Brown’s Data Management Group
Every day a person sends/receives 140 messages (incl. spam)

- = 52,000 messages per year
- = 500 MB per year

- 27 GB per lifetime per person
- just for mails
1 person takes 500-2000 pictures per year
= 1.5-10GB per year
= 100 – 680 GB per lifetime per person
10 TB pictures added to facebook every day
1 PetaByte reported every second by LHC
Data itself is useless... unless you refine it
Business Applications

Netflix

Google
Why is it so damn hard?
Everybody cares about Data...not Queries

- Tool complexity
- Explorative
- Multi-hypotheses Pitfall
- Money
- Time
- Quality

- Volume
- Variety
- Velocity

Everybody cares about Data...not Queries
Brown Projects

DBNav

SciDB

CrowdDB

H-Store

Data Tamer

MLbase

TupleWare
## The Problem

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

and in the end

**Ask For Help**
A Declarative Approach to ML

SQL → Result

SQL

Result
A Declarative Approach to ML
A Declarative Approach to ML

SQL → Result → MQL → Model
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("twitter_network")
var hubs-nodes = topKDegreeNodes(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

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 $X = \text{load("als\_clinical", 2 \text{ to } 10)}$
var $y = \text{load("als\_clinical", 1)}$
var (fn-model, summary) = \text{top(doClassify}(X, y, \textbf{SVM}), 5\text{min})$
MLbase Architecture

- **Binders full of algorithms** allows to add more operators
- **Statistics** about algorithms and data
- **Distributed Runtime** build for fast (in-memory) iteration

**User** initiates a **Declarative ML Task** (e.g., fn-model & summary) with the **Master Server**.

The **ML Contract + Code** is parsed by the **Parser** into **Meta-Data** and **Binders of Algorithms**.

**Statistics** about algorithms and data are provided to the **COML (Optimizer)**.

The **Executor/Monitoring** component manages the execution and monitoring of the tasks.

**Result** is computed through a process of approximation and continuous refinement.

- **Adaptive Optimizer** estimates runtime and quality improvement.

**Distributed Runtime** is used for fast (in-memory) iteration.
Binders Full of Algorithms

Implementation
Uses patterns with optimization hints

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

Side-products: ML patterns, ontology for ML and we are going to release the binders separately
Today: Half-Full Binders

Common to state-of-the-art algorithms

- SVMs, LogisMc Regression, Naïve Bayes, LogitBoost, Linear Regression, Ridge Regression, LASSO, Matrix FactorizaMon via SGD, DFC, K-Means, DP-Means

