Probabilistic Graphical Models, Inference via Message-Passing

Piyush Rai

Probabilistic Machine Learning (CS772A)

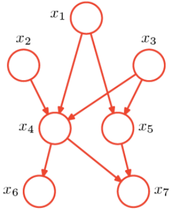
Nov 9, 2017

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

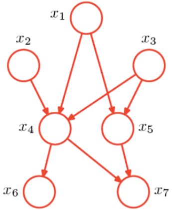
Directed Graphical Models (DGM)

- 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

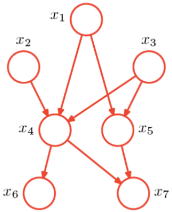


- Vertices denotes r.v. and structure of the graph tells us the conditional dependencies

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

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



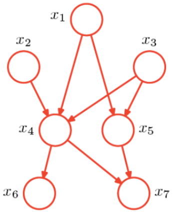
- Vertices denotes r.v. and structure of the graph tells us the conditional dependencies

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

- 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 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

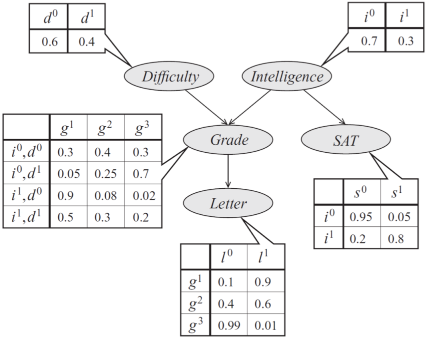
$$p(\boldsymbol{x}) = \prod_{k=1}^{K} p(x_k|\text{pa}_k)$$

Directed Graphical Models: An Example

- An Example: Consider a model of student grades

$$p(\ell, g, i, d, s) = p(\ell \mid g)p(g \mid i, d)p(i)p(d)p(s \mid i)$$

- The Bayes net representation of this model



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^1, d^0	0.9	0.08	0.02
i^1, d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

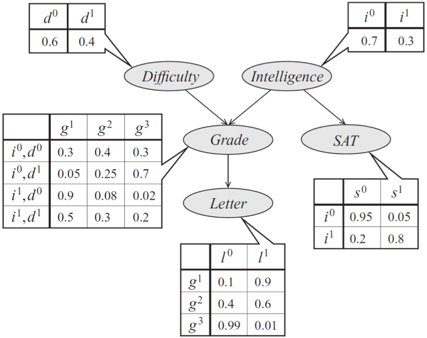
	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Directed Graphical Models: An Example

- An Example: Consider a model of student grades

$$p(\ell, g, i, d, s) = p(\ell \mid g)p(g \mid i, d)p(i)p(d)p(s \mid i)$$

- The Bayes net representation of this model



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

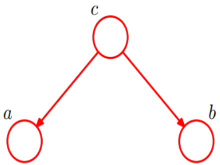
	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

- The conditional independence structure leads to a substantial reduction in the number of parameters to represent $p(\ell, g, i, d, s)$. Naïve way would require $2 \times 3 \times 2 \times 2 \times 2 = 48$ params

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



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

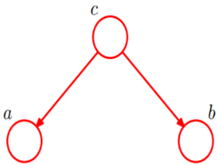
$$p(a, b) = \sum_c p(a|c)p(b|c)p(c)$$

$$a \not\perp b \mid \emptyset$$

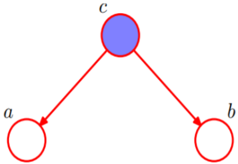
DGM and Conditional Independence

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



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



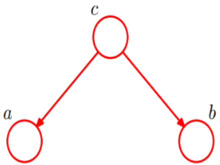
$$
\begin{aligned}
p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\
&= p(a|c)p(b|c) \\
a \perp\!\!\!\perp b &\mid c
\end{aligned}
$$

Thus conditioning on c makes a and b conditionally independent

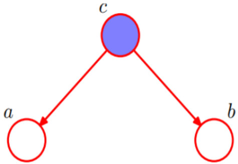
DGM and Conditional Independence

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



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



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

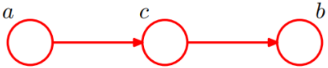
Thus conditioning on c makes a and b conditionally independent

