Probabilistic Graphical Models, Inference via Message-Passing

Piyush Rai

Topics in Probabilistic Modeling and Inference (CS698X)

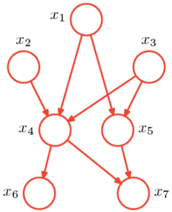
April 10, 2019

Outline for today

- Directed Graphical Models (DGM)
  - Have already seen these before in almost every model we studied!
- Checking conditional independence in DGM
- Undirected Graphical Models (UGM)
- Message Passing algorithms for inference in DGM/UGM

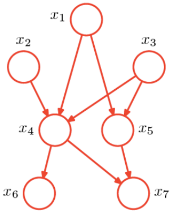
Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph



Directed Graphical Models (DGM)

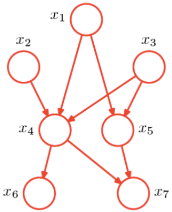
- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph



- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph

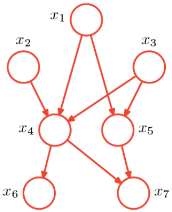


- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$$

Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph

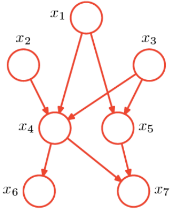


- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1)$$

Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph

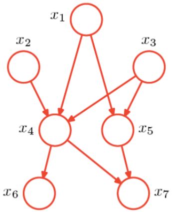


- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1)p(x_2)$$

Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph

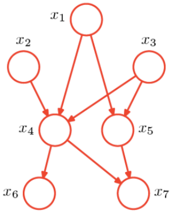


- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1)p(x_2)p(x_3)$$

Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph

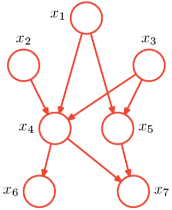


- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)$$

Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph

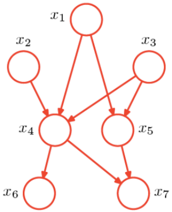


- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)$$

Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph

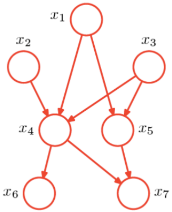


- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$

Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph



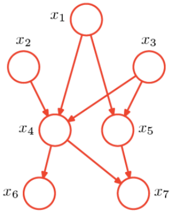
- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$

- Directed GMs represent the joint distribution as a product of "local" conditional distributions

Directed Graphical Models (DGM)

- Have already seen and used these many times. Also known as Bayesian Networks or Bayes Nets
- Basically, represent the joint distribution of a set of random variables using a directed acyclic graph



- Vertices denotes r.v. and structure of the graph directly tells us the conditional dependencies

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$

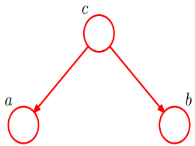
- Directed GMs represent the joint distribution as a product of "local" conditional distributions
  - In a DGM, the local conditional of a node x_k only depends on its parent nodes pa_k

$$p(\boldsymbol{x}) = \prod_{k=1}^{K} p(x_k|\text{pa}_k)$$

DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

Note: Shaded = node's value known



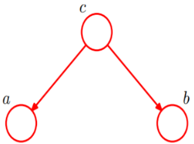
$$
\begin{aligned}
p(a,b,c) &= p(a|c)p(b|c)p(c) \\
p(a,b) &= \sum_c p(a|c)p(b|c)p(c) \\
a \not\perp b &\mid \emptyset
\end{aligned}
$$

Figure courtesy: PRML (Bishop)

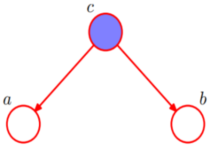
DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

Note: Shaded = node's value known



$$
\begin{aligned}
p(a, b, c) &= p(a|c)p(b|c)p(c) \\
p(a, b) &= \sum_c p(a|c)p(b|c)p(c) \\
a \not\perp\!\!\!\perp b &\mid \emptyset
\end{aligned}
$$



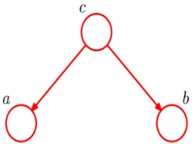
$$
\begin{aligned}
p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\
&= p(a|c)p(b|c) \\
a \perp\!\!\!\perp b &\mid c
\end{aligned}
$$

Figure courtesy: PRML (Bishop)

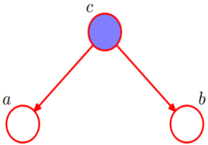
DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

Note: Shaded = node's value known



$$
\begin{aligned}
p(a,b,c) &= p(a|c)p(b|c)p(c) \\
p(a,b) &= \sum_c p(a|c)p(b|c)p(c) \\
a \not\perp\!\!\!\perp b &\mid \emptyset
\end{aligned}
$$



$$
\begin{aligned}
p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\
&= p(a|c)p(b|c) \\
a \perp\!\!\!\perp b &\mid c
\end{aligned}
$$

