Design of BigQuery ML

Umar Syed
Google

About me

- Researcher in machine learning at Google.
- Involved in design and implementation of BigQuery ML since its inception.
- Not a database expert!

Agenda

What are BigQuery and BigQuery ML?

Design

Implementation

Discussion

Future work

BigQuery

- Google's cloud-based SQL database-as-a-service.

Enterprise data warehouse for analytics

Convenience of standard SQL

Fully managed and serverless

Petabyte-scale storage and queries

Encrypted, durable and highly available

Real-time analytics on streaming data

BigQuery

- BigQuery users typically perform simple analysis like:

```
> SELECT AVG(income) FROM census_data GROUP BY state;
```

BigQuery ML

- With BigQuery ML, they can perform sophisticated analysis like:

```
> CREATE MODEL income_model
    OPTIONS (model_type='linear_reg', labels=['income'])
    AS SELECT state, job, income FROM census_data;

> SELECT predicted_income FROM PREDICT(MODEL 'income_model',
    SELECT state, job FROM customer_data);
```

- Enables **in-database machine learning** for BigQuery users.

BigQuery ML

- Democratizes ML for business customers.
  - Experts in TensorFlow, scikit-learn, etc are rare.
  - Experts in SQL are far more common.

- Avoids slow, cumbersome moving of data to/from of database.
  - Learn ML models directly in BigQuery UI.

Customer use cases

Customer churn prediction

Audience conversion prediction for media planning

Weather-based harsh driving prediction for smart cities

Customer subscription prediction

Traffic prediction for smart cities

Automated IP address threat prediction

Try it yourself at https://cloud.google.com/bigquery/
Send feedback to bqml-feedback@google.com

Agenda

What are BigQuery and BigQuery ML?

Design

Implementation

Discussion

Future work

Google

Design desiderta

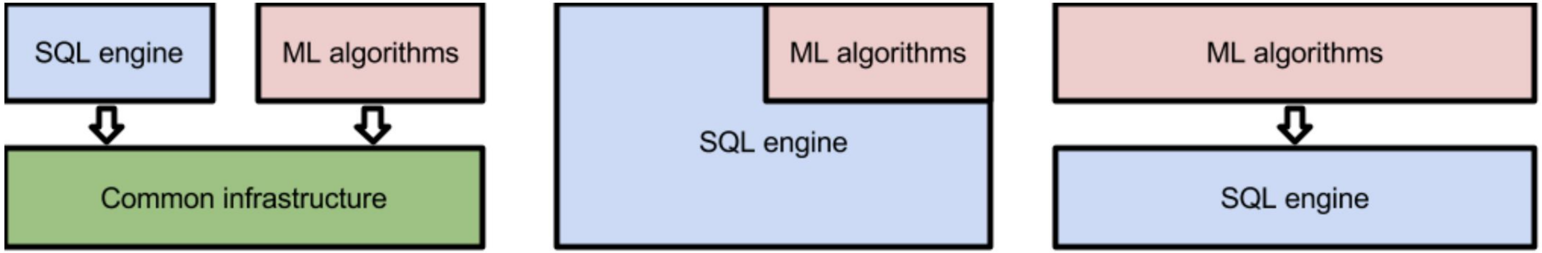
1. **Adaptable to BigQuery infrastructure.** While leveraging its strengths.

2. **Scalable**. No limit on dataset or model size.
   - Should easily handle billions of examples, millions of features.

3. **General purpose**. Able to learn many kinds of ML models.

Design landscape

Published in-database ML systems can be divided into 3 categories:

