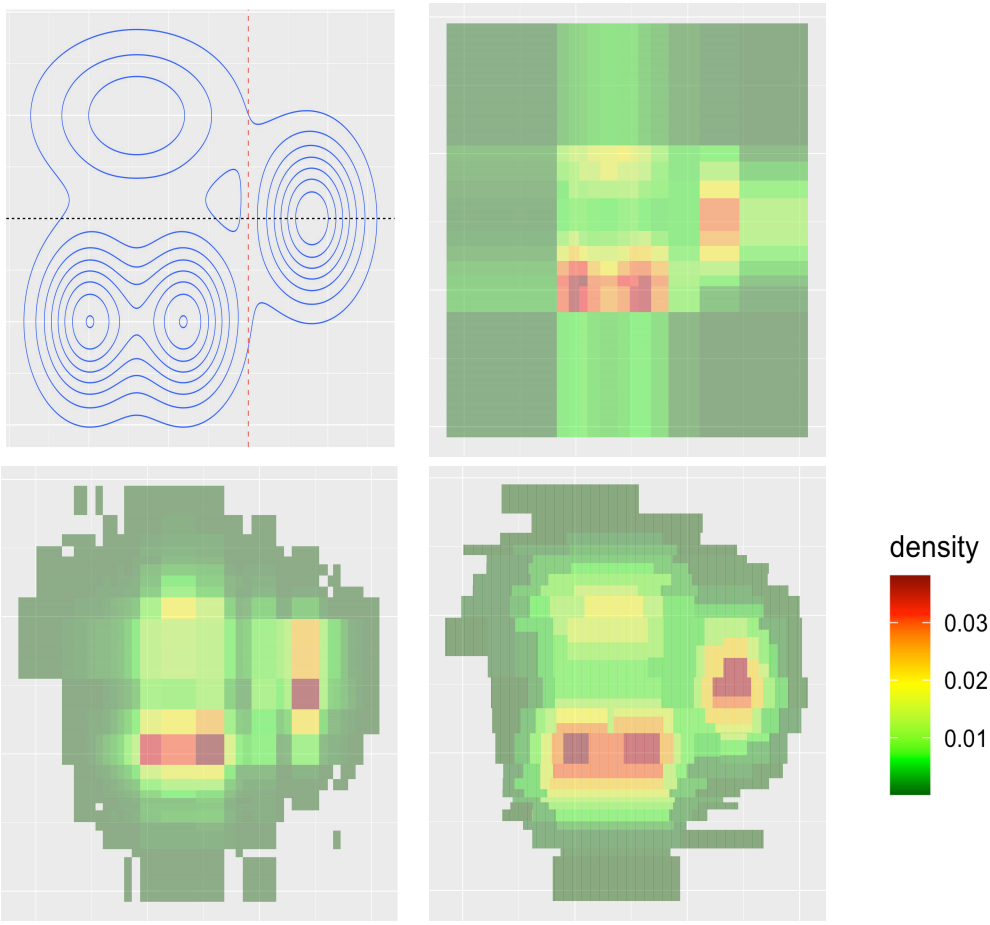

Figure 5.1: Distributions of a two-dimensional dataset simulated from a mixture of Gaussian distributions; from top-left to bottom-right: 1) true probability density contour, 2) partitioning by IPD (Nguyen et al. 2014), 3) partitioning by separately discretizing each dimension with the MDL histogram (Kontkanen and Myllymäki 2007b), 4) flexible partitioning by PALM, our algorithm.

Approach and contributions. We consider the problem of learning two-dimensional

histogram models that enable far more flexible partitions than regular adaptive grids. That is, we allow any partition that can be obtained by iteratively merging adjacent cells of a fixed grid, which allows for learning models that provide accurate density estimates while not having more bins than strictly necessary (thereby avoiding overfitting and providing clear region boundaries, i.e., adjacent bins must have different density estimates).

We formalize the two-dimensional histogram construction problem as a model selection task using the MDL principle. For this we build on the one-dimensional MDL-based histogram selection problem as introduced in the seminal work by Kontkanen and Myllymäki (2007b), because it is both theoretically elegant and practically fast. Specifically, it adopts the *normalized maximum likelihood* (NML) encoding scheme, a form of *refined MDL* (Grünwald 2007; Grünwald and Roos 2019) that provides minimax regret, and employs a fast dynamic programming algorithm to find the optimal solution.

The existing approach for one-dimensional histograms cannot be trivially extended to multiple dimensions though, hence we make a number of technical contributions.

First, we solve the challenge of computing the so-called *parametric complexity* (Grünwald and Roos 2019) for the multi-dimensional case.

Second, we observe that efficiently finding the MDL-optimal two-dimensional histogram is infeasible and propose PALM, a heuristic algorithm for learning two-dimensional histograms. PALM combines top-down (partition) and bottom-up (merge) search strategies by 1) first partitioning the data by iteratively splitting regions, and 2) then iteratively merging neighboring regions if their densities are similar. In each step, the MDL principle is used as decision criterion; as a result, our algorithm requires neither hyper-parameters[2] nor any pre-defined stopping criterion to be specified. It automatically adapts to both local density structure, as shown in the example in Figure 5.1 and, later, in Sections 5.7 and 5.8.

Third, we make several improvements to the dynamic programming algorithm used for the one-dimensional MDL histogram, which we use as a building block for our algorithm. Specifically, as described in Section 5.5, we 1) correct a minor theoretic flaw related to computing the code length that is needed to encode the histogram model, and 2) reduce the time complexity by simplifying the dynamic

[2]The precision with which the data is recorded can be used to set the granularity of the initial base grid.

programming recursion.

We perform extensive experiments to show that our algorithm 1) accurately recovers ground truth histograms, 2) approximates well ground truth partitions that are not within the model class, and 3) outperforms IPD (Nguyen et al. 2014), the state-of-the-art algorithm for unsupervised multi-dimensional discretization. Further, case studies on spatial data show that, compared to kernel density estimation (KDE), our algorithm not only reveals more detailed density changes, but also fits unseen data better, as measured by the log-likelihood.

We restrict the scope of this chapter to two-dimensional data for three reasons. First, two-dimensional discretization methods have many potential applications in the domain of spatial data analysis, e.g., using GPS data, where ad-hoc discretization methods are still widely used (Cao et al., 2014). The case studies demonstrate that our method can successfully reveal interesting patterns from GPS data. Second, as our approach uses more flexible partitions than adaptive grids, the search space is very large even for two-dimensional data. Our algorithm for the two-dimensional case should be regarded as a step towards solving the algorithmic challenge for higher dimensions, but does not solve it completely. Third, focusing on the two-dimensional case allows us to more easily examine the results empirically, e.g., to verify desired properties such as adaptivity to sample size and local density structure.

5.2 Related work

We briefly review previous work concerning discretization methods, histogram models, and tree-based models for density estimation.

Unsupervised univariate discretization. Most unsupervised univariate discretization methods are rather straightforward and concern equal-width or equal-frequency binning, which in practice usually involve ad-hoc choices for the number of bins or for the frequency in each bin.

Clustering techniques such as k-means (Friedman et al. 2001) or Bayesian clustering (Kontkanen et al. 1997) are also used in discretization; however, they ignore the possible heterogeneity within the cluster and choices of hyper-parameters are usually required.

More advanced criteria rely on density estimation and specifically construct-

ing adaptive histograms. Apart from the MDL-based histogram (Kontkanen and Myllymäki 2007b) already mentioned in Section 5.1, Schmidberger and Frank (2005) proposed to construct adaptive histograms by recursive binary partition with cross-validation. A local heuristic is used to decide the cut point, and cross-validation is used to choose the number of intervals; in contrast, the MDL-based histogram (Kontkanen and Myllymäki 2007b) uses a global score with a dynamic algorithm that optimizes the cut points and the number of bins simultaneously. Moreover, an adaptive histogram can also be selected as the one whose density estimation result is closest to the result of *kernel density estimation* (Biba et al. 2007), where cross-validation is used to prevent overfitting. As the true density is apparently not known, cross-validation is performed by Monte Carlo sampling-based methods. However, cross-validation is known to be computationally expensive, and the influence of choosing different kernels on discretization is not reported.

Bayesian approaches have been widely used in adaptive histograms (Gasparini 1996; Liu and Wong 2014; Lu et al. 2013; Scricciolo 2007; Van Der Pas and Rocková 2017). These methods treat all possible histograms as the model class and put a prior distribution on it, and the resulting posterior distribution is directly used for density estimation (by calculating the marginal distribution). Therefore, although these Bayesian approaches often provide theoretic guarantees as density estimation methods, they do not provide an individual adaptive histogram that can be used for discretization.

Unsupervised multivariate discretization. Since discretizing each dimension of multivariate data independently will ignore the dependencies among different dimensions, some methods attempt to reduce the dependencies by PCA- or ICA-based methods (Kang et al. 2006; Mehta et al. 2005)[3]. However, as both methods are based on *linear transformation* of the random vector, they may fail to eliminate nonlinear dependencies. Note that extending these methods to nonlinear PCA or nonlinear ICA may not be suitable for unsupervised discretization tasks, as the uniform distribution is not invariant under nonlinear transformation, and hence we cannot obtain an adaptive histogram of the original data by inversely transforming the adaptive histogram constructed on the nonlinearly transformed data.

[3]Note that the ICA-based method (Kang et al. 2006) is designed for supervised discretization, but we noticed that the ICA transformation there is not restricted to supervised discretization only.

Lud and Widmer (2000) proposed the so-called "relative unsupervised discretization". The core of this method is to perform clustering on an individual dimension, using different subsets of values. These different subsets are obtained by filtering the dataset using other dimensions, in order to keep the dependency among different dimensions. However, this method does not control the information loss about the probability distribution of the dimension that is to be discretized.

Further, methods trying to optimize the discretization of all dimensions simultaneously exist. One approach is to start from a very fine grid, and merge neighboring subintervals for each dimension if the multivariate probabilities of the data within these two consecutive subintervals are similar (Bay 2001; Nguyen et al. 2014). These methods are based on certain choices of similarity metrics, and require explicit specification of the similarity threshold. We empirically show in Section 5.7 that IPD, the method by Nguyen et al. (2014) that is also based on the MDL principle and is considered the state-of-the-art multivariate discretization method, does not converge in practice.

Finally, Kameya extended the one-dimensional MDL-histogram (Kameya 2011) specifically for time series data, who proposed to discretize time series data by iteratively adjusting the cut points on each dimension until convergence, using the coordinate descent optimization approach.

All these multivariate discretization methods try to optimize the adaptive grid and produce (hyper)rectangular regions. Our method, in contrast, is proposed to produce far more flexible segmentation, which allows the binning of one dimension to be dependent on the values of other dimensions.

Density estimation tree. Algorithmically, our method is very similar to methods using tree models for density estimation (Liu and Wong 2014; Ram and Gray 2011; Yang and Wong 2014), as partitioning the data space by iteratively partitioning each dimension is identical to growing a tree. However, these density estimation trees were developed by adapting the scores used in growing, stopping, and pruning (supervised) decision and regression trees. That is, while our algorithm employs a consistent MDL-based framework for selecting the best model, these density estimation trees use separate optimization scores respectively to fit the model and to control the model complexity, often with user-specified hyperparameters and/or computationally expensive cross-validation.

Moreover, these density estimation trees, as is like most supervised tree models, only do binary partitioning in a greedy manner. On the contrary, our method can split a dimension into multiple bins (from 1 to a pre-determined K_{max}) instead of just two, which is not only more flexible, but also more interpretable, as after partitioning on a certain dimension, within each bin the data points on that dimension can be regarded as approximately uniform.

Finally, our method has an additional merging step, which creates much more flexible partitions of data, resulting in models that are more informative for pattern mining and exploratory data analysis.

Supervised discretization. When discretization is needed for a supervised task such as classification, we can use *supervised discretization*, which means that the target variable is used to assess how much information on the target the discretization maintains. Several criteria can be put in this category, which are mostly based on statistical hypothesis testing or entropy, as summarized in the survey paper by Kotsiantis and Kanellopoulos (2006). The MDL principle has also been used for supervised discretization (Fayyad and Irani 1993; Ferrandiz and Boullé 2005; Gupta et al. 2010; Pfahringer 1995; Zhang et al. 2007), but all of them use the so-called *crude MDL* principle (Grünwald 2007), which is theoretically suboptimal.

5.3 Problem Statement

Informally, we consider the problem of inferring the best two-dimensional histogram for a given sample of continuous data. To make this problem precise, we start off by introducing our notation and definitions. Note that all $\log(\cdot)$ should be read as $\log_2(\cdot)$ unless specified otherwise.

5.3.1 Notation and definitions of data, model, and model class

Consider as data a vector of length n, i.e., $x^n = (x_1, ..., x_n)$, sampled independently from a *random variable* X.

The *sample space* of X, denoted as S, is a *bounded* subset of $\mathbb{R}^2$. Although the sample space of a random variable, e.g., a Gaussian, can be infinite in theory, we always assume it to be a bounded "box" when dealing with a given dataset.

The task of estimating S from the data directly is another research topic, usually referred to as "support estimation" in statistical literature (Cuevas, Fraiman, et al. 1997), and hence is out of the scope of our main focus in this article.

Conceptually, a histogram—no matter whether it is one- or multi-dimensional—is a *partition* of the sample space S, denoted by $\widetilde{S}$ and parametrized by a vector $\vec{f} = (f_1, \ldots, f_K)$. A partition $\widetilde{S}$ is defined as a set of *disjoint* subsets of S, and the union of all these subsets is S itself, i.e., $\widetilde{S} = \{S_1, S_2, \ldots, S_K\}$, where $\forall j \in \{1, \ldots, K\}$, $S_j \subseteq S$, $\bigcup_{j=k}^{K} S_j = S$, and $\forall j, k \in \{1, \ldots, K\}$, $S_j \cap S_k = \emptyset$. We also call these subsets, i.e., elements of $\widetilde{S}$, as *regions*.

Next, we assume that the probability density of X, denoted by $f(X)$, is given by

$$f(X) = \sum_{j \in \{1,\ldots,K\}} \mathbb{1}_{S_j}(X) f_j, \tag{5.1}$$

where $\mathbb{1}_{\{\cdot\}}(\cdot)$ is the *indicator function*. Each f_j is a *constant* and $\vec{f}$ satisfies $\sum_{i=1}^{K} f_j |S_j| = 1$, where $|S_j|$ denotes the geometric area of S_j, i.e., when $X \in S_j$, $f(X) = f_j$. We refer to any partition $\widetilde{S}$ as a *histogram model* that contains a family of probability distributions; i.e., $\forall \vec{f} \in \mathbb{R}^K$, we denote a single probability distribution by $\widetilde{S}_{\vec{f}}$.

We denote the model class as $\mathbb{M}$, representing all possible partitions with K regions that can be obtained by clustering cells of a fixed grid covering S, where $K \in \{1, \ldots, K_{max}\}$. The granularity of the grid, denoted as ϵ, and K_{max} are fixed in advance, but note that they can be set arbitrarily small and large, respectively.

Geometrically, this is equivalent to drawing *inner boundaries* within S along the fixed grid. In practice, ϵ can represent the precision up to which the data is recorded or that is useful for the given task. Although the model class we consider only has inner boundaries consisting of *line segments*, we will show that such a model class is flexible enough to approximate curved inner boundaries in Section 5.7.

5.3.2 Histogram model selection by the MDL principle

We now formally define the task of two-dimensional data discretization as an MDL-based model selection task, using histogram models as the model class.

The MDL principle is arguably one of the best off-the-shelf model selec-

tion methods and has been successfully applied to many machine learning tasks (Grünwald 2007; Hansen and Yu 2001). It has solid theoretical foundations in information theory and naturally prevents overfitting as the optimization criterion always includes the model complexity, defined as the code length (in bits) needed to encode that model (Grünwald 2007).

The basic idea is to *losslessly encode* the model and data together, by firstly encoding the model and then compressing the data using that model. The model resulting in the shortest total *code length* is defined to be MDL-optimal, i.e.,

$$\widetilde{S^*} = \arg\min_{\widetilde{S}\in\mathbb{M}} L(x^n, \widetilde{S}) = \arg\min_{\widetilde{S}\in\mathbb{M}} (L(\widetilde{S}) + L(x^n|\widetilde{S})), \tag{5.2}$$

where $L(\widetilde{S})$ and $L(x^n|\widetilde{S})$ are respectively the code length of the model and the code length of the data compressed by that model. Note that $L(\cdot|\cdot)$ denotes the *conditional* code length (Grünwald 2007); informally, $L(A|B)$ represents the code length of the message a *decoder* needs to receive in order to be able to losslessly reconstruct message A after having already received message B.

We will show in Section 5.4 that properly encoding the model and calculating its corresponding code length $L(\widetilde{S})$ turns out to be very difficult. As a result, we unfortunately cannot regard our model selection task simply as an optimization problem.

To alleviate this, we divide the model selection task into two steps, namely 1) partitioning alternately and 2) merging.

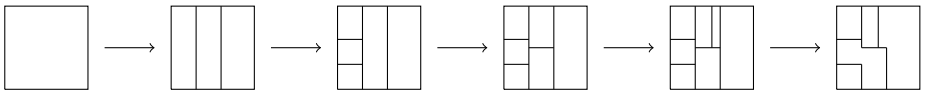

Figure 5.2: An illustration of the partitioning and merging steps. From left to right: alternatively partitioning each region until compression cannot be further improved, and finally merging some of the neighboring regions to further improve compression.

First, we alternately split each region within partition $\widetilde{S}$ (initially $\widetilde{S} = \{S\}$) in one of the two dimensions, then update $\widetilde{S}$ accordingly, and repeat the process. In other words, in each iteration we further split each region within $\widetilde{S}$ in one dimension (i.e., horizontally or vertically), which is equivalent to selecting the best set of horizontal or vertical *cut lines*.

Denote the subset of data points within a certain region $S' \in \widetilde{S}$ as $\{x^n \in S'\}$.

We formally define the task of selecting the set of MDL-optimal cut lines set as

$$\begin{aligned} C^*_{S'} &= \arg\min_{C_{S'} \in \mathbb{C}_{S'}} L(\{x^n \in S'\}, C_{S'}) \\ &= \arg\min_{C_{S'} \in \mathbb{C}_{S'}} (L(C_{S'}) + L(\{x^n \in S'\} | C_{S'})), \end{aligned} \tag{5.3}$$

where $\mathbb{C}_{S'}$ are all possible sets of cut lines, containing $K = \{0, 1, \ldots, K_{max}\}$ cut lines, for the certain region $S' \in \widetilde{S}$ in one certain dimension (i.e., horizontal or vertical), and K_{max} is predetermined a priori to be "large enough" given the task at hand.

In Section 5.5, we will show that searching for the MDL-optimal cut lines for (a subset of) two-dimensional data is the same as searching for the MDL-optimal cut points for the one-dimensional data that is the projection of the two-dimensional data onto the x- or y-axis.

The partitioning step will automatically stop once for each region the MDL-optimal set of cut lines is the null set, i.e., no further partitioning is needed.

Second, we search for all possible clusterings of neighboring regions gained in the previous partitioning step, in a greedy manner. In other words, we consider all possible clustering of regions of the partition gained by the previous partitioning step, which is actually a subset of the full model class $\mathbb{M}$ as defined in Section 5.3.1. We denoted this constrained model class by $\mathbb{M}_c$, and we formally define the merging step as selecting the MDL-optimal model within $\mathbb{M}_c$, i.e.,

$$\widetilde{S}^*_{merge} = \arg\min_{\widetilde{S} \in \mathbb{M}_c} L(x^n, \widetilde{S}) = \arg\min_{\widetilde{S} \in \mathbb{M}_c} (L(\widetilde{S}) + L(x^n | \widetilde{S})). \tag{5.4}$$

Figure 5.2 shows an illustrative example of the partitioning and merging process.

5.4 Calculating the code length

We now discuss the details of the code length (in bits) needed to encode the data and the model.

We first show the calculation of code length of data given a histogram model, encoded by the normalized maximum likelihood (NML) code (Grünwald 2007; Grünwald and Roos 2019). Specifically, we show that the *parametric complexity* term in the code length is independent of data dimensionality, which is an

important observation that makes it feasible to compute the NML code length.

Next, we discuss in detail the difficulties of encoding all possible models $\widetilde{S} \in \mathbb{M}$ if we would want to directly optimize over the full model class $\mathbb{M}$ using Equation (5.2), which motivates our (more pragmatic) solution of dividing the model selection task into two separate steps.

Finally, we discuss the calculation of the code length of a model in the partitioning and merging step respectively, i.e., $L(C_{S'})$ and $L(\widetilde{S})$ of Equations (5.3) and (5.4).

5.4.1 Code length of the data

Extending the work that was previously done for the one-dimensional case (Kontkanen and Myllymäki 2007b), we use the same code—i.e., the *Normalized Maximum Likelihood* (NML) code—to encode the two-dimensional data. This code has the desirable property that it is theoretically optimal because it has minimax regret. The code length of the NML code consists of two terms, namely the maximum likelihood and the parametric complexity (also referred to as *regret*), and is given by

$$L(x^n|\widetilde{S}) = -\log\left(\frac{P(x^n|\widetilde{S}_{\hat{\vec{f}}(x^n)})}{\mathrm{COMP}(n,\widetilde{S})}\right), \tag{5.5}$$

where $P(x^n|\widetilde{S}_{\hat{\vec{f}}(x^n)})$ is the probability of the data given $\widetilde{S}_{\hat{\vec{f}}(x^n)}$, i.e., the parameters $\vec{f} = (f_1, ..., f_K)$ are estimated by the *maximum likelihood estimator* given dataset x^n, denoted as $\hat{\vec{f}}(x^n) = (\hat{f}_1, \ldots, \hat{f}_K)$. The term $\mathrm{COMP}(n, \widetilde{S})$ is the so-called *parametric complexity*, which is defined as

$$\mathrm{COMP}(n,\widetilde{S}) = \sum_{y^n \in S^n} P(y^n|\widetilde{S}_{\hat{\vec{f}}(y^n)}), \tag{5.6}$$

where $\sum_{y^n \in S^n}$ is the sum over *all possible sequences* y^n within the Cartesian product of sample space S that can be *generated* by the histogram model $\widetilde{S}$, i.e., the order of individual values within vector y^n *does* matter.

We will now first describe the calculation of $P(x^n|\widetilde{S}_{\hat{\vec{f}}(x^n)})$, and then the calculation of $\mathrm{COMP}(n, \widetilde{S})$.

For any single data point $x_i \in x^n$, let $x_i = (x_{i1}, x_{i2})$ denote the pair of values

for its two dimensions. We then have

$$P(x^n|\widetilde{S}_{\hat{\hat{f}}(x^n)}) = \prod_{i=1}^{n} P(x_i|\widetilde{S}_{\hat{\hat{f}}(x^n)}) = \prod_{j=1}^{K} \left(\prod_{x_i \in S_j} P(x_i|\widetilde{S}_{\hat{\hat{f}}(x^n)}) \right), \tag{5.7}$$

as the data points are assumed to be independent. Note that K represents the number of regions of $\widetilde{S}$.

Since we assume our data to have precision ϵ, we can define the probability of the data, also referred to as its *maximum likelihood*, as

$$P(x_i|\widetilde{S}_{\hat{\hat{f}}(x^n)}) = P(X \in [x_{i1} - \frac{\epsilon}{2}, x_{i1} + \frac{\epsilon}{2}] \times [x_{i2} - \frac{\epsilon}{2}, x_{i2} + \frac{\epsilon}{2}] \mid \widetilde{S}_{\hat{\hat{f}}(x^n)}) = \hat{f}_j \epsilon^2. \tag{5.8}$$

The maximum likelihood estimator for the histogram model (Scott 2015) is

$$\hat{f}_j = \frac{h_j}{n\,|S_j|}, \forall j, \tag{5.9}$$

where h_j is the number of data points within S_j, and $|S_j|$ is the area of S_j. Thus, following Equations (5.7),(5.8), and (5.9),

$$P(x^n|\widetilde{S}_{\hat{\hat{f}}(x^n)}) = \prod_{j=1}^{K} (\hat{f}_j\,\epsilon^2)^{h_j} = \prod_{j=1}^{K} (\frac{h_j\,\epsilon^2}{n\,|S_j|})^{h_j}. \tag{5.10}$$

Next, we describe the calculation of COMP$(n, \widetilde{S})$. Although it may be surprising at first glance, we show that

Proposition 3. *The parametric complexity COMP$(n, \widetilde{S})$ of a histogram model is a function of sample size n and the number of bins K. Given n and K, COMP$(n, \widetilde{S})$ is independent of the dimensionality of the data.*

We leave the formal proof to Appendix A, but the proposition is based on the following important observations. First, as Kontkanen and Myllymäki (2007b) proved, COMP$(n, \widetilde{S})$ is a function of sample size n and the number of bins K for one-dimensional histograms. The remaining question is whether this holds for two (and higher) dimensional histograms as well. Observe that the maximum likelihood given a two-dimensional histogram model for any data is a function of h_j and $|S_j|/\epsilon^2$, respectively representing the number of data points in each region, and the total number of *possible positions* of data points in each region,

which are both some form of "counts" and hence are "dimensionality free". Finally, $\mathrm{COMP}(n, \widetilde{S})$, as defined in Equation (5.6), is just the sum of maximum likelihoods. Based on these observation, it is trivial to prove that $\mathrm{COMP}(n, \widetilde{S})$ has the same form for one- and multi-dimensional histograms.

Therefore, for both one- and multi-dimensional histogram models, we can denote $\mathrm{COMP}(n, \widetilde{S})$ as $\mathrm{COMP}(n, K)$, and as shown by Kontkanen and Myllymäki (2007b),

$$\mathrm{COMP}(n, K) = \sum_{h_1+\ldots+h_K=n} \frac{n!}{h_1!\ldots h_K!} \prod_{j=1}^{K} (\frac{h_j}{n})^{h_j}, \tag{5.11}$$

which turns out to be the same as the parametric complexity for the multinomial model (Kontkanen and Myllymäki 2007a). We can calculate $\mathrm{COMP}(n, K)$ in linear time (Kontkanen and Myllymäki 2007a) by means of the following recursive formula:

$$\mathrm{COMP}(n, K) = \mathrm{COMP}(n, K-1) + \frac{n}{K-2}\mathrm{COMP}(n, K-2). \tag{5.12}$$

5.4.2 Code length of the model

We first discuss in detail why properly encoding all models in the model class is difficult, and then describe the code length of model in the partitioning step and the merging step respectively.

Encoding all models in the model class is difficult

According to Kraft's inequality, encoding all models in the model class is equivalent to assigning a prior probability distribution to all models (Grünwald 2007). This prior distribution should reflect the model complexities (Grünwald 2004), especially when there exists some hierarchical structure in the model class. For models with similar model complexity, the prior distribution should be non-informative. Particularly, a common practice is to divide the model class into sub-classes according to the hierarchical structure, and then assign the prior distribution to each model by first assigning some prior to all the sub-classes and then assigning a uniform prior to all models within each sub-class.

The model class of all histogram models (i.e., all partitions of S) has an apparent hierarchical structure with respect to model complexity. That is, the model

class could be divided into sub-classes based on a combination of two factors: 1) the number of regions, and 2) the number of line segments composing the inner boundaries. Nevertheless, it is extremely challenging to assign a proper (or even an intuitively "natural") prior distribution based on this complexity hierarchical structure, because of the following two reasons.

First, it is difficult to specify a *joint* prior distribution on the number of regions and the number of line segments, as they are dependent on each other, though specifying marginal prior distributions for each of the factors may be feasible.

