1 PetaByte reported every second by LHC
My Hidden Motivation
Why is it so hard?
Everybody thinks about

Data

...not Queries

Tool complexity

Volume

Variety

Velocity

Explorative

Money

Time

Quality

Multi-hypotheses Pitfall
Brown Projects

DBNav

SciDB

H-Store

Data Tamer

ML base

TupleWare
DB-hard Queries

<table>
<thead>
<tr>
<th>Company_Name</th>
<th>Address</th>
<th>Market Cap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>Googleplex, Mtn. View CA</td>
<td>$210Bn</td>
</tr>
<tr>
<td>Intl. Business Machines</td>
<td>Armonk, NY</td>
<td>$200Bn</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Redmond, WA</td>
<td>$250Bn</td>
</tr>
</tbody>
</table>

SELECT Market_Cap
From Companies
Where Company_Name = “IBM”

Number of Rows: 0

Problem:
Entity Resolution
DB-hard Queries

<table>
<thead>
<tr>
<th>Company_Name</th>
<th>Address</th>
<th>Market Cap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>Googleplex, Mtn. View CA</td>
<td>$210Bn</td>
</tr>
<tr>
<td>Intl. Business Machines</td>
<td>Armonk, NY</td>
<td>$200Bn</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Redmond, WA</td>
<td>$250Bn</td>
</tr>
</tbody>
</table>

```sql
SELECT Market_Cap
FROM Companies
WHERE Company_Name = "Apple"
```

Number of Rows: 0

Problem: 
Missing Data
DB-hard Queries

```
SELECT Image
From Pictures
Where Image contains "professor with beard"
```

Number of Rows: 0

Problem: Missing Intelligence
Easy Queries

SELECT Image
From Pictures
Where Image contains
“professor with beard”
Micro-Task CrowdSourcing

Make Money by working on HITs

HITs - Human Intelligence Tasks - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:
- Can work from home
- Choose your own work hours
- Get paid for doing good work

Get Results from Mechanical Turk Workers

Ask workers to complete HITs - Human Intelligence Tasks - and get results using Mechanical Turk. Get started.

As a Mechanical Turk Requester you:
- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you’re satisfied with the results

Find HITs Now

Fund your account
Load your tasks
Get results

Get Started
Overview

Problem

• How to integrate this new resource “humans” for DB-hard queries
• How to ensure high-quality results

Contributions

• CrowdDb Systems
  • Architecture
  • Query language
  • Query execution
• Quality Control for Sets
Queries in the Open World

CREATE CROWD TABLE PEOPLE(name, age, picture, beard, occupation)
Big Questions

When should we stop asking questions?

Can we estimate query result set size?
Querying the crowd

• SELECT name FROM US_States
  – Experiment runs on Mechanical Turk
  – Avg. “accumulation curve”
Species estimation
Species estimation

• Sample drawn from a population
  – There are $N$ different types within the population, $N$ unknown
  – Analog: worker answers are samples from item distribution

• Calculate query progress
  – based on estimate of $N$
  – Use Chao92 estimator, suitable for open-world
Worker behavior: example

- United Nations member countries (192)
  - Simulated vs. actual cardinality estimate
Worker behavior

\( \rho \) = sampling process with replacement
\( \lambda \) = sampling process without replacement

“Streakers” [Heer10]

Streakers provide a lot of unique answers
Streaker-tolerant estimator

• Chao92 estimator
  – Non-parametric, “frequency of frequencies” statistic
    • $f_1 = \text{singletons}$, $f_2 = \text{doubletons}$, $f_0 = \text{unobserved}$
    • Uses notion of sample coverage: $\hat{C} = 1 - f_1/n$
      \[
      \hat{N}_{\text{chao92}} = \frac{c}{\hat{C}} + \frac{n(1 - \hat{C})}{\hat{C}} \hat{\gamma}^2
      \]

• Adding streaker-tolerance
  – Estimator over-predicts cardinality with abundance of unique answers ($f_1$)
  – Remove $f_1$ outliers
    \[
    \hat{N}_{\text{crowd}} = \frac{cn}{n - \sum_i \min(f_1(i), 2\hat{\sigma}_i + \bar{x}_i)}
    \]
    with coefficient of variance = 0
Streaker-tolerant estimator: results