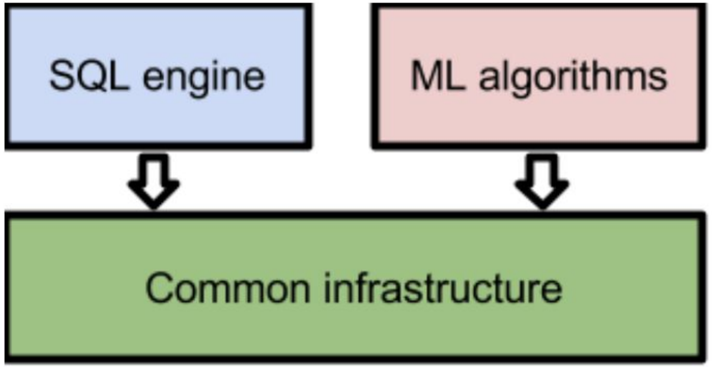


Integrated system

UDA-based system

Pure SQL system

Integrated system



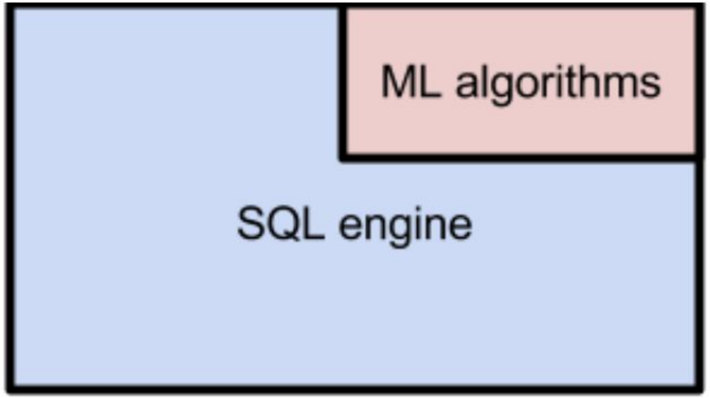
- Query processing engine and ML algorithms are implemented on top of common infrastructure.
- **Example**: Shark, a.k.a., Spark SQL.[1]

[1] Xin, Rosen, Zaharia, Franklin, Shenker, Stoica (2012). Shark: SQL and rich analytics at scale.

Disadvantages of integrated system

- Re-implementing BigQuery was totally infeasible in the short-term.

UDA-based system



- *User-defined aggregate* functions extend the query processing engine to support ML algorithms.
- **Example**: Bismarck[1], part of the MADlib open source library.

[1] X. Feng, A. Kumar, B. Recht, and C. Re (2012). Towards a unified architecture for in-rdbms analytics.

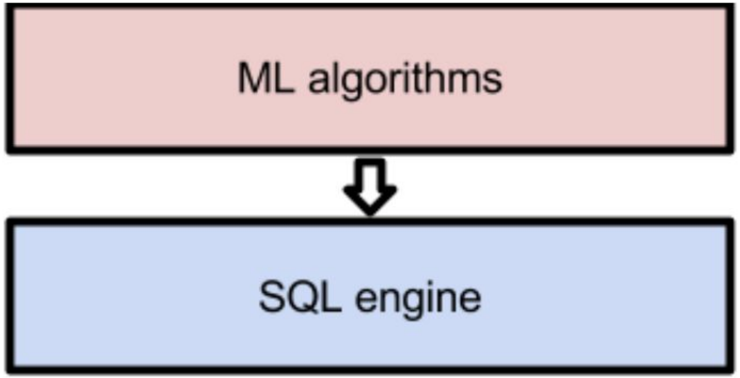
Google

Disadvantages of UDA-based system

- UDAs assume ML model can fit in memory.
  - ML model = State of the UDA.

- UDAs assume invariance to how data is distributed on disk.
  - Can lead to poor performance (we'll see some experiments later).

Pure SQL system



- ML algorithms are implemented in SQL; query processing engine itself is unchanged.
- **Examples**: Clustering[1], Naive Bayes classification.[2]

[1] Ordonez (2006). Integrating k-means clustering with a relational DBMS using SQL.
[2] Pitchaimalai and Ordonez (2009). Bayesian classifiers programmed in SQL.

“Disadvantages” of pure SQL system

- Conventional wisdom held that pure SQL is inadequate for implementing sophisticated ML algorithms.

- From the MADlib[1] developers:

  “The portable core of ‘vanilla’ SQL is often not quite enough to express the kinds of algorithms needed for advanced analytics.”

- And yet **BigQuery ML is a pure SQL system**.

[1] Hellerstein, Schoppmann, Wang, Fratkin, Gorajek, Ng, Welton, Feng, Li, Kumar (2012). The MADlib analytics library or MAD skills, the SQL.

