CompSci 590.6

Understanding Data:

Theory and Applications

Lecture 11

Probabilistic Databases

Part-I

Instructor: Sudeepa Roy

Email: *sudeepa@cs.duke.edu*

What did we learn so far?
What will we learn?

DB Systems — DB Systems + Theory — DB Theory

Data Cube
Association rule mining

Provenance, Why-not,
Deletion propagation

Probabilistic,
Incomplete,
Inconsistent DB

Causality in DB, Stat, AI

Crowdsourcing
Usability

Systems for analytics
ML, Visualization, Large-scale

Lectures 11 and 12

- Probabilistic databases
  - Introduction
  - Simple tuple independent model
  - Query evaluation
  - Complexity (#P-hardness)

- Other uncertain data model

- Material and acknowledgement:
  1. Probabilistic database book, Suciu-Olteanu-Re-Koch (up to chapter 5)
  2. Dr. Benny Kimelfeld's course on uncertain data: http://webcourse.cs.technion.ac.il/236605/Spring2015/
  3. EDBT/ICDT 2011 keynote by Dr. Dan Suciu: http://homes.cs.washington.edu/~suciu/talk-icdt2011.pdf
  4. Papers listed on the website

Uncertain Data

- Unreliable data acquisition processes and noisy sources lead to uncertain data
  - Surveys
  - Crowd
  - Faulty sensors
  - Automatic text processing
  - J. Doe: John? Jerry? Jacob? Jack?

Prob. DB book
Suciu-Olteanu-Koch-Re

Example

- NELL : Never Ending Language Learner (CMU)
  - http://rtw.ml.cmu.edu/rtw/
  - Running from 2010
  - (Feb 2011) extracted 537k tuples of the form (entity, relation, value)
    - E.g. (Sony, ProducesProduct, Walkman)
  - "Belief/confidence" with each tuple
  - 87% of tuples had probability < 1.0 (= uncertain)
  - Cannot just remove them (valuable info)

- Need a DBMS to understand and process uncertain data

Levels of Uncertainty

- Tuple-level
  - Each tuple is a random variable
  - E.g. NELL
  - Every tuple has an associated belief/confidence
- Attribute-level uncertainty
  - Value of an attribute is a random variable
  - Each choice has an associated probability Pr[A = a]

Probabilistic Databases

- Uncertain Data

- How to
  - Conceptualize?
    - Semantic
  - Represent and store?
    - Syntax
    - Assumptions
    - Restricted uncertain data models
  - Evaluate Query?
    - Semantic
    - Complexity

Prob DB: Possible World Semantics

- The database instance can be in one of several states
  - Each state has a probability
- Prob DB D
  - States
  - D1: p1
  - D2: p2
  - ....
- $\sum_i p_i = 1$

Prob DB: Possible World Semantics

- A probabilistic database is
  - a probability space D= (**W**, P)
  - P: **W** -> (0, 1]
  - S.T. $\sum_{W \in \mathbf{W}} P(W) = 1$

- $D = (R_1, \ldots, R_k)$
- $W = (W^1, \ldots, W^n)$
- $W^i = R_1^i, \ldots., R_k^i$
- The marginal probability of a tuple = tuple confidence
  - $P(t \in R_j) = \sum_{t \in Rij, \mathbf{i=1..n}} P(W^i)$

- What is a good representation?
  - Nell has 537k "uncertain tuples"
  - 2^{537000} states/possible worlds!
- What is the query semantic?

Query

- Union of Conjunctive Queries

- $Q := R(x1,\ldots,xk) \mid \exists x.Q \mid Q1 \wedge Q2 \mid Q1 \vee Q2$
  - Base relationn | project | join | union

- $Q(x, y) := R(x)\ S(x, y)\ T(y)$
  - Not shown $\wedge$

- $Q() := \exists x \exists y\ R(x)\ S(x, y)\ T(y)$
- $Q() := \exists x \exists y\ R(x)\ S(x, y) \vee \exists x \exists y\ S(x, y)\ T(y)$
  - Boolean query (answer is T or F)
  - Considered in this lecture wlog. (Why?)

