Christopher Ré
Joint work with the Hazy Team
http://www.cs.wisc.edu/hazy
Two Trends that Drive Hazy

1. Data in unprecedented number of formats

2. Arms race for deeper understanding of data

Automated ➔ Statistical **AND** Manage Data ➔ RDBMS

Hazy integrates statistical techniques into an RDBMS

**Hazy Hypothesis:** Handful of statistical operators capture a diverse set of applications.
A relational database management system (RDBMS) is a software artifact that simplifies building applications that use large amounts of data by providing:

1. data storage,
2. sophisticated query processing infrastructure, and
3. a programming model that simplifies concurrent access to data (transactions).

Model: data stored as set of relations (sets of tuples) and transformed via first-order logic statements (SQL).

Hazy: Extend an RBDMS to handle the requirements of applications that use statistical data analysis.
Outline

Three Application Areas for Hazy

Drill Down: One Text Application

Maintaining the Output of Classification

Hazy Heads to the South Pole
Data constantly generated on the Web, Twitter, Blogs, and Facebook

**Extract** and **Classify** sentiment about products, ad campaigns, and customer facing entities.

*Build tools to lower cost of analysis*

Statistical tools for **extraction** (e.g., CRFs) and **classification** (e.g., SVM). Performance and maintenance are data management challenges (DMC)
A physicist **interpolates** sensor readings and uses **regression** to more deeply understand their data.

DMC: Transform and maintain large volumes of sensor data and derived analysis.

Models that extract **entities** from **sequences of words** are similar to models that extract **physical meaning** from sensor readings.
A social scientist wants to **extract** the frequency of **synonyms** of English words in 18\(^{th}\) century texts.

Getting text is challenging! (statistical model errors)

Output of speech and OCR models similar to output of text labeling models

**DMC**: Process large volumes of statistical data
Takeaway and Implications

Statistical processing on large data enables a wide variety of new applications.

*Hazy Hypothesis:* Handful of statistical operators capture a diverse set of applications.

Key challenges are maintenance and performance (data management challenges).
Outline

Three Application Areas for Hazy

Drill Down: One Text Application

Maintaining the Output of Classification

Hazy Heads to the South Pole
Classify publications by subject area
The workflow requires several steps

**Classify** publication by subject area

Simplified workflow

1. Paper references are crawled from the Web.
2. Entities (Papers, Authors,...) are *extracted* and *deduplicated*.
3. Each paper is **classified** by subject area
4. DB is queried to render Web page.

We still use the RDBMS for rendering, reports, etc.

Hazy Evidence: We know names for these operators
An Example of How Hazy Helps
Learning and Inference
Declaratively Specified Together

Tuples In. Tuples out. *Hazy handles the statistical and traditional details.*

CREATE CLASSIFICATION VIEW V(id,label)
ENTITIES FROM Papers
EXAMPLES FROM Example

Declarative SQL-Like Program

Hazy/RDBMS
Hazy Helps with Corrections

Paper 10 is not about query optimization -- it is about Information Extraction

CREATE CLASSIFICATION VIEW V(id,label)
ENTITIES FROM Papers
EXAMPLES FROM Example

Declarative SQL-Like Program

Easy as an INSERT: Update fixes that entry – and perhaps more – automatically.
Design Goals: Hazy should...

• ... look like standard language (SQL)
  – Ideal: application unaware of statistical techniques
  – Build on solutions for classical data management problems

• ... automate routine tasks
  – E.g., updates propagate through the system
  – Eventually, order operators for performance
Where Hazy is Now
In PostgreSQL, we’ve built:

– **Classification**: SVMs, Least Squares

– **Cluster/Equivalence**: Synonyms and Coreference

– **Factor Analysis**: Low-Rank Matrix Factorization

– **Transducers for Sequences**: Text, Audio, & OCR

– **Sophisticated Reasoning**: Markov Logic Networks

Model-based Views (Deshpande et al)

**Beat them at their own game**: Using Hazy, we rebuilt prior systems with higher quality and performance!
Reasoning by Analogy...

