Towards a Numerical Rule Mining Language

Luis Galárraga
Télécom ParisTech
Paris, France

Fabian M. Suchanek
Télécom ParisTech
Paris, France

1 Introduction

A knowledge base (KB) is a computer-processable collection of knowledge about the world. Recent years have seen the rise of KBs such as YAGO [17], Freebase [1], DBpedia [3], or the Google Knowledge Vault [5]. These KBs typically contain facts of the form *spouse(ElvisPresley, PriscillaPresley)*. Modern KBs contain facts about millions of entities. These KBs can be mined for *rules*, such as

$$spouse(x, y) \wedge hasChild(x, z) \Rightarrow hasChild(y, z)$$

This rule says that if x is married to y and x has a child z, then z is also the child of y. Such rules do not always hold, and hence they are usually given a weight or confidence measure. Recent work [8] allows us to mine such rules at large scale in a few minutes on today's KBs. This works by systematically enumerating all possible rules in a search tree, and pruning this tree aggressively.

We propose to enlarge the scope of rule mining to *numerical rules*. A numerical rule contains variables that do not bind to named entities, but to numerical values. Consider, e.g., the rule

$$type(x, marketEcon) \wedge import(x, y) \wedge export(x, z) \wedge cad(x, w) \Rightarrow w \approx 1.2 \times (y - z)$$

This rule says that if x is a market economy, then its current account deficit (*cad*) is roughly 120% of the difference between its import and export (measured in dollars, for example). Such rules would add tremendous value to today's KBs: First, the rules could be used to spot errors in the data. If the cad of a country does not fall in the expected range, then the value could be flagged as erroneous or less confident. Second, such rules could be used to compute missing information. If, e.g., we do not have the cad of France, we could compute it from the data. Finally, the rules carry human-understandable insight. The finding that the cad of a country correlates with its import and export has a value in itself [16].

Mining such rules is challenging for several reasons. First, there are infinitely many numerical constants. In the example, the factor could be refined to 1.21 or 1.3. Second, there are infinitely many ways in which numerical variables can correlate. For example, the cad could be the sum, product or any other function of the ages of its members of parliament. Thus, the search space becomes unbearably large. Finally, numerical rules bridge two traditionally different areas of research: inductive logic programming (ILP) and statistical data mining. While statistical data mining techniques can spot correlations of values, they are less able to simultaneously derive logical constraints such as *type(x, marketEcon)*. ILP, on the other hand, traditionally does not deal with numerical values.

In this vision paper, we study the different types of numeric rules. We develop a unified language for writing down numerical rules, and we explain which types of rules can already be mined by existing work. With this thrust, we hope to lay the ground for research that will enable us to mine numerical rules just as efficiently as logical rules.

2 Numerical Rules

2.1 Horn Rules

The starting point of most ILP and rule mining approaches are *Horn rules*. Horn rules are based on *atoms*. An atom takes the form $r(x_1, \ldots, x_m)$ where r is a relation name and at least one argument in $x_1, ..., x_m$ is a variable. A Horn rule takes the form

$$B_1 \wedge ... \wedge B_n \Rightarrow H \qquad \text{(abbreviated} \quad \vec{B} \Rightarrow H)$$

[1] http://freebase.com

Constraint	Example rule with this constraint
$x \geq \phi$	$age(x, y) \Rightarrow y \geq 0$
$x > \phi$	$type(x, solvent) \wedge balance(x, y) \Rightarrow x > 0$
$x = \phi$	$type(x, dog) \wedge numLegs(x, y) \Rightarrow y = 4$
$x \in [a, b]$	$lat(x, y) \Rightarrow y \in [-90, 90]$
$x \approx \phi$	$wikiLinks(x, y) \wedge articleLength(x, z) \Rightarrow z \approx 3.4 \times log(y)$

Table 1: Numerical constraints and some examples

where $B_1, ..., B_n$ are the *body atoms* and H is the *head atom*. An *instantiation* of a rule is a substitution for the variables. An atom a *holds* under an instantiation σ in a KB $\mathcal{K}$, if $\sigma(a) \in \mathcal{K}$. A rule *holds* under σ in $\mathcal{K}$, if either one of the body atoms does not hold or the head atom holds.

