Uncertainty in Artificial Intelligence

Inference

Tinne De Laet and Luc De Raedt

2014–2015

These slides are based on the slides and the book *Bayesian Reasoning and Machine Learning* by David Barber.

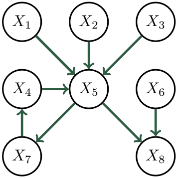
- The book and demos can be downloaded from `www.cs.ucl.ac.uk/staff/D.Barber/brml`.
- The slides have been adapted for use in the UAI course at KU Leuven.

overview

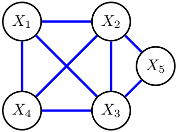
overview

1. What have we learned so far?
2. inference
3. General inference
4. message passing idea
5. sum-product
6. conclusion

- probability and probabilistic reasoning
- belief networks:
  - independence
  - expressiveness
  - simple calculations



- Markov networks:
  - independence
  - expressiveness



overview

Inference

inference

Inference corresponds to using the distribution to answers questions about the environment.

examples

- What is the probability $p(x = 4|y = 1, z = 2)$?
- What is the most likely joint state of the distribution $p(x, y)$?
- What is the entropy of the distribution $p(x, y, z)$?
- What is the probability that this example is in class 1?
- What is the probability the stock market will do down tomorrow?

inference

We can already do inference. So why bother?

$\rightarrow$ **computational efficiency**

- Inference can be computationally very expensive and we wish to characterise situations in which inferences can be computed efficiently.
- For singly-connected graphical models (trees), and certain inference questions, there exist efficient algorithms based on the concept of message passing.
- In general, the case of multiply-connected models is computationally inefficient.

overview

1. What have we learned so far?
2. inference
3. General inference
   - variable elimination
   - bucket elimination
4. message passing idea
5. sum-product
6. conclusion

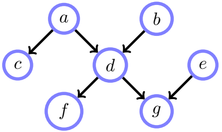
overview

1. What have we learned so far?
2. inference
3. General inference
   - variable elimination
   - bucket elimination
4. message passing idea
5. sum-product
6. conclusion

variable elimination

calculate a marginal from a joint distribution
$\Rightarrow$ marginalize over all variables except marginal variables
e.g.

$$p(f) = \sum_{a,b,c,d,e,g} p(a,b,c,d,e,f,g)$$



(a)

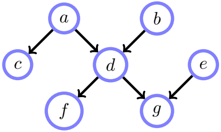
$$\begin{aligned} p(f) &= \sum_{a,b,c,d,e,g} p(a,b,c,d,e,f,g) \\ &= \sum_{a,b,c,d,e,g} p(f|d)p(g|d,e)p(c|a)p(d|a,b)p(a)p(b)p(e) \end{aligned}$$

If we push the sumations inside in order e, c, b, g, a, d

$$p(f) = \sum_d p(f|d) \sum_a p(a) \sum_g \sum_b p(d|a,b)p(b) \sum_c p(c|a) \sum_e p(g|d,e)p(e)$$

variable elimination

What would be a better order?



(a)

$$p(f) = \sum_{a,b,c,d,e,g} p(a,b,c,d,e,f,g)$$
$$= \sum_{a,b,c,d,e,g} p(f|d)p(g|d,e)p(c|a)p(d|a,b)p(a)p(b)p(e)$$

If we push the sumations inside in order g, e, c, d, b, a

$$p(f) = \sum_a p(a) \sum_b p(b) \sum_d p(f|d)p(d|a,b) \underbrace{\sum_c p(c|a) \underbrace{\sum_e p(e) \underbrace{\sum_g p(g|d,e)}_{=1}}_{=1}}_{=1}$$

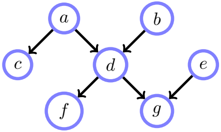
overview

1. What have we learned so far?
2. inference
3. General inference
   - variable elimination
   - bucket elimination
4. message passing idea
5. sum-product
6. conclusion

bucket elimination

bucket elimination

is a general marginal variable elimination method that works for any distribution (including multiply connected graphs). It can be considered as a way to organise the distributed summation when doing marginal variable elimination.



(a)

$$p(f) = \sum_{a,b,c,d,e,g} p(a,b,c,d,e,f,g)$$
$$= \sum_{a,b,c,d,e,g} p(f|d)p(g|d,e)p(c|a)p(d|a,b)p(a)p(b)p(e)$$

If we push the sumations inside in order e, c, b, g, a, d, f

$$p(f) = \sum_{f}\sum_{d} p(f|d) \sum_{a} p(a) \sum_{g}\sum_{b} p(d|a,b)p(b) \sum_{c} p(c|a) \sum_{e} p(g|d,e)p(e)$$

bucket elimination

bucket elimination
is a message passing algorithm that reflects this “pushing summations inside”

procedure

1. define an ordering of the variables beginning with “marginal variable”

bucket elimination

F - D - A - G - B - C - E

bucket elimination

bucket elimination
is a message passing algorithm that reflects this "pushing summations inside"

procedure

1. define an ordering of the variables beginning with "marginal variable"
2. draw the buckets starting with the "marginal variable" at the bottom

bucket elimination

bucket elimination

bucket elimination
is a message passing algorithm that reflects this "pushing summations inside"

procedure

1. define an ordering of the variables beginning with "marginal variable"
2. draw the buckets starting with the "marginal variable" at the bottom
3. distribute the potentials over the buckets in the first column:
   - start with the highest bucket and put all potentials mentioning the variable (the bucket potentials)
   - go to the next bucket and put all REMAINING potentials mentioning the variable
   - ...

