CompSci 590.6
Understanding Data:
Theory and Applications

Lecture 12
Probabilistic Databases
Part-II

Instructor: Sudeepa Roy
Email: *sudeepa@cs.duke.edu*

Announcement

- Review#3
  - will be replaced by a homework

- Review#5
  - Hands-on experience on <u>one</u> data-analytics system of your choice (listed or not-listed on the course website)
    - Install, choose a dataset, run queries, write about your observation and attach the graphs/tables (< 1 page)
    - You can try to use it for your project
  - You may want to start early

Review: Lecture 11

- Part-I of probabilistic databases
  - Probabilistic DB overview
  - Possible world semantic
  - Compact representation for tuple independent databases
  - Extensional and intensional query evaluation in prob. Db.
  - Complexity class #P
  - #P-hardness proof for H_0() :- R(x) S(x, y) T(y)

- 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
  4. Papers listed on the website

Today: Lecture 12

Dichotomy for CQ- (no self join, no union)

- Any CQ- Q is
  - Either "safe", i.e. a "safe query plan" exists that can compute Pr[Q(D)] in poly-time for all D
  - Or "unsafe", #P-hard
    - Still Pr[Q(D)] can be computed in poly-time for some D
    - Recall "read-once formulas" as provenance
    - There are generalized "knowledge compilation forms"

      BDD, OBDD, FBDD, dec-DNNF, d-DNNF

      Not covered in this course

Safe/Unsafe Plans: Join ⋈

R			
A	B		
a1	b1	x1	0.3
a2	b1	x2	0.2
a2	b2	x3	0.9

S			
B	C		
b1	c1	y1	0.6
b2	c1	y2	0.5
b3	c2	y3	0.4

- q(x, y, z) :- R(x, y) S(y, z)
- Plan for q
  - Return R ⋈ S
  - Multiply probabilities
  - Annotation variables are only shown for convenience
- Plans should also generate probabilities of output tuples
- Is this plan
  - safe (correct)?
  - unsafe (wrong)?

R ⋈ S				
A	B	C		
a1	b1	c1	x1y1	0.3 * 0.6
a2	b1	c1	x2y1	0.2 * 0.6
a2	b2	c1	x3y2	0.9 * 0.5

Safe/Unsafe Plans: Join ⋈

R			
A	B		
a1	b1	x1	0.3
a2	b1	x2	0.2
a2	b2	x3	0.9

S			
B	C		
b1	c1	y1	0.6
b2	c1	y2	0.5
b3	c2	y3	0.4

- q(x, y, z) :- R(x, y) S(y, z)
- Safe Plan for q
  - Return R ⋈ S
  - "Independent Join"
  - Multiply probabilities
  - Pr[x1x2..] = Pr[x1]Pr[x2]…
- No projection: return any plan
  - E.g. 3 reln R, S, T
  - Return (R ⋈ S) ⋈ T
    or R ⋈(S ⋈ T)

R ⋈ S				
A	B	C		
a1	b1	c1	x1y1	0.3 * 0.6
a2	b1	c1	x2y1	0.2 * 0.6
a2	b2	c1	x3y2	0.9 * 0.5

Safe/Unsafe Plans: Project Π

R			
A	B		
a1	b1	x1	0.3
a2	b1	x2	0.2
a2	b2	x3	0.9

Π_A R		
A		
a1	x1	0.6
a2	X2 + x3	1 – (1 – 0.2)(1 -0.9)

- q(x) :- R(x, y)
- Safe Plan for q
  - Return Π_A R
  - “Independent project”
  - Apply Pr[x1 + x2 + ….] = 1 – (1 – Pr[x1])(1 –Pr[x2])…

Safe/Unsafe Plans

Join ⋈ + Project Π

R			
A	B		
a1	b1	x1	0.3
a2	b2	x2	0.2
a2	b3	x3	0.9

S			
B	C		
b1	c1	y1	0.6
b1	c2	y2	0.5
b2	c2	y3	0.4

$R \bowtie_B S$				
A	B	C		
a1	b1	c1	x1y1	0.3 * 0.6
a1	b1	c2	x1y2	0.3 * 0.5
a2	b2	c2	x2y3	0.2 * 0.4

$\Pi_{A,B} (R \bowtie S)$			
A	B		
a1	b1	x1y1+x1y2	1 – (1-0.18)(1-0.15)
a2	b2	x2y3	0.8

- **q(x, y) :- R(x, y) S(y, z)**
- Plan-1
  - $q = \Pi_{A,B} (R \bowtie_B S)$
- Step 1:
  - $q1 = R \bowtie_B S$
  - Independent join
- Step 2:
  - $q = \Pi_{A,B}\, q1$
  - Independent project?
  - Wrong!!
  - x1y1 and x1y2 are NOT independent events
- Plan-1 is NOT SAFE