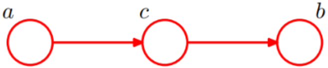
Thus conditioning on c makes a and b conditionally independent

Figure courtesy: PRML (Bishop)

DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

Note: Shaded = node's value known



$$p(a, b, c) = p(a)p(c|a)p(b|c)$$

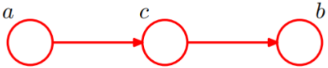
$$p(a, b) = p(a)\sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

$$a \not\perp b | \emptyset$$

DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

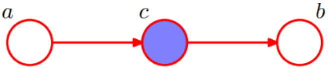
Note: Shaded = node's value known



$$p(a, b, c) = p(a)p(c|a)p(b|c)$$

$$p(a, b) = p(a)\sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

$$a \not\perp\!\!\!\perp b|\emptyset$$



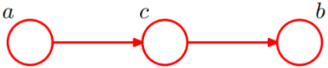
$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

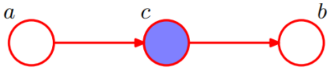
$$a \perp\!\!\!\perp b|c$$

DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

Note: Shaded = node's value known





$$p(a, b, c) = p(a)p(c|a)p(b|c)$$

$$p(a, b) = p(a)\sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

$$a \not\perp\!\!\!\perp b | \emptyset$$

$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

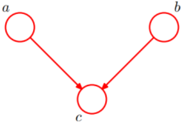
$$a \perp\!\!\!\perp b | c$$

Thus conditioning on c makes a and b conditionally independent

DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

Note: Shaded = node's value known

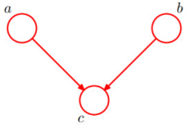


$$\begin{aligned} p(a,b,c) &= p(a)p(b)p(c|a,b) \\ p(a,b) &= p(a)p(b) \\ a \perp\!\!\!\perp b &\mid \emptyset \end{aligned}$$

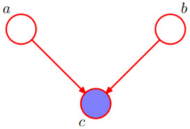
DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

Note: Shaded = node's value known



$$
\begin{aligned}
p(a,b,c) &= p(a)p(b)p(c|a,b) \\
p(a,b) &= p(a)p(b) \\
a \perp\!\!\!\perp b &\mid \emptyset
\end{aligned}
$$

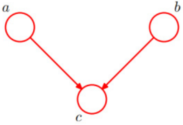


$$
\begin{aligned}
p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\
&= \frac{p(a)p(b)p(c|a,b)}{p(c)} \\
a \not\perp\!\!\!\perp b &\mid c
\end{aligned}
$$

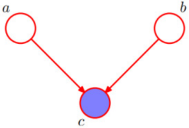
DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

Note: Shaded = node's value known



$$\begin{aligned} p(a,b,c) &= p(a)p(b)p(c|a,b) \\ p(a,b) &= p(a)p(b) \\ a \perp\!\!\!\perp b &\mid \emptyset \end{aligned}$$



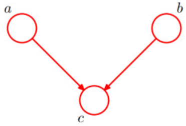
$$\begin{aligned} p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\ &= \frac{p(a)p(b)p(c|a,b)}{p(c)} \\ a \not\perp\!\!\!\perp b &\mid c \end{aligned}$$

Opposite behavior as compared to the previous two cases! Conditioning makes a and b dependent.

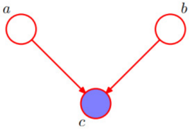
DGM and Conditional Independence

Would like to test whether two nodes a and b are independent in the presence of a third node c

Note: Shaded = node's value known



$$\begin{aligned} p(a,b,c) &= p(a)p(b)p(c|a,b) \\ p(a,b) &= p(a)p(b) \\ a \perp\!\!\!\perp b &\mid \emptyset \end{aligned}$$

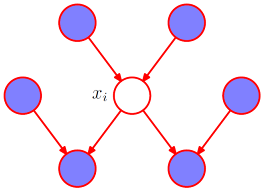


$$\begin{aligned} p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\ &= \frac{p(a)p(b)p(c|a,b)}{p(c)} \\ a \not\!\perp\!\!\!\perp b &\mid c \end{aligned}$$

Opposite behavior as compared to the previous two cases! Conditioning makes a and b dependent.

