Advanced Probabilistic Machine Learning
SSY316

Graphical Models (3)

Alexandre Graell i Amat
alexandre.graell@chalmers.se
https://sites.google.com/site/agraellamat

November 24, 2023

CHALMERS

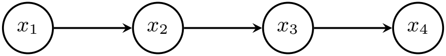
Inference in graphical models

Inference in graphical models: Computing posterior probabilities of unobserved variables given observed ones.

- Discriminative probabilistic models: Obtain $p(t|\mathcal{D}, \boldsymbol{x})$ by evaluating $p(\boldsymbol{w}|\mathcal{D})$
- Generative probabilistic models: Obtain $p(t|\boldsymbol{x}, \boldsymbol{\theta})$ from learned model $p(\boldsymbol{x}, t|\boldsymbol{\theta})$
- Bayesian supervised learning: Obtain $p(t|\mathcal{D}, \boldsymbol{x}, \beta)$ by evaluating $p(\boldsymbol{w}|\mathcal{D}, \beta)$

Can exploit graph structure to devise efficient algorithms for inference.

Exact inference on a chain



$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)$$

Marginal $p(x_4)$?

Exact inference on a chain

$$x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow x_4$$

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)$$

Direct approach:
(assume x discrete taking on K values)

$$p(x_4) = \sum_{x_1}\sum_{x_2}\sum_{x_3} p(x_1, x_2, x_3, x_4)$$

Complexity: $\mathcal{O}(K^4)$ (General case $\mathcal{O}(K^N)$)

Exact inference on a chain

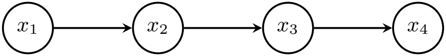
$$x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4$$

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)$$

Can exploit factorization!

$$\begin{aligned}
p(x_4) &= \sum_{x_1}\sum_{x_2}\sum_{x_3} p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3) \\
&= \sum_{x_2}\sum_{x_3}\Bigg[\underbrace{\sum_{x_1} p(x_1)p(x_2|x_1)}_{\mu_2(x_2)}\Bigg] p(x_3|x_2)p(x_4|x_3) \\
&= \sum_{x_3}\Bigg[\underbrace{\sum_{x_2} \mu_2(x_2)p(x_3|x_2)}_{\mu_3(x_3)}\Bigg] p(x_4|x_3) \\
&= \underbrace{\sum_{x_3} \mu_3(x_3)p(x_4|x_3)}_{\mu_4(x_4)}
\end{aligned}$$

Exact inference on a chain



$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)$$

Can exploit factorization!

$$\begin{aligned} p(x_4) &= \sum_{x_1}\sum_{x_2}\sum_{x_3} p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3) \\ &= \sum_{x_3}\left[\sum_{x_2}\left[\sum_{x_1} p(x_1)p(x_2|x_1)\right] p(x_3|x_2)\right] p(x_4|x_3) \end{aligned}$$

Complexity: $\mathcal{O}(3K^2)$ (General case $\mathcal{O}(NK^2)$)

A message passing interpretation

$$x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow x_4$$

$$p(x_4) = \sum_{x_2}\sum_{x_3}\underbrace{\left[\sum_{x_1} p(x_1)p(x_2|x_1)\right]}_{\mu_2(x_2)} p(x_3|x_2)p(x_4|x_3)$$

$$= \sum_{x_3}\underbrace{\left[\sum_{x_2} \mu_2(x_2)p(x_3|x_2)\right]}_{\mu_3(x_3)} p(x_4|x_3) = \underbrace{\sum_{x_3} \mu_3(x_3)p(x_4|x_3)}_{\mu_4(x_4)}$$

$\mu_2(x_2)$: Message from x_2 to x_3

$\mu_3(x_3)$: Message from x_3 to x_4 (obtained multiplying $\mu_2(x_2)$ with local function)

$\mu_4(x_4)$: Message computed by x_4 (obtained multiplying $\mu_3(x_3)$ with local function)

Marginalization can be performed by message passing along the chain!

Message passing and factor graphs

Idea of message passing on a chain can be generalized to more general graphical models.

Directed and undirected graphs: Allow to express global function as product of factors over subsets of variables

Factor graphs: Make decomposition explicit by introducing additional factor nodes.