*Hazy Hypothesis*: Handful of statistical operators capture a diverse set of applications.

<table>
<thead>
<tr>
<th>Classical RA</th>
<th>Hazy Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>Classification</td>
</tr>
<tr>
<td>Projection</td>
<td>Clustering (Equivalence)</td>
</tr>
<tr>
<td>Join</td>
<td>Factor Analysis</td>
</tr>
<tr>
<td>SQL’s LIKE</td>
<td>Transducer algebra</td>
</tr>
<tr>
<td>Constraints</td>
<td>Markov Logic Networks</td>
</tr>
</tbody>
</table>

Left hand-side (+ set difference) = First Order Logic
Hazy Heads to the South Pole

IceCube
IceCube

IceCube Lab

IceTop
60 Stations, each with
2 IceTop Cherenkov detector tanks
2 optical sensors per tank
320 optical sensors

2010: 79 strings in operation
2011: Project completion, 86 strings

IceCube Array
86 strings including 6 DeepCore strings
60 optical sensors on each string
5160 optical sensors

Amanda II Array
(precurser to IceCube)

DeepCore
6 strings-spacing optimized for low
360 optical sensors

Eiffel Tower
324 m

Digital Optical Module (DOM)
Workflow of IceCube

*In Madison:* Lots of data analysis.

*Via satellite:* Interesting DOM readings

*At Pole:* Algorithm says “Interesting!”

*In Ice:* Detection occurs.
A Key Step: Detecting Track

Mathematical structure used to help track neutrinos is similar to labeling text/tracking/OCR!

Here, Speed \approx Quality
Framework: Regression Problems

\[
\min_x P(x) + \sum_{i=1}^{N} f(x, y_i)
\]

Examples:

1. **Neutrino Tracking**: \(y_i\) is a DOM (sensor) reading
2. **CRFs**: \(y_i\) is (document, labeling)
3. **Netflix**: \(y_i\) is (user, movie, rating)

Others tools also fit this model, e.g., SVMs

**Claim**: General data analysis technique that is amenable to RDBMS processing

<table>
<thead>
<tr>
<th>X</th>
<th>the model</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_i)</td>
<td>A data item</td>
</tr>
<tr>
<td>(f)</td>
<td>Scores the error</td>
</tr>
<tr>
<td>(P)</td>
<td>Enforces prior</td>
</tr>
</tbody>
</table>
Background: Gradient Methods

\[ F(x) = P(x) + \sum_{i=1}^{N} f(x, y_i) \]

Gradient Methods: Iterative.
1. Take current \( x \),
2. Derivate \( F \) wrt \( x \),
3. Move in opposite direction

\[ x^{k+1} = x^k - \nabla F(x^k) \]
Incremental Gradient Methods

\[ F(x) = P(x) + \sum_{i=1}^{N} f(x, y_i) \]

Gradient Methods: Iterative.
1. Take current \( x \),
2. Approximate derivative of \( F \) wrt \( x \),
3. Move in opposite direction

\[
\nabla F(x) \approx \nabla P(x) + N \nabla f(x, y_j)
\]

Sample a single data item to approximate the gradient
Incremental Gradient Methods (iGMs)

Why use iGMs? Provably, iGMs converge to an optimal solution for many problems, but the real reason is:

iGMs are fast.

Technical connection: iGM processing isomorphic to processing a tuple, so RDBMS processing techniques apply.

RDBMS can choose high performance data execution plans using cost models of disk, memory, CPU, etc.
RDBMS abilities are not fully utilized

We may be able to access data orders of magnitude faster at the expense of some bias in the iGM steps. What is the trade off?