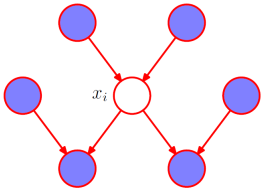
D-Separation (Pearl, 1988): A more general method for checking conditional independence in DGM (See Bishop, Chap 8)

DGM and Markov Blanket



- Markov Blanket of a node in DGM consists of
    - Its parents
    - Its children
    - Its co-parents (other parents of its children)

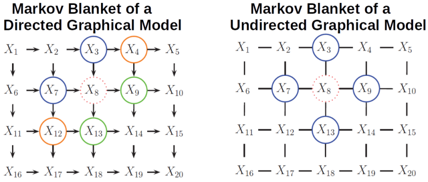
DGM and Markov Blanket



- Markov Blanket of a node in DGM consists of
  - Its parents
  - Its children
  - Its co-parents (other parents of its children)
- Basically, the minimum set of nodes that separate the node from rest of the graph

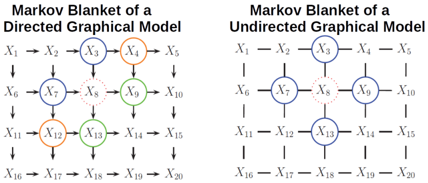
Undirected Graphical Models (UGM)

- The "causal" dependencies of DGM are sometimes unclear/unintuitive
- Consider the "grid" of pixels in an image



Undirected Graphical Models (UGM)

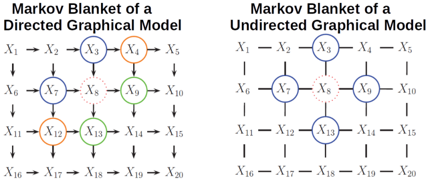
- The "causal" dependencies of DGM are sometimes unclear/unintuitive
- Consider the "grid" of pixels in an image



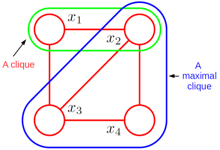
- The (in)dependence structure (Markov blanket) implied by a UGM is more natural here

Undirected Graphical Models (UGM)

- The "causal" dependencies of DGM are sometimes unclear/unintuitive
- Consider the "grid" of pixels in an image

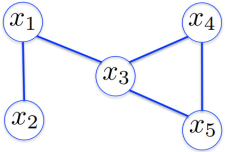


- The (in)dependence structure (Markov blanket) implied by a UGM is more natural here
- UGMs are defined in terms of "cliques" (groups of connected nodes)



Undirected Graphical Models (UGM) a.k.a. Markov Random Field

- Represent joint distributions as product of non-negative *potentials* $\psi()$ defined over cliques

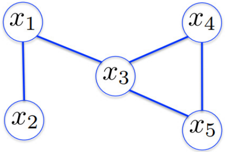


$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$$

where $Z = \sum_{x_1, x_2, x_3, x_4, x_5} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$ is a normalizer

Undirected Graphical Models (UGM) a.k.a. Markov Random Field

- Represent joint distributions as product of non-negative *potentials* $\psi()$ defined over cliques



$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$$

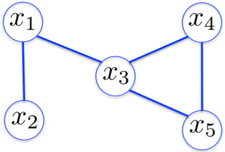
where $Z = \sum_{x_1,x_2,x_3,x_4,x_5} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$ is a normalizer

- Can also write each potential using an energy function

$$\psi_c(\boldsymbol{x}_c) = e^{-E(\boldsymbol{x}_c|\theta_c)}$$

Undirected Graphical Models (UGM) a.k.a. Markov Random Field

- Represent joint distributions as product of non-negative *potentials* $\psi()$ defined over cliques



$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$$

where $Z = \sum_{x_1,x_2,x_3,x_4,x_5} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$ is a normalizer

- Can also write each potential using an energy function

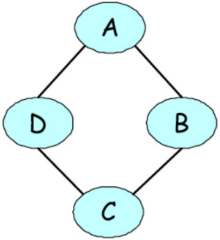
$$\psi_c(\boldsymbol{x}_c) = e^{-E(\boldsymbol{x}_c|\theta_c)}$$

- The joint distribution of a UGM can be then written as

$$p(\boldsymbol{x}) = \frac{1}{Z}\prod_{c\in C}\psi_c(\boldsymbol{x}_c) = \frac{1}{Z}\prod_{c\in C} e^{-E(\boldsymbol{x}_c|\theta_c)} = \frac{1}{Z}e^{-\sum_{c\in C} E(\boldsymbol{x}_c|\theta_c)}$$

Undirected Graphical Models: An Example

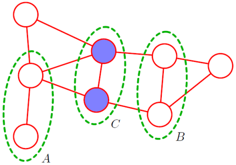
Consider a 4 node UGM with 4 cliques. Each node takes one of 2 possible values



