Differentiating Relational Queries

PhD Workshop at VLDB21
Paul Peseux
*Tutors:* T.Paquet, M.Berar, V.Nicollet

August 16, 2021
Figure: Classic Machine Learning Pipeline.
costly data transfer (Schüle 2019)

Figure: Classic Machine Learning Pipeline.
costly data transfer (Schüle 2019)
ML libraries built for computer vision, NLP . . .
→ inadapted to relational data

Figure: Classic Machine Learning Pipeline.
Figure: Classic Machine Learning Pipeline.
Figure: Proposed Pipeline.
Many Machine Learning methods are based on gradient methods.

**Figure:** Gradient Descent, source (Hutson)
Many Machine Learning methods are based on gradient methods.

Figure: Gradient Descent, source (Hutson)

→ To optimize models, relational queries differentiation is missing (Schüle 2019)
Differentiating Relational Queries ⇔ Derivative of the Relational Queries

This is not differential dataflow (Mcsherry 2021)
Figure: What we are looking for

```
SELECT X FROM Observations
should give
SELECT 1 FROM Observations
```

```
SELECT X * X FROM Observations
should give
SELECT 2 * X FROM Observations
```
Outline

1. Context

2. Formalization

3. Tables Relations

4. Automatic Differentiation

5. Implementation

6. Conclusion
Formalization

For the rest of the presentation, optimisation means minimisation and is allowed through gradient descent.

\[ x^* = \arg \min_x f(x) \]

**Figure:** Gradient Descent, source (Hutson)

\( f \) is called *loss*
Formalization

We want to minimize (and thus compute the gradient of):

\[ \text{SELECT } \text{sum(loss)} \text{ FROM Observations} \]

For that we need:
- a framework
- constraints on the query
Minimization is only possible on scalar.

\[
Loss = \sum_{i \in Obs} loss_i = \sum_{i \in Obs} f(data_i)
\]
Minimization is only possible on scalar.

\[ \text{Loss} = \sum_{i \in \text{Obs}} \text{loss}_i = \sum_{i \in \text{Obs}} f(\text{data}_i) \]

**Constraint 1**

*Loss is computed* line by line.
Let’s make it concrete with the Chicago taxi trip dataset.

<table>
<thead>
<tr>
<th>Taxid</th>
<th>Company</th>
<th>Distance</th>
<th>Tips</th>
</tr>
</thead>
<tbody>
<tr>
<td>1226ead8b8629917152b37ff43</td>
<td>Choice Taxi Association</td>
<td>0.3</td>
<td>1.1</td>
</tr>
<tr>
<td>7965168ea516aa1c4437d4ebe4</td>
<td>KOAM Taxi Association</td>
<td>13.2</td>
<td>5.7</td>
</tr>
<tr>
<td>365689b9f3107b807470fe16b7</td>
<td>Taxi Affiliation Services</td>
<td>0.2</td>
<td>1.4</td>
</tr>
<tr>
<td>8c4ce532e3fa081753ea28c1b</td>
<td>Taxi Affiliation Services</td>
<td>1.3</td>
<td>20.0</td>
</tr>
<tr>
<td>627de0f7c9251f731fe27af6bb</td>
<td>Taxi Affiliation Services</td>
<td>2.7</td>
<td>2.0</td>
</tr>
<tr>
<td>705c88d7a216145f6c762aa70</td>
<td>Dispatch Taxi Affiliation</td>
<td>3.7</td>
<td>2.3</td>
</tr>
<tr>
<td>01dfe8a384fbd9173842964e7</td>
<td>Dispatch Taxi Affiliation</td>
<td>1.2</td>
<td>4.0</td>
</tr>
<tr>
<td>493b6af5931ea2c706a82d9d6e</td>
<td>Taxi Affiliation Services</td>
<td>0.9</td>
<td>7.0</td>
</tr>
<tr>
<td>e5a471f52ec43f40471c7e4d0</td>
<td>Choice Taxi Association</td>
<td>2.7</td>
<td>1.0</td>
</tr>
<tr>
<td>9aabfe03b5b0f6742bc86499ea</td>
<td>Choice Taxi Association</td>
<td>13.4</td>
<td>13.0</td>
</tr>
<tr>
<td>e61ce97d61bec30e506e2ff55ea</td>
<td>Chicago Medallion Management</td>
<td>2.2</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**Figure:** Chicago trips dataset, source (Chicago )
Objective: explain the trip’s tip with distance and company "quality".