- We will discuss message passing using the formalism of factor graphs

Factor graphs

- A powerful graphical model to represent explicitly the factorization of a joint distribution as a product of local factors
- Lead naturally to a message passing algorithm (sum-product algorithm, belief propagation) for efficient inference
- Exact inference if the factor graph is a tree (or polytree)

- Introduced by Frey, Kschischang, Loeliger and Wiberg (1997)

Factor graphs

- A collection of variables $\boldsymbol{x} = (x_1, \ldots, x_n)$, $x_i \in \mathcal{A}_i$
- A real-valued function $g(x_1, \ldots, x_n)$,

$$g : \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \to \mathbb{R}$$

- Assume $g(x_1, \ldots, x_n)$ can be factored as

$$g(x_1, \ldots, x_n) = \prod_{j \in \mathcal{J}} f_j(\mathcal{X}_j),$$

$g(\cdot)$: Global function

$f_j(\cdot)$: Local function

$\mathcal{J}$: Discrete index set

$\mathcal{X}_j$: Subset of variables

Factor graphs

Factor graph: A bipartite graph that expresses the structure of the factorization

$$g(x_1, \ldots, x_n) = \prod_{j \in \mathcal{J}} f_j(\mathcal{X}_j)$$

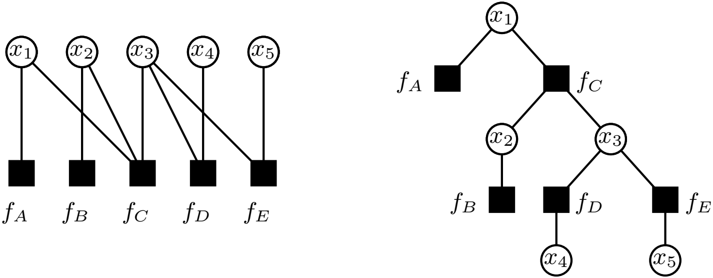
The factor graph has one variable node for each variable x_i, one factor node for each local function f_j, and an undirected edge connecting a variable node x_i with a factor node f_j if and only if x_i is an argument of f_j.

Different types of elements:

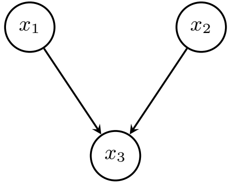
- Variable nodes (circles): represent RVs
- Factor nodes (filled squares): represent factors of the joint distribution
- Undirected edges: Assign variables to factors

Factor graphs: An example

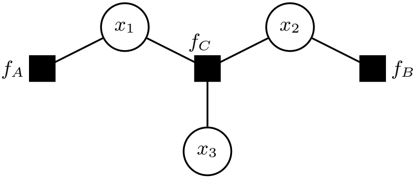
$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5)$$



From directed graphs to factor graphs



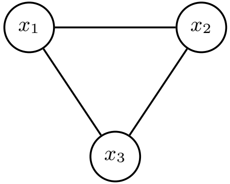
$$p(x_1)p(x_2)p(x_3|x_1, x_2)$$



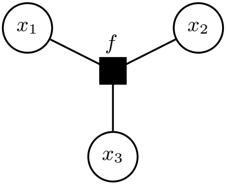
$$f_A = p(x_1)$$
$$f_B = p(x_2)$$
$$f_C = p(x_3|x_1, x_2)$$

1. Add one factor for each node
2. Connect variable nodes to factor nodes according to edges

From undirected graphs to factor graphs



$\psi(x_1, x_2, x_3)$



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

1. Add one factor for each maximal clique
2. Connect variable nodes to clique factors

Factor graphs can represent more general factorizations.

Marginal functions

Factor graphs: Tool for efficient computation of marginal functions when $g(x_1, \ldots, x_n)$ represents a probability distribution and corresponding factor graph is cycle-free (exact marginalization).

- Marginalization computed by message passing over the factor graph

Marginal with respect to x_i,

$$\begin{aligned} g_i(x_i) &= \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_n} g(x_1, \ldots, x_n) \\ &= \sum_{\sim x_i} g(x_1, \ldots, x_n) \end{aligned}$$

Idea: Perform marginalization via message passing over the factor graph by

- Exploiting factorization and the distributive law
- Reusing partial sums

Distributive law

Let $\mathcal{F}$ be a set of elements on which two binary operations, "$+$" and "$\cdot$," are defined. Operation "$\cdot$" is said to be distributive over "$+$" if for all $a, b, c \in \mathcal{F}$,

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

and

$$(b + c) \cdot a = b \cdot a + c \cdot a$$

Marginal functions: An example

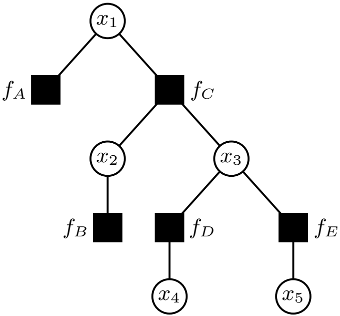
$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5)$$