Agenda

What are BigQuery and BigQuery ML?

Design

Implementation

Discussion

Future work

Background: Generalized linear models

- A **generalized linear model** has the form:

$$\mathbf{x} \mapsto p\left(\mathbf{w}^\top \mathbf{x}\right)$$

  where:

  - **x** is an example's *feature vector*.
  - **w** is the model's *weight vector* (or *parameter vector*).
  - p() is the model's *prediction function*.

Training a generalized linear model

- Collect labeled training examples ($\mathbf{x}_1$, y_1), ..., ($\mathbf{x}_n$, y_n).

- Minimize the objective:

$$f(\mathbf{w}) = \sum_{i=1}^{n} \ell\left(\mathbf{w}^\top \mathbf{x}_i, y_i\right)$$

where *loss function* ℓ() measures discrepancy between model's prediction for example $\mathbf{x}_i$ and true label y_i.

Types of generalized linear models

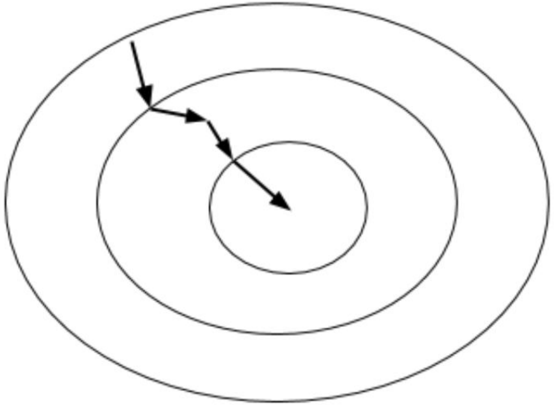
- Many ML models can be expressed as generalized linear models:

Model	Prediction function	Loss function	Label type
Linear regressor	$p(s) = s$	$\ell(s,y) = \frac{1}{2}(s-y)^2$	Real
Binary classifier	$p(s) = 1/(1+\exp(-s))$	$\ell(s,y) = \log(1+\exp(-ys))$	Binary
Poisson regressor	$p(s) = \exp(s)$	$\ell(s,y) = ys - \exp(s)$	Count
Support vector machine	$p(s) = \operatorname{sign}(s)$	$\ell(s,y) = \max(0, 1-ys)$	Binary

Training a generalized linear model

- Objective f(**w**) is minimized via **gradient descent**:

$$\mathbf{w}_{t+1} = \mathbf{w}_{t+1} - \eta \nabla f(\mathbf{w}_t)$$



Training a generalized linear model in BigQuery ML

- Gradient descent implemented as sequence of **pure SQL queries**.

- **Both data and models are represented as tables:**

`data`

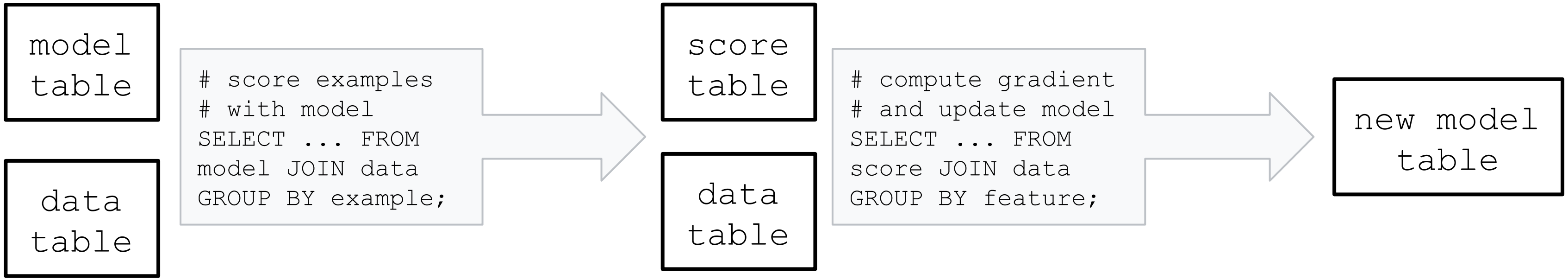
state	**job**	**label**
NY	nurse	65
CA	chef	55
...	...	...