Second, given the number of regions, denoted by K, and the number of line segments composing the inner boundaries, denoted by T, it is challenging to count the number of models with K regions and T line segments. Hence, the prior probability of each model (with the uniform prior) within this sub-class is also difficult to obtain. On one hand, there is no analytical formula to obtain such count (to the best of our knowledge). On the other hand, to count this number algorithmically, we would first need to decide how many line segments each region has, i.e., to assign positive integers to $\{T_1, \ldots, T_K\}$ such that $T_1 + \ldots + T_K = T$. The number of possible values of $\{T_1, \ldots, T_K\}$ grows exponentially as K increases. Further, we would need to decide where to put these line segments to form K regions. The number of possible positions is enormous if ϵ is reasonably small. Finally, we would need to go over all individual cases to check for repeated counting for T, since regions can share line segments, which makes the counting computationally infeasible.

Code length of the model in the partitioning and merging steps

As properly encoding all possible models within $\mathbb{M}$ turns out to be too difficult, we now discuss how to calculate the code length of the model separately for the partitioning and merging step.

Partitioning. For a region $S' \in \widetilde{S}$, assume that there are E candidate positions for cut lines, either horizontally or vertically. To encode the set of cut lines, we first encode the number of regions $K \in \{1, \ldots, K_{max}\}$, where K_{max} is predetermined. We assign a uniform prior to K, and thus the code length needed to encode K becomes a constant, which has no effect on the result of the partitioning step. Given K, we then encode the positions of $(K-1)$ cut lines, with again a uniform

prior to all possible sets of $(K-1)$ cut lines. The code length needed in bits is

$$L(C_{S'}) = \log \binom{E}{K-1} \tag{5.13}$$

Merging. Next we discuss the code length of encoding all models in the constrained model class $\mathbb{M}_c$, which contains all possible models that can be obtained by merging neighboring regions of the partition after the partitioning step.

We argue that we should have a non-informative prior on $\mathbb{M}_c$. First, as discussed before, it is challenging to specify a joint prior to both the number of line segments and the number of regions. Second, if neighboring regions are merged, the partition of the sample space tends to have fewer regions but more geometric complexity. Hence, there exists no obvious ways to compare model complexities, even in an intuitive manner.

Thus, we treat the model complexities to be roughly equivalent and we assign a uniform prior to all models in $\mathbb{M}_c$. As a result, the code length of all models within $\mathbb{M}_c$ is a constant and has no effect on the result of the merging step. In other words, we only consider the code length of data in the merging step.

5.5 Revisiting MDL histograms for one-dimensional data

In this section, we elaborate the link of our work to the MDL-based histograms to one-dimensional data.

We first show that searching for the best cut lines on one certain dimension of given two-dimensional data is equivalent to searching for the best cut points for the corresponding one-dimensional data. We then review the algorithm for inferring MDL histograms for one-dimensional data as proposed by Kontkanen and Myllymäki (2007b), and describe how we improve it both theoretically and practically.

Notation and relation to our problem. To be able to distinguish it from two-dimensional data x^n, we denote one-dimensional data as $z^n = (z_1, ..., z_n)$, with precision equal to ϵ. Further, we define the sample space of z^n as $[\min z^n, \max z^n]$.

We define the one-dimensional histogram model with K bins as a set of

cut points, denoted as $C^K = \{C_0 = \min z^n, C_1, ..., C_K = \max z^n\} \subseteq C_a$, with $K \in \{0, 1, \ldots, K_{max}\}$, where K_{max} is pre-determined and C_a is defined as

$$C_a = \{\min z^n, \min z^n + \epsilon, ..., \min z^n + E \cdot \epsilon, \max z^n\}, \tag{5.14}$$

with $E = \lfloor \frac{\max z^n - \min z^n}{\epsilon} \rfloor$. Note that we assume all subintervals to be closed on the left and open on the right, except that the rightmost subinterval is closed on both sides.

The code length needed to encode the model C^K is

$$L(C^K) = \log \binom{E}{K-1}, \tag{5.15}$$

which is the same as Equation (5.13). Further, based on the calculation of maximum likelihood given any histogram model (Section 5.4.1) and Proposition 3, the code length needed to encode z^n given C^K by the NML code is

$$\begin{aligned} L(z^n|C^K) &= -\log P(z^n|C^K) + \log \mathrm{COMP}(n, K) \\ &= -\log \prod_{j=1}^{K} \left(\frac{h_j\, \epsilon}{n\,(C_{j+1} - C_j)}\right)^{h_j} + \log \mathrm{COMP}(n, K). \end{aligned} \tag{5.16}$$

If we compare $L(z^n|C^K)$ and $L(C^K)$ with Equations (5.10) and (5.13), we can see that the definition of the two-dimensional MDL-optimal cut lines and the one-dimensional MDL-optimal cut points only differ by a constant. Thus, given a two-dimensional dataset $x^n = \{(x_{11}, x_{21}), \ldots, (x_{1n}, x_{2n})\}$, the optimization task of searching for the MDL-optimal vertical (or horizontal) cut lines is equivalent to the task of searching for the MDL-optimal one-dimensional cut points based on one-dimensional dataset $z^n = \{x_{11}, \ldots, x_{1n}\}$ (or $z^n = \{x_{21}, \ldots, x_{2n}\}$). That is, z^n is the projection of x^n on the x- or y-axis.

In other words, the algorithm for constructing MDL-based one-dimensional histograms proposed by Kontkanen and Myllymäki (2007b) can be directly applied to the partitioning step of our model selection task. We now briefly review this algorithm and show how we improve it both theoretically and practically.

Improved one-dimensional MDL-based histograms. We improve the one-dimensional algorithm proposed by Kontkanen and Myllymäki (2007b) in two ways. First, in their previous work, the candidate cut points, denoted as C'_a, are

chosen based on the data z^n, i.e., $C'_a = \bigcup_{i=1}^{n} \{z_i \pm \epsilon\}$, and hence the code length of model is calculated *dependent* on given dataset, i.e., $L(C^K|z^n)$ is calculated instead of $L(C^K)$, which is theoretically sub-optimal, because generally

$$L(z^n, C^K) = L(z^n|C^K) + L(C^K) \neq L(z^n|C^K) + L(C^K|z^n). \tag{5.17}$$

In practice, this will cause significantly worse results when the sample size is very small. In such cases, the size of the set C'_a will be very small, and hence the code length of model will be significantly underestimated, leading to serious overfitting. We fix this problem by encoding the model independent of the data, as defined by Equations (5.14) and (5.15).

Further, we show that we do not need to consider *all* candidate cut points within C_a, but just those cut points with a data point near it from left or right, without other cut points in between. That is, we have the following.

Proposition 4. *For any two cut points $C_i, C_k \in C_a$, suppose $C_i < C_k$ and no data points exist in the interval $[C_i, C_k]$, then any cut point $C_j \in [C_i, C_k]$ would not be in the MDL-optimal set of cut points, i.e., we can skip all such C_j during the search process.*

This reduces the search space to a subset of C_a, and hence reduces the computational requirements. We include the proof in Appendix B.

Finally, we simplify the recursion formula for the dynamic programming proposed by Kontkanen and Myllymäki (2007b) in their original paper, which significantly reduces empirical computation time.

Dynamic programming algorithm. Kontkanen and Myllymäki (2007b) derived the recursion formula based on the total code length $L(z^n, C^K)$, i.e.,

$$\begin{aligned} L(z^n, C^K) &= L(z^n|C^K) + L(C^K) \\ &= -\log(P(z^n|C^K) + \log \mathrm{COMP}(n, K) + \log \binom{E}{K-1}. \end{aligned} \tag{5.18}$$

We show that we can simplify the recursion by only including the probability of the data, i.e., $P(z^n|C^K)$, instead of $L(z^n, C^K)$. Observe that when the number of bins K is fixed, $L(C^K)$ and $\mathrm{COMP}(n, K)$ become constant. Then, for fixed K, minimizing $L(z^n, C^K)$ is equivalent to minimizing $\{-\log(P(z^n|C^K)\}$, i.e., maximizing the likelihood.

Therefore, minimizing $L(z^n, C^K)$, for all $K \in \{1, \ldots, K_{max}\}$, can be done in two steps: 1) find the maximum likelihood cut points with fixed K, denoted as $\hat{C}^K$, for each K, using the following dynamic algorithm; and 2) calculate $L(z^n|\hat{C}^K)$ for each K, and find the $\hat{K} \in \{1, \ldots, K_{max}\}$ that minimizes $L(z^n, \hat{C}^K)$. Then,

$$\hat{C}^{\hat{K}} = \arg \min_{K \in \{1,\ldots,K_{max}\}, C^K \in C_a} L(z^n, C^K). \tag{5.19}$$

Now we describe the dynamic programming algorithm for finding $\hat{C}^K$ for each $K \in \{1, \ldots, K_{max}\}$. The (log) probability of z^n given any cut points is

$$\begin{aligned}
\log P(z^n|C^K) &= \sum_{i=1}^{n} \log P(z_i|C^K) \\
&= \sum_{j=1}^{K} \sum_{z_i \in [C_{j-1}, C_j)} \log P(z_i|C^K) \\
&= \sum_{j=1}^{K-1} \sum_{z_i \in [C_{j-1}, C_j)} \log P(z_i|\{C^K \setminus C_K\}) + \sum_{z_i \in [C_{K-1}, C_K]} \log P(z_i|C_K) \\
&= \log P(z^n_{C_{K-1}}|\{C^K \setminus C_K\}) + \sum_{z_i \in [C_{K-1}, C_K]} \log P(z_i|C_K)
\end{aligned} \tag{5.20}$$

where $z^n_{C_K}$ is a constrained dataset containing all data points smaller than C_K, i.e.,

$$z^n_{C_{K-1}} = \{z \in z^n | z < C_{K-1}\}. \tag{5.21}$$

Given the previous, the recursion formula is given by

$$\begin{aligned}
\max_{C^K \subseteq C_a} \log P(z^n|C^K) = \max_{C_K \in C_a} [&\max_{\{C^K \setminus C_K\} \subseteq C_a} \log P(z^n_{C_{K-1}}|\{C^K \setminus C_K\}) \\
&+ \sum_{z_i \in [C_{K-1}, C_K]} \log P(z_i|C_K)]
\end{aligned} \tag{5.22}$$

and hence a dynamic programming algorithm can be applied to search all $K \in \{1, \ldots, K_{max}\}$. In practice, K_{max} is pre-determined, and larger K_{max} should be investigated if $\hat{K} = K_{max}$.

The disadvantage of implementing the dynamic programming algorithm based on $L(z^n, C^K)$, $\forall K \in \{1, \ldots, K_{max}\}$, is that we would need to calculate the para-

metric complexity COMP(·) for every constrained dataset. Our improved version, in contrast, involves only $P(z^n|C^K)$, and thus we only need to calculate COMP(·) for the full dataset z^n when calculating $L(z^n, \hat{C}^K)$ for each K, which will be much faster in practice.

The essential component of the dynamic programming algorithm is to construct the constrained dataset $z^n_{C_{K-1}}$, $\forall K \in \{1, \ldots, K_{max}\}$. These constrained datasets are easy to construct in the one-dimensional case with a natural order, but infeasible for two or higher dimensional cases. Hence we resort to the heuristic algorithm presented in the next section.

5.6 The PALM Algorithm for Partitioning and Merging

We propose a heuristic algorithm named PALM, which infers histogram models for two-dimensional data by decomposing the overall model selection problem into two steps: 1) partition space S alternately based on the discretization result from previous iterations until it stops automatically; and then 2) merge neighboring regions if their densities are very similar. Both steps use the MDL principle as the decision criterion, with the code length defined in Section 5.4.

The PALM algorithm is given in Algorithm 5. Specifically, we first initiate $\widetilde{S} = \{S\}$ and choose the starting direction (line 1); then we iterate over all regions in $\widetilde{S}$ and partition each of them by searching for the MDL-optimal cut lines in the chosen direction (lines 3–5), and update $\widetilde{S}$ accordingly (lines 8–10); then, we keep iterating until $\widetilde{S}$ is no longer updated (lines 2 and 6–7), which completes the partitioning step.

Next, the merging step searches, in a greedy manner, for the MDL-optimal partition of S over all possible partitions that can be obtained by merging any two neighboring regions of the partition that is obtained in the partitioning step. That is, we list all the neighboring pairs of regions in $\widetilde{S}$, i.e., two regions that share part of their boundaries (line 15); then, we merge the pair that compresses the data most (or equivalently, decreases the MDL score most) and update the neighboring pairs list (lines 21–23); finally, we stop the merging step when no better compression can be obtained by merging any neighboring two pairs in $\widetilde{S}$ (lines 19–20).

Algorithm complexity. We now discuss the worst-case algorithm complexity for the partitioning and merging step respectively, and we will show the empirical runtime in Section 5.7.6.

For the first iteration of the partitioning step (i.e., when $\widetilde{S} = \{S\}$), the algorithm has a complexity of $\mathcal{O}(K_{max}E^2)$, the same as the one-dimensional case (Kontkanen and Myllymäki 2007b), where E is the number of possible locations for vertical (or horizontal) lines within the whole sample space S, based on the fixed grid with granularity ϵ. The second iteration has a worst-case time complexity of $\mathcal{O}(K_{max}^2E^2)$ when the first iteration produces exactly K_{max} regions. Following this line, the worst-case time complexity of the partitioning step is $\mathcal{O}(K_{max}^IE^2)$, where I is the number of iterations.

As for the merging step, the time complexity is bounded by K_pK_0, where K_0 denotes the number of regions after the partitioning step, and K_p denotes the number of neighboring pairs. That is, we can merge at most $(K_p - 1)$ times, and each merging requires going over all the neighboring pairs.

Although the worst-case time cost for the partitioning step is exponential, and K_0 and K_p could be large in practice, we will show in Section 5.7.6 that the empirical runtime may scale much better than exponential growth.

Choosing the hyper-parameter settings. We now briefly discuss how to choose ϵ and K_{max} in practice. First, we should set ϵ to be the same as the precision of the data by default; data is always recorded up to a precision in practice. Further, when prior knowledge exists given a specific task, ϵ may be larger than the recording precision, because the domain expert or data analyst may decide that the data is only meaningful up to a more coarse precision.

Second, theoretically we should set K_{max} to be sufficiently large, and hence in practice K_{max} is a "budget" rather than a hyper-parameter like the threshold or stopping criterion in other discretization methods (e.g., Nguyen 2014, Kerber 1992). That is, unlike these hyper-parameters, which can be either too large or too small and hence need to be carefully tuned, K_{max} can be simply picked to be as large as possible.

This makes our method practically hyper-parameter-free, in the sense that—given the guidelines above—no tedious hyper-parameter tuning should be necessary to obtain the best possible results.

5.7 Experiments

In this section, we investigate the performance of PALM using synthetic data, after which we will apply it to real-world data in the next section. We show that PALM can construct two-dimensional histograms that are adaptive to both local densities and sample size of the data.

We start off by defining the "loss" that we use for quantifying the quality of the "learned" partitions. We then present experiment results on a wide variety of synthetic data. Although our algorithm relies on heuristics, we show that it has a number of desirable properties, as follows.

First, if the data is generated by a histogram model within our model class $\mathbb{M}$, PALM is able to identify the "true" histogram given a large enough sample size. The results are discussed in Section 5.7.2.

Second, in Section 5.7.3 we show that PALM has the flexibility to approximate histogram models outside the model class $\mathbb{M}$. Specifically, we study the behavior of PALM on a dataset generated as follows: we set the sample space $S = [0, 1] \times [0, 1]$, and partition it by a sine curve; we then generate data points uniformly distributed above and below the sine curve, with different densities.

Third, we study the performance of PALM on data generated by two dimensional Gaussian distributions in Section 5.7.4. We show that it inherits the property of the one-dimensional MDL histogram method (Kontkanen and Myllymäki 2007b) that the bin sizes of the histogram are self-adaptive: the two-dimensional bin sizes become smaller locally where the probability density changes more rapidly.

Fourth, in Section 5.7.5 we compare PALM with the IPD algorithm (Nguyen et al. 2014), using a simple synthetic dataset that is almost identical to what has been used to study the performance of IPD (Nguyen et al. 2014).

Note that we always set $\epsilon = 0.001$, and all simulations are repeated 500 times unless specified otherwise[4]. The initial partitioning direction is fixed as vertical, to make the visualizations of the inferred partitions comparable.

[4]The code is available at: https://github.com/ylincen/PALM

5.7.1 Measuring the difference between two-dimensional histograms

As PALM produces a histogram model and can be regarded as a density estimation method, one of the most intuitive "loss" functions is the *Mean Integrated Squared Error (MISE)* (Scott 2015), defined as

$$\mathrm{MISE}(\hat{f}) = \mathbb{E}[\int_S (f(x) - \hat{f}(x))^2 dx], \tag{5.23}$$

where f is the true probability density and $\hat{f}$ is the histogram model density estimator. We report the empirical MISE by calculating the integral numerically, and estimating $\mathbb{E}[\cdot]$ by the empirical mean of results over all repetitions of the simulation.

As MISE cannot indicate whether there are more "bins" than necessary, we also propose two "loss" functions that directly quantify the distances between the inner boundaries of the learned and true partitions of a sample space S. We first break up the line segments of the inner boundaries into *pixels* with a precision set to $0.01 = 10\epsilon$ (merely to speed up the calculation). Then we introduce two loss functions based on the idea of *Hausdorff distance*, considering *false positives* and *false negatives* respectively. Namely, we propose L_{learn}, based on the learned partition, and L_{true}, based on the true partition:

$$L_{\text{learn}} = \sum_{p \in P} min_{q \in Q} ||p - q||^2; L_{\text{true}} = \sum_{q \in Q} min_{p \in P} ||p - q||^2 \tag{5.24}$$

where $||\cdot||$ denotes the Euclidean distance and P and Q are the sets of *pixels* on the line segments of the learned partition and the true partition, respectively.

The intuition for L_{learn} is that, for a given pixel on a line segment of the learned partition, we find on the line segments of the true partition the pixel closest to it, and measure their distance; for L_{true} it is the other way around. Thus, if L_{learn} is large, the learned partition must have unnecessary extra line segments, whereas if L_{true} is large, the learned partition fails to identify part of the line segments that actually exist.

5.7.2 Revealing ground truth two-dimensional histograms

We describe the settings for simulating the data and then our experiment results, to empirically show that our algorithm can identify the "true" histogram model if the data is generated by it.

Experiment settings. To randomly generate the "true" partitions, we use a generative process that is very similar to the search process of our algorithm: we fix a rectangular region, $S = [0, 1] \times [0, 1]$, randomly generate vertical cut lines to split it into K_1 regions, and randomly generate horizontal cut lines to split each of the K_1 regions into $(K_{21}, ..., K_{2,K_1})$ regions respectively. Then, for each pair of neighboring regions, we merge them with a pre-determined probability P_{merge}.

We set these hyper-parameters as follows:

$$K_1 = K_{21} = K_{22} = ... = K_{2,K_1} = 5; P_{merge} = 0.4; \epsilon = 0.001. \tag{5.25}$$

With these hyper-parameters, our generative process is able to generate a diverse subset of $\mathbb{M}$, as P_{merge} is chosen delicately to be not too small or too large. Figure 5.3 shows four random examples of the true partitions and learned partitions. These learned partitions are produced with the sample size set as 10 000.

After the partition is fixed, we generate "true" density parameters for the histogram model using a uniform distribution, i.e.,

$$f_j \sim \text{Uniform}(0, 1), \forall i = 1, 2, ..., K; \tag{5.26}$$

and normalize them such that $\sum_{j=1}^{K} f_j |S_j| = 1$, where K is the number of regions in total and $|S_j|$ is the geometric area of S_j. Note that we do not force the f_j to be different from each other.

Results. Figure 5.4 shows that MISE is already small for small sample size, and converges to almost 0 as the sample size increases. We also show, in Figure 5.5, that L_{learn} and L_{true} converge to almost zero except for some outliers.

The outliers of L_{learn} are due to sampling variance when generating data points, the number of which decreases significantly as the sample size grows.

The outliers of L_{true}, however, are due to the random generation of the density parameters f_j. As we do not force all f_j's to be different, they could accidentally turn out to be very similar. In that case, some of the "true" inner boundaries are

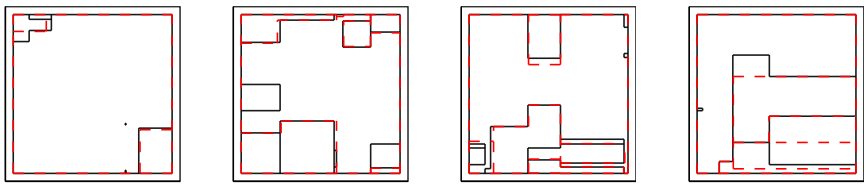

Figure 5.3: Random examples of true (black solid) and learned partitions (red dashed) of the experiment in Section 5.7.2, mainly to show that our experiment settings can produce very flexible partitions of $[0, 1] \times [0, 1]$. Note that the sample size is set as 10 000, which is *not* enough for MISE (Equation 5.23) to converge to almost 0, but the learned partitions by PALM already look promising: it can partly identify the true partitions.

actually unnecessary, and our algorithm will "fail" to discover them. Table 5.1 confirms that this is the cause of outliers when the sample size is large ($\geq 1e5$): when PALM fails to identify part of the "true" inner boundaries and $L_{true} > 1$, the learned histogram still estimates the density very accurately. The only explanation is then that some regions of the true partition accidentally have very similar f_j's.

Moreover, when the sample size is moderate, e.g., 5000, L_{learn} is already small, meaning that PALM can partly identify the true partition quite precisely, and rarely produces unnecessary extra regions. As the sample size increases, L_{true} decreases, indicating that the learned partition becomes more and more complex; i.e., it is shown that the model selection process is self-adaptive to sample size.

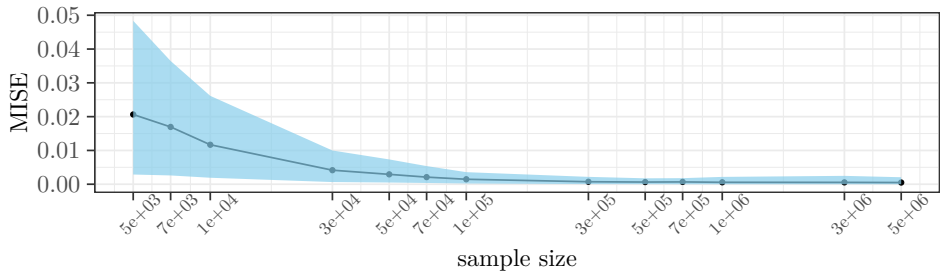

Figure 5.4: Sample size vs MISE: MISE converges to almost 0 when the sample size becomes larger than 100 000. The range between the 5th and 95th percentiles is shown in blue.

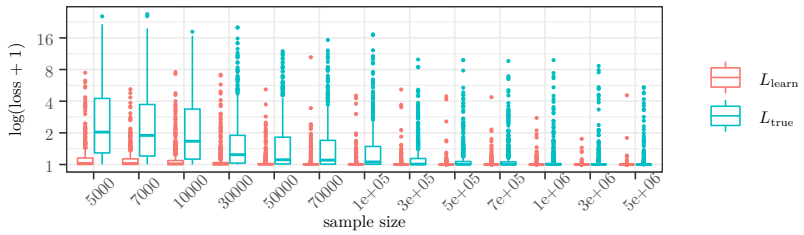

Figure 5.5: Boxplots showing the sample size versus L_{learn} and L_{true} as defined in Equation (5.24). Note that the y-axis has a logarithmic scale. L_{learn} is generally much smaller than L_{true}, meaning that it is very rare that PALM produces unnecessary extra regions. When the sample size is large enough for MISE to converge ($n \geq 1e5$), outliers of L_{true} are due to sampling variance when generating the true parameters f_j defined in Equation (5.26), see Table 5.1; the number of outliers for L_{learn} decreases rapidly as the sample size becomes larger, as they are due to sampling variance when generating the data.

5.7.3 Approximating histogram models outside model class $\mathbb{M}$

We now investigate the case where the true model is not within model class $\mathbb{M}$, while the data is still generated uniformly within each region.

We show that, although the model class $\mathbb{M}$ is based on a grid, it is indeed flexible and expressive: in practice, the learned partitions can approximate true partitions outside $\mathbb{M}$, and the approximation becomes more and more accurate as the sample size increases.

Experiment settings. As an illustrative example, we partition $S = [0,1] \times [0,1]$ by several sine curves, defined as

$$g(x) = \frac{1}{4} \sin 2m\pi x + \frac{1}{2} \tag{5.27}$$

and where m is a hyper-parameter.

We randomly generate data from a uniform distribution above and under the sine curve, and we set the probability density above $g(x)$ to be twice as large as below $g(x)$, i.e., we uniformly sample $\frac{2}{3}n$ data points above $g(x)$, and $\frac{1}{3}n$ data points below $g(x)$, where n is the total sample size.

Results. We empirically show that the learned partitions approximate the sine curves quite precisely, though occasionally a few extra undesired regions are produced. Figure 5.6 (left) shows the learned partitions on single simulated datasets, with $m \in \{2, 4, 6\}$ to control the degree of oscillation, and sample size $n \in \{1e4, 1e5, 1e6\}$. We see that, as the sample size grows, our approximation

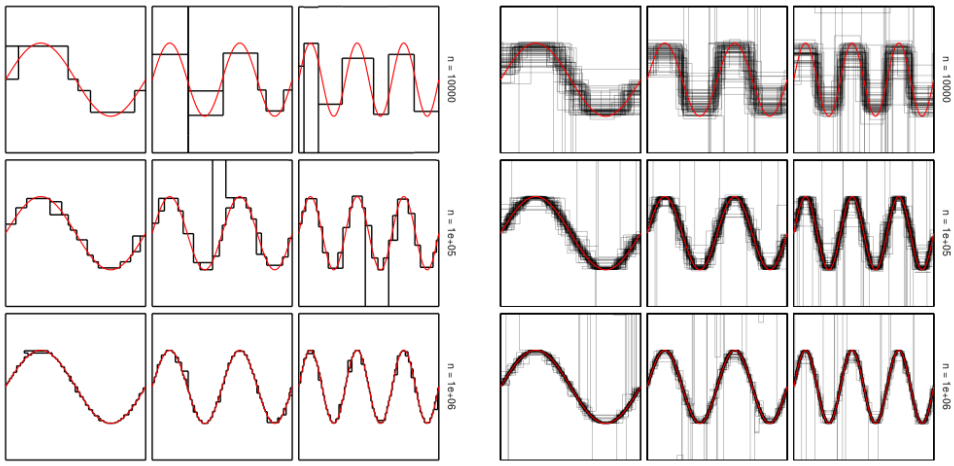

Figure 5.6: (Left) Sine curve defined in Equation (5.27) (red), with $m \in \{2, 4, 6\}$ from left to right on each row, and the learned partition by PALM (black). Data is randomly generated by uniforms distribution above and below the sine curve, within $S = [0, 1] \times [0, 1]$. Densities above and below the since curve are 2:1. From top to bottom, the sample sizes of the simulated data are $n \in \{1e4, 1e5, 1e6\}$. (Right) 50 partition results of 50 different simulated datasets are plotted *together*. It shows that PALM is not guaranteed to be absolutely stable, as it occasionally produces undesired extra line segments, but the line segments of the learned partitions mostly gather around the true sine curve.

becomes more and more accurate.

