Probabilistic Machine Learning
Lecture 17
Factor Graphs

Philipp Hennig

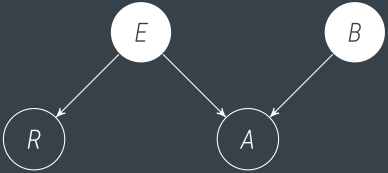
21 June 2021

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning

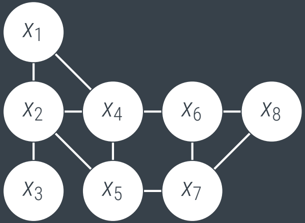
#	date	content	Ex	#	date	content	Ex
1	20.04.	Introduction	1	14	09.06.	Generalized Linear Models	
2	21.04.	Reasoning under Uncertainty		15	15.06.	Exponential Families	8
3	27.04.	Continuous Variables	2	16	16.06.	Graphical Models	
4	28.04.	Monte Carlo		17	22.06.	Factor Graphs	9
5	04.05.	Markov Chain Monte Carlo	3	18	23.06.	The Sum-Product Algorithm	
6	05.05.	Gaussian Distributions		19	29.06.	Example: Topic Models	10
7	11.05.	Parametric Regression	4	20	30.06.	Mixture Models	
8	12.05.	Learning Representations		21	06.07.	EM	11
9	18.05.	Gaussian Processes	5	22	07.07.	Variational Inference	
10	19.05.	Understanding Kernels		23	13.07.	Topics	12
11	26.05.	Gauss-Markov Models		25	14.07.	Example: Inferring Topics	
12	25.05.	An Example for GP Regression	6	24	20.07.	Example: Kernel Topic Models	
13	08.06.	GP Classification	7	26	21.07.	Revision	

Directed Graphical Models / Bayesian Networks



- directly encode a facorization of the joint (it can be read off by parsing the graph from the children to the parents)
- however, reading off conditional independence structure is tricky (it requires considering *d*-separation)
- directed graphs are for encoding **generative knowledge** (think: scientific modelling)

Undirected Graphical Models / Markov Random Fields (MRFs)



- directly encode conditional independence structure (by definition)
- however, reading off the joint from the graph is tricky (it requires finding all maximal cliques, normalization constant is intractable)
- MRFs are for encoding **computational constraints** (think: computer vision)

$$p(\boldsymbol{x}) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_2) \cdots p(x_n \mid x_{n-1})$$

From Directed to Undirected Graphs

Example: Markov Chain

[from Bishop, PRML, 2006]

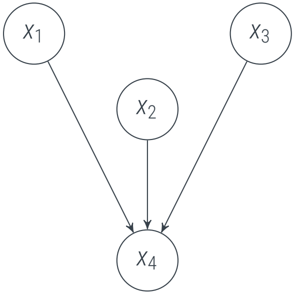
$$\begin{aligned} p(\boldsymbol{x}) &= p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_2) \cdots p(x_n \mid x_{n-1}) \\ &= \frac{1}{Z} \psi_{1,2}(x_1, x_2) \cdot \psi_{2,3}(x_2, x_3) \cdots \psi_{n-1,n}(x_{n-1}, x_n) \end{aligned}$$

The MRF for a directed chain graph is a **Markov Chain**.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

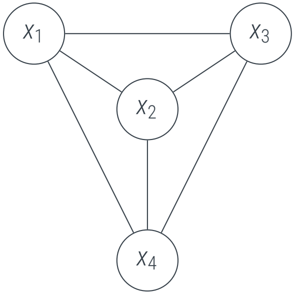
[from Bishop, PRML, 2006]

- we need to ensure that each conditional term in the directed graph are captured in at least one clique of the undirected graph
- for nodes with only one parent, we can thus simply drop the arrow, and get $p(x_c \mid x_p) = \phi_{c,p}(x_c, x_p)$
- but for nodes with several parents, we have to connect ("marry") all the parents. This process is known as **moralization**.
- moralization frequently leads to densely connected graphs, losing all value of the graph.



$$p(\boldsymbol{x}) = p(x_1) \cdot p(x_2) \cdot p(x_3) \cdot p(x_4 \mid x_1, x_2, x_3)$$

- we need to ensure that each conditional term in the directed graph are captured in at least one clique of the undirected graph
- for nodes with only one parent, we can thus simply drop the arrow, and get $p(x_c \mid x_p) = \phi_{c,p}(x_c, x_p)$
- but for nodes with several parents, we have to connect ("marry") all the parents. This process is known as **moralization**.
- moralization frequently leads to densely connected graphs, losing all value of the graph.



$$p(\mathbf{x}) = p(x_1) \cdot p(x_2) \cdot p(x_3) \cdot p(x_4 \mid x_1, x_2, x_3)$$