$\psi(A,B)$			$\psi(B,C)$			$\psi(C,D)$			$\psi(D,A)$		
a^0	b^0	30	b^0	c^0	100	c^0	d^0	1	d^0	a^0	100
a^0	b^1	5	b^0	c^1	1	c^0	d^1	100	d^0	a^1	1
a^1	b^0	1	b^1	c^0	1	c^1	d^0	100	d^1	a^0	1
a^1	b^1	10	b^1	c^1	100	c^1	d^1	1	d^1	a^1	100

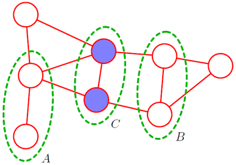
UGM: Tests for Independence

- Usually simpler to check than the DGM case (UGMs have no causal relationships)
- $A \perp\!\!\!\perp B | C$ if all paths from A to B pass through one or more nodes in C



UGM: Tests for Independence

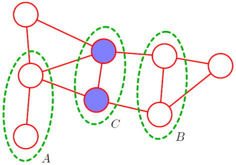
- Usually simpler to check than the DGM case (UGMs have no causal relationships)
- $A \perp\!\!\!\perp B | C$ if all paths from A to B pass through one or more nodes in C



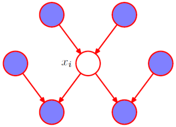
- Another way: See if removing all the nodes (with their edges) in C will "disconnect" A and C

UGM: Tests for Independence

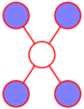
- Usually simpler to check than the DGM case (UGMs have no causal relationships)
- $A \perp\!\!\!\perp B | C$ if all paths from A to B pass through one or more nodes in C



- Another way: See if removing all the nodes (with their edges) in C will "disconnect" A and C
- Also, unlike DGM, Markov blanket of a UGM node only consists only of nodes it is connected to



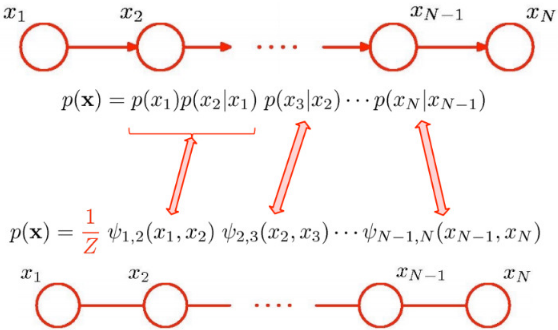
Markov Blanket for a DGM



Markov Blanket for a UGM

Converting DGM to UGM

- A DGM can be converted to an equivalent UGM
- Straightforward for chain-structured DGM



Here Z = 1

Converting DGM to UGM

- A DGM can be converted to an equivalent UGM
- Straightforward for chain-structured DGM

$$p(\mathbf{x}) = \underbrace{p(x_1)p(x_2|x_1)}\ p(x_3|x_2)\cdots p(x_N|x_{N-1})$$

Here Z = 1

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1,x_2)\ \psi_{2,3}(x_2,x_3)\cdots\psi_{N-1,N}(x_{N-1},x_N)$$

- In general, the conditional distributions are mapped to cliques

Converting DGM to UGM

- A DGM can be converted to an equivalent UGM
- Straightforward for chain-structured DGM

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)\,p(x_3|x_2)\cdots p(x_N|x_{N-1})$$

$$p(\mathbf{x}) = \frac{1}{Z}\,\psi_{1,2}(x_1,x_2)\,\psi_{2,3}(x_2,x_3)\cdots\psi_{N-1,N}(x_{N-1},x_N)$$

Here Z = 1

- In general, the conditional distributions are mapped to cliques
  - Need to perform some other operations (e.g., "moralization") to ensure that the conditional independence structures are preserved (refer to Bishop Chap 8 for details)

Inference in Graphical Models

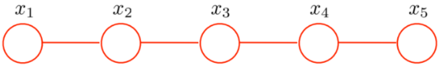
- We may wish to perform inference in a GM. Some nodes may be observed, some unobserved
  - Observed nodes are simply clamped to their values

Inference in Graphical Models

- We may wish to perform inference in a GM. Some nodes may be observed, some unobserved
  - Observed nodes are simply clamped to their values
- Some typical inference tasks: Computing marginals or MAP assignments of nodes

Inference in Graphical Models

- We may wish to perform inference in a GM. Some nodes may be observed, some unobserved
  - Observed nodes are simply clamped to their values
- Some typical inference tasks: Computing marginals or MAP assignments of nodes
- Consider a chain structured GM with 5 nodes (each discrete valued with K possible values)

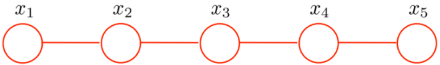


Joint distribution $$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\psi_{3,4}(x_3, x_4)\psi_{4,5}(x_4, x_5)$$

Wish to compute the marginal $$p(x_3) = \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(\mathbf{x})$$

