Correlation-Based Refinement of Rules with Numerical Attributes

André Melo
University of Mannheim
andre@informatik.uni-mannheim.de

Martin Theobald
University of Antwerp
martin.theobald@ua.ac.be

Johanna Völker
University of Mannheim
voelker@informatik.uni-mannheim.de

Abstract

Learning rules is a common way of extracting useful information from knowledge or data bases. Many of such data sets contain numerical attributes. However, approaches like Inductive Logic Programming (ILP) or association rule mining are optimized for data with categorical values, and considering numerical attributes is expensive. In this paper, we present an extension to the top-down ILP algorithm, which enables an efficient discovery of datalog rules from data with both numerical and categorical attributes. Our approach comprises a preprocessing phase for computing the correlations between numerical and categorical attributes, as well as an extension to the ILP refinement step, which enables us to detect interesting candidate rules and to suggest refinements with relevant attribute combinations. We report on experiments with U.S. Census data, Freebase and DBpedia, and show that our approach helps to efficiently discover rules with numerical intervals.

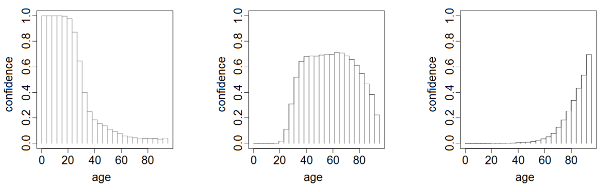
Introduction

Discovering patterns by learning rules from knowledge or data bases enables us to obtain concise descriptions of a domain, to predict new facts and to detect anomalies. Especially by considering numerical values such as birth dates or measurements, which are common in many data sets, we are often able to unveil interesting patterns (e.g. spatial or seasonal). We illustrate the benefits and challenges of learning rules with numerical attributes by an example from the U.S. Census[1] dataset, where we would like to discover rules describing the marital status of a person with a minimum confidence threshold of 0.7. The rules shown in Table 1 do not satisfy the given threshold, but we can significantly increase their confidence values if we restrict the numerical *age* variable Y to specific intervals.

In particular, by observing how the confidence values are distributed over the *age* variable Y (cf. Figure 1), we notice that for the numerical intervals $[0, 30]$, $[30, 80]$ and $[90, \infty)$ the confidences are much higher. In other words, the figure shows that younger people are likely to be single (1a), middle-aged married (1b), and elderly widowed (1c).

id	rule	conf
r_1	*maritalStatus(X,single) :- age(X,Y)*	0.40
r_2	*maritalStatus(X,married) :- age(X,Y)*	0.46
r_3	*maritalStatus(X,widowed) :- age(X,Y)*	0.06

Table 1: Example of rules without numerical intervals



(a) r_1 (b) r_2 (c) r_3

Figure 1: Confidence distribution over age Y

Given this insight, we can refine the rules r_1, r_2 and r_3 by restricting Y to the aforementioned intervals, and obtain rules which satisfy the confidence threshold (cf. Table 2).

id	rule	conf
r_4	*maritalStatus(X,single) :- age(X,Y),*$Y \in [0, 30]$	0.91
r_5	*maritalStatus(X,married) :- age(X,Y),*$Y \in [30, 80]$	0.72
r_6	*maritalStatus(X,widowed) :- age(X,Y),*$Y \in [90, \infty)$	0.71

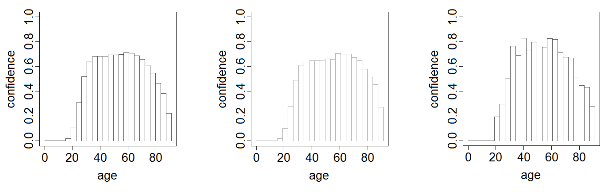
Table 2: Example of rules refined with numerical intervals