Two Query Semantics

- Input D= (**W**, P), Query Q
  - W = (W^1, …, W^n)

- "Possible answer set" semantic
  - Output: (Q(W^1), …, Q(W^n))
  - Too many answers
  - But "compositional"
    - another query Q'
    - Output (Q'(Q(W^1)), …, Q'(Q(W^n)))
- "Possible answers" semantic
  - Output Q(D), a single set of tuples with a distribution
  - Much smaller
  - But not compositional
    - We lost track of how they were produced

What are Prob DB systems?

- Prototypes in academia:
  - MayBMS (Oxford&Cornell)
  - Trio (Stanford)
  - MystiQ (UW)
  - ProbDB (Maryland)
  - Orion (Purdue)
- NO commercial systems
  - We do not know how to build scalable prob db systems
  - Query evaluation in prob db is computationally hard
  - Even for tuple-independent Prob DB

Tuple Independent Prob DB

Boolean query **Q:** **∃ x ∃ y R (x) ∧ S (x, y) ∧ T(y)**

	R	
x_1	a1	0.3
x_2	a2	0.4

	S		
y_1	a1	b1	0.7
y_2	a1	b2	0.5
y_3	a2	b2	0.2

	T	
z_1	b1	0.2
z_2	b2	1.0

Provenance $F_{Q,D} = x_1y_1z_1 + x_1y_2z_2 + x_2y_3z_2$

- $x, y, z \in \{0, 1\}$ random variables with probability in (0, 1]
- Pr[$\mathbf{F}_{Q,D}$] = the probability that query Q is true on database D

  $= \sum_{D' \in \mathbf{W},\ Q(D') = T} P(D')$

– Can use Pr[yz] = Pr[y] Pr[z] and Pr[y + z] = 1 – (1 – Pr[y])(1 – Pr[z])

– Compact representation that matches the possible world semantic

Query Evaluation in Tuple Independent Prob DB

Boolean query **Q:** $\exists x \exists y\, R(x) \wedge S(x,y) \wedge T(y)$

	R	
x_1	a1	0.3
x_2	a2	0.4

	S		
y_1	a1	b1	0.7
y_2	a1	b2	0.5
y_3	a2	b2	0.2

	T	
z_1	b1	0.2
z_2	b2	1.0

Provenance $F_{Q,D} = x_1y_1z_1 + x_1y_2z_2 + x_2y_3z_2$

- Step 1: Compute provenance $\mathbf{F_{Q,D}}$
  - Easy = poly-time "data complexity"
- Review: Data vs. Query Complexity
- Step 2: Compute $Pr[F_{Q,D}]$
  - #P-Hard in general
  - There are "easy formulas", e.g. read-once formulas $x(y + z)$

FO Formula vs. Propositional Formula

	R	
x_1	a1	0.3
x_2	a2	0.4

	S		
y_1	a1	b1	0.7
y_2	a1	b2	0.5
y_3	a2	b2	0.2

	T	
z_1	b1	0.2
z_2	b2	1.0

Boolean query Q: $\exists x \exists y\, R(x) \wedge S(x, y) \wedge T(y)$

- First-Order formula

Provenance $F_{Q,D} = x_1y_1z_1 + x_1y_2z_2 + x_2y_3z_2$

- Propositional formula
  - Equiv. to $R(a_1)S(a_1, b_1)T(b_1) \vee R(a_1)S(a_1, b_2)T(b_2) \vee R(a_2)S(a_2, b_2)T(b_2)$
  - "Grounding" of the FO formula

Model Counting Problem

Model counting

- Given a propositional formula ϕ, count the number of satisfying assignments $\#\phi$
  - E.g. $\phi = xy + yz$, $\#\phi = 3$
  - Aside: the above formula is read-once, i.e. has a read-once form: $y(x+z)$ where every variable appears exactly once. Discussion on board.
  - Model counting is easy (poly-time) for read-once formulas

