Exploiting Correlated Attributes in Acquisitional Query Processing

Amol Deshpande
University of Maryland

Joint work with Carlos Guestrin@CMU, Sam Madden@MIT, Wei Hong@Intel Research
Motivation

- Emergence of large-scale distributed information systems
  - Web, Grid, Sensor Networks etc..
- Characterized by high data *acquisition* costs
  - Data doesn’t reside on the local disk; must acquire it
  - Goal: reduce data acquisition as much as possible
Motivation

► Many of these systems exhibit strong data correlations
  ▪ E.g. in sensor networks...
    ► Temperature and voltage
    ► Temperature and light
    ► Temperature and humidity
    ► Temperature and time of day
    ► etc.
Motivation

► Many of these systems exhibit strong data correlations
  ▪ E.g. in sensor networks...
    ► Temperature and voltage
    ► Temperature and light
    ► Temperature and humidity
    ► Temperature and time of day
    ► etc.
  ▪ Observing one attribute \(\Rightarrow\) information about other attributes
Motivation

► Database systems typically ignore correlations

► Our agenda:
  - Exploit the naturally occurring spatial and temporal correlations in novel ways to reduce data acquisition

► Model-driven Acquisition in Sensor Networks [VLDB’04]
  - A new way to look at data management

► This talk:
  - A more traditional query optimization question
  - Significant benefits possible even here
    - Even in traditional domains
Conjunctive Queries

- Queries of the form:
  
  SELECT X1, X2, X3
  WHERE pred1(X1)
  AND pred2(X2)
  AND pred3(X3)

- Common in acquisitional environments
  - Sensor networks:
    - Find sensors reporting temperature between 10°C and 20°C and light less than 100 Lux
    - Are there any sensors not within above bounds?
  - Web data sources:
    - What fraction of people who donated to Gore in 2000 also have patents?
Conjunctive Queries

Queries of the form:

```
SELECT X1, X2, X3
WHERE X1 = x1
    AND X2 = x2
    AND X3 = x3
```

Current approach: *Sequential Paths*

- Choose a single order of attributes to observe, for all tuples

May involve:
- turning a sensor on and sampling, or
- querying a web index over the network

Observe $X_1$

If $X_1 = x_1$, then Observe $X_2$

If $X_1 = x_1$ and $X_2 = x_2$, then Observe $X_3$
Finding Sequential Plans

► Naïve:
  - Order predicates by \( \text{cost}/(1 - \text{selectivity}) \)
    - \( \text{selectivity} \) = fraction of tuples that pass the predicate
  - Ignores correlations
  - Used by most current query optimizers

► Optimal:
  - Find the optimal order considering correlations
  - NP-Hard

► A 4-approximate solution:
  - Greedily choose the attribute that maximizes the return [Munagala, Babu, Motwani, Widom, ICDT 05]
Observation

- Cheap, correlated attributes can improve planning
  - By improving estimates of selectivities
  - Extreme Case - Perfect Correlation: $X_4 \rightarrow X_1$
  - Observing $X_4$ sufficient to decide whether $X_1 = x_1$ is true
    - Would be a better plan if $X_4$ cheaper to acquire
      - E.g. temperature and voltage in SensorNets
    - Unfortunately, perfect correlations rarely exist $\rightarrow$ False +ve’s and –ve’s
      - Approach taken by Shivakumar et al [VLDB 1998]

- Our Approach: Choose different plans based on observed attribute values
  - Query always evaluated correctly
  - Sometimes, observe attributes not involved in query
  - Sounds adaptive, but we generate complete plans \textit{a priori}
  - Applicable in both exact and approximate case
Example

SELECT * FROM sensors
WHERE light < 100 Lux \textit{and} temp > 20\degree C

\begin{itemize}
\item Light > 100 Lux
  \begin{itemize}
  \item Cost = 100
  \item Selectivity = .5
  \end{itemize}
\item Temp < 20\degree C
  \begin{itemize}
  \item Cost = 100
  \item Selectivity = .5
  \end{itemize}
\item Temp < 20\degree C
  \begin{itemize}
  \item Cost = 100
  \item Selectivity = .5
  \end{itemize}
\item Light > 100 Lux
  \begin{itemize}
  \item Cost = 100
  \item Selectivity = .5
  \end{itemize}
\end{itemize}

\textbf{Expected Cost} = 150
Example

A Conditional Plan

\[
\text{SELECT * FROM sensors WHERE light } < 100 \text{ Lux} \text{ and temp } > 20^{\circ}C
\]

Expected Cost
\[
= 110
\]
Problem Statement

Given a conjunctive query of the form

\[ X_1 = x_1 \ and \ X_2 = x_2 \ and \ldots \ and \ X_m = x_m \]

and additional attributes \( X_{m+1}, \ldots, X_n \) not referenced in the query, find the optimal conditional plan

- We will restrict ourselves to *binary* conditional plans

---

Observe \( X \)

Evaluate predicate of form \( X > a \)

Return *false*

Return *true*

Terminal Nodes
Costing a conditional plan

Satisfied Predicates

\( \Gamma \leq a \)

\( \Gamma < a \)

\( \Gamma \geq a \)
Costing a conditional plan

\[
\Gamma \quad < a \quad \Gamma_{<a} \quad \Gamma_{\geq a} \\
\Pi \quad \geq a
\]