However, note that if we had attempted to learn a concept which is uncorrelated to the attribute *age*, such as *quarterOfBirth*, a costly search for interesting *age* intervals would have been useless. This is because for every quarter of the year the confidence distribution over Y is practically uniform, as seen in the Figure 3 example for the first quarter case (*quarterOfBirth(X,q1) :- age(X,Y)*). Hence, there is no interval for Y that yields significant confidence gain when compared to the overall Y domain. Generally, it is desirable to avoid the computation of intervals for uninteresting combinations of numerical and categorical attributes, since the queries to obtain the support and confidence distributions are expensive and the search space can easily become too large. So, *before* running these expensive queries for a rule

[1]U.S. Census (2000): http://www.rdfabout.com/demo/census/

with unrestricted numerical attributes variables, we would like to estimate the likelihood that the rule has a confidence distribution with interesting intervals.

This interestingness estimation can also be used for choosing better categorical refinements. For example, we could further extend rule r_2 (cf. Table 1) by also considering the U.S federal state a person lives in. We observe that the confidence distribution of rule r_2 refined with *Florida* (Figure 2b) is very similar to the confidence distribution of the more general rule r_2 (Figure 2a), whereas for *South Dakota* (Figure 2c) the distribution is less uniform and yields higher confidence values. Therefore, we are more likely to discover interesting rules if we choose *South Dakota* instead of *Florida* as a refinement of the base rule.



(a) USA (b) Florida (c) South Dakota

Figure 2: Confidence distribution of rule r_2 over the *age* attribute for the overall USA population (a) as well as refined by the states Florida (b) and South Dakota (c)

In this paper, we propose an efficient approach to mining datalog rules with numerical and categorical attributes which employs information theory techniques to speed up the ILP algorithm. We estimate the likelihood of any given rule to have an interesting refinement in the form of an interval restriction on the numerical attribute. This way of measuring a rule's (potential) interestingness enables us to efficiently explore the search space, rank the rules, and prune uninteresting rules (e.g. *quarterOfBirth(X,q1)* :- *age(X,Y)*). Our approach comprises a preprocessing phase for computing the correlations between numerical and categorical attributes, as well as an extension to the ILP refinement step. We demonstrate the feasibility of our approach by means of experiments with three large-scale datasets (U.S. Census data, Freebase and DBpedia).

Problem Definition

Firstly we need to introduce the notions of *base rules* and *refined rules*:

Definition 1 *A Base Rule is a rule whose body contains a free numerical attribute variable.*

Definition 2 *A Refined Rule is a base rule with the numerical attribute variable restricted to some interval.*

For example, the rules r_1, r_2 and r_3 are examples of base rules, while r_4, r_5 and r_6 are their refined counterparts. Note that we only consider base rules with a single numerical attribute. Our goal is to learn refined rules which yield significant confidence gain over their corresponding base rules.

Definition 3 *Confidence Gain is the ratio of the confidence of a refined rule to its base rule ($conf(r_{ref})/conf(r_{base})$).*

The confidence of a rule h *:-* b (with h being the head literal and b being the body clause of the rule) is defined as $conf(h\text{:-}b) = supp(\{h,b\})/supp(b)$. Its support is given by the number n^+ of covered positive examples, i.e., $supp(h\text{:-}b) = n^+(\{h,b\})$.

We consider a base rule *interesting* if it has a refined rule that satisfies the minimum support, confidence and confidence gain thresholds. As we want to efficiently explore the datalog base rules search space in ILP, our goal is to predict the interestingness of a base rule before computing its confidence distribution.

Related Work

Related work consists of approaches for learning rules with numerical attribute intervals (also called quantitative rules) in association rule mining as well as Inductive Logic Programming (ILP). To the best of our knowledge, there is no prior work which focuses on the efficient exploration of the base-rules search space in ILP. The works presented in this section address different problems which are tightly related to our proposed approach.