- More to come
Optimization

(1) MQL

```
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 = \texttt{load}("als\_clinical", 2 \text{ to } 10) \\
var y = \texttt{load}("als\_clinical", 1) \\
var (\texttt{fn-model}, \texttt{summary}) = \texttt{top}(doClassify(X, y), 10\text{min})
var X = load("als_clinical", 2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) =
top(doClassify(X, y), 10min)
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
- ...

\(\rightarrow\) Cost-based Optimization Rules
- Expected quality improvement based on the history
- Consider cost of pre-cleaning, normalization, algorithm complexity,...
- ...

Load (als_clinical)

Down-sample 10%

Standard feature normalizer

Create 10-folds

Cross-validation

SVM kernel: RBF

\(\lambda = 10^6\)

\(\sigma = 1/d \times 10^6\)

Baseline-check:
- Most common label
- Nearest neighbor

Train model

Calculate misclassification rate

(model-params, cross-validation-summary)

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

\(\rightarrow\) Cost-based Optimization Rules
- Expected quality improvement based on the history
- Consider cost of pre-cleaning, normalization, algorithm complexity,...

Set sample-size to fit Input-Data as well as intermediate result in memory

Parition data according to cross-validation

\(\rightarrow\) Cost-based Optimization Rules
- Expected quality improvement based on the history
- Consider cost of pre-cleaning, normalization, algorithm complexity,...

Set sample-size to fit Input-Data as well as intermediate result in memory

Parition data according to cross-validation

\(\rightarrow\) Cost-based Optimization Rules
- Expected quality improvement based on the history
- Consider cost of pre-cleaning, normalization, algorithm complexity,...
Imbalance of Label

\[\frac{38}{40} = 95\%\]
Imbalance of Label

\[ \frac{38}{40} = 95\% \]

A \rightarrow B

B \rightarrow A

\[ \frac{38}{40} = 95\% \]
We tend to be I/O bound during model training.

- Idea from databases – shared cursor!
- Single pass over the data, many models trained.
- Example – Logistic Regression via SGD.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td></td>
<td>Dog</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td></td>
<td>Cat</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td></td>
<td>Cat</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td></td>
<td>Cat</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td></td>
<td>Dog</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td></td>
<td>Horse</td>
</tr>
<tr>
<td>4</td>
<td>g</td>
<td></td>
<td>Monkey</td>
</tr>
</tbody>
</table>
Why Optimize?

<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></td>
<td></td>
</tr>
<tr>
<td>australian</td>
<td></td>
<td></td>
</tr>
<tr>
<td>breast</td>
<td></td>
<td></td>
</tr>
<tr>
<td>diabetes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fourclass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>splice</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Why Optimize?

SVM

AdaBoost

Scale-factor
- $10^{-6}$
- $10^{-3}$
- 1
- $10^3$
- $10^6$

regularization
- 25
- 50
- 100
- 200
Run-Time

- Support various runtimes
- Hide implementation details through higher-level abstraction
- Develop over time to ML-Algebra
Current Abstractions: MLTable, MLMatrix, MLOpt

• Patterns for building reusable code.
• Consistent interfaces for ML.
  – MLTable – A table-like object containing data.
    • Supports Map/Reduce/MatrixBatchMap/Select/Project/Join
  – MLMatrix – block matrix that is operated on by mappers.
    • Allows for linear algebra on partitions of data.
    • Natural interface for an ML developer.
  – MLOpt – Common ML Optimization Patterns
    • Implemented once and for all.
    • First patterns: SGD, ADMM
MLTable

- Supports feature extraction as series of transformations
- Heterogenous data across columns.
- Common interface for all ML Algorithms.
- Supports Map/Reduce and Relational Operators.
MLMatrix

• Linear Algebra on Local Partitions
• Minibatch Logistic Regression
• Alternating Least Squares
  – Matrix Factorization
  – Solving small systems of linear equations.

\[
A = \begin{bmatrix}
a_0 & a_{-1} & a_{-2} & \ldots & \ldots & a_{-n+1} \\
a_1 & a_0 & a_{-1} & \ddots & & \vdots \\
a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & a_{-1} & a_{-2} & \vdots \\
a_{m-1} & \ldots & \ldots & a_2 & a_1 & a_0
\end{bmatrix}
\]
MLOpt

- Two common patterns for optimization
- Stochastic Gradient Descent
  - Applicable to summable ML loss functions (most!)
- ADMM
  - Decomposition/coordination procedure
Runtime Support

- In-Memory
- Batch-Iterator Model
Runtime Support

- In-Memory
- Batch-Iterator Model

- In-Memory
- Learn various models in parallel, but single algorithm is not distributed
In-Memory
Batch-Iterator Model

In-Memory
Learn various models in parallel, but single algorithm is not distributed

TupleWare
- Language independent
- Order of magnitude faster for ML
- Interactive workbench
- Future: interactive visualization, crowd-sourcing capabilities,…

R

Julia
Python
Case Study: TweetLocalize

- Start with 10% of Twitter Firehose (for one year)
  - Some percentage come with lat/long information.
- Goal – infer author’s State from tweet text
- Featurized data
  - Used Stanford’s NER software.
- Build Multinomial Logistic Regression Model
  - Code now in Mlbase.
Case Study: ImageNet

- Sophisticated ML developers.
  - Home-made infrastructure.
- Want an easy way to train a classifier at scale.
- Built a classifier on 1.1TB of ImageNet data.
  - 1,000,000 labeled images
  - 1000 classes
  - 160000 dense features/image
Case Study: TweetLocalize

- Start with 10% of Twitter Firehose (for one year)
  - Some percentage come with lat/long information.
- Goal – infer author’s State from tweet text
- Featurized data
  - Used Stanford’s NER software.
- Build Multinomial Logistic Regression Model
  - Code now in Mlbase.

TweetLocalize

Inferring Author Location in Social Media

Motivation

Users in Social Media occasionally self-report location. However, these attributes may be incomplete. We aim to infer location of an author based on the content of their tweets, whether or not they provide direct location information.

Approach

We built a multinomial logistic regression model to classify most likely state based on each tweet. Our training dataset is 10% of the Twitter firehose for most of 2012 and early 2013, labeled by the tweets that include location.

We featurized the tweets using the Stanford Named Entity Recognizer, and use tagged words as features. Focusing purely on locations, organizations, and people help model accuracy.

An implementation of Multinomial Logistic Regression was built on Spark using Stochastic Gradient Descent.

Results

Our model yields fairly accurate results, particularly in the presence of named entities. F1 scores generally fall in the 0.4-0.6 range. Errors are dominated by false positives in highly populated states or neighboring states.

The use of named entity recognized features improved results dramatically.
Case Study: ImageNet

- Sophisticated ML developers.
  - Home-made infrastructure.
- Want an easy way to train a classifier at scale.
- Built a classifier on 1.1TB of ImageNet data.
  - 1,000,000 labeled images
  - 1000 classes
  - 160000 dense features/image
MLbase and AT&T

- You have **DATA**, we need **DATA** :-)  
- MLbase can help to make ML (our and YOUR algorithms) **more accessible** inside AT&T → contracts, declarative approach, ...  
- MLbase can help to **make your algorithm scale** → Through design patterns, high-performance runtime, ...  
- Help with **deployment** of ML algorithms  
  - Lifecycle Management  
  - Continuous re-training  
    (our data model already supports it)  
  - Not yet on our agenda yet, but a possible joint project
The Master Plan

• Late summer: 1\textsuperscript{st} release
  – Restricted query language for classification and clustering
  – Simple rule-based query optimizer
  – Half-full Binders / Contracts
  – End-to-end use cases: VMware, LBL, ..
  – Feature extraction for text (and image?) data

• Future
  – Cost/Quality-based optimizer (end of this year)
  – Unified language for End Users and ML developers
  – Advanced ML capabilities: Graph-based algorithms, asynchronous computation, ...
  – Integration into TupleWare: High-Performance analytic platform
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 not only integrate MLbase and CrowdDB, but leverage ideas from Programming Languages to significantly speed-up ML and explorative data analysis