bucket elimination

E $p(e)p(g|d,e)$

C $p(c|a)$

B $p(b)p(d|a,b)$

G

A $p(a)$

D $p(f|d)$

F

bucket elimination

bucket elimination
is a message passing algorithm that reflects this “pushing summations inside”

procedure

1. define an ordering of the variables beginning with “marginal variable”
2. draw the buckets starting with the “marginal variable” at the bottom
3. distribute the potentials over the buckets in the first column:
4. eliminate the buckets:
   - go to the highest (non-marginalized) bucket and:
     - marginalize the product of the bucket potentials and the bucket messages over the bucket variable
     - send the result = (message) to the highest bucket with bucket variable present in the message
     - write non-eliminated potentials and the message in the next column

bucket elimination

E	$p(e)p(g\|d,e)$	
C	$p(c\|a)$	$p(c\|a)$
B	$p(b)p(d\|a,b)$	$p(b)p(d\|a,b)$
G		$\gamma_E(d,g)$
A	$p(a)$	$p(a)$
D	$p(f\|d)$	$p(f\|d)$
F		

$$\gamma_E(d,g) = \sum_E p(e)p(g|d,e)$$

bucket elimination

E	$p(e)p(g\|d,e)$	
C	$p(c\|a)$	$p(c\|a)$
B	$p(b)p(d\|a,b)$	$p(b)p(d\|a,b)$
G		$\gamma_E(d,g)$
A	$p(a)$	$p(a)$
D	$p(f\|d)$	$p(f\|d)$
F		

$$\gamma_C() = \sum_C p(c|a) = 1$$

bucket elimination

E	$p(e)p(g\|d,e)$		
C	$p(c\|a)$	$p(c\|a)$	
B	$p(b)p(d\|a,b)$	$p(b)p(d\|a,b)$	
G		$\gamma_E(d,g)$	$\gamma_E(d,g)$
A	$p(a)$	$p(a)$	$p(a)\gamma_B(d,a)$
D	$p(f\|d)$	$p(f\|d)$	$p(f\|d)$
F			

$$\gamma_B(d,a) = \sum_B p(b)p(d|a,b)$$

bucket elimination

E	$p(e)p(g\|d,e)$			
C	$p(c\|a)$	$p(c\|a)$		
B	$p(b)p(d\|a,b)$	$p(b)p(d\|a,b)$		
G		$\gamma_E(d,g)$	$\gamma_E(d,g)$	
A	$p(a)$	$p(a)$	$p(a)\gamma_B(d,a)$	$p(a)\gamma_B(d,a)$
D	$p(f\|d)$	$p(f\|d)$	$p(f\|d)$	$p(f\|d)\gamma_G(d)$
F				

$$\gamma_G(d) = \sum_G \gamma_E(d,g)$$

bucket elimination

E	$p(e)p(g\|d,e)$				
C	$p(c\|a)$	$p(c\|a)$			
B	$p(b)p(d\|a,b)$	$p(b)p(d\|a,b)$			
G		$\gamma_E(d,g)$	$\gamma_E(d,g)$		
A	$p(a)$	$p(a)$	$p(a)\gamma_B(d,a)$	$p(a)\gamma_B(d,a)$	
D	$p(f\|d)$	$p(f\|d)$	$p(f\|d)$	$p(f\|d)\gamma_G(d)$	$p(f\|d)\gamma_G(d)\gamma_A(d)$
F					

$$\gamma_A(d) = \sum_A p(a)\gamma_B(d,a)$$

bucket elimination

E	$p(e)p(g\|d,e)$					
C	$p(c\|a)$	$p(c\|a)$				
B	$p(b)p(d\|a,b)$	$p(b)p(d\|a,b)$				
G		$\gamma_E(d,g)$	$\gamma_E(d,g)$			
A	$p(a)$	$p(a)$	$p(a)\gamma_B(d,a)$	$p(a)\gamma_B(d,a)$		
D	$p(f\|d)$	$p(f\|d)$	$p(f\|d)$	$p(f\|d)\gamma_G(d)$	$p(f\|d)\gamma_G(d)\gamma_A(d)$	
F						$\gamma_D(f)$

$$\gamma_D(f) = \sum_D p(f|d)\gamma_G(d)\gamma_A(d)$$

bucket elimination

bucket elimination
is a message passing algorithm that reflects this "pushing summations inside"

procedure

1. define an ordering of the variables beginning with "marginal variable"
2. draw the buckets starting with the "marginal variable" at the bottom
3. distribute the potentials over the buckets in the first column:
4. eliminate the buckets:
5. the product of the bucket potentials and bucket messages on the last row, last column is the marginal

bucket elimination

E	$p(e)p(g\|d,e)$					
C	$p(c\|a)$	$p(c\|a)$				
B	$p(b)p(d\|a,b)$	$p(b)p(d\|a,b)$				
G		$\gamma_E(d,g)$	$\gamma_E(d,g)$			
A	$p(a)$	$p(a)$	$p(a)\gamma_B(d,a)$	$p(a)\gamma_B(d,a)$		
D	$p(f\|d)$	$p(f\|d)$	$p(f\|d)$	$p(f\|d)\gamma_G(d)$	$p(f\|d)\gamma_G(d)\gamma_A(d)$	
F						$\gamma_D(f)$

$$p(f) = \gamma_D(f)$$

bucket elimination

remarks

- if you need **other marginal**, need to re-order the variables and **repeat bucket elimination** $\rightarrow$ not efficient
- bucket elimination constructs multi-variable messages from bucket to bucket. The **storage** requirements of a multi-variable message are in general **exponential** in the number of variables of the message.
- for **singly connected graphs: perferct ordering exists** that makes computational complexity linear in the number of variables. However, orderings exist for which bucket elimination will be extremely inefficient.

bucket elimination

things you should think about

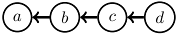
- How to handle **evidence**? (e.g. $p(f|a)$)
- How to calculate **marginals over multiple variables**? (e.g. $p(a, f)$)

$\Rightarrow$ if you have doubt just go back to the general formula (variable elimination through marginalization)

overview

message passing idea

Let's consider the following simple belief network (a Markov chain) with binary variables:



The joint factorizes as: $p(a, b, c, d) =$?

message passing idea

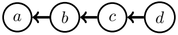
Let's consider the following simple belief network (a Markov chain) with binary variables:

$$a \leftarrow b \leftarrow c \leftarrow d$$

The joint factorizes as: $p(a, b, c, d) = \ p(a|b)\ p(b|c)\ p(c|d)\ p(d)$

message passing idea

Let's consider the following simple belief network (a Markov chain) with binary variables:



The joint factorizes as: $p(a, b, c, d) = \; p(a|b)\; p(b|c)\; p(c|d)\; p(d)$

Let's now **infer** $p(a = 0)$:

$p(a = 0) =$?