2.2 Numerical Constraints

One of the most intuitive ways to add numerical constraints to Horn rules is to permit *constraints*, i.e., atoms of the form $x \circ \phi$. Here, x is a numerical variable, $\circ$ is a binary operator that returns a truth value, and ϕ is an expression that may or may not involve variables. The atom *holds* under an instantiation σ, if $\sigma(x) \circ \sigma(\phi) = true$. Atoms that are not constraints are called *categorical atoms*. Categorical atoms that contain a numerical variable are called *numerical atoms*. Table 1 shows some types of numerical constraints and example rules.

We could also aim to mine cardinality constraints such as $hasWonPrize(x, EmmyAward) \wedge actedIn(x, y) \Rightarrow \#y > 10$. Such constraints are particularly interesting because they allow us to mine negations by setting $\#y = 0$ or $\#y < c$ for $c \in \mathbb{N}$. Negations, however, are inconsistent with the Open World Assumption that KBs make. We could also mine rules such as $type(x, city) \wedge hasPopulation(x, y) \Rightarrow y \sim N(\mu; \sigma)$, saying that a set of values follows a certain distribution. We leave such rules as future work and concentrate on the operators from Table 1. We notice that the addition of numerical constraints may lead to pathological cases such as

$$p(x, y) \quad \Rightarrow \quad z = y \tag{1}$$

$$p(x, y) \quad \wedge \quad z = y \quad \Rightarrow \quad r(x, y) \tag{2}$$

$$p(x, y) \quad \wedge \quad z > y \quad \Rightarrow \quad r(x, z) \tag{3}$$

In cases (1) and (2), we are binding a variable z that serves no purpose. In case (3), we are predicting $r(x, z)$ where z has no defined value. These phenomena are a particularity of constraints. If we replace the constraints by categorical atoms, the problem does not appear:

$$p(x, y) \quad \Rightarrow \quad q(z, y) \tag{4}$$

$$p(x, y) \quad \wedge \quad q(z, y) \quad \Rightarrow \quad r(x, y) \tag{5}$$

$$p(x, y) \quad \wedge \quad q(z, y) \quad \Rightarrow \quad r(x, z) \tag{6}$$

To avoid the pathological rules, we impose that every numerical variable has to appear in at least one categorical atom. We also impose that the head atom can only contain *bound* variables, i.e., variables that appear in the body either in a categorical atom or in a constraint that restricts it to one particular value.[2] Different from [8], we do not impose that every variable has to appear at least in two atoms, because we want to allow rules such as $hasAdvisor(x, y) \wedge age(y, z) \Rightarrow z > 30$.

2.3 Descriptive and Predictive rules

Consider the following rule:

$$gdp(x, y) \quad \wedge \quad natDebt(x, z) \quad \Rightarrow \quad z = 0.9y \tag{7}$$

This rule says that if x has a GDP and a national debt, then the debt can be computed as 90% of the GDP. If x has no national debt, then the rule will always hold trivially. We say that the rule is a *descriptive rule* because it will not try to *predict* the debt of x. Any rule that does not have a categorical atom in the head is descriptive. Now consider the following variant of the rule:

$$gdp(x, y) \quad \Rightarrow \quad \exists z : natDebt(x, z) \wedge z = 0.9y \tag{8}$$

[2]This is usually the case for $=$, although not necessarily, cf. $x = x$. $\approx$ also binds, as does $x \in [a, a]$.

This rule says that every x that has a GDP also has a national debt, and that this debt amounts to 90% of the GDP. This formula allows for prediction of the national debt of a country given only its GDP. However, it does not qualify as a Horn rule, because it contains a conjunction and an existential quantifier in the head. We propose to write this rule as

$$gdp(x, y) \quad \wedge \quad z = 0.9y \quad \Rightarrow \quad natDebt(x, z)$$

This rule has the same semantics, but does not require the existential quantifier[3]. We call rules with a categorical atom in the head *predictive rules*.

2.4 Functions

A *function* is a binary atom that has, for each first argument, at most one second argument in the KB. It is very common for rule mining systems to assume that numerical atoms are functions. In our survey of the state of the art (Section 3), we did not find any approach that would not make this assumption. For example, a country can have only one national debt value. This assumption makes sense only if we do not take into account the dimension of time.