With Linear Regression as the machine learning model.
Linear Regression on the Chicago dataset

Model

\[ \text{Tip}_{\text{estimated}} = a_{\text{company}} \times \text{distance} + b \]

One slope per company; Intercept is shared among all the taxis.

Figure: Model
Comparing the matrix approach (ML Libraries) and relational one
(M.A) \circ X + b

○ is the point-wise product

Figure: Matrix approach
**Approach**

**Figure:** Relational approach
Approach

Figure: Matrix approach

Figure: Relational approach
Linear Regression on the Chicago dataset

Model

\[ \text{Tip}_{\text{estimated}} = a_{\text{company}} \times \text{distance} + b \]

In SQL it gives
WITH TaxisWithSlope AS (  
SELECT *
FROM Taxis
INNER JOIN Companies
    ON Taxis.company = Companies.company)

SELECT tripId,
   POWER(Estimated - tip, 2) AS Loss
FROM (  
SELECT *
   , TaxisWithSlope.slope * Trips.distance + @intercept AS Estimated
FROM Trips
INNER JOIN TaxisWithSlope
    ON Trips.taxiId = TaxisWithSlope.taxiId )
AS Observations;

SQL query of our model.
Formalization

\[ \text{Trips} = \text{Observations} \]

\[
\text{Loss} = \sum_{t \in \text{Trips}} \text{loss}_t = \sum_{t \in \text{Trips}} f(data_t) = \sum_{t \in \text{Trips}} (a_{\text{comp}_t} \times \text{dist}_t + b - \text{tip}_t)^2
\]

with

\[
f(a, x, b, y) = (ax + b - y)^2
\]
Formalization

$\text{Trips} = \text{Observations}$

$$\text{Loss} = \sum_{t \in \text{Trips}} \text{loss}_t = \sum_{t \in \text{Trips}} f(data_t) = \sum_{t \in \text{Trips}} (a_{\text{comp}_t} \times \text{dist}_t + b - \text{tip}_t)^2$$

with

$$f(a, x, b, y) = (ax + b - y)^2$$

Then it is feasible to compute gradients!

$$\frac{\partial f}{\partial a}, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial b}, \frac{\partial f}{\partial y}$$

**Constraint 2**

$f$ has to be differentiable.
Formalization

\[ f(a,x,b,y) = (ax + b - y)^2 \]

**Figure:** Inputs origin.
Formalization

\[ f(a, x, b, y) = (ax + b - y)^2 \]

Figure: Inputs origin.
WITH TaxisWithSlope AS (
    SELECT *
    FROM Taxis
    INNER JOIN Companies
    ON Taxis.company = Companies.company)

SELECT tripId, 
    POWER(Estimated - tip, 2) AS Loss
FROM (
    SELECT *
    FROM Trips
    INNER JOIN TaxisWithSlope
    ON Trips.taxiId = TaxisWithSlope.taxiId)
    AS Observations;

SQL query of our model.
Figure: Path to Differentiating Relational Queries.

- $Q$: query
- $\mathcal{G}_T$: tables graph
- $f$: loss function

Approach
Outline

1. Context
2. Formalization
3. Tables Relations
4. Automatic Differentiation
5. Implementation
6. Conclusion
Figure: Path to Differentiating Relational Queries.

- $Q$: query
- $G_T$: tables graph
- $f$: loss function

Approach
Definition 1 (Broadcast)

Let’s note “$T_A \rightarrow T_B$” when the primary key of $T_A$ is a foreign key in $T_B$. It is said that $T_A$ broadcasts into $T_B$. 

Companies

Trips

Figure: Graph from our linear regression model.
Definition 1 (Broadcast)

Let’s note ” $T_A \rightarrow T_B$” when the primary key of $T_A$ is a foreign key in $T_B$. It is said that $T_A$ broadcasts into $T_B$.

Tables used in the query with the relationship $\rightarrow$ forms a graph $G_T$. 
Definition 1 (Broadcast)

Let’s note ”$T_A \rightarrow T_B$” when the primary key of $T_A$ is a foreign key in $T_B$. It is said that $T_A$ broadcasts into $T_B$.

Tables used in the query with the relationship $\rightarrow$ forms a graph $G_T$.