1.
2.

message passing idea

Let's consider the following simple belief network (a Markov chain) with binary variables:

$$a \leftarrow b \leftarrow c \leftarrow d$$

The joint factorizes as: $p(a, b, c, d) = \ p(a|b)\ p(b|c)\ p(c|d)\ p(d)$

Let's now **infer** $p(a = 0)$:

$p(a = 0) = \ \sum_{b,c,d} p(a = 0|b)\ p(b|c)\ p(c|d)\ p(d)$

1

2

message passing idea

Let's consider the following simple belief network (a Markov chain) with binary variables:

$$a \leftarrow b \leftarrow c \leftarrow d$$

The joint factorizes as: $p(a,b,c,d) = \; p(a|b)\; p(b|c)\; p(c|d)\; p(d)$

Let's now **infer** $p(a=0)$:

$p(a=0) = \; \sum_{b,c,d} p(a=0|b)\; p(b|c)\; p(c|d)\; p(d)$

1. **First procedure:** calculate the summation
   $\rightarrow$ we need **x** summations
2.

message passing idea

Let's consider the following simple belief network (a Markov chain) with binary variables:

$$a \leftarrow b \leftarrow c \leftarrow d$$

The joint factorizes as: $p(a, b, c, d) = \; p(a|b)\; p(b|c)\; p(c|d)\; p(d)$

Let's now **infer** $p(a = 0)$:

$p(a = 0) = \; \sum_{b,c,d} p(a = 0|b)\; p(b|c)\; p(c|d)\; p(d)$

1. **First procedure:** calculate the summation
   $\rightarrow$ we need **7** summations

2.

message passing idea

Let's consider the following simple belief network (a Markov chain) with binary variables:

$$a \leftarrow b \leftarrow c \leftarrow d$$

The joint factorizes as: $p(a, b, c, d) = \; p(a|b)\; p(b|c)\; p(c|d)\; p(d)$

Let's now **infer** $p(a = 0)$:

$p(a = 0) = \; \sum_{b,c,d} p(a = 0|b)\; p(b|c)\; p(c|d)\; p(d)$

1. **First procedure:** calculate the summation
   $\rightarrow$ we need **7** summations
2. **Second procedure:** push summations inside

   - push d: $p(a = 0) =$

message passing idea

Let's consider the following simple belief network (a Markov chain) with binary variables:

$$a \leftarrow b \leftarrow c \leftarrow d$$

The joint factorizes as: $p(a, b, c, d) = \; p(a|b)\; p(b|c)\; p(c|d)\; p(d)$

Let's now **infer** $p(a = 0)$:

$p(a = 0) = \; \sum_{b,c,d} p(a = 0|b)\; p(b|c)\; p(c|d)\; p(d)$

1. **First procedure:** calculate the summation
   $\rightarrow$ we need **7** summations
2. **Second procedure:** push summations inside
   - push d: $p(a = 0) = \sum_{b,c} p(a = 0|b)\; p(b|c)\; \underbrace{\sum_d \left(p(c|d)\; p(d)\right)}_{\gamma_d(c)}$

   $\rightarrow$ for $\gamma_d(c)$ we need **x** summations

message passing idea

Let's consider the following simple belief network (a Markov chain) with binary variables:

$$a \leftarrow b \leftarrow c \leftarrow d$$

The joint factorizes as: $p(a,b,c,d) = \; p(a|b)\; p(b|c)\; p(c|d)\; p(d)$

Let's now **infer** $p(a=0)$:

$p(a=0) = \; \sum_{b,c,d} p(a=0|b)\; p(b|c)\; p(c|d)\; p(d)$

1. **First procedure:** calculate the summation
   $\rightarrow$ we need **7** summations
2. **Second procedure:** push summations inside

   - push d: $p(a=0) = \sum_{b,c} p(a=0|b)\; p(b|c)\; \underbrace{\sum_{d} (p(c|d)\; p(d))}_{\gamma_d(c)}$

   $\rightarrow$ for $\gamma_d(c)$ we need **2** summations

message passing idea - continued

1. **Second procedure:** push summations inside

   - **push sum d**: $p(a=0) = \sum_{b,c} p(a=0|b)\ p(b|c)\ \underbrace{\sum_d \left(p(c|d)\ p(d)\right)}_{\gamma_d(c)}$

     $\rightarrow$ for $\gamma_d(c)$ we need **2** summations
   - **push sum c**: $p(a=0) =$
   - **push sum b**: $p(a=0) =$

message passing idea - continued

1. **Second procedure:** push summations inside
   - **push sum d**: $p(a=0) = \sum_{b,c} p(a=0|b)\ p(b|c) \underbrace{\sum_{d} (p(c|d)\ p(d))}_{\gamma_d(c)}$

     $\rightarrow$ for $\gamma_d(c)$ we need **2** summations
   - **push sum c**: $p(a=0) = \sum_{b} p(a=0|b) \underbrace{\sum_{c} (p(b|c)\ \gamma_d(c))}_{\gamma_c(b)}$

     $\rightarrow$ for $\gamma_c(b)$ we need **x** summations
   - **push sum b**: $p(a=0) =$

message passing idea - continued

1. **Second procedure:** push summations inside
   - **push sum d**: $p(a=0) = \sum_{b,c} p(a=0|b)\ p(b|c) \underbrace{\sum_d (p(c|d)\ p(d))}_{\gamma_d(c)}$

     $\rightarrow$ for $\gamma_d(c)$ we need **2** summations
   - **push sum c**: $p(a=0) = \sum_b p(a=0|b) \underbrace{\sum_c (p(b|c)\ \gamma_d(c))}_{\gamma_c(b)}$

     $\rightarrow$ for $\gamma_c(b)$ we need **2** summations
   - **push sum b**: $p(a=0) =$

message passing idea - continued

1. **Second procedure:** push summations inside
   - **push sum d**: $p(a=0) = \sum_{b,c} p(a=0|b)\ p(b|c)\ \underbrace{\sum_{d} (p(c|d)\ p(d))}_{\gamma_d(c)}$

