ATLANTIS PRESS
Atlantis Studies in Uncertainty Modelling, volume 3
Joint Proceedings of the 19th World Congress of the International Fuzzy Systems Association (IFSA), the 12th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT), and the 11th International Summer School on Aggregation Operators (AGOP)

Discovery of Pairwise Ordinal Edit Rules

*Milan Peelman[a] and Antoon Bronselaer[b] and Guy De Tré[c]
[a] Department of Telecommunications and Information Processing,
Ghent University, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium, milan.peelman@ugent.be
[b] Department of Telecommunications and Information Processing,
Ghent University, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium, antoon.bronselaer@ugent.be
[c] Department of Telecommunications and Information Processing,
Ghent University, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium, guy.detre@ugent.be

Abstract

Edit rules are simple tuple-level constraints that concisely model which tuples are not permitted in a consistent relation. Previously, developed algorithms mostly assumed nominal data. This implies that ordinal data either had to be discarded or discretized according to expert knowledge. We can omit this by working with the ordinal data directly. In this paper we explore the discovery of low lift pairwise ordinal edit rules and propose an efficient algorithm employing several pruning strategies derived from the lift measure. Our experiments show that we can obtain a similar precision as nominal algorithms, while having an acceptable computational cost.

Keywords: Data Quality, Edit rules, Ordinal data

1 Introduction

Data quality is becoming an increasingly important part of data management. Although there are several dimensions to data quality, such as accuracy, completeness and currency, a significant part of research focuses on ensuring *consistency*. In particular, rule-based approaches towards safeguarding consistency have been investigated deeply. A milestone in this research is the seminal work by Fellegi and Holt, in which they provided a framework of so-called *edit rules* [7]. Edit rules are simple tuple-level constraints that concisely model which tuples are not permitted in a consistent relation. In that regard, they are much simpler than, for example, (conditional) functional dependencies or (conditional) inclusion dependencies. In the nominal case, a concise representation is usually obtained by means of cross products of subsets of the attribute domains. The main result about this type of edit rules is that for an inconsistent tuple (i.e., a tuple that fails some rules) one can easily produce a minimal set of attributes that can be changed in order to make the tuple consistent. In the very essence, this method relies on Fourier-Motzkin elimination for additive data and a variant thereof for nominal data [5].

An interesting, yet surprising fact is that edit rules for data measured on a scale beyond the nominal level are assumed to be linear equations or inequalities. Clearly, this hinges on two implicit assumptions, the first being that data are additive and the second being that interactions between variables are linear. Of course, edit rules in these forms have many applications. For example, in business surveys, the assumption of linear interactions often holds [5]. It is also necessary to note that other forms of edit rules, such as ratio edits, have been studied [5]. Nevertheless, the case of ordinal data has never received much attention in this framework. In this paper, we aim to fill that void and provide an in-depth study of edit rules in a pure ordinal setting. The contributions of this paper can be summarized as follows.

- First and foremost, we provide a simple definition of edit rules in the ordinal setting and we investigate the properties of such edit rules.
- Second, we show that edit rules for ordinal data naturally occur in many settings. We are able to draw connections between ordinal edit rules and solutions from outlier detection, machine learning and multi-criteria decision making.
- Third, we present a simple, yet effective approach for finding errors in data where ordinal edit rules are the underlying data quality formalism.

The remainder of this paper is structured as follows. In Section 2 we provide preliminary concepts regarding nominal edit rules. This is followed by Section 3, in

which we focus on edit rules in the ordinal data setting. In Section 4 we introduce the lift measure and we explain how we use this measure to create an edit rule discovery algorithm employing several pruning strategies. Some related work is discussed in Section 5. In Section 6 we report the precision score of the discovered edit rules and we provide an analysis of the runtime of the algorithm. Lastly, in Section 7 we give some concluding remarks and propose a few future research directions.

2 Preliminaries

Before we introduce ordinal edit rules, we first recall some basic definitions regarding the relational model and edit rules in the nominal case. Let $\mathscr{A}$ be a countable set of attributes where A represents the domain of an attribute $a \in \mathscr{A}$. A *schema* $\mathscr{R} = \{a_1, \ldots, a_k\}$ is defined by a non-empty and finite subset of $\mathscr{A}$. A *relation* R over $\mathscr{R}$ is defined by a finite set $R \subseteq A_1 \times \ldots \times A_k$. Each element of a relation R with schema $\mathscr{R}$ is called a *tuple* r over $\mathscr{R}$.