`model`

feature	**weight**
state:CA	+5.7
job:nurse	-3.5
...	...

Model training in BigQuery ML

- Each algorithm iteration issues SQL queries that join model to data, update model, then write model back to disk.



Model training in BigQuery ML

- Query to update model weights:

```
EXPORT new_model AS
SELECT
  feature,
  $update(weight, g, $D(count), $eta,
          $lambda_1, $lambda_2)
    AS weight,
FROM (
  SELECT
    feature,
    SUM($loss_prime(score, label)) AS g,
    ANY_VALUE(weight) AS weight,
    COUNT(*) AS count
  FROM
    data_model_scores
  GROUP BY feature);
```

Model training in BigQuery ML

- Query to compute inner products per example:

```
EXPORT scores AS
SELECT
  data.id AS id,
  SUM(model.weight) AS score
FROM
  data JOIN model
  ON data.feature = model.feature
GROUP BY id;
```

Agenda

What are BigQuery and BigQuery ML?

Design

Implementation

Discussion

Future work

Google

Why a batch method?

- Modern ML algorithms tend to be “incremental”.
    - Many iterations, each processing a few examples.

- But BigQuery ML algorithm is “batch”.
    - Few iterations, each processing every example.

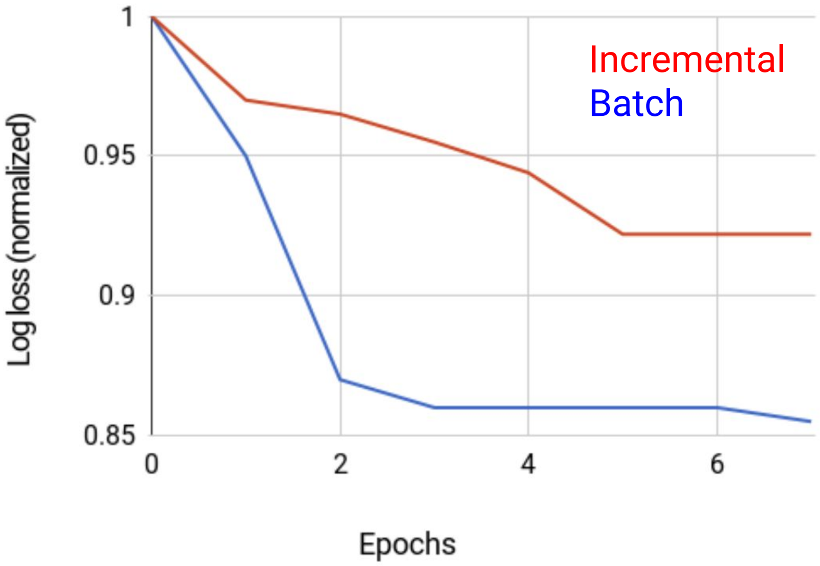
Why a batch method?

- Incremental ML algorithms require efficient random sampling and access.
  - No support for this in BigQuery.

- Batch algorithm + BigQuery's parallelism still yields good performance.
  - Can train model with 2B examples, 10M features in ~ 1 hour.

Why a batch method?

- Batch ML algorithms less sensitive to how data distributed on disk.
  - Batch vs incremental on non-randomly distributed data:



Why a batch method?

- Batch methods can sometimes estimate model **in closed form**.
- Consider linear regression with n examples, d features, and $n \gg d$.
- Learned model weights are $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\mathbf{y}$
  - **X** is (large) n-by-d matrix of feature vectors
  - **y** is (large) n-by-1 vector of labels.
- Batch algorithm:
  - Use distributed matrix multiplication via SQL query to form $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{y}$.
  - In memory: Invert (small) $\mathbf{X}^T\mathbf{X}$ matrix, multiply by (small) $\mathbf{X}^T\mathbf{y}$ matrix.

Future work

- Support for more BigQuery ML functionality to be announced **next week** at Google Cloud Next conference.

https://cloud.withgoogle.com/next/sf/

Google

Thanks!

Questions?