

Probabilistic Databases

Amol Deshpande

University of Maryland

Goal

- Introduction to probabilistic databases
- Focus on an overview of:
 - Different possible representations
 - Challenges in using them

Probabilistic Data

- Traditional databases store solid “facts” that can be considered certain
- In many cases, we don’t know things precisely
 - When storing beliefs etc
 - Answering queries with vague predicates
 - institute_name like “IITB”
 - If you use probabilistic models for predicting some variables
 - Or classification models
 - Similarly, use statistical models to predict missing data
 - I saw a bird, but not sure if it was a dove or a sparrow
- Uncertain, incomplete data is becoming more and more common

Representing uncertainty

- A high-level classification can be made:
- Tuple-level uncertainty
 - All attributes in a tuple are known precisely; existence of the tuple is uncertain
 - Vague predicates: *name* approx like '*iit*'
 - Tuple ("IIT Bombay", ...) will be present in the answer with some uncertainty
- Attribute-level uncertainty
 - Tuples (identified by *keys*) exist for certain; an attribute value is however uncertain
 - Tomorrow temperature will be somewhere between 20C and 30C

Classification

- Is this classification fundamental ?
 - Can proper normalization solve this problem ?
- Probably yes.
- Tuple-level uncertainty can be converted into attribute-level by adding a boolean attribute
 - Things will probably get messy
- Other way round is harder
 - Continuous distributions on attributes are common
 - Gaussian on temperature for tomorrow.

Classification

- Trying to create a general model that can represent everything is probably doomed to fail
- We will discuss two papers next:
 - Simple tuple-level uncertainty model by Norbert Fuhr et al
 - A not-too-complex attribute-level uncertainty model that we use in a sensor network application
- Intractability issues are encountered very soon

Roadmap

- Tuple-level uncertainty model originally proposed by Norbert Fuhr, and later work by Dalvi, Suciu
 - Possible Worlds Semantics
 - Intensional vs Extensional Semantics
 - Query execution
- Attribute-level uncertainty model we used in a sensor network application
 - Query execution
- An attempt to put other related work in this framework

Tuple-level Uncertainty

- Proposed by Fuhr et al, more work recently by Dalvi and Suciu
 - The examples are from Dalvi, Suciu [VLDB04]
- With each tuple, a probability of existence is associated

$$S^p =$$

	A	B	
s_1	'm'	1	0.8
s_2	'n'	1	0.5

$$T^p =$$

	C	D	
t_1	1	'p'	0.6

Tuple-level Uncertainty

- Lets assume that the tuple existence events are independent of each other
 - So, prob that $S^P = \{s_1, s_2\}$ is $0.5 * 0.8 = 0.4$
 - Similarly prob that $T^P = \{\}$ is empty is 0.4
 - And prob that $S^P = \{s_1, s_2\}$ and $T^P = \{\}$ is $0.4 * 0.4 = 0.16$
- In fact, we can assign a probability to each such possibility

Possible Worlds

database instance	probability
$D_1 = \{s_1, s_2, t_1\}$	0.24
$D_2 = \{s_1, t_1\}$	0.24
$D_3 = \{s_2, t_1\}$	0.06
$D_4 = \{t_1\}$	0.06
$D_5 = \{s_1, s_2\}$	0.16
$D_6 = \{s_1\}$	0.16
$D_7 = \{s_2\}$	0.04
$D_8 = \phi$	0.04

$pwd(D^P) =$

Possible Worlds Semantics

- A probabilistic relation is simply a collection of different possible deterministic relations (worlds) with associated probabilities
- Probabilities add up to 1

Query Evaluation

- Say you want to execute a query:
 - $S \text{ Join } T \text{ on } B = C, \text{ project on } D$
- Execute the query on each possible world separately
- The final result is a probabilistic relation that represents the end result

Aside

- Selections:
 - The result contains all tuples that match a specified predicate
- Joins:
 - Given two relations, find pairs of matching tuples and concatenate
- Projection:
 - Throw away all attributes except the ones specified