For a relation R with schema $\mathscr{R}$ and $X \subseteq \mathscr{R}$, the *projection* of R over X is defined by a relation $R[X]$ with schema X that is obtained by taking the tuples in R, retaining only the values of attributes in X and removing all duplicate tuples. The projection of a single tuple is denoted by $r[X]$ and if $X = \{a\}$, we write $r[a]$. The *selection* over a relation R by a predicate P is defined by $R_P = \{r \mid r \in R \wedge P(r)\}$.

The theoretical framework for edit rules was initially presented in [7]. In general terms, edit rules are tuple-level constraints that provide a concise representation of those tuples that are not permitted in a consistent relation R. In the case of *nominal* data, all domains A_i are assumed to be finite sets and an *edit rule* on $\mathscr{R}$ is defined as follows.

Definition 1 (Nominal Edit rule)**.** *A nominal edit rule E on $\mathscr{R} = \{a_1, \ldots, a_k\}$ with finite attribute domains is an expression of the form $E_1 \times \ldots \times E_k$ where $E_i \subseteq A_i$. A tuple r is said to* satisfy *E (denoted by $r \models E$) if $r \notin E$ and to* fail *E (denoted by $r \not\models E$) if $r \in E$. The set of all edit rules on a schema $\mathscr{R}$ will be denoted by $\mathbb{E}$.*

For a nominal edit rule E, there are two extremes. First, if there exists some $a_i \in \mathscr{R}$ such that $E_i = \emptyset$, then E is always satisfied. In other words, the statement $r \models E$ is a logical tautology. Second, if all $a_i \in \mathscr{R}$ satisfy $E_i = A_i$, then E is never satisfied. In other words, the statement $r \models E$ is a logical contradiction. In the following, unless explicitly stated otherwise, we assume that rules are neither tautologies nor contradictions.

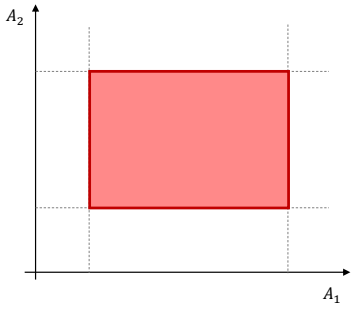

Figure 1: A simple example constraint that cannot be written as a linear edit rule. The red area models non-permitted values.

3 Edit rules for ordinal data

3.1 A justification for ordinal edit rules

In Section 2, we have introduced edit rules in the case of nominal data. For other types of data, edit rules have been proposed as well. For example, if data are measured on a ratio scale, edit rules are often defined by means of a linear equality or linear inequality. An edit rule on $\{a_1, \ldots, a_k\}$ states that the linear combination $b_0 + b_1 \cdot a_1 + \ldots + b_k \cdot a_k$ should be greater than, lower than or equal to zero. The corresponding rules are called *linear edit rules*. Many concepts introduced for nominal edit rules, such as attribute involvement, tautologies and contradictions can then be transferred to linear edit rules quite straightforwardly. Rule implication is less obvious to transfer, but comes down to variable substitution in the case of equality and Fourier-Motzkin elimination in the case of inequality [5]. The concept of linear edit rules however comes with some issues.

A first problem is that rule implication becomes complicated when a mixture of nominal and linear edit rules is used [5]. The composition of sufficient sets then becomes extremely difficult. In practice, sufficient sets are not used in that setting and one resorts to branch-and-bound methods that rely on 'local' rule implication for each erroneous tuple independently. These methods can however have scaling issues when the number of rules or errors within one tuple get large. A second problem (actually, a problem by design) is that linear edit rules rely on the prerequisite that (non)-permitted combinations of values can be expressed by means of a linear connection. We argue that there are simple, yet non-linear connections that are impossible to model by this constraint. For example, the constraint on two domains A_1 and A_2 as shown in Figure 1 cannot be written as a linear edit rule. We will show that these issues are mitigated by introducing ordinal edit rules.