How does bias affect convergence of iGMs?
Noise-free Least Squares

\[
\min_{x \in \mathbb{R}^d} \sum_{i=1}^{m} (a_i^T x - b_i)^2 = \min_{x \in \mathbb{R}^d} \|Ax - b\|_2^2
\]

Simplification: noise-free (exists some \(x^*\) s.t. \(Ax^*=b\))

\[
x^{(m)} = x_* + \prod_{i=1}^{m} (I - \alpha a_{\eta(i)} a_{\eta(i)}^T) (x^{(0)} - x_*)
\]

Constant stepsize \(\alpha/2\)

How we select \(\eta(i)\) matters:

1. With replacement, converges at 1/k rate. (fastest in theory)
2. Worst case, deterministic ordering (quadratically slower)
3. Without replacement, empirically fastest (no better than 2)
Gradients Map to Matrix Norms

Update rules explicitly after m steps (epoch)

\[ E_{wr}(x^{(m)} - x_*) = \left( I - \frac{\alpha}{m} \sum_{i=1}^{m} a_i a_i^T \right)^m (x^{(0)} - x_*) \]

For some ordering \( \eta \)

\[ x^{(m)} - x_* = \prod_{i=1}^{m} \left( I - \alpha a_{\eta(i)} a_{\eta(i)}^T \right) (x^{(0)} - x_*) \]

\( \left( \frac{1}{N} \sum_{i=1}^{N} c_i \leq \right)^N \geq \prod_{i=1}^{N} c_i \)

AGMI for scalars

From scalar AGMI one may hope

For any \( \eta \) geometric mean smaller 2-norm arithmetic mean
AGMI fails for two or more matrices

The Arithmetic-Geometric Mean Inequality does not hold for matrices. (fails for 3 matrices)! BUT,

**Theorem** Let $X_1, \ldots, X_m$ be $d \times d$ positive definite matrices. Then

$$
\left\| \prod_{i=1}^{m} X_i \right\| \leq \left\| \frac{d}{m} \sum_{i=1}^{m} X_i \right\|^{m} . \quad (7)
$$

In 2d, worst ordering as $k=1,\ldots,n$ is given by.

$$
X_k = \begin{bmatrix}
\cos^2 \left( \frac{\pi k}{n} \right) & \cos \left( \frac{\pi k}{n} \right) \sin \left( \frac{\pi k}{n} \right) \\
\cos \left( \frac{\pi k}{n} \right) \sin \left( \frac{\pi k}{n} \right) & \sin^2 \left( \frac{\pi k}{n} \right)
\end{bmatrix}
$$
But, we only need the bound to hold on average

**Theorem** For every $N \geq 1$ and $X_k$ defined above,

$$\left\| \frac{1}{N!} \sum_{\pi \in S_N} X_{\sigma(i)} \right\|_2 = -\lambda(N)2^{-N} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

where $F$ is the hypergeometric function and

$$\lambda(N) = \genfrac{[}{]}{0pt}{}{2}{F_3} \begin{bmatrix} 1/2 & -N/2 + 1/2 & -N/2 \\ 1/2 & -N + 1 & 0 \end{bmatrix} \in O(N^{-1})$$

Even in this case, without replacement is asymptotically faster (on average).

Proof exploits symmetry: frame is a representation of $Z_n$.
More applications than a cube of ice!

- **Recommending Movies on Netflix**
  - Experts: Low-rank Factorization.
  - Old SOTA : 4+ hours.
  - In RDBMS : 2.5 hours.
  - Hazy-MM : 2 minutes.

**Buzzwords:** A novel parallel execution strategy for incremental gradient methods to optimize convex relaxations with constraints or proximal point operators.

Same Quality

Prof. Benjamin Recht
Where can DB help optimization?

Scalability: Operate on data sets much larger than memory with reasonable performance

High-level data manipulation layer.
Many apps simple models + lots of data = win.

Cost models for data access -- more than counting steps.
Conclusion

Future of data management is in managing these less precise sources

_Hazy Hypothesis_: Handful of statistical operators capture a diverse set of applications.

Key challenges: performance and maintenance. Hazy attacks this.