DB Systems optimised for ML

Tamino Steinert

University Jena

Abstract

It was demonstrates that machine learning (ML) can be directly integrated into data management systems (DMS) by adding two new optimised operators for gradient descent and automatic differentiation. The integration has positive effects. Data doesn't need to be copied and managed outside a DMS. It can used for model reproducibility and versioning. It can also help by creating multi-user data access and workflows. It enables safe hardware sharing to improve hardware utilisation.

Features of standard Database Systems (DBS)

The most common way to store, manage and access data is with the help of DBSs. These are some key features of standard DBS:

Transactions with ACID (Atomicity, Consistency, Isolation, Durability) property

Multi User Support enables data sharing and collaboration

Scalability and Performance Optimisation ensures that a DBS can handle every workload

Backup and Recovery are easy with offered tools

Data Integrity ensures that data remain accurate, consistent and valid over time

Version Control and Auditing are valuable when managing evolving datasets and ensuring reproducibility

Flawes of standard Machine Learning (ML) Pipelines

Once the data has been retrieved from a database to be used in a processing pipeline with Python, all the **features of a DBS are lost**. The user has to manage the data now carefully and has to ensure data quality, consistency and security. Often, data transfer is accomplished through plain **CSV files**, which results in the serialization of the data.

Move the ML pipeline into the DBS

To move the complete machine learning pipeline into a DBS, a standard DBS (e.g. postgresql) needs to be extended.

Examining the features that a machine learning pipeline requires and their implementation in standard databases. The items marked with ✕ are currently available in standard SQL.

☒ Preprocessing:
- ☒ **Input Parser:** Converts raw data into a usable format for analysis.
- ☒ **Feature Extractor:** Transforms data into meaningful features for models.
- ☒ **Anomaly Detector:** Identifies outliers or irregularities in data.

☒ **Gradient Descent:** Optimisation method that fine-tunes model parameters to minimize errors.

☐ **Automatic Differentiation:** Computes gradients for model parameter updates.

☐ Neural Network:
- ☒ **Representation:** Defines the configuration of layers and connections within the neural network.
- ☐ **Training:** Adjusts network parameters to improve predictions.

DBSs are built for data processing and storage. All preprocessing can be done within them. They even **outperform Keras**[a] and can also be used to detect and debug technical biases. [3]

A model can be optimised within a DBS. This can be achieved by creating a **recursive table** of weights. At each iteration, the weights can be updated by manually derived gradients. [2] This is typically not optimised for training a large model.

```
CREATE TABLE data (x float, y float); INSERT INTO data ...
WITH RECURSIVE gd (id, a, b) AS (SELECT 0, 1::float, 1::float UNION ALL
  SELECT id+1, a-0.05*avg(2*x*(a*x+b-y)), b-0.05*avg(2*(a*x+b-y))
  FROM gd, data WHERE id<5 GROUP BY id, a, b)
SELECT * FROM gd ORDER BY id;
```

Listing 1 Gradient descent using a recursive table (manually derived). [2]

Only **fully connected neural networks can be expressed** using the array data type. Thus, a forward pass consists of matrix multiplications and evaluations of an aggregate function on arrays. [2]

[a] Python library for ML

Optimised Gradient Descent

To optimise the gradient descent, it must use the hardware efficiently. To do this a **separate operator for gradient descent** is proposed. It works by batching and parallelizing the data on GPUs. [2]

```
SELECT * FROM umbra.gd(
  TABLE (SELECT * FROM data),
  TABLE (select 1::float a, 1::float b),
  lambda (x,y) ((y.a * x.x + y.b - x.y)^2),
  1, 0.05, 10);
```

Listing 2 Gradient descent as operator in Umbra [2]

Automatic Differentiation

To get the derivative for each argument of a function, reverse order automatic differentiation can be used to get it all in one pass. [1]