Figure: Graph from our linear regression model.
Tables Relations

\[ f(a, x, b, y) = (ax + b - y)^2 \]

Figure: Inputs origin.
Let be

- $T$ a table used in the query
- $T.A$ be a column of $T$
- $a$ the input of $f$ representing $T.A$

If $T$ (transitively) broadcasts into $Observations$ then $a$ the input of $f$ representing $T.A$ is a **scalar**.
Tables Relations

Let be

- $T$ a table used in the query
- $T.A$ be a column of $T$
- $a$ the input of $f$ representing $T.A$

If $T$ (transitively) broadcasts into $Observations$ then $a$ the input of $f$ representing $T.A$ is a **scalar**.

\[
Tip_{estimated} = a_{company} \times distance + b
\]

**Figure:** Graph from our linear regression model.
Approach

Figure: Path to Differentiating Relational Queries.

- $Q$: query
- $G_T$: tables graph
- $f$: loss function
Outline

1. Context
2. Formalization
3. Tables Relations
4. Automatic Differentiation
5. Implementation
6. Conclusion
Figure: Path to Differentiating Relational Queries.

- $Q$: query
- $G_T$: tables graph
- $f$: loss function
- AD: Automatic Differentiation
Automatic Differentiation

$P$ a program that apply the mathematical function $f$ to its inputs.

Automatic Differentiation constructs program the program $P'$ that apply $f'$ to its inputs.
A program $P$ that apply the mathematical function $f$ to its inputs.

Automatic Differentiation constructs program the program $P'$ that apply $f'$ to its inputs.

Figure: Automatic Differentiation.
Automatic Differentiation

- Fortran, C: Tapenade
- Python: Tangent, Myia
- Julia: Zygote
- F#: DiffSharp
- ...

ADSL is closed by differentiation.
Automatic Differentiation

- Fortran, C: Tapenade
- Python: Tangent, Myia
- Julia: Zygote
- F#: DiffSharp
- ...

not differentiating a specific programming language.

define a narrowed programming language: **ADSL**. Similar to (Abadi 2019) (Hu 2020) (Mak 2020).

**ADSL** is **closed by differentiation**
We can use this pipeline to differentiate a function written in any programming language *You just need to pay the price of compilation.*
Implementation

This work has been implemented at Lokad:

- on the DSL *Envision*
- live in production

Optimization through gradient descent is used daily and triggers orders on millions of SKUs.
In this work we’ve presented a framework on automatic differentiation on relational queries.

\[ Q \rightarrow G_T \times f \rightarrow Q' \rightarrow G_T \times f' \]

**Figure:** Path to Differentiating Relational Queries.
This will unlock ML model construction and optimisation in databases.

Figure: Proposed Pipeline.
Conclusion

Thanks for listening!


Link to an example
\[ S \] ::= 
| \langle v \leftarrow e \rangle \quad \text{Variable assignment}
| \langle \text{Cond} (v \Psi P_T P_E \Phi) \rangle \quad \text{Conditional}
| \langle \text{For} (\chi P \Xi) \rangle \quad \text{Loop}
| \langle \text{Return} v \rangle \quad \text{Output of a program}

\[ e \] ::= 
| \langle v \rangle \quad \text{Variable}
| \langle f \rangle \quad \text{Scalar}
| \langle b \rangle \quad \text{Boolean}
| \langle v + w \rangle \quad \text{Variable Addition}
| \langle \text{Call1} \text{ op } v \rangle \quad \text{Function Call}
| \langle \text{Call2} \text{ op } v \ w \rangle \quad \text{Function Call (2 parameters)}
| \langle \text{Param} \ i \rangle \quad \text{Parameter access}
| \langle \text{Const} \ i \rangle \quad \text{Constant access}
| \langle v \leftarrow \beta \rangle \quad \text{Broadcast Projector}
| \langle v \triangleright \alpha \rangle \quad \text{Aggregation Projector}
| \langle \text{Pred} \rangle \quad \text{Predicate}
\( \langle \text{Pred} \rangle ::= \).
\( \langle \text{And } v \ w \rangle \)
\( \langle \text{Or } v \ w \rangle \)
\( \langle \text{Not } v \rangle \)
\( \langle v < w \rangle \)
\( \langle v \leq w \rangle \)
Figure: PolyStar
Realtional versus Math

Figure: Realtional - Math decomposition