Marginal with respect to x_1:

$$\begin{aligned}
g_1(x_1) &= \sum_{\sim x_1} g(x_1, \ldots, x_7) \\
&= \sum_{\sim x_1} f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5) \\
&= f_A(x_1) \left(\sum_{x_2} f_B(x_2) \left(\sum_{x_3} f_C(x_1, x_2, x_3) \left(\sum_{x_4} f_D(x_3, x_4) \right) \left(\sum_{x_5} f_E(x_3, x_5) \right) \right) \right) \\
&= f_A(x_1) \left(\sum_{\sim x_1} f_C(x_1, x_2, x_3) f_B(x_2) \left(\sum_{\sim x_3} f_D(x_3, x_4) \right) \left(\sum_{\sim x_3} f_E(x_3, x_5) \right) \right)
\end{aligned}$$

Marginal functions: An example

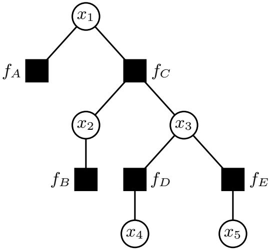
$$g_1(x_1) = f_A(x_1)\left(\sum_{\sim x_1} f_C(x_1,x_2,x_3) f_B(x_2)\left(\sum_{\sim x_3} f_D(x_3,x_4)\right)\left(\sum_{\sim x_3} f_E(x_3,x_5)\right)\right)$$



- Factor graph: unambiguously characterizes marginalization
- Marginalization with respect to x_i: x_i root of tree

- Nodes $\equiv$ processors
- Edges $\equiv$ channels

Message passing

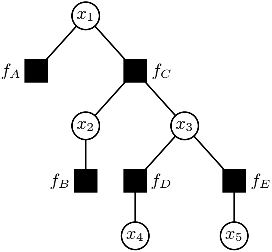


A message passing algorithm to compute marginalization.

- Computation of marginal begins at leaves of tree
- Operations:
  - Variable node: product of incoming messages from children; forwards result to parent
  - Factor node: product of incoming messages and local function $f_j(\mathcal{X}_j)$, applies to it not-sum operator; forwards result to parent
- Marginalization terminates at root node: $g_i(x_i)$ obtained as product of all incoming messages to root node x_i

Message passing

- Operations:
  - Variable node: product of incoming messages from children; forwards result to parent
  - Factor node: product of incoming messages and local function $f_j(\mathcal{X}_j)$, applies to it not-sum operator; forwards result to parent

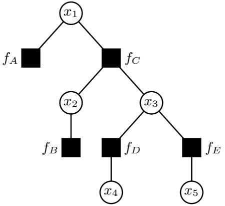


Notation:

- $\mu_{x \to f}$: message from VN x to FN f
- $\mu_{f \to x}$: message from FN f to VN x

Operations:

- Leaf VN: $\mu_{x \to f} = 1$
- Single-child VN: forwards incoming message
- Leaf FN: $\mu_{f_B \to x_2} = \sum_{\sim x_2} f_B(x_2) = f_B(x_2)$
- Single-child FN: $\mu_{f_E \to x_3} = \sum_{\sim x_3} f_E(x_3, x_5)$



$$g_1(x_1) = f_A(x_1) f_B(x_2) f_C(x_1,x_2,x_3) f_D(x_3,x_4) f_E(x_3,x_5)$$

$$= f_A(x_1)\left(\sum_{\sim x_1} f_C(x_1,x_2,x_3) f_B(x_2)\left(\sum_{\sim x_3} f_D(x_3,x_4)\right)\left(\sum_{\sim x_3} f_E(x_3,x_5)\right)\right)$$

Step 1:

$$\mu_{f_A\to x_1} = f_A(x_1), \quad \mu_{f_B\to x_2} = f_B(x_2), \quad \mu_{x_4\to f_D} = 1, \quad \mu_{x_5\to f_E} = 1$$

Step 2:

$$\mu_{x_2\to f_C} = \mu_{f_B\to x_2} = f_B(x_2), \quad \mu_{f_D\to x_3} = \sum_{\sim x_3} f_D(x_3,x_4), \quad \mu_{f_E\to x_3} = \sum_{\sim x_3} f_E(x_3,x_5)$$

Step 3:

$$\mu_{x_3\to f_C} = \mu_{f_D\to x_3}\cdot \mu_{f_E\to x_3} = \left(\sum_{\sim x_3} f_D(x_3,x_4)\right)\left(\sum_{\sim x_3} f_E(x_3,x_5)\right)$$

Step 4:

$$\mu_{f_C\to x_1} = \sum_{\sim x_1} f_C(x_1,x_2,x_3)\mu_{x_2\to f_C}\mu_{x_3\to f_C}$$