However, since our algorithm is greedy in nature, there is no guarantee to find the partition with the global minimum score. In practice, PALM will occasionally produce undesired, extra line segments. Thus, to investigate the stability of the learned partitions, we repeat the simulation 50 times for each combination of m and n, and plot *all* partition results in one single plot in Figure 5.6 (right).

Figure 5.6 (right) shows that the undesired extra regions are produced more frequently as m increases, but seems independent of sample size n. However, as sample size increases, the learned partitions become indeed more stable as they gather around the sine curves more closely.

5.7.4 Gaussian random variables

In this section, we show the performance of our algorithm on data generated from a two-dimensional Gaussian distribution. Specifically, we consider two of them, i.e., $N[\binom{0}{0}, \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)]$ and $N[\binom{0}{0}, \left(\begin{smallmatrix}1 & 0.5\\ 0.5 & 1\end{smallmatrix}\right)]$, of which the key difference is whether

the two dimensions are independent. We assume $S = [-5, 5] \times [-5, 5]$, as the true Gaussian density outside such S is negligible.

Figure 5.8 shows the learned partitions as well as the learned empirical densities from a random simulated dataset with different sample sizes, $n \in \{5\,000, 10\,000, 50\,000\}$. Note that bin size is self-adaptive with regard to sample size and local structure of the probability density. We also mention that the empirical runtime for a single dataset generated by such Gaussian distributions is at most a few minutes, for all $n \leq 50\,000$.

To quantify the quality of the learned partitions by PALM, we compare the MISE of PALM to the MISE of fixed equally-spaced grid partitions with different granularities. Figure 5.7 shows the mean and standard deviation of MISE for different cases, and we conclude that, to achieve roughly the same level of MISE with a fixed grid, a fixed grid needs to have five times as many regions as a partition learned by PALM.

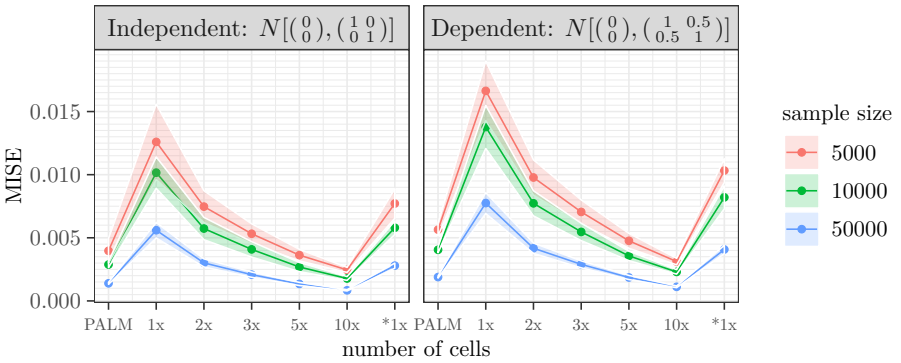

Figure 5.7: For data generated from a two-dimensional Gaussian distribution, described in Section 5.7.4, the mean and standard deviation of MISE is calculated for different partitions: (from left to right) PALM, fixed grid with the same number of regions as PALM (denoted as '1x'), fixed grid with two times number of regions as PALM (denoted as '2x'), ..., and fixed grid with the same number of regions before the merging step of PALM (denoted as '*1x'). We assume $S = [-5, 5] \times [-5, 5]$, as the true Gaussian density outside S is negligible.

5.7.5 Comparison with IPD

Since—to the best of our knowledge—no existing discretization method can produce partitions as expressive as PALM, it seems not so meaningful to compare

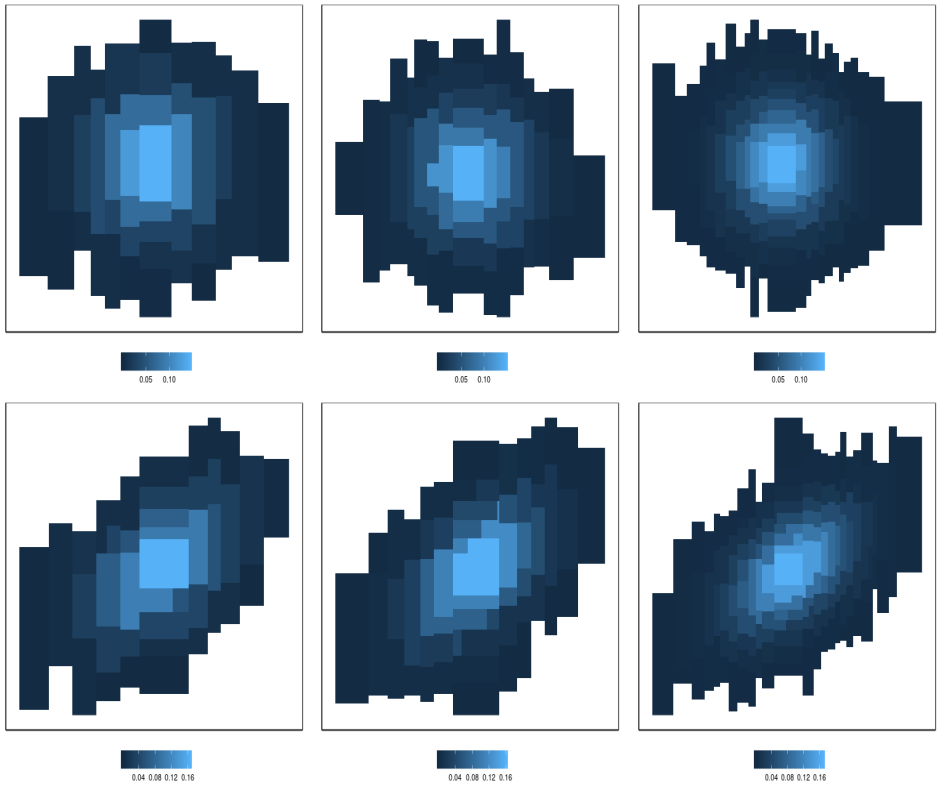

Figure 5.8: Learned partitions and estimated densities by PALM. The data is generated from two-dimensional Gaussian distributions, with sample size $n \in \{5\,000, 10\,000, 50\,000\}$, from left to right. The top and bottom row is respectively generated from independent and dependent two-dimensional Gaussian distributions.

with any existing algorithm. However, we do include a comparison with the IPD algorithm (Nguyen et al. 2014), mainly to show that our algorithm not only can produce more flexible partitions by definition, but also beats this state-of-the-art algorithm on a "simple" task, i.e., when the "true" partition is an adaptive two-dimensional grid.

We use simple synthetic data, similar to one of the synthetic datasets used to study the performance of IPD (Nguyen et al. 2014). The data is generated to be uniform within four regions in $S = [0, 1] \times [0, 1]$. These regions are produced by partitioning S by one vertical line $x = V_x$ and one horizontal line $y = H_y$, where $V_x, H_y \sim \text{Uniform}(0, 1)$. The number of data points within each region is equal.

We compare the loss, as defined in Equation (5.24), and we show in Figure 5.9 that 1) PALM has better performance on small datasets, and 2) as the sample size gets larger, PALM converges but IPD partitions S into more and more regions, as can be witnessed from an increasing L_{true}.

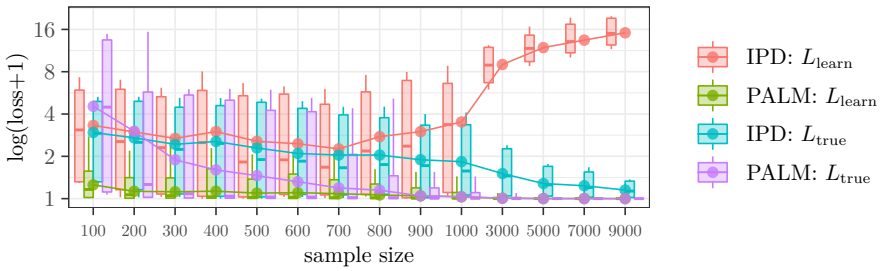

Figure 5.9: Comparison of PALM and IPD, using the box-plot and the mean of L_{learn} and L_{true}, as defined in Equation (5.24). PALM not only performs better when the sample size is small, but also converges as the sample size increases, while IPD does not converge.

5.7.6 Empirical runtime

We next discuss the empirical runtime with respect to K_{max}, the maximum number of bins to search, and E, the number of candidate cut points.

Specifically, we use two-dimensional datasets simulated from independent standard Gaussian distributions to examine the relationship between K_{max} and runtime, with fixed sample size equal to 500 and $\epsilon = 0.001$. The results are illustrated in Figure 5.10, showing that the runtime increases linearly with K_{max}. Further, to investigate the relationship between E and the runtime, we again simulate from two-dimensional Gaussian distributions with different variance σ^2 to control the E [5]. We fixed the sample size to be 1 000 and $\epsilon = 0.001$. The results show that, the runtime grows quadratically with E (as shown by the blue dashed curve), but the second-order coefficient is quite small (as it is very close to the red dashed line with a linear trend). We report the runtime based on 500 repetitions.

[5]For reproducibility, we first simulate 10 000 data points from $N[\binom{0}{0}, \begin{pmatrix} 0 & \sigma^2 \\ \sigma^2 & 0 \end{pmatrix}]$, where $\sigma^2 = E\epsilon/2$, where E is the desired number of candidate cut points. Since the corresponding desired data range with E candidate cut points is $[-E\epsilon/2, E\epsilon/2]$, we next remove the data points outside this desired data range, and we finally randomly select 1 000 data points from the remaining data points.

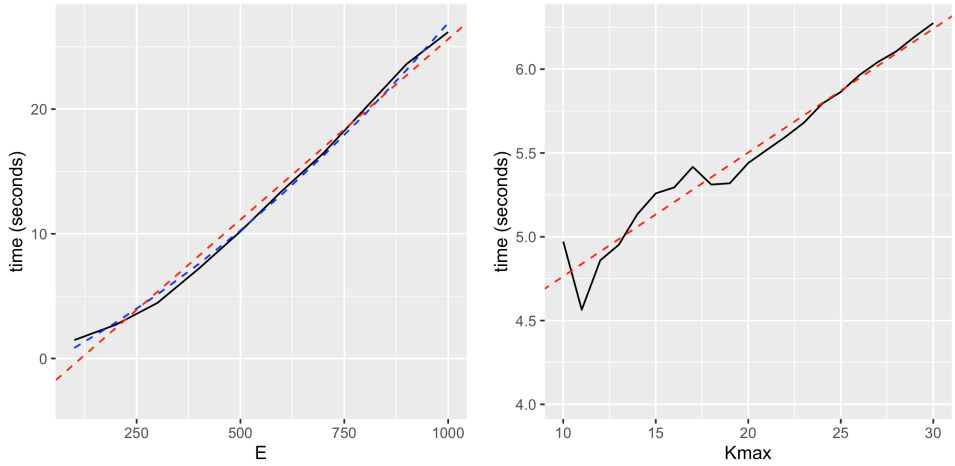

Figure 5.10: Empirical time complexity on simulated two-dimensional Gaussian data, with respect to E, the number of candidate cut points, and the runtime, and K_{max}, the maximum number of bins we search.

5.8 Case study

We now show the results of applying our algorithm to real-world spatial datasets. We start with describing the three datasets we use in Section 5.8.1. Next, we describe our case study tasks in Section 5.8.2. Specifically, we inspect the results by visualizing the histograms, to illustrate that our method can be used as an explanatory data analysis (EDA) tool. We also compare with kernel density estimation (KDE), arguably the most widely used EDA method for spatial datasets, both for the visualizations and the goodness-of-fit on unseen data. In Section 5.8.3, we report our results and show that 1) PALM can produce partitions that characterize more detailed density changes than KDE, and 2) PALM fits better on unseen data (i.e., a test dataset), in the sense that the partition of PALM has larger log-likelihood on the test dataset than KDE. Finally, we report the runtimes and detailed algorithm settings, respectively in Section 5.8.4 and 5.8.5.

5.8.1 Datasets

We consider three diverse real world datasets: locations of Airbnb housing in Amsterdam[6], GPS locations of destinations of DiDi taxi queries in Chengdu, China[7], and GPS recordings of visitors' movement in an amusement park[8].

Visitors movement data in the DinoFun amusement park. All visitors at the amusement park need to carry a device or use a smartphone app to check in at different attractions (e.g., roller coasters). Further, the amusement park is segmented into 100×100 cells (all of them are roughly 5 meters $\times$ 5 meters), and each cell has a sensor which can track the position of each visitor. The device (or the mobile app), together with the sensors, checks the position of the visitor every few seconds and records the position *if the visitor moves to another cell*. Thus, applying PALM on this dataset will reveal the densities of places that people have been in the amusement park. This data has a sample size of 9 078 623, in which every row represents a single position that one individual visitor visited (or passed by), with information like visitor's ID, timestamp, and location.

Amsterdam airbnb locations. This data has a sample size of 20 244, and the location of each house is recorded by its longitude and latitude. Applying PALM on this dataset shows the distribution of Airbnb housing in Amsterdam.

DiDi taxi data in Chengdu. The sample size of the data is 107 573, which collects the longitude and latitude of taxi destinations. Applying PALM on this dataset shows the densities of different regions that people visited by taxi in Chengdu, China.

5.8.2 Case study tasks

Explanatory data analysis and visualizations. We first partition the three two-dimensional datasets by PALM and estimate the densities of all regions, using the full datasets. We visualize the densities by the heat maps in Figure 5.11, and compare the visualizations obtained by two-dimensional kernel density estimation (KDE) (Duong et al. 2007), also with the full dataset. We also include the visualization results of the discretization obtained by IPD (Nguyen et al. 2014) for

[6] http://insideairbnb.com

[7] https://gaia.didichuxing.com

[8] http://vacommunity.org/VAST+Challenge+2015

comparison, although IPD is not primarily designed for two-dimensional datasets. The background of Figure 5.11 are the map of the DinoFun amusement part (provided together with the dataset), and the map of Amsterdam and Chengdu (from Google Maps API and the R package "ggmap" (Kahle and Wickham 2013)).

Comparison of KDE and PALM on the goodness-of-fit. Further, to quantitatively compare how KDE and PALM fit unseen data, and thus generalize to the underlying data distribution, we randomly split each dataset into training and testing set, obtain the PALM and KDE result from the training set, and compare the log-likelihoods on the testing dataset. We repeat the random splitting 100 times[9].

5.8.3 Case study results

We first analyze the result on each dataset respectively, based on which we give our concluding remarks for the case study at the end of this section.

Visitors movement data in the DinoFun amusement park. As shown in Figure 5.11, both KDE and PALM reveal the walking path of the amusement park purely from the movement data (i.e., without knowing the map as additional information). Although KDE seems to capture more density changes, we show that it fits unseen data much worse than PALM, measured by the log-likelihood on the test dataset, shown in Table 5.2. Thus, we conclude that KDE may overfit on this dataset.

Amsterdam airbnb locations. The visualizations of PALM and KDE look generally similar: if we treat red and orange regions in the center as the "dense region", the rigid boundary between the dense region and the rest obtained by PALM approximates well the corresponding curve boundary obtained by KDE. However, note that more density changes are captured within the dense region, and PALM revealed two dense spots outside the central areas that KDE neglects, respectively on the top right and the bottom right of the map[10]. Further, as shown in Table 5.2, the (average) log-likelihood of PALM and KDE on the test set is

[9] To speed up the process, we randomly sampled a subset of the Chengdu taxi dataset that contains 10% of the full sample size; also, for the amusement park movement dataset, we only use the subset of the data that is between 4 hours and 5 hours after the opening of the park, with sample size 713 846. Note that this is only for the comparison of goodness-of-fit but not for the visualizations and empirical runtime evaluation

[10] The top right dense spot is close to the "AMSTERDAM NOORD" text on the map (on the "T"), and bottom right dense spot is near "Amstel Business Park".

almost the same, which indicates that PALM does not overfit on this dataset, i.e., the dense spots revealed by PALM are valid.

DiDi taxi data in Chengdu. The visualizations of PALM and KDE lead to different understandings of this dataset: while KDE reveals several hot clusters of taxi destinations, PALM shows that the density can change drastically within very small range of areas. As PALM fits better on the testing dataset, we conclude that PALM does not overfit but KDE may over-smooth this dataset.

By default the PALM algorithm always starts by splitting the x-axis. Starting by splitting the y-axis leads to slightly different models, and thus somewhat different visualizations, but those differences are so minimal that they can be ignored for practical purposes. That is, the differences mostly appear in sparse regions, with very low densities, where no interesting patterns occur. To demonstrate that the differences are negligible, we compare the log-likelihoods obtained on unseen data when starting splitting on either the x-axis or the y-axis. The log-likelihoods are indeed very similar for both starting directions, as shown in Table 5.2, confirming that the resulting histogram models can only be different in sparse and less important regions; otherwise the log-likelihoods would be substantially different.

Based on the analysis above, we conclude that 1) although PALM partitions the dataset with rigid boundaries, PALM fits the data better than KDE when the datasets have drastic local density changes, such as the Chengdu taxi dataset and the amusement park dataset; 2) when we have smoother two-dimensional data such as the Amsterdam housing dataset, PALM and KDE fit the data equally well; 3) when we look at the visualizations, PALM tends to capture more density changes than KDE, and PALM can reveal dense spots that KDE neglects; in other words, KDE tends to over-smooth the dataset.

Last, we include the visualizations of the IPD discretization in Appendix C, in which we demonstrate that the discretization results obtained by IPD have much coarser granularity. Hence, our discretization results preserve more information from the original datasets.

5.8.4 Empirical runtime

We examine the empirical runtime on these three datasets in Table 5.3 (using the full datasets, without the split of training and testing set). We conclude that

KDE is generally much faster, except on the amusement park dataset, which has a very large sample size but small E.

Note that the runtime of KDE highly depends on the number of evaluation points, the bandwidth selection methods, and whether to use the binned kernel estimation as an approximation to the exact kernel estimation for bandwidth selection and/or density estimation. The runtime we report here is based on the following settings: 1) the number of evaluation points is the same as the number of pixels evaluated by PALM, i.e., the pixels on the fixed grid with the granularity ϵ; 2) the binned approximation for the plug-in bandwidth selection is used; otherwise it becomes too slow[11]; 3) the binned approximation for the density estimation is not used. Note that we use these same settings not only for the runtime evaluation, but also for visualizations and calculating the log-likelihood on the testing datasets.

5.8.5 Algorithm settings

We now describe some additional algorithm settings for reproducibility for PALM and KDE.

Kernel density estimation (KDE). We choose the Gaussian kernel for KDE, the most commonly used kernel by default. We also experiment with several bandwidth selection methods, including both plug-in methods and cross-validation methods. We find that plug-in methods are generally both more stable and much faster in these three cases, and we choose the one that is specifically designed for two-dimensional cases (Duong and Hazelton 2003).

Also, we visualize the KDE results by directly plotting the density of each "pixel"; another common practice is to use a contour function, which will further smooth the KDE results and hence hamper the straightforward comparisons with the PALM results.

PALM. We set $\epsilon = 1$ and $K_{max} = 100$ for the amusement park dataset, as the amusement park is divided into a 100×100 grid, so the data is recorded at precision of 1 and the maximum number of bins cannot exceed 100. For the other two datasets, the precision of the dataset is set as $\epsilon = 0.001$, which is roughly

[11] It cost more than 10 minutes for the Amsterdam housing data, and more than two days for the amusement park data, both on the full dataset (no splitting for training and testing set).

100 meters. During the partitioning step, we set $K_{max} = 300$ to make sure that $\hat{K} < K_{max}$.

5.9 Conclusions

We proposed to discretize two-dimensional data by histograms with far more flexible partitions than adaptive grids, as we observed that the appropriate binning of one dimension may depend on the value of the other dimension.

Next, we formalized this task based on the MDL principle. Building upon the one-dimensional MDL histogram, we made several technical contributions so as to extend both the formulation and algorithm to the two-dimensional case. Specifically, we solved the problem of calculating the parametric complexity for multi-dimensional cases. Also, we revisited and improved the algorithm for one-dimensional dataset by 1) correcting a minor flaw related to the model encoding, and 2) simplifying the dynamic programming recursion and hence improving the time complexity.

Further, we proposed a novel heuristic algorithm PALM, which combines the top-down and bottom-up search strategies, and we extensively examined the performance of the PALM algorithm on both synthetic and real-world datasets. That is, we verified our algorithm on various synthetic datasets, and showed that: 1) PALM reveals the ground-truth histogram and converges, in contrast to IPD that produces more and more bins as sample size increases; 2) PALM approximates well to the partitions outside the model class; 3) PALM is self-adaptive to local density structures and sample sizes.

Finally, we applied our algorithm on three diverse real-world spatial datasets, and demonstrated that PALM not only captures more densities changes than KDE, but also fits the unseen data better than KDE, as measured by the log-likelihood.

5.10 Appendix A: Proof of Proposition 3

Proposition 3: *The parametric complexity* $COMP(n, \widetilde{S})$ *of a histogram model is a function of sample size* n *and the number of bins* K*. Given* n *and* K*,* $COMP(n, \widetilde{S})$ *is independent of the dimensionality of the data.*

Assume $S \subset \mathbb{R}^l$, $\widetilde{S}$ is any partition of S with K regions, and $\forall S_j \in \widetilde{S}$, $|S_j|$ represents the (hyper-)volume of S_j; for any y^n that can be generated by $\widetilde{S}$, $h_j(y^n)$ denotes the number of data points in region S_j.

$$\begin{aligned}
\mathrm{COMP}(n, \widetilde{S}) &= \sum_{y^n \in S^n} P(y^n | \widetilde{S}_{\vec{f}=\hat{\vec{f}}(y^n)}) \\
&= \sum_{y^n \in S^n} [\prod_{j=1}^{K} (\frac{h_j(y^n)\,\epsilon^l}{n\,|S_j|})^{h_j}] \\
&= \sum_{h_1+\ldots+h_K=n, h_j \geq 0, \forall j} \quad \sum_{\{y^n : h_j(y^n)=h_j, \forall j\}} [\prod_{j=1}^{K} (\frac{h_j(y^n)\,\epsilon^l}{n\,|S_j|})^{h_j}]
\end{aligned} \tag{5.28}$$

To count the elements in the set $\{y^n : h_j(y^n) = h_j, \forall j\}$, we observe that the number of possible ways of distributing $(h_1, ..., h_K)$ data points into each region of $\widetilde{S}$ respectively is

$$\binom{n}{h_1}\binom{n-h_1}{h_2}\cdots\binom{n-h_1-\ldots-h_{K-1}}{h_K} = \frac{n!}{h_1!...h_K!}. \tag{5.29}$$

As we assume the precision to be ϵ, for any S_j, the number of possible locations for those $h_j(y^n)$ points is equal to $(\frac{|S_j|}{\epsilon^l})^{h_j}$. Thus, the number of elements in the set $\{y^n : h_j(y^n) = h_j, \forall j\}$ is

$$\frac{n!}{h_1!...h_K!} \prod_{j=1}^{K} \left(\frac{|S_j|}{\epsilon^l}\right)^{h_j} \tag{5.30}$$

Therefore,

$$\begin{aligned}
\mathrm{COMP}(n, \widetilde{S}) &= \sum_{h_1+\ldots+h_K=n} [\frac{n!}{h_1!...h_K!} \prod_{j=1}^{K} [(\frac{|S_j|}{\epsilon^l})^{h_j} \prod_{j=1}^{K} (\frac{h_j \cdot \epsilon^l}{n \cdot |S_j|})^{h_j}] \\
&= \sum_{h_1+\ldots+h_K=n} [\frac{n!}{h_1!...h_K!} \prod_{j=1}^{K} [(\frac{|S_j|}{\epsilon^l})^{h_j} (\frac{h_j \cdot \epsilon^l}{n \cdot |S_j|})^{h_j}] \\
&= \sum_{h_1+\ldots+h_K=n} \frac{n!}{h_1!...h_K!} \prod_{j=1}^{K} \left(\frac{h_j}{n}\right)^{h_j},
\end{aligned} \tag{5.31}$$

which completes the proof.

Note that for continuous data y^n, $\mathrm{COMP}(n, \widetilde{S})$ becomes an integral over

$y^n \in S^n$, but by the definition of Riemann integral, (which always exists since ϵ cancels out), the result of COMP$(n, \widetilde{S})$ is the same as Equation (5.31).

5.11 Appendix B: Proof of Proposition 4

Proposition 4: *For any two cut points $C_i, C_k \in C_a$, suppose $C_i < C_k$ and no data points exist in the interval $[C_i, C_k]$, then any cut point $C_j \in [C_i, C_k]$ would not be in the MDL-optimal set of cut points, i.e., we can skip all such C_j during the search process.*

Consider one-dimensional data z^n, and a partition of the data space S, by a set of cut points, denoted as $C^K = \{C_0 = \min z^n, C_1, ..., C_K = \max z^n\}$, the probability of data is

$$P(z^n|C^K) = \prod_{j=1}^{K} (\frac{h_j\,\epsilon}{n\,|S_j|})^{h_j} \tag{5.32}$$

where h_j is the number of data points within the subinterval S_j, and $|S_j|$ is the length of the subinterval S_j.

We regard $P(x^n|C^K)$ as a *continuous* function of the vector $\vec{S} = (|S_1|, ..., |S_K|)$, i.e., we forget about the granularity ϵ for now, and clearly all h_j's are fixed once we fix the $\vec{S}$.

On the other hand, if we keep all h_j's fixed, we can still "move" all the cut points to change $\vec{S}$ while keeping the h_j's fixed, i.e., we can move a cut point V_x within some closed interval, denoted as $[a, b]$, within which no data points exist.

We prove that the maximum of $P(x^n|C^K)$ will always achieved when $V_x = a$ or $V_x = b$ as we keep other cut points fixed. By doing this, we also prove that, given candidate cut points, we only need to consider cut points that are near to the data points, i.e., if for any candidate cut point, it is another two cut points that are closest to it, other than one or more data points, we can then skip this candidate cut point.

Since when we move one single cut point, it only affects the subinterval left and right to that cut point, while all other $|S_j|$'s remain the same, it is sufficient to just prove for the case $K = 2$.

Since now $C_0 = \min_{i\in[n]} x_{i1}$ and $C_2 = \max_{i\in[n]} x_{i1}$, $P(x^n|C^2)$ becomes a function of C_1, and equivalently a function of $|S_1|$, where both C_1 and $|S_1|$ are

bounded as we need to keep h_1 and h_2 fixed, i.e.,

$$\log P(x^n|C^2) = \log \left(\left(\frac{\epsilon h_1}{n|S_1|} \right)^{h_1} \left(\frac{\epsilon h_2}{n(|S| - |S_1|)} \right)^{h_2} \right) \tag{5.33}$$

where we assume $|S_1| \in [a, b]$ for some certain closed interval $[a, b]$. As we want to maximize $\log P(x^n|C^2)$, it is equivalent to *minimizing*

$$F(|S_1|) := h_1 \log |S_1| + h_2 \log(|S| - |S_1|) \tag{5.34}$$

as other terms in Equation (5.33) are constant. Since

$$F'(|S_1|) = \frac{h_1(|S| - |S_1|) - h_2|S_1|}{(|S| - |S_1|)|S_1|}, \tag{5.35}$$

by setting $F'(|S_1|) = 0$, we have

$$|S_1|^* = \frac{h_1}{h_1 + h_2} L. \tag{5.36}$$

We also have

$$F''(|S_1|) = \frac{-(h_1 + h_2)|S_1|^2 + 2h_1|S||S_1| - h_1|S|^2}{(|S| - |S_1|)^2|S_1|^2} < 0 \tag{5.37}$$

because 1) the denominator is always positive apparently, and 2) the numerator is a simple quadratic function which is always negative. The reason is that 1) $-(h_1 + h_2)|S_1| < 0$ and 2) the numerator has no real roots, since

$$(2h_1|S|)^2 - 4(-(h_1 + h_2))(h_1|S|^2) = -4h_2h_1|S_1|^2 < 0. \tag{5.38}$$

Therefore, if $|S_1|^* \notin [a, b]$, $F(|S_1|)$ is monotonic within $[a, b]$; if $|S_1|^* \in [a, b]$, $|S_1|^*$ reaches the *maximum*. In both cases, the minimum of $F(|S_1|)$ will be either a or b, which completes the proof.