3.2 Intervals on totally ordered sets

As we will see in the following, the notion of ordinal edit rules is built on the concept of intervals. In mathematics, intervals are usually defined as convex subsets of the real numbers, but in this paper, we want to use the concept of an interval on countable sets as well. For that reason, we first provide some notations and definitions of this more general notion of intervals. We consider attributes a_i where the domain A_i is equipped with a *total* order $\leq_i$. In case i is understood, we denote $\leq_i$ simply by $\leq$. The set A together with the total order $\leq$ is often denoted by $(A, \leq)$ and is called a *chain*.

Definition 2. *An interval I on a chain $(A, \leq)$ is a subset of A such that:* $\forall x, y \in I : \forall z \in A : (x \leq z \leq y \implies z \in I)$

We assume all A_i to be finite. If A_i is finite, it always contains unique minimal and maximal elements which we denote respectively by $\underline{a}_i$ and $\overline{a}_i$.

3.3 Ordinal edit rules

With the definitions and notation set, we provide the following definition of an *ordinal edit rule*.

Definition 3 (Ordinal Edit rule)**.** *An ordinal edit rule E on $\mathscr{R} = \{a_1, ..., a_k\}$ with totally ordered attribute domains is an expression of the form $I_1 \times ... \times I_k$ where I_i is an interval on the chain $(A_i, \leq_i)$. A tuple r is said to* satisfy *E (denoted by $r \models E$) if $r \notin E$ and to* fail *E (denoted by $r \not\models E$) if $r \in E$.*

It can be seen that the definition of ordinal edit rules is closely related to Definition 1. The only difference is that sets are replaced by intervals. From this point of view, ordinal edit rules are a natural extension of nominal edit rules to the ordinal case. Indeed, intervals are sets that can be concisely described in terms of a lower and upper bound. However, not every subset from A_i can be written as an interval. In the strict sense, this is not an obstacle with respect to expressiveness as it trivially holds that any subset of $A_1 \times \ldots \times A_k$ can be expressed by a set of ordinal edit rules. In other words, any set of forbidden value combinations has some representation in terms of a set of edit rules. Of course, some representations will contain many edit rules and if we extend our reasoning to the case of infinite domains, then some representations would contain infinitely many edit rules. In practice, however, the restriction to intervals poses no great limitations on the formalism.

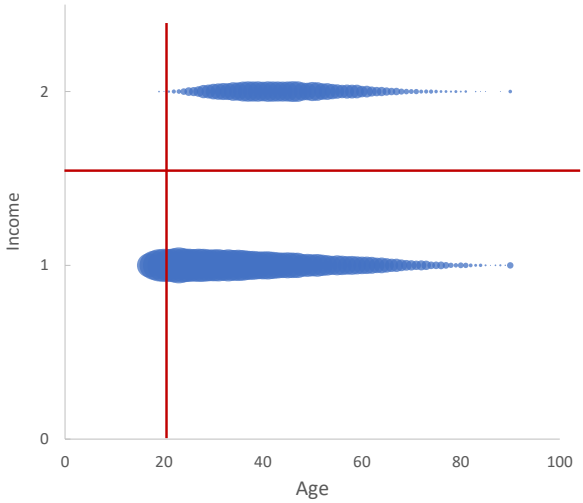

Figure 2: Bubble plot of the attributes 'age' and 'income' from the Census Income dataset, where the bubble size represents the estimated probability. The red lines indicate splits on the domains. Income has two levels where 1 represents '$\leq 50K$ per year' and 2 represents '$> 50K$ per year'.

4 Discovery of ordinal edit rules

In this section, we investigate the automated discovery of ordinal edit rules. Such discovery techniques can be used in scenarios where manual construction of rules is a tedious and expensive operation. For nominal edit rules, algorithms based on low lift are known to work well as discovery techniques [12, 11, 1]. In this paper, we use low lift on pairs of ordinal attributes, leading to edit rules involving exactly two attributes. In practice, the resulting edit rules will then often resemble a breach in a monotone relationship between two attributes.

The restriction to two attributes can be justified by the fact that a large majority of edit rules, discovered for nominal data, involves only two attributes [11]. Additionally, in this first paper on ordinal edit rules, we want to explore the idea of discovery in a simple setting and verify whether useful rules can be found. The more general case of rules involving more than two attributes is considered future research.

Having that said, the idea of rule discovery we propose here can be illustrated with an example. Consider the two attributes 'age' and 'income' of the Census Income dataset depicted in Figure 2, where a bubble plot shows the estimated probability of each combination of values for these attributes.

