Lifted inference in relational graphical models and (potentially) probabilistic programs

David Poole

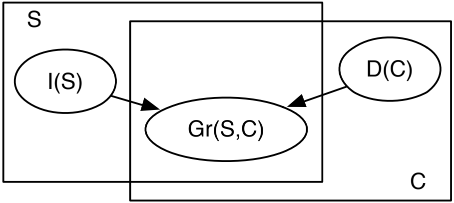
Department of Computer Science,
University of British Columbia
Leverhulme Trust visting professor at the University of Oxford

March 2015

Outline

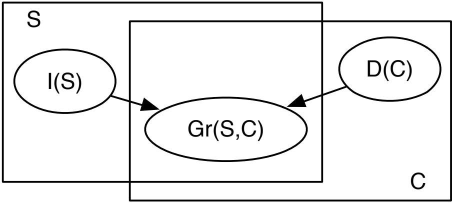
1. Relational Graphical Models
2. Exact Inference
   - Recursive Conditioning
   - Lifted Inference
   - Lifted Recursive Conditioning
3. Lifting Probabilistic Programs (?)

Plate Notation



- S, C logical variables representing students, courses
- the set of individuals of a type is called a population
- $I(S)$, $Gr(S,C)$, $D(C)$ are parametrized random variables
- Specify $P(I(S))$, $P(D(C))$, $P(Gr(S,C) \mid I(S), D(C))$

Plate Notation

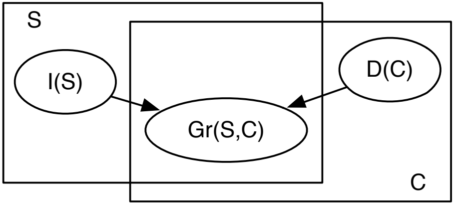


- S, C logical variables representing students, courses
- the set of individuals of a type is called a population
- $I(S)$, $Gr(S,C)$, $D(C)$ are parametrized random variables
- Specify $P(I(S))$, $P(D(C))$, $P(Gr(S,C) \mid I(S), D(C))$

Grounding:

- for every student s, there is a random variable $I(s)$
- for every course c, there is a random variable $D(c)$
- for every s, c pair there is a random variable $Gr(s,c)$

Plate Notation



- With 1000 students and 100 courses, grounding contains
  - 1000 $I(s)$ variables
  - 100 $D(C)$ variables
  - 100000 $Gr(s, c)$ variables

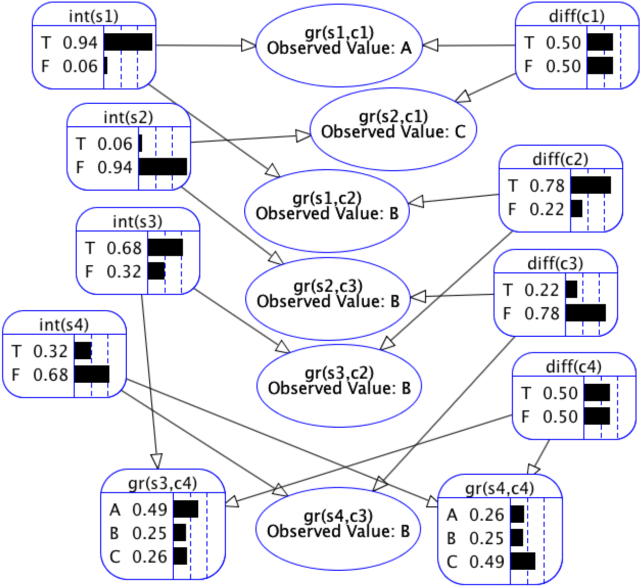
  total: 101100 variables
- Suppose Gr has 3 possible values. Numbers to be specified to define the probabilities:
  1 for $I(s)$, 1 for $D(C)$, 8 for $Gr(S, C) = 10$ parameters.

Example: Predicting Relations

Student	*Course*	*Grade*
s_1	c_1	A
s_2	c_1	C
s_1	c_2	B
s_2	c_3	B
s_3	c_2	B
s_4	c_3	B
s_3	c_4	?
s_4	c_4	?

- Students s_3 and s_4 have the same averages, on courses with the same averages.
- Which student would you expect to better?