Query execution

- S Join T on B = C, project on D

$S^p =$

	A	B	
s_1	'm'	1	0.8
s_2	'n'	1	0.5

$T^p =$

	C	D	
t_1	1	'p'	0.6

	database instance	probability	Result
$pwd(D^p) =$	$D_1 = \{s_1, s_2, t_1\}$	0.24	{'p'}
	$D_2 = \{s_1, t_1\}$	0.24	{'p'}
	$D_3 = \{s_2, t_1\}$	0.06	{'p'}
	$D_4 = \{t_1\}$	0.06	{}
	$D_5 = \{s_1, s_2\}$	0.16	{}
	$D_6 = \{s_1\}$	0.16	{}
	$D_7 = \{s_2\}$	0.04	{}
	$D_8 = \phi$	0.04	{}

answer	probability
{'p'}	0.54
\emptyset	0.46

Query evaluation

- This evaluation is semantically correct
 - But returning this answer is not practical
- Instead try to convert it to a probabilistic relation
 - For each tuple, compute the probability it is in the answer
 - By summing over all worlds which contain that tuple
 - Called 'rank' (given the focus of the work)

D	Rank
'p'	0.54

Another example

- S Join T on B = C, project on A

$S^p =$

	A	B	
s_1	'm'	1	0.8
s_2	'n'	1	0.5

$T^p =$

	C	D	
t_1	1	'p'	0.6

database instance	probability	Result
$D_1 = \{s_1, s_2, t_1\}$	0.24	{'m','n'}
$D_2 = \{s_1, t_1\}$	0.24	{'m'}
$D_3 = \{s_2, t_1\}$	0.06	{'n'}
$D_4 = \{t_1\}$	0.06	{}
$D_5 = \{s_1, s_2\}$	0.16	{}
$D_6 = \{s_1\}$	0.16	{}
$D_7 = \{s_2\}$	0.04	{}
$D_8 = \phi$	0.04	{}

$pwd(D^p) =$

answer	probability
{'m','n'}	0.24
{'m'}	0.24
{'n'}	0.06
\emptyset	0.46

Returned Answer

- S Join T on B = C, project on A

$S^p =$

	A	B	
s_1	'm'	1	0.8
s_2	'n'	1	0.5

$T^p =$

	C	D	
t_1	1	'p'	0.6

answer	probability
{'m', 'n'}	0.24
{'m'}	0.24
{'n'}	0.06
\emptyset	0.46

D	Rank
'm'	0.48
'n'	0.30

Note that this final relation does not satisfy independence;
The tuples after a query evaluation may become dependent

Query Evaluation

- Evaluating a general query:
 - Converting to possible worlds, executing the query separately for each one, and then combining them is not feasible
 - Two alternative solutions that work directly on the associated tuple-level probabilities
 - Extensional and Intensional Semantics

Extensional Semantics

- We will operate directly on the probabilities
- Take a normal query plan for the query, and execute the query normally
- When a new tuple is created, compute a probability for it
 - Assuming independence (for now)
- In the end, the result tuples will have probabilities associated

Extensional: Example

$$S^p =$$

	A	B	
s_1	'm'	1	0.8
s_2	'n'	1	0.5

$$T^p =$$

	C	D	
t_1	1	'p'	0.6

Joins: assume independence

A	B	C	D	<i>prob</i>
'm'	1	1	'p'	$0.8 * 0.6 = 0.48$
'n'	1	1	'p'	$0.5 * 0.6 = 0.30$

Projection: union probability; assume independence

D	<i>prob</i>
'p'	$(1 - (1 - 0.48)(1 - 0.3)) = 0.636$

Umm.. This is wrong !!

Why ? The two tuples above are not independent.

Alternate Query Plan

$$S^p = \begin{array}{l} s_1 \\ s_2 \end{array} \begin{array}{|c|c|} \hline \mathbf{A} & \mathbf{B} \\ \hline \text{'m'} & 1 \\ \hline \text{'n'} & 1 \\ \hline \end{array} \begin{array}{l} 0.8 \\ 0.5 \end{array} \quad T^p = \begin{array}{l} t_1 \end{array} \begin{array}{|c|c|} \hline \mathbf{C} & \mathbf{D} \\ \hline 1 & \text{'p'} \\ \hline \end{array} 0.6$$

Projection: union probability; assume independence

$$\begin{array}{|c|} \hline \mathbf{B} \\ \hline 1 \\ \hline \end{array} \begin{array}{l} \text{prob} \\ (1 - (1 - 0.8)(1 - 0.5)) = 0.9 \end{array}$$

Join: assume independence

$$\begin{array}{|c|c|c|} \hline \mathbf{B} & \mathbf{C} & \mathbf{D} \\ \hline 1 & 1 & \text{'p'} \\ \hline \end{array} \begin{array}{l} \text{prob} \\ 0.9 * 0.6 = 0.54 \end{array}$$

This is correct.