The red lines indicate threshold values in the domains

of both attributes. If we have such a threshold value for each of two attributes, we obtain four distinct regions in the cross product of the domains. For each of these regions, there is an associated probability for an arbitrary tuple to fall in the region. Since the four regions encompass the entire cross product of the two attributes, these probabilities will sum to one. In the depicted case, we observe that the upper left region contains relatively few tuples. We will now formalize how we use the lift measure to obtain an ordinal edit rule that will precisely match the upper left region of the cross product of the two attributes.

We assume the case of an increasing relationship, such as in Figure 2. Decreasing relationships are treated in the same way, *mutatis mutandis*.

Consider a relation R with schema $\mathscr{R}$ such that the domain of each attribute is a chain. Consider now two attributes a_1 and a_2 and consider a pair of values $(v_1, v_2) \in A_1 \times A_2$. The resulting lift measures are:

$$\overline{\lambda}(v_1, v_2) = \frac{\Pr[a_1 \leq v_1, a_2 > v_2]}{\Pr[a_1 \leq v_1]\Pr[a_2 > v_2]} \tag{1}$$

$$\underline{\lambda}(v_1, v_2) = \frac{\Pr[a_1 > v_1, a_2 \leq v_2]}{\Pr[a_1 > v_1]\Pr[a_2 \leq v_2]} \tag{2}$$

We can now provide the following definition.

Definition 4. *For a relation R with schema $\mathscr{R}$ and two attributes a_1 and a_2, a pair of values $(v_1, v_2) \in A_1 \times A_2$ induces the ordinal edit rule*

$$[\underline{a}_1, v_1] \times]v_2, \overline{a}_2] \times A_3 \times \ldots \times A_k$$

if $\overline{\lambda}(v_1, v_2) \leq \tau$ and induces the ordinal edit rule

$$]v_1, \overline{a}_1] \times [\underline{a}_2, v_2] \times A_3 \times \ldots \times A_k$$

if $\underline{\lambda}(v_1, v_2) \leq \tau$. Hereby, τ is a threshold chosen in $]0, 1]$.

A few remarks can be made here. First, if $\overline{\lambda}(v_1, v_2) = 0$ or $\underline{\lambda}(v_1, v_2) = 0$, then the corresponding edit rules are not very interesting as they are not violated in R. So, in practice, we only want induced edit rules for which the lift measure is very low, but not zero. Second, we use open intervals here only for convenience of notation: an entirely equivalent notation of the induced edit rules could be written down in terms of closed intervals only. Third, although τ can be chosen between 0 and 1, induced edit rules will only make sense if τ is chosen sufficiently close to 0. Section 6 will provide some empirical results on this matter.

Suppose now we are given a relation R and some threshold τ, then how can we find all induced edit rules? A naive algorithm checks all pairs of attributes and, for each pair, tests all possible value pairs (v_1, v_2) to see if either $\overline{\lambda}(v_1, v_2)$ or $\underline{\lambda}(v_1, v_2)$ lies below τ. This means that, for each pair (a_1, a_2), we require $\mathscr{O}(|A_1||A_2|)$ time and this can rapidly become a tedious task. To speed up the search for all induced edit rules, we point out some properties of $\overline{\lambda}(v_1, v_2)$ and $\underline{\lambda}(v_1, v_2)$.

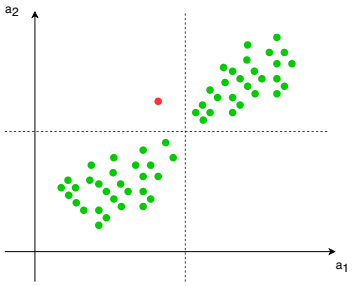

Figure 3: A synthetic dataset with a distribution of tuples and a choice of thresholds (v_1, v_2) that induces the smallest non-zero value for $\overline{\lambda}(v_1, v_2)$.

An extreme case for $\overline{\lambda}(v_1, v_2)$ is illustrated in Figure 3. This case represents the smallest value for $\overline{\lambda}(v_1, v_2)$ that is still greater than 0, given R. This smallest value can be seen as a lower bound for τ, under which no edit rules will be induced.

For the case with a single tuple in the upper left region, a $\overline{\lambda}(v_1, v_2)$ edit rule can only be induced if the distribution of tuples after the first split (i.e., the split by v_1) is not too skew. That is, if we choose v_1 too close to $\underline{a}_1$ or $\overline{a}_1$, then the denominator $\Pr[a_1 \leq v_1]\Pr[a_2 > v_2]$ will be too small and no edit rule can be induced.