Example: Predicting Relations



Outline

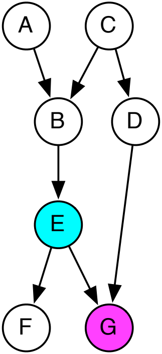
1. Relational Graphical Models
2. Exact Inference
   - Recursive Conditioning
   - Lifted Inference
   - Lifted Recursive Conditioning
3. Lifting Probabilistic Programs (?)

Why Exact Inference?

Why do we care about exact inference?

- Gold standard
- Size of problems amenable to exact inference is growing
- Learning for inference
- Basis for efficient approximate inference:
  - Rao-Blackwellization
  - Variational Methods

Inference via factorization in graphical models



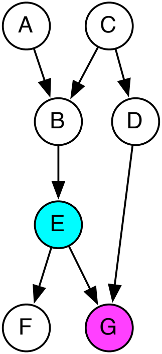
$$P(E \mid g) = \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$

$$P(E \wedge g)$$

$$= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC)$$

$$P(C)P(D \mid C)P(E \mid B)P(F \mid E)P(g \mid ED)$$

Inference via factorization in graphical models



$$P(E \mid g) = \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$

$$\begin{aligned}
&P(E \wedge g) \\
&= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC) \\
&\quad P(C)P(D \mid C)P(E \mid B)P(F \mid E)P(g \mid ED) \\
&= \left(\sum_F P(F \mid E)\right) \\
&\quad \sum_B P(E \mid B) \sum_C \left(P(C)\left(\sum_A P(A)P(B \mid AC)\right)\right. \\
&\quad \left.\left(\sum_D P(D \mid C)P(g \mid ED)\right)\right)
\end{aligned}$$

Recursive Conditioning

- Variable elimination is the dynamic programming variant of recursive conditioning.
- Recursive Conditioning is the search variant of variable elimination
- They do the same additions and multiplications.
- Complexity $O(nr^t)$, for n variables, range size r, and treewidth t.

Recursive Conditioning

procedure $rc(Con:$ context$, Fs:$ set of factors$)$:
  if $\exists v$ such that $\langle\langle Con, Fs\rangle, v\rangle \in cache$
    return v
  else if $vars(Con) \not\subseteq vars(Fs)$
    return $rc(\{X = v \in Con : X \in vars(Fs)\}, Fs)$
  else if $\exists F \in Fs$ such that $vars(F) \subseteq vars(Con)$
    return $eval(F, Con) \times rc(Con, Fs \setminus \{F\})$
  else if $Fs = Fs_1 \uplus Fs_2$ where $vars(Fs_1) \cap vars(Fs_2) \subseteq vars(Con)$
    return $rc(Con, Fs_1) \times rc(Con, Fs_2)$
  else select variable $X \in vars(Fs)$
    $sum \leftarrow 0$
    for each $v \in domain(X)$
      $sum \leftarrow sum + rc(Con \cup \{X = v\}, Fs)$
    $cache \leftarrow cache \cup \{\langle\langle Con, Fs\rangle, sum\rangle\}$
    return sum

Recursive Conditioning

procedure $rc(Con:$ context, $Fs:$ set of factors):
    if $\exists v$ such that $\langle\langle Con, Fs\rangle, v\rangle \in cache$
        return v
    else if $vars(Con) \not\subseteq vars(Fs)$
        return $rc(\{X = v \in Con : X \in vars(Fs)\}, Fs)$
    else if $\exists F \in Fs$ such that $vars(F) \subseteq vars(Con)$
        return $eval(F, Con) \times rc(Con, Fs \setminus \{F\})$
    else if $Fs = Fs_1 \uplus Fs_2$ where $vars(Fs_1) \cap vars(Fs_2) \subseteq vars(Con)$
        return $rc(Con, Fs_1) \times rc(Con, Fs_2)$
    else select variable $X \in vars(Fs)$
        $sum \leftarrow 0$
        for each $v \in domain(X)$
            $sum \leftarrow sum + rc(Con \cup \{X = v\}, Fs)$
        $cache \leftarrow cache \cup \{\langle\langle Con, Fs\rangle, sum\rangle\}$
        return sum