• “UN member nations” (run 1)
  – Streaker during the middle ameliorated

• “UN member nations” (run 2)
  – Streaker at beginning
  – Other workers shared skewed distribution, yields low cardinality estimate
Now that we have the data…

…how do we analyze it
The Little Secret

Machine Learning is like Teenage Sex
- Everybody talks about it
- Nobody knows how to do it
- Everyone thinks everyone else is doing it
- So everyone claims they are doing it
## The Problem

<table>
<thead>
<tr>
<th>What you want to do</th>
<th>What you have to do</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Build a Classifier</strong></td>
<td>• Learn the internals of ML classification algorithms, sampling, feature selection, X-validation, ...</td>
</tr>
<tr>
<td></td>
<td>• Potentially learn Spark/Hadoop/...</td>
</tr>
<tr>
<td></td>
<td>• Implement 3-4 algorithms</td>
</tr>
<tr>
<td></td>
<td>• Implement grid-search to find the right algorithm parameters</td>
</tr>
<tr>
<td></td>
<td>• Implement validation algorithms</td>
</tr>
<tr>
<td></td>
<td>• Experiment with different sampling-sizes, algorithms, features</td>
</tr>
<tr>
<td></td>
<td>• ....</td>
</tr>
</tbody>
</table>

and in the end

Ask For Help
1\textsuperscript{st} Goal: Simplify the use of ML algorithms

2\textsuperscript{nd} Goal: Make it easier to implement distributed ML algorithms
Collaborators

and others.....
A Declarative Approach to ML
A Declarative Approach to ML

SQL → Result → MQL → Model
Use Cases

Supervised Classification: ALS Prediction

```javascript
var X = load("als_clinical", 2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) = top(doClassify(X, y), 5min)
```

Unsupervised Feature Extraction: Twitter

```javascript
var G = loadGraph("twiqer_network")
var hubs-nodes = findTopKDegreeNodes(G, k = 1000)
var text-features = textFeaturize(load("twitter_tweet_data"))
var T-hub = join(hub-nodes, "u-id", text-features, "u-id")
findTopFeatures(T-hub)
```
Use Cases

Supervised Classification: ALS Prediction

```plaintext
var X = load("als_clinical", 2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) = top(doClassify(X, y), 5min)
```
Hints

Supervised Classification: ALS Prediction

var X = load("als_clinical", 2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) = top(doClassify(X, y, SVM), 5min)
Streaming-like Data Model

Infinite ordered stream of items, being either models (i.e., higher-ordered functions) or tuples
MLbase Architecture

1. **MLI Interface**
   - Interface to simplify implementing distr. ML algorithms

2. **Binders full of algorithms**
   - Allows to add more operators
   - ML Contract + Code

3. **Statistics**
   - About algorithms and data

---

**User**

**Declarative ML Task**

(ML Developer)

**Master Server**

- Meta-Data
- Binders of Algorithms
- Statistics
- Parser
- COML (Optimizer)
- Executor/Monitoring

**Result**

(e.g., fn-model & summary)

---

**Master Optimizer**

Estimates runtime and quality improvement

---

**2** Binders full of algorithms allows to add more operators

**3** Statistics about algorithms and data

---

**1** MLI Interface to simplify implementing distr. ML algorithms
MLbase Architecture

1. MLI (ML Interface) Interface to simplify implementing distr. ML algorithms
2. Binders full of algorithms allows to add more operators
3. Statistics about algorithms and data

User
Declarative ML Task (e.g., fn-model & summary)

ML Contract + Code

Master Server
- Meta-Data
- Binders of Algorithms
- Statistics

Parser

LLP

COML (Optimizer)

Executor/Monitoring

Adaptive Optimizer estimates run-time and quality improvement

Runtime

ML Developer

ML Developer

Master
MLI: Machine Learning Interface

- Shield ML Developers from low-level-details: provide familiar mathematical operators in distributed setting
- Physical independence between ML algorithm and run-time
- Initial abstractions: MLTable, MLMatrix, MLOpt
- Current supported run-times:

  ![Spark](image)