5.12 Appendix C: IPD visualizations on case study datasets

Figure 5.12 shows the visualization of the IPD discretization results on two of the case study datasets.

Algorithm 5: PALM

Input: data x^n, data precision ϵ, sample space S, maximum number of splits per partitioning step K_{max}

Output: $\widetilde{S}$, a partition of S

$dir \leftarrow 0$ or 1;
while ***true*** **do**
  foreach $S_k \in \widetilde{S}$ **do**
    Partition S_k as $\widetilde{S_k}$ by finding the optimal cut lines for S_k in direction dir;
    $C^*_{S_k} = \arg\min_{C_{S_k}} L(\{x^n \in S_k\}, C_{S_k})$;
  if $\widetilde{S_k} = \{S_k\}$ *for all* $S_k \in \widetilde{S}$ **then**
    break;
  else
    $\widetilde{S} \leftarrow \bigcup \widetilde{S_k}$;
    $dir \leftarrow 1 - dir$;
$\widetilde{S}_{merge} \leftarrow \widetilde{S}$;
$K_{merge} \leftarrow$ the number of regions of $\widetilde{S}_{merge}$;
while ***true*** **do**
  Get all neighboring pairs of regions of $\widetilde{S}_{merge}$, $Pairs \leftarrow \{(S_j, S_k), \ldots\}$;
  foreach $(S_j, S_k) \in Pairs$ **do**
    $\widetilde{S'_{j,k}} \leftarrow$ merge the pair (S_j, S_k) in $\widetilde{S}_{merge}$;
    Calculate
    $$L(x^n, \widetilde{S'_{j,k}}) = -\log\left(P(x^n|\widetilde{S'_{j,k}})\right) + \log \mathrm{COMP}(n, K_{merge} - 1);$$
  if $\min_{S'_{j,k}} L(x^n, \widetilde{S'_{j,k}}) > L(x^n, \widetilde{S}_{merge})$ **then**
    return $\widetilde{S}_{merge}$;
  else
    $\widetilde{S}_{merge} \leftarrow \arg\min_{\widetilde{S'_{i,j}}} L(x^n, \widetilde{S'_{i,j}})$;
    $K_{merge} \leftarrow K_{merge} - 1$;

Sample size	MISE for subgroup: $L_{true} > 1$	overall MISE
100 000	0.00148	0.00148
300 000	0.00055	0.00074
500 000	0.00051	0.00065
700 000	0.00019	0.00069
1 000 000	0.00023	0.00058
3 000 000	0.00017	0.00055
5 000 000	0.00006	0.00051

Table 5.1: The average MISE of cases when $L_{true} > 1$, and the overall mean of MISE. We show that, when PALM fails to identify part of the true partitions, the learned histogram model still estimates the probability density accurately. The only explanation for these cases is that some neighboring regions in the true partitions have very similar "true" f_j as defined in Equation (5.26), as a result of which PALM does not deem it necessary to further partition these regions.

	Dataset	l_{palm}	l'_{palm}	l_{kde}	$(l_{palm} - l_{kde})/l_{kde}$
1	Amsterdam housing	29976.36	29983.31	30069.78	-0.00
2	Amusement park	270.56	262.0688	227.22	0.19
3	Chengdu taxi	14904.28	14742.05	14073.42	0.06

Table 5.2: The log-likelihood of PALM with partitioning vertically first, l_{palm}, and with partitioning horizontally first, l'_{palm}, and the log-likelihood of KDE, l_{kde}, on the test set for each of the three datasets.

	Dataset	sample size	PALM	KDE
1	Amsterdam housing	20 244	106.821	6.73
2	Amusement park	9 078 623	1134.581	2017.215
3	Chengdu taxi	107 573	60977.285	29.197

Table 5.3: Empirical runtime (in seconds) for the three case study datasets.

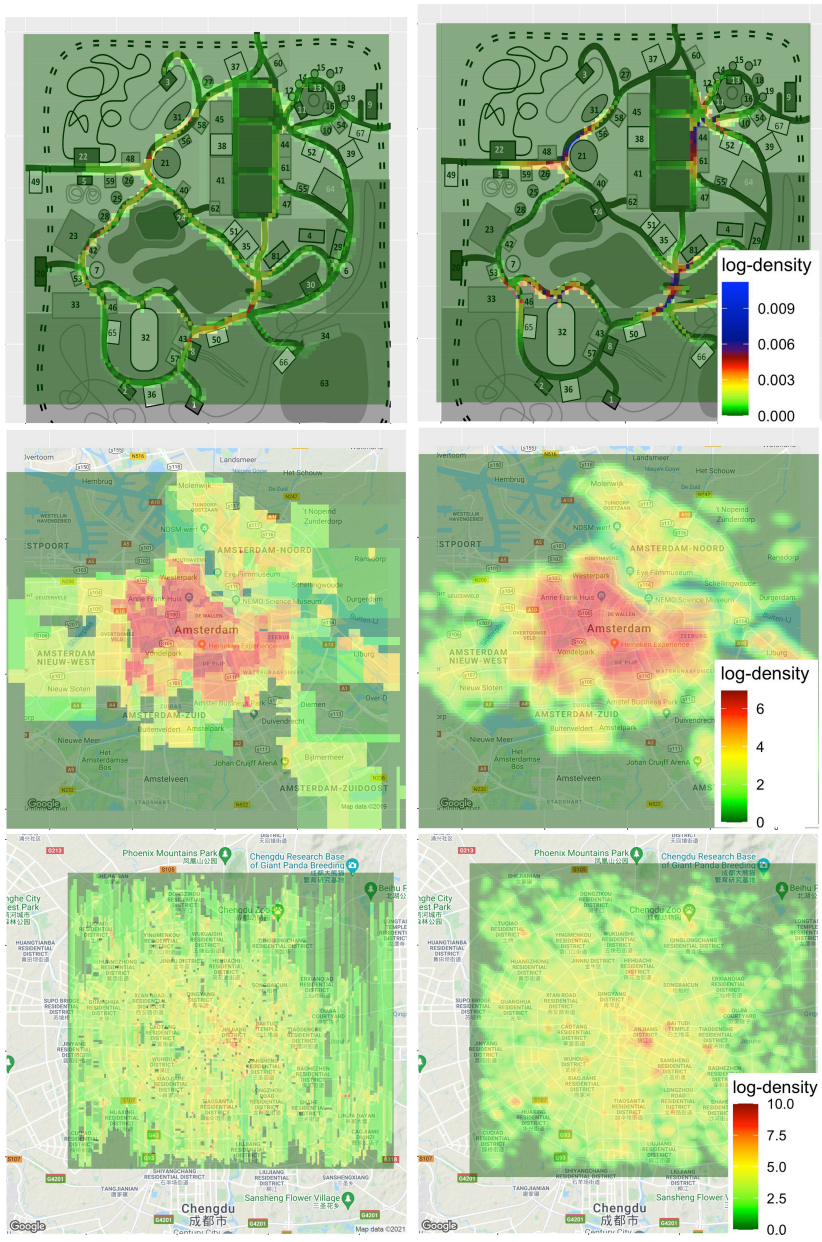

Figure 5.11: Estimated densities on three real-world datasets using PALM (left) and KDE (right); from top to bottom: DinoFun amusement park, Amsterdam Airbnb housing, and taxi destinations in Chengdu.

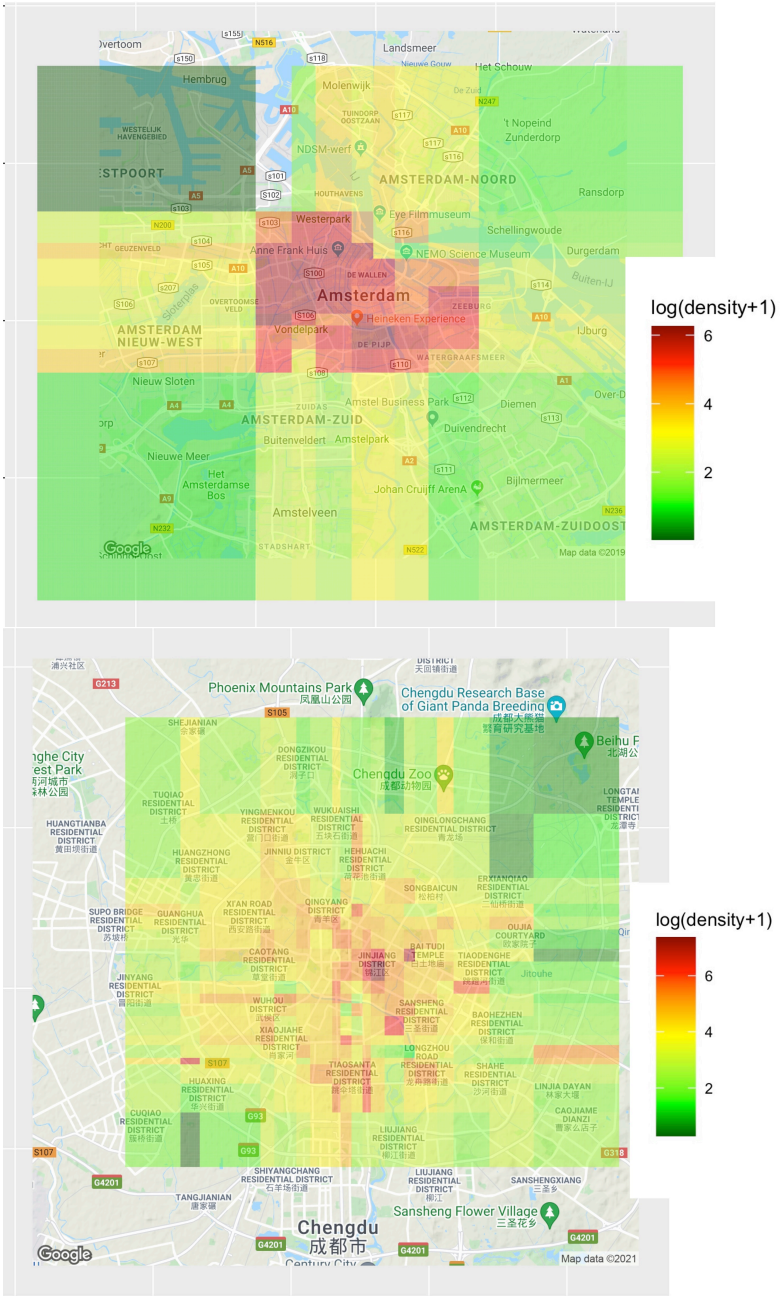

Figure 5.12: Visualization of the IPD discretization results on two of the case study datasets (we fail to obtain the result of IPD on the Amusement Park data within four hours).

Chapter 6

Interpretable Conditional Mutual Information Estimation with Adaptive Histograms

---

This chapter has been published as Marx, A, Yang, L, and van Leeuwen, M **Estimating Conditional Mutual Information for Discrete-Continuous Mixtures using Multi-Dimensional Adaptive Histograms.** In: *Proceedings of the SIAM Conference on Data Mining 2021 (SDM'21), SIAM, 2021.*

Chapter Abstract

Estimating conditional mutual information (CMI) is an essential yet challenging step in many machine learning and data mining tasks. Estimating CMI from data that contains both discrete and continuous variables, or even discrete-continuous mixture variables, is a particularly hard problem. In this chapter, we show that CMI for such mixture variables, defined based on the Radon-Nikodym derivative, can be written as a sum of entropies, just like CMI for purely discrete or continuous data. Further, we show that CMI can be consistently estimated for discrete-continuous mixture variables by learning an adaptive histogram model. In practice, we estimate such a model by iteratively discretizing the continuous data points in the mixture variables. To evaluate the performance of our estimator, we benchmark it against state-of-the-art CMI estimators as well as evaluate it in a causal discovery setting.

6.1 Introduction

In many research areas, such as classification (Lee and Kim 2013), feature selection (Vinh et al. 2014), and causal discovery (Spirtes et al. 2000), estimating the strength of a dependence plays a key role. A theoretically appealing way to measure dependencies is through mutual information (MI) since it has several important properties, such as the chain rule, the data processing inequality, and—last but not least—it is zero if (and only if) two random variables are independent of each other (Cover and Thomas 2012). For structure identification, such as causal discovery, conditional mutual information (CMI) is even more interesting since it can help to distinguish between different graph structures. For instance, in a simple Markov chain $X \to Z \to Y$, X and Y may be dependent, but are rendered independent given Z. Vice versa, a collider structure such as $X \to Z \leftarrow Y$ may introduce a dependence between two marginally independent variables X and Y when conditioned on Z.

While estimating (conditional) mutual information for purely discrete or continuous data is a well-studied problem (Cover and Thomas 2012; Darbellay and Vajda 1999; Gao et al. 2016; Han et al. 2015; Paninski and Yajima 2008), many real-world settings concern a mix of discrete and continuous random variables, such as age (in years) and height, or even random variables that can individually consist of a *mixture* of discrete and continuous components. Although several discretization-based approaches that can estimate MI for a mix of discrete and continuous random variables have recently emerged (Cabeli et al. 2020; Mandros et al. 2020; Suzuki 2016), so far only methods based on k-nearest neighbour (kNN) estimation were shown to work on *mixed variables*, which may consist of discrete-continuous mixture variables (Gao et al. 2017; Mesner and Shalizi 2020; Rahimzamani et al. 2018).

Regardless of the success of kNN-based estimators, discretization-based approaches have attractive properties, e.g., with regard to global interpretation. That is, a natural and understandable way to discretize a continuous random variable is via creating a histogram model, where we cut the sample space of the continuous variable in multiple non-overlapping parts called bins (Scott 2015), or (hyper)rectangles for multi-dimensional variables. Within a bin, we consider the distribution to be constant, which allows us to estimate the density function via

Riemann integration by making the bins smaller and smaller (Cover and Thomas 2012). This definition, however, is less straightforward when mixed variables are involved.

In this chapter, we approach the problem as follows: we first extend the definition of entropy for a univariate discrete-continuous mixture variable given by Politis (Politis 1991) to multivariate variables. Using this definition, we show that CMI for mixed random variables can be written as a sum of entropies that are well-defined through the Radon-Nikodym derivative (see Section 6.2). Exploiting this property, we propose a consistent CMI estimator for such data that is based on adaptive histogram models in Section 6.3. To efficiently learn adaptive histograms from data, in Section 6.4 we define a model selection criterion based on the minimum description length (MDL) principle (Grünwald 2007). Subsequently, we propose an iterative greedy algorithm that aims to obtain the histogram model that minimizes the proposed MDL score in Section 6.5. We discuss related work in Section 6.6 and in Section 6.7, we empirically show that our method performs favourably to state-of-the-art estimators for mixed data and can be used in a causal discovery setting.

6.2 Entropy for Mixed Random Variables

We consider multi-dimensional *mixed random variables*, of which any individual dimension can be discrete, continuous, or a discrete-continuous mixture. Further, we call a vector of such mixed random variables a *mixed random vector*. For a mixed random vector (X, Y), where X and Y are possibly multivariate, we need to adopt the most general definition of mutual information (MI), i.e., the measure-theoretic definition:

$$I(X;Y) = \int_{\mathcal{X}\times\mathcal{Y}} \log \frac{dP_{XY}}{dP_X P_Y} dP_{XY} \ ,$$

where $dP_{XY}/(dP_X P_Y)$ is the Radon-Nikodym derivative, dP_{XY} the joint measure, and $P_X P_Y$ the product measure. It has been proven that $P_X P_Y$ is *absolutely continuous* with respect to P_{XY} (Gao et al. 2017), i.e., $P_{XY} = 0$ whenever $P_X P_Y = 0$; and therefore, such a Radon-Nikodym derivative always exists and $I(X, Y)$ is well-defined. This measure-theoretic definition can be extended to CMI using the

chain rule: $I(X;Y|Z) = I(X;\{Y,Z\}) - I(X;Z)$.

It is common knowledge that for purely discrete or continuous random variables, CMI can be written as a sum of entropies, i.e., $I(X;Y|Z) = H(X,Z) + H(Y,Z) - H(X,Y,Z) - H(Z)$. What is not clear, however, is if this formula also holds when (X,Y,Z) contains discrete-continuous mixture random variables. We investigate this problem in two steps. We first define the measure-theoretic entropy for a (possibly multi-dimensional) discrete-continuous mixture random variable and prove it to be well-defined, though previous work claimed the opposite (Gao et al. 2017). Second, using this definition, we prove that (conditional) MI for a mixed random vector can be written as the sum of measure-theoretic entropies, just like purely continuous or discrete random vectors.

6.2.1 A Generalized Definition of Entropy

The measure-theoretic entropy is defined only for one-dimensional random variables (Politis 1991). Building upon this definition, we give an explicit proof that such a one-dimensional measure-theoretic entropy is well-defined, and then extend this definition to the multi-dimensional case, which we prove is also well-defined.

Generalized One-Dimensional Entropy

We start off by reviewing the existing definition for the one-dimensional case (Politis 1991). Given a one-dimensional random variable X, entropy H is defined as

$$H(X) = \int_{\mathbb{R}} \frac{dP_X(x)}{dv(x)} \log \frac{dP_X(x)}{dv(x)} dv(x), \tag{6.1}$$

where $v(\cdot)$ is a measure defined on all one-dimensional Borel sets (Politis 1991). If $v(\cdot)$ is the Lebesgue measure, which we denote as $u(\cdot)$, $H(X)$ becomes the differential entropy. Alternatively, if $v(\cdot)$ is a counting measure, $H(X)$ becomes the common (discrete) entropy.

If, however, X is a discrete-continuous mixture variable, v is defined as follows. We split $\mathbb{R}$ into three disjoint subsets s.t. $\mathbb{R} = S_d \cup S_c \cup S_o$. First, S_o is the subset of $\mathbb{R}$ on which X has zero probability measure, i.e., $P_X(S_o) = 0$. Second, the set S_d contains all discrete points, i.e., S_d is countable and $\forall x \in S_d, P_X(x) > 0$. Third, S_c covers the continuous points, hence $P_X(S_c) + P_X(S_d) = 1$ and for any

Borel set $A \subseteq S_c$ satisfying $u(A) = 0$, we have $P_X(A) = 0$. Based on these three subsets S_d, S_c, and S_o, we can define v as

$$v(A) = u(A \cap S_c) + |A \cap S_d| \,, \tag{6.2}$$

where $|A \cap S_d|$ is the cardinality of this intersection.

To show that the generalized one-dimensional entropy is well-defined, we need to prove that the Radon-Nikodym derivative dP_X/dv always exists. This we show in the following lemma.

Lemma 1. *Given a one-dimensional discrete-continuous random variable X with probability measure P_X, P_X is absolutely continuous w.r.t. v, i.e., $P_X = 0$ whenever $v = 0$, and hence dP_X/dv always exists.*

We provide the proof of Lemma 1, as well as for Lemmas 2 and 3 in Supplementary Material 6.9.1.

Generalized Multi-Dimensional Entropy

In the following, we extend the measure-theoretic entropy definition to a mixed k-dimensional random vector $W = (W_1, \ldots, W_k)$. For each W_i, we define S_d^i, S_c^i, S_o^i and measure v^i as above, and also define the *product measure* v for the k-dimensional random vector as $v = v^1 \times \ldots \times v^k$. Then, define the entropy for W as

$$H(W) = \int_{\mathbb{R}^k} \frac{dP_W(w)}{dv(w)} \log \frac{dP_W(w)}{dv(w)} dv(w). \tag{6.3}$$

To prove that such entropy is well-defined, we show that dP_W/dv always exists.

Lemma 2. *Given a mixed k-dimensional random vector $W = (W_1, \ldots, W_k)$ with probability measure P_W, dP_W/dv always exists.*

Last, based on Lemma 1 and 2, we can prove that just like for a purely continuous or discrete random vector, conditional mutual information for a mixed random vector can be written as a sum of entropies.

Lemma 3. *Given a mixed random vector (X, Y, Z) with joint probability measure P_{XYZ}, we can write $I(X;Y|Z) = H(X,Z) + H(Y,Z) - H(Z) - H(X,Y,Z)$, where each entropy can be defined as in Equation* (6.3).

As a direct implication of the above proof, it follows that mutual information can also be written as the sum of entropies, since it is a special case of CMI with $Z = \emptyset$. With this generalized definition, we can now show how to estimate CMI using adaptive histogram models.

6.3 Adaptive Histogram Models

Adaptive histogram models have been thoroughly studied for continuous random variables (Scott 2015); however, to the best of our knowledge, there exists no rigorous definition of histograms for mixed random variables. Thus, to use histogram models as a foundation to estimate the measure-theoretic (conditional) MI, we need to rigorously define histograms for mixed random variables. We start with the one-dimensional case.

6.3.1 One-Dimensional Histogram Models

A histogram model is typically defined based on a set of consecutive intervals called *bins* (Scott 2015). However, to deal with discrete-continuous mixture random variables, we define the set of bins, denoted as B, such that each bin is either an interval or a set containing only a single point. That is, $B = B' \cup B''$, where B' and B'' are sets of subsets of $\mathbb{R}$, with B' consisting of countable consecutive intervals and B'' consisting of countable single point sets. Last, we define the "width" of a bin using the measure v as defined in Equation 6.2, i.e., for a bin $B_j \in B$ we have

$$v(B_j) = u(B_j \cap B') + |B_j \cap B''| \,. \tag{6.4}$$

As any $B_j \in B''$ contains only a single discrete point, $v(B_j) = 1$ for all $B_j \in B''$.

Further, we define a histogram model M as a set of bins equipped with a parameter vector of length K, where $K = |B|$ is the number of bins. That is, a histogram model M is a family of probability distributions $P_{X,\theta}$, parametrized by the vector $\theta = (\theta_1, \ldots, \theta_K)$. Each element of θ represents the Radon-Nikodym derivative (or density) of each bin. Note that this definition generalizes to purely continuous random variables when $B'' = \emptyset$ and also to discrete random variables if $B' = \emptyset$. For the latter case, the histogram model degenerates to a multinomial model.

6.3.2 Multi-Dimensional Histograms

First, we define the set of multi-dimensional bins. For a mixed k-dimensional random vector $W = (W_1, \ldots, W_k)$, we define the set of *bins* for each W_i as in Section 6.3.1, denoted as B^i. Consequently, we can define a set of k-dimensional *bins*, denoted B, by the Cartesian product $B = B^1 \times \ldots \times B^k$.

Since each B^i is countable, B is also countable, and we can hence assume B is indexed by j. Then, we split B in a similar way as in the one-dimensional case, i.e., $B = B' \cup B''$, where B'' contains *only discrete values*. That is, for any k-dimensional bin $B_j \in B''$, each dimension of B_j is a set that contains a single one-dimensional point. Note that, however, for any $B_j \in B'$, each dimension of B_j can either be a one-dimensional interval or a one-dimensional single-point set. Further, we define the volume of a multi-dimensional bin $B_j \in B$ using the product measure $v(B_j)$ (see Section 6.2.1).

Similar to one-dimensional histograms, a multi-dimensional histogram model M can be described by a probability distribution $P_{W,\theta}$ parametrized by the vector $\theta = (\theta_1, \ldots, \theta_K)$, where K is the number of bins and θ_i is the Radon-Nikodym derivative for each bin.

6.3.3 Maximum Likelihood Estimator

Given a possibly multi-dimensional histogram with K bins, we denote the Radon-Nikodym derivative $dP_{W,\theta}/dv$ as f^h_θ and its maximum likelihood estimator as $f^h_{\hat{\theta}}$. Observe that for any parameter $\theta_j \in \theta$, the product $\theta_j v(B_j)$ follows a multinomial distribution. Thus, given a dataset $D = \{D_i\}_{i=1,\ldots,n}$, with D_i representing a row, the maximum log-likelihood is denoted as and equal to

$$l_M(D) = \log f^h_{\hat{\theta}(D)}(D) = \log \prod_{j=1}^{K} \left(\frac{c_j}{n \cdot v(B_j)} \right)^{c_j}, \tag{6.5}$$

where c_j and $v(B_j)$ are respectively the number of data points and the bin volumes of bin $j \in \{1 \ldots K\}$. Notice that this maximum likelihood generalizes to the purely discrete case (i.e., multinomial distribution) where all $v(B_j) = 1$, and to the purely continuous case (Scott 2015) where v becomes the Lebesgue measure.

6.3.4 Conditional Mutual Information Estimator

Combining all previous theoretical discussions, we can now estimate conditional mutual information for three (possibly multivariate) random variables X, Y and Z by

$$I^h(X;Y|Z) = H^h(X,Z)+H^h(Y,Z)-H^h(X,Y,Z)-H^h(Z)\ .$$

The corresponding measure-theoretic entropies are estimated from k-dimensional data over (X,Y,Z), where k_X, k_Y and k_Z are the corresponding number of dimensions of X, Y and Z. We estimate the entropies as

$$\begin{aligned} H^h(X,Y,Z) &= -\int_{\mathbb{R}^k} f^h_{\hat{\theta}}(x,y,z)\log(f^h_{\hat{\theta}}(x,y,z))dv \\ H^h(X,Z) &= -\int_{\mathbb{R}^{k_X+k_Z}} f^h_{\hat{\theta}}(x,z)\log(f^h_{\hat{\theta}}(x,z))dv \\ H^h(Y,Z) &= -\int_{\mathbb{R}^{k_Y+k_Z}} f^h_{\hat{\theta}}(y,z)\log(f^h_{\hat{\theta}}(y,z))dv \\ H^h(Z) &= -\int_{\mathbb{R}^{k_Z}} f^h_{\hat{\theta}}(z)\log(f^h_{\hat{\theta}}(z))dv \end{aligned} \tag{6.6}$$

in which $f^h_{\hat{\theta}}(x,y,z)$ is the maximum likelihood estimator given the data, while we obtain $f^h_{\hat{\theta}}(x,z)$, $f^h_{\hat{\theta}}(y,z)$, and $f^h_{\hat{\theta}}(z)$ via marginalization from $f^h_{\hat{\theta}}(x,y,z)$. Next, we will prove that I^h is a strongly consistent estimator for conditional mutual information on mixed data.

Theorem 1. Given a mixed random vector (X,Y,Z) with probability measure P_{XYZ},

$$\lim_{v'\to 0}\lim_{n\to\infty} I^h(X;Y|Z) = I(X;Y|Z)$$

almost surely, where n refers to the sample size and v' refers to the maximum of the histogram volumes for bins in B' (defined in Section 6.3.2).

The proof is provided in Supplementary Material 6.9.1. Informally, our proof is based on the following key aspects: 1) All volume-related terms in I^h cancel out, 2) discrete empirical entropy converges to the true entropy almost surely (Antos and Kontoyiannis 2001), and 3) in the limit, differential entropy can be obtained by discretizing a continuous random variable into "infinitely" small bins Cover and Thomas 2012, Theorem 8.3.1. Notably, the order of the double limit in Theorem 1 inherently indicates that n should grow faster than the number of bins (Rudin et al. 1964), which is also required for histograms on purely continuous

data to converge (Scott 2015).