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



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

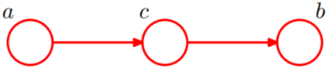
$$p(a, b) = p(a)\sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

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

DGM and Conditional Independence

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

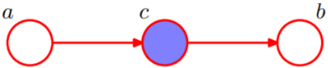
Note: Shaded = node's value known



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

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

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



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

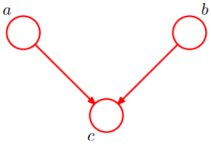
$$a \perp\!\!\!\perp b | c$$

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

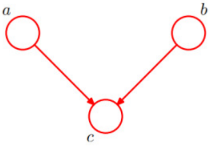


$$\begin{aligned} p(a,b,c) &= p(a)p(b)p(c|a,b) \\ p(a,b) &= p(a)p(b) \\ a \perp\!\!\!\perp b &\mid \emptyset \end{aligned}$$

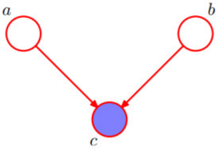
DGM and Conditional Independence

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



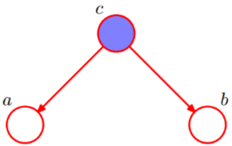
$$
\begin{aligned}
p(a,b,c) &= p(a)p(b)p(c|a,b) \\
p(a,b) &= p(a)p(b) \\
a \perp\!\!\!\perp b &\mid \emptyset
\end{aligned}
$$



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

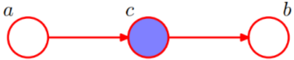
Opposite behavior as compared to the previous two cases! Conditioning makes a and b dependent.

DGM and Conditional Independence: Summary



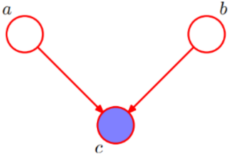
$$a \perp\!\!\!\perp b \mid c$$

Tail-to-Tail + Observed: Blocks Path



$$a \perp\!\!\!\perp b \mid c$$

Head-to-Tail + Observed: Blocks Path



$$a \not\perp\!\!\!\perp b \mid c$$

Head-to-Head + Observed: Unblocks Path

D-Separation (Pearl, 1988)

- How to test for conditional independence in a general DGM ?

D-Separation (Pearl, 1988)

- How to test for conditional independence in a general DGM ?
- Can make use of the three basic primitives we just saw

D-Separation (Pearl, 1988)

- How to test for conditional independence in a general DGM ?
- Can make use of the three basic primitives we just saw
- Assume A, B, C to be arbitrary sets of non-intersecting nodes in a DGM

D-Separation (Pearl, 1988)

- How to test for conditional independence in a general DGM ?
- Can make use of the three basic primitives we just saw
- Assume A, B, C to be arbitrary sets of non-intersecting nodes in a DGM
- We would like to test about a particular conditional independence, e.g., whether $A \perp\!\!\!\perp B|C$

D-Separation (Pearl, 1988)

- How to test for conditional independence in a general DGM ?
- Can make use of the three basic primitives we just saw
- Assume A, B, C to be arbitrary sets of non-intersecting nodes in a DGM
- We would like to test about a particular conditional independence, e.g., whether $A \perp\!\!\!\perp B|C$
- To test $A \perp\!\!\!\perp B|C$, consider all paths from A to B

D-Separation (Pearl, 1988)

- How to test for conditional independence in a general DGM ?
- Can make use of the three basic primitives we just saw
- Assume A, B, C to be arbitrary sets of non-intersecting nodes in a DGM
- We would like to test about a particular conditional independence, e.g., whether $A \perp\!\!\!\perp B | C$
- To test $A \perp\!\!\!\perp B | C$, consider all paths from A to B
- Define a path from A to B as blocked if the path includes a node such that either

D-Separation (Pearl, 1988)

- How to test for conditional independence in a general DGM ?
- Can make use of the three basic primitives we just saw
- Assume A, B, C to be arbitrary sets of non-intersecting nodes in a DGM
- We would like to test about a particular conditional independence, e.g., whether $A \perp\!\!\!\perp B | C$
- To test $A \perp\!\!\!\perp B | C$, consider all paths from A to B
- Define a path from A to B as blocked if the path includes a node such that either
    - Arrows in the path meet head-to-tail or tail-to-tail at the node AND the node is in C