Satisfied Predicates
\[= \Pi \text{ and } (X < a)\]

Cost(\(\Gamma\)) = C(X | \Pi) + P(X < a | \Pi) \cdot Cost(\(\Gamma_{<a}\)) + P(X \geq a | \Pi) \cdot Cost(\(\Gamma_{\geq a}\))
Complexity

- Given an oracle that can compute any conditional probabilities in $O(1)$ time, deciding whether a plan with expected cost $< K$ exists is #P-hard
  - Reduction from #3-SAT (counting version of 3-SAT)

- Given a dataset $D$, finding the optimal conditional plan for that dataset is NP-hard
  - Reduction from complexity of finding binary decision trees
Solution Steps

► Plan Costing
  ▪ Need method to estimate conditional probabilities

► Plan Enumeration
  ▪ Exhaustive vs. heuristic
Conditional Probability Estimation

- We estimate conditional probabilities using observations over historical data
- Options:
  - Build a complete multidimensional distribution over attributes
    - Can read off probabilities
    - Very large memory requirements
  - Scan historical data as estimates are needed
    + Minimal memory requirements
    + Can use random samples if datasets too large
  - Build a model that allows quick estimation of probabilities
    - E.g., graphical models
    - Allows reasoning about unobserved events
    - Avoids overfitting
Solution Steps

► Plan Costing
  ▪ Need Method to Estimate Conditional Probabilities

► Plan Enumeration
  ▪ Exhaustive vs. heuristic
Exhaustive Search

- Dynamic programming applicable

Subproblem defined by conditioning predicates ($\Pi$) → Can solve independently
Exhaustive Search

- Dynamic programming applicable
- Complexity: $O(K^{2n})$
- Prohibitive in most cases !!
  - Even if we use branch-and-bound techniques
Greedy Binary Split Heuristic

- Uses optimal sequential plans as base case
- Chooses locally optimal splits to improve greedily

Example Query:
\[ X_1 = 1 \text{ and } X_2 = 1 \]

1. Optimal Sequential Plan

2. Check all possible splits:

Use optimal sequential plans to solve the (smaller) subproblems
Greedy Binary Split Heuristic

- Uses optimal sequential plans as base case
- Chooses locally optimal splits to improve greedily

Example Query:
\[ X_1 = 1 \text{ and } X_2 = 1 \]

1. Optimal Sequential Plan
2. Check all possible splits:
   - Eg: \( X_3 < 10 \)
   - \( X_3 \geq 10 \)
3. Choose locally optimal split
4. Recurse
Evaluation

- Datasets from real deployments
  - Lab
    - 45 motes deployed in Intel Berkeley Lab
    - 400,000 readings
    - Total 6 attributes; 3-predicate queries
  - Garden-11:
    - 11 motes deployed in a forest
    - 3 attributes per mote, temperature, voltage, and humidity.
    - Total 34 attributes; 33-predicate queries
    - Queries are issued against the sensor network as a whole
  - Also experimented with Garden-5, and synthetic datasets

- Separated test from training
- Randomly generated range queries for a given set of query variables
- Java implementation, simulated execution based on cost model
  - Costs represent data acquisition only
Example Plan: Lab Dataset

Query: SELECT * FROM sensors WHERE humidity in [36.5%, 42.5%] AND temp in [17.8°C, 19.8 °C] AND light in [593 lux, 2093 lux]

{H,L,T} denotes a sequential plan that samples Humidity, then Light, then Temperature
Comparing Naive and Heuristic-10

Fraction of Experiments

<table>
<thead>
<tr>
<th>Performance Gain</th>
<th>Fraction of Experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0.9</td>
<td>1.2</td>
</tr>
<tr>
<td>&gt;1</td>
<td>1.0</td>
</tr>
<tr>
<td>&gt;1.1</td>
<td>0.8</td>
</tr>
<tr>
<td>&gt;1.5</td>
<td>0.6</td>
</tr>
<tr>
<td>&gt;2</td>
<td>0.4</td>
</tr>
<tr>
<td>&gt;2.5</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Comparing CorrSeq and Heuristic-10

Fraction of Experiments

Performance Gain

>0.85  >1   >1.1  >1.3  >1.5
Extensions

► Probabilistic queries with confidences

► General queries
  ▪ E.g. Disjunctive queries

► A large class of Join queries
  ▪ E.g. “Star” queries with K-FK join predicates

► Existential queries
  ▪ E.g., “tell me k answers to this query”
    ▪ Can order the observations so that tuples most likely to satisfy are observed first

► Adaptive conditional planning
  ▪ With *eddies*
Conditional Planning for Probabilistic Queries

- Basic idea similar
  - Cost computation is different; typically requires numerical integration

Conditional Planning with BBQ

![Graph showing energy (mJ) vs. confidence for different planning methods involving BBQ]
Conclusions

► Large-scale distributed information systems
  ▪ Must acquire data carefully
    ► High disparate acquisition costs
  ▪ Exhibit strong temporal and spatial correlations
► Conditional planning
  ▪ Change plans based on observed attribute values
  ▪ Significant benefits even for traditional tasks
► Many other opportunities to exploit such correlations...
Questions?