In this regard, we can provide the following results.

Proposition 1. *For a relation R with schema $\mathscr{R}$ and two attributes a_1 and a_2, a pair of values $(v_1, v_2) \in A_1 \times A_2$ can only induce an edit rule if:*

$$\tau \geq \frac{4|R|}{(|R|+1)^2}$$

and in addition

$$|R_{a_1 \leq v_1}| \geq \frac{\tau(|R|+1) - \sqrt{\tau^2(|R|+1)^2 - 4\tau|R|}}{2\tau}$$

$$|R_{a_1 \leq v_1}| \leq \frac{\tau(|R|+1) + \sqrt{\tau^2(|R|+1)^2 - 4\tau|R|}}{2\tau}$$

Proof. In order to induce an edit rule for some relation R and some τ, it must be possible for $\overline{\lambda}(v_1, v_2)$ and $\underline{\lambda}(v_1, v_2)$ to be smaller than τ. This is only possible if the smallest value for $\overline{\lambda}(v_1, v_2)$ and $\underline{\lambda}(v_1, v_2)$ is not greater than τ. This smallest value is obtained if the

numerator in the lift measure equals $1/|R|$ and the denominator is maximized accordingly. If we plug these numbers into the definition of $\overline{\lambda}(v_1, v_2)$ we obtain, after some rewriting, that edit rules can only be induced if:

$$\tau|R_{a_1 \leq v_1}|\,(|R| - |R_{a_1 \leq v_1}| + 1) - |R| \geq 0$$

This is a second order polynomial where $|R|$ and τ are constants. Note that, if $|R_{a_1 \leq v_1}| = 0$ the polynomial takes a negative value and for the inequality to hold, a positive value is required.

It is straightforward to calculate the roots of the polynomial and $|R_{a_1 \leq v_1}|$ needs to lie between those roots for the polynomial to be positive. Lastly, if we solve the polynomial to $|R_{a_1 \leq v_1}|$, we find that it only has real solutions if:

$$\tau \geq \frac{4|R|}{(|R|+1)^2}$$

□

Proposition 1 provides us with an initial constraint that has to be satisfied. If τ does not exceed the lower bound in terms of $|R|$, then there is no point in continuing the search for edit rules since we know none will be found.

Additionally, Proposition 1 provides us with a pruning strategy. For some split value v_1, we can verify whether that value can ever lead to the induction of edit rules, given τ and $|R|$. Note that this is particularly useful because if we sort tuples in R according to their values for a_1, then a linear pass over R allows us to find all splits and during that pass, $|R_{a_1 \leq v_1}|$ will increase monotonically.

With Proposition 1 in place, we have a tool to avoid that the splits for all $v_1 \in A_1$ need to be computed. Suppose now we have some value v_1 that splits the relation R into two parts: $R_{a_1 \leq v_1}$ and $R_{a_1 > v_1}$. In a naive approach, we now need to consider all values $v_2 \in A_2$ to make a second split. However, we can make a similar reasoning as we have done before. In the case where we are looking for $\overline{\lambda}(v_1, v_2)$ edit rules, we can iterate over values in A_2 from largest to smallest and if we do so, $\Pr[a_1 \leq v_1, a_2 > v_2]$ will increase. In Figure 3, one can imagine that the horizontal dotted line moves from top to bottom, which increases the number of tuples in the upper left region. At some $v_2 \in A_2$, the number $\Pr[a_1 \leq v_1, a_2 > v_2]$ will be too large and we not need to search further. The same holds for $\underline{\lambda}(v_1, v_2)$, but then we iterate over values in A_2 from smallest to largest. The remaining question is then what the boundary value for A_2 is. To answer that question, we provide the following proposition.