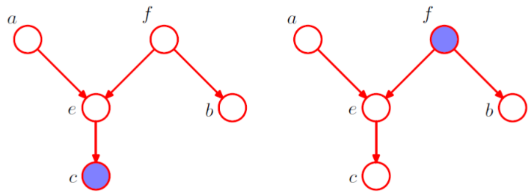
D-Separation (Pearl, 1988)

- How to test for conditional independence in a general DGM ?
- Can make use of the three basic primitives we just saw
- Assume A, B, C to be arbitrary sets of non-intersecting nodes in a DGM
- We would like to test about a particular conditional independence, e.g., whether $A \perp\!\!\!\perp B|C$
- To test $A \perp\!\!\!\perp B|C$, consider all paths from A to B
- Define a path from A to B as blocked if the path includes a node such that either
  - Arrows in the path meet head-to-tail or tail-to-tail at the node AND the node is in C, or
  - Arrows meet head-to-head at the node, and neither the node, nor any of its descendants, are in C

D-Separation (Pearl, 1988)

- How to test for conditional independence in a general DGM ?
- Can make use of the three basic primitives we just saw
- Assume A, B, C to be arbitrary sets of non-intersecting nodes in a DGM
- We would like to test about a particular conditional independence, e.g., whether $A \perp\!\!\!\perp B|C$
- To test $A \perp\!\!\!\perp B|C$, consider all paths from A to B
- Define a path from A to B as blocked if the path includes a node such that either
  - Arrows in the path meet head-to-tail or tail-to-tail at the node AND the node is in C, or
  - Arrows meet head-to-head at the node, and neither the node, nor any of its descendants, are in C
- If all paths are blocked then we say that A is D-separated from B by C

D-Separation: Example



a to b is unblocked by e

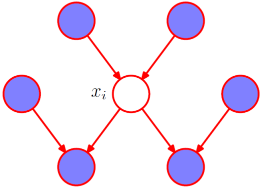
Thus $a \not\perp\!\!\!\perp b \mid e$

a to b blocked by f

Thus $a \perp\!\!\!\perp b \mid f$

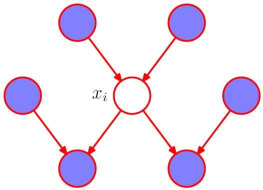
- In the left figure, a to b path is NOT blocked by e and f
  - e is head-to-head but its descendant is in the conditioning set (fails to fulfil D-separation)
  - f is tail-to-tail but is not in the conditioning set (unshaded) (fails to fulfil D-separation)
- In the right figure, a to b path is blocked by f (and also by e)
  - f is tail-to-tail and is in the conditioning set (fulfils D-separation)
  - e is head-to-head and it or any of its any descendants are NOT in conditioning set (fulfils D-sep)

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)

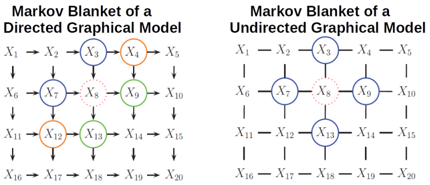
DGM and Markov Blanket



- 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

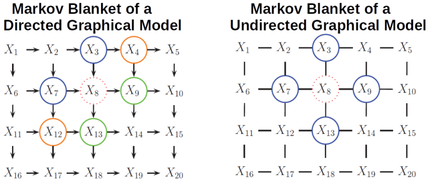
Undirected Graphical Models (UGM)

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



Undirected Graphical Models (UGM)

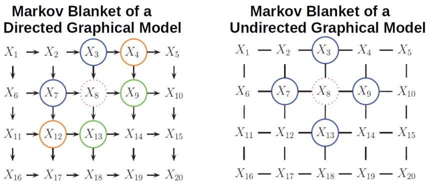
- The "causal" dependencies of DGM are sometimes unclear/unintuitive
- Consider the "grid" of pixels in an image



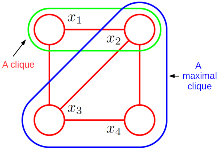
- The (in)dependence structure (Markov blanket) implied by a UGM is more natural here

Undirected Graphical Models (UGM)

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

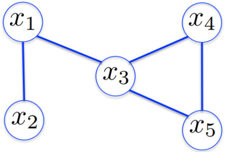


- The (in)dependence structure (Markov blanket) implied by a UGM is more natural here
- UGMs are defined in terms of "cliques" (groups of connected nodes)