If all numerical atoms are functions, we can use them in constraints. For example, we can write

$$natDebt(x) > 5 \times gdp(x) \quad \Rightarrow \quad hasProblem(x)$$

to mean

$$gdp(x, y) \quad \wedge \quad natDebt(x, z) \quad \wedge \quad z > 5y \quad \Rightarrow \quad hasProblem(x)$$

In general, any *function term* $f(x)$ is replaced by a fresh variable y, and the categorical atom $f(x, y)$ is added to the body. This rewriting applies also if the function term is in the head:

$$indebted(x) \quad \Rightarrow \quad natDebt(x) = 5 \times gdp(x)$$

becomes

$$indebted(x) \quad \wedge \quad natDebt(x, y) \quad \wedge \quad gdp(x, z) \quad \Rightarrow \quad y = 5 \times z$$

This rewriting, however, allows only the descriptive semantics of $=$. There is no way to predict that x has a national debt, if x doesn't have that property already. To allow for this subtlety, we permit the assignment atom $f(x) := \phi$ in the head. Such an atom is replaced by $f(x, y)$ (where y is a fresh variable), and this time the constraint $y = \phi$ goes to the body. Thus,

$$indebted(x) \quad \Rightarrow \quad natDebt(x) := 5 \times gdp(x)$$

becomes

$$indebted(x) \quad \wedge \quad gdp(x, z) \quad \wedge \quad y = 5 \times z \quad \Rightarrow \quad natDebt(x, y)$$

Every rule with function terms can be transformed into an equivalent rule without function terms. Vice versa, every rule in which all numerical atoms are functions can be rewritten to an equivalent rule without numerical variables. This is because every numerical variable y has to be bound in some categorical atom $f(x, y)$. Thus, we can replace y by $f(x)$. As we will show in the next section, function terms allow us to extend the notions of support and confidence — designed originally for categorical rules — to rules with constraints. For this reason, we propose to use the language of Horn rules with function terms as a unified language for rules with numerical constraints.

2.5 Evaluation Metrics

Support. The statistical significance of a rule is measured by the *support* metric. The support of a rule $\vec{B} \Rightarrow H$ is the absolute number of cases in the KB where the rule holds. Following [8], we count the support as the number of instantiations of the head variables:

$$support(\vec{B} \Rightarrow H) := \#x_1, \ldots, x_m : \exists z_1, \ldots, z_k : \vec{B} \wedge H$$

Here, $\#x_1, \ldots, x_m$ are the variables in H, $z_1, ..., z_k$ are the variables that appear only in $\vec{B}$, and $\#x_1, \ldots, x_m : \Phi$ stands for $|\{x_1, ..., x_m : \Phi\}|$. This definition of support does not transfer well to rules that have a constraint in the head, such as $natDebt(x, y) \wedge gdp(x, z) \Rightarrow y > z$. This is because we would want to count the different countries for which this rule holds, not the different debt values.

[3]The FORS system [10] uses the same notation, albeit with the Prolog predicate *is/2*.

If $natDebt$ is a function, we could associate y to x and count on x instead. In our language of function terms, we get this behavior for free: The rule becomes $\Rightarrow natDebt(x) > gdp(x)$ (with an empty body). In such a rule, we count the support on all variables in the head as given by the function terms – i.e., on x in this case.

Confidence. The *standard confidence* of a rule measures the ratio of correct conclusions:

$$stdconf(\vec{B} \Rightarrow H) := \frac{support(\vec{B} \Rightarrow H)}{\#x_1, \ldots, x_m : \exists z_1, \ldots, z_k : \vec{B'}}$$

Here, $\vec{B'}$ is the body of the rule after all function terms have been rewritten. For example, the standard confidence of the predictive rule $R : indebted(x) \Rightarrow natDebt(x) := 5 \times gdp(x)$, rewritten as $indebted(x) \wedge gdp(x, z) \wedge y = 5 \times z \Rightarrow natDebt(x, y)$, is given by

$$stdconf(R) := \frac{\#x : \exists\, y, z : indebted(x) \wedge gdp(x, z) \wedge z = 5 \times y \wedge natDebt(x, y)}{\#x : \exists\, y, z : indebted(x) \wedge gdp(x, z) \wedge y = 5 \times z}$$

where the equality constraint in the denominator is trivially satisfied and could be omitted. The standard confidence was originally designed for descriptive rules under the Closed World Assumption, that is, it punishes rules concluding facts that are not in the KB. In the example, any indebted country with a GDP will lower the confidence value, no matter whether it has the wrong debt value or no value at all. However, KBs operate under the Open World Assumption, meaning that a fact that is not in the KB is not necessarily false. To avoid unfairly punishing a rule, we can use the Partial Completeness Assumption (PCA) [8]. It says that if a KB knows *some r-values* for an entity, then it knows *all its r values* and new values outside this set are assumed false. If a rule predicts, e.g., additional children for a person, the PCA assumes the KB is partially complete and counts these predictions as false, but if the person has no children, the predictions are not used as counter-examples. Likewise, if a rule predicts the national debt of a country for which this value is unknown, the PCA does not use this as counter-evidence. This notion is encoded in the formula

$$pcaconf(\vec{B} \Rightarrow r(x, y)) := \frac{support(\vec{B} \Rightarrow r(x, y))}{\#x_1, ..., x_n : \exists z_1, \ldots, z_k, y' : \vec{B} \wedge r(x, y')}$$

