Data Cleaning using Probabilistic Models of Integrity Constraints

Anonymous Author(s)
Affiliation
Address
email

Abstract

In data cleaning, data quality rules provide a valuable tool for enforcing the correct application of semantics on a dataset. Traditional rule discovery techniques assume a reasonably clean dataset, and fail when faced with a dirty one. Enforcement of these rules for error detection is much less effective when mined on dirty data.

In the databases literature, a popular and expressive type of logic-based data quality rule (or Integrity Constraint) is the *constant Conditional Functional Dependency* (cCFD) [Fan et al., 2011], which can be easily understood by a data analyst.

We introduce a probabilistic model that combines error detection and rule induction (cCFDs), we show that this methodology performs better than just traditional logic-based error detection. Moreover, after inference is performed, we provide a set of rules which is statistically sound and with low redundancy. To the best of our knowledge this is the first work to combine statistical anomaly detection with logic-based approaches to data cleaning.

1 Introduction

In practical machine learning, usually a data scientist has to spend 80% of his time wrangling and checking consistency of the data [Kandel et al., 2012]. In industry, most of the data usually comes in tabular (e.g. CSV files, Excel sheets), or from a different perspective, relational form (e.g. relational database systems).

The most common data science pipeline starts with raw data, often dirty. Error detection is an important step in data cleaning. Dirty data usually has a number of issues, and data quality rules have to be enforced and updated constantly in order to keep value to the business, such that predictions are made on solid foundation.

In fact, there is a need to infer, apply and monitor data quality rules, particular those for tabular datasets. Data repairing, often uses these rules that are either inferred automatically, by a data analyst or knowledge expert. Moreover, these rules are often part of the schema (i.e. blueprint) of relational databases. In reality, it is unrealistic to expect a human to perform rule discovery for data quality, particularly without any tools given the intricacy and the size of today's datasets.

We tackled this problem from a probabilistic point-of-view, trying to provide an algorithm for error detection, and robust rule induction for cCFDs - by removing redundant and spurious rules from a candidate set. Traditional approaches for CFD and cCFD induction are defined in [Fan et al., 2011].

Our probabilistic model (graphical model) was implemented using *Structural Expectation Maximization* (SEM) in [Friedman, 1998]. We attempt robust rule induction, and show that traditional discovery techniques do not perform as well.

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

We obtained good results for error detection with our model: both with the set of rules induced, and directly from the model itself. The final set of induced rules was reduced, thus less redundant. We obtain better results than traditional techniques under significant noise.

2 Related work

In the data cleaning pipeline, one of the first steps towards cleaning the dataset is to detect errors. Often, error detection can be reduced to the problem of anomaly detection, particularly in tabular datasets. In tabular datasets, quantitative or logic-based methods can be used to detect anomalies. The quantitative approach is statistically inspired, meanwhile the logic-based can use integrity constraints or data quality rules, as well as user-defined data transformations.

Formally, error detection using integrity constraints (ICs) usually involves detecting the tuples (rows) that violate a set of constraints seen as describing the dataset. Recently, two good surveys have been published [Ilyas and Chu, 2015] and [Fan, 2015] on data cleaning, mostly focusing on logic-based approaches.

On the other hand, in quantitative error detection, there has been considerable work in outlier detection for quantitative data, as seen in survey [Hellerstein, 2008]. Most of this work is based on robust estimators, and methods for univariate and multivariate outlier detection, but it also contains observations on relational data. A tutorial on outlier detection can be found in [Kriegel et al., 2009]. Methods have also been developed for distributional change detection in [Dasu et al., 2009].

3 Preliminary Definitions

Let R be a relational schema, and I an instance of that schema. A relational schema usually refers to the organization of a dataset as a blueprint of how an overall database is constructed, that is the number of tables and the number of columns and domain of each column. This can involve rules called integrity constraints, on the type of data accepted - formatting and values.

For instance, one can have rules on the behaviour of single columns of a table (features of a dataset), or rules that mix multiple columns. Attributes (features) of R are denoted as $attr(R) = \{A_1, ..., A_m\}$, and for each attribute A in R, let $dom(A)$ denote the domain of A. I consists of a set of tuples (rows in a table, or examples in dataset), each of which belongs to the domain $dom(A_1) \times ... \times dom(A_m)$. One also assumes that there is a unique tuple identifier associated with each tuple $t \in I$. Further, one can denote a cell of attribute A of a tuple t in I by $I(t[A])$, or by simply writing $t[A]$.