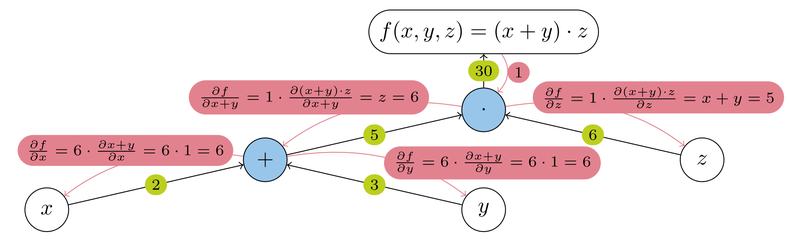

Figure 1 Reverse mode automatic differentiation $f(x,y,z) = (x+y) * z$ on x=2, y=3, z=6 [2]

As for the optimised gradient descent, a separate operator is created that uses **reverse mode automatic differentiation** with **just-in-time compilation**. After compiling a partial derivative, its subexpressions are cached in an LLVM register. They can be reused to generate the remaining partial derivatives, which accelerates runtime execution of the remaining derivatives. [2]

```
SELECT * FROM umbra.derivation(
  TABLE(SELECT 2::float x, 3::float y, 6::float z),
  lambda(a) ((a.x+a.y)*a.z));
-- x y z d_x d_y d_z
-- 2 3 6  6   6   5
```

Listing 3 Automaitc differentiation within SQL in Umbra [2]

Training a Neural Network

Gradient descent and automatic differentiation allows a neural network to be trained by implementing **derivatives of matrix multiplication and activation functions**. [1]

Training a neural network consists mainly of matrix multiplications. Libraries for **basic linear algebra subroutines (cuBLAS) are used to speed up computation on GPUs**. [2]

End-to-end analysis

Figure 2 shows the time is needed to train a neural network on the MINIST dataset using Keras and the DMS Umbra with different approaches. Keras spends a lot of time loading data. This kind of work is not necessary if the data is processed by a DMS that already stores it. It is shown that an **optimised DMS can outperform a standard ML pipeline using Python**. [2]

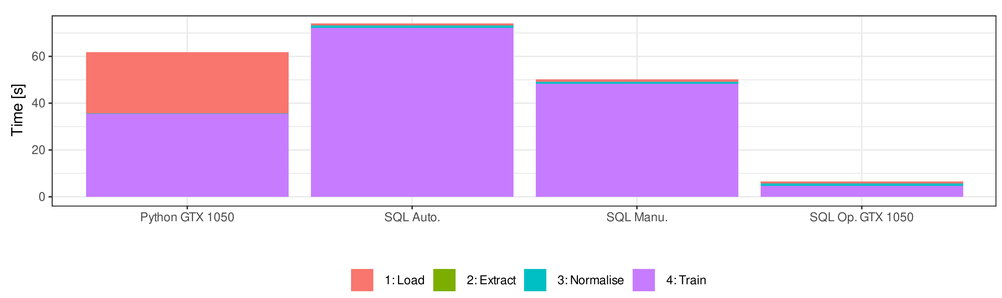

Figure 2 End-to-end analysis of a neural network machine learning pipeline (MNIST, stochastic gradient descent) [2]

References

[1] Iain Murray. Machine learning and pattern recognition (MLPR): Backpropagation of Derivatives. `https://mlpr.inf.ed.ac.uk/2022/notes/w7d_backprop.pdf`, 2022. Accessed: 2023-08-20.

[2] Maximilian E. Schüle, Harald Lang, Maximilian Springer, Alfons Kemper, Thomas Neumann, and Stephan Günnemann. Recursive SQL and gpu-support for in-database machine learning. *Distributed Parallel Databases*, 40(2-3):205–259, 2022.

[3] Maximilian E. Schüle, Luca Scalerandi, Alfons Kemper, and Thomas Neumann. Blue elephants inspecting pandas: Inspection and execution of machine learning pipelines in SQL. In Julia Stoyanovich, Jens Teubner, Nikos Mamoulis, Evaggelia Pitoura, Jan Mühlig, Katja Hose, Sourav S. Bhowmick, and Matteo Lissandrini, editors, *Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023*, pages 40–52. OpenProceedings.org, 2023.