The Machine Learning Database

Datacratic - Montréal, Canada - mldb@datacratic.com

***Abstract*—The Machine Learning Database[1] (MLDB) is a novel open source system for solving big data machine learning problems, from data collection and storage through analysis and the training of machine learning models all the way to the deployment of real-time prediction endpoints.**

I. Introduction

Solving a production big data machine learning problem can usually be split into three dimensions: data storage, data science and machine learning, and model deployment and real-time scoring. While many popular machine learning packages (R, scikit-learn) or execution engines (Hadoop, Spark) can be used to train models, most do not provide a means to run models in a high-throughput, low-latency request-response context. They also assume your data has already been collected and is available in a suitable format. We introduce the Machine Learning Database (MLDB), an open source project released under the Apache license that offers a single end-to-end solution for production-ready big data machine learning.

II. Machine Learning with Big Data

Machine learning training algorithms for tasks such as classification typically exhibit an extremely sublinear relationship between data volume and task performance. The downside of this property is that it can take vast amounts of additional data to increase task performance, but this upside is that the available data can be sampled without material loss of task performance. In addition, machine learning algorithms are by nature difficult to distribute: learning how to generalize about a whole dataset does not lend itself well to divide-and-conquer approaches.

Big Data is frequently taken to mean data that is too big to fit in main memory on a single node, but the increasingly affordable availability of high-memory nodes on a utility-computing basis (e.g. the upcoming 2TB X1 instances on AWS EC2) makes vertical scaling a very attractive approach for machine learning workloads on Big Data, and this is the approach we took in designing MLDB. Focusing our algorithmic optimization efforts on efficiently using high-memory multi-core machines, we can achieve comparable task performance results on large datasets with lower training times on a single node than distributed machine learning systems such as H2O and Spark are able to achieve with multiple nodes, leading to massive improvement in Return on Investment (ROI) on hardware.

III. MLDB overview

Big data machine learning problems decompose into problems of data storage and collection, of model training and model deployment, and of real-time scoring. MLDB offers a unified platform to solve all three of these problems, uniquely marrying the power and expressiveness of the Structured Query Language (SQL) and the ease of integration of the HyperText Transfer Protocol REpresentational State Transfer (HTTP REST) design pattern with Machine Learning techniques.

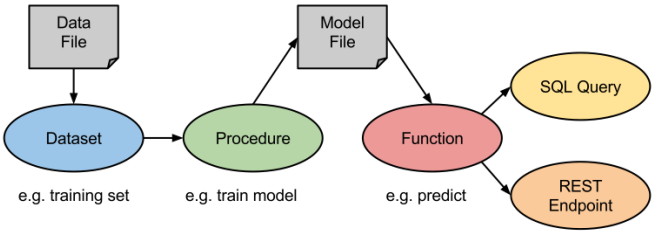

FIG. 1: Relationship between the different MLDB entities

A. Data representation: 3-dimensional sparse matrices

An MLDB dataset is a schema-free, append-only 3- dimensional sparse matrix. Each datum is stored as a tuple (row, column, time, value). Rows are named as well as columns, enabling trivial transposition operations, which are critical to many machine learning algorithms. Rows and columns are automatically created as data is inserted, and this design scales to millions of columns and millions of rows, allowing for the natural representation of a variety of kinds of datasets, as in table I.

Dataset	Row	Column	Value	Time
Behavioral data	User A	Visited page B	binary	timesamp
IoT	Office sensor	Temperature	real	timestamp
Text	Document	Word	counts	timestamp
Metadata	item	tag	binary	timestamp

TABLE I: Examples of the MLDB data representation for different types of data.

B. Data Analysis and Model Training: Mapping Machine Learning to SQL

MLDB provides a comprehensive implementation of the SQL SELECT statement, treating datasets as tables, with rows as relations. This makes MLDB easy to learn and use for data analysts familiar with existing Relational Database Management Systems (RDBMS). MLDB provides a number of extensions to standard SELECT syntax so as to be able to operate on millions of columns, dataset transpositions and the time dimension.

That said, MLDB does not add new keywords to the SQL language to provide machine learning or predictive functionalities. Instead, we use the well-established RDBMS concepts of stored procedures and user-defined functions, as the way to run ML training jobs on MLDB. Supervised and unsupervised training algorithms are exposed as procedures. The output of a training procedure is a model-parameter file, which can be used to instantiate a function, which can then apply that model to new data. Functions can be used in SELECT statements, and

[1] http://mldb.ai

are also immediately available as HTTP endpoints (see section III-C).