The correctness unfortunately depends on the plan used.
Called “safe plans” [Dalvi, Suciu 2004]

Safe plans

- [Dalvi, Suciu 2004] give an algorithm to find a safe plan if one exists for a given query
- If no safe plan, the complexity of query evaluation is in #P-Complete
- Use intensional semantics for them (next)

Intensional Semantics

- Associate a separate event with each base tuple
- For each intermediate tuple, associated an explicit event expression
- Compute the actual probabilities only when required at the end

Intensional Semantics

$$S^p = \begin{array}{l} s_1 \\ s_2 \end{array} \begin{array}{|c|c|} \hline \mathbf{A} & \mathbf{B} \\ \hline \text{'m'} & 1 \\ \hline \text{'n'} & 1 \\ \hline \end{array} \begin{array}{l} 0.8 \\ 0.5 \end{array} \quad T^p = \begin{array}{l} t_1 \end{array} \begin{array}{|c|c|} \hline \mathbf{C} & \mathbf{D} \\ \hline 1 & \text{'p'} \\ \hline \end{array} 0.6$$

Join (intersection):

A	B	C	D	E
'm'	1	1	'p'	$s_1 \wedge t_1$
'n'	1	1	'p'	$s_2 \wedge t_1$

Projection (union):

D	E
'p'	$(s_1 \wedge t_1) \vee (s_2 \wedge t_1)$

This would result in the correct probability.
Can also see correlations.

Intensional Semantics

- Does not depend on the query plan used
- Unfortunately...
 - This is computationally too expensive
 - Creating and carrying around the event expressions is itself quite painful
 - Evaluating a complex event expression is #P-complete

Recap

- Tuple-level probabilities
 - Extensional semantics: Limited use
 - Safe plans [Dalvi, Suciu 04]
 - Intensional semantics: Intractable
- This model has extended to include some kinds of tuple correlations [Fuhr, Roelleke]
 - e.g. tuple disjointness

Roadmap

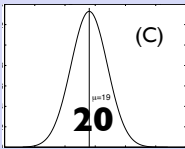

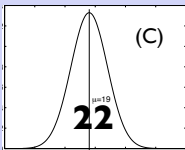
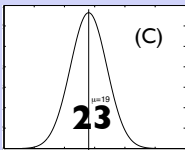
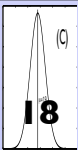
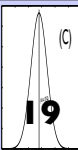
- Tuple-level uncertainty model originally proposed by Norbert Fuhr, and later work by Dalvi, Suciu
 - Possible Worlds Semantics
 - Intensional vs Extensional Semantics
 - Query execution
- Attribute-level uncertainty model we used in a sensor network application
 - Query execution
- An attempt to put other related work in this framework

Attribute-level Uncertainties

- Sensor network application
- Consider two temperature sensors monitoring temperatures at two locations
 - Location 1 : temp1
 - Location 2 : temp2
- We propose using a probabilistic model of the evolution of these two variables over time [DGMHH' VLDB 04]
 - Goal was to use the attribute correlations to avoid sensing temperature as much as possible
 - The correlations tend to be very strong