3.1 Definition and Discovery of constant CFDs

A Functional Dependency (FD) $s = X \rightarrow Y$, where $X \subseteq attr(R)$ and $Y \subseteq attr(R)$ are sets of attributes (features), define a dependency between attributes. A dataset I with schema R supports FD s if for any two tuples t_α, t_β in I if $t_\alpha[X] = t_\beta[X]$ then $t_\alpha[Y] = t_\beta[Y]$ is true. Relevant rule induction algorithms for FDs include TANE ([Huhtala et al., 1999]) and FastFD ([Wyss et al., 2001]).

This means that for any two tuples sharing the same value for attributes X, but different values for attribute(s) Y, there must be an error in either tuple. Usually, one denotes X as being the left-hand-side LHS(s) and Y as being the right-hand-side RHS(s).

A constant Conditional Functional Dependency (cCFD) s over R is a pair $(X \rightarrow Y, t_p)$. $X \rightarrow Y$ is a standard FD. The pattern tuple t_p with attributes (features) in X and Y, where for each $B \in X \cup Y$, is such that $t_p[B]$ is a set of constants $a \in dom(B)$. Moreover, one could enforce $|Y| = 1$, i.e. RHS(s) has one attribute (feature) only. Traditional logic-only induction algorithms for mining CFDs are defined in [Fan et al., 2011], where the author defines an extension to both TANE and FastFD, meanwhile CFDMiner (for cCFDs only) builds upon concepts of *Association Rule* mining. These traditional methods assume that rules to be mined have confidence level 1 (100 %), i.e. I completly supports s. Note however that a confidence level 1 assumption is not always true for cCFD induction in a dirty dataset I.

cCFDs can be obtained for a dataset I using non-redundant *Association Rule* mining techniques like [Szathmary et al., 2007], and then splitting their RHS(s) such that only rules with $|\text{RHS}(s)| = 1$ are obtained. These mined cCFDs can then be used as candidate rules to be selected by our model.

4 Generative Model of the Dataset

We present a probabilistic model for a tuple $t \in N$ in dataset I, with $N = |I|$, assuming a known set of cCFDs named $\mathcal{S}$. The graphical model in Figure 1 defines our approach.

The generative story for our graphical model can be defined for each tuple (example) $t \in N$ and attribute (feature) A, such that each cell value $x_{t[A]} \in \mathbf{x}_t$ can be drawn from either: $P_{data}(x_{t[A]})$ the clean data model (e.g. density model, belief network); or $P_{noise}(x_{t[A]})$ noise model (e.g. uniform distribution for standard outlier detection); or $\mathcal{F}_s(\mathbf{x}_{t[v]}|u_{ts}, \mathbf{z}_{t[v]})$ a deterministic factor for each cCFD rule $s \in \mathcal{S}$ that is defined as 1 if the rule exists or is not violated in t, 0 otherwise.

This approach can be seen as a mixture of experts with binary gating functions, selecting between the model of clean data, the deterministic factor(s) of the logical rule(s) available in $\mathcal{S}$, or a noise model defining an outlier. For this purpose, two different indicator variables $\mathbf{z}_t$ and $\mathbf{u}_t$ are defined.

Variable $z_{t[A]} \in \mathbf{z}_t$ defines whether attribute (feature) in t is considered clean with $z_{t[A]} = 1$, or dirty with $z_{t[A]} = 0$. One can define a Bernoulli prior on $z_{t[A]}$, such that $z_{t[A]} \sim Bern(\theta_A)$, with θ_A how clean attribute (feature) A is.

Further, the model is learning the set of cCFD rules $\mathcal{S}$, and it provides an indicator variable u_{ts} for the support of each rule $s \in S$ by tuple t, allowing multiple rules to generate $x_{t[A]}$. We say that a cCFD rule is supported by t if for rule $s = (X \rightarrow Y, t_p)$, given attributes $v \in X \cup Y$, the pattern tuple of s supports the attributes in t, such that their values $t_p[v] = x_{t[v]}$, and associated hidden variables $z_{t[v]}$ are considered clean ($= 1$) for all in v. The factor $\mathcal{F}_s(\mathbf{x}_{t[v]}|u_{ts}, \mathbf{z}_{t[v]})$ is defined as follows:

$$\mathcal{F}_s(\mathbf{x}_{t[v]}|u_{ts}, \mathbf{z}_{t[v]}) = \begin{cases} 0, & \text{if } u_{ts} = 1, \mathbf{z}_{t[v]} = 1, \text{ and } \mathbf{x}_{t[X]} = t_p[X], \text{and } \mathbf{x}_{t[Y]} \neq t_p[Y] \\ 1, & \text{otherwise} \end{cases}$$

Intuitively, $\mathcal{F}_s(\mathbf{x}_{t[v]}|u_{ts}, \mathbf{z}_{t[v]})$ is 0 only if t violates rule s, except that $\mathcal{F}_s(\mathbf{x}_{t[v]}|u_{ts}, \mathbf{z}_{t[v]})$ is disabled if any of u_{ts} or $\mathbf{z}_t$ is zero. Note that $\mathbf{x}_{t[X]} = t_p[X]$ means tuple t supports the values of LHS(s), and $\mathbf{x}_{t[Y]} \neq t_p[Y]$ means that t does not support values of RHS(s).