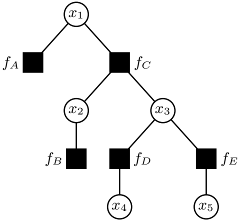
Final step:

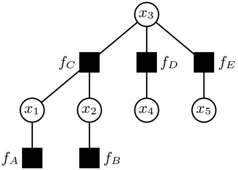
$$g_1(x_1) = f_A(x_1)\left(\sum_{\sim x_1} f_C(x_1,x_2,x_3) f_B(x_2)\left(\sum_{\sim x_3} f_D(x_3,x_4)\right)\left(\sum_{\sim x_3} f_E(x_3,x_5)\right)\right)$$

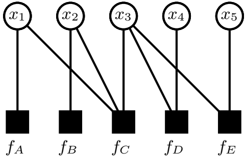
The sum-product algorithm: Computing all marginal functions

How to compute several (or all) marginals $g_i(x_i)$?

- Message passing separately for each marginal $g_i(x_i) \rightarrow$ wasteful!
- Message passing to simultaneously compute the marginals







The sum-product algorithm: Computing all marginal functions

- Algorithm initiates at the leaves of the tree: Leaf VN x sends $\mu_{x\to f} = 1$; leaf FN f sends $\mu_{f\to x} = f(x)$
- A message from a node to one of its neighbors computed once all messages from all other neighbors are received
- Variable node update:

$$\mu_{x\to f} = \prod_{f' \in \mathcal{N}(x)\setminus\{f\}} \mu_{f'\to x}$$

- Factor node update:

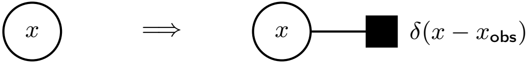
$$\mu_{f\to x} = \sum_{\sim x} \left(f(\mathcal{X}) \prod_{x' \in \mathcal{N}(f)\setminus\{x\}} \mu_{x'\to f} \right)$$

- Algorithm terminates once two messages have been passed over every edge
- $g_i(x_i)$ obtained as the product of all incoming messages to VN x_i

The sum-product algorithm with observed variables

In most applications: Some variables are observed, and want to compute posterior conditioned on observed variables.

- For each observed variable, add a Dirac delta factor:



- Factor graph describes

$$p(\boldsymbol{x}) \prod_{x \in \mathcal{X}} \delta(x - x_{\textsf{obs}}) = p(\boldsymbol{x} \backslash \mathcal{X}, \mathcal{X} = \mathcal{X}_{\textsf{obs}}) \propto p(\boldsymbol{x} \backslash \mathcal{X} | \mathcal{X} = \mathcal{X}_{\textsf{obs}})$$

  $\mathcal{X}$: set of all observed variables

- Posterior marginals $p(x_i | \mathcal{X} = \mathcal{X}_{\textsf{obs}})$ can be computed by sum-product algorithm ($x_i \in \mathcal{A}$)

  $\mathcal{A}$: set of non-observed variables

Bayes' theorem through a message passing lens

Latent variable: x

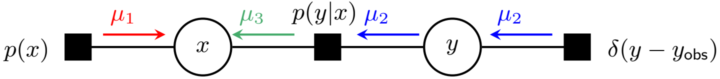
Observed variable: $y = y_{\text{obs}}$

Goal: Infer $p(x|y)$

Bayes' theorem:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \propto p(y|x)p(x)$$

- $p(x, y) = p(y|x)p(x)$



$$\mu_1(x) = p(x) \qquad \mu_2(y) = \delta(y - y_{\text{obs}}) \qquad \mu_3(x) = \int p(y|x)\mu_2(y)\mathrm{d}y = p(y_{\text{obs}}|x)$$

$p(x|y_{\text{obs}}) \propto \mu_1(x)\mu_3(x) = p(x)p(y_{\text{obs}}|x)$ Bayes theorem!

The sum-product algorithm in factor graphs with cycles

Sum-product algorithm: provides exact marginals for cycle-free factor graphs.

What if factor graph has cycles?

- No natural termination of the algorithm $\longrightarrow$ iterative algorithm
- Sum-product algorithm strictly suboptimal, but gives excellent performance in many cases!

The max-product (max-sum) algorithm

Finding a configuration of the variables with largest probability can be performed via the max-sum algorithm.

$$\boldsymbol{x}_{\mathsf{max}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x})$$

Reference

Z. Zhang, F. Wu, W. Sun Lee, “Factor Graph Neural Network,” *NeurIPS*, 2020.

Reading

"Pattern recognition and machine learning,"

Chapter 8 (8.4)