     $\rightarrow$ for $\gamma_d(c)$ we need **2** summations
   - **push sum c**: $p(a=0) = \sum_{b} p(a=0|b)\ \underbrace{\sum_{c} (p(b|c)\ \gamma_d(c))}_{\gamma_c(b)}$

     $\rightarrow$ for $\gamma_c(b)$ we need **2** summations
   - **push sum b**: $p(a=0) = \underbrace{\sum_{b} p(a=0|b)\ \gamma_c(b)}_{\gamma_b(a)}$

     $\rightarrow$ for $\gamma_b(a)$ we need **x** summations

message passing idea - continued

1. **Second procedure:** push summations inside

   - **push sum d**: $p(a=0) = \sum_{b,c} p(a=0|b)\ p(b|c)\ \underbrace{\sum_d (p(c|d)\ p(d))}_{\gamma_d(c)}$

     $\rightarrow$ for $\gamma_d(c)$ we need **2** summations
   - **push sum c**: $p(a=0) = \sum_b p(a=0|b)\ \underbrace{\sum_c (p(b|c)\ \gamma_d(c))}_{\gamma_c(b)}$

     $\rightarrow$ for $\gamma_c(b)$ we need **2** summations
   - **push sum b**: $p(a=0) = \underbrace{\sum_b p(a=0|b)\ \gamma_c(b)}_{\gamma_b(a)}$

     $\rightarrow$ for $\gamma_b(a)$ we need **1** summations

message passing idea - continued

1. **Second procedure:** push summations inside

   - **push sum d**: $p(a=0) = \sum_{b,c} p(a=0|b)\ p(b|c)\ \underbrace{\sum_{d} (p(c|d)\ p(d))}_{\gamma_d(c)}$

     $\rightarrow$ for $\gamma_d(c)$ we need **2** summations
   - **push sum c**: $p(a=0) = \sum_{b} p(a=0|b)\ \underbrace{\sum_{c} (p(b|c)\ \gamma_d(c))}_{\gamma_c(b)}$

     $\rightarrow$ for $\gamma_c(b)$ we need **2** summations
   - **push sum b**: $p(a=0) = \underbrace{\sum_{b} p(a=0|b)\ \gamma_c(b)}_{\gamma_b(a)}$

     $\rightarrow$ for $\gamma_b(a)$ we need **1** summations

$\rightarrow$ in total we need **5** summations

message passing idea - continued

variable elimination

second procedure = **variable elimination**: each time you sum to eliminate one variables

general Markov chain with $T+1$ binary variables length:

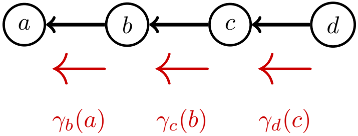
1. **first procedure:** calculating sum requires $2^T - 1$ summations $\rightarrow$ **exponential!**
2. **second procedure:** variable elimination $2T - 1$ summations $\rightarrow$ **linear!**

variable elimination and message passing

variable elimination can be seen as message passing

- messages from one node to the other
- messages are $\gamma_i(j)$, unnormalized potentials in general

message passing idea - continued



$$p(a=0) = \underbrace{\sum_b p(a=0|b) \; \underbrace{\sum_c p(b|c) \; \underbrace{\sum_d p(c|d) \; p(d)}_{\gamma_d(c)}}_{\gamma_c(b)}}_{\gamma_b(a)}$$

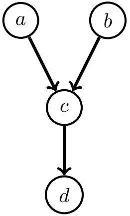
overview

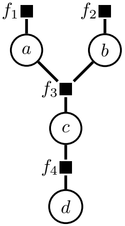
1. What have we learned so far?
2. inference
3. General inference
4. message passing idea
5. sum-product
   - example
6. conclusion

sum-product algorithm

sum-product algorithm
inference algorithm based on message passing for singly-connected factor graphs (trees)

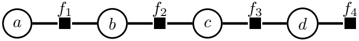
factor graph
both Markov and belief networks can be represented as factor graphs
example: belief network $p(a,b,c,d) = \underbrace{p(a)}_{f_1}\ \underbrace{p(b)}_{f_2}\ \underbrace{p(c|a,b)}_{f_3}\ \underbrace{p(d|c)}_{f_4}$





sum-product algorithm - non branching tree

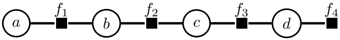
$p(a,b,c,d) \propto f_1\,(a,b)\, f_2\,(b,c)\, f_3\,(c,d)\, f_4\,(d)$ $\quad a,b,c,d$ binary variables



$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a,b,c,d) \\
&\propto \sum_{b,c,d} f_1\,(a,b)\, f_2\,(b,c)\, f_3\,(c,d)\, f_4\,(d) \Rightarrow 2^3 \text{ sums}
\end{aligned}
$$

sum-product algorithm - non branching tree

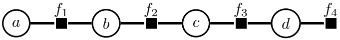
$p(a,b,c,d) \propto f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) \quad a,b,c,d$ binary variables



$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a,b,c,d) \\
&\propto \sum_{b,c,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) \Rightarrow 2^3 \text{ sums} \\
&= \sum_{b} f_1(a,b) \sum_{c} f_2(b,c) \sum_{d} f_3(c,d)\, f_4(d) \Rightarrow 2 \times 3 \text{ sums}
\end{aligned}
$$

sum-product algorithm - non branching tree

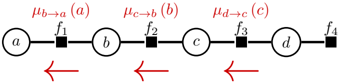
$p(a,b,c,d) \propto f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) \quad a,b,c,d$ binary variables



$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a,b,c,d) \\
&\propto \sum_{b,c,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) \\
&= \underbrace{\sum_b f_1(a,b) \underbrace{\sum_c f_2(b,c) \underbrace{\sum_d f_3(c,d)\, f_4(d)}_{\mu_{d\to c}(c)}}_{\mu_{c\to b}(b)}}_{\mu_{b\to a}(a)}
\end{aligned}
$$

sum-product algorithm - non branching tree

$p(a,b,c,d) \propto f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d)$ $\quad a,b,c,d$ binary variables