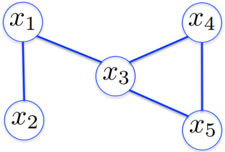
Undirected Graphical Models

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



Undirected Graphical Models

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



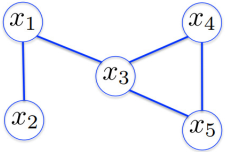
- The joint distribution can be written as

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

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

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



- The joint distribution can be written as

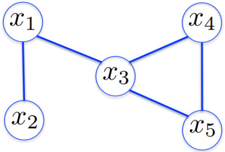
$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$$

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

- Note: Local potentials are not probability distributions (which is why the normalizer is needed)
  - Each local potential is a measure of "compatibility" of the nodes involved

Undirected Graphical Models

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



- The joint distribution can be written as

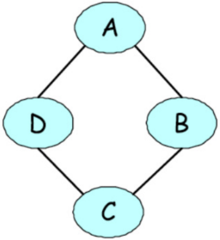
$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$$

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

- Note: Local potentials are not probability distributions (which is why the normalizer is needed)
    - Each local potential is a measure of "compatibility" of the nodes involved
- UGMs are also known as Markov Random Fields (applications in Vision, NLP, etc)

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

Undirected Graphical Models in terms of "Energy

- Since potentials are non-negative, we can define them using a more general "energy function"

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

.. where θ_c denotes the params defining the corresponding real-valued energy function

Undirected Graphical Models in terms of "Energy

- Since potentials are non-negative, we can define them using a more general "energy function"

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

.. where θ_c denotes the params defining the corresponding real-valued energy function

- 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 in terms of "Energy

- Since potentials are non-negative, we can define them using a more general "energy function"

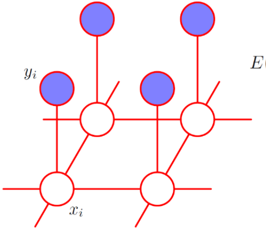
$$\psi_c(\mathbf{x}_c) = e^{-E(\mathbf{x}_c|\theta_c)}$$

.. where θ_c denotes the params defining the corresponding real-valued energy function

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

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

- Such models are often called "energy-based models"



$$E(\mathbf{x},\mathbf{y}) = h\sum_i x_i - \beta\sum_{\{i,j\}} x_i x_j - \eta\sum_i x_i y_i$$

$$p(\mathbf{x},\mathbf{y}) = \frac{1}{Z}\exp\{-E(\mathbf{x},\mathbf{y})\}$$

Converting DGM to UGM

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

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

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

Here Z = 1

Converting DGM to UGM

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

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

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

Here Z = 1

- In general, the conditional distributions are mapped to cliques

Converting DGM to UGM

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

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

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
  - 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

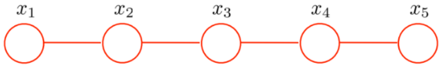
- We may wish to perform inference in a GM. Some nodes may be observed, some unobserved
  - Observed nodes are simply clamped to their values

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)

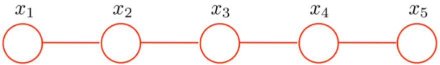


Joint distribution $$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\psi_{3,4}(x_3, x_4)\psi_{4,5}(x_4, x_5)$$

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

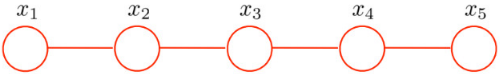
$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N)$$

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

K^N Computations needed

An Efficient Way: Message-Passing

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

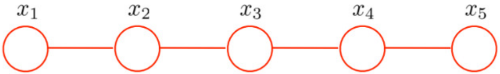


$$p(x_3) = \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(\mathbf{x})$$

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

An Efficient Way: Message-Passing

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

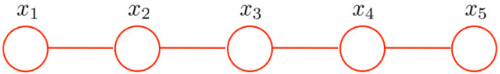


$$p(x_3) = \sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(\mathbf{x})$$

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

An Efficient Way: Message-Passing

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



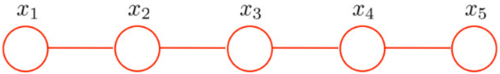
$$p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} p(\mathbf{x})$$