Inference in Graphical Models

- We may wish to perform inference in a GM. Some nodes may be observed, some unobserved
  - Observed nodes are simply clamped to their values
- Some typical inference tasks: Computing marginals or MAP assignments of nodes
- Consider a chain structured GM with 5 nodes (each discrete valued with K possible values)



Joint distribution $$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\psi_{3,4}(x_3, x_4)\psi_{4,5}(x_4, x_5)$$

Wish to compute the marginal $$p(x_3) = \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(\mathbf{x})$$

- Likewise, for an N node chain graph, the problem will be

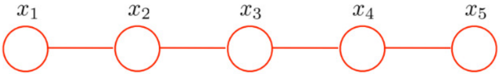
$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N)$$

$$p(x_n) = \sum_{x_1}\cdots\sum_{x_{n-1}}\sum_{x_{n+1}}\cdots\sum_{x_N} p(\mathbf{x})$$

K^N Computations needed

An Efficient Way: Message-Passing

- We can re-arrange the order of computations for efficiency

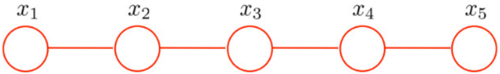


$$p(x_3) = \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(\mathbf{x})$$

$$p(x_3) \quad = \quad \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5)$$

An Efficient Way: Message-Passing

- We can re-arrange the order of computations for efficiency

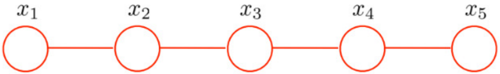


$$p(x_3) = \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(\mathbf{x})$$

$$\begin{aligned}
p(x_3) &= \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5) \\
&= \frac{1}{Z}\sum_{x_2}\sum_{x_1}\sum_{x_4}\sum_{x_5} \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5)
\end{aligned}$$

An Efficient Way: Message-Passing

- We can re-arrange the order of computations for efficiency



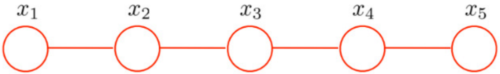
$$p(x_3) = \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(\mathbf{x})$$

$$\begin{aligned}
p(x_3) &= \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5) \\
&= \frac{1}{Z}\sum_{x_2}\sum_{x_1}\sum_{x_4}\sum_{x_5} \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5) \\
&= \frac{1}{Z}\underbrace{\sum_{x_2}\psi_{2,3}(x_2,x_3)\sum_{x_1}\psi_{1,2}(x_1,x_2)}_{\mu_\alpha(x_3)}\underbrace{\sum_{x_4}\psi_{3,4}(x_3,x_4)\sum_{x_5}\psi_{4,5}(x_4,x_5)}_{\mu_\beta(x_3)}
\end{aligned}$$

Vectors of size K

An Efficient Way: Message-Passing

- We can re-arrange the order of computations for efficiency



$$p(x_3) = \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(\mathbf{x})$$

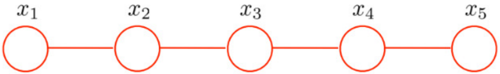
$$\begin{aligned}
p(x_3) &= \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5)\\
&= \frac{1}{Z}\sum_{x_2}\sum_{x_1}\sum_{x_4}\sum_{x_5}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5)\\
&= \frac{1}{Z}\underbrace{\sum_{x_2}\psi_{2,3}(x_2,x_3)\sum_{x_1}\psi_{1,2}(x_1,x_2)}_{\mu_\alpha(x_3)}\underbrace{\sum_{x_4}\psi_{3,4}(x_3,x_4)\sum_{x_5}\psi_{4,5}(x_4,x_5)}_{\mu_\beta(x_3)}
\end{aligned}$$

$\mu_\alpha(x_3) \leftarrow$ Vectors of size K $\rightarrow \mu_\beta(x_3)$

- Inference (computing marginal here) reduces to passing messages (vectors) between nodes!

An Efficient Way: Message-Passing

- We can re-arrange the order of computations for efficiency



$$p(x_3) = \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(\mathbf{x})$$

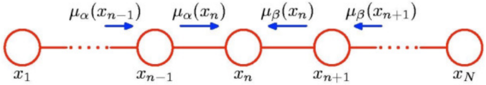
$$\begin{aligned}
p(x_3) &= \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5) \\
&= \frac{1}{Z}\sum_{x_2}\sum_{x_1}\sum_{x_4}\sum_{x_5} \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5) \\
&= \frac{1}{Z}\underbrace{\sum_{x_2}\psi_{2,3}(x_2,x_3)\sum_{x_1}\psi_{1,2}(x_1,x_2)}_{\mu_\alpha(x_3)}\underbrace{\sum_{x_4}\psi_{3,4}(x_3,x_4)\sum_{x_5}\psi_{4,5}(x_4,x_5)}_{\mu_\beta(x_3)}
\end{aligned}$$