The related works on association rule mining focus on learning rules of the form $(A_1 \in [l_1, u_1]) \wedge C_1 \Rightarrow C_2$, where A_1 is an uninstantiated numerical attribute, l_1 and u_1 are the lower and upper boundaries of A_1, and C_1 and C_2 are instantiated conditions. Srikant and Agrawal (1996) use a priori discretization of the numerical attribute domain into fine partitions in order to treat them as regular categorical attributes. They learn rules for the individual partitions and then try to merge rules with adjacent partitions into wider intervals, but their approach does not guarantee the optimality of the generated intervals. Brin, Rastogi, and Shim (1999) propose an algorithm with linear complexity for finding optimal intervals. It features a supervised bucketing technique, which collapses instances together, reducing input size without sacrificing optimality. Mata, Alvarez, and Riquelme (2002) employ evolutionary techniques which do not require the discretization of the numerical attributes, and Salleb-Aouissi, Vrain, and Nortet (2007) employ a genetic-base algorithm in order to find the optimal interval boundaries. Note that these algorithms could be integrated into our ILP extension for finding the optimal lower and upper boundaries, after we have detected an interesting base rule.

Inductive Logic Programming (ILP), introduced by Muggleton (1991), is a state-of-the-art technique for learning concepts from examples, which we base our approach on. There are two main induction methods: top-down (which starts with specific clauses and searches for generalizations), and bottom-up (which starts with a general clause and searches for specializations). In this paper, we use the top-down approach of FOIL Quinlan (1990), which essentially consists of a covering and a specialization loop. The former ensures completeness, i.e. all positive examples are covered, whereas the latter ensures consistency, i.e. no negative examples are covered. In order to handle noisy data, we use a minimum expected accuracy (namely the probability that an example covered by the clause is positive) as a stopping criterion in the specialization loop. We also restrict the hypothesis space to safe datalog rules, which support arithmetical predicates providing the expressiveness required to define the intervals of numerical attributes.

Interestingness Measure

We are interested in base rules with non-uniform confidence distributions of low entropy, which can result in interesting refined rules with high confidence gain. This kind of confidence distribution is the result of divergent support distributions of a rule's body and positive examples.

In Figure 3 we compare the rules *quarterOfBirth(X,q1):-age(X,Y)* and *maritalStatus(X,married):-age(X,Y)* which share the same body. As we see in Figures 3a and 3d, the first rule's body and positives distribution are practically the same resulting in an uniform confidence distribution (3e), while for the latter rule, its divergent positives distribution (3b) leads to a more interesting confidence distribution (3c).

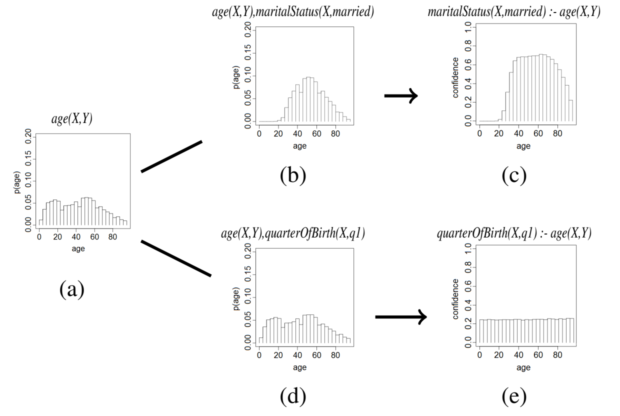


Figure 3: Example of a body support distribution (3a) with different positives distributions (3b) and (3d)

The interestingness of a rule can be measured by computing the divergence of the $\{body\}$ and $\{body, head\}$ support distributions. Hence, we discretize the domain of Y into k disjoint buckets $b_1, \dots, b_k$, in order to obtain the support histogram of a clause c, which is defined as:

$$h(c) =< h_1(c), \dots, h_k(c) >, \text{ where}$$
$$h_i(c) = supp(c|Y \in b_i) \text{ and } |h(c)|_1 = supp(c)$$

from which we obtain the probability density distribution of support $f(c) = h(c)/supp(c)$, where $|f(c)|_1 = 1$.