6.4 Learning Adaptive Histograms from Data

To efficiently estimate a histogram model that inherits the consistency guarantees from Theorem 1 we need to consider the following requirements. First of all, we need to ensure that we learn a joint histogram model over (X, Y, Z). This is due to the fact that we obtain the lower-dimensional entropies such as $H^h(X, Z)$ by marginalization over the likelihood estimator $f^h_{\hat{\theta}}(x, y, z)$. If we would not learn a joint model, the volume-related terms in $H^h(X, Y, Z), H^h(X, Z), H^h(Y, Z)$, and $H^h(Z)$ would not cancel out. In addition, we need to make sure that the number of bins is in $o(n)$ and increases if we were to increase the number of samples n, while at the same time the size of the bins decreases.

One way to achieve those properties would be to fix the bin width or the number of bins depending on the number of samples. However, such an approach is not very flexible and does not allow for variable bin widths. To allow for a more flexible model, we formally consider the problem of constructing an adaptive multi-dimensional histogram as a model selection problem and employ a selection criterion based on the minimum description length (MDL) principle (Rissanen 1978). MDL-based model selection has been successfully used for learning one-dimensional (Kontkanen and Myllymäki 2007b) and two-dimensional histograms (Kameya 2011; Yang et al. 2023), demonstrating adaptivity to both local density changes and sample size.

We now briefly introduce the MDL principle and define the MDL-optimal histogram model. Specifically, while previous work (Kameya 2011; Kontkanen and Myllymäki 2007b; Yang et al. 2023) only considers purely continuous data (or more precisely, data with arbitrarily small precision), we apply the MDL principle to mixed-type data, based on our rigorous definition of histogram models for mixed random variables. On top of that, we empirically show that our score fulfils the desired properties—i.e. the number of bins grows as $o(n)$.

6.4.1 MDL and Stochastic Complexity

The minimum description length principle is arguably one of the best off-the-shelf model selection criteria (Grünwald 2007), which has been successfully

applied to many machine learning and data mining tasks. The general idea is to assign a code length to data D compressed by a model M, e.g., a histogram model. Given a collection of candidate models, denoted as $\mathcal{M}$, MDL selects the model M^* that minimizes the joint code length of the model and the data. Formally, our goal is to find

$$M^* = \arg\min_{M \in \mathcal{M}} L(D|M) + L(M), \tag{6.7}$$

where $L(D|M)$ denotes the code length[1] of the data given the model, while $L(M)$ denotes the code length needed to encode the model.

The optimal way of encoding data D given M, in the sense that it will result in minimax *regret*, is to use the *normalized maximum likelihood (NML)* code (Grünwald 2007). Accordingly, the code length of the data is called *stochastic complexity (SC)*, which is defined as the sum of the negative log-likelihood $-l_M(D)$, defined in Equation 6.5, and the *parametric complexity* (also called *regret*) $\log R(n, K)$ (Grünwald 2007). The parametric complexity of a histogram model with K bins is given by (Kontkanen and Myllymäki 2007b; Yang et al. 2023)

$$R(n, K) = \sum_{c_1 + \cdots + c_K = n} \frac{n!}{c_1! \cdots c_K!} \prod_{i=1}^{K} \left(\frac{c_i}{n}\right)^{c_i} ,$$

and can be computed in sub-linear time (Mononen and Myllymäki 2008).

6.4.2 Code Length of the Model

Given a dataset D with n rows and k individual columns D^j, we now define the model class $\mathcal{M}$. First, we create fixed bins according to B'' (as defined in Section 6.3.2) per discrete value that occurs in D_j. Next, we enumerate all possible bins for B' with fixed precision ϵ. To this end, denote the remaining non-discrete data points in D_j as D_j^c. If D_j^c is empty D_j corresponds to a discrete variable and we can stop here. Otherwise, we create all possible cut points for D_j^c as $C_j^0 = \{\min(D_j^c), \min(D_j^c)+\epsilon, \ldots, \max(D_j^c)\}$. By selecting a subset of cut points $C_j \subseteq C_j^0$, we get a valid solution for B'. We can enumerate all possible segmentations by enumerating each $C_j \subseteq C_j^0$.

By repeating this process for each dimension, we obtain our model class $\mathcal{M}$.

[1] The code length L denotes the number of bits needed to describe the given object. Hence, all logarithms are to base 2 and $0 \log 0 = 0$.

Further, we get the code length for a model $M \in \mathcal{M}$ by encoding all combinations of cut points for each dimension (Kontkanen and Myllymäki 2007b), i.e.,

$$L(M) = \sum_{j \in \{1,\ldots,k\}} L(C_j) = \sum_{j \in \{1,\ldots,k\}} \log \binom{|C_j^0|}{|C_j|} . \tag{6.8}$$

This completes the definition of our final optimization score $L(D|M) + L(M)$.

To proof consistency for this score, we need to show that the number of selected bins grows at rate $o(n)$. Since the theoretical analysis is rather difficult, we instead empirically demonstrate this property for Gaussian distributed data in Supplementary Material 6.9.3. In the next section, we present an iterative greedy algorithm that optimizes our MDL score.

6.5 Implementation

In this section, we describe our algorithm to estimate the joint entropy $H(X_1, \ldots, X_k)$ for a k-dimensional discrete-continuous mixture random vector.

6.5.1 Algorithm

To discretize a one-dimensional random variable X, we first create bins for the discrete values of X and then discretize the continuous values. We detect discrete data points by checking if a single value x in the domain $\mathcal{X}$ of X occurs multiple times. If a user-defined threshold t, e.g., 5 is reached, we create a bin for this point. To discretize the remaining continuous values, we start by splitting $\mathcal{X}$ into K_{init} equi-width bins, which we can safely choose from the complexity class $o(\sqrt{n})$ (see Supplementary Material 6.9.3). Using dynamic programming, we compute the variable-width histogram model M that minimizes $L(D, M)$ in quadratic time w.r.t. K_{init} (Kontkanen and Myllymäki 2007b).

Since the runtime complexity to compute the optimal variable-width histogram over a multi-dimensional random variable would grow exponentially w.r.t. k, we opt for an iterative greedy algorithm (we provide the pseudocode in Supplementary Material 6.9.2). We start by initializing the optimization: for every dimension, we fix bins for the discrete values and put the remaining continuous values into a single bin. Then, in each iteration, we compute a candidate dis-

cretization for each dimension and keep the discretization of that dimension that provides the highest gain in compression. To compute a candidate discretization for a dimension X_j, we extend the one-dimensional algorithm described above. That is, we determine those cut points for X_j that provide the highest gain in $L(D, M)$, while keeping the bins for the remaining dimensions fixed. We repeat this until the maximum number of iterations $i_{\max}$ is reached or we cannot further decrease $L(D, M)$.

6.5.2 Complexity

The complexity of discretizing a univariate random variable is in $\mathcal{O}(K_{\max} \cdot (K_{\text{init}})^2)$ and depends on the number of initial bins K_{init} and the maximum number of bins $K_{\max}$, which we typically chose as a fraction of K_{init} (both in $o(\sqrt{n})$). In a multi-dimensional setting we have to multiply this complexity by the current domain size of the remaining variables, since we have to update each bin conditioned on those. In the worst case, this number is equal to $(K_{\max})^{k-1}$. Overall, we apply this procedure—if all variables are continuous—$i_{\max} \cdot k$ times.

6.6 Related Work

We discuss related methods for adaptive histograms and (conditional) mutual information estimation.

Both theoretical properties and practical issues of density estimation using histograms have been studied for decades (Scott 2015). Various algorithms have been proposed for the challenging task of constructing an adaptive one-dimensional histogram, among which the MDL-based histogram (Kontkanen and Myllymäki 2007b) is considered to be the state-of-the-art, as it is self-adaptive to both local density structure and sample size and does not have any hyperparameters. Learning adaptive multivariate histograms is even harder due to the combinatorial explosion of the search space. One approach is to resort to the dyadic CART algorithm (Klemelä 2009); various methods designed for specific tasks also exist (Kameya 2011; Weiler and Eggert 2007). Our algorithm is similar to that of Kameya (Kameya 2011), but they only consider the two-dimensional case.

For discrete data, (conditional) mutual information estimation is a well-studied problem (Cover and Thomas 2012; Han et al. 2015; Marx and Vreeken 2019; Paninski 2003; Valiant and Valiant 2011) and it has been shown that mutual information can be estimated using the $3H$ principle (Han et al. 2015). An important observation is that for discrete data, the empirical estimator for entropy is sub-optimal (Paninski 2003), which encouraged the design of more efficient entropy estimators with sub-linear sample complexity (Han et al. 2015; Valiant and Valiant 2011).

For estimating (conditional) mutual information on continuous data or a mix of discrete and continuous data, three classes of approaches exist. The first class concerns kernel density estimation (KDE) methods (Gao et al. 2016; Paninski and Yajima 2008), which perform well on continuous data; however, no KDE-based MI and CMI estimation methods exist that are designed for discrete-continuous mixture random variables. Moreover, bandwidth tuning for KDE can be computationally expensive, which becomes even worse for mixed data, as different bandwidths may be needed for discrete random variables. The second class of methods relies on k-nearest neighbour (kNN) estimates (Frenzel and Pompe 2007; Kozachenko and Leonenko 1987; Kraskov et al. 2004), which have been established as the state of the art (Gao et al. 2017; Rahimzamani et al. 2018). kNN approaches can be applied not only to a mix of discrete and continuous variables, but can also be used as consistent MI (Gao et al. 2017) and CMI (Mesner and Shalizi 2020; Rahimzamani et al. 2018) estimators for discrete-continuous mixtures. The third class of methods first discretizes the continuous random variables and then calculates mutual information from the discretized variables (Cover and Thomas 2012; Darbellay and Vajda 1999; Suzuki 2016). Two recent approaches based on adaptive partitioning for mixed random variables have been proposed (Cabeli et al. 2020; Mandros et al. 2020). While Mandros et al. (2020) focus on mutual information and its application to functional dependency discovery, Cabeli et al. (2020), similar to us, build upon an MDL-based score to estimate MI and CMI, to which we compare in Section 6.7. The key difference is that Cabeli et al. (2020) compute $I(X;Y|Z)$ as $(I(X;\{Y,Z\}) - I(X;Z) + I(Y;\{X,Z\}) - I(Y;Z))/2$ and maximize each of the four terms (with penalty terms) directly, while we first learn a joint histogram.

To the best of knowledge, we are the first to propose a CMI estimator for

discrete-continuous mixture variables based on discretization or histogram density estimation. Our method can consistently estimate CMI on mixed random variables containing discrete-continuous mixtures. We focus on histogram-based models instead of kNN estimation, since histograms are more interpretable (Scott 2015) and do not require tuning of the parameter k, which can have a large impact on the outcome.

6.7 Experiments

In this section, we empirically evaluate the performance of our approach. First, we will benchmark our estimator against state-of-the-art CMI estimators on different data types. After that, we evaluate how well our estimator is suited to test for conditional independence in a causal discovery setup. For reproducibility, we make our code available online.[2]

6.7.1 Mutual Information Estimation

On the mutual information estimation task, we compare our approach to the state-of-the-art MI estimators. In particular, we compare against FP (Frenzel and Pompe 2007), RAVK (Rahimzamani et al. 2018) and MS (Mesner and Shalizi 2020), which all rely on kNN estimates, and MIIC (Cabeli et al. 2020), which is a discretization-based method. All of those can be applied to our setup, but only the authors of RAVK and MS specifically consider discrete-continuous mixture variables. We apply MIIC using the default parameters and use $k = 10$ for all kNN-based approaches.[3] For our algorithm, we set the maximum number of iterations and the threshold to detect discrete points in a mixture variable to 5, set $K_{\text{init}} = 20 \log n$ and $K_{\max} = 5 \log n$. To comply with the literature, we compute all entropies in this section using the *natural logarithm*.

Experiment I-IV

As a sanity check, we start with an experiment on purely continuous data. That is, for **Experiment I**, let X and Y be Gaussian distributed random variables

[2] https://github.com/ylincen/CMI-adaptive-hist.git

[3] We evaluated all approaches with $k = 5, 10, 20$. Since $k = 10$ had the best trade-off and is close to $k = 7$ as used by Mesner and Shalizi (Mesner and Shalizi 2020), we report those results.

with mean 0, variance 1, and covariance 0.6. Consequently, the correlation ρ between X and Y is 0.6 and true MI can be calculated as $I(X;Y) = -\frac{1}{2}\log(1-\rho^2)$. In **Experiment II**, X is discrete and drawn from $\text{Unif}(0, m-1)$, with $m = 5$ and Y is continuous with $Y \sim \text{Unif}(x, x+2)$ for $X = x$. Therefore, $I(X;Y) = \log(m) - \frac{(m-1)\log 2}{m}$ (Gao et al. 2017). Next, for **Experiment III**, X is exponentially distributed with rate 1 and Y is a zero-inflated Poissonization of X—i.e., $Y = 0$ with probability $p = 0.15$ and $Y \sim \text{Pois}(x)$ for $X = x$ with probability $1 - p$. The ground truth is $I(X;Y) = (1-p)(2\log 2 - \gamma - \sum_{k=1}^{\infty} \log k \cdot 2^{-k}) \approx (1-p)0.3012$, where γ is the Euler-Mascheroni constant (Gao et al. 2017). Last, in **Experiment IV**, we generate the data according to the Markov chain $X \to Z \to Y$ (see Mesner and Shalizi (Mesner and Shalizi 2020)). In particular, X is exponentially distributed with rate $\frac{1}{2}$, $Z \sim \text{Pois}(x)$ for $X = x$ and Y is binomial distributed with size $n = z$ for $Z = z$ and probability $p = \frac{1}{2}$. Due to the Markov chain structure, the ground truth $I(X;Y \mid Z) = 0$.

For each of the above experiments, we sample data with sample size $n \in \{100, 200, \ldots, 1\,000\}$ and generate 100 data sets per sample size. We run each of the estimators on the generated data and show the mean squared error (MSE) of each estimator in Figure 6.1. Overall, our estimator performs best or very close to the best throughout the experiments and reaches an MSE lower than 0.001 with at most 1 000 samples. The best competitors are MS and MIIC; however, both are biased when we consider discrete-continuous mixture variables, as we show in Experiment V.

Experiment V

Next, we generate data according to a discrete-continuous mixture (Gao et al. 2017). Half of the data points are continuous, with X and Y being standard Gaussian with correlation $\rho = 0.8$, while the other half follows a discrete distribution with $P(1,1) = P(-1,-1) = 0.4$ and $P(1,-1) = P(-1,1) = 0.1$. In addition, we generate Z independently with $Z \sim \text{Binomial}(3, 0.2)$. Hence the ground truth is equal to $I(X;Y) = I(X;Y \mid Z) = 0.4 \cdot \log \frac{0.4}{0.5^2} + 0.1 \cdot \log \frac{0.1}{0.5^2} - \frac{1}{4}\log(1-0.8^2) \approx 0.352$.

In Figure 6.2 (top) we show the mean and MSE for this experiment. We see that our estimator starts by overestimating the true value, but its average quickly converges to the true value, while the competing estimators seem to have a slightly positive or negative bias. Especially FP and MIIC, which were not designed for

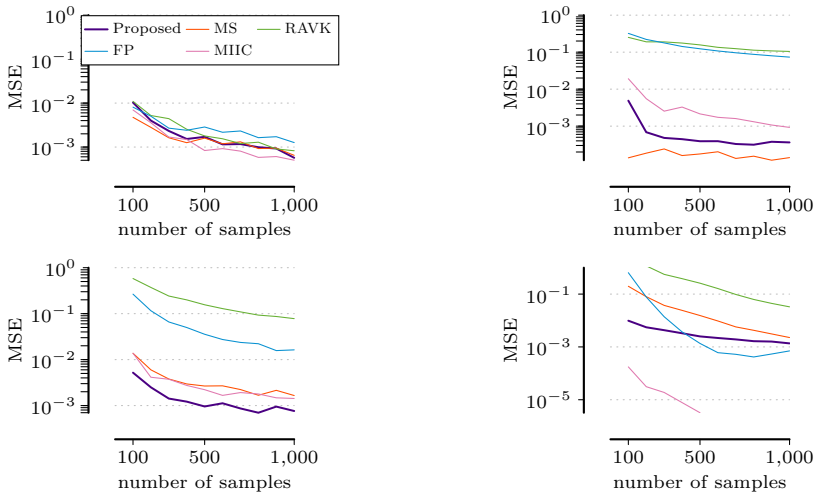

Figure 6.1: Synthetic data with known ground truth. Ordered from top-left to bottom-right, we show the MSE for Experiments I-IV, for our estimator and competing algorithms MS, RAVK, FP and MIIC.

this setup, have a clear bias even for 1 000 data points. The same trend can be observed for MSE.

Experiment VI

Last, we test how sensitive our method is to dimensionality. We generate X and Y as in Experiment II, but fix n to 2 000 and add k independent random variables, $Z_k \sim \text{Binomial}(3, 0.5)$.

Figure 6.2 (bottom) shows the mean and MSE. Our estimator recovers the true CMI up to a negligible error up to $k = 2$. After that, it starts to slowly underestimate the true CMI. This can be explained by the fact that the model costs increase linearly with the domain size and hence, we will fit fewer bins to the continuous variable for large k. We validated this conjecture by repeating the experiment for $n = 10\,000$. On this larger sample size, the MSE for our estimator remained below 0.001 even for $k = 4$. While MIIC is slightly more stable for $k \geq 3$, the competing kNN-based estimators deviate quite a bit from the true estimate for higher dimensions.

Overall, we are on par with or outperform the best competitor throughout Experiments I–VI. Especially on mixture data, which is our main focus, our method

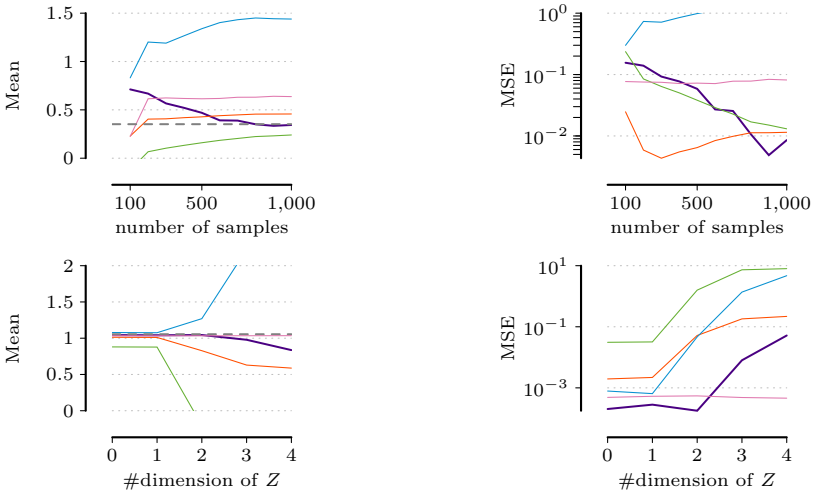

Figure 6.2: Top row: Experiment V, where we show the mean of the estimators (left) with the true CMI as a dashed gray line and the MSE (right). Bottom row: Experiment VI, where the sample size is constant at 2 000 and the x-axis refers to the number of dimensions of Z. We show the mean (left) and MSE (right). The color coding is chosen as in Figure 6.1.

is the only one that converges to the true estimate.

6.7.2 Independence Testing

In theory, two random variables X and Y are conditionally independent given a set of random variables Z, denoted as $X \perp\!\!\!\perp Y \mid Z$, if $I(X;Y \mid Z) = 0$. Vice versa, X and Y are dependent given Z, if $I(X;Y \mid Z) > 0$. In practice, we cannot simply rely on our estimator to conclude independence: due to the monotonicity of mutual information, i.e., $I(X;Y) \leq I(X;Y \cup Z)$, estimates will rarely be *exactly zero* in the limited sample regime, but only *close to zero* (Marx and Vreeken 2019; Vinh et al. 2014). To address this problem, we use our algorithm to discretize X, Y and Z, and compute $I_{\mathcal{C}}(X;Y|Z) := \max\{0, I_n(X_d;Y_d|Z_d) + \mathcal{C}_n(X_d;Y_d|Z_d)\}$, where $\mathcal{C}_n$ is a correction term calculated from the discretized variables, which is negative. In the following, we evaluate our estimator with two different correction criteria. The first one is a correction for mutual information based on the Chi-squared distribution, with $\mathcal{C}_n$ equal to $-\mathcal{X}_{\alpha,l}/2n$ (Vinh et al. 2014), where $\mathcal{X}_{\alpha,l}$ refers to the critical value of the Chi-squared distribution with significance level α and degrees of freedom l. We can compute the degrees of freedom l from the domain

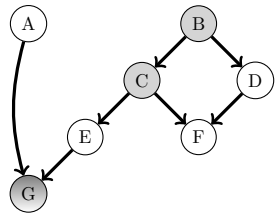

Figure 6.3: Synthetic network with continuous (white), discrete (gray) and mixed (shaded) random variables consisting of different causal structures, such as colliders, a chain ($C \to E \to G$), and a fork ($C \leftarrow B \to D$).

sizes of the discretized variables for the conditional case as $l = (|\mathcal{X}| - 1)(|\mathcal{Y}| - 1)|\mathcal{Z}|$ (Suzuki 2019), and for the unconditional case as $l = (|\mathcal{X}| - 1)(|\mathcal{Y}| - 1)$. For the second correction, we replace each empirical entropy in $I_n(X_d; Y_d|Z_d)$ with its corresponding stochastic complexity term as defined in Section 6.4.1. If we subtract the regret terms for $H_n(X_d, Y_d, Z_d)$ and $H_n(Z_d)$ from those for $H_n(X_d, Z_d)$ and $H_n(Y_d, Z_d)$, we are guaranteed to get a negative value, thus a valid regret term (Marx and Vreeken 2019). In the following, we refer to the test using the Chi-squared correction as $I_{\mathcal{X}^2}$ and to the one based on stochastic complexity as I_{SC}.

To test how well $I_{\mathcal{X}^2}$ and I_{SC} perform on mixed-type and continuous data, we benchmark both against state-of-the-art kernel-based tests RCIT and RCoT (Strobl et al. 2019), as well as JIC (Suzuki 2016), and MIIC (Cabeli et al. 2020), which are both discretization-based methods using a correction based on stochastic complexity.[4] To apply RCIT and RCoT on mixed data, we treat the discrete data points as integers. In the following, we evaluate the performance of each test in a causal discovery setup. In addition, we provide a more detailed description of the data generation and experiments on individual collider and non-collider structures in Supplementary Material 6.9.3.

Causal Discovery

To evaluate our test in a causal discovery setting, we generate data according to a small synthetic network—shown in Figure 6.3—that consists of a mixture of generating mechanisms that we used in experiments I-IV and includes continuous

[4]Note that MIIC calculates stochastic complexity based on factorized NML and JIC uses an asymptotic approximation of stochastic complexity, while we use quotient NML for I_{SC} (Marx and Vreeken 2019; Silander et al. 2018).

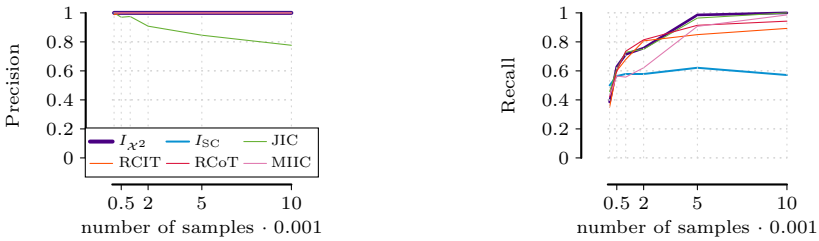

Figure 6.4: Precision (left) and recall (right) on undirected graphs inferred using the PC-stable algorithm equipped with the corresponding independence test. The data is generated from the graph shown in Figure 6.3.

and discrete (ordinal) random variables and one mixture variable, which is partially Gaussian and partially Poisson distributed (for details see Supplementary Material 6.9.3). To evaluate how well the ground truth graph can be recovered, we apply the PC-stable algorithm (Colombo and Maathuis 2012; Spirtes et al. 2000) equipped with the different independence tests, where we use $\alpha = 0.01$ for $I_{\mathcal{X}^2}$, RCIT and RCoT.

Fig 6.4 shows recovery precision and recall for the undirected graph, averaged over 20 draws per sample size $n \in \{100, 500, 1\,000, 2\,000, 5\,000, 10\,000\}$.

We see that overall $I_{\mathcal{X}^2}$ performs best and is the only method that reaches both a perfect accuracy and recall. While JIC also reaches a perfect recall, it finds too many edges leading to a precision of only 80%. Although also MIIC, RCIT and RCoT have a perfect precision, their recall is worse than for $I_{\mathcal{X}^2}$. Neither of the kernel-based tests manages to recall all the edges even for 10 000 samples. After a closer inspection, this is due to the edge $E \to G$ that involves the discrete-continuous variable G. If we compare $I_{\mathcal{X}^2}$ to I_{SC}, we clearly see that the latter is too conservative, which leads to a bad recall.

6.8 Conclusion

We proposed a novel approach for the estimation of conditional mutual information from data that may contain discrete, continuous, and mixture variables. To be able to deal with discrete-continuous mixture variables, we defined a class of generalized adaptive histogram models. Based on our observation that CMI for mixture-variables can be written as a sum of entropies, we presented a CMI

estimator based on such histograms, for which we proved that it is consistent.

Further, we used the minimum description length principle to formally define optimal histograms, and proposed a greedy algorithm to practically learn good histograms from data. Finally, we demonstrated that our algorithm outperforms state-of-the-art (conditional) mutual information estimation methods, and that it can be successfully used as a conditional independence test in causal graph structure learning. Notably, for both setups, we observe that our approach performs especially well when mixture variables are present.

6.9 Supplementary Material

The supplementary material is structured as follows. First, we provide proofs for all lemmas and theorems. After that, we provide the pseudocode for our algorithm. Last, we provide additional experiments and details for the data generation for the causal discovery experiment.