  ![TupleWare](image)
MLTable

- **Flexibility when loading data**
  - e.g., CSV, JSON, XML
  - Heterogeneous data across columns
  - Missing Data
  - Feature extraction
- **Common Interface**
- Supports MapReduce and Relational Operators
- Inspired by DataFrames (R) and Pandas (Python)
MLSubMatrix

• **Linear algebra on local partitions**
  – E.g., matrix-vector operations for mini-batch logistic regression
  – E.g., solving linear systems of equations for Alternating Least Squares

• **Sparse and Dense Matrix Support**
• **Distributed implementations of common optimization patterns**
  - E.g., Stochastic-Gradient-Descent: Applicable to summable ML losses
  - E.g., LBFGS: An approximate 2nd order optimization method
  - E.g., ADMM: Decomposition / coordination procedure
MLbase Architecture

1. MLI Interface to simplify Implementing distr. ML algorithms

2. Binders full of algorithms allows to add more operators

3. Statistics about algorithms and data

User

Declarative ML Task
(e.g., fn-model & summary)

Master Server

Meta-Data

Binders of Algorithms

Statistics

Parser

LLP

COML (Optimizer)

PLP

Executor/Monitoring

Runtime

ML Developer

Adaptive Optimizer estimates runtime and quality improvement
2 Binders Full of Algorithms

Implementation
On top of MLI
(with optimization hints)

Contract
• Type (e.g., classification)
• Parameters
• Runtime (e.g., O(n))
• Input-Specification
• Output-Specification
• ...

ML Developer
Today: Half-Full Binders

• **Regression**: Linear Regression (+Lasso, Ridge)
  • **Classification**: Logistic Regression, Linear SVM (+L1, L2), Multinomial Regression, [Naïve Bayes, Decision Trees]
  • **Collaborative Filtering**: Alternating Least Squares, [DFC]
  • **Clustering**: K-Means, [DP-Means]
  • **Optimization Primitives**: SGD, Parallel Gradient, [L-BFGS, ADMM, Adagrad]
  • **Feature Extraction**: [PCA], N-grams, feature cleaning normalization
  • **Other tools**: Cross Validation, Evaluation Metrics
  • **Released as part of Spark and MLlib**
Example: Alternating Least Squares

<table>
<thead>
<tr>
<th>System</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>20</td>
</tr>
<tr>
<td>Mahout</td>
<td>865</td>
</tr>
<tr>
<td>GraphLab</td>
<td>383</td>
</tr>
<tr>
<td>MLI</td>
<td>32</td>
</tr>
</tbody>
</table>

![Graph showing walltime (s) vs number of machines for different systems]
MLbase Architecture

1. MLI Interface to simplify implementing distr. ML algorithms
2. Binders full of algorithms allows to add more operators
3. Statistics about algorithms and data

User

Declarative ML Task
result (e.g., fn-model & summary)

Master Server
- Meta-Data
- Binders of Algorithms
- Statistics

Parser
- LLP
- PLP

COML (Optimizer)

Executor/Monitoring

ML Developer

ML Contract + Code

Adaptive Optimizer estimates run-time and quality improvement
var X = load("als_clinical", 2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) =
  top(doClassify(X, y), 10min)
Optimization Goals

1. Return **meaningful** results
2. Optimize the **whole processing** pipeline
3. Optimize **quality and time** simultaneously
Current Optimization Approach

Idea: **3-Step** Process

![Diagram showing 3 steps: Expand (Avoid pitfalls), Candidate Generation (Quality), Physical Optimization (Speed)]

**WARNING!**
STILL UNDER HEAVY DEVELOPMENT
var X = load("als_clinical", 2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) =
top(doClassify(X, y), 10min)
var X = load("als_clinical", 2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) = top(doClassify(X, y), 10min)
var X = load("als_clinical", 2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) =
top(doClassify(X, y), 10min)
(2) Generic Logical Plan

- load (als_clinical)
- down-sample
- grid-search
- configure model
  - featurization
    - original, bin, normalized
  - technique
    - SVM, Adaboost
  - kernel
    - RBF, linear, stumps
  - params
    - regularization, rounds
- cross-validate
- train model
- model/data interpretation
  - summary
  - top-1
- (fn-model, summary)

