MauveDB: Managing Uncertain Data using Probabilistic Models

Amol Deshpande, University of Maryland

(partly joint work w/ Samuel Madden@MIT)
Motivation

- Unprecedented, and rapidly increasing, instrumentation of our every-day world

- Distributed measurement networks (e.g. GPS)

- Wireless sensor networks

- Industrial Monitoring

- RFID
Motivation

- Unprecedented, and rapidly increasing, instrumentation of our every-day world
  - Huge data volumes generated *continuously* that must be processed in *real-time*
  - Typically *imprecise, unreliable and incomplete* data
    - Inherent measurement noises (e.g. GPS)
    - Low success rates (e.g. RFID)
    - Communication link or sensor node failures (e.g. wireless sensor networks)
    - Spatial and temporal biases

- Traditional data management tools are ill-equipped to handle these challenges
Example: Wireless Sensor Networks

Moteiv Invent:
8Mhz uProc, 250kbps 2.4GHz Transreceiver
10K RAM, 48K program/ 512k data flash
Rechargeable Battery (USB)
Light, temperature, acceleration, and sound sensors

A wireless sensor network deployed to monitor temperature
Example: Wireless Sensor Networks

User

select time, avg(temp) from sensors epoch 1 hour

1. Spatially biased deployment ➔ these are not true averages
2. High data loss rates ➔ averages of different sets of sensors
3. Measurement errors propagated to the user

{10am, 23.5}
{11am, 24}
{12pm, 30}

A wireless sensor network deployed to monitor temperature

<table>
<thead>
<tr>
<th>time</th>
<th>id</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>10am</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>10am</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>10am</td>
<td>7</td>
<td>29</td>
</tr>
</tbody>
</table>

{12pm, 70}
Example: Wireless Sensor Networks

User

<table>
<thead>
<tr>
<th>time</th>
<th>id</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>10am</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>10am</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10am</td>
<td>7</td>
<td>29</td>
</tr>
</tbody>
</table>

Impedance mismatch
User wants to query the “underlying environment”, and not the sensor readings at selected locations

A wireless sensor network deployed to monitor temperature
Example: Moving Objects Data

1. GPS Measurement Noise

User wants to query trajectories, not location updates

2. Insufficient data resolution

Monitoring moving objects using GPS
Data Processing Step 1

- Process data using a statistical/probabilistic model
  - Regression and interpolation models
    - To eliminate spatial or temporal biases, handle missing data, prediction
  - Filtering techniques (e.g. Kalman Filters), Bayesian Networks
    - To eliminate measurement noise, to infer hidden variables etc

Temperature monitoring

Regression/interpolation models

GPS Data

Kalman Filters …
Statistical Modeling of Sensor Data

- Little support in database systems --> Database ends up being used as a backing store
  - Much replication of functionality, lack of declarative tools
  - Non-trivial: expert knowledge & MATLAB familiarity required
- Prevents real-time analysis of the data in most cases
- **Goal:** Push statistical modeling of data inside databases

---

Table raw-data

<table>
<thead>
<tr>
<th>time</th>
<th>id</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>10am</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>10am</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10am</td>
<td>7</td>
<td>28</td>
</tr>
</tbody>
</table>

1. Extract all readings into a file
2. Run MATLAB
3. Write output to a file
4. Write data processing tools to process/aggregate the output
Outline

- Motivation
- MauveDB [SIGMOD’06]
  - Abstraction of model-based views
  - Regression- and Interpolation-based views
    - View creation syntax, query processing
    - Implementation and evaluation
- Ongoing work
- Model-driven Data Acquisition [VLDB’04]
Abstraction: Model-based Views

- An abstraction analogous to traditional database views
- Provides independence from the messy measurement details

A traditional database view (defined using an SQL query)

<table>
<thead>
<tr>
<th>acct-no</th>
<th>balance</th>
<th>zipcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>a</td>
<td>20001</td>
</tr>
<tr>
<td>102</td>
<td>b</td>
<td>20002</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>accounts</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A model-based database view (defined using a statistical model)

<table>
<thead>
<tr>
<th>time</th>
<th>id</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>10am</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>10am</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>10am</td>
<td>7</td>
<td>29</td>
</tr>
<tr>
<td>raw-temp-data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User

avg-balances
select zipcode, avg(balance) from accounts group by zipcode

No difference from a user’s perspective

User

temperatures
Use Regression to predict missing values and to remove spatial bias

User
Example: Linear Regression

- Models a **dependent variable** as a function of a set of **independent variables**

