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:

  ```sql
  > SELECT AVG(income) FROM census_data GROUP BY state;
  ```
With BigQuery ML, they can perform sophisticated analysis like:

```sql
> 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

**HEARST newspapers**
- Customer churn prediction
- Audience conversion prediction for media planning
- Weather-based harsh driving prediction for smart cities

**News UK**
- Customer subscription prediction
- Traffic prediction for smart cities
- Automated IP address threat prediction

Try it yourself at [https://cloud.google.com/bigquery/](https://cloud.google.com/bigquery/)
Send feedback to bqml-feedback@google.com
Agenda

What are BigQuery and BigQuery ML?

Design

Implementation

Discussion

Future work
Design desiderata

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.
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

---

Disadvantages of integrated system

- Re-implementing BigQuery was totally infeasible in the short-term.
User-defined aggregate functions extend the query processing engine to support ML algorithms.

Example: Bismarck, part of the MADlib open source library.

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
A generalized linear model has the form:

\[ \mathbf{x} \mapsto p(\mathbf{w}^T \mathbf{x}) \]

where:
- \( \mathbf{x} \) is an example’s feature vector.
- \( \mathbf{w} \) 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 \((x_1, y_1), \ldots, (x_n, y_n)\).

- Minimize the objective:

\[
    f(w) = \sum_{i=1}^{n} \ell(w^\top x_i, y_i)
\]

where loss function \(\ell()\) measures discrepancy between model’s prediction for example \(x_i\) and true label \(y_i\).
Many ML models can be expressed as generalized linear models:

<table>
<thead>
<tr>
<th>Model</th>
<th>Prediction function</th>
<th>Loss function</th>
<th>Label type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regressor</td>
<td>$p(s) = s$</td>
<td>$\ell(s, y) = \frac{1}{2} (s - y)^2$</td>
<td>Real</td>
</tr>
<tr>
<td>Binary classifier</td>
<td>$p(s) = 1/(1 + \exp(-s))$</td>
<td>$\ell(s, y) = \log(1 + \exp(-ys))$</td>
<td>Binary</td>
</tr>
<tr>
<td>Poisson regressor</td>
<td>$p(s) = \exp(s)$</td>
<td>$\ell(s, y) = ys - \exp(s)$</td>
<td>Count</td>
</tr>
<tr>
<td>Support vector machine</td>
<td>$p(s) = \text{sign}(s)$</td>
<td>$\ell(s, y) = \max(0, 1 - ys)$</td>
<td>Binary</td>
</tr>
</tbody>
</table>
Training a generalized linear model

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

$$w_{t+1} = w_{t+1} - \eta \nabla f(w_t)$$
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:

<table>
<thead>
<tr>
<th>data</th>
<th>model</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>feature</td>
</tr>
<tr>
<td>job</td>
<td>state:CA</td>
</tr>
<tr>
<td>label</td>
<td>job:nurse</td>
</tr>
<tr>
<td>NY</td>
<td>...</td>
</tr>
<tr>
<td>nurse</td>
<td>...</td>
</tr>
<tr>
<td>65</td>
<td>state:CA</td>
</tr>
<tr>
<td>CA</td>
<td>job:nurse</td>
</tr>
<tr>
<td>chef</td>
<td>state:CA</td>
</tr>
<tr>
<td>55</td>
<td>job:nurse</td>
</tr>
<tr>
<td>...</td>
<td>feature</td>
</tr>
<tr>
<td>...</td>
<td>state:CA</td>
</tr>
<tr>
<td>...</td>
<td>job:nurse</td>
</tr>
<tr>
<td>...</td>
<td>state:CA</td>
</tr>
<tr>
<td>...</td>
<td>job:nurse</td>
</tr>
</tbody>
</table>
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:

```sql
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
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 $w = (X^T X)^{-1} X^T y$
  - $X$ 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 $X^T X$ and $X^T y$.
  - In memory: Invert (small) $X^T X$ matrix, multiply by (small) $X^T y$ 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/
Thanks!

Questions?