Based on this we define the interestingness measure as $I_Y(l|c)$, where l is the literal to be added to the clause c, and Y is the free numerical attribute variable we want to find an interval for. To compute I_Y we use a divergence measure D compare the distributions of c and$\{c, l\}$ over Y :

$$I_Y(l|c) = D(f(c)||f(\{c, l\})) \qquad (1)$$

Because of sampling error, this divergence measure tends to yield higher values for clauses with lower support, and thus favors them over higher support clauses. In order to compensate this effect, and because we are also interested in rules with high support, we propose a hybrid interestingness measure combining divergence and support:

$$I_Y(l|c) = supp(\{c, l\}) * D(f(c)||f(\{c, l\})) \qquad (2)$$

In our implementation, we use Kullback-Leibler (1951) as the divergence measure D, and we apply Laplace smoothing to all distributions prior to the divergence value calculation in order to remove the zeros.

Correlation Lattice

In a preprocessing step, we create a correlation lattice for each numerical property. The purpose of the lattice is to model the correlations between a given numerical property and different categories, which in our implementation are defined by categorical literals.

A correlation lattice is structurally similar to an itemset lattice, introduced by Agrawal, Imieliński, and Swami (1993). It describes the correlations between a numerical property and multiple categorical attributes. The target numerical property with a join variable X and a free numerical variable Y is used as the root literal, and the categorical literals correspond to the items in the itemset lattice. All of the categorical literals are joined by the join variable X, forming in each node n a unique clause c_n. Figure 4 shows an example of lattice for the attribute *age(X,Y)*.

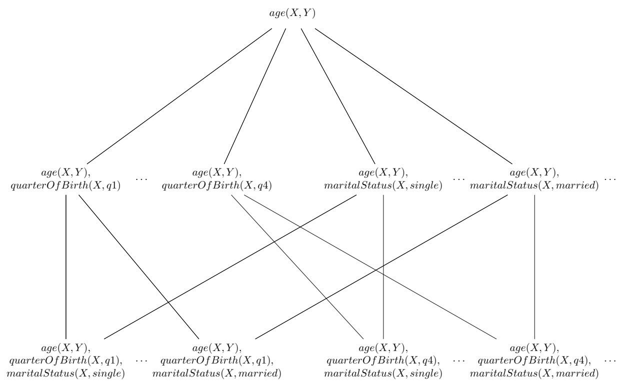

age(X, Y)
age(X, Y), quarterOfBirth(X, q1)
age(X, Y), quarterOfBirth(X, q4)
age(X, Y), maritalStatus(X, single)
age(X, Y), maritalStatus(X, married)
age(X, Y), quarterOfBirth(X, q1), maritalStatus(X, single)
age(X, Y), quarterOfBirth(X, q1), maritalStatus(X, married)
age(X, Y), quarterOfBirth(X, q4), maritalStatus(X, single)
age(X, Y), quarterOfBirth(X, q4), maritalStatus(X, married)

Figure 4: Example of correlation lattice for the numerical attribute *age*

Additionally, every node n has a corresponding histogram $h(c_n)$ with c_n support distribution over Y. Therewith, all the edges between a parent node and a child node have an associated interestingness value of adding the child's new literal to the parent's clause. Since we need to compare the support distributions, the bucket boundaries of the histograms must be consistent through the whole lattice. These boundaries are defined in the root node based on the overall population. In our implementation, the number of buckets is arbitrarily defined, and we use equal frequencies as discretization method, because of its robustness with regard to skewed distributions.

The correlation lattice is built with an Apriori-based algorithm, similar to that introduced by Agrawal and Srikant

(1994), which takes advantage of the anti-monotonicity property of the support of the clauses in the lattice. Thus, it is possible to safely prune the nodes which do not satisfy the support threshold. We also limit the number of levels in the lattice to d_{max}, which is upper bounded by the maximum number of literals in clauses allowed in the ILP algorithm.