Model temperature as a function of \((x, y)\)

\[
\text{temp} = w_1 + w_2 \cdot x + w_3 \cdot x^2 + w_4 \cdot y + w_5 \cdot y^2
\]

Can be used to provide a uniform, holistic view of the temperature in the 2-D space
Grid Abstraction

A Regression-based View

Use Regression to model temperature as:
\[ \text{temp} = w_1 + w_2 x + w_3 x^2 + w_4 y + w_5 y^2 \]

User

Consistent uniform view

Apply regression; Compute “temp” at grid points

raw-temp-data

<table>
<thead>
<tr>
<th>time</th>
<th>id</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>10am</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>10am</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10am</td>
<td>7</td>
<td>29</td>
</tr>
</tbody>
</table>
Grid Abstraction

\[ \text{select } * \]
\[ \text{from rview} \]
\[ \text{where } x = 20 \text{ and } y = 30 \]

\[ \text{time} = 10am \]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{time} & X & Y & \text{temp} \\
\hline
10am & 10 & 10 & 21 \\
10am & 10 & 20 & 18.5 \\
\ldots & \ldots & \ldots & \ldots \\
10am & 50 & 40 & 15.82 \\
\hline
\end{array}
\]

regression-based view

\[
\begin{array}{|c|c|c|c|}
\hline
\text{time} & X & Y & \text{temp} \\
\hline
10am & 2 & 2.5 & 20 \\
10am & 19 & 9 & 17 \\
\ldots & \ldots & \ldots & \ldots \\
10am & 55 & 44 & 15 \\
\hline
\end{array}
\]

raw-temp-data
select *
from rview
where x = 20 and y = 30
epoch 30 min

time = 10am

time = 10:30am
MauveDB System

- Being written using the **Apache Derby** Java open source database system codebase
- Supports the abstraction of **Model-based User Views**
  - Declarative language constructs for creating such views
  - SQL queries over model-based views
  - Keep the models up-to-date as new data is inserted in database
Motivation

MauveDB [SIGMOD’06]
  - Abstraction of model-based views
  - Regression- and Interpolation-based views
    - View creation syntax, query processing
  - Implementation and evaluation

Ongoing work

Model-driven Data Acquisition [VLDB’04]
Creating a Regression-based View

CREATE VIEW
RegView(time [0::1], x [0:100:10], y[0:100:10], temp)
AS
FIT temp USING time, x, y
BASES 1, x, x^2, y, y^2
FOR EACH time T
TRAINING DATA
SELECT temp, time, x, y
FROM raw-temp-data
WHERE raw-temp-data.time = T

Matlab-like syntax used for specifying the grid

Schema of the View

Model to be used

Training data for learning parameters
A Interpolation-based View

CREATE VIEW

\[
\text{IntView}(t [0::1], \text{sensorid} [::1], y[0:100:10], \text{temp})
\]

AS

\[
\text{INTERPOLATE temp USING time, sensorid}
\]

FOR EACH sensorid M

TRAINING DATA

\[
\text{SELECT temp, time, sensorid}
\]

\[
\text{FROM raw-temp-readings}
\]

\[
\text{WHERE raw-temp-readings.sensorid = M}
\]
Analogous to traditional views

So:

- `select * from reg-view`
  - Lists out temperatures at all grid-points
- `select * from reg-view where x = 15 and y = 20`
  - Lists temperature at (15, 20) at all times
- ...

Two operators per view type that support `get_next()` API

- **ScanView**
  - Returns the contents of the view one-by-one
- **IndexView (condition)**
  - Returns tuples that match a condition
    - e.g. return `temperature` where \((x, y) = (10, 20)\)

```
select *
from locations l, reg-view r
where (l.x, l.y) = (r.x, r.y) and r.time = "10am"
```
View Maintenance Strategies

- **Option 1:** Compute the view as needed from base data
  - For regression view, scan the tuples and compute the weights

- **Option 2:** Keep the view materialized
  - Sometimes too large to be practical
    - E.g. if the grid is very fine
  - May need to be recomputed with every new tuple insertion
    - E.g. a regression view that fits a single function to the entire data

- **Option 3:** Lazy materialization/caching
  - Materialize query results as computed