Given a trained model loaded as the `my_predictor` function, MLDB supports the following natural expression of a batch application of the model:

```
SELECT my_predictor(features)
FROM observations
```

MLDB provides a plugin Software Development Kit (SDK) so as to be able to add new procedures and functions. Support for common training algorithms such as Generalized Linear Models (GLZ), decision trees and random forests, bagging, boosting and neural networks is built-in, and the plugin mechanism has already been used to add support for Support Vector Machines (SVM) via SVMlib and Deep Learning via TensorFlow.

C. Model Deployment: Mapping SQL to HTTP REST

Applications of machine learning are varied, going to web sites, mobile applications, IoT devices, etc. MLDB employs a RESTful JSON-over-HTTP interface to interact with MLDB, making it trivial to integrate with any language, platform or device.

SQL comprises the Data Definition Language (DDL) which defines the commands CREATE, ALTER, RENAME and DROP, and the Data Manipulation Language (DML) which defines the commands INSERT, UPDATE, DELETE and SELECT. HTTP defines only four verbs: GET, PUT, POST and DELETE.

The DDL is mostly non-applicable to MLDB, due to its schema-free datasets, but its CREATE and DROP commands map well onto PUT/POST and DELETE verbs, respectively. Similarly, the DML is only partly applicable to MLDB as its datasets are append-only, but the INSERT command maps well onto the POST verb and the SELECT command maps well onto the idempotent GET verb, as MLDB does not support the INTO clause.

Finally, as mentioned above, all user-defined functions (i.e. trained models) can also be called directly via HTTP without using SELECT:

```
GET /v1/functions/my_predictor?input=feats
```

MLDB supports tens of thousands of such queries per second with latency below 25ms on commodity hardware. This is a big step forward in terms of ease of deployment: as soon as a model is trained, it is already available to any internet-connected device.

Using HTTP REST as the interface for MLDB has the added advantage of allowing plugins created with the Plugins SDK to create new endpoints so that entire applications can be hosted by MLDB. This includes all backend and frontend logic as well as the ML code, reducing the dependency on external servers and components. This further reduces the complexity of a deployment.

IV. Comparison to other systems

A. Performance

One of the main design hypotheses of MLDB is that a database designed from the ground up for single node ML-only performance could be an order of magnitude faster than a general purpose, distributed system. Figure 2 shows the results of an open performance benchmark[2] comparing many popular ML frameworks on speed and accuracy. While every framework is arguably equivalent when it comes to accuracy, the benchmark validates that MLDB is vastly superior to every other framework in terms of speed, without sacrificing accuracy.

MLDB also scales to much larger datasets. For example, training a random forest on the *Airline On-Time Performance dataset*[3] that contains 150M examples takes several minutes on a single node, as compared to 10 hours for comparable systems[4].

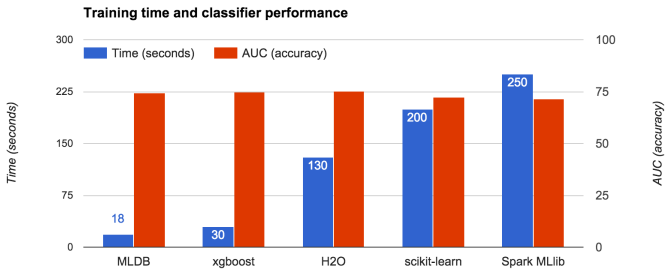

FIG. 2: Benchmark of popular machine learning frameworks on a binary classification task, training a random forest with 100 trees of depth 20 on 1M examples

B. Architecture

A single MLDB installation can be used to solve an entire big data machine learning problem, unlike other systems which can be used to solve only a subset of the problem, as is shown in table II. This integrated approach enables greater agility at the user-level from having to learn only one system, and better ROI due to a tight technical integration leading to lower costs.

Data Management	Training	Deployment
MLDB	MLDB	MLDB
Tachyon	Spark/MLlib	Prediction.io
HDFS	Hadoop/Mahout	Open Scoring
Files on disk	R / Python	Custom code

TABLE II: Comparison of solving a big data machine learning problem using MLDB versus other technology stacks.

V. Conclusion

MLDB is a database designed specifically to handle machine learning workloads. It solves big data machine learning problems end-to-end, from data collection to production deployment, and offers world-class performance yielding potentially dramatic increases in ROI when compared to other Big Data platforms.

[2] https://github.com/szilard/benchm-ml/tree/master/z-other-tools

[3] http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236

[4] https://speakerdeck.com/szilard/benchmarking-machine-learning-tools-for-scalability-speed-and-accuracy-la-ml-meetup-at-eharmony-june-2015