(3) Optimized Plan

- load (als_clinical)
- down-sample 10%
- standard feature normalizer
- create 10-folds
- cross-validation
  - SVM
    - kernel: RBF
    - \( \lambda = 10^6 \)
    - \( \sigma = 1d \times 10^6 \)
- cross-validation
  - linear
- cross-validation
  - stumps
- cross-validation
  - regularization
  - rounds
- cross-validation
  - AdaBoost
  - rounds = 20
- top-1
- train model
- baseline-check: nearest neighbor
- calculate misclassification rate
- baseline-check: most common label
- (fn-model, summary)
DB Optimizer meets ML Parameter Tuning

More than Grid-Search, more than Relational Query Optimization

MLbase cost-based optimization:

**Quality & Time** (=budget)

- Considers **algorithms, system techniques and best practice workflows together**
- **Statistics** about data and **algorithms**
  - Hope to find strong correlation between data statistics and the quality of an algorithm
- Optimization **across steps** (e.g., cleaning, feature extraction, classification,...)
- **Robustness/Avoiding Overfitting & Hypothesis Pitfall** (messing up quality is worse than time in traditional query optimization)
Possible Optimizations (classification)

Relational Optimizations (Top-K Pushdown, Join-Ordering,...)

Static ML Selection Rules
- Imbalance of labels
- SVMs are more sensitive to the scale-parameter than AdaBoost to rounds
- If SVM \(\rightarrow\) normalize data between \([-1, 1]\)
- If data contains outliers \(\rightarrow\) pre-clean data or forego AdaBoost
- ...

Run-Time Optimization Rules
- Caching: If 2\(^{nd}\) run and deterministic, start with previously most successful model
- Set sample-size to fit Input-Data as well as intermediate result in memory
- Partition data according to cross-validation
- ...

Cost-based Optimization Rules
- Materialization and indexing
- Expected quality improvement based on the history
- Consider cost of pre-cleaning, normalization, algorithm complexity,...
- ...

Diagram:
- loading data
- downsampling
- standard feature normalization
- creating 10-folds
- cross-validation
- SVM kernel: RBF
- \(\lambda = 10^6\)
- \(\sigma = \frac{1}{d} \times 10^6\)
- top-1
- train model
- baseline-check: most common label
- baseline-check: nearest neighbor
- calculate misclassification rate
- (fn-model, summary)
Why Optimize? Pitfalls

38/40 = 95%

38/40 = 95%
Why Optimize?

Quality

<table>
<thead>
<tr>
<th></th>
<th>SVM</th>
<th>AdaBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>original</td>
<td>scaled</td>
</tr>
<tr>
<td>a1a</td>
<td><strong>82.93</strong></td>
<td><strong>82.93</strong></td>
</tr>
<tr>
<td>australian</td>
<td>85.22</td>
<td>85.51</td>
</tr>
<tr>
<td>breast</td>
<td>70.13</td>
<td><strong>97.22</strong></td>
</tr>
<tr>
<td>diabetes</td>
<td>76.44</td>
<td><strong>77.61</strong></td>
</tr>
<tr>
<td>fourclass</td>
<td><strong>100.00</strong></td>
<td>99.77</td>
</tr>
<tr>
<td>splice</td>
<td>88.00</td>
<td>87.60</td>
</tr>
</tbody>
</table>

Figure 3: Classifier accuracy using SVM with an RBF kernel and using AdaBoost.

3.6.3 Preliminary Results

To demonstrate the possible advantages of an optimizer just for selecting among different ML algorithms even without considering the system aspect, we implemented a prototype using two algorithms: SVM and AdaBoost. For both algorithms, we used publicly available implementations: LIBSVM [33] for SVM and the ML AdaBoost Toolbox [1] for AdaBoost. We evaluated the optimizer for a classification task similar to the one in Figure 2 with 6 datasets from the LIBSVM website: 'a1a', 'australian', 'breast-cancer', 'diabetes', 'fourclass', and 'splice'. To better visualize the impact of finding the best ML model, we performed a full grid search over a fixed set of algorithm parameters, i.e., number of rounds ($r$) for AdaBoost and regularization ($\alpha$) and RBF scale ($\gamma$) parameters for SVM. Specifically, we tested $r = \{25, 50, 100, 200\}$, $\alpha = \{10^{-6}, 10^{-3}, 1\}$, and $\gamma = \{10^{-6}, 10^{-3}, 1\}$, where $d$ is the number of features in the dataset. For each algorithm, set of features and parameter settings, we performed 5-fold cross validation, and report the average results across the held-out fold.