- Generic options shared between all view types
View Maintenance Strategies

- Option 4: Maintain an efficient *intermediate representation*
  - Typically model-specific

- Regression-based Views
  - Say $temp = f(x, y) = w_1 h_1(x, y) + \ldots + w_k h_k(x, y)$
  - Maintain the *weights* for $f(x, y)$ and a *sufficient statistic*
    - Two matrices ($O(k^2)$ space) that can be incrementally updated
  - ScanView: Execute $f(x, y)$ on all grid points
  - IndexView: Execute $f(x, y)$ on the specified point
  - InsertTuple: Recompute the coefficients
    - Can be done very efficiently using the sufficient statistic

- Interpolation-based Views
  - Build and maintain a tree over the tuples in the *TRAINING DATA*
Outline

- Motivation
- MauveDB [SIGMOD’06]
  - Abstraction of model-based views
  - Regression- and Interpolation-based views
    - View creation syntax, query processing
    - Implementation and evaluation
- Ongoing work
- Model-driven Data Acquisition [VLDB’04]
MauveDB: Implementation Details

- Written in the **Apache Derby** Java open source database system
- Support for *Regression*- and *Interpolation-based views*
- Much of the additional code (approx 3500 lines) fairly generic in nature
  - A view manager (for bookkeeping)
  - Query processing operators
  - View maintenance strategies
- Model-specific code
  - Intermediate representation
  - Part of the view creation syntax
MauveDB: Experimental Evaluation

- **Intel Lab Dataset**
  - 54-node sensor network monitoring *temperature, humidity etc*
  - Approx 400,000 readings
  - Attributes used
    - Independent - *time, sensorid, x-coordinate, y-coordinate*
    - Dependent - *temperature*
Contour plot over the data obtained using:

```sql
select *
from reg-view
where time = 2100
```
Average temperature over raw sensor readings

Interpolation

Over 40% missing data

Average temperature over an interpolation-view over the raw sensor readings
Comparing View Maintenance Options

- 50000 tuples initially
- Mixed workload:
  - insert 1000 records
  - issue 50 point queries
  - issue 10 average queries

- Brief summary:
  - Intermediate representation typically the best
  - Among others, dependent on the view properties, and query workload

112.6s
Outline

- Motivation
- MauveDB [SIGMOD’06]
  - Abstraction of model-based views
  - Regression- and Interpolation-based views
    - View creation syntax, query processing
    - Implementation and evaluation
- Ongoing work
- Model-driven Data Acquisition [VLDB’04]
Ongoing and Future Work

- Adding support for views based on dynamic Bayesian networks (e.g. Kalman Filters) [ICDE 08]
  - A very general class of models with wide applicability
  - Generate probabilistic data
- Developing APIs for adding arbitrary models
  - Minimize the work of the model developer
- Query processing, query optimization, and view maintenance issues

- Much research still needs to be done
A Motivating Example

- Inferring “transportation mode”/ “activities” [Henry Kautz et al]
  - Using easily obtainable sensor data (GPS, RFID proximity data)
  - Can do much if we can infer these automatically

Have access to noisy “GPS” data
Infer the transportation mode:
  walking, running, in a car, in a bus
A Motivating Example

- Inferring “transportation mode”/ “activities” [Henry Kautz et al]
  - Using easily obtainable sensor data (GPS, RFID proximity data)
  - Can do much if we can infer these automatically

**Preferred end result:**

*Clean path annotated with transportation mode*
Dynamic Bayesian Network

Use a “generative model” for describing how the observations were generated

Time = t

Transportation Mode: Walking, Running, Car, Bus

True velocity and location

Need conditional probability distributions
e.g. a distribution on (velocity, location)
given the transportation mode

Observed location

Prior knowledge or learned from data
Use a “generative model” for describing how the observations were generated

Transportation Mode:
Walking, Running, Car, Bus

True velocity and location

Observed location

Time = t

Time = t+1
Dynamic Bayesian Network

Given a sequence of observations \((O_t)\), find the most likely \(M_t\)'s that explain it.