Recursive Conditioning

procedure $rc(Con:$ context, $Fs:$ set of factors):
    if $\exists v$ such that $\langle\langle Con, Fs\rangle, v\rangle \in cache$
        return v
    else if $vars(Con) \not\subseteq vars(Fs)$
        return $rc(\{X = v \in Con : X \in vars(Fs)\}, Fs)$
    else if $\exists F \in Fs$ such that $vars(F) \subseteq vars(Con)$
        return $eval(F, Con) \times rc(Con, Fs \setminus \{F\})$
    else if $Fs = Fs_1 \uplus Fs_2$ where $vars(Fs_1) \cap vars(Fs_2) \subseteq vars(Con)$
        return $rc(Con, Fs_1) \times rc(Con, Fs_2)$
    else select variable $X \in vars(Fs)$
        $sum \leftarrow 0$
        for each $v \in domain(X)$
            $sum \leftarrow sum + rc(Con \cup \{X = v\}, Fs)$
        $cache \leftarrow cache \cup \{\langle\langle Con, Fs\rangle, sum\rangle\}$
        return sum

Recursive Conditioning

procedure $rc(Con : \text{ context}, Fs : \text{ set of factors})$:
  if $\exists v$ such that $\langle\langle Con, Fs\rangle, v\rangle \in cache$
    return v
  else if $vars(Con) \not\subseteq vars(Fs)$
    return $rc(\{X = v \in Con : X \in vars(Fs)\}, Fs)$
  else if $\exists F \in Fs$ such that $vars(F) \subseteq vars(Con)$
    return $eval(F, Con) \times rc(Con, Fs \setminus \{F\})$
  else if $Fs = Fs_1 \uplus Fs_2$ where $vars(Fs_1) \cap vars(Fs_2) \subseteq vars(Con)$
    return $rc(Con, Fs_1) \times rc(Con, Fs_2)$
  else select variable $X \in vars(Fs)$
    $sum \leftarrow 0$
    for each $v \in domain(X)$
      $sum \leftarrow sum + rc(Con \cup \{X = v\}, Fs)$
    $cache \leftarrow cache \cup \{\langle\langle Con, Fs\rangle, sum\rangle\}$
    return sum

Outline

1. Relational Graphical Models
2. Exact Inference
   - Recursive Conditioning
   - Lifted Inference
   - Lifted Recursive Conditioning
3. Lifting Probabilistic Programs (?)

Lifted Inference

- Idea: treat those individuals about which you have the same information as a block; just count them.
- Use the ideas from lifted theorem proving - no need to ground.
- Potential to be exponentially faster in the number of non-differentialed individuals.
- Relies on knowing the number of individuals (the population size).

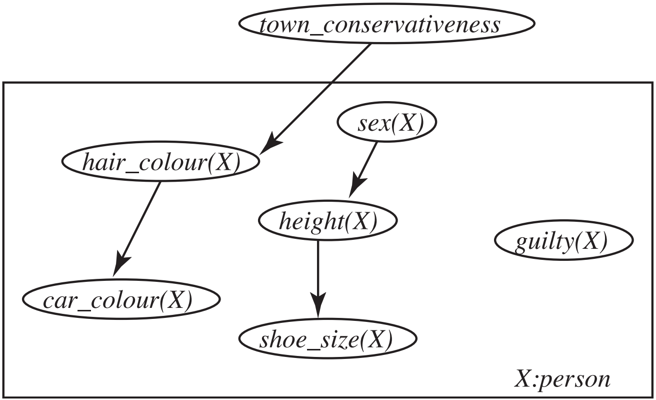
Queries depend on population size

Suppose we observe:

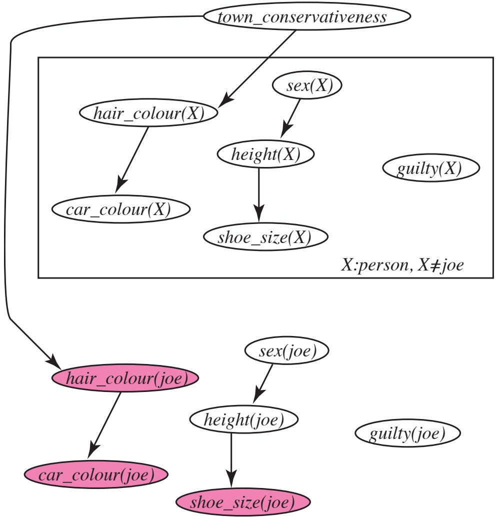
- Joe has purple hair, a purple car, and has big feet.
- A person with purple hair, a purple car, and who is very tall was seen committing a crime.