Independence Test

With the information contained in the lattice, we can test whether two categorical literals are independent given a certain clause. Let us assume we have two nodes n and m with common parent p, and common child q, with clauses c_p, $c_n = \{c_p, l_n\}$, $c_m = \{c_p, l_m\}$ and $c_q = \{c_p, l_m, l_n\}$, where l_m and l_n are literals not contained in the clause c_p. It is possible to estimate the histogram $\hat{h}$ of a node q, assuming that l_n and l_m are conditionally independent given c_p, as $\hat{h}(c_q) = h(c_n)h(c_m)/h(c_p)$. With the estimated $\hat{h}(c_q)$ and the observed actual $h(c_q)$, we can perform a Pearson's chi-squared test of independence.

Figure 5 shows an example where we check whether *maritalStatus(X,single)* and *employmentStatus(X,unemployed)* are independent. There the test yields a p-value of 0.96, which means that we have a high confidence that the two literals are conditionally independent.

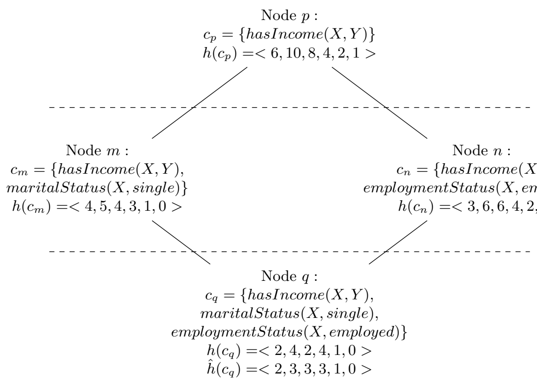

Node p :
$c_p = \{hasIncome(X, Y)\}$
$h(c_p) =< 6, 10, 8, 4, 2, 1 >$
Node m :
$c_m = \{hasIncome(X, Y),$
$maritalStatus(X, single)\}$
$h(c_m) =< 4, 5, 4, 3, 1, 0 >$
Node n :
$c_n = \{hasIncome(X, Y),$
$employmentStatus(X, employed)\}$
$h(c_n) =< 3, 6, 6, 4, 2, 1 >$
Node q :
$c_q = \{hasIncome(X, Y),$
$maritalStatus(X, single),$
$employmentStatus(X, employed)\}$
$h(c_q) =< 2, 4, 2, 4, 1, 0 >$
$\hat{h}(c_q) =< 2, 3, 3, 3, 1, 0 >$

Figure 5: Example of a test for independence between the literals *maritalStatus(X,single)* and *employmentStatus(X,unemployed)*

Detecting independence is important because, when we refine a clause in the specialization loop by adding a new literal, the confidence distribution is not likely to change significantly, if this new literal is conditionally independent of the head given the body. For example, if we refine l_m:-c_p by adding l_n to the body, the confidence distribution does not change significantly because of the conditional independence ($P(l_m|c_p) \approx P(l_m|c_p, l_n)$). Therefore, it is useful to set a maximum p-value threshold in order to define and detect independence. Note that for deeper levels in the lattice, a particular node can be generated from different pairs of joining nodes, therefore we cannot prune the conditionally independent nodes. Instead, we mark the two edges joining the conditionally independent nodes as independent, and take that into account when refining a clause in order to prune a conditionally independent refinement.

Scalability

Building a complete lattice can become an unfeasible task even with the apriori pruning and the maximum depth. In this case the lattice with k categories can have up to $\sum_{i=1}^{d_{max}} \binom{k}{i}$ nodes. As employing a very large k would be prohibitive, we suggest to use the interestingness measures introduced in the previous section as pruning heuristics. We also evaluate their performance in the experiments section.

Although the lattice construction can take a significant amount of time, it is important to point out that the clauses contained in the lattice comprise a portion of the ILP search space. The information in the lattice can be reused in the core ILP algorithm preventing the execution of unnecessary queries. In addition, we can directly extract rules of the form:

$$h(X, k_h) \text{ :- } b_1(X, k_{b_1}), ..., b_n(X, k_{b_n}), r(X, Y), Y \in [l, u]$$

where r is the root numerical property, l and u are the lower and upper boundaries of Y, $h(X, k_h)$ and $b_1(X, k_{b_1}), \ldots, b_n(X, k_{b_n})$ are the categorical literals, with $n < d_{max}$. Rules of this form are semantically equivalent to association rules with numerical attributes, with every instantiation of X considered as a transaction.