6.9.1 Proofs

Proof of Lemma 1

Proof. Given a Borel set $A \subseteq \mathbb{R}$ such that $v(A) = u(A \cap S_c) + |A \cap S_d| = 0$, we have $u(A \cap S_c) = 0$ due to non-negativity of any measure, as well as $|A \cap S_d| = 0$. Since $A \cap S_c \subseteq S_c$, by the definition of S_c we have $P(A \cap S_c) = 0$. It remains to show that $A \cap S_d = \emptyset$, which we do by contradiction. Assume that $A \cap S_d \neq \emptyset$, then there exists $x \in A \cap S_d$ s.t. for a set containing only x, $|\{x\}| = 1$. Then $|A \cap S_d| \geq |\{x\}| = 1$, which contradicts $|A \cap S_d| = 0$. Thus, we must have $A \cap S_d = \emptyset$ and then $P_X(A) = 0$. □

Proof of Lemma 2

Proof. Given a k-dimensional Borel set A, there exist one-dimensional Borel sets $A_1, \ldots, A_k$ such that $A = A_1 \times \ldots \times A_k$. If $v(A) = 0$, then there exists at least one $v^i, i \in \{1, \ldots, k\}$, such that $v^i(A_i) = 0$. Thus, by Lemma 1, $P_{W_i}(A_i) = 0 \Rightarrow P_W(\mathbb{R} \times \ldots \times \mathbb{R} \times A_i \times \mathbb{R} \times \ldots \times \mathbb{R}) = 0 \Rightarrow P_W(A) = 0$, as $A = A_1 \times \ldots \times A_k \subseteq \mathbb{R} \times \ldots \times \mathbb{R} \times A_i \times \mathbb{R} \times \ldots \times \mathbb{R}$. □

Proof of Lemma 3

Proof. We first proof the statement for $Z \neq \emptyset$, for which we can write $I(X;Y|Z) = I(X;\{Y,Z\}) - I(X;Z)$ by the chain rule for mutual information. Thus, it suffices to prove that $I(X;Z) = H(X) + H(Z) - H(X,Z)$ and $I(X;\{Y,Z\}) = H(X) + H(Y,Z) - H(X,Y,Z)$. Next, denote v as the product measure defined based on (X,Z), where $v = v^1 \times \ldots \times v^{k_{XZ}}$, and k_{XZ} is the number of dimensions of X plus that of Z; then by Lemma 2, we also have $P_{XZ} \ll v$. Then, we show that $P_X P_Z \ll v$. For some k_{XZ}-dimensional Borel set $A = A_1 \times \ldots \times A_{k_{XZ}}$ satisfying $v(A) = 0$ there exists $v^i \in \{v^1, \ldots, v^{k_{XZ}}\}$ such that $v^i(A_i) = 0$. Hence, $P_X P_Z(A) = 0$ because $0 \leq P_X P_Z(A) = P_X P_Z(A_1 \times \ldots \times A_{k_{XZ}}) \leq P_X P_Z(\mathbb{R} \times \ldots \mathbb{R} \times A_i \times \mathbb{R} \ldots \times \mathbb{R}) = P_i(A_i) = 0$, where P_i is the marginalization of the product measure $P_X P_Z$ to the ith dimension and $P_i(A_i) = 0$ is because $v^i(A_i) = 0$ by the definition of v.

Finally, by the chain rule of the Radon-Nikodym derivative we have that

$$
\begin{aligned}
I(X;Z) &= \int \log \frac{dP_{XZ}}{dP_X P_Z} dP_{XZ} && (6.9)\\
&= \int \log \frac{dP_{XZ}/dv}{dP_X P_Z/dv} (dP_{XZ}/dv) dv && (6.10)\\
&= H(X) + H(Z) - H(X,Z)\ . && (6.11)
\end{aligned}
$$

The proof for $I(X;\{Y,Z\})$ is equivalent. If $Z = \emptyset$, CMI reduces to $I(X;Y)$, for which we can prove the statement in the same manner. □

Proof of Theorem 1

To proof Theorem 1 we need several intermediate results. Lemma 6 shows that a histogram results in a valid discretization as all terms corresponding to volumes in I^h cancel out, and hence I^h can be written as a sum of plug-in estimators of *discrete entropies*. Then, Lemma 4 shows a classic result that the plug-in estimator of discrete entropies will converge to the true entropy almost surely. Further, we show in Lemma 5 that as the volumes of histogram bins containing continuous values go to 0, the true entropies of discretized variables (which are discretized by the histogram) converges to the true entropy of original variables.

Definition 1. *Given discrete random variables X_d, Y_d, Z_d (possibly multi-dimensional),*

with support S_d^X, S_d^Y, S_d^Z*, and given dataset* $D = (x_i, y_i, z_i)_{i\in\{1,\dots,n\}}$ *with sample size* n*, the plug-in estimator of discrete entropy* H *is denoted and defined as*

$$H_n(X_d, Y_d, Z_d) = -\sum_{j\in S_d^X\times S_d^Y\times S_d^Z} \hat{p}(j)\log\hat{p}(j)$$

with probability estimates

$$\hat{p}(j) = \frac{|\{(x_i, y_i, z_i)_{i\in\{1,\dots,n\}} : (x_i, y_i, z_i) = q_j\}|}{n} ,$$

where $|\cdot|$ *represents the cardinality of a set, and* q_j *is the* j*th element in* $S_d^X \times S_d^Y \times S_d^Z$*.*

Lemma 4. *Given a discrete random vector* (X_d, Y_d, Z_d)*,* $lim_{n\to\infty} H_n(X_d, Y_d, Z_d) = H(X_d, Y_d, Z_d)$ *almost surely (Antos and Kontoyiannis 2001).*

Lemma 5. *Given a random vector* (X, Y, Z) *that contains discrete-continuous mixture random variables, with bins* $B = B' \cup B''$ *and the resulting discretized random vector* (X_d, Y_d, Z_d)*, where* B'' *contains discrete data points (of which every dimension has a discrete value) and* $B' = B \setminus B''$*, we have*

$$\lim_{v'\to 0} H(X_d, Y_d, Z_d) = H(X, Y, Z) ,$$

where $v' = \max_{B_j\in B'}(v(B_j))$*.*

Proof. Firstly, it is well-known that this result holds if (X, Y, Z) is a continuous random vector (Cover and Thomas 2012); then, if (X, Y, Z) contains mixture variables,

$$H(X, Y, Z) = \lim_{v'\to 0} \sum_{B_j\in B'} \frac{P_{X_dY_dZ_d}}{v(B_j)} \log \frac{P_{X_dY_dZ_d}}{v(B_j)} \tag{6.12}$$

$$+ \sum_{B_j\in B''} \frac{P_{X_dY_dZ_d}}{v(B_j)} \log \frac{P_{X_dY_dZ_d}}{v(B_j)} \tag{6.13}$$

$$= \lim_{v'\to 0} H(X_d, Y_d, Z_d) , \tag{6.14}$$

which concludes the proof. □

Definition 2. *Given a random vector* (X, Y, Z) *that contains mixture variables, and an adaptive grid* B*, we define the discretized random variable* X_d, Y_d, Z_d*, with*

probability measure (probability mass function)

$$P_{X_d,Y_d,Z_d}((j_1,j_2,j_3)) = \int_{B_j} \frac{d_{XYZ}}{dv} dv \ ,$$

where B_j denotes the jth bin of B.

Lemma 6. *Given a k-dimensional random vector (X,Y,Z) that contains mixture variables with an unknown probability measure P_{XYZ}, a dataset $D = (x_i, y_i, z_i)_{i\in\{1,\dots,n\}}$ generated by P_{XYZ}, a histogram model M, and corresponding discretized random vector (X_d, Y_d, Z_d), we have $I^h(X,Y|Z)$ is equal to*

$$H_n(X_d,Z_d) + H_n(Y_d,Z_d) - H_n(X_d,Y_d,Z_d) - H_n(Z_d) \ .$$

That is, the terms corresponding to volumes in I^h cancel out and our histogram model results a valid discretization.

Proof. Denote the adaptive grid of histogram model M as B^{XYZ}, which is the Cartesian product of bins defined on X, Y, Z—i.e. $B^{XYZ} = B^X \times B^Y \times B^Z$, and denote the corresponding MLE of histogram density function as $f^h_{\hat{\theta}_{XYZ}}$. Further, define a function v_X, such that for each x_i in D, $v_X(x_i) = v(B^X_j)$ if $x_i \in B^X_j$, where B^X_j is a bin of B^X and v is defined based on the random variable X. Then, define $v_Y, v_Z, v_{XZ}, v_{YZ}, v_{XYZ}$ similarly.

By the definition $I^h(X,Y|Z)$ is equal to

$$H^h(X,Z) + H^h(Y,Z) - H^h(X,Y,Z) - H^h(Z) \ .$$

First consider $H^h(X,Z)$. We write $B^{XZ} = B^X \times B^Z$, with marginal density function $f^h_{\hat{\theta}_{XZ}}$. W.l.o.g. suppose that B_{XZ} consists of K bins, denoted as $B^{XZ}_j, j \in$

$\{1, \ldots, K\}$. Then,

$$\begin{aligned} H^h(X,Z) &= -\int_{\mathbb{R}^{k_X+k_Z}} f^h_{\hat{\theta}_{XZ}} \log f^h_{\hat{\theta}_{XZ}} dv \\ &= -\sum_{j=1}^{K} \int_{B_j^{XZ}} f^h_{\hat{\theta}_{XZ}} \log f^h_{\hat{\theta}_{XZ}} dv \\ &= -\sum_{j=1}^{K} c_j \log\left(\frac{c_j}{nv(B_j)}\right) \\ &= -\sum_{j=1}^{K} c_j \log\left(\frac{c_j}{n}\right) + \sum_{i=1}^{n} \log(v_{XZ}(x_i,z_i)) \\ &= H_n(X_d,Z_d) + \sum_{i=1}^{n} \log(v_{XZ}(x_i,z_i)) \ , \end{aligned} \tag{6.15}$$

where c_j is the number of data points in B_j and $v_{XZ}(x_i,z_i) = v_X(x_i)v_Z(z_i)$. The remaining entropies can be calculated similarly. Hence, $I^h(X,Y|Z) = H_n(X_d,Z_d) + H_n(Y_d,Z_d) - H_n(X_d,Y_d,Z_d) - H_n(Z_d)$, as the sum of the volume related terms

$$\sum_{i=1}^{n} \log(v_{XZ}(x_i,z_i)) + \sum_{i=1}^{n} \log(v_{YZ}(y_i,z_i)) \tag{6.16}$$

$$-\sum_{i=1}^{n} \log(v_{XYZ}(x_i,y_i,z_i)) - \sum_{i=1}^{n} \log(v_Z(z_i)) \tag{6.17}$$

is equal to zero. □

To proof Theorem 1, we link the above results:

$$\begin{aligned} &\lim_{v'\to 0} \lim_{n\to\infty} I^h(X;Y \mid Z) \\ =& \lim_{v'\to 0} \lim_{n\to\infty} (H^h(X,Z) + H^h(Y,Z) - H^h(X,Y,Z) - H^h(Z)) \\ =& \lim_{v'\to 0} \lim_{n\to\infty} (H_n(X_d,Z_d) + H_n(Y_d,Z_d) - \\ & H_n(X_d,Y_d,Z_d) - H_n(Z_d)) \\ =& \lim_{v'\to 0} (H(X_d,Z_d) + H(Y_d,Z_d) - H(X_d,Y_d,Z_d) - H(Z_d)) \\ =& I(X;Y \mid Z) \ . \end{aligned} \tag{6.18}$$

6.9.2 Implementation Details

As discussed in Section 6.5, our goal is to find that discretization that minimizes the joint entropy over a set of k random variables via an iterative greedy algorithm. We provide the pseudocode in Algorithm 6. As input, we are given a dataset $D = \{D^1, \ldots, D^k\}$ consisting of n rows and k columns, representing a sample of size n from a k-dimensional random vector X, and a user-specified parameter $i_{\max}$ specifying the maximum number of iterations. First, we initialize the discretization X_d (line 1) by creating single bin histograms for the continuous points in D_j and a bin with bin-width 1 per discrete point. To detect the latter, we check if there exist $|\{x \in \mathcal{X}_j \mid D_j = x\}| \geq t$, where t is a user-defined threshold. After that, we iteratively update the discretization for that X_j providing the highest gain in stochastic complexity, until either the score cannot be improved or the maximum number of iterations has been reached (lines 3 – 13). To update the discretization of a variable X_j we call the function refine (line 6), which receives as input the data D_j and the discretization after iteration i. It then re-discretizes X_j using an extension of the dynamic programming algorithm by Kontkanen and Myllymäki (2007b). In essence, instead of simply discretizing X_j independently of the remaining variables, we keep the discretizations for all $X_i \neq X_j$ fixed and find the optimal histogram model M^* over X_j s.t. the overall score $L(D, M)$ is minimized.

6.9.3 Data Generation and Additional Experiments

In the following, we first provide an empirical analysis on how the number of bins depends on the number of samples, then we give the details of the data generation for the experiments carried out on the synthetic causal network and last we provide additional experiments to evaluate $I_{\mathcal{X}^2}$ and I_{SC}.

Sample Size and Number of Bins

As discussed in Section 6.3, an important requirement to ensure consistency is that the number of bins grows as a sub-linear function w.r.t. the number of samples. We demonstrate that MDL-optimal histograms have this desirable property when learned on one-dimensional Gaussian distributions in Figure 6.5: the number of bins K grows with n, but slower than $\sqrt{n}$. In addition, for multi-

Algorithm 6: Discretization of a Mixture Model

Input: Data $D = \{D_1, \ldots, D_k\}$ representing a sample from a k-dimensional random vector X, maximum number of iterations $i_{\max}$

Output: Discretization X_d

```
X_d ← init(D)                          ▷ Initialize the discretization;
i ← 1;
while X_d changes ∧ i ≤ i_max do
    X_d^i ← X_d;
    for j ∈ {1, ..., k} do
        X_d^ij ← refine(D_j | X_d)     ▷ Refine discretization;
        if Score(X_d^ij) < Score(X_d^i) then
            X_d^i ← X_d^ij;
    X_d ← X_d^i;
    i ← i + 1;
return X_d                             ▷ Return final discretization;
```

dimensional data, for which we can only approximate the histogram model that minimizes $L(D, M)$, we observe that if the number of dimensions increases, the average number of bins per dimension decreases if we keep n fixed.

Synthetic Network

Here, we describe the data generation for the synthetic network shown in Figure 6.3. The source nodes of the network are A and B. A is generated as $A \sim \text{Exp}(1)$ and $B \sim \text{Unif}(0, 4)$ (discrete). To get $B \to C$ we generate C as $C \sim \text{Binom}(b, 0.5)$ for $B = b$, for $B \to D$ we sample D as $D \sim N(b - 2, 1)$ for $B = b$ and E is sampled is exponentially distributed with rate $\frac{1}{c+1}$ for $C = c$. F is generated as a function of C and D. First, we generate C' by rounding the values of C and then we write F as $F = D^{\frac{C'}{2}} + N(0, 1)$. Last, we generate G as the zero inflated Poissonization of A. Let $E' = \frac{\text{sign}(E-1)+1}{2}$, which ensures that E' is either zero or one dependent on the value of E. Then $G \sim N(a, 1)$ if $E' = 0$ and $A = a$, and $G \sim \text{Poisson}(a)$ for $A = a$ if $G = 1$.

Figure 6.5: Left: Average number of bins k to discretize $X \sim N(0,1)$ for increasing sample sizes (20 repetitions). Right: Per dimension of a multivariate Gaussian distribution with $X_i \perp\!\!\!\perp X_j$ and $X_i \sim N(0,1)$, we show the average number of bins ($n = 2\,000$, 20 repetitions).

Detecting Collider and Non-Collider Structures

To evaluate how well $I_{\mathcal{X}^2}$ and I_{SC} can identify conditional (in)dependencies, we evaluate both variants on various generating mechanisms that involve collider and non-collider structures. Those structures are at the core of causal discovery, since collider structures can be inferred by detecting conditional dependencies, while non-collider structures impose conditional independencies. As in the causal discovery experiment, we set $\alpha = 0.01$ for $I_{\mathcal{X}^2}$, RCIT and RCoT.

Collider Structures We generate data according to a collider structure, which can be represented by a directed acyclic graph as, e.g., $X \to Z \leftarrow Y$. According to this structure, we model X and Y by some distribution and write Z as a non-deterministic function of X and Y. We generate data for different generating mechanisms, including two continuous and four mixed settings.

1. $X \perp\!\!\!\perp Y$ and X, Y are either drawn from $N(0,1)$ or Uniform$(-2,2)$. Z is an additive function of polynomials up to degree three or the tangent function plus additive noise $N \sim N(0, 0.1)$—e.g. $Z = X^3 + \tan(Y) + N$. We pick the type of the distribution of X, Y, as well as the function type, uniform at random.

2. X, Y are drawn from a standard Gaussian distribution, with $X \perp\!\!\!\perp Y$ and $Z = \text{sign}(X \cdot Y) \cdot \text{Exp}(1/\sqrt{2})$.

3. $X, Y \sim N(0,1)$ with $X \perp\!\!\!\perp Y$ and $Z = \text{sign}(X \cdot Y)$, where we randomly assign a $z \in \mathcal{Z}$ to 10% of the values in Z to make the function non-deterministic.

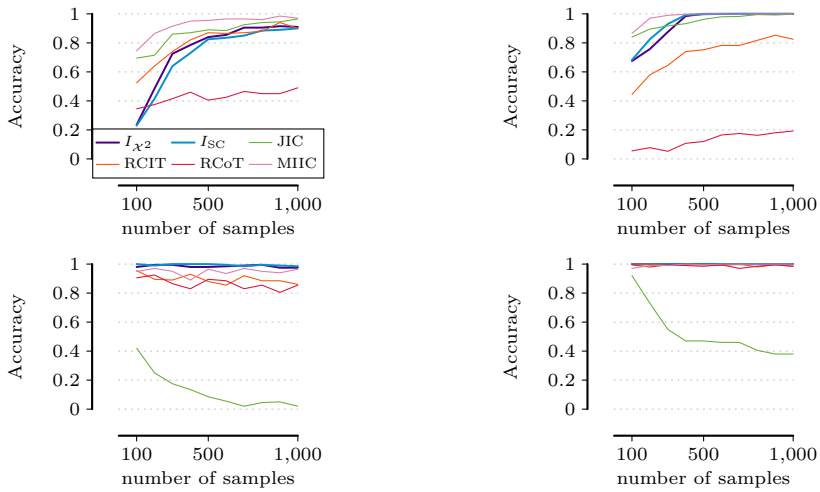

Figure 6.6: Accuracy for detecting continuous (left) and mixed-type (right) dependencies in collider structures (top) and independencies in non-collider structures (bottom) for different sample sizes.

4. $X \sim N(0,1)$, $Y \sim \text{Poisson}(\lambda)$, with parameter λ selected uniformly at random from $\{1,2,3\}$. We generate Z as X modulo Y and assign 10% of the data points randomly.

5. X, Y are unbiased coins. $Z' = X \oplus Y$, where $\oplus$ denotes the xor operator. From Z' we calculate Z as $N(0, 0.1)$ if $Z' = 0$ and $\text{Poisson}(5) \cdot N(0, 0.1)$ under the condition that $Z' = 1$.

6. We generate X, Y and Z' as above, but this time we generate Z as $\text{Poisson}(5) + N(0, 0.1)$ if $Z' = 1$ and as $N(0, 0.1)$ if $Z' = 0$.

For each generating mechanism, including two purely continuous and four mixed mechanisms, we generate 100 data sets and report the averaged results, separately for the continuous and mixed data, in Figure 6.6 (top). On the continuous data, both of our approaches perform on par with RCIT and JIC for more than 400 data points, whereas MIIC has a slightly better performance and RCoT is not able to detect the dependence for the sign function and hence has an accuracy of about 50%. Since the functions for mixed data include an xor and the modulo operator, it is difficult to treat all discrete variables as ordinal and hence RCIT only reaches up to 80% accuracy—which is mostly due to an xor determining the

scaling of a Gaussian distributed variable. On the other hand, both of our tests perform very well and only need 400 samples to obtain an accuracy close to 100%. JIC and MIIC perform on par with our tests.

Non-Collider Structures Similar to collider structures, there also exist non-collider structures of the form $X \to Z \to Y$ or $X \leftarrow Z \to Y$. In both cases, the ground truth is that $X \perp\!\!\!\perp Y \mid Z$. To simulate data according to these graphs, we consider two continuous mechanisms based on polynomial functions and two mixed generating mechanisms.

1. $X \sim N(0, 1)$, Z is an additive noise function of X and Y is an additive noise function of Z. The functions can be polynomials up to degree three or the tangent function.
2. $Z \sim N(0, 1)$, X and Y are independent additive noise functions of Z, as defined above.
3. X, Y and Z are generated as in Experiment IV.
4. X and Y are generated according to Experiment II and $Z \sim N(\mu, x)$ for $X = x$ and $\mu \in [-4, 4]$.

In essence, Figure 6.6 (bottom) shows that both our tests obtain almost perfect accuracies for the continuous and mixed data, whereas RCIT and RCoT fail to detect up to 20% of the independencies for continuous data, MIIC does not detect up to 11% and JIC seems to generally overestimate dependencies for those test cases. If we consider these results in comparison to the results for detecting dependencies for the collider setting, we suspect that both MIIC and JIC have a larger tendency to falsely detect dependencies, while our approach is more conservative and hence needs more samples to detect true dependencies.

Chapter 7

Conclusions

The importance of interpretability is widely accepted in machine learning tasks in which humans are responsible for making a decision for taking action. In such scenarios, it is crucial for domain experts to trust the machine learning model. As a result, research on interpretable machine learning has received a lot of attention in recent years. This dissertation contributes to this area by proposing intrinsically interpretable and transparent methods for supervised and unsupervised tasks, both for predictive modeling and for knowledge discovery.

In this Chapter we provide our conclusions.

7.1 Summary

Truly Unordered Rule Sets. In the field of rule set models, we considered the problem of increasing the interpretability of rule set models by removing the ad-hoc schemes for handling conflicts caused by overlaps of rules, in which an overlap refers to a subset of instances covered by multiple rules simultaneously.

In order to achieve this goal, we first considered allowing overlaps for expressing uncertainty and exception, which eliminated the need for imposing implicit orders among rules. Building upon it, we next formally defined truly unordered rule set (TURS) models, which informally only "allow" rules with similar outputs to overlap. Lastly, we showcased through a case study with patient data collected at Leiden University Medical Center that our TURS model paves the way to interactive rule learning. That is, the rule set model can be automatically updated with feedback from domain experts.

Multi-dimensional MDL-based Histograms. We studied multi-dimensional MDL-based histograms, which can be used as a transparent tool for various tasks in machine learning and data mining, including density estimation, explanatory data analysis, discretization, entropy estimation, and conditional mutual information estimation. With a series of papers, we first extended the one-dimensional MDL-based histogram to the two-dimensional case and showcased its use for analyzing spatial datasets. Secondly, we extended MDL-based histograms for analyzing multi-dimensional and mixed-type datasets (with discrete-continuous mixture variables), specifically for analyzing its dependency structures via conditional mutual information estimation.

7.2 Answers to Research Questions

In the following, we revisit the proposed research questions and provide our answers and a discussion for each of them.

Research Question 1: How can we formalize rule sets as probabilistic models such that rules are independent of each other? Further, how to learn such a model from data?

A set of rules, when put together, can form a global model for the whole dataset. While defining a single rule as a local probabilistic model is straightforward (given that the target variable is discrete for classification tasks), defining a global model for a rule set is far more complicated, mostly due to the nuisance caused by overlaps, i.e., one instance covered by multiple rules at the same time.

To remove implicit and explicit orders among rules, we treated rules equally, i.e., one rule does not have a higher "quality" than the other. We started by considering the informal implication of an overlap of two rules; i.e., what is the *implication* of the overlap in the sense that why the instances covered by the intersection of these two rules cannot form a rule on itself (by concatenating the conditions of the two rules)? This leads to our justification of the overlap: if the class probabilities of the instances covered by the overlap are not very different from those of the instances covered by each single rule respectively, it is not desirable to separate the instances in the overlap to nether of the two rules. Informally, this can be caused by close class probability estimates and/or by a small sample size of the overlap (which leads to a large variance). In this case, we interpret overlaps as "uncertainty", in the sense that we do not have enough data to decide that the instances covered by the overlap "belong" to a single rule.

Thus, our first answer to Research Question 1 is that we treat overlap as uncertainty when formalizing rule sets as probabilistic models. This approach is very different from previous methods, which either minimize the size of overlaps or takes post-hoc conflict resolving schemes.

Further, when regarding an overlap as uncertainty, an overlap of two rules, e.g., rule S_i and rule S_j, can be interpreted as "instances that are covered by the overlap "belong" to rule S_i **or** rule S_j", in which the "or" represents uncertainty. This intuition motivated us to consider taking the union of rules for modeling instances covered by the overlap, which leads to our second answer to the proposed

research question.

Our second answer to Research Question 1 is that we formally define a probabilistic model for any given rule set that may have overlaps, i.e., the truly unordered rule set (TURS) model. The key innovation is to define

$$P(Y = y|X = x) := P(Y|\{X \in \cup S_i\}), \forall x \in \cap S_i.$$

In this way, the likelihood of a TURS model incorporates how different the class probability estimates of rules that form an overlap are. Thus, we now only "allow" overlaps that have similar class probability estimates by penalizing the situation when two rules with very different class probability estimates overlap.

Lastly, as learning a TURS model from data requires taking into consideration modeling overlaps, existing formalizations of the problem of learning rules from data cannot be applied to learning a TURS model. Also, existing rule learning algorithms cannot be used directly or with modification easily.

Therefore, our third answer to Research Question 1 is that we formally defined the problem of learning a TURS model as an MDL-based model selection problem, and we developed a novel heuristic algorithm for finding good models.

Introducing the MDL principle removes the regularization parameter for controlling the model complexity. Setting such regularization parameters in an ad-hoc way reduces the algorithm transparency, while tuning it with cross-validation can be time-consuming. Moreover, our algorithm is equipped with several algorithmic innovations, including 1) taking "patience" into account, 2) using a dual-beam, and 3) using a look-ahead strategy based on a MDL-based local test. Our algorithm is shown to have competitive predictive performance and simple model complexity; further, more importantly, the TURS models learned by our algorithm are shown to be empirically "truly unordered", in the sense that the predictive performance is hardly affected by randomly chosen rules for making predictions for instances covered by overlaps.

Research Question 2: How can we construct parameter-free two-dimensional histograms with transparent and informative patterns (bins)?

Eliminating user-defined parameters for controlling the bin sizes of histograms

increases the transparency of how a histogram model is obtained from data. Hence, the ambiguity to data analysts caused by different histograms showing different data summarization is removed.

In order to remove parameters that control the bin sizes of histograms, we formalize the problem of learning such histogram models as an MDL-based model selection problem. That is, we adopted the MDL principle to define the "optimal" model for such an unsupervised task and formalized the best model as the one that leads to the best compression of data and model together.

In addition, to obtain more interpretable bins (patterns) in the sense that 1) instances within each bin can be considered to have homogeneous density, and 2) neighboring bins have different densities, we proposed a greatly flexible model class that includes any data partition formed by unions of disjoint rectangles. Lastly, we developed an efficient algorithm that combines top-down and bottom-up search, and showcased that the learned two-dimensional histograms carry meaningful patterns that generalize well to unseen data, both on simulated datasets with known ground truth and real-world case study datasets.

Thus, our answer to Research Question 2 is to formalize the problem of learning a two-dimensional histogram based on the MDL principle, and to obtain more informative patterns (bins) by 1) considering dependencies between dimensions and 2) using more flexible data partitions.

Research Question 3: How can we construct a multi-dimensional adaptive histogram-based model for interpretable CMI estimation?

Learning dependency structure via estimating the conditional mutual information (CMI) is a challenging task, especially when the data contains mixed types (discrete, continuous, and possibly also discrete-continuous mixtures).

To construct histograms for mixed type data, we first formalized the problem of estimating CMI for mixed type data. Specifically, we adopted measure-theoretic tools to prove that the CMI for mixed-type data can be written as the sum of four entropy terms, just like the CMI for purely continuous and discrete data.

Further, we proposed an entropy estimator based on multi-dimensional histogram models, and consequently a plug-in estimator for CMI. Next, we formalized the problem of learning a multi-dimensional adaptive histogram as an MDL-based model selection task. Leveraging the MDL principle reduced the

hyper-parameters to be set and hence increased the transparency of how a model is obtained. Lastly, we proposed an alternating algorithm for learning such a multi-dimensional histogram from data and showcased the effectiveness of such an approach by benchmarking against competitor methods in various tasks that involve CMI estimation.

In conclusion, our answer to Research Question 3 is 1) to adopt the MDL principle to formalize the learning problem, 2) to leverage the measure-theoretic definition of entropy for mixed-type of data, and 3) to design an alternating algorithm for learning such a histogram form data.