Attribute-level

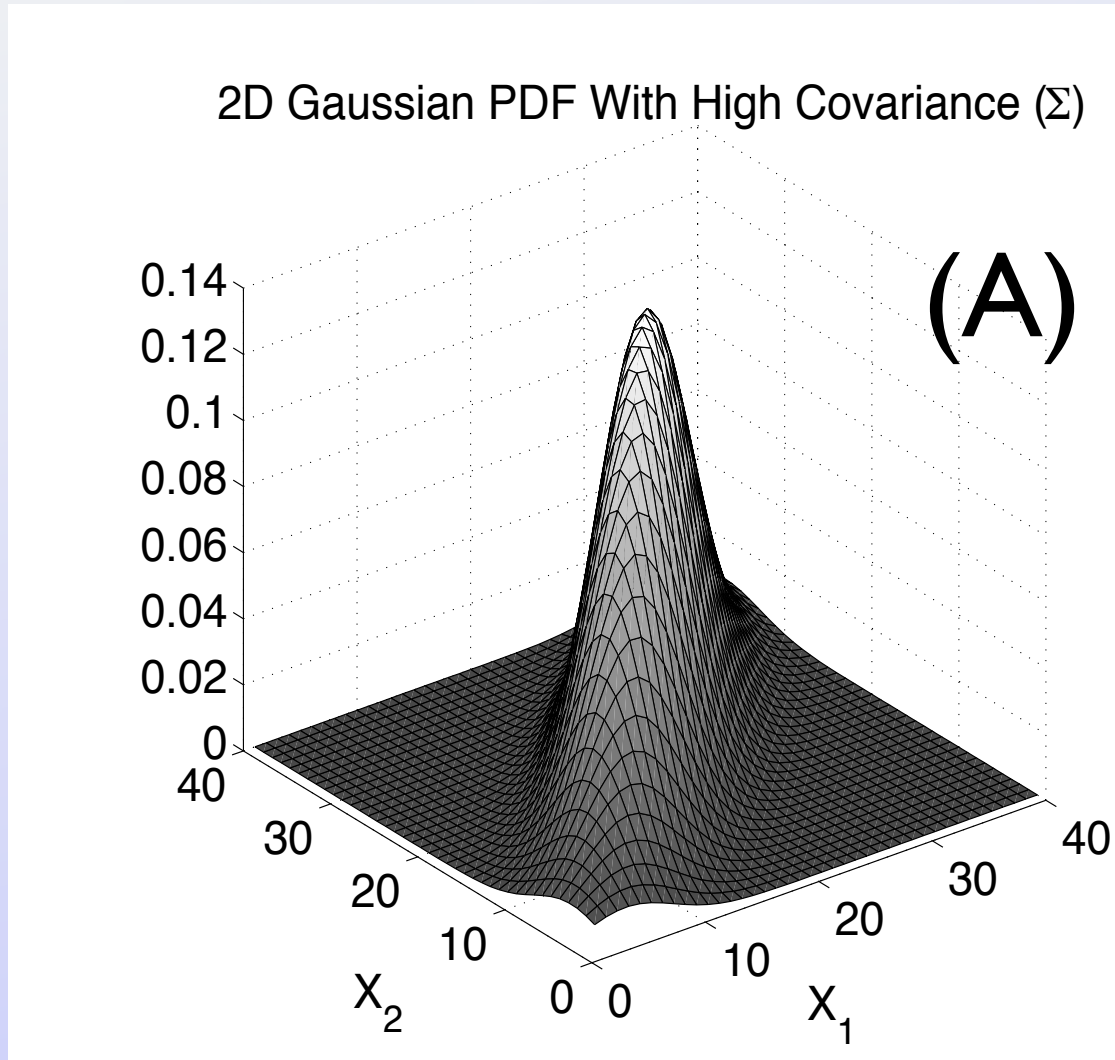
- Tuples exist with certainty
 - temperature at time t_1 at location l etc.
- But the attribute values (temperatures) are uncertain
 - In particular, each temperature value is a *Gaussian*
 - $p(\text{temp}_l \text{ at time } l)$ is a Gaussian distribution

<u>time</u>	<u>temp1</u>	<u>temp2</u>
1		
2		
3		

Attribute-level

- Moreover
 - Temperatures at different locations are *spatially correlated*
 - $temp_1^{t=1}$ and $temp_2^{t=1}$ are correlated
 - Temperatures across time are *temporally correlated*
 - $temp_1^{t=1}$ and $temp_1^{t=2}$ are correlated
- All these correlations are represented by using multi-dimensional Gaussian distributions over the uncertain attributes

Example 2-dim Gaussian



Query execution

- Range Query: Is X_i within $[a_i, b_i]$?