Weighted model counting/probability computation

- Assuming independence and given Pr[x] for all variables x in ϕ, compute $\Pr[\phi]$
  - As hard as model counting
  - Assume weight = ½ for all variables. Then $\#\phi = 2^n \Pr[\phi]$
  - Note: 2^n is represented using n bits, so multiplication in poly-time

#P

- A complexity class introduced by Valiant (1979)
- Given a poly-time non-deterministic Turing maching, compute the #accepting computation
- Model counting problem: #SAT = compute #ϕ for a formula ϕ is in #P
  - #SAT answers SAT
  - Check if #ϕ > 0
- #P-hard problems: #3SAT, #2SAT, #2DNF
- Note: 3SAT is NP-hard but, DNF, 2SAT are not

Reduction from PP2DNF

- PP2DNF:
  - A propositional formula F is a *Positive, Partite, 2DNF* if $F = \bigvee_{i,j} Xi\ Yj$
  - **Example:**
    $F = X1\ Y1 \vee X1\ Y2 \vee X2\ Y3 \vee X2\ Y4 \vee X2\ Y5$

- For PP2DNFs ϕ, #ϕ is #P-hard (Provan-Ball'83)
- Follows that prob. Query evaluation for H_0 = R(x) S(x,y) T(y) is #P-hard

- Reduction on whiteboard

Extensional vs. Intensional Query Evaluation

R
1
2

S	
1	1
1	2
2	1
2	2

- Consider
  - Query Q():- R(x) S(x, y), Database D
  - Grounding $F_{\mathbf{Q,D}} = R_1S_{11} + R_1S_{12} + R_2S_{21} + R_2S_{22}$
- Extensional query evaluation
  - Entirely guided by query expression Q
  - Computes a "safe plan" if possible – works for all D
  - Possible only for some query (like above)
- Intensional query evaluation
  - First compute F, then compute Pr[F]
  - Possible for all queries
  - Can perform worse than extensional query evaluation in some cases

- Aside: extensional and intensional databases

Dichotomy [Dalvi-Suciu]

A series of papers

- VLDB'04 (10-years test-of-time award in VLDB'14)
- PODS '07, '10
- JACM '12 (includes all), also the book

- For any UCQ Q
  - Either for all D, Pr[Q(D)] can be computed in poly-time
  - Or, evaluation of Pr[Q(D)] is #P-hard
  - JACM 2012, PODS 2010
  - Uses "Mobius ring"

- A simpler proof today/next lecture from VLDB '04
  - Dichotomy for "CQ without self-join"
  - Q(): - R(x, y)R(y, z) query with self-join
  - Q():- R(x, y) S(y, z) no self-join (no repeated relation symbol)
  - Notation: CQ-

Hierarchical Query

- Consider CQ- Q
  - E.g. Q1() :- R(x) S(x, y) T(y), Q2() :- R(x) S(x, y)
- For a variable x $\in$ vars(Q),
  - Let Atoms(x) = {α $\in$ Atoms(Q) | x $\in$ vars(α)}
  - In Q1, Atoms(x) = {R, S}, Atoms(y) = {S, T}
  - In Q2, Atoms(x) = {R, S}, Atoms(y) = {S}
- Hierarchical query Q: If for every two variables x and y in Q, at least one below holds:
  - Atoms(x) $\subseteq$ Atoms(y)
  - Atoms(y) $\subseteq$ Atoms(x)
  - Atoms(x) $\cap$ Atoms(y) = $\varnothing$
- Q2 is hierarchical, Q1 is not
- A root variable of Q is a variable x $\in$ vars(Q) such that
  - Atoms(x) is maximal w.r.t. set containment
  - Which are the root variables in Q2
- If Q is hierarchical, then every subquery of Q (subset of Q's atoms) is hierarchical

Dichotomy for CQ-

- Hierarchical query: poly-time
  - By extensional evaluation

- Not hierarchical: #P-hard
  - Step1: $H_0() :- R(x)\ S(x, y)\ T(y)$ is hard
    - proved
  - Step2: Any non-hierarchical query reduces to H_0

- To be continued in Lecture 12