Passing variable-to-variable messages from d up to a

$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a,b,c,d) \\
&\propto \sum_{b,c,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) \\
&= \underbrace{\sum_b f_1(a,b) \underbrace{\sum_c f_2(b,c) \underbrace{\sum_d f_3(c,d)\, f_4(d)}_{\mu_{d\to c}(c)}}_{\mu_{c\to b}(b)}}_{\mu_{b\to a}(a)}
\end{aligned}
$$

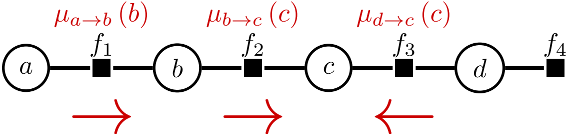
sum-product algorithm - non-branching tree

What if we want to calculate $p(c)$?

$$p(c) \propto \sum_{a,b,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d)$$

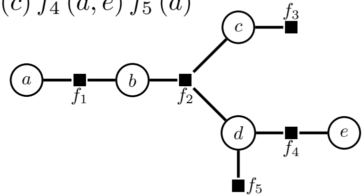
$$= \underbrace{\sum_b \underbrace{\sum_a f_1(a,b)}_{\mu_{a\to b}(b)} f_2(b,c)}_{\mu_{b\to c}(c)} \underbrace{\sum_d f_3(c,d)\, f_4(d)}_{\mu_{d\to c}(c)}$$

$\rightarrow$ **need to send messages in both directions**



sum-product algorithm – branching tree

$$p(a,b,c,d,e) \propto f_1(a,b)\, f_2(b,c,d)\, f_3(c)\, f_4(d,e)\, f_5(d)$$



to define factor-to-variable messages and variable-to-factor messages

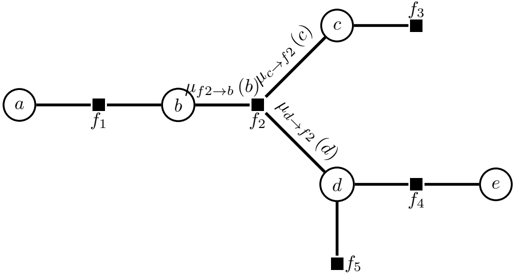
$$p(a) \propto \underbrace{\sum_b f_1(a,b) \underbrace{\sum_{c,d} f_2(b,c,d) \underbrace{f_3(c)}_{\mu_{c\to f_2}(c)=\mu_{f_3\to c}(c)} \underbrace{\underbrace{f_5(d)}_{\mu_{f_5\to d}(d)} \underbrace{\sum_e f_4(d,e)}_{\mu_{f_4\to d}(d)}}_{\mu_{d\to f_2}(d)}}_{\mu_{b\to f_1}(b)=\mu_{f_2\to b}(b)}}_{\mu_{f_1\to a}(a)}$$

$\Rightarrow$ **marginal inference for a singly-connected structure is easy**.

sum-product algorithm – branching tree

message schedule: a message can be sent from a node or factor only when that node has received all requisite messages from its neigbours.

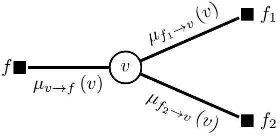
$$p(a,b,c,d,e) \propto f_1(a,b)\, f_2(b,c,d)\, f_3(c)\, f_4(d,e)\, f_5(d)$$



sum-product algorithm for factor graphs

variable to factor message

$$\mu_{v\to f}(v) = \prod_{f_i \sim v \backslash f} \mu_{f_i\to v}(v)$$

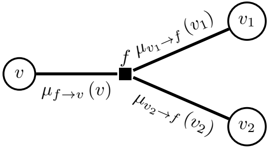


messages from extremal variables are set to 1

---

factor to variable message

$$\mu_{f\to v}(v) = \sum_{\{v_i\}} f(v, \{v_i\}) \prod_{v_i \sim f \backslash v} \mu_{v_i\to f}(v_i)$$



messages from extremal factors are set to the factor

Marginal

$p(v) \propto \prod_{f_i \sim v} \mu_{f_i\to v}(v)$

sum-product algorithm for factor graphs

In a tree exact inference of all the marginals can be done by two passes of the sum-product algorithm

procedure

1. pick one node as the root node

sum-product algorithm for factor graphs

In a tree exact inference of all the marginals can be done by two passes of the sum-product algorithm

procedure

1. pick one node as the root node
2. initialize:
    - messages from leaf factor nodes initialized to factors

sum-product algorithm for factor graphs

In a tree exact inference of all the marginals can be done by two passes of the sum-product algorithm

procedure

1. pick one node as the root node
2. initialize:
    - messages from leaf factor nodes initialized to factors
    - messages from leaf variable nodes set to unity

sum-product algorithm for factor graphs

In a tree exact inference of all the marginals can be done by two passes of the sum-product algorithm

procedure

1. pick one node as the root node
2. initialize:
   - messages from leaf factor nodes initialized to factors
   - messages from leaf variable nodes set to unity
3. step 1: propagate messages from leaves to root

sum-product algorithm for factor graphs

In a tree exact inference of all the marginals can be done by two passes of the sum-product algorithm

procedure

1. pick one node as the root node
2. initialize:
   - messages from leaf factor nodes initialized to factors
   - messages from leaf variable nodes set to unity
3. step 1: propagate messages from leaves to root
4. step 2: propagate messages from root to leaves

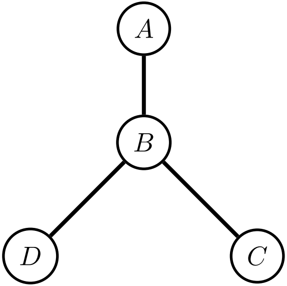
overview

1. What have we learned so far?
2. inference
3. General inference
4. message passing idea
5. sum-product
   - example
6. conclusion

sum-product example

$$p(A, B, C, D) \propto f_1(A, B)\ f_2(B, C)\ f_3(B, D)\ f_4(C)$$

A, B, C, D binary variables



A	B	$f_1(A, B)$
0	0	10
0	1	1
1	0	1
1	1	10

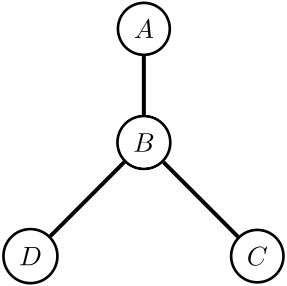
B	D	$f_3(B, D)$
0	0	10
0	1	1
1	0	1
1	1	10