- Compute the probability:

$$P(X_i \in [a_i, b_i]) = \int_{a_i}^{b_i} p(x_i) dx_i.$$

- Answer is YES with that probability
- This simplest integral is unfortunately non-computable
 - Need to use numerical integration
 - In this particular case, tables can be used

Query Execution

- For Gaussian distributions, some queries can be answered using closed form expressions
 - But range queries can't be
- Also using a range query can result in a partial gaussian, which would be hard to deal with further
- How do you do joins ?
 - Cheng, Prabhakar address some of these issues in their work

Query Execution

- Things get more complicated if you are using more complex probability distributions
- In general, exact answers are probably not achievable
- But approximations should suffice in many cases

Recap

- Attribute-level uncertainty means the attribute values are uncertain
- In general, say the uncertain attributes are $U_i, i = 1, \dots, N$
- In the simplest case, the attributes are independent of each other
 - Even then things can get hairy
 - For example, if the attributes are continuous and the probability distributions used are *Gaussians*
 - For continuous attributes, anything but “*uniform within range $[a, b]$* ” would probably be non-trivial to handle
 - Even for discrete distributions (see later)

Recap

- In the case we considered, the correlations were very important, and were explicitly modeled
- A single multi-dimensional probability distribution on all U_i simultaneously
- Things get messy very soon even for the simplest continuous distribution considered
- Query execution times can be very high

Roadmap

- Tuple-level uncertainty model originally proposed by Norbert Fuhr, and later work by Dalvi, Suciu
 - Possible Worlds Semantics
 - Intensional vs Extensional Semantics
 - Query execution
- Attribute-level uncertainty model we used in a sensor network application
 - Query execution
- An attempt to put other related work in this framework

Barbara et al [1992]

- “The management of probabilistic data”, IEEE TKDE 1992
- Attribute-level Uncertainty
- Discrete variables
- Had a notion of “missing probability”
 - Assumed to be distributed over the entire domain, but no assumptions on exactly how
 - Semantics of relational operators ended up being a bit messy

TABLE II
EXAMPLE OF MISSING PROBABILITIES

EMPLOYEE	DEPARTMENT	QUALITY BONUS	SALES
Jon Smith	Toy	0.3 [Great yes]	0.3
		0.4 [Good yes]	[\$30–34K]
		0.2 [Fair *]	0.5
		0.1 [* *]	[\$35–39K]
			0.2 [*]

Barbara et al [1992]

- Can model correlations between attributes
- This model has similarities to graphical models, conditional independence etc...
- Especially when multiple relations and joins are considered

TABLE II
EXAMPLE OF MISSING PROBABILITIES

EMPLOYEE	DEPARTMENT	QUALITY BONUS	SALES
Jon Smith	Toy	0.3 [Great yes]	0.3
		0.4 [Good yes]	[\$30–34K]
		0.2 [Fair *]	0.5
		0.1 [* *]	[\$35–39K]
			0.2 [*]

Cheng, Prabhakar et al

- e.g. “Evaluating probabilistic queries over imprecise data”; SIGMOD 2003
- Attribute-level uncertainty
 - The range of an uncertain attribute is assumed to be known and hopefully not too large
 - No assumptions on how the value is distributed in this range
 - So temp_l is in [18, 23] definitely; the actual distribution in this range could be anything (e.g. a cut-off gaussian)
- Assumed independence
- Focus on answering queries such as nearest-neighbor queries
 - The *ranges* on the attribute values are used heavily in these algorithms

Probview

- Lakshaman, Leone, Ross, Subrahmanian [TODS 97]
- An attempt to generalize many of the previous models
- Tuple-level uncertainties
 - But for each tuple, we have a range associated
 - An upper bound and a lower bound
- To a large extent, succeeded in combining the different types of probabilistic models proposed before
- But the resulting model is quite complex

TRIO

- Project recently started at Stanford
- “Working models for uncertain data”; ICDE 06
- In the models presented in this paper, they don’t really have *probabilities*
- Semantics similar to possible worlds
- An “uncertain” relation is defined to be a set of possible relation instances
- Example: Bird spotting relation (spotter, date, location, bird)

I1: empty

I2: [Carol, 12/25/04, Los Altos, bluebird]

I3: [Carol, 12/25/04, Los Altos, bluebird],
[Carol, 12/26/04, Los Altos, bluebird]

TRIO

- Present one “complete” model that is closed under all operations:
 - An uncertain relation is:
 - A deterministic relation with a “variable” associated with each tuple
 - A boolean formula $f(T)$ over these variables
 - If the formula is true, that particular instance exists
 - This model is probably intractable computationally
- A series of less complex “incomplete” models which are probably better suited for implementation

```
t1 = [Carol, 12/25/04, Los Altos, bluebird]
t2 = [Carol, 12/26/04, Los Altos, bluebird]
constraint: t2 => t1
```

Questions ?