The correlation lattice can be easily extended to incorporate more complex categories by adopting a broader definition of categories. If we use categorical clauses as defined in Melo (2013) instead of categorical literals, we are able to access categories which are not directly connected to the lattice join variable X. This allows us to, for instance, incorporate categories from linked datasets by using the *owl:sameAs* property as a linking relation, enabling us to add cross-domain information to the lattice. However, the problem also becomes more challenging as the number of categories considered and joins required can dramatically increase.

ILP Algorithm

In order to take advantage of the proposed preprocessing when learning datalog rules with numerical attributes, we need to extend the core ILP algorithm. In the specialization loop, whenever we detect a base rule whose numerical attribute has a correlation lattice, we query the lattice in order to check whether it is relevant to consider refining it with numerical intervals.

Algorithm 1 first checks whether a clause is a base rule, then for each base rule we query the lattice to obtain its interestingness value (cf. Algorithm 2). Subsequently, if the interestingness value satisfies the minimum threshold, we search for the numerical interval (cf. Algorithm 3), where the support and confidence distributions are queried.

These distributions are then analyzed by an algorithm which searches for interesting intervals which is represented by the *getInterestingIntervals* function. Note that for the experiments described further below, we did not search for an *optimal* interval as proposed by Brin, Rastogi, and Shim (1999), for example, but we only checked if there exists *any* interesting interval. This is sufficient, in order to find out whether our interestingness predictions are correct. Finally, we add the refined rules with the optimal intervals to the refinement graph and continue the ILP algorithm normally.

Algorithm 1: Numerical Attribute Interval Refinement

```
foreach literal $l$ in the body of the clause $c$ do
    if $l$ is numerical and
    $l$ has free numerical attribute variable and
    $\exists$ a lattice $lattice_l$ for $l$'s property then
        if getInterestingness($c$,$lattice_l$) $\geq min_{interest}$ then
            searchNumericalIntervals($c$,$l$);
```

Algorithm 2: getInterestingness(c_{base},$lattice_l$)

```
$Y \leftarrow$ numerical variable of $l_{num}$;
$b \leftarrow$ root node of $lattice_l$;
foreach $l_i \in$ body of $c_{base}$ do
    if $l_i \neq l_{num}$ and $l_i$ joins $l_{num}$ and $l_i$ is categorical then
        $b \leftarrow b$.getChild($l_i$);
$l_{head} \leftarrow$ head of $c_{base}$;
$c_{body} \leftarrow$ clause of node $b$;
$h \leftarrow b$.getChild($l_{head}$);
if $h \neq b$ and edge $\{b, h\}$ is not independent then
    return $I_Y(l_{head}, c_{body})$;
else
    return 0;
```

Algorithm 3: searchNumericalIntervals(c_{base},l_{num})

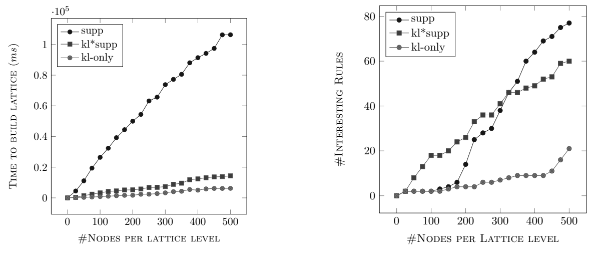
```
$Y \leftarrow$ numerical variable of $l_{num}$;
$conf$[] $\leftarrow$ queryConfidenceDistribution($c_{base}$,$Y$);
$supp$[] $\leftarrow$ querySupportDistribution($c_{base}$,$Y$);
$intervals \leftarrow$ getInterestingIntervals($conf$,$supp$);
foreach $interval_i \in intervals$ do
    $c_{ref} \leftarrow c$ refined with $interval_i$;
    add $c_{ref}$ to ILP's refinement graph;
```

Experiments

For our experiments we developed a Java-based implementation of the proposed ILP extension, which uses RDF3X (Neumann and Weikum 2010) as a knowledge base backend. First we focus on evaluating the quality of the correlation lattice and the scalability of its construction in terms of runtime. Afterwards we evaluate how the proposed correlation-based approach affects the core ILP algorithm. Since there is no prior work on the exploration of base rules search space, we compare our approach against the naïve exhaustive search, which searches for all the base rules in the ILP refinement graph. This allows us to evaluate the core contribution of the proposed approach by measuring the runtime reduction and its impact on the number of learned rules in comparison to the naïve approach.