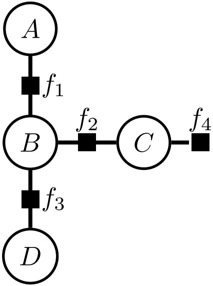
B	C	$f_2(B, C)$
0	0	1
0	1	10
1	0	10
1	1	1

C	$f_4(C)$
0	10
1	1

sum-product example

convert to factor graph





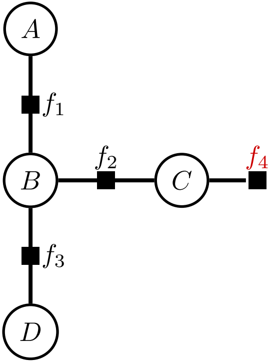
$$p(A,B,C,D) \propto f_1(A,B)\ \ f_2(B,C)\ \ f_3(B,D)\ \ f_4(C)$$

A, B, C, D binary variables

sum-product example

procedure

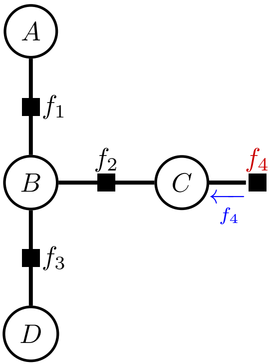
1. pick one node as the root node



sum-product example

procedure

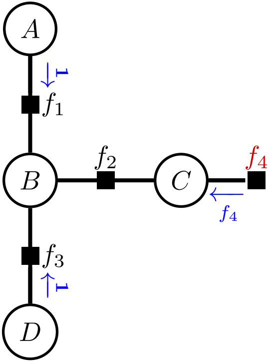
1. pick one node as the root node
2. initialize:
   - messages from leaf factor nodes initialized to factors



sum-product example

procedure

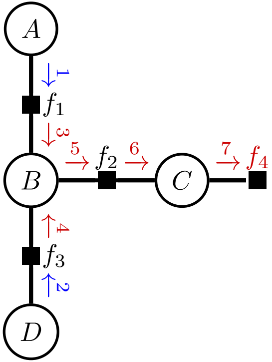
1. pick one node as the root node
2. <u>initialize</u>:
   - messages from leaf factor nodes initialized to factors
   - messages from leaf variable nodes set to unity



sum-product example

procedure

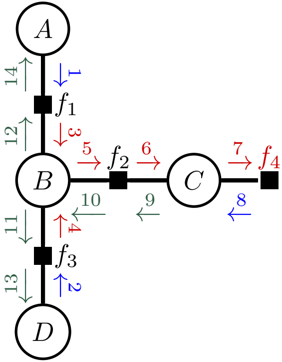
1. pick one node as the root node
2. initialize:
   - messages from leaf factor nodes initialized to factors
   - messages from leaf variable nodes set to unity
3. step 1: propagate messages from leaves to root



sum-product example

procedure

1. pick one node as the root node
2. initialize:
   - messages from leaf factor nodes initialized to factors
   - messages from leaf variable nodes set to unity
3. step 1: propagate messages from leaves to root
4. step 2: propagate messages from root to leaves



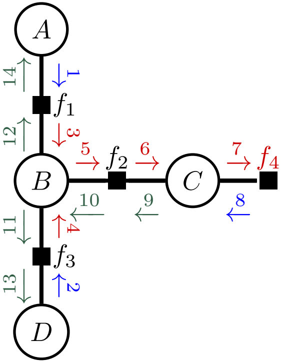
sum-product example

- 1 was initialized to one so:

$$\begin{aligned} \mu_{A\to f_1}(A) &= \mathbf{1} \\ \mu_{A\to f_1}(0) &= 1 \\ \mu_{A\to f_1}(1) &= 1 \end{aligned}$$

- 2 was initialized to one so:

$$\begin{aligned} \mu_{D\to f_3}(D) &= \mathbf{1} \\ \mu_{D\to f_3}(0) &= 1 \\ \mu_{D\to f_3}(1) &= 1 \end{aligned}$$



sum-product example

- 3

$$\mu_{f_1 \to B}(B) = \sum_A f_1(A,B)\,\mu_{A \to f_1}(A)$$
$$= f_1(A=0,B)\,\mu_{A \to f_1}(0) + f_1(A=1,B)\,\mu_{A \to f_1}(1)$$

$$\mu_{f_1 \to B}(0) = \sum_A f_1(A,B=0)\,\mu_{A \to f_1}(A)$$
$$= \underbrace{f_1(A=0,B=0)}_{10}\,\underbrace{\mu_{A \to f_1}(0)}_{1} + \underbrace{f_1(A=1,B=0)}_{1}\,\underbrace{\mu_{A \to f_1}(1)}_{1} = 11$$

$$\mu_{f_1 \to B}(1) = \sum_A f_1(A,B=1)\,\mu_{A \to f_1}(A)$$
$$= \underbrace{f_1(A=0,B=1)}_{1}\,\underbrace{\mu_{A \to f_1}(0)}_{1} + \underbrace{f_1(A=1,B=1)}_{10}\,\underbrace{\mu_{A \to f_1}(1)}_{1} = 11$$

sum-product example

- 4

$$\mu_{f_3\to B}(B) = \sum_D f_3(B,D)\,\mu_{D\to f_3}(D)$$
$$= f_3(B,D=0)\,\mu_{D\to f_3}(0) + f_3(B,D=1)\,\mu_{D\to f_3}(1)$$