Proposition 2. *For a relation R with schema $\mathscr{R}$ and two attributes a_1 and a_2, a pair of values $(v_1, v_2) \in A_1 \times A_2$ can only induce an edit rule by $\overline{\lambda}(v_1, v_2)$ if:*

$$|R_{a_1 \leq v_1 \wedge a_2 > v_2}| \leq \frac{\tau|R_{a_1 \leq v_1}||R_{a_1 > v_1}|}{|R| - \tau|R_{a_1 \leq v_1}|}$$

and can only induce an edit rule by $\underline{\lambda}(v_1, v_2)$ if

$$|R_{a_1 > v_1 \wedge a_2 \leq v_2}| \leq \frac{\tau|R_{a_1 \leq v_1}||R_{a_1 > v_1}|}{|R| - \tau|R_{a_1 > v_1}|}.$$

Proof. The proof is similar to the proof of Proposition 1, except now we can assume that $|R_{a_1 \leq v_1}|$ and $|R_{a_1 > v_1}|$ are constants. We obtain linear equations in terms $|R_{a_1 \leq v_1 \wedge a_2 > v_2}|$ and $|R_{a_1 > v_1 \wedge a_2 \leq v_2}|$ and solving these equations yields the boundaries given in the proposition. □

Algorithm 1 OrdinalBinaryLiftMiner

```
1: 𝓔 ← ∅
2: A1 ← sort(A1, ≤)
3: A2 ← sort(A2, ≥)
4: if τ ≥ 4|R| / (|R|+1)^2 then
5:     for all v1 ∈ A1 do
6:         if Prop1Prune(v1, τ, R) then
7:             continue
8:         end if
9:         for all v2 ∈ A2 do
10:            if Prop2Prune(v1, v2, τ, R) then
11:                break
12:            end if
13:            if λ̄(v1, v2) ≤ τ then
14:                𝓔 ← 𝓔 ∪ [a̲1, v1]×]v2, ā2]
15:            end if
16:        end for
17:    end for
18: end if
19: return 𝓔
```

The results from Propositions 1 and 2 can now be combined in an algorithm to find all induced edit rules for some relation R and some threshold τ. Algorithm 1 illustrates how the propositions are used to speed up the discovery of $\overline{\lambda}(v_1, v_2)$ edit rules. After testing whether τ is not too low, we consider each pair of attributes (a_1, a_2) once and test those splits that are not discarded by the strategies as explained in the previous. For each case where either $\overline{\lambda}(v_1, v_2)$ or $\underline{\lambda}(v_1, v_2)$ is below τ, we add an edit rule to the output and continue until all combinations are tested.

We end this section with a few remarks. First, as mentioned before we have explained only the case of an increasing relationship between a_1 and a_2, but the dual scenario of decreasing relationships is basically approached in the same way. In fact, we can search for

both types of relationships in one pass. Second, if in future research one wishes to extend this discovery algorithm, we point out the importance of the first condition in Proposition 1. An example extension of the algorithm could be that we search for *conditional* monotonic relationships (i.e., where some third attribute is restricted). In that setting, the verification on τ and the cardinality of the sub relation of R can be a very powerful pruning strategy. Third, an important advantage of the algorithm we described here, is that unlike in the case of nominal data, we do not need to apply any categorization of the data first, before we can start searching for edit rules. Indeed, if one first categorizes attributes into nominal data, one can question whether or not the categorization is not all too biased in the direction of some rules.

5 Related work

To start this section, we want to point out that ordinal edit rules as defined here, naturally arise in many different computational methods. For example, Classification and Regression Tree (CART) models use binary splits where each split is an inequality. Each path of a CART model, excluding the leaf node, thus corresponds to our definition of an ordinal edit rule. Another example is that of Isolation Forests [8], where again, each path of internal nodes of an isolation tree corresponds to one ordinal edit rule. Similarly, monotone decision trees [9, 10], where some monotonic constraint needs to hold between independent and dependent variables, exploit the order of data to construct a learning model. Finally, ordinal edit rules as defined here are a generalization of Sugeno Utility Functionals [2].

Related to ordinal edit rules, are the previously mentioned linear edit rules. In an extreme case, a linear edit rule involves only one attribute, say a_1 and the coefficient of that attribute (i.e., b_1) equals one. The edit rule is then either $a_i \geq -b_0$, $a_i = -b_0$ or $a_i \leq -b_0$, where $-b_0$ is a suitable constant from A_i. Clearly, these are special cases of ordinal edit rules, but they are the only connections between both types of rules. Indeed, whenever two or more attributes are involved, a linear edit rule cannot be written as a single ordinal edit rule and vice versa. This means that, both linear and ordinal edit rules are concise descriptions of particular types of non-permitted value combinations that are mutually complementary.