We used support only (*supp*) as a baseline, Kullback-Leibler only (*kl-only*), and Kullback-Leibler combined with support (*kl*supp*). Experiments with other divergence measures such as Jensen-Shannon and Chi-square were carried out, too. However, preliminary results showed no major performance differences. Hence, we only report results for Kullback-Leibler focusing on the comparison of support only, divergence only, and hybrid measures. Our experiments were conducted on a Intel i7-3770 3.40 GHz with 32 GB RAM.

U.S. Census. In the first experiment, we greedily built limited size correlation lattices using the different interestingness measures as heuristics. We varied the limit size and counted the number of interesting rules with numerical intervals which can be found in the lattice, in order to assess its quality. Finally, we measured the required time for building the lattices. For this experiment, we use the *U.S. Census* (2000) dataset, because of its high quantity of numerical and categorical properties, high completeness and low noise. We built a lattice for the *income* property and 16 categorical properties totalizing 224 categories with various degrees of correlation with *income*. The lattice was constructed on a 1.7 GB partition (with 884,365 person entities) and the extracted rules were tested on a 0.5 GB partition (with 369,024 *person* entities). The partitions were created with random sampling, and they are mutually disjoint.



(a) Build time per lattice size

(b) Learned rules per lattice size

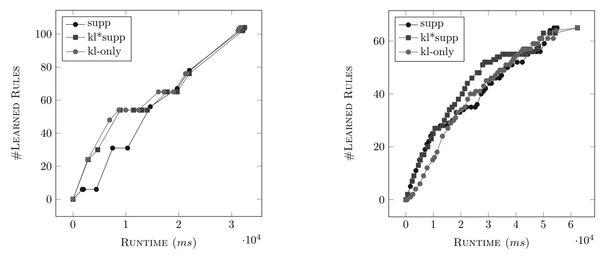
Figure 6: Evaluation of the correlation lattice construction

In Figure 6a, we see that *supp* takes the longest to build the lattice, *kl-only* takes the shortest, while *kl*supp* lies in-between. This is because the support prioritizes nodes with higher support clauses, while the divergence, because of sampling error, prioritizes lower support.

Figure 6b shows that although for the hybrid measure the lattice requires much less time to be built than for *supp*, the number of resulting interesting rules is roughly the same on average, and significantly higher for smaller lattices. The comparatively low efficiency of the *supp* measure for smaller lattice sizes can be accounted for by the fact that the literals with the highest support, such as *hasDeficiency(X,none)*, tend to have very low divergence and low correlation to *income*. We also tested the accuracy of facts predicted by the learned rules. For all measures, the average accuracy was 0.81, with the exception of *kl-only* (0.75), since the sampling error caused a lower average support.

In the second experiment, we evaluate the proposed ILP extension, and compare the effectiveness of the different interestingness measures. When the minimum interestingness thresholds are set to zero, our approach is equivalent to a naïve exhaustive search. Therefore, we know the number of interesting base rules which exist in the search space. This is reflected by the top-right points in Figure 7. For the evaluation reported in the following, we use *DBpedia* 3.9[2] and *Freebase*[3]. Although these datasets do not contain as many numerical facts as the U.S. Census data, they consist of richer RDF graphs with a high number of links between different entities. Hence, they enable us to evaluate our extension's performance when it comes to learning more complex datalog rules.