What is the probability that Joe is guilty?

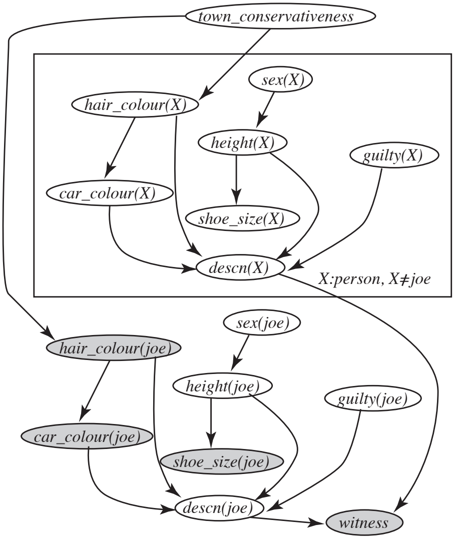
Background parametrized belief network



Observing information about Joe



Observing Joe and the crime



Parametric Factors

A parametric factor (parfactor) is a triple $\langle C, V, t\rangle$ where

- C is a set of inequality constraints on parameters,
- V is a set of parametrized random variables
- t is a table representing a factor from the random variables to the non-negative reals.

$$\left\langle \{X \neq sue\}, \{interested(X), boring\}, \begin{array}{|ll|l|} \hline interested & boring & Val \\ \hline yes & yes & 0.001 \\ yes & no & 0.01 \\ & \ldots & \\ \hline \end{array} \right\rangle$$

Factored Parametric Factors

A factored parametric factor is a triple $\langle C, V, t\rangle$ where

- C is a set of inequality constraints on parameters,
- V an assignment to parametrized random variables
- t number

Parfactor:

$$\left\langle \{X \neq sue\}, \{interested(X), boring\}, \begin{array}{|ll|l|} \hline interested & boring & Val \\ \hline yes & yes & 0.001 \\ yes & no & 0.01 \\ & \ldots & \\ \hline \end{array} \right\rangle$$

becomes

$\langle\{X \neq sue\}, interested(X) \wedge boring, 0.001\rangle$

$\langle\{X \neq sue\}, interested(X) \wedge \neg boring, 0.01\rangle$

...

Outline

1. Relational Graphical Models
2. Exact Inference
   - Recursive Conditioning
   - Lifted Inference
   - Lifted Recursive Conditioning
3. Lifting Probabilistic Programs (?)

Lifted Recursive Conditioning

$lrc(Con, Fs)$

- Con is a set of assignments to random variables and counts to assignments of instances of relations. e.g.:

$$\begin{aligned}\{\neg A,\ &\#_x F(x) \wedge G(x) = 7,\\ &\#_x F(x) \wedge \neg G(x) = 5,\\ &\#_x \neg F(x) \wedge G(x) = 18,\\ &\#_x \neg F(x) \wedge \neg G(x) = 0\}\end{aligned}$$

- Fs is a set of factored parametrized factors, e.g.,

$$\begin{aligned}\{\ &\langle\{\}, \neg A \wedge \neg F(x) \wedge G(x), 0.1\rangle\ ,\\ &\langle\{\}, A \wedge \neg F(x) \wedge G(x), 0.2\rangle\ ,\\ &\langle\{\}, F(x) \wedge G(y), 0.3\rangle\ ,\\ &\langle\{\}, F(x) \wedge H(x), 0.4\rangle\}\end{aligned}$$

Evaluating ParFactors

Con:

$$\{\neg A,\ \#_x F(x) \wedge G(x) = 7,$$
$$\#_x F(x) \wedge \neg G(x) = 5,$$
$$\#_x \neg F(x) \wedge G(x) = 18,$$
$$\#_x \neg F(x) \wedge \neg G(x) = 0\}$$

