Properties of Inconsistency Measures for Databases

Samantha Berg
A20456450
sberg1@hawk.iit.edu

Hernan Razo
A20436834
hrazo@hawk.iit.edu

Christopher Anthony
A20496584
canthony1@hawk.iit.edu

Introduction

Livshitz et al. discuss how to quantify the inconsistencies of a database that violates integrity constraints. It is crucial to measure these inconsistencies to maintain proper function of typical database needs such as progress indication and action prioritization in cleaning systems. But to choose the correct inconsistency measure, we need to identify the needed properties in the application and understand which ones are least expected in practice.

Livshitz et al. try to answer this question by reviewing previously established measures from both knowledge representation and database communities. Specifically, they define four properties of inconsistency measures in the context of repair systems:

1. **Positivity** - A measure is strictly positive if and only if the database is inconsistent.
2. **Monotonicity** - Inconsistency cannot be reduced if the constraints get stricter
3. **Continuity** - A single operation can have a limited relative impact on inconsistency
4. **Progression** - We can always find an operation that reduces inconsistency

These properties are used to analyze a few inconsistency measures adapted from the KR and Logic literature. It is shown that most of these measures violate at least one of the following properties:

1. I'_{MC} - The set of all self-inconsistencies in a database.
2. I_d - Drastic inconsistency value. It is the indicator function of inconsistency.
3. I_{MI} - MI Shapely Inconsistency. It is the cardinality of the set.
4. I_P - Count of facts that belong to the minimal inconsistent subset (problematic facts)
5. I_{MC} - The set of all maximal consistent subsets of a database
6. consistencies in a database
7. I_R - The minimal cost of a sequence of operations that repairs the database.

Also, they propose a new measure for satisfying all desiderata. This new measure offers a combination of rationality and tractability.

Computational Complexity

Measuring inconsistencies in databases obviously comes at a cost. To measure the computational complexity of each scenario, the authors focused on the class of denial constraints and the special class of functional dependencies.

For I_d, the computational complexity relies on testing consistency. This can be done in polynomial time.

For both I_{MI} and I_P, these can be computed in polynomial time. This is possible by enumerating all the subsets in a database of a bounded size.

For I_{MC} and I'_{MC}, if we have a set of integrity constraints that include only functional dependencies, the set of all self-inconsistencies equal to the number of maximal independent sets of the conflict graph wherein the tuples of the database are nodes and there are edges between every pair of tuples that violate a functional dependency. So, if you represent the set of integrity constraints that include only functional dependencies as a P4-free graph, then the set is computable in polynomial time.

The measure I_R should also be intractable for functional dependencies. So, this measure

should be the size of the minimum vertex cover of the conflict graph. As long as the set of integrity constraints contains one functional dependency per relation, then this too can be computed in polynomial time. Otherwise, the complexity changes.

The Subset Repair System

Livshitz et al. propose a new measure that is both rational and tractable in the case where a constraint is a dependency constraint and operations are tuple deletions. This new measure is described as follows:

Let D be a database, Σ be a finite set of dependency constraints, and let the repair system consist of tuple deletions.

The measure $I_R(\Sigma, D)$ is the result of the Integer Linear Program where each x_i for $i \in ids(D)$, determines whether to delete the i^{th} tuple or not. The solution to the linear relaxation of this Integer Linear Program is a linear program whose last constraint is replaced with "$\forall i \in ids(D) : 0 \leq xi \leq 1$."

Comparing this to the previously stated measure I_R, two databases under I_R and I_R^{lin} are consistent with each other if they have sufficient separation under I_R^{lin}. More formally, for two databases D_1, D_2 we have $I_R^{lin}(\Sigma, D_1) >= \mu \cdot I_R(\Sigma, D_2)$, where μ is the integrality gap of the LP relaxation. The maximum number of tuples involved in a violation of a constraint in Σ gives the upper bound on this integrality gap. Finally, this all implies that $I_R^{lin}(\Sigma, D_1) >= I_R^{lin}(\Sigma, D_2)$.

This new theorem is more desirable than I_R for tuple deletions and dependency constraints because it avoids the inherent hardness of I_R while still fulfilling the desired properties.

Experimental Evaluation

To test all the concepts mentioned above, Livshitz et al. used real-world datasets including Stock, Hospital, Food, Airport, Adult, Flight, and voter and the synthetic Tax dataset. They used a dependency constraint mining algorithm to obtain a set of dependency constraints for each dataset. All measurements were done using Python and the Pandas library. All conflicting pairs of tuples were materialized using SQL.

In terms of hardware, all experiments were done on a server with two Intel Xeon(R) Gold 6130 CPUs with 512 GB of RAM running on Ubuntu 20.04.

Each experiment was repeated five times and only the average times were reported.

Results

The behavior of the measures were evaluated on samples of 10K tuples from each dataset. First, they ran 200 iterations of the CONoise algorithm to add noise to each dataset and computed the measure values at each iteration. Then, they ran the RNoise algorithm until they modified 1% of the values in the dataset.

From running variations with the added noise, it was observed that the drastic measure I_d jumps from zero to one when they introduced the first violation. The I_P measure also behaves in a similar way.

The measure I_R is able to recognize that the dataset contains a single erroneous tuple, and reacts to this small change in a proportional way. This is also seen with the I_R, I_R^{lin} , and I_{MI} measures.

The measure I_{MC} incorporates high computational costs so its behavior was tested on a smaller sample. Specifically, the sample was composed of 100 tuples from each dataset. Other than that, the procedure was the same as the previous tests. Results show that this measure is the least stable.

From analyzing the data, the error rate increases with the number of iterations, affects the values of each measure, but has no effect on actual behavior. Also, the graphs seem similar to each other with varying variables in the noise algorithms. This means that data skew and different distributions of error types do not seem to affect the results. Lastly, to measure the effects of overlap of dependencies, the authors

computed the ratio of dependency constraints that overlapped for every dataset and for every dependency constraint. There was no correlation between the behavior of the measures and level of overlap.