This formula is defined only for binary relations in the head. Most KBs that implement the OWA use binary relations only anyway. If the head atom is an assignment atom, the rule has to be rewritten into its function term free form, and the function of the assignment atom takes the role of $r(x, y)$. In our example, this yields:

$$pcaconf(R) := \frac{\#x : \exists\, y, z : indebted(x) \wedge gdp(x, z) \wedge z = 5 \times y \wedge natDebt(x, y)}{\#x : \exists\, y, z, y' : indebted(x) \wedge gdp(x, z) \wedge y = 5 \times z \wedge natDebt(x, y')}$$

Again, the equality constraint in the denominator is trivially fulfilled and could be omitted. We see that the PCA confidence counts as counter-examples only those countries for which a national debt value is known. For rules that have a constraint $x \circ \phi$ in the head in their function term free form, the head atom becomes $x \circ y'$ in the denominator. Thus, it is trivially fulfilled and can be omitted, which makes the PCA confidence boil down to the standard confidence.

Confidence and support measure two orthogonal dimensions of the quality of a rule, which can be combined [12], but are kept distinct in most rule mining approaches [8, 1, 9, 13].

Error measures. If a rule contains a numerical constraint in the head, other measures may become applicable. For example, if the rule concludes $y \approx 3x$, then the quality of the rule could depend on the difference $|y - 3x|$. If we know that x follows certain probability distribution, $x \sim F(\mu, \sigma)$, then the confidence could be the probability that our predictions were drawn from such a distribution, i.e., $P(y \mid y = 3x \wedge x \sim F(\mu; \sigma))$. While such measures abound, the difficulty is to make numeric measures interoperable with classical measures. For example, if a rule mining algorithm produces both numerical and non-numerical rules, and if numerical rules have an error measure but no confidence, then it is unclear how these rules can be ranked together. We propose to replace a constraint $y \approx \phi$ by $y > \phi - \epsilon \ \wedge \ y < \phi + \epsilon$ for a suitable error margin ϵ. If the constraint appears in the head, the rule has to be split into two rules that each have one of the conjuncts in the head. The value of ϵ depends on the domain of y. If the domain of y operates under a *value scale*, i.e., scales for which ratios are not defined (such as temperatures or geographic coordinates), ϵ can be defined as an

absolute error. This requires the choice of an appropriate numerical constant, which depends, e.g., on the unit system that y uses. An absolute error of 2 units for predicting temperatures in Celsius may be too loose if the scale is changed to Fahrenheit or if we are trying to predict latitudes. If y operates under a *ratio scale* (as is the case for populations or distances), we can use a *relative error* $\epsilon = \alpha y$, $\alpha \in (0, 1)$. Then, α is independent of the unit system and can be set to a default value.

With this change, all error measures become binary measures, and hence interoperable with the categorical support and confidence measures.

3 Existing Work

We survey here existing work in the area of numerical rule mining, and express the rules in our proposed language with function terms. The work of [14] studies the problem of mining optimized rules of the form $A \in [a, b] \wedge \ C_1 \Rightarrow C_2$ on a database table. Here, A is a column name and C_1, C_2 are equality constraints on columns, as in $year \in [1990, 2000] \wedge src\_city = NY \Rightarrow dest\_city = Paris$. In our language, we would define a unique identifier for each row of the table and express the columns as binary relations, e.g., $year(id) \in [1990, 2000] \wedge src\_city(id, NY) \Rightarrow dest\_city(id, Paris)$. The problem consists in finding the values of a and b that maximize either support or confidence given a minimum threshold for the other metric. The general problem with an arbitrary disjunction of inequality constraints is shown to be NP-Hard [14]. One way to reduce the search space is to *discretize* it. [7] uses equi-depth bucketing and computational geometry techniques to calculate a and b almost optimally, providing bounds for the approximation error. The approach described in [4] proposes to optimize for the gain, a function of support and confidence. A method based on genetic algorithms [15] allows mining confidence-optimized rules with categorical atoms and multiple interval constraints, as in $height(x) \in [150, 180] \Rightarrow weight(x) \in [50, 90]$.