Table 3 shows the best accuracy after tuning the parameters using grid search for the different datasets and algorithms, with and without scaling the features (the best combination is marked in bold). The results show first that there is no dominant combination for all datasets. Sometimes AdaBoost outperforms SVM, sometimes scaling the features helps, sometimes it does not.

Next we turn to understanding the search problem for the parameters themselves, depicted in Figures 4(a) and 4(b). Figure 4(a) shows, for fixed regularization $\alpha$, the impact of the parameter $\gamma$ in the RBF kernel on the accuracy, whereas Figure 4(b) visualizes the accuracy for varying the number of rounds $r$ for AdaBoost. As shown in Figure 4(a), the choice of $\gamma$ in the SVM problem clearly has a huge impact on quality; automatically selecting $\gamma$ is important. On the other hand, for the same datasets, it appears that the number of rounds in AdaBoost is not quite as significant once $r > 25$ (shown in Figure 4(b)). Hence, an optimizer might decide to initially use AdaBoost - without scaling and with a fixed round parameter - in order to quickly provide the user with a first classifier. Afterwards, the system might explore SVMs with scaled features to improve the model, before extending the search space to the remaining combinations.

The general accuracy of algorithms is just one of the aspects an optimizer may take into account. Statistics about the dataset itself, different data layouts, algorithm speed and parallel execution strategies (as described in the next section) are just a few additional dimensions the optimizer may exploit to improve the learning process. In this project, we will evaluate these freedoms of choice and build the foundation for cost-based (query) optimization for machine learning.
This project builds on the strong foundation of declarative languages and query optimization techniques. run-time, but our system aims beyond a single machine and extends the presented optimization to optimize inference algorithms in a probabilistic DBMS. We leverage these techniques in our a relational-friendly convex-optimization problem, whereas the authors of [70] present techniques physical plan optimization. In [41], the authors show how many ML algorithms can be expressed as training step. Still, the ideas of SystemML are compelling and we might leverage them as part of our to MapReduce. However, SystemML tries to support ML experts to develop e SystemML [46] proposes an R-like language and shows how it can be optimized and compiled down is not on inventing a new run-time for machine learning; instead we will use Spark. memory operations to better support ML algorithms. As mentioned earlier, the goal of this project Recently, there have been e relational model. focus of this project is on the optimization for ML instead of the language integration within the vision by supporting all kinds of ML algorithms, not just predictive models. Furthermore, the natively support predictive models and present a first prototype called Longview. We extend this to MapReduce. However, SystemML tries to support ML experts to develop e

### Why Optimize? Quality

**SVM**

**AdaBoost**

![Figure 4: Parameter Impact](image-url)
Why Optimize? Speed

• Running **1 algorithm** tends to be **I/O bound**
• Idea: **train in parallel** with different algorithms and parameters → Similar to **shared cursors** in DB-world

• Questions:
  – How many models?
    → How to make it cache-aware
  – Impact of sampling?
  – How to leverage modern CPUs, in particular vectorization and CPU pipelining?
Direction

• Released:
  – MLI interface
  – Half-full binders as part of Spark
  – Some simple feature extractors
  – (End-to-end use cases)

• Working on:
  – Optimization techniques
  – Cost-based optimizer
  – Unified language for end users and ML developers
  – Advanced ML capabilities: Time-series algorithms, graphical models, advanced optimizations, online updates, sampling for efficiency
  – Integration into TupleWare: High-Performance analytic platform
  – Visualization
MLBase - Summary

• **MLbase is a first declarative machine-learning system**

• **It simplifies ML in the same way as databases simplify data management**

• **Teaser:** *TupleWare* will integrate **Mlbase** and leverage ideas from *programming languages* to significantly speed-up **ML** and explorative data analysis

---

**Tim Kraska**

tim_kraska@brown.edu