$\mu_\alpha(x_3) \leftarrow$ Vectors of size K $\rightarrow \mu_\beta(x_3)$

- Inference (computing marginal here) reduces to passing messages (vectors) between nodes!
  - To compute $p(x_3)$, we multiply the incoming messages to this node and normalize

An Efficient Way: Message-Passing

- For a general chain of arbitrary length, we can do it as



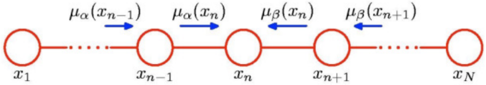
$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right]\right] \cdots \right]}_{\mu_\alpha(x_n)}$$

Cost = O(NK²)

$$\times \underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right] \cdots \right]}_{\mu_\beta(x_n)}$$

An Efficient Way: Message-Passing

- For a general chain of arbitrary length, we can do it as



$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right]\right] \cdots \right]}_{\mu_\alpha(x_n)}$$

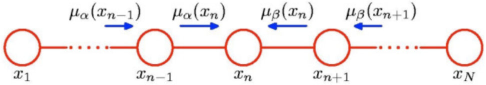
Cost = O(NK²)

$$\times \underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right] \cdots \right]}_{\mu_\beta(x_n)}$$

- Inference (computing marginal here) reduces to passing messages (vectors) between nodes!

An Efficient Way: Message-Passing

- For a general chain of arbitrary length, we can do it as



$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \right] \cdots \right]}_{\mu_\alpha(x_n)}$$

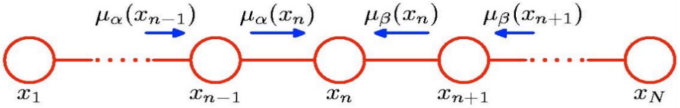
Cost = O(NK²)

$$\times \underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]}_{\mu_\beta(x_n)}$$

- Inference (computing marginal here) reduces to passing messages (vectors) between nodes!
  - To compute $p(x_n)$, we multiply the incoming messages $\mu_\alpha(x_n)$ and $\mu_\beta(x_n)$ and normalize

Recursively Computing Messages

- The forward and backward messages can be computed recursively



Forward message:

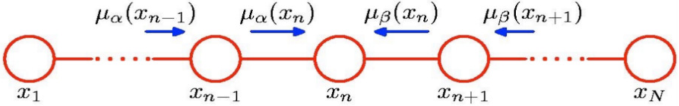
$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right]$$
$$= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \boxed{\mu_\alpha(x_{n-1})}$$

Backward message:

$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \left[\sum_{x_{n+2}} \cdots \right]$$
$$= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \boxed{\mu_\beta(x_{n+1})}$$

Recursively Computing Messages

- The forward and backward messages can be computed recursively



Forward message:
$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right] = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \boxed{\mu_\alpha(x_{n-1})}$$

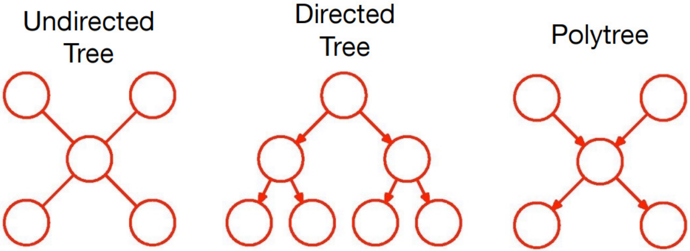
Backward message:
$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \left[\sum_{x_{n+2}} \cdots \right] = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \boxed{\mu_\beta(x_{n+1})}$$

- Start computing μ_α from first node, μ_β from last node

$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2) \qquad \mu_\beta(x_{N-1}) = \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)$$

Message-Passing for Other Tree-Structured Graphs

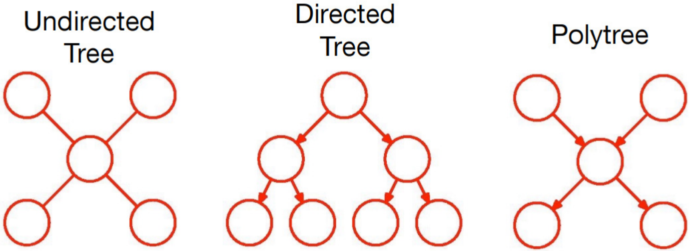
- The message-passing for chain-structured graphs can be generalized to other graphs



- This is popularly known as the sum-product algorithm (recall that the algorithm was based on computing a series of sums and products to compute each marginal)