Fs:

$$\{\ \langle \{\}, \neg A \wedge \neg F(x) \wedge G(x), 0.1\rangle\ ,$$
$$\langle \{\}, A \wedge \neg F(x) \wedge G(x), 0.2\rangle\ ,$$
$$\langle \{\}, F(x) \wedge G(y), 0.3\rangle\ ,$$
$$\langle \{\}, F(x) \wedge H(x), 0.4\rangle\}$$

$lrc(Con, Fs)$ returns:

Evaluating ParFactors

Con:

$$\{\neg A,\ \#_x F(x) \wedge G(x) = 7, \\ \#_x F(x) \wedge \neg G(x) = 5, \\ \#_x \neg F(x) \wedge G(x) = 18, \\ \#_x \neg F(x) \wedge \neg G(x) = 0\}$$

Fs:

$$\{\ \langle \{\}, \neg A \wedge \neg F(x) \wedge G(x), 0.1 \rangle\ , \\ \langle \{\}, A \wedge \neg F(x) \wedge G(x), 0.2 \rangle\ , \\ \langle \{\}, F(x) \wedge G(y), 0.3 \rangle\ , \\ \langle \{\}, F(x) \wedge H(x), 0.4 \rangle \}$$

$lrc(Con, Fs)$ returns:

$$0.1^{18} * 0.3^{12*25} * lrc(Con, \{\langle \{\}, F(x) \wedge H(x), 0.4 \rangle\})$$

Branching

Con:

$$
\begin{aligned}
\{\neg A,\ & \#_x F(x) \land G(x) = 7, \\
& \#_x F(x) \land \neg G(x) = 5, \\
& \#_x \neg F(x) \land G(x) = 18, \\
& \#_x \neg F(x) \land \neg G(x) = 0\}
\end{aligned}
$$

Fs:

$$\{\, \langle \{\}, F(x) \land H(x), 0.4 \rangle \,, \ldots \,\}$$

Branching on H for the 7 "x" individuals s.th. $F(x) \land G(x)$:
$lrc(Con, Fs) =$

Branching

Con:

$$\begin{aligned}\{\neg A, \ &\#_x F(x) \land G(x) = 7,\\ &\#_x F(x) \land \neg G(x) = 5,\\ &\#_x \neg F(x) \land G(x) = 18,\\ &\#_x \neg F(x) \land \neg G(x) = 0\}\end{aligned}$$

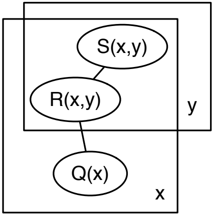
Fs:

$$\{ \langle \{\}, F(x) \land H(x), 0.4 \rangle, \ldots \}$$

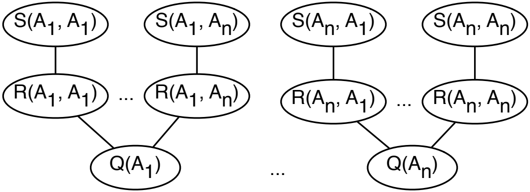
Branching on H for the 7 "x" individuals s.th. $F(x) \land G(x)$:
$lrc(Con, Fs) =$

$$\sum_{i=0}^{7} \binom{7}{i} lrc(\{\neg A, \quad \begin{aligned}&\#_x F(x) \land G(x) \land H(x) = i,\\ &\#_x F(x) \land G(x) \land \neg H(x) = 7 - i,\\ &\#_x F(x) \land \neg G(x) = 5, \ldots\}, Fs)\end{aligned}$$

Recognizing Disconnectedness



Relational Model



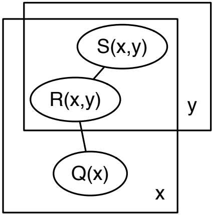
Grounding

Parfactors *Fs*:

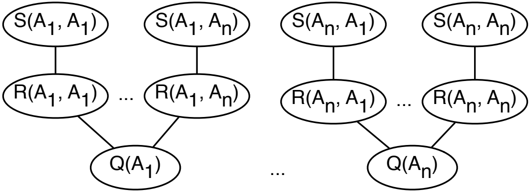
$$\{\langle\{\},\{S(x,y),R(x,y)\},t_1\rangle$$
$$\langle\{\},\{Q(x),R(x,y)\},t_2\rangle\}$$

$$lrc(Con, Fs) =$$

Recognizing Disconnectedness



Relational Model



Grounding

Parfactors Fs:

$$\{\langle\{\}, \{S(x,y), R(x,y)\}, t_1\rangle$$
$$\langle\{\}, \{Q(x), R(x,y)\}, t_2\rangle\}$$