One can now define the joint distribution of the hidden variables $z_{t[A]} \in \mathbf{z}_t$, $u_{ts} \in \mathbf{u}_t$ and visible variable $x_{t[A]} \in \mathbf{x}_t$, given the Bernoulli parameters $\theta = \{\theta_A\}_1^{attr(R)}$ for each attribute (feature) $A \in attr(R)$:

$$P(\mathbf{x}_t, \mathbf{z}_t, \mathbf{u}_t|\theta) \propto \prod_{A \in attr(R)} \left[[\theta_A P_{data}(x_{t[A]})]^{z_{t[A]}} [(1-\theta_A) P_{noise}(x_{t[A]})]^{1-z_{t[A]}}\right]^{\prod_{s'}(1-u_{ts'})} \prod_{s \in \mathcal{S}} \mathcal{F}_s(\mathbf{x}_{t[v]}|u_{ts}, \mathbf{z}_{t[v]})^{u_{ts}}$$

This model can be learnt using Viterbi Expectation Maximization (Hard-EM) for variables $\mathbf{z}_t$ and $\mathbf{u}_t$ (E-Step), along with parameter θ (M-Step). Periodically, after a set of iterations of Hard-EM, we test if a new rule s can be accepted into $\mathcal{S}$ depending on whether the likelihood of the model increases, otherwise candidate cCFD s is rejected. This last structural M-Step defines the Structural Expectation Maximization (SEM) algorithm [Friedman, 1998], which is used to infer the whole model. Hence, the model is concurrently and iteratively learning set S. $P_{data}(x_{t[A]})$ can either be given, or obtained as the algorithm progresses by only using clean data (i.e. $z_t[A] = 1$).

5 Results

For our experiments we used the Adult dataset (UCI Machine Learning Repository), with both categorical and continuous features, injected with random errors (outliers and typos). Further, our model (Prob-Log) used density estimation for $P_{data}(x_{t[A]})$, and a uniform distribution for $P_{noise}(x_{t[A]})$, hence performing outlier detection. Our model uses the variables $z_{t[A}$ for cell error detection, and thus tuple error detection.

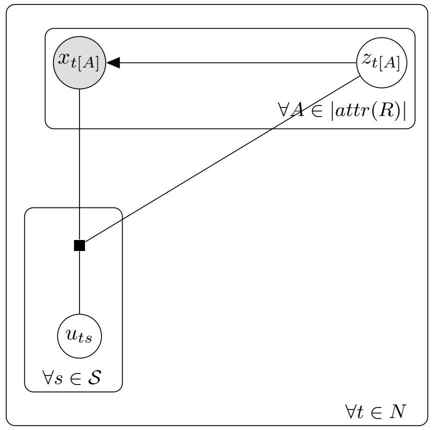

Figure 1: Graphical model for joint Error Detection and Rule Learning. Note that $x_{t[A]}$ is the only visible variable, representing the values of cells in dataset I

We compared our model (Prob-Log) to two other methods in error detection and rule induction (cCFDs) performance. The first traditional method, which assumes a moderately clean dataset is CFDMiner, mines for rules with confidence 1, in the *Association Rule* sense. The second is ZART which is a non-redundant *Association Rule* mining algorithm that was modified to obtain cCFDs, its definition allows for us to mine rules with confidence less than 1, thus more robust to noise. Intuitively, error detection with CFDMiner and ZART can be achieved by searching for the tuples in the dataset that violate their induced set of cCFD rules.

We present results for both number of rules induced (Table 1), and F-Measure for the error detection process (Figure 2). The noise is injected at random from 0 % to 20 % of the cells in the dataset. In Table 1 low_conf are rules mined with ZART which exhibit poor confidence in the dataset, in the *Association Rule* sense, whilst $high\_conf$ are high confidence rules (near 1).

Results in Figure 2 show that our model (Prob-Log, in purple) obtained good error detection performance, against rule-based detection using the set of cCFDs mined by ZART (Candidate Set fed into Prob-Log, in blue), CFDMiner (Ground-Truth, in red), and the rule set $\mathcal{S}$ induced by our model (in green). Note that CFDMiner had access to the clean dataset to induce its cCFDs, thus we name it Ground-Truth.