Message-Passing for Other Tree-Structured Graphs

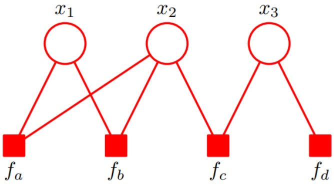
- The message-passing for chain-structured graphs can be generalized to other graphs



- This is popularly known as the sum-product algorithm (recall that the algorithm was based on computing a series of sums and products to compute each marginal)
- The same algo works for both directed and undirected graphs (by converting both into a "factor graph" representation, and doing message-passing on this factor graph)

Factor Graphs

- A unified representation for general DGM/UGM
- Useful for designing message-passing algos for general DGM/UGM
- A bipartite graph consisting of variable nodes and factor nodes



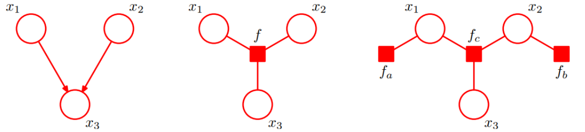
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

- Basic idea: Original nodes become the variable nodes, conditionals/potentials becomes factor nodes (which represent a computation over the variable nodes connected to the factor node)

Factor Graphs

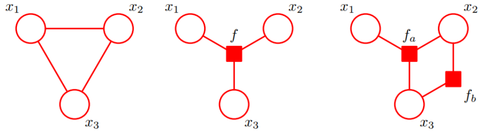
- Both DGM and UGM can be converted into a factor graph representation



$p(x) = p(x_1)p(x_2)p(x_3|x_1,x_2)$

$f(x_1,x_2,x_3) = p(x_1)p(x_2)p(x_3|x_1,x_2)$

$f_a(x_1) = p(x_1)$
$f_b(x_2) = p(x_2)$
$f_c(x_1,x_2,x_3) = p(x_3|x_2,x_1)$



$\psi(x_1,x_2,x_3)$

$f(x_1,x_2,x_3) = \psi(x_1,x_2,x_3)$

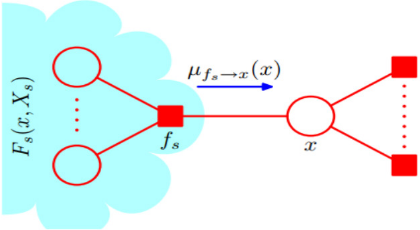
$f_a(x_1,x_2,x_3)f_b(x_2,x_3) = \psi(x_1,x_2,x_3)$

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing
- Example: Computing marginal $p(x) = \sum_{x \setminus \boldsymbol{x}} p(\boldsymbol{x})$ of some node given the factor graph of $p(\boldsymbol{x})$



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

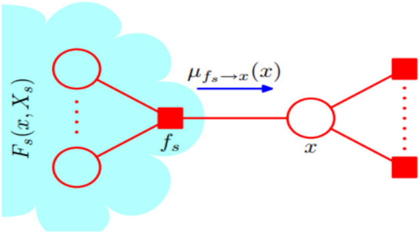
$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

$$p(x) = \prod_{s \in Ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in Ne(x)} \mu_{f_s \to x}(x)$$

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing
- Example: Computing marginal $p(x) = \sum_{x \setminus \boldsymbol{x}} p(\boldsymbol{x})$ of some node given the factor graph of $p(\boldsymbol{x})$



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

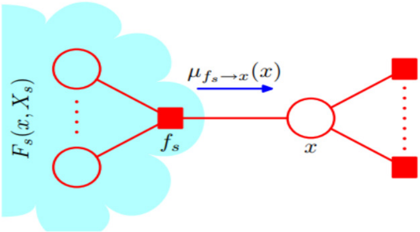
$$p(x) = \prod_{s \in Ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in Ne(x)} \mu_{f_s \to x}(x)$$

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

- This is the sum-product algorithm for marginals with each message $\mu_{f_s \to x}$ recursively defined

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing
- Example: Computing marginal $p(x) = \sum_{\boldsymbol{x} \setminus x} p(\boldsymbol{x})$ of some node given the factor graph of $p(\boldsymbol{x})$



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

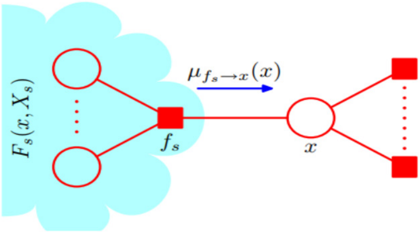
$$p(x) = \prod_{s \in Ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in Ne(x)} \mu_{f_s \to x}(x)$$

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

- This is the sum-product algorithm for marginals with each message $\mu_{f_s \to x}$ recursively defined
- For tree-structured graphs, converges in a finite many steps and gives the exact marginals