Safe/Unsafe Plans

Join ⋈ + Project Π

R			
A	B		
a1	b1	x1	0.3
a2	b2	x2	0.2
a2	b3	x3	0.9

S			
B	C		
b1	c1	y1	0.6
b1	c2	y2	0.5
b2	c2	y3	0.4

Π_B S		
B		
b1	y1 + y2	1 - (1 – 0.6) * (1 – 0.5) = 0.8
b2	y3	0.4

R ⋈ $_B$ (Π_B S)			
A	B		
a1	b1	x1 (y1+y2)	0.3 * 0.8
a2	b2	x2y3	0.2 * 0.4

- **q(x, y) :- R(x, y) S(y, z)**
- Plan-2
  - q = R ⋈$_B$ (Π_B S)
- Step 1:
  - q1= Π_B S
  - Independent project
- Step 2:
  - q = R ⋈$_B$ q1
  - Independent join
  - Correct!!
  - x1 and (y1+y2) ARE INDEPENDENT EVENTS
- Plan-2 is SAFE!

The right (= safe) plan matters

- If the plan is right = SAFE, we compute the correct probabilities as we go along
  - Note: NO NEED TO COMPUTE THE PROVENANCE EXPRESSIONS
- The “Safe-Plan” algorithm by Dalvi-Suciu’04 makes sure that if a plan is returned, then it is SAFE
- What if the algorithm fails?
  - Then NO SAFE PLAN exists
  - Further, the query is then #P-hard!
- This gives a dichotomy on CQ-

Notations

- Attr(q) = Set of all attributes in all relations in q

- Head(q) = Set of attributes that are in output of the query q

- q(x, y) :- R(x, y) S(y, z)
- Attr(q) = {x, y, z}
- Head(q) = {x, y}

Extensional Operators

$$
\begin{aligned}
Pr_{\sigma_c^e(p)}(t) &= \begin{cases} Pr_p(t) & \text{if } c(t) \text{ is true} \\ 0 & \text{if } c(t) \text{ is false} \end{cases} \\
Pr_{\Pi_A^e(p)}(t) &= 1 - \prod_{t' : \Pi_A(t') = t} (1 - Pr_p(t')) \\
Pr_{p \times^e p'}(t, t') &= Pr_p(t) \times Pr_{p'}(t')
\end{aligned}
$$

- Select
- Independent Project
- Independent Join

Algorithm

Algorithm 1 SAFE-PLAN(q)

```
if Head(q) = Attr(q) then
  return any plan p for q
    (p is projection-free, hence safe)
end if
for A ∈ (Attr(q) − Head(q)) do
  let q_A be the query obtained from q
    by adding A to the head variables
  if Π_{Head(q)}(q_A) is a safe operator then
    return Π_{Head(q)}(SAFE-PLAN(q_A))
  end if
end for
Split q into q_1 ⋈_c q_2 (see text)
if no such split exists then
  return error("No safe plans exist")
end if
return SAFE-PLAN(q_1) ⋈_c SAFE-PLAN(q_2)
```

Algorithm

Algorithm 1 SAFE-PLAN(q)

- **if** $Head(q) = Attr(q)$ **then**
  - **return** any plan p for q
    - (p is projection-free, hence safe)
- **end if**
- **for** $A \in (Attr(q) - Head(q))$ **do**
  - let q_A be the query obtained from q
    - by adding A to the head variables
  - **if** $\Pi_{Head(q)}(q_A)$ is a safe operator **then**
    - **return** $\Pi_{Head(q)}(\text{SAFE-PLAN}(q_A))$
  - **end if**
- **end for**
- Split q into $q_1 \Join_c q_2$ (see text)
- **if** no such split exists **then**
  - **return** error("No safe plans exist")
- **end if**
- **return** $\text{SAFE-PLAN}(q_1) \Join_c \text{SAFE-PLAN}(q_2)$

- Example
  - q(x, y) :- R(x, y) S(x, y)
- Return any plan
  - E.g. $q = R \Join S$
- How to compute probability?
  - Just multiply
  - Why is this correct?
- In general, “Functional dependencies” matter

Functional dependencies (FD)

- X -> Y
  - X, Y subset of attributes
  - Any assignment of values to X uniquely determines the value of Y
- E.g.
  - A -> A
  - AB -> A
  - A -> AB in a relation R(A, B) if A is a “key”
  - X -> Y and Y -> Z imply X -> Z
  - etc
- What are some F.D. Γ in a prob db?
  - E.g. R.attr -> R.E (E = event expression), for any relation R
  - R.E -> R.attr (if R is a base relation)
  - For every join predicate Ri.A = Rj.B in q
    - Both Ri.A -> Rj.B and Rj.B -> Ri.A are in Γ(q)

Safe Operators

- Selections and joins (for CQ-) are always safe
  - Subsequent operators can be unsafe
  - Need to be careful for Joins

Projection

- For q, projecting to a subset of head variables A1,…,Ak is safe
  - if for every probabilistic relation R in the body,
  - there is an FD A1,...,Ak, R.E -> Head(q)
  - E = "event" (= provenance) attribute of all tables
- Why?
  - Projection to A1,…,Ak ⇔ disjunction of all tuples that have the same values of {A1,…,Ak}
  - To be independent (i.e. input contributes to unique output), each event from each table must be sufficient to distinguish tuples that contribute to the output