Finally, results in Table 1 suggest that Prob-Log offers substantial reduction in number of cCFDs (less redundancy) without much loss in error detection performance. Particularly when the rules are more spurious (less confidence), tagged low_conf (Table 1).

Corruption Level	Candidate Type	ZART (Candidate Set)	Prob-Log (Set $\mathcal{S}$)	CFDMiner
0.1 %	high_conf	58	43	1352
1 %	high_conf	46	38	538
1 %	low_conf	265	115	538
3 %	high_conf	58	48	19
5 %	high_conf	69	59	0
5 %	low_conf	248	133	0
7 %	high_conf	71	58	0
10 %	high_conf	70	54	0
10 %	low_conf	265	156	0
15 %	high_conf	66	48	0
15 %	low_conf	270	169	0
20 %	high_conf	128	86	0

Table 1: Number of Rules generated per method, for each injected noise level in Adult dataset - from 0.1% to 20 % erroneous cells, corrupted at random. Ground-Truth cCFD rules using CFDMiner registers 611 rules on the clean dataset.

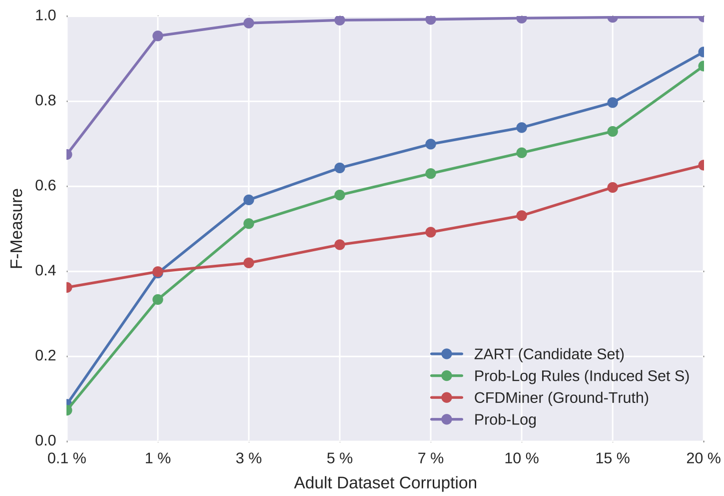

Figure 2: F-Measure of error detection per method, for each injected noise level in Adult dataset - from 0.1% to 20 % erroneous cells, corrupted at random.

Acknowledgments

References

Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. Discovering conditional functional dependencies. *IEEE Trans. on Knowl. and Data Eng.*, 23(5):683–698, May 2011. ISSN 1041-4347. doi: 10.1109/TKDE.2010.154. URL `http://dx.doi.org/10.1109/TKDE.2010.154`.

Sean Kandel, Andreas Paepcke, Joseph M Hellerstein, and Jeffrey Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18 (12):2917–2926, 2012.

Nir Friedman. The bayesian structural em algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 129–138, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-555-X. URL `http://dl.acm.org/citation.cfm?id=2074094.2074110`.

Ihab F. Ilyas and Xu Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends® in Databases*, 5(4):281–393, 2015. ISSN 1931-7883. doi: 10.1561/1900000045. URL `http://dx.doi.org/10.1561/1900000045`.

Wenfei Fan. Data quality: From theory to practice. *SIGMOD Record*, 44(3):7–18, 2015. doi: 10.1145/2854006.2854008. URL `http://doi.acm.org/10.1145/2854006.2854008`.

Joseph M Hellerstein. Quantitative data cleaning for large databases. *United Nations Economic Commission for Europe (UNECE)*, 2008.

Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Outlier detection techniques. *Tutorial in PAKDD 2009*, 2009.

T. Dasu, S. Krishnan, D. Lin, S. Venkatasubramanian, and K. Yi. Change (Detection) You Can Believe in: Finding Distributional Shifts in Data Streams. *Lecture Notes in Computer Science*, 5772:21, 2009. doi: 10.1007/978-3-642-03915-7_3.

Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111, 1999.

Catharine Wyss, Chris Giannella, and Edward Robertson. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract.

In *International Conference on Data Warehousing and Knowledge Discovery*, pages 101–110. Springer, 2001.

L. Szathmary, A. Napoli, and S. O. Kuznetsov. ZART: A Multifunctional Itemset Mining Algorithm. In *Proc. of the 5th Intl. Conf. on Concept Lattices and Their Applications (CLA '07)*, pages 26–37, Montpellier, France, Oct 2007. URL `http://hal.inria.fr/inria-00189423/en/`.