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing
- Example: Computing marginal $p(x) = \sum_{\boldsymbol{x} \setminus x} p(\boldsymbol{x})$ of some node given the factor graph of $p(\boldsymbol{x})$



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

$$p(x) = \prod_{s \in Ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in Ne(x)} \mu_{f_s \to x}(x)$$

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

- This is the sum-product algorithm for marginals with each message $\mu_{f_s \to x}$ recursively defined
- For tree-structured graphs, converges in a finite many steps and gives the exact marginals
- PRML Chapter 8 contains more details of such algorithms

Some Comments

- Sum-product computes marginals. Similar message-passing exist for computing MAP assignment

$$\hat{\boldsymbol{x}} = \arg \max_{x_1, x_2, \ldots, x_N} p(\boldsymbol{x})$$

- Max-sum and max-product are message passing algos that do this

Some Comments

- Sum-product computes marginals. Similar message-passing exist for computing MAP assignment

$$\hat{\boldsymbol{x}} = \arg \max_{x_1, x_2, \ldots, x_N} p(\boldsymbol{x})$$

- Max-sum and max-product are message passing algos that do this
- We described the sum-product assuming all nodes are unobserved.
  - In practice, some nodes are observed, and they are "clamped" to their known values

Some Comments

- Sum-product computes marginals. Similar message-passing exist for computing MAP assignment

$$\hat{\boldsymbol{x}} = \arg \max_{x_1, x_2, \ldots, x_N} p(\boldsymbol{x})$$

- Max-sum and max-product are message passing algos that do this
- We described the sum-product assuming all nodes are unobserved.
  - In practice, some nodes are observed, and they are "clamped" to their known values
- For tree-structured graphs, these message passing algos give exact answer

Some Comments

- Sum-product computes marginals. Similar message-passing exist for computing MAP assignment

$$\hat{\boldsymbol{x}} = \arg \max_{x_1, x_2, \ldots, x_N} p(\boldsymbol{x})$$

- Max-sum and max-product are message passing algos that do this
- We described the sum-product assuming all nodes are unobserved.
    - In practice, some nodes are observed, and they are "clamped" to their known values
- For tree-structured graphs, these message passing algos give exact answer
- For loopy graphs, they still work (but not guaranteed to give exact answer)

Some Comments

- Sum-product computes marginals. Similar message-passing exist for computing MAP assignment

$$\hat{\boldsymbol{x}} = \arg\max_{x_1, x_2, \ldots, x_N} p(\boldsymbol{x})$$

- Max-sum and max-product are message passing algos that do this
- We described the sum-product assuming all nodes are unobserved.
  - In practice, some nodes are observed, and they are "clamped" to their known values
- For tree-structured graphs, these message passing algos give exact answer
- For loopy graphs, they still work (but not guaranteed to give exact answer)
- Some variants of these algorithms are known as "belief propagation"

Summary

- Probabilistic graphical models (directed/undirected) allow specifying model structure compactly
- Can often "read off" conditional independence structures by inspecting the graph

Summary

- Probabilistic graphical models (directed/undirected) allow specifying model structure compactly
- Can often "read off" conditional independence structures by inspecting the graph
- Both directed and undirected graphical models have different properties that they can encode/express. Both important in their own contexts depending on the problem (though DGMs are more commonly used)

Summary

- Probabilistic graphical models (directed/undirected) allow specifying model structure compactly
- Can often "read off" conditional independence structures by inspecting the graph
- Both directed and undirected graphical models have different properties that they can encode/express. Both important in their own contexts depending on the problem (though DGMs are more commonly used)
- Factor graphs provide a way to represent both models in a similar way and apply same algorithms (message-passing such as sum-product) for doing inference

Summary

- Probabilistic graphical models (directed/undirected) allow specifying model structure compactly
- Can often "read off" conditional independence structures by inspecting the graph
- Both directed and undirected graphical models have different properties that they can encode/express. Both important in their own contexts depending on the problem (though DGMs are more commonly used)
- Factor graphs provide a way to represent both models in a similar way and apply same algorithms (message-passing such as sum-product) for doing inference
- Exact inference possible for tree-structured models using message passing algos on factor graphs

Summary

- Probabilistic graphical models (directed/undirected) allow specifying model structure compactly
- Can often "read off" conditional independence structures by inspecting the graph
- Both directed and undirected graphical models have different properties that they can encode/express. Both important in their own contexts depending on the problem (though DGMs are more commonly used)
- Factor graphs provide a way to represent both models in a similar way and apply same algorithms (message-passing such as sum-product) for doing inference
- Exact inference possible for tree-structured models using message passing algos on factor graphs
- We assumed that the graphical model structure is known. This itself may need to be learned (a lot of work on graphical model structure learning)