$lrc(Con, Fs) = lrc(Con, Fs\{x/C\})^n$

...now we only have unary predicates

Observations and Queries

- Observations become the initial context.
  Observations can be ground or lifted.
- $P(q|obs) = rc(q \wedge obs, Fs)/(rc(q \wedge obs, Fs) + rc(\neg q \wedge obs, Fs))$
  calls can share the cache
- "How many?" queries are also allowed

Complexity

As the population size n of undifferentiated individuals increases:

- If grounding is polynomial — instances must be disconnected — lifted inference is constant in n (taking r^n for real r)
- Otherwise, for unary relations, grounding is exponential and lifted inference is polynomial.
- If non-unary relations become unary, above holds.
- Otherwise, ground an argument.
  Always exponentially better than grounding everything.

What we can and cannot lift

We can lift a model that consists just of

$$\langle \{x, z\}, \{F(x), \neg G(z)\}, \alpha_4 \rangle$$

or just of

$$\langle \{x, y, z\}, \{F(x, z), G(y, z)\}, \alpha_2 \rangle$$

or just of

$$\langle \{x, y, z\}, \{F(x, z), G(y, z), H(y)\}, \alpha_3 \rangle$$

We cannot lift (still exponential) a model that consists just of:

$$\langle \{x, y, z, w\}, \{F(x, z), G(y, z), H(y, w)\}, \alpha_3 \rangle$$

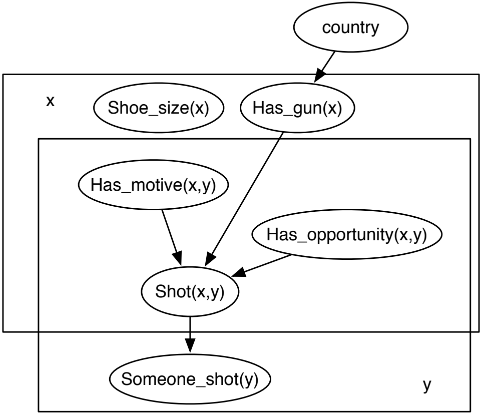
or

$$\langle \{x, y, z\}, \{F(x, z), G(y, z), H(y, x)\}, \alpha_3 \rangle$$

Outline

1. Relational Graphical Models
2. Exact Inference
   - Recursive Conditioning
   - Lifted Inference
   - Lifted Recursive Conditioning
3. Lifting Probabilistic Programs (?)

Example: Predicting Relations



Fred has unusual shoe size. Someone with unusual shoe size shot Joe. What is the probability Fred shot Joe?

Probabilistic Program

```
america := draw(0.2)
for x in range(0,1000000):
   size_23_shoe[x] := draw(0.00001)
   if america: has_gun[x] := draw(0.7)
      else: has_gun[x] := draw(0.02)
   for y in range(0,1000000):
      has_motive[x,y] := draw(0.001)
      has_opp[x,y] := draw(0.05)
      if has_motive[x,y] and has_gun[x] and has_opp[x,y]:
         actually_shot[x,y] := draw(0.1)
      if actually_shot[x,y]:
         someone_shot[y] := True
observe someone_shot[joe]
observe size_23_shoe[fred]
query actually_shot[fred,joe]
```

Lifting probabilistic programs?

- When we create many instances of one object, just create the “generic object”
- When we have to branch on a value; just count the qualitatively different answers
- If caching states in MCMC, assignments with the same counts can be treated as the same
- If computing some parts analytically, this provides one more technique in the toolbox

Conclusion

- Often probabilities depend on the number of individuals (even if not observed).
- Lifting exploits symmetry / exchangeability in relational models.
- Unary relations (properties) can be lifted. Binary relations cannot all be.
- Approximate lifted inference looks for cases that are approximately exchangeable or uses lifting in approximate algorithms
- Probabilistic logic programs use lifted inference.
  Can other probabilistic programming languages?