Alternatively, could provide a probability distribution on the possible \(M_t\)'s.

\[
\begin{align*}
\text{Time} &= t & \text{Time} &= t+1 \\
\text{Transportation Mode:} & \quad \text{Walking, Running, Car, Bus} \\
\text{True velocity and location} & \quad X_t \quad X_{t+1} \\
\text{Observed location} & \quad O_t \quad O_{t+1}
\end{align*}
\]
### Original noisy GPS data

<table>
<thead>
<tr>
<th>User</th>
<th>Time</th>
<th>Location</th>
<th>Mode</th>
<th>prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>5pm</td>
<td>(x’1, y’1)</td>
<td>Walking</td>
<td>0.9</td>
</tr>
<tr>
<td>John</td>
<td>5pm</td>
<td>(x’1, y’1)</td>
<td>Car</td>
<td>0.1</td>
</tr>
<tr>
<td>John</td>
<td>5:05pm</td>
<td>(x’2, y’2)</td>
<td>Walking</td>
<td>0</td>
</tr>
<tr>
<td>John</td>
<td>5:05pm</td>
<td>(x’2, y’2)</td>
<td>Car</td>
<td>1</td>
</tr>
</tbody>
</table>

### User view of the data
- Smoothed locations
- Inferred variables

```sql
select sum(prob)
group by mode
sliding window 5 min
```
**Correlations**

<table>
<thead>
<tr>
<th>User</th>
<th>Time</th>
<th>Location</th>
<th>Mode</th>
<th>prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>5pm</td>
<td>(x’1, y’1)</td>
<td>Walking</td>
<td>0.9</td>
</tr>
<tr>
<td>John</td>
<td>5pm</td>
<td>(x’1, y’1)</td>
<td>Car</td>
<td>0.1</td>
</tr>
<tr>
<td>John</td>
<td>5:05pm</td>
<td>(x’2, y’2)</td>
<td>Walking</td>
<td>0</td>
</tr>
<tr>
<td>John</td>
<td>5:05pm</td>
<td>(x’2, y’2)</td>
<td>Car</td>
<td>1</td>
</tr>
</tbody>
</table>

**Strong and complex correlations across tuples**

- Mutual exclusivity
- Temporal correlations

**Solution:**

*Probabilistic Databases*

**Ongoing work (w/ Prithviraj Sen and Lise Getoor)**

*Representing correlated data in probabilistic databases [ICDE 2007]*
Outline

- Motivation
- MauveDB [SIGMOD’06]
  - Abstraction of model-based views
  - Regression- and Interpolation-based views
    - View creation syntax, query processing
    - Implementation and evaluation
- Ongoing work
- Model-driven Data Acquisition [VLDB’04]
Sensor networks query processing is typically acquisitional
- Too much data produced to collect all of it centrally
  - Sensing and communication are energy-expensive
- Instead, only acquire if needed by a user query/application

Model-driven Data Acquisition
- Use a probabilistic model to guide the data acquisition
- Can naturally exploit the correlations in the data

(joint work w/ Carlos Guestrin, Sam Madden, Joe Hellerstein, Wei Hong)
BBQ: Model-driven Data Acquisition

Declarative Query
Select nodeID, temp ± .1C, conf(.95)
Where nodeID in {1..6}

USER

Query Results
1, 22.73, 100%
...
6, 22.1, 99%

Observation Plan
{[temp, 1],
[voltage, 3],
[voltage, 6]}

Probabilistic Model

Query Processor

Data
1, temp = 22.73,
3, voltage = 2.73
6, voltage = 2.65

SENSOR NETWORK
BBQ Summary

- Used a *linear multi-variate Gaussian model*
  - Represents the strong *spatial* correlations
  - Captures the *temporal* evolution
- Supported various types of declarative queries
  - SQL-like, augmented with constructs to specify approximations and confidences
- Several *planning* techniques
  - To choose which sensors to observe
- Still many open research challenges
  - Some addressed in follow-up work
Conclusions

- Increasing number of applications generate and need to process uncertain data

- Statistical/probabilistic modeling provide an elegant framework to handle such data
  - But little support in current database systems

- Proposed the abstraction of model-based views
  - Enables declarative querying over noisy, imprecise data
Conclusions

- **MauveDB**
  - Supports the abstraction of Model-based User Views
  - Exploits commonalities to define, to create, and to process queries over such views

- **MauveDB Prototype implementation**
  - Using the Apache Derby open source DBMS
  - Supports Regression- and Interpolation-based views
  - Supports many different view maintenance strategies

- **Model-driven Data Acquisition**
  - Exploits attribute correlations to optimize data acquisition