The work presented in [2] incorporates interval constraints into the standard ILP framework, mining rules such as $age(x) \in [a, b] \Rightarrow status(x, Single)$. The goal is to find a, b that maximize the rule confidence. Before refining the rule, the system performs an independence test between the age of people and their marital status. If the variables are not independent, then the algorithm reports intervals where it is more likely to find single people, i.e., intervals where confidence is maximized.

Regression trees [11] are a common method to learn functional correlations in numerical structured data. Imagine we want to predict the length of a Wikipedia article as a function of its number of links: $\Rightarrow wikiLength(x) := g(wikiLinks(x))$. Most mining systems use the writing $wikiLinks(x, y') \Rightarrow \exists\ y : wikiLength(x, y) \wedge y = g(y')$, and start with a training set consisting of all the articles x where the dependent variable y is known. If some quality conditions are met, a regression model $y \approx g(y')$ is induced and the algorithm stops. Otherwise, the training set is split into positive and negative examples and their hypothesis refined with new categorical atoms, e.g., $type(x, Scientist)$. The algorithm is then recursively applied to the new hypotheses and training sets. FORS [10] was one of the first systems in integrating functional correlations using regression trees into the ILP framework. In the syntax of FORS, the rule is $wikiLinks(x, y') \wedge is(y, g(y')) \Rightarrow wikiLength(x, y)$. FORS cannot mine the descriptive variant of such a rule. Furthermore, the system was tried out only on toy examples and is unlikely to scale to the millions of facts in today's KBs. A more recent approach [6] relies on regression trees for the prediction of numerical attributes in OWL KBs. The approach learns rules that predict the value of one fixed numerical attribute of an entity, based on the properties of that entity, as in $talkShow(x) \wedge dealsWith(x, sports) \Rightarrow popularity(x) := 12\%$.

4 Conclusion

We have seen that there has been quite some work on numeric rule mining. In our proposed rule language, these approaches become comparable. However, so far, these works solve only islands of numeric rule mining. None of the approaches can mine full-fledged rules with function terms at scale. This, however, may be about to change: As we make progress on small scale numerical rule mining, and as categorical rule mining starts to scale, these islands may soon become connected. A unified syntax for numerical rules may be the first step.

5 Acknowledgments

This work is supported by the Chair "Machine Learning for Big Data" of Télécom ParisTech.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, 1993.

[2] M. T. Andr Melo and J. Vlker. Correlation-based refinement of rules with numerical attributes. In *FLAIRS*, 2014.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *ISWC*, 2007.

[4] S. Brin, R. Rastogi, and K. Shim. Mining optimized gain rules for numeric attributes. In *SIGKDD*, 1999.

[5] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[6] N. Fanizzi, C. d'Amato, and F. Esposito. Towards numeric prediction on owl knowledge bases through terminological regression trees. In *ICSC*, 2012.

[7] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. *Journal of Computer and System Sciences*, 58(1):1 – 12, 1999.

[8] L. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.

[9] B. Goethals and J. Van den Bussche. Relational Association Rules: Getting WARMER. In *Pattern Detection and Discovery*, volume 2447. Springer Berlin / Heidelberg, 2002.

[10] A. Karali and I. Bratko. First order regression. *Machine Learning*, 26(2-3):147–176, 1997.

[11] S. Kramer. Structural regression trees. In *AAAI*, 1996.

[12] F. Mahdisoltani, J. Biega, and F. Suchanek. YAGO3: A knowledge base from multilingual Wikipedias. In *CIDR*, 2015.

[13] S. Muggleton. Learning from positive data. In *ILP*. Springer-Verlag, 1997.

[14] R. Rastogi and K. Shim. Mining optimized association rules with categorical and numeric attributes. *IEEE Trans. on Knowl. and Data Eng.*, 14(1):29–50, Jan. 2002.

[15] A. Salleb-aouissi, C. Vrain, and C. Nortet. Quantminer: A genetic algorithm for mining quantitative association rules. In *IJCAI*, pages 1035–1040, 2007.

[16] F. Suchanek and N. Preda. Semantic culturomics (vision paper). In *VLDB*, 2014.

[17] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A Core of Semantic Knowledge. In *WWW*, 2007.