Edit rules are a relatively simple formalism. They can not model more advanced integrity constraints, such as the constraints that can be modelled with conditional functional dependencies (CFDs) [6] or denial constraints (DCs) [4, 3]. Yet in practice, it is often the case that most of the used constraints are very simple: they concern single tuples and only involve constants. Modelling these constraints can be done with edit rules. Besides, the discovery of edit rules also tends to be significantly faster. Therefore we deem it beneficial to expand the previous literature on edit rules to the ordinal data setting.

The discovery of edit rules using a lift measure has been previously studied in a nominal data setting [12, 1]. A limitation of this previous work is that ordinal attributes need to be discarded or discretized according to expert knowledge to be able to use the discovery algorithm. In this work, we omit this requirement by leveraging a lift formulation that is adapted to ordinal data.

6 Experimental analysis

6.1 Goals and datasets

In this section, we report empirical findings that can serve as evidence for the usefulness of ordinal edit rules in a practical setting.

First, we will evaluate the precision of the discovered ordinal edit rules for different values of τ. This will give us an idea of the quality of the obtained rules and will aid us in choosing a suitable τ value for situations in which it is hard to manually verify the correctness of obtained rules.

Afterwards we will evaluate the runtime of the edit rule discovery algorithm, which we will refer to as OrdinalBinaryLiftMiner. And lastly, we will determine the influence of the proposed pruning strategies.

We used two datasets for experimental verification of the proposed algorithm. The first one is the Adult dataset from the UCI repository [1]. This dataset contains 32561 instances (train set) with 14 attributes. However, because our algorithm is specifically made for ordinal attributes we will restrict the used attributes to the following: 'age', 'education-num', 'hours-per-week' and 'income'. The Adult dataset was selected on the one hand due to the presence of ordinal attributes and on the other hand due to the relative ease of interpretation of the interaction between different attributes. The second dataset is the EudraCT (European Union Drug Regulating Authorities Clinical Trials) dataset [2]. This dataset contains 86670 instances with 13 attributes. All of the attributes are ordinal, though most attributes have only two or three possible values. The main appeal of this dataset is the availability of a ground truth for pairwise edit rules. This

[1] http://archive.ics.uci.edu/ml/
[2] https://eudract.ema.europa.eu/

τ	0.01	0.024	0.037	0.067	0.084	0.1
Nr. rules	1	4	6	11	14	18
Precision	100%	75%	67%	64%	57%	56%

Table 1: Precision of discovered edit rules on EudraCT dataset.

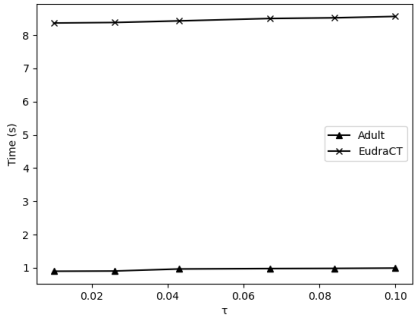

Figure 4: Runtime of OrdinalBinaryLiftMiner in function of maximum lift threshold τ.

allows us to get an objective view on the quality of the recovered edit rules.

OrdinalBinaryLiftMiner was implemented in Java and tested on a single core of an Intel i7-5500U (2.40 GHZ) CPU.

6.2 Analysis of rule discovery

We report the precision of the discovered rules for the EudraCT dataset in Table 1. We do this for different values of τ in the range of $[0.01, 0.1]$, which is the range that is most interesting in practice since the discovered edit rules should have a relatively low lift. This is also the range commonly used in previous literature [12, 1]. We observe that our results in terms of precision are similar to the results reported in [12], which is no surprise since we use a modification of the lift measure adapted to ordinal data.

Next, we tried to obtain a precision score for the rules for the Adult dataset, but we found it more difficult to objectively make a binary decision about the correctness of the rules. While the Adult dataset is generally considered easy for interpretation w.r.t. edit rules, the most clear edit rules involve the attributes 'marital-status' and 'relationship', which are absent in our case due to the fact that these attributes are not ordinal. Though to the best of the authors' judgement, the overall quality of the rules was comparable to the rules of the EudraCT dataset.