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

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

An Efficient Way: Message-Passing

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



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

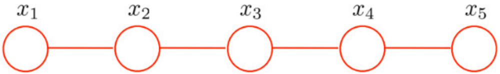
$$\begin{aligned}
p(x_3) &= \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5)\\
&= \frac{1}{Z}\sum_{x_2}\sum_{x_1}\sum_{x_4}\sum_{x_5}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5)\\
&= \frac{1}{Z}\underbrace{\sum_{x_2}\psi_{2,3}(x_2,x_3)\sum_{x_1}\psi_{1,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

Vectors of size K

- 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



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

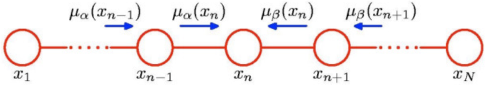
$$\begin{aligned}
p(x_3) &= \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5)\\
&= \frac{1}{Z}\sum_{x_2}\sum_{x_1}\sum_{x_4}\sum_{x_5}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\psi_{3,4}(x_3,x_4)\psi_{4,5}(x_4,x_5)\\
&= \frac{1}{Z}\underbrace{\sum_{x_2}\psi_{2,3}(x_2,x_3)\sum_{x_1}\psi_{1,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

$\mu_\alpha(x_3) \leftarrow$ 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



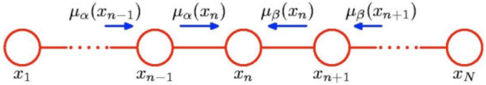
$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right]\right] \cdots \right]}_{\mu_\alpha(x_n)}$$

Cost = O(NK^2)

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

An Efficient Way: Message-Passing

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



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

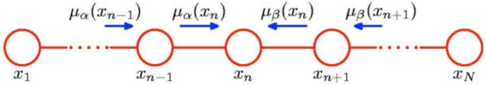
Cost = $O(NK^2)$

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

- 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



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

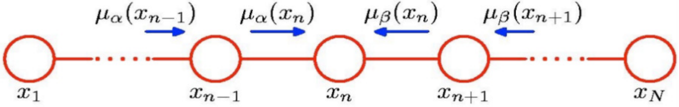
Cost = $O(NK^2)$

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

- 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

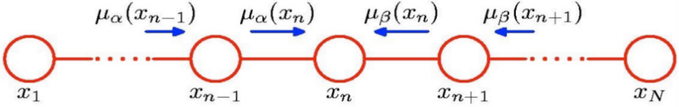


Forward message:
$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right]$$
$$= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \boxed{\mu_\alpha(x_{n-1})}$$

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

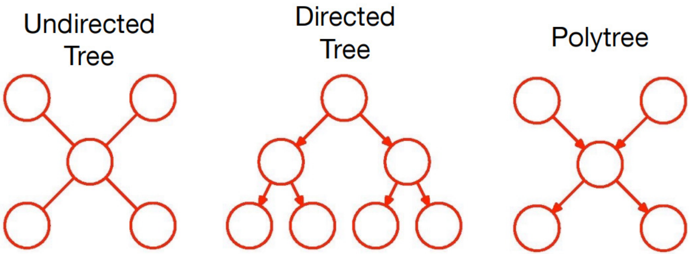
Backward message:
$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \left[\sum_{x_{n+2}} \cdots \right]$$
$$= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \boxed{\mu_\beta(x_{n+1})}$$

- 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

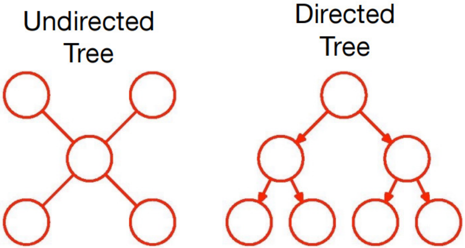
- The message-passing for chain-structured graphs can be generalized to other graphs



- 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

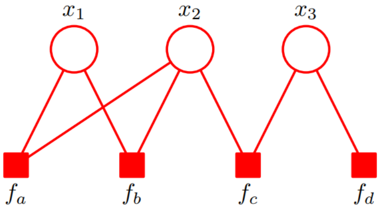
- The message-passing for chain-structured graphs can be generalized to other graphs