7.3 Future Work

We conclude this chapter by discussing a few possible future work directions following this dissertation.

First, we consider a crucial problem to formally define *human comprehensibility* as a measure in interpretable machine learning, which characterizes how easy a machine learning model can be comprehended by a human. Notably, the concept of human comprehensibility may be defined both for intrinsically interpretable models and explainable artificial intelligence (XAI) methods that provide post-hoc explanations for black-box models. One key challenge is to properly define the "required level" of human comprehension, which can be different for various machine learning tasks.

Second, it is a fundamental research question to formalize as an optimization problem the task of automatic model updating given human feedback, which is the cornerstone of any interactive machine learning system. One potential approach is to borrow the idea from the *subjective interestingness* in information-theoretic data mining (De Bie 2011a,b). However, subjective interestingness in data mining is, informally, about maximizing the "surprisingness" to the data miner based on their prior "beliefs" about the dataset. Thus, the goal is to search the pattern with the maximum amount of information in the data that the user does not know. In contrast, to formalize automatic human-guided model updating, the goal could be set as searching for a model that maximizes the "trust" from human users. As an example, such a model could be a probabilistic rule set that contains rules that

the user knows or considers trustworthy.

Third, further research about how to develop a flexible interactive data exploration system may be another crucial component for human-in-the-loop machine learning systems. It may be useful for building trust between humans and models if we allow human users to explore subsets of datasets with the help of specific types of machine learning models, including examining the statistical characteristics of (subsets of) datasets.

Bibliography

Abdelhamid, Neda and Fadi Thabtah (2014). "Associative classification approaches: review and comparison". In: *Journal of Information & Knowledge Management* 13.03.

Adebayo, Julius, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim (2018). "Sanity checks for saliency maps". In: *Advances in neural information processing systems* 31.

Angelino, Elaine, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin (2017). "Learning certifiably optimal rule lists for categorical data". In: *arXiv preprint arXiv:1704.01701*.

Antos, András and Ioannis Kontoyiannis (2001). "Convergence properties of functional estimates for discrete distributions". In: *Random Structures & Algorithms* 19.3-4, pp. 163–193.

Bay, Stephen D (2001). "Multivariate discretization for set mining". In: *Knowledge and Information Systems* 3.4, pp. 491–512.

Biba, Marenglen, Floriana Esposito, Stefano Ferilli, Nicola Di Mauro, Teresa Maria Altomare Basile, et al. (2007). "Unsupervised Discretization Using Kernel Density Estimation." In: *International Joint Conference on Artificial Intelligence*. Vol. 7, pp. 696–701.

Boullé, Marc (2004). "Khiops: A statistical discretization method of continuous attributes". In: *Machine learning* 55.1, pp. 53–69.

— (2006). "MODL: a Bayes optimal discretization method for continuous attributes". In: *Machine learning* 65.1, pp. 131–165.

Breiman, Leo, Jerome Friedman, Charles J Stone, and Richard A Olshen (1984). *Classification and regression trees*. CRC press.

Cabeli, Vincent, Louis Verny, Nadir Sella, Guido Uguzzoni, Marc Verny, and Hervé Isambert (2020). "Learning clinical networks from medical records based on information estimates in mixed-type data". In: *PLoS computational biology* 16.5, e1007866.

Caruana, Rich, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad (2015). "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1721–1730.

Chen, Guoqing, Hongyan Liu, Lan Yu, Qiang Wei, and Xing Zhang (2006). "A new approach to classification based on association rule mining". In: *Decision Support Systems* 42.2, pp. 674–689.

Clark, Peter and Robin Boswell (1991). "Rule induction with CN2: Some recent improvements". In: *European Working Session on Learning*. Springer, pp. 151–163.

Clark, Peter and Tim Niblett (1989). "The CN2 induction algorithm". In: *Machine learning* 3.4, pp. 261–283.

Cohen, William W (1995). "Fast effective rule induction". In: *Machine learning proceedings 1995*. Elsevier, pp. 115–123.

Colombo, Diego and Marloes H Maathuis (2012). "A modification of the PC algorithm yielding order-independent skeletons". In: *CoRR, abs/1211.3295*.

Cover, Thomas M and Joy A Thomas (2012). *Elements of information theory*. John Wiley & Sons.

Cuevas, Antonio, Ricardo Fraiman, et al. (1997). "A plug-in approach to support estimation". In: *The Annals of Statistics* 25.6, pp. 2300–2312.

Darbellay, Georges A and Igor Vajda (1999). "Estimation of the information by an adaptive partitioning of the observation space". In: *IEEE Transactions on Information Theory* 45.4, pp. 1315–1321.

Dash, Sanjeeb, Oktay Gunluk, and Dennis Wei (2018). "Boolean Decision Rules via Column Generation". In: *Advances in Neural Information Processing Systems* 31, pp. 4655–4665.

De Bie, Tijl (2011a). "An information theoretic framework for data mining". In: *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 564–572.

— (2011b). “Maximum entropy models and subjective interestingness: an application to tiles in binary databases”. In: *Data Mining and Knowledge Discovery* 23, pp. 407–446.

Demšar, Janez, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan (2013). “Orange: Data Mining Toolbox in Python”. In: *Journal of Machine Learning Research* 14, pp. 2349–2353. URL: `http://jmlr.org/papers/v14/demsar13a.html`.

Doshi-Velez, Finale and Been Kim (2017). “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint arXiv:1702.08608*.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: `http://archive.ics.uci.edu/ml`.

Duong, Tarn and Martin Hazelton (2003). “Plug-in bandwidth matrices for bivariate kernel density estimation”. In: *Journal of Nonparametric Statistics* 15.1, pp. 17–30.

Duong, Tarn et al. (2007). “ks: Kernel density estimation and kernel discriminant analysis for multivariate data in R”. In: *Journal of Statistical Software* 21.7, pp. 1–16.

Fayyad, Usama and Keki Irani (1993). “Multi-interval discretization of continuous-valued attributes for classification learning”. In: *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pp. 1022–1027.

Ferrandiz, Sylvain and Marc Boullé (2005). “Multivariate discretization by recursive supervised bipartition of graph”. In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, pp. 253–264.

Filip, Jiri and Tomas Kliegr (2019). “PyIDS-Python Implementation of Interpretable Decision Sets Algorithm by Lakkaraju et al, 2016.” In: *RuleML+ RR (Supplement)*.

Frank, Eibe and Ian H Witten (1998). “Generating accurate rule sets without global optimization”. In: *Working Paper Series ISSN 1170-487X*.

Frenzel, Stefan and Bernd Pompe (2007). “Partial mutual information for coupling analysis of multivariate time series”. In: *Physical Review Letters* 99.20, p. 204101.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2001). *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York.

Friedman, Jerome H and Nicholas I Fisher (1999). "Bump hunting in high-dimensional data". In: *Statistics and computing* 9.2, pp. 123–143.

Fürnkranz, Johannes and Peter A Flach (2005). "Roc 'n'rule learning—towards a better understanding of covering algorithms". In: *Machine learning* 58.1, pp. 39–77.

Fürnkranz, Johannes, Dragan Gamberger, and Nada Lavrač (2012). *Foundations of rule learning*. Springer Science & Business Media.

Galbrun, Esther (2020). "The minimum description length principle for pattern mining: A survey". In: *arXiv preprint arXiv:2007.14009*.

— (2022). "The minimum description length principle for pattern mining: A survey". In: *Data mining and knowledge discovery* 36.5, pp. 1679–1727.

Gao, Weihao, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath (2017). "Estimating mutual information for discrete-continuous mixtures". In: *Advances in neural information processing systems*, pp. 5986–5997.

Gao, Weihao, Sewoong Oh, and Pramod Viswanath (2016). "Breaking the Bandwidth Barrier: Geometrical Adaptive Entropy Estimation". In: Curran Associates, Inc., pp. 2460–2468.

Gasparini, Mauro (1996). "Bayesian density estimation via Dirichlet density processes". In: *Journal of Nonparametric Statistics* 6.4, pp. 355–366.

Gay, Dominique and Marc Boullé (2012). "A bayesian approach for classification rule mining in quantitative databases". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 243–259.

Grünwald, Peter (2004). "A tutorial introduction to the minimum description length principle". In: *arXiv preprint math/0406077*.

— (2007). *The minimum description length principle*. MIT press.

Grünwald, Peter and Teemu Roos (2019). "Minimum Description Length Revisited". In: *arXiv preprint arXiv:1908.08484*.

Gupta, Ankit, Kishan G Mehrotra, and Chilukuri Mohan (2010). "A clustering-based discretization for supervised learning". In: *Statistics & probability letters* 80.9-10, pp. 816–824.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten (2009). "The WEKA data mining software: an update". In: *ACM SIGKDD explorations newsletter* 11.1, pp. 10–18.

Han, Songqiao, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao (2022). "Adbench: Anomaly detection benchmark". In: *Advances in Neural Information Processing Systems* 35, pp. 32142–32159.

Han, Yanjun, Jiantao Jiao, and Tsachy Weissman (2015). "Adaptive estimation of shannon entropy". In: *IEEE International Symposium on Information Theory*. IEEE, pp. 1372–1376.

Hansen, Mark H and Bin Yu (2001). "Model selection and the principle of minimum description length". In: *Journal of the American Statistical Association* 96.454, pp. 746–774.

Holzinger, Andreas (2016). "Interactive machine learning for health informatics: when do we need the human-in-the-loop?" In: *Brain Informatics* 3.2, pp. 119–131.

Hond, Anne AH de, Ilse MJ Kant, Mattia Fornasa, Giovanni Cinà, Paul WG Elbers, Patrick J Thoral, M Sesmu Arbous, and Ewout W Steyerberg (2023). "Predicting readmission or death after discharge from the ICU: external validation and retraining of a machine learning model". In: *Critical Care Medicine* 51.2, p. 291.

Hornik, Kurt, Christian Buchta, and Achim Zeileis (2009). "Open-Source Machine Learning: R Meets Weka". In: *Computational Statistics* 24.2, pp. 225–232. DOI: 10.1007/s00180-008-0119-7.

Hu, Xiyang, Cynthia Rudin, and Margo Seltzer (2019). "Optimal sparse decision trees". In: *Advances in Neural Information Processing Systems* 32.

Hühn, Jens and Eyke Hüllermeier (2009). "FURIA: an algorithm for unordered fuzzy rule induction". In: *Data Mining and Knowledge Discovery* 19.3, pp. 293–319.

Jin, Ruoming, Yuri Breitbart, and Chibuike Muoh (2009). "Data discretization unification". In: *Knowledge and Information Systems* 19.1, pp. 1–29.

Kahle, David and Hadley Wickham (2013). "ggmap: Spatial Visualization with ggplot2". In: *The R Journal* 5.1, pp. 144–161. URL: https://journal.r-project.org/archive/2013-1/kahle-wickham.pdf.

Kameya, Yoshitaka (2011). "Time Series Discretization via MDL-based Histogram Density Estimation". In: *IEEE 23rd international conference on tools with artificial intelligence*. IEEE, pp. 732–739.

Kang, Ye, Shanshan Wang, Xiaoyan Liu, Hokyin Lai, Huaiqing Wang, and Baiqi Miao (2006). "An ICA-based multivariate discretization algorithm". In: *International Conference on Knowledge Science, Engineering and Management*. Springer, pp. 556–562.

Kerber, Randy (1992). "Chimerge: Discretization of numeric attributes". In: *Proceedings of the tenth national conference on Artificial intelligence*, pp. 123–128.

Klemelä, Jussi (2009). "Multivariate histograms with data-dependent partitions". In: *Statistica sinica*, pp. 159–176.

Kontkanen, Petri and Petri Myllymäki (2007a). "A linear-time algorithm for computing the multinomial stochastic complexity". In: *Information Processing Letters* 103.6, pp. 227–233.

— (2007b). "MDL Histogram Density Estimation". In: *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*. Ed. by Marina Meila and Xiaotong Shen. Vol. 2. Proceedings of Machine Learning Research. PMLR.

Kontkanen, Petri, Petri Myllymäki, Tomi Silander, and Henry Tirri (1997). "A Bayesian approach to discretization". In: *Proceedings of the European symposium on intelligent techniques*.

Kotsiantis, Sotiris and Dimitris Kanellopoulos (2006). "Discretization techniques: A recent survey". In: *GESTS International Transactions on Computer Science and Engineering* 32.1, pp. 47–58.

Kotsiantis, Sotiris B (2013). "Decision trees: a recent overview". In: *Artificial Intelligence Review* 39.4, pp. 261–283.

Kozachenko, L. F. and N. N. Leonenko (1987). "Sample Estimate of the Entropy of a Random Vector". In: *Probl. Peredachi Inf.* 23 (2), pp. 9–16.

Kramer, Andrew A, Thomas L Higgins, and Jack E Zimmerman (2013). "The association between ICU readmission rate and patient outcomes". In: *Critical care medicine* 41.1, pp. 24–33.

Kraskov, Alexander, Harald Stögbauer, and Peter Grassberger (2004). "Estimating mutual information". In: *Physical review E* 69.6, p. 066138.

Kurgan, Lukasz A and Krzysztof J Cios (2004). "CAIM discretization algorithm". In: *IEEE transactions on Knowledge and Data Engineering* 16.2, pp. 145–153.

Kwedlo, Wojciech and Marek Kretowski (1999). "An evolutionary algorithm using multivariate discretization for decision rule induction". In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pp. 392–397.

Lakkaraju, Himabindu, Stephen H Bach, and Jure Leskovec (2016). "Interpretable decision sets: A joint framework for description and prediction". In: *Proceedings of the 22nd ACM SIGKDD*, pp. 1675–1684.

Lee, Jaesung and Dae-Won Kim (2013). "Feature selection for multi-label classification using multivariate mutual information". In: *Pattern Recognition Letters* 34.3, pp. 349–357.

Li, Bo, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou (2023). "Trustworthy AI: From principles to practices". In: *ACM Computing Surveys* 55.9, pp. 1–46.

Li, Wenmin, Jiawei Han, and Jian Pei (2001). "CMAR: Accurate and efficient classification based on multiple class-association rules". In: *Proceedings 2001 IEEE international conference on data mining*. IEEE, pp. 369–376.

Liu, Bing, Wynne Hsu, Yiming Ma, et al. (1998). "Integrating classification and association rule mining." In: *KDD*. Vol. 98, pp. 80–86.

Liu, Linxi and Wing Hung Wong (2014). "Multivariate Density Estimation Based on Adaptive Partitioning: Convergence Rate, Variable Selection and Spatial Adaptation". In: *arXiv preprint arXiv:1401.2597*.

Lu, Luo, Hui Jiang, and Wing H Wong (2013). "Multivariate density estimation by Bayesian sequential partitioning". In: *Journal of the American Statistical Association* 108.504, pp. 1402–1410.

Lud, Marcus-Christopher and Gerhard Widmer (2000). "Relative unsupervised discretization for association rule mining". In: *European conference on principles of data mining and knowledge discovery*. Springer, pp. 148–158.

Mandros, Panagiotis, David Kaltenpoth, Mario Boley, and Jilles Vreeken (2020). "Discovering Functional Dependencies from Mixed-Type Data". In: pp. 1404–1414.

Marx, Alexander and Jilles Vreeken (2019). "Testing Conditional Independence on Discrete Data using Stochastic Complexity". In: *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 496–505.

Marx, Alexander, Lincen Yang, and Matthijs van Leeuwen (2021). "Estimating Conditional Mutual Information for Discrete-Continuous Mixtures using Multi-Dimensional Adaptive Histograms". In: *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, pp. 387–395.

Mehta, Sameep, Srinivasan Parthasarathy, and Hui Yang (2005). "Toward unsupervised correlation preserving discretization". In: *IEEE Transactions on Knowledge and Data Engineering* 17.9, pp. 1174–1185.

Mesner, Octavio César and Cosma Rohilla Shalizi (2020). "Conditional Mutual Information Estimation for Mixed, Discrete and Continuous, Data". In: *IEEE Transactions on Information Theory*.

Molnar, Christoph (2020). *Interpretable machine learning*. Lulu.com.

Mononen, Tommi and Petri Myllymäki (2008). "Computing the Multinomial Stochastic Complexity in Sub-Linear Time". In: *Proceedings of the 4th European Workshop on Probabilistic Graphical Models*, pp. 209–216.

Mosqueira-Rey, Eduardo, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal (2023). "Human-in-the-loop machine learning: A state of the art". In: *Artificial Intelligence Review* 56.4, pp. 3005–3054.

Murdoch, W James, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu (2019). "Interpretable machine learning: definitions, methods, and applications". In: *arXiv preprint arXiv:1901.04592*.

Nguyen, Hoang-Vu, Emmanuel Müller, Jilles Vreeken, and Klemens Böhm (2014). "Unsupervised interaction-preserving discretization of multivariate data". In: *Data Mining and Knowledge Discovery* 28.5-6, pp. 1366–1397.

Paninski, Liam (2003). "Estimation of entropy and mutual information". In: *Neural Computation* 15.6, pp. 1191–1253.

Paninski, Liam and Masanao Yajima (2008). "Undersmoothed kernel entropy estimators". In: *IEEE Transactions on Information Theory* 54.9, pp. 4384–4388.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D.

Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pfahringer, Bernhard (1995). "Compression-based discretization of continuous attributes". In: *Machine Learning Proceedings 1995*. Elsevier, pp. 456–463.

Politis, Dimitris N (1991). "On the entropy of a mixture distribution". In: *Technical Report 91-67, Purdue University. Department of Statistics*.

Proença, Hugo M and Matthijs van Leeuwen (2020). "Interpretable multiclass classification by MDL-based rule lists". In: *Information Sciences* 512, pp. 1372–1393.

Quinlan, J. Ross (1990). "Learning logical definitions from relations". In: *Machine learning* 5, pp. 239–266.

Quinlan, J Ross (2014). *C4. 5: programs for machine learning*. Elsevier.

Rahimzamani, Arman, Himanshu Asnani, Pramod Viswanath, and Sreeram Kannan (2018). "Estimators for multivariate information measures in general probability spaces". In: *Advances in Neural Information Processing Systems*, pp. 8664–8675.

Ram, Parikshit and Alexander G Gray (2011). "Density estimation trees". In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 627–635.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). "Why should i trust you? Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

Rissanen, Jorma (1978). "Modeling by shortest data description". In: *Automatica* 14.5, pp. 465–471.

— (1983). "A universal prior for integers and estimation by minimum description length". In: *The Annals of statistics* 11.2, pp. 416–431.

— (1996). "Fisher information and stochastic complexity". In: *IEEE transactions on information theory* 42.1, pp. 40–47.

Rudin, Cynthia (2019). "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5, pp. 206–215.

Rudin, Cynthia, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong (2022). "Interpretable machine learning: Fundamental principles and 10 grand challenges". In: *Statistic Surveys* 16, pp. 1–85.

Rudin, Walter et al. (1964). *Principles of mathematical analysis*. Vol. 3. McGraw-hill New York.

Schmidberger, Gabi and Eibe Frank (2005). "Unsupervised discretization using tree-based density estimation". In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pp. 240–251.

Scott, David W (2015). *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.

Scricciolo, Catia (2007). "On rates of convergence for Bayesian density estimation". In: *Scandinavian Journal of Statistics* 34.3, pp. 626–642.

Silander, Tomi, Janne Leppä-aho, Elias Jääsaari, and Teemu Roos (2018). "Quotient Normalized Maximum Likelihood Criterion for Learning Bayesian Network Structures". In: *International conference on artificial intelligence and statistics*. Vol. 84. PMLR, pp. 948–957.

Silander, Tomi, Teemu Roos, Petri Kontkanen, and Petri Myllymäki (2008). "Factorized normalized maximum likelihood criterion for learning Bayesian network structures". In: *Proceedings of the 4th European workshop on probabilistic graphical models (PGM-08)*. Citeseer, pp. 257–272.

Spirtes, Peter, Clark N Glymour, Richard Scheines, and David Heckerman (2000). *Causation, prediction, and search*. MIT press.

Strobl, Eric V, Kun Zhang, and Shyam Visweswaran (2019). "Approximate kernel-based conditional independence tests for fast non-parametric causal discovery". In: *Journal of Causal Inference* 7.1.

Suzuki, Joe (2016). "An estimator of mutual information and its application to independence testing". In: *Entropy* 18.4, p. 109.

— (2019). "Mutual Information Estimation: Independence Detection and Consistency". In: *IEEE International Symposium on Information Theory (ISIT)*. IEEE, pp. 2514–2518.

Teso, Stefano and Kristian Kersting (2019). "Explanatory interactive machine learning". In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 239–245.

Valiant, Gregory and Paul Valiant (2011). "Estimating the unseen: an n/log (n)-sample estimator for entropy and support size, shown optimal via new CLTs". In: *International Conference on Neural Information Processing*, pp. 685–694.

Van der Meijden, Siri (2021). "Predicting Intensive Care Unit Readmission: Performance and Explainability of Machine Learning Algorithms". In: *Master's thesis, Leiden University*.

Van Der Pas, Stéphanie and Veronika Rocková (2017). "Bayesian dyadic trees and histograms for regression". In: *arXiv preprint arXiv:1708.00078*.

Van Leeuwen, Matthijs and Arno Knobbe (2012). "Diverse subgroup set discovery". In: *Data Mining and Knowledge Discovery* 25.2, pp. 208–242.

Veloso, Adriano, Wagner Meira, and Mohammed J Zaki (2006). "Lazy associative classification". In: *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, pp. 645–654.

Vinh, Nguyen Xuan, Jeffrey Chan, and James Bailey (2014). "Reconsidering mutual information based feature selection: A statistical significance view". In: *The AAAI Conference on Artificial Intelligence*.

Wang, Tong, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille (2017). "A bayesian framework for learning rule sets for interpretable classification". In: *The Journal of Machine Learning Research* 18.1.

Wang, Zijie J, Alex Kale, Harsha Nori, Peter Stella, Mark E Nunnally, Duen Horng Chau, Mihaela Vorvoreanu, Jennifer Wortman Vaughan, and Rich Caruana (2022). "Interpretability, then what? editing machine learning models to reflect human knowledge and values". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4132–4142.

Weiler, Daniel and Julian Eggert (2007). "Multi-dimensional histogram-based image segmentation". In: Springer, pp. 963–972.

Woldhek, Annemarie L, Saskia Rijkenberg, Rob J Bosman, and Peter Hj Van Der Voort (2017). "Readmission of ICU patients: A quality indicator?" In: *Journal of critical care* 38, pp. 328–334.

Wu, Mike, Michael Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez (2018). "Beyond sparsity: Tree regularization of deep models for interpretability". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1.

Yang, Fan, Kai He, Linxiao Yang, Hongxia Du, Jingbang Yang, Bo Yang, and Liang Sun (2021). "Learning Interpretable Decision Rule Sets: A Submodular Optimization Approach". In: *Advances in Neural Information Processing Systems* 34.

Yang, Hongyu, Cynthia Rudin, and Margo Seltzer (2017). "Scalable Bayesian rule lists". In: *International Conference on Machine Learning*. PMLR, pp. 3921–3930.

Yang, Kun and Wing Hung Wong (2014). "Density estimation via adaptive partition and discrepancy control". In: *arXiv preprint arXiv:1404.1425*.

Yang, Lincen, Mitra Baratchi, and Matthijs van Leeuwen (2023). "Unsupervised discretization by two-dimensional mdl-based histogram". In: *Machine Learning*, pp. 1–35.

Yin, Xiaoxin and Jiawei Han (2003). "CPAR: Classification based on predictive association rules". In: *Proceedings of the 2003 SIAM international conference on data mining*. SIAM, pp. 331–335.

Zhang, Guangyi and Aristides Gionis (2020). "Diverse Rule Sets". In: *Proceedings of the 26th ACM SIGKDD*, pp. 1532–1541.

Zhang, Xiao-Hang, Jun Wu, Ting-Jie Lu, and Yuan Jiang (2007). "A discretization algorithm based on Gini criterion". In: *2007 International Conference on Machine Learning and Cybernetics*. Vol. 5. IEEE, pp. 2557–2561.

Acknowledgements

I would like to express my deepest gratitude to my promoter and daily supervisor Dr. Matthijs van Leeuwen. It has been an incredible journey for me in the past few years, and thank you for all the support, and for always being the most considerate supervisor ever.

I would like to thank my promoter Prof.dr.Aske Plaat. Thank you for all the support, for always giving super swift responses, and for our inspiring scientific discussions in the past four years.

My heartfelt thanks to my parents. Without your support and love, I would never have made it this far.

I would also like to specially thank Prof.dr.Peter Grunwald, for always being approachable and for always willing to help.

Further, sincere acknowledgements to Dr. Furong Ye, Dr. Fuyu Cai, Yiming Dong, Dr. Xinru Li, Kaihua Liu, Dr. Alexander Marx, and Dr. Meng Jiang, for the nice time we spent together, and the most sincere academic, career, and/or life advices you have given.

I am fortunate to have been working with a lot of smart and warm-hearted colleagues. To my EDA colleagues Daniela, Francesco, Hugo, Iris, Manon, Marieke, Saber, Sander, and Shannon, thank you for creating a "research home" for me at university. I would like to also thank Wenjing Chu, Hui Feng, Rachel de Jong, Shuaiqun Pan, Nan Pu, Minkang Tan, Yumeng Wang, Weikang Weng, Ying Yin, Yuxian Zhao, for our enjoyable lunch, coffee, and other relaxing times we have shared.

Further, special thanks to Dr. Hao Wang, Bing Xu and Zhaoxing Wang, for the best kind of friendship you have given, and for all the help and encouragement over the years. To Zhong Li, Jiayang Shi, and Zhao Yang, for the nice working,

exercising, and relaxing time we have had together.

Last, the biggest acknowledgement to my wife Chi Zhang. I feel so lucky to have you always here with me.

Summary

Data is recorded wherever digital systems are. For instance, websites and mobile phone applications record how people interact with them. Sensors measure and record parameters (such as temperature) of the manufacturing process of industrial products. Financial computer software systems record transactions of financial activities. Similarly, healthcare information management systems record conditions of patients.

Such data can be used to reveal information about the physical process that 'generates' them. Research in the field of *data mining* and *machine learning* concerns developing models and algorithms that can extract and leverage information contained in the dataset effectively and efficiently.

For instance, the information revealed from the medical records of a large number of patients in a hospital can be used to understand why a certain condition is very risky for one type of patient but not for the rest. Additionally, monitoring systems can be developed to automatically alert medical staff to dangerous situations for hospitalized patients

This dissertation focuses on developing new methods that can construct *partition-based models* from data. By partitioning a dataset into subsets where each subset is homogeneous from certain perspectives, a partition-based model extracts data regularities, such as patterns indicating which types of patients in hospitals are at risk for certain diseases. It also makes data-driven predictions, such as raising alarms for potentially dangerous situations. Thus, the goal is to develop data mining and machine learning methods that partition the data at hand 'properly'—the concept of *properness* in this dissertation is defined based on an information-theoretic approach.

Specifically, we focus on studying partition-based models for different tasks,

including interpretable multi-class classification, data discretization and summarization, and dependency structure learning. First of all, for interpretable multi-class classification, we develop an interpretable machine learning algorithm that can extract *probabilistic rules* from data. As an example, a probabilistic rule could be *If weather is foggy and flight time is before 9 am, Then the probability of a flight delay is 10%*. Moreover, our developed method is applied to a case study to predict the risk of patients in the ICU of a hospital being readmitted to the ICU after they are discharged, in which we showcase that the model can improve itself by incorporating feedback from medical experts—a pilot study towards human-AI collaboration.