Safe Operators

Join

- Want to split q into q1 $\bowtie$ q2 “safely”
- Next: define separation among relations

Separation

- CQ- q
- Connected and Separate Relations
  - Two relations Ri, Rj $\in$ Rels(q) are called connected if
  - q has a join condition Ri.A = Rj.B
  - And either Ri.A or Rj.B is NOT in Head(q)
  - Ri, Rj are separate if they are not connected
- Separation
  - Two sets of relations **R1** and **R2** is a separation for q if
  - They partition the set Rels(q)
  - All pairs Ri $\in$ **R1** and Rj $\in$ **R2** are separate
- See the journal version in VLDB 2007 (click here)

Constraint graph for separation

- Graph G(q)
- Nodes are rels(q) = relations in q
- Edges are pairs (Ri, Rj) such that Ri, Rj are connected
- Find the connected components of G(q)
- If G(q) is a connected graph (= 1 component)
  - No separation/split is possible
- Otherwise
  - Split in any fashion
  - Can use cost-based optimization

Separation Examples

- q1() :- R(A), S(B, C), T(C)
  - Graph G(q1) : R –S – T
  - One connected component, no split possible
- q(B, C, D) :- S(A, B), T(C, D), B = C
  - Both join attributes B, C appear in head
  - NOTE the algo: for a join, either both attributes present or none are present
    - Otherwise a safe projection will be possible
  - S, T are separated, no edge
  - Split possible
  - $q = q1(B) \bowtie_{B=C} q2(C, D)$
  - q1(B) :- S(A, B)
  - q2(C, D) :- T(C, D)

Safe Plan Algorithm

- Top-Down
- Push all safe projections late in the plan
  - i.e. apply early
- When you can't, split the query q into two sub-queries q1 and q2 such that their join is q
  - if possible
- If stuck, the query is unsafe

Algorithm

Algorithm 1 SAFE-PLAN(q)

if $Head(q) = Attr(q)$ **then**
  return any plan p for q
    (p is projection-free, hence safe)
end if
for $A \in (Attr(q) - Head(q))$ **do**
  let q_A be the query obtained from q
    by adding A to the head variables
  if $\Pi_{Head(q)}(q_A)$ is a safe operator **then**
    return $\Pi_{Head(q)}(\text{SAFE-PLAN}(q_A))$
  end if
end for
Split q into $q_1 \Join_c q_2$ (see text)
if no such split exists **then**
  return error("No safe plans exist")
end if
return $\text{SAFE-PLAN}(q_1) \Join_c \text{SAFE-PLAN}(q_2)$

- Example on whiteboard

q(D) :- S(A, B), T(C, D), B = C

- Final Safe Plan:

$\Pi_D((\Pi_B S) \Join_{B=C} T)$

Dichotomy

All below are equivalent

1. q contains three subgoals of the form
   L(x, ….), J(x, y, …..), R(y, ….) where x, y not in Head(q)
2. q is #P-hard
3. The Safe-plan algo fails

- 2 => 3 is obvious (from the correctness of the algo)
- 3 => 1 needs a detailed analysis
  - Proof in the full journal version in VLDB 2007 (click here)
- 1 => 2 next

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) = $\{\alpha \in$ Atoms(Q) $\mid x \in$ 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
- For Boolean CQ-, Hierarchical queries $\Leftrightarrow$ Safe queries

Not hierarchical: #P-hard

- Step1: $H_0() :- R(x)\ S(x, y)\ T(y)$ is hard
  - Proved in Lecture 11
- Step2: H_0 reduces to any non-hierarchical query

Non-hierarchical CQ-: Step 2

- Reduction from $H_0 = R(x), T(x, y), S(y)$ to Q
- We can choose variables x and y and atoms αx, αy and αxy such that:
  - $x \in vars(r_x)$, $y \notin vars(r_y)$ ($\notin$ = not in)
  - $y \in vars(r_y)$ and $x \notin vars(r_x)$
  - $x,y \in vars(r_{xy})$
- Q = U(x,z), V(x,u), W(x,y,z), Y(y,a)
  - $r_x = U$, $r_y = Y$, $r_{xy} = W$
- Reduction idea: On whiteboard
  - U, Y, W gets the same+extended tuples as in R, S, T
  - Other relations (e.g. V) are deterministic
  - Map all variables/attributes other than x, y to a new constant c
  - Note: "a" in Y(y, a) has to be unchanged.
  - Identical "provenance"

Approximations

- “Exact” evaluation is hard
- Approximation is always possible for UCQ
  - But even approximation may be impossible if the query has negation
- Extensions to DNF counting approx algo by Karp and Luby’1983

Later in "TBD" lectures

- Probabilistic Relational Model
  - Probabilistic Soft Logic
  - Markov Logic Network