DBpedia. In the DBpedia experiment, we used 27 properties from the domain *movies* and related domains containing 3.5 million facts, and focused on learning rules with numerical intervals for the movie *budget* property with 14,769 facts. Firstly we ran the naïve algorithm where for all base rules in the ILP refinement graph we perform the numerical intervals search, in order to find out how many interesting refined rules exist and how long the exhaustive search takes. Afterwards, we run our proposed extension with different interestingness threshold values, and for each threshold value, we measured the runtime spent with the search for refined rules and the number of interesting rules discovered. Figure 7a shows the number of learned rules per runtime for the different measures. We notice that *supp* has a nearly linear growth, while the other measures have an overall better performance, indicating that including divergence in the interestingness measure helps speed up the ILP algorithm.



(a) DBpedia (b) Freebase

Figure 7: Evaluation of the ILP extension

Freebase. We performed the same experiment on 27 properties of the Freebase domain *people* and related domains containing a total of 10 million facts, and learned rules with numerical intervals for the properties *weight*, *dateOfBirth*, *height* and *salary*, with 99,918, 1,351,870, 160,326 and 16,452 facts, respectively. This difference in terms of performance highly depends on characteristics of the data, such as the degree of correlation between the numerical attributes and categories or the average support of the interesting base rules. Nevertheless, we notice that the hybrid measure are the most robust of the proposed measures, consistently being the best overall performer in the experiments.

Conclusions

In this paper, we presented an extension to the ILP algorithm which uses correlation lattices to enable the efficient discovery of rules with numerical attribute intervals. We suggested different interestingness measures for constructing these lattices and evaluated their performance on the lattice construction and interestingness prediction tasks. Our experiments indicate that the combination of divergence and support is the best performing measure both as a pruning heuristic and for speeding up the core ILP algorithm.

Possible next steps are to study of the lattice build time and ILP speed up trade-off, to extend our approach to handle multidimensional as numerical attributes, to investigate numerical domain discretization methods and their effect on our approach, and to determine optimal minimum interestingness threshold values by analyzing the lattices structure. Finally, we are envisioning an application of our approach to multiple Linked Data sources. By building correlation lattices with categories from distinct but related sources we hope to enable an efficient discovery of cross-domain rules.

Acknowledgement

The work reported in this paper was partially funded by the DFG project GOLD[4]. Johanna Völker is supported by a Margarete-von-Wrangell scholarship of the European Social Fund (ESF) and the Ministry of Science, Research and the Arts Baden-Württemberg.

References

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

Agrawal, R.; Imieliński, T.; and Swami, A. 1993. Mining association rules between sets of items in large databases. *SIGMOD Rec.* 22(2):207–216.

Brin, S.; Rastogi, R.; and Shim, K. 1999. Mining optimized gain rules for numeric attributes. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 135–144. ACM Press.

Kullback, S., and Leibler, R. A. 1951. On information and sufficiency. *Ann. Math. Statist.* 22(1):79–86.

Mata, J.; Alvarez, J.; and Riquelme, J. 2002. An evolutionary algorithm to discover numeric association rules. In *In Proceedings of the ACM symposium on Applied computing (SAC)*, 590–594.

Melo, A. 2013. Learning rules with numerical and categorical attributes from linked data sources. Master's thesis, Universität des Saarlandes, Saarbrücken.

Muggleton, S. 1991. Inductive logic programming. *New Generation Computing* 8(4):295–318.

Neumann, T., and Weikum, G. 2010. The RDF-3x engine for scalable management of RDF data. *The VLDB Journal* 19(1):91–113.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5:239–266.

Salleb-Aouissi, A.; Vrain, C.; and Nortet, C. 2007. Quantminer: A genetic algorithm for mining quantitative association rules. In Veloso, M. M., ed., *IJCAI*, 1035–1040.

Srikant, R., and Agrawal, R. 1996. Mining quantitative association rules in large relational tables. *SIGMOD Rec.* 25(2):1–12.

[2]http://dbpedia.org/Downloads39

[3]https://developers.google.com/freebase/data (2013-10-06)

[4]http://gold.linkeddata.org