Further, we consider the task of discretization and summarization of two-dimensional datasets, with the potential use case of summarizing destinations of trajectories recorded by GPS devices. The corresponding research question is how to partition datasets that record the locations with a self-adaptive granularity, neither too coarse nor too refined, as the former means a large amount of information is lost while the latter means noise instead of regularities in the data is captured.

Lastly, we study the problem of understanding '(conditional) dependency' structure in data, which is about developing algorithms that can automatically draw conclusions like 'the risk of forest fire is independent of which month it is, conditioned on the temperature and humidity' from data (i.e., once the temperature and humidity are known, the risk of forest fire is the same no matter what month it is). Both theoretical and methodological research is conducted specifically for a challenging data type, i.e., the mixture of discrete and continuous values. For instance, forest fire historic data are often recorded in this type, which contains 'no fire' (a discrete label) and 'area of fires' (a quantitative, continuous value).

To sum up, partition-based models are studied for various tasks, with the goal of making them more interpretable and transparent to the end-user.

Samenvatting

Overal waar digitale systemen zijn worden gegevens vastgelegd. Websites en mobiele applicaties registreren bijvoorbeeld hoe mensen die gebruiken. Sensoren meten en registreren parameters (zoals temperatuur) van het productieproces van industriële producten. Financiële softwaresystemen registreren transacties van financiële activiteiten. Op dezelfde manier registreren zorginformatiebeheersystemen de condities van patiënten.

Dergelijke gegevens kunnen worden gebruikt om informatie over het onderliggende fysieke proces dat deze data 'genereert' bloot te leggen. Onderzoek op het gebied van *data mining* en *machine learning* houdt zich bezig met het ontwikkelen van modellen en algoritmen die effectief en efficiënt informatie uit een dataset kunnen extraheren.

Zo kan informatie die wordt gevonden in de medische dossiers van grote aantallen patiënten in een ziekenhuis gebruikt worden om te begrijpen waarom een bepaalde conditie zeer risicovol is voor één type patiënt maar niet voor de rest. Daarnaast kunnen monitorsystemen ontwikkeld worden om medisch personeel te waarschuwen voor gevaarlijke situaties bij gehospitaliseerde patiënten.

Deze dissertatie richt zich op het ontwikkelen van nieuwe methoden die *partitie-gebaseerde modellen* uit gegevens kunnen construeren. Door een dataset te partitioneren in subgroepen, waar elke subgroep homogeen is vanuit bepaalde perspectieven, extraheert een op partities gebaseerd model patronen in de data, zoals patronen die aangeven welke typen patiënten in ziekenhuizen risico lopen op bepaalde ziektes. Het maakt ook data-gedreven voorspellingen mogelijk, zoals het activeren van alarmen voor potentieel gevaarlijke situaties. Het doel is dus om data mining en machine learning methoden te ontwikkelen die de data 'juist' partitioneren—het concept van *juistheid* in deze dissertatie is gedefinieerd op basis

van een informatie-theoretische benadering.

Specifiek richten we ons op het bestuderen van partitie-gebaseerde modellen voor verschillende taken, inclusief interpreteerbare classificatie met meerdere klassen, discretisatie en samenvatten van data, en het leren van afhankelijkheidsstructuren. Voor interpreteerbare classificatie met meerdere klassen ontwikkelen we een interpreteerbaar machine learning algoritme dat *probabilistische regels* uit gegevens kan extraheren. Een probabilistische regel zou bijvoorbeeld kunnen zijn: *Als het weer mistig is en de vluchttijd is voor 9 uur 's ochtends, dan is de kans op vluchtvertraging 10%*. We passen de door ons ontwikkelde methode toe op een casestudy om het risico van patiënten op de intensive care van een ziekenhuis te voorspellen om opnieuw te worden opgenomen nadat ze zijn ontslagen, waarmee we onder andere laten zien dat het model kan leren van feedback van medische experts—een pilotstudie naar mens-AI samenwerking.

Verder onderzoeken we de taak van discretisatie en samenvatten van tweedimensionale datasets, met als potentieel gebruiksscenario het samenvatten van bestemmingen van trajecten geregistreerd door GPS-apparaten. De onderzoeksvraag hierbij is hoe datasets van geregistreerde locaties te partitioneren met een adaptieve granulariteit. Dit moet noch te grof noch te verfijnd, aangezien het eerste betekent dat een grote hoeveelheid informatie verloren gaat, terwijl het laatste betekent dat ruis in plaats van patronen in de gegevens worden ontdekt.

Tot slot bestuderen we het probleem van het begrijpen van de structuur van (conditionele) afhankelijkheden in gegevens. Dit gaat over het ontwikkelen van algoritmen die op basis van data automatisch conclusies kunnen trekken zoals 'het risico op bosbrand is onafhankelijk van de maand, geconditioneerd op de temperatuur en vochtigheid uit gegevens' (d.w.z., zodra de temperatuur en vochtigheid bekend zijn, is het risico op bosbrand hetzelfde ongeacht de maand van het jaar). We doen zowel theoretisch als methodologisch onderzoek voor een uitdagend gegevenstype, namelijk de mix van discrete en continue waarden. Historische gegevens van bosbranden worden bijvoorbeeld vaak op deze manier geregistreerd, waar de data zowel 'geen brand' (een discreet label) als een oppervlakte van een brand (een kwantitatieve, continue waarde) bevat.

Samengevat, op partitie-gebaseerde modellen voor verschillende taken worden bestudeerd, met als doel ze interpreteerbaarder en transparanter te maken voor de eindgebruiker.

Titles in the SIKS dissertation series since 2016

2016 01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines

02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow

03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support

04 Laurens Rietveld (VUA), Publishing and Consuming Linked Data

05 Evgeny Sherkhonov (UvA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers

06 Michel Wilson (TUD), Robust scheduling in an uncertain environment

07 Jeroen de Man (VUA), Measuring and modeling negative emotions for virtual training

08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data

09 Archana Nottamkandath (VUA), Trusting Crowdsourced Information on Cultural Artefacts

10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms

11 Anne Schuth (UvA), Search Engines that Learn from Their Users

12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems

13 Nana Baah Gyan (VUA), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UvA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VUA), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VUA), Refining Statistical Data on the Web
19 Julia Efremova (TU/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UvA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
22 Grace Lewis (VUA), Software Architecture Strategies for Cyber-Foraging Systems
23 Fei Cai (UvA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VUA), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (TiU), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring

32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations

33 Peter Bloem (UvA), Single Sample Statistics, exercises in learning from just one example

34 Dennis Schunselaar (TU/e), Configurable Process Trees: Elicitation, Analysis, and Enactment

35 Zhaochun Ren (UvA), Monitoring Social Media: Summarization, Classification and Recommendation

36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies

37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry

38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design

39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect

40 Christian Detweiler (TUD), Accounting for Values in Design

41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance

42 Spyros Martzoukos (UvA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora

43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice

44 Thibault Sellam (UvA), Automatic Assistants for Database Exploration

45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control

46 Jorge Gallego Perez (UT), Robots to Make you Happy

47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks

48 Tanja Buttler (TUD), Collecting Lessons Learned

49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis

50 Yan Wang (TiU), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

2017
01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
05 Mahdieh Shadi (UvA), Collaboration Behavior
06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
08 Rob Konijn (VUA), Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TU/e), Robust Process Mining with Guarantees
13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (TiU), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UvA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UvA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)

22 Sara Magliacane (VUA), Logics for causal inference under uncertainty
23 David Graus (UvA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VUA), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors
28 John Klein (VUA), Architecture Practices for Complex Contexts
29 Adel Alhuraibi (TiU), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"
30 Wilma Latuny (TiU), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
35 Martine de Vos (VUA), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TU/e), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR

40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems

41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle

42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets

43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval

44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering

45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement

46 Jan Schneider (OU), Sensor-based Learning Support

47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration

48 Angel Suarez (OU), Collaborative inquiry-based learning

2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations

02 Felix Mannhardt (TU/e), Multi-perspective Process Mining

03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction

04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks

05 Hugo Huurdeman (UvA), Supporting the Complex Dynamics of the Information Seeking Process

06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems

07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems

08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems

09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations

10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology

11 Mahdi Sargolzaei (UvA), Enabling Framework for Service-oriented Collaborative Networks
12 Xixi Lu (TU/e), Using behavioral context in process mining
13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
14 Bart Joosten (TiU), Detecting Social Signals with Spatiotemporal Gabor Filters
15 Naser Davarzani (UM), Biomarker discovery in heart failure
16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
17 Jianpeng Zhang (TU/e), On Graph Sample Clustering
18 Henriette Nakad (UL), De Notaris en Private Rechtspraak
19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
20 Manxia Liu (RUN), Time and Bayesian Networks
21 Aad Slootmaker (OU), EMERGO: a generic platform for authoring and playing scenario-based serious games
22 Eric Fernandes de Mello Araújo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
26 Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
27 Maikel Leemans (TU/e), Hierarchical Process Mining for Scalable Software Analysis
28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
29 Yu Gu (TiU), Emotion Recognition from Mandarin Speech
30 Wouter Beek (VUA), The ”K” in ”semantic web” stands for ”knowledge”: scaling semantics to the web

2019 01 Rob van Eijk (UL),Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification

02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
03 Eduardo Gonzalez Lopez de Murillas (TU/e), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
05 Sebastiaan van Zelst (TU/e), Process Mining with Streaming Data
06 Chris Dijkshoorn (VUA), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
07 Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
09 Fahimeh Alizadeh Moghaddam (UvA), Self-adaptation for energy efficiency in software systems
10 Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
11 Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
12 Jacqueline Heinerman (VUA), Better Together
13 Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
14 Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
15 Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
16 Guangming Li (TU/e), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
17 Ali Hurriyetoglu (RUN),Extracting actionable information from microtexts
18 Gerard Wagenaar (UU), Artefacts in Agile Team Communication
19 Vincent Koeman (TUD), Tools for Developing Cognitive Agents

20 Chide Groenouwe (UU), Fostering technically augmented human collective intelligence
21 Cong Liu (TU/e), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
22 Martin van den Berg (VUA),Improving IT Decisions with Enterprise Architecture
23 Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
24 Anca Dumitrache (VUA), Truth in Disagreement - Crowdsourcing Labeled Data for Natural Language Processing
25 Emiel van Miltenburg (VUA), Pragmatic factors in (automatic) image description
26 Prince Singh (UT), An Integration Platform for Synchromodal Transport
27 Alessandra Antonaci (OU), The Gamification Design Process applied to (Massive) Open Online Courses
28 Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations
29 Daniel Formolo (VUA), Using virtual agents for simulation and training of social skills in safety-critical circumstances
30 Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems
31 Milan Jelisavcic (VUA), Alive and Kicking: Baby Steps in Robotics
32 Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games
33 Anil Yaman (TU/e), Evolution of Biologically Inspired Learning in Artificial Neural Networks
34 Negar Ahmadi (TU/e), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES
35 Lisa Facey-Shaw (OU), Gamification with digital badges in learning programming
36 Kevin Ackermans (OU), Designing Video-Enhanced Rubrics to Master Complex Skills
37 Jian Fang (TUD), Database Acceleration on FPGAs

	38	Akos Kadar (OU), Learning visually grounded and multilingual representations
2020	01	Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour
	02	Marcos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models
	03	Mostafa Deghani (UvA), Learning with Imperfect Supervision for Language Understanding
	04	Maarten van Gompel (RUN), Context as Linguistic Bridges
	05	Yulong Pei (TU/e), On local and global structure mining
	06	Preethu Rose Anish (UT), Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support
	07	Wim van der Vegt (OU), Towards a software architecture for reusable game components
	08	Ali Mirsoleimani (UL),Structured Parallel Programming for Monte Carlo Tree Search
	09	Myriam Traub (UU), Measuring Tool Bias and Improving Data Quality for Digital Humanities Research
	10	Alifah Syamsiyah (TU/e), In-database Preprocessing for Process Mining
	11	Sepideh Mesbah (TUD), Semantic-Enhanced Training Data AugmentationMethods for Long-Tail Entity Recognition Models
	12	Ward van Breda (VUA), Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment
	13	Marco Virgolin (CWI), Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming
	14	Mark Raasveldt (CWI/UL), Integrating Analytics with Relational Databases
	15	Konstantinos Georgiadis (OU), Smart CAT: Machine Learning for Configurable Assessments in Serious Games
	16	Ilona Wilmont (RUN), Cognitive Aspects of Conceptual Modelling
	17	Daniele Di Mitri (OU), The Multimodal Tutor: Adaptive Feedback from Multimodal Experiences

18 Georgios Methenitis (TUD), Agent Interactions & Mechanisms in Markets with Uncertainties: Electricity Markets in Renewable Energy Systems

19 Guido van Capelleveen (UT), Industrial Symbiosis Recommender Systems

20 Albert Hankel (VUA), Embedding Green ICT Maturity in Organisations

21 Karine da Silva Miras de Araujo (VUA), Where is the robot?: Life as it could be

22 Maryam Masoud Khamis (RUN), Understanding complex systems implementation through a modeling approach: the case of e-government in Zanzibar

23 Rianne Conijn (UT), The Keys to Writing: A writing analytics approach to studying writing processes using keystroke logging

24 Lenin da Nóbrega Medeiros (VUA/RUN), How are you feeling, human? Towards emotionally supportive chatbots

25 Xin Du (TU/e), The Uncertainty in Exceptional Model Mining

26 Krzysztof Leszek Sadowski (UU), GAMBIT: Genetic Algorithm for Model-Based mixed-Integer opTimization

27 Ekaterina Muravyeva (TUD), Personal data and informed consent in an educational context

28 Bibeg Limbu (TUD), Multimodal interaction for deliberate practice: Training complex skills with augmented reality

29 Ioan Gabriel Bucur (RUN), Being Bayesian about Causal Inference

30 Bob Zadok Blok (UL), Creatief, Creatiever, Creatiefst

31 Gongjin Lan (VUA), Learning better – From Baby to Better

32 Jason Rhuggenaath (TU/e), Revenue management in online markets: pricing and online advertising

33 Rick Gilsing (TU/e), Supporting service-dominant business model evaluation in the context of business model innovation

34 Anna Bon (UM), Intervention or Collaboration? Redesigning Information and Communication Technologies for Development

35 Siamak Farshidi (UU), Multi-Criteria Decision-Making in Software Production

2021 01 Francisco Xavier Dos Santos Fonseca (TUD),Location-based Games for Social Interaction in Public Space

02 Rijk Mercuur (TUD), Simulating Human Routines: Integrating Social Practice Theory in Agent-Based Models

03 Seyyed Hadi Hashemi (UvA), Modeling Users Interacting with Smart Devices

04 Ioana Jivet (OU), The Dashboard That Loved Me: Designing adaptive learning analytics for self-regulated learning

05 Davide Dell'Anna (UU), Data-Driven Supervision of Autonomous Systems

06 Daniel Davison (UT), "Hey robot, what do you think?" How children learn with a social robot

07 Armel Lefebvre (UU), Research data management for open science

08 Nardie Fanchamps (OU), The Influence of Sense-Reason-Act Programming on Computational Thinking

09 Cristina Zaga (UT), The Design of Robothings. Non-Anthropomorphic and Non-Verbal Robots to Promote Children's Collaboration Through Play

10 Quinten Meertens (UvA), Misclassification Bias in Statistical Learning

11 Anne van Rossum (UL), Nonparametric Bayesian Methods in Robotic Vision

12 Lei Pi (UL), External Knowledge Absorption in Chinese SMEs

13 Bob R. Schadenberg (UT), Robots for Autistic Children: Understanding and Facilitating Predictability for Engagement in Learning

14 Negin Samaeemofrad (UL), Business Incubators: The Impact of Their Support

15 Onat Ege Adali (TU/e), Transformation of Value Propositions into Resource Re-Configurations through the Business Services Paradigm

16 Esam A. H. Ghaleb (UM), Bimodal emotion recognition from audio-visual cues

17 Dario Dotti (UM), Human Behavior Understanding from motion and bodily cues using deep neural networks

18 Remi Wieten (UU), Bridging the Gap Between Informal Sense-Making Tools and Formal Systems - Facilitating the Construction of Bayesian Networks and Argumentation Frameworks

19 Roberto Verdecchia (VUA), Architectural Technical Debt: Identification and Management

20 Masoud Mansoury (TU/e), Understanding and Mitigating Multi-Sided Exposure Bias in Recommender Systems

21 Pedro Thiago Timbó Holanda (CWI), Progressive Indexes

22 Sihang Qiu (TUD), Conversational Crowdsourcing

23 Hugo Manuel Proença (UL), Robust rules for prediction and description

24 Kaijie Zhu (TU/e), On Efficient Temporal Subgraph Query Processing

25 Eoin Martino Grua (VUA), The Future of E-Health is Mobile: Combining AI and Self-Adaptation to Create Adaptive E-Health Mobile Applications

26 Benno Kruit (CWI/VUA), Reading the Grid: Extending Knowledge Bases from Human-readable Tables

27 Jelte van Waterschoot (UT), Personalized and Personal Conversations: Designing Agents Who Want to Connect With You

28 Christoph Selig (UL), Understanding the Heterogeneity of Corporate Entrepreneurship Programs

---

2022 01 Judith van Stegeren (UT), Flavor text generation for role-playing video games

02 Paulo da Costa (TU/e), Data-driven Prognostics and Logistics Optimisation: A Deep Learning Journey

03 Ali el Hassouni (VUA), A Model A Day Keeps The Doctor Away: Reinforcement Learning For Personalized Healthcare

04 Ünal Aksu (UU), A Cross-Organizational Process Mining Framework

05 Shiwei Liu (TU/e), Sparse Neural Network Training with In-Time Over-Parameterization

06 Reza Refaei Afshar (TU/e), Machine Learning for Ad Publishers in Real Time Bidding

07 Sambit Praharaj (OU), Measuring the Unmeasurable? Towards Automatic Co-located Collaboration Analytics

08 Maikel L. van Eck (TU/e), Process Mining for Smart Product Design

09 Oana Andreea Inel (VUA), Understanding Events: A Diversity-driven Human-Machine Approach
10 Felipe Moraes Gomes (TUD), Examining the Effectiveness of Collaborative Search Engines
11 Mirjam de Haas (UT), Staying engaged in child-robot interaction, a quantitative approach to studying preschoolers' engagement with robots and tasks during second-language tutoring
12 Guanyi Chen (UU), Computational Generation of Chinese Noun Phrases
13 Xander Wilcke (VUA), Machine Learning on Multimodal Knowledge Graphs: Opportunities, Challenges, and Methods for Learning on Real-World Heterogeneous and Spatially-Oriented Knowledge
14 Michiel Overeem (UU), Evolution of Low-Code Platforms
15 Jelmer Jan Koorn (UU), Work in Process: Unearthing Meaning using Process Mining
16 Pieter Gijsbers (TU/e), Systems for AutoML Research
17 Laura van der Lubbe (VUA), Empowering vulnerable people with serious games and gamification
18 Paris Mavromoustakos Blom (TiU), Player Affect Modelling and Video Game Personalisation
19 Bilge Yigit Ozkan (UU), Cybersecurity Maturity Assessment and Standardisation
20 Fakhra Jabeen (VUA), Dark Side of the Digital Media - Computational Analysis of Negative Human Behaviors on Social Media
21 Seethu Mariyam Christopher (UM), Intelligent Toys for Physical and Cognitive Assessments
22 Alexandra Sierra Rativa (TiU), Virtual Character Design and its potential to foster Empathy, Immersion, and Collaboration Skills in Video Games and Virtual Reality Simulations
23 Ilir Kola (TUD), Enabling Social Situation Awareness in Support Agents
24 Samaneh Heidari (UU), Agents with Social Norms and Values - A framework for agent based social simulations with social norms and personal values

25 Anna L.D. Latour (UL), Optimal decision-making under constraints and uncertainty

26 Anne Dirkson (UL), Knowledge Discovery from Patient Forums: Gaining novel medical insights from patient experiences

27 Christos Athanasiadis (UM), Emotion-aware cross-modal domain adaptation in video sequences

28 Onuralp Ulusoy (UU), Privacy in Collaborative Systems

29 Jan Kolkmeier (UT), From Head Transform to Mind Transplant: Social Interactions in Mixed Reality

30 Dean De Leo (CWI), Analysis of Dynamic Graphs on Sparse Arrays

31 Konstantinos Traganos (TU/e), Tackling Complexity in Smart Manufacturing with Advanced Manufacturing Process Management

32 Cezara Pastrav (UU), Social simulation for socio-ecological systems

33 Brinn Hekkelman (CWI/TUD), Fair Mechanisms for Smart Grid Congestion Management

34 Nimat Ullah (VUA), Mind Your Behaviour: Computational Modelling of Emotion & Desire Regulation for Behaviour Change

35 Mike E.U. Ligthart (VUA), Shaping the Child-Robot Relationship: Interaction Design Patterns for a Sustainable Interaction

---

2023 01 Bojan Simoski (VUA), Untangling the Puzzle of Digital Health Interventions

02 Mariana Rachel Dias da Silva (TiU), Grounded or in flight? What our bodies can tell us about the whereabouts of our thoughts

03 Shabnam Najafian (TUD), User Modeling for Privacy-preserving Explanations in Group Recommendations

04 Gineke Wiggers (UL), The Relevance of Impact: bibliometric-enhanced legal information retrieval

05 Anton Bouter (CWI), Optimal Mixing Evolutionary Algorithms for Large-Scale Real-Valued Optimization, Including Real-World Medical Applications

06 António Pereira Barata (UL), Reliable and Fair Machine Learning for Risk Assessment

07 Tianjin Huang (TU/e), The Roles of Adversarial Examples on Trustworthiness of Deep Learning

08 Lu Yin (TU/e), Knowledge Elicitation using Psychometric Learning
09 Xu Wang (VUA), Scientific Dataset Recommendation with Semantic Techniques
10 Dennis J.N.J. Soemers (UM), Learning State-Action Features for General Game Playing
11 Fawad Taj (VUA), Towards Motivating Machines: Computational Modeling of the Mechanism of Actions for Effective Digital Health Behavior Change Applications
12 Tessel Bogaard (VUA), Using Metadata to Understand Search Behavior in Digital Libraries
13 Injy Sarhan (UU), Open Information Extraction for Knowledge Representation
14 Selma Čaušević (TUD), Energy resilience through self-organization
15 Alvaro Henrique Chaim Correia (TU/e), Insights on Learning Tractable Probabilistic Graphical Models
16 Peter Blomsma (TiU), Building Embodied Conversational Agents: Observations on human nonverbal behaviour as a resource for the development of artificial characters
17 Meike Nauta (UT), Explainable AI and Interpretable Computer Vision –From Oversight to Insight
18 Gustavo Penha (TUD), Designing and Diagnosing Models for Conversational Search and Recommendation
19 George Aalbers (TiU), Digital Traces of the Mind: Using Smartphones to Capture Signals of Well-Being in Individuals
20 Arkadiy Dushatskiy (TUD), Expensive Optimization with Model-Based Evolutionary Algorithms applied to Medical Image Segmentation using Deep Learning
21 Gerrit Jan de Bruin (UL), Network Analysis Methods for Smart Inspection in the Transport Domain
22 Alireza Shojaifar (UU), Volitional Cybersecurity
23 Theo Theunissen (UU), Documentation in Continuous Software Development
24 Agathe Balayn (TUD), Practices Towards Hazardous Failure Diagnosis in Machine Learning

25 Jurian Baas (UU), Entity Resolution on Historical Knowledge Graphs
26 Loek Tonnaer (TU/e), Linearly Symmetry-Based Disentangled Representations and their Out-of-Distribution Behaviour
27 Ghada Sokar (TU/e), Learning Continually Under Changing Data Distributions
28 Floris den Hengst (VUA), Learning to Behave: Reinforcement Learning in Human Contexts
29 Tim Draws (TUD), Understanding Viewpoint Biases in Web Search Results

---

2024 01 Daphne Miedema (TU/e), On Learning SQL: Disentangling concepts in data systems education
02 Emile van Krieken (VUA), Optimisation in Neurosymbolic Learning Systems
03 Feri Wijayanto (RUN), Automated Model Selection for Rasch and Mediation Analysis
04 Mike Huisman (UL), Understanding Deep Meta-Learning
05 Yiyong Gou (UM), Aerial Robotic Operations: Multi-environment Cooperative Inspection & Construction Crack Autonomous Repair
06 Azqa Nadeem (TUD), Understanding Adversary Behavior via XAI: Leveraging Sequence Clustering to Extract Threat Intelligence
07 Parisa Shayan (TiU), Modeling User Behavior in Learning Management Systems
08 Xin Zhou (UvA), From Empowering to Motivating: Enhancing Policy Enforcement through Process Design and Incentive Implementation
09 Giso Dal (UT), Probabilistic Inference Using Partitioned Bayesian Networks
10 Cristina-Iulia Bucur (VUA), Linkflows: Towards Genuine Semantic Publishing in Science
11 withdrawn
12 Peide Zhu (TUD), Towards Robust Automatic Question Generation For Learning
13 Enrico Liscio (TUD), Context-Specific Value Inference via Hybrid Intelligence

14 Larissa Capobianco Shimomura (TU/e), On Graph Generating Dependencies and their Applications in Data Profiling
15 Ting Liu (VUA), A Gut Feeling: Biomedical Knowledge Graphs for Interrelating the Gut Microbiome and Mental Health
16 Arthur Barbosa Câmara (TUD), Designing Search-as-Learning Systems
17 Razieh Alidoosti (VUA), Ethics-aware Software Architecture Design
18 Laurens Stoop (UU), Data Driven Understanding of Energy-Meteorological Variability and its Impact on Energy System Operations
19 Azadeh Mozafari Mehr (TU/e), Multi-perspective Conformance Checking: Identifying and Understanding Patterns of Anomalous Behavior
20 Ritsart Anne Plantenga (UL), Omgang met Regels
21 Federica Vinella (UU), Crowdsourcing User-Centered Teams
22 Zeynep Ozturk Yurt (TU/e), Beyond Routine: Extending BPM for Knowledge-Intensive Processes with Controllable Dynamic Contexts
23 Jie Luo (VUA), Lamarck's Revenge: Inheritance of Learned Traits Improves Robot Evolution
24 Nirmal Roy (TUD), Exploring the effects of interactive interfaces on user search behaviour
25 Alisa Rieger (TUD), Striving for Responsible Opinion Formation in Web Search on Debated Topics
26 Tim Gubner (CWI), Adaptively Generating Heterogeneous Execution Strategies using the VOILA Framework
27 Lincen Yang (UL), Information-theoretic Partition-based Models for Interpretable Machine Learning
28 Leon Helwerda (UL), Grip on Software: Understanding development progress of Scrum sprints and backlogs
29 David Wilson Romero Guzman (VUA), The Good, the Efficient and the Inductive Biases: Exploring Efficiency in Deep Learning Through the Use of Inductive Biases
30 Vijanti Ramautar (UU), Model-Driven Sustainability Accounting

Curriculum Vitae

Yang, Lincen was born in Zigong, China, in 1993. Lincen studied and obtained his Bachelor's degree in Statistics from Sichuan University in 2016, during which he was also a visiting exchange student at the Department of Statistics in University of Washington from September 2014 to June 2015. Lincen then obtained his Master's degree in Statistics from Leiden University in 2018. After a short work experience in Finance, he started as a PhD candidate in computer science at Leiden University in 2019.

Lincen's research interests include interpretable machine learning, probabilistic models, and the minimum description length principle.