Towards Observability for Machine Learning Pipelines

Shreya Shankar
UC Berkeley
shreyashankar@berkeley.edu

Aditya Parameswaran
UC Berkeley
adityagp@berkeley.edu

ABSTRACT

Software organizations are increasingly incorporating machine learning (ML) into their product offerings, driving a need for new ML-centric data management tools. Such tools facilitate the initial development and deployment of ML applications, contributing to a crowded landscape of disconnected solutions targeted at different stages, or components, of the ML lifecycle. A lack of end-to-end ML pipeline visibility makes it hard to address any issues that may arise during a production deployment, such as unexpected output values or lower-quality predictions. In this paper, we introduce our prototype and our vision for MLTRACE, a platform-agnostic system that provides observability to ML practitioners.

1. INTRODUCTION

Organizations, big and small, are devoting increasingly more resources towards developing and deploying applications powered by machine learning (ML). ML applications consist of pipelines that span multiple heterogeneous stages or *components*, such as ETL and model training. To support visibility into one or more components of the ML pipeline, the database community has proposed a variety of solutions, e.g., for identifying data bugs during preprocessing [6, 10], and for logging models and model metadata during training for post-hoc debugging [13, 14, 8, 5]. Additionally, industry solutions such as MLFlow [15] and Weights & Biases [2] have garnered widespread adoption by handling data management issues during the training component of a pipeline. Unfortunately, these piecemeal approaches do not address the problem of end-to-end visibility, thereby making debugging and maintenance difficult.

One promising approach for end-to-end visibility is to employ a holistic ML framework, such as TFX [9] or Overton [11]. Users need to use the framework to implement all components—from data preprocessing to deployment—to benefit from it. This rewrite can be cumbersome to users, who often prefer to reuse existing code and avoid vendor lock-in [7, 1]. Other approaches involve the use of various best practices and understanding of failure case studies [3, 12]; neither of which provide easy-to-use software solutions. Instead, in this work, we propose a *lightweight, platform-agnostic system for end-to-end ML application observability*.

12th Annual Conference on Innovative Data Systems Research (CIDR '22) January 10-13, 2022, Chaminade, USA.

2. ML OBSERVABILITY

Typical software observability solutions present metadata in the form of *logs*, *metrics*, and *traces* and synthesize this information to allow engineers to ask questions about system health. ML applications add new criteria to system health, as performance extends beyond system uptime; it also relies on the quality of predictions or outputs. We use the three aforementioned pillars of software observability as inspiration for the key facets of our ML observability approach, and outline the corresponding research challenges, next.

Logging. ML applications add complexity to logging since they can experience silent failures (e.g., covariate shift or concept drift), which may not be reflected in typical output or error logging statements. Thus, our approach must capture inputs and outputs of intermediate components, both of which can be computationally expensive and bulky to store (e.g., large DNN models).

Monitoring. Monitoring is not straightforward in ML because the success of an ML application depends on model performance, which cannot easily be measured; labels, or true values, are often not provided in real-time in the production endpoint. Even if direct feedback is available (e.g., when a user clicks on an ad), ML metrics, such as F1 and t-test scores, can be expensive to compute at scale. Users should also be able to set alerts, triggers, or constraints to ensure overall ML pipeline health. Determining when data has "drifted" to alert users to retrain models is an unsolved algorithmic problem. Computing simple metrics like the mean and median is a good start but can fail when skew and kurtosis changes. Computing well-known metrics like the Kolmogorov-Smirnov test statistic can be expensive and produce too many false positive alerts [4].

Querying. To debug traditional software applications, engineers typically first inspect a trace, or the end-to-end journey of a data point, to understand the lineage of that request and determine where the bug may lie. This lineage may be straightforward to track when transformations are done in a single framework (e.g., REST), but ML applications are built using heterogeneous stacks of many tools, fragmenting access to information about data flow and provenance.

2.1 MLTRACE

We propose MLTRACE, a lightweight end-to-end framework centered around the execution of individual pipeline components that supports: (*i*) logging of component runs, inputs and outputs, (*ii*) a pluggable library of metrics and alerts to sustain ML performance (*iii*) a flexible querying framework and UI for users to debug their pipelines. With over 300 GitHub stars, MLTRACE has received initial interest from practitioners (`github.com/loglabs/mltrace`) and flexibly interoperates with existing tools used in ML application development.

3. REFERENCES

[1] A. Agrawal et al. Cloudy with high chance of dbms: a 10-year prediction for enterprise-grade ml. In *CIDR'20*, 2020.

[2] L. Biewald. Tracking with weights and biases `www.wandb.com/`, 2020.

[3] E. Breck et al. The ml test score: A rubric for ml production readiness and technical debt reduction. In *Big Data'17*, 2017.

[4] E. Breck, M. Zinkevich, N. Polyzotis, S. Whang, and S. Roy. Data validation for machine learning. In *Proceedings of SysML*, 2019.

[5] R. Garcia et al. Hindsight logging for model training. In *VLDB'21*, 2021.

[6] S. Grafberger, S. Guha, J. Stoyanovich, and S. Schelter. Mlinspect: A data distribution debugger for machine learning pipelines. In *SIGMOD'21*, 2021.

[7] E. Liberty et al. Elastic machine learning algorithms in amazon sagemaker. pages 731–737, 06 2020.

[8] H. Miao, A. Li, L. Davis, and A. Deshpande. Towards unified data and lifecycle management for deep learning. In *ICDE'17*, 2017.

[9] A. N. Modi et al. Tfx: A tensorflow-based production-scale machine learning platform. In *KDD 2017*, 2017.

[10] E. Rezig et al. Dagger: A data (not code) debugger. In *CIDR*, 2020.

[11] C. Ré, F. Niu, P. Gudipati, and C. Srisuwananukorn. Overton: A data system for monitoring and improving machine-learned products. In *CIDR*, 2020.

[12] D. Sculley et al. Hidden technical debt in ml systems. In *NIPS*, 2015.

[13] M. Vartak. Modeldb: a system for machine learning model management. In *HILDA '16*, 2016.

[14] M. Vartak et al. Mistique: A system to store and query model intermediates for model diagnosis. In *SIGMOD '18*, 2018.

[15] M. Zaharia et al. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41:39–45, 2018.