RLEX: Saftey and Data Quality in Reinforcement Learning-based and Adaptive Systems

Sanjay Krishnan
UC Berkeley
sanjaykrishnan@berkeley.edu

1. EXTENDED ABSTRACT

Adaptive online learning is now a crucial part of content recommendation, automation, and A/B testing systems. Example algorithms include Multi-Arm Bandits and Reinforcement Learning, and they are characterized by their ability to iteratively optimize a performance metric by automatically *exploring* a search space. The promise of these systems is that they can adaptively cope with changing data, changing metrics, and can optimize weak signals rather than explicitly labeled data. So far, the database community has largely built systems to support only supervised learning. However, increasing use cases of adaptive learning algorithms—drawing inspiration from recent successes of Google DeepMind on AlphaGo project [1] — will force the community to consider the additional challenges of this domain.

What makes such systems interesting from a data management perspective? The algorithms that underpin these systems are actually relatively straight-forward. For example, a basic Multi-Arm Bandits algorithm can be expressed in a handful of lines of Python code. However, the complexity often lies in the surrounding data infrastructure that integrates the input data, ensures that the predictions are properly and safely executed, and monitoring utilities to track the performance of a stateful system. Surveys of data scientists suggest that this data management infrastructure can actually lock in the very assumptions that the adaptive learning system was trying to avoid in the first place [5].

Declarative APIs for specifying safety and data quality restrictions can lead to more reliable systems and maintainable infrastructure. However, it simply enough to raise an exception when one of these online adaptive systems encounters a violated assertion. For example, it may not be an acceptable solution for an HVAC system that sees spurious data to shutdown. We argue that this problem is analogous to *data cleaning*. In data cleaning, a inconsistent relation that violates a set of constraints is acceptably transformed into a consistent relation (acceptably in the sense that cleaning is rarely perfect). Cleaning an inconsistent relation allows queries to proceed as before, but with some degraded accuracy. We explore whether we can apply a similar philosophy to violated execution invariants in adaptive systems, namely, if a violation quickly and automatically find an acceptable execution that satisfies the invariant. This novel perspective casts safety in adaptive systems as a data cleaning problem–building on existing theory and practice.

Consider an example scenario of HVAC system that adaptively controls the temperature in a large building based on feedback from a array of environment sensors. The developer declares a safety invariant that the temperature should never be set to above $25C$. Due to a sensor failure, spurious input data arrives at the control system causing a temperature control of $29C$. An ideal system should detect this violation, trace the provenance of the violation (i.e., blaming a specific input), and automatically take a sensible action (e.g., clip the temperature control to $25C$ or offer a prediction ignoring the broken sensor).

We build on our prior work in data cleaning [4] and systems for safety-critical surgical robotics [2] to propose RLEX, which is a programming framework for specifying safety and data quality constraints. The architecture includes the following major components:

1. *Constraint Language:* We plan to initially consider domain integrity constraints, i.e., enforcing that every value in input data must conform to a pre-defined domain.
2. *Repair Selector:* Given a violation of a defined domain integrity constraint, the repair selector selects whether to offer a default prediction or clean violated values.
3. *Provenance:* Given a violation, we propose a lineage framework that can trace the error to a given data source. We will have to use time-series transition analysis techniques to determine root causes [3]. These techniques identify common failure modes and states that trigger changes in behavior.

2. REFERENCES

[1] Google deepmind. `https://deepmind.com`.

[2] S. Krishnan, A. Garg, R. Liaw, L. Miller, F. T. Pokorny, and K. Goldberg. HIRL: hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards. *CoRR*, abs/1604.06508, 2016.

[3] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg. Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. In *International Symposium of Robotics Research. Springer STAR*, 2015.

[4] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, T. Kraska, T. Milo, and E. Wu. Sampleclean: Fast and reliable analytics on dirty data. *IEEE Data Eng. Bull.*, 38(3):59–75, 2015.

[5] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young. Machine learning: The high interest credit card of technical debt. 2014.

8th Biennial Conference on Innovative Data Systems Research (CIDR âĂŹ17). January 8-11, 2017, Chaminade, California, USA.
.