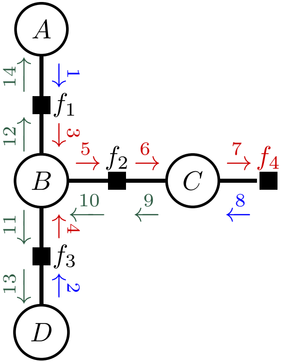
$$\mu_{f_3\to B}(0) = \sum_D f_3(B=0,D)\,\mu_{D\to f_3}(D)$$
$$= \underbrace{f_3(B=0,D=0)}_{10}\,\underbrace{\mu_{D\to f_3}(0)}_{1} + \underbrace{f_3(B=0,D=1)}_{1}\,\underbrace{\mu_{D\to f_3}(1)}_{1} = 11$$

$$\mu_{f_3\to B}(1) = \sum_D f_3(B=1,D)\,\mu_{D\to f_3}(D)$$
$$= \underbrace{f_3(B=1,D=0)}_{1}\,\underbrace{\mu_{D\to f_3}(0)}_{1} + \underbrace{f_3(B=1,D=1)}_{10}\,\underbrace{\mu_{D\to f_3}(1)}_{1} = 11$$

sum-product example

- 5

$$\mu_{B\to f_2}(B) = \mu_{f_1\to B}(B)\,\mu_{f_3\to B}(B)$$
$$\mu_{B\to f_2}(0) = \underbrace{\mu_{f_1\to B}(0)}_{11}\underbrace{\mu_{f_3\to B}(0)}_{11} = 121$$
$$\mu_{B\to f_2}(1) = \underbrace{\mu_{f_1\to B}(1)}_{11}\underbrace{\mu_{f_3\to B}(1)}_{11} = 121$$



sum-product example

- 6

$$\mu_{f_2 \to C}(C) = \sum_B f_2(B, C)\, \mu_{B \to f_2}(B)$$
$$= f_2(B=0, C)\, \mu_{B \to f_2}(0) + f_2(B=1, C)\, \mu_{B \to f_2}(1)$$

$$\mu_{f_2 \to C}(0) = \sum_B f_2(B, C=0)\, \mu_{B \to f_2}(B)$$
$$= \underbrace{f_2(B=0, C=0)}_{1} \underbrace{\mu_{B \to f_2}(0)}_{121} + \underbrace{f_2(B=1, C=0)}_{10} \underbrace{\mu_{B \to f_2}(1)}_{121} = 1331$$

$$\mu_{f_2 \to C}(1) = \sum_B f_2(B, C=1)\, \mu_{B \to f_2}(B)$$
$$= \underbrace{f_2(B=0, C=1)}_{10} \underbrace{\mu_{B \to f_2}(0)}_{121} + \underbrace{f_2(B=1, C=1)}_{1} \underbrace{\mu_{B \to f_2}(1)}_{121} = 1331$$

sum-product example

- 7

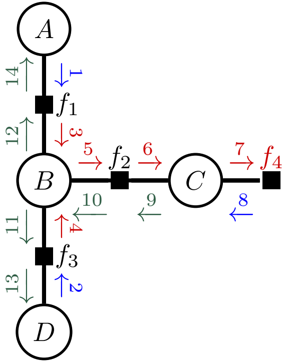
$$\begin{aligned} \mu_{C\to f_4}(C) &= \mu_{f_2\to C}(C) \\ \mu_{C\to f_4}(0) &= \mu_{f_2\to C}(0) = 1331 \\ \mu_{C\to f_4}(1) &= \mu_{f_2\to C}(1) = 1331 \end{aligned}$$

- 8 was initialized to the factor:

$$\begin{aligned} \mu_{f_4\to C}(C) &= f_4(C) \\ \mu_{f_4\to C}(0) &= f_4(C=0) = 10 \\ \mu_{f_4\to C}(1) &= f_4(C=1) = 1 \end{aligned}$$

- 9

$$\begin{aligned} \mu_{C\to f_2}(C) &= \mu_{f_4\to C}(C) \\ \mu_{C\to f_2}(0) &= \mu_{f_4\to C}(C)\, C = 0 = 10 \\ \mu_{C\to f_2}(1) &= \mu_{f_4\to C}(C)\, C = 1 = 1 \end{aligned}$$



sum-product example

- 10

$$\mu_{f_2 \to B}(B) = \sum_C f_2(B, C)\, \mu_{C \to f_2}(C)$$
$$= f_2(B, C=0)\, \mu_{C \to f_2}(0) + f_2(B, C=1)\, \mu_{C \to f_2}(1)$$

$$\mu_{f_2 \to B}(0) = \sum_C f_2(B=0, C)\, \mu_{C \to f_2}(C)$$
$$= \underbrace{f_2(B=0, C=0)}_{1}\, \underbrace{\mu_{C \to f_2}(0)}_{10} + \underbrace{f_2(B=0, C=1)}_{10}\, \underbrace{\mu_{C \to f_2}(1)}_{1} = 20$$

$$\mu_{f_2 \to B}(1) = \sum_C f_2(B=1, C)\, \mu_{C \to f_2}(C)$$
$$= \underbrace{f_2(B=1, C=0)}_{10}\, \underbrace{\mu_{C \to f_2}(0)}_{10} + \underbrace{f_2(B=1, C=1)}_{1}\, \underbrace{\mu_{C \to f_2}(1)}_{1} = 101$$

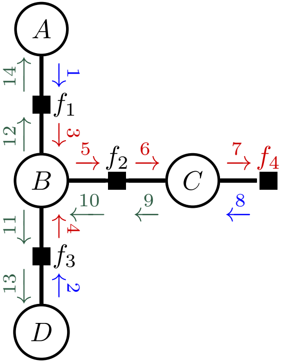
sum-product example

- 11

$$\mu_{B\to f_3}(B) = \mu_{f_1\to B}(B)\,\mu_{f_2\to B}(B)$$
$$\mu_{B\to f_3}(0) = \underbrace{\mu_{f_1\to B}(0)}_{11}\underbrace{\mu_{f_2\to B}(0)}_{20} = 220$$
$$\mu_{B\to f_3}(1) = \underbrace{\mu_{f_1\to B}(1)}_{11}\underbrace{\mu_{f_2\to B}(1)}_{101} = 1111$$