In Figure 4 we can see the runtime of OrdinalBinaryLiftMiner on Adult and EudraCT. We used values

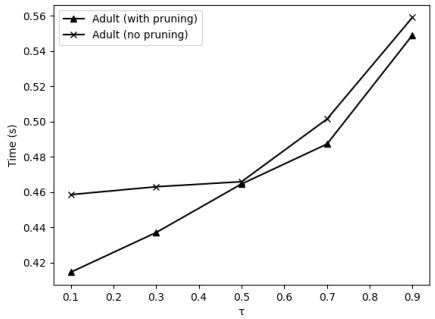

Figure 5: Runtime of OrdinalBinaryLiftMiner with and without pruning, in function of maximum lift threshold τ.

of τ in the $[0.01, 0.1]$ range. We can observe that EudraCT takes significantly longer than Adult for all values of τ. This is due to its larger size, i.e. higher amount of tuples and attributes. Additionally, we can see that increasing the value of τ yields only a modest increase in runtime. This is due to the fact that the algorithm spends a considerable amount of time on preprocessing the data, so the time spend searching for edit rules becomes relatively small. We will refer to this as a constant overhead, i.e. overhead independent from the choice of τ.

To evaluate the effectiveness of the previously introduced pruning strategies we run our algorithm on the 'age' and 'education-num' attributes of Adult. The reason for using only two attributes is to reduce the algorithm's constant overhead, which should aid in seeing the effect of pruning. Additionally, we select these two attributes because they have a relatively large amount of possible values, which should enhance the impact of pruning. If we were to pick two attributes with binary values, then pruning becomes much less effective of course.

The obtained runtime measurements can be seen in Figure 5. This time we used values of τ in the range $[0.1, 0.9]$. We note that with pruning, the runtime increases linearly with τ, whereas without pruning the runtime for small values of τ is clearly increased w.r.t. the expected linear regime. This is of particular importance because for practical use we expect values of τ to be ≤ 0.1.

7 Conclusion

We have developed an efficient algorithm for the discovery of pairwise ordinal edit rules. Based on properties of the lift measure, we have derived certain bounds which are employed in pruning strategies to speed up

the search for edit rules. Hereby, we extend the previous work on nominal edit rules to an ordinal setting. Our results show that we can obtain a comparable edit rule precision in an ordinal setting, without incurring an excessive computational cost.

In future work, it would be of interest to investigate approaches for scaling the algorithm to more than two attributes and how this affects the lift measure and pruning strategies. And lastly, it would be useful to construct a method for rule implication that exploits the total order available in an ordinal setting.

Acknowledgement

This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

References

[1] T. Boeckling, A. Bronselaer, G. De Tré, Mining data quality rules based on T-dependence, in: Proceedings of the 11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019), Vol. 1, 2019, pp. 184–191.

[2] Q. Brabant, M. Couceiro, D. Dubois, H. Prade, A. Rico, Extracting decision rules from qualitative data via sugeno utility functionals, in: Proceedings of the IPMU conference, 2018, pp. 253–265.

[3] X. Chu, I. F. Ilyas, P. Papotti, Discovering denial constraints, Proc. VLDB Endow. 6 (13) (2013) 1498–1509.
URL `https://doi.org/10.14778/2536258.2536262`

[4] X. Chu, I. F. Ilyas, P. Papotti, Holistic data cleaning: Putting violations into context, in: 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013, pp. 458–469.

[5] T. De Waal, J. Pannekoek, S. Scholtus, Handbook of statistical Data Editing and Imputation, Wiley, 2001.

[6] W. Fan, F. Geerts, X. Jia, A. Kementsietsidis, Conditional functional dependencies for capturing data inconsistencies 33 (2).
URL `https://doi.org/10.1145/1366102.1366103`

[7] I. Fellegi, D. Holt, A systematic approach to automatic edit and imputation, Journal of the American Statistical Association 71 (353) (1976) 17–35.

[8] F. T. Liu, K. M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Transactions on Knowledge Discovery from Data 6 (2008) 1–39.

[9] D. Petturiti, C. Marsala, Monotone classification with decision trees, in: Proceedings of the EUSFLAT conference, 2013, pp. 810–817.

[10] R. Potharst, J. Bioch, Decision trees for ordinal classification, Intelligent Data Analysis 4 (2) (2000) 97–112.

[11] J. Rammelaere, F. Geerts, Cleaning data with forbidden itemsets, IEEE Transactions on Knowledge and Data Engineering (2019) 1–1.

[12] J. Rammelaere, F. Geerts, B. Goethals, Cleaning data with forbidden itemsets, in: 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017, IEEE Computer Society, 2017, pp. 897–908.