- 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 Graph

- Basically a bipartite graph with two sets of nodes: Variable nodes and factor nodes



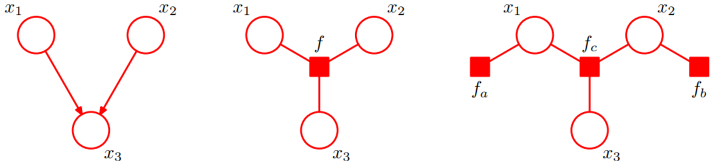
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

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

- Each factor node represent a computation on the variable nodes connected to it
- A DGM or UGM can be converted into a factor graph

Directed Model to Factor Graph

- For each node in the DGM, create a variable node
- For each conditional distribution of DGM, create a factor node



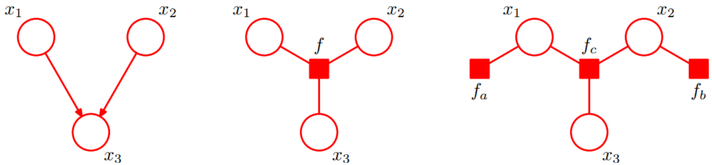
$$p(x) = p(x_1)p(x_2)p(x_3|x_1, x_2)$$

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

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

Directed Model to Factor Graph

- For each node in the DGM, create a variable node
- For each conditional distribution of DGM, create a factor node



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

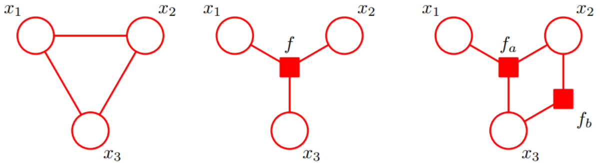
$f(x_1, x_2, x_3) =$
$p(x_1)p(x_2)p(x_3|x_1, x_2)$

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

- Multiple factor graphs possible for a given DGM
    - The product of all factor node functions should be equal to $p(\boldsymbol{x})$

Undirected Model to Factor Graph

- For each node in the DGM, create a variable node
- For each maximal clique, create a factor node



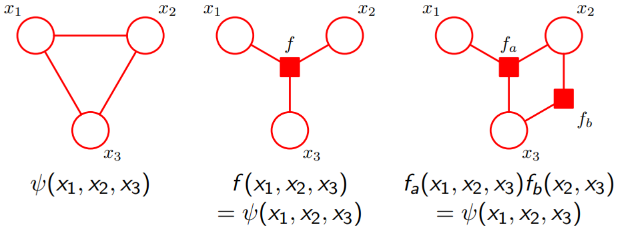
$\psi(x_1, x_2, x_3)$

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

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

Undirected Model to Factor Graph

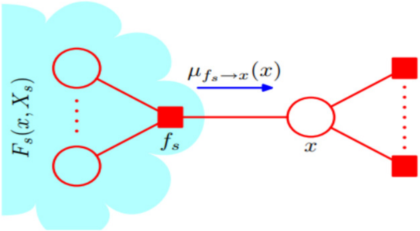
- For each node in the DGM, create a variable node
- For each maximal clique, create a factor node



- Multiple factor graphs possible for a each maximal clique in the UGM
  - The product of all factor node functions should be equal to the clique potential

Sum-Product on Factor Graph: High-Level Idea

- Based on passing messages between factor nodes and variable nodes



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

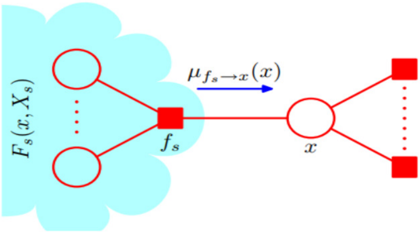
$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

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

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

Sum-Product on Factor Graph: High-Level Idea

- Based on passing messages between factor nodes and variable nodes

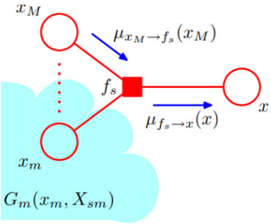


$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

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

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

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



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

$$F_s(x, X_s) = f_s(x, x_1, ..., x_M) G_1(x_1, X_{s1}) \ldots G_M(x_M, X_{sM})$$

- Chapter 8 of PRML contains more details if you are interested

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)