- 12

$$\mu_{B\to f_1}(B) = \mu_{f_2\to B}(B)\,\mu_{f_3\to B}(B)$$
$$\mu_{B\to f_1}(0) = \underbrace{\mu_{f_2\to B}(0)}_{20}\underbrace{\mu_{f_3\to B}(0)}_{11} = 220$$
$$\mu_{B\to f_1}(1) = \underbrace{\mu_{f_2\to B}(1)}_{101}\underbrace{\mu_{f_3\to B}(1)}_{11} = 1111$$



sum-product example

- 13

$$\mu_{f_3\to D}(D) = \sum_B f_3(B,D)\,\mu_{B\to f_3}(B)$$
$$= f_3(B=0,D)\,\mu_{B\to f_3}(0) + f_3(B=1,D)\,\mu_{B\to f_3}(1)$$

$$\mu_{f_3\to D}(0) = \sum_B f_3(B,D=0)\,\mu_{B\to f_3}(B)$$
$$= \underbrace{f_3(B=0,D=0)}_{10}\underbrace{\mu_{B\to f_3}(0)}_{220} + \underbrace{f_3(B=1,D=0)}_{1}\underbrace{\mu_{B\to f_3}(1)}_{1111} = 3311$$

$$\mu_{f_3\to D}(1) = \sum_B f_3(B,D=1)\,\mu_{B\to f_3}(B)$$
$$= \underbrace{f_3(B=0,D=1)}_{1}\underbrace{\mu_{B\to f_3}(0)}_{220} + \underbrace{f_3(B=1,D=1)}_{10}\underbrace{\mu_{B\to f_3}(1)}_{1111} = 11330$$

sum-product example

- 14

$$\mu_{f_1 \to A}(A) = \sum_B f_1(A,B)\,\mu_{B \to f_1}(B)$$
$$= f_1(A, B=0)\,\mu_{B \to f_1}(0) + f_1(A, B=1)\,\mu_{B \to f_1}(1)$$

$$\mu_{f_1 \to A}(0) = \sum_B f_1(A=0,B)\,\mu_{B \to f_1}(B)$$
$$= \underbrace{f_1(A=0, B=0)}_{10}\,\underbrace{\mu_{B \to f_1}(0)}_{220} + \underbrace{f_1(A=0, B=1)}_{1}\,\underbrace{\mu_{B \to f_1}(1)}_{1111} = 3311$$

$$\mu_{f_1 \to A}(1) = \sum_B f_1(A=1,B)\,\mu_{B \to f_1}(B)$$
$$= \underbrace{f_1(A=1, B=0)}_{1}\,\underbrace{\mu_{B \to f_1}(0)}_{220} + \underbrace{f_1(A=1, B=1)}_{10}\,\underbrace{\mu_{B \to f_1}(1)}_{1111} = 11330$$

sum-product example

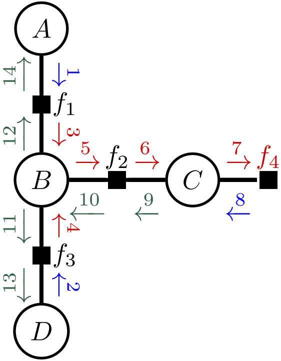
calculating marginals $p(v) \propto \prod_{f_i \sim v} \mu_{f_i \to v}(v)$

- A

$$p(A) \propto \mu_{f_1 \to A}(A)$$
$$p(A=0) = \frac{3311}{3311 + 11330} = 0.23$$
$$p(A=1) = \frac{11330}{3311 + 11330} = 0.77$$

- D

$$p(D) \propto \mu_{f_3 \to D}(D)$$
$$p(D=0) = \frac{3311}{3311 + 11330} = 0.23$$
$$p(D=1) = \frac{11330}{3311 + 11330} = 0.77$$



sum-product example

calculating marginals $p(v) \propto \prod_{f_i \sim v} \mu_{f_i \to v}(v)$

- B

$$p(B) \propto \mu_{f_1 \to B}(B)\, \mu_{f_2 \to B}(B)\, \mu_{f_3 \to B}(B)$$

$$p(B=0) = \frac{11 * 20 * 11}{11 * 20 * 11 + 11 * 101 * 11} = 0.17$$

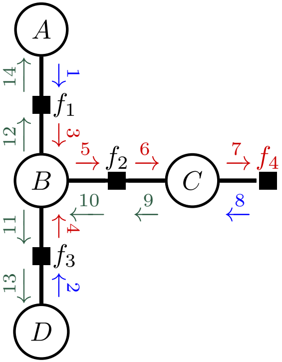
$$p(B=1) = \frac{11 * 101 * 11}{11 * 20 * 11 + 11 * 101 * 11} = 0.83$$

- C

$$p(C) \propto \mu_{f_2 \to C}(C)\, \mu_{f_4 \to C}(C)$$

$$p(C=0) = \frac{1331 * 10}{1331 * 10 + 1331 * 1} = 0.91$$

$$p(C=1) = \frac{1331 * 1}{1331 * 10 + 1331 * 1} = 0.09$$



sum-product notes

---

dealing with evidence
for all evidence variables x with corresponding value e introduce an indicator function $\delta(x, e) = 1$ and $\delta(x, e') = 0$ when $e' \neq e$
$\rightarrow$ **exercise session**

---

normalization for calculating marginals

$$Z = \sum_{\mathcal{X}} \prod_{f} \phi_f(\mathcal{X}_f) \text{ as } Z = \sum_{x} \prod_{f \in ne(x)} \mu_{f \rightarrow x}(x)$$

---

use logarithm
multiplications can result in very small values $\rightarrow$ use logarithm and summation instead $\lambda = \log \mu$

overview

inference

two inference methods

1. **bucket elimination:** general inference method even for multiply connected graphs
2. **sum-product algorithm**: message passing inference method for singly connected graphs

inference using message passing

inference using message passing

- also known as **'belief propagation' or 'dynamic programming'**
- for **non-branching graphs** (they look like 'lines'), **only variable-to-variable messages** are required
- for **branching but singly connected graphs** , **message-to-variable and variable-to-message messages** are required
- for message passing to work
  1. the operator over the factors has to be able to be distributed (the operator algebra is a **semiring**)
  2. the graph is **singly-connected**
- if the above conditions hold, 'marginal' inference **scales linearly with the number of nodes** in the graph.