Limits of Both Model Families

The Graph for Two Coins and a Bell

example by Stefan Harmeling

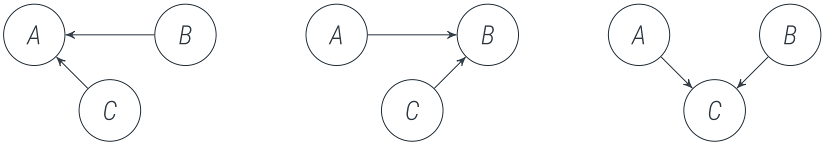
$$P(A=1)=0.5 \qquad P(C=1\mid A=1,B=1)=1 \qquad P(C=1\mid A=1,B=0)=0$$
$$P(B=1)=0.5 \qquad P(C=1\mid A=0,B=1)=0 \qquad P(C=1\mid A=0,B=0)=1$$

These CPTs imply $P(A|B)=P(A)$, $P(B|C)=P(B)$ and $P(C|A)=P(C)$ and $P(C\mid B)=P(C)$.

We thus have three factorizations:

1. $P(A,B,C)=P(C|A,B)\cdot P(A|B)\cdot P(B)=P(C|A,B)\cdot P(A)\cdot P(B)$
2. $P(A,B,C)=P(A|B,C)\cdot P(B|C)\cdot P(C)=P(A|B,C)\cdot P(B)\cdot P(C)$
3. $P(A,B,C)=P(B|C,A)\cdot P(C|A)\cdot P(A)=P(B|C,A)\cdot P(C)\cdot P(A)$

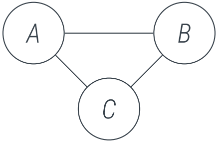
Each corresponds to a graph. Note that each can only express some of the independencies:



Limits of Both Families

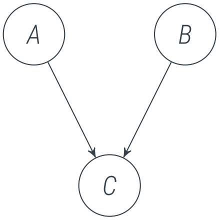
undirected case

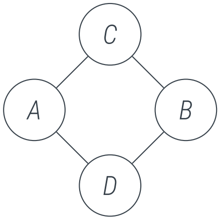
The MRF for "two coins and a bell", however, is totally useless.
It does not capture any of the conditional independencies.



Limits of Both Families

[from Bishop, PRML, 2006]





$A \perp\!\!\!\perp B \mid \emptyset$ and $A \not\perp\!\!\!\perp B \mid C$

$x \not\perp\!\!\!\perp y \mid \emptyset \ \forall x, y$ and $C \perp\!\!\!\perp D \mid A \cup B$ and $A \perp\!\!\!\perp B \mid C \cup D$

The conditional independence properties of the directed graph on the left can not be represented by any MRF over the same three variables; and the conditional independence properties of the MRF on the right can not be represented by any directed graph on the same four variables.

Directed and Undirected Graphs fit different problems

[cf. Bishop, 2006]

- Consider a distribution $p(\mathbf{x})$ and a graph $G = (V_{\mathbf{x}}, E)$.
- If every conditional *in*dependence statement satisfied by the distribution can be read off from the graph, then G is called an D-**map** of p. (The fully disconnected Graph is a trivial D-map for every p)
- If every conditional independence statement implied by G is also satisfied by p, then G is called a I-**map** of p. (The fully connected graph is a trivial I-map for every p).
- A G that is both an I-map and a D-map of p is called a **perfect map** of p.
- The set of distributions p for which there exists a directed graph that is a perfect map is distinct from the set of p for which there exists a perfect MRF map. (see two examples on previous slide. Markov Chains are an example where both MRF and directed graph are perfect). And there exist p for which there exists neither a directed nor an undirected perfect map (e.g. two coins and bell)

Summary so far:

- directed and undirected graphs offer tools to graphically represent and inspect properties of joint probability distributions. Both are primarily a **design tool**
- each framework has its strengths and weaknesses. Strong simplification:

  Bayes Nets for encoding **structured generative knowledge** over heterogeneous variable sets, e.g. in scientific modelling

  MRFs for encoding **computational constraints** over large sets of similar variables, e.g. in computer vision (pixels)

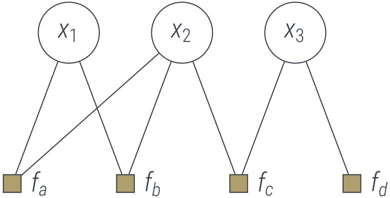
next goal:

- a third type of graphical model, particularly well-suited for *automated* inference
- a general-purpose algorithm for *automated* inference
- a variant for efficient MAP inference

- Both directed and undirected graphs provide a **factorization** of a distribution into **functions** over sets of **variables**

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

directed: $f_s(\boldsymbol{x}_s)$ – conditional distribution
undirected: $f_s(\boldsymbol{x}_s)$ – potential function ($Z = f_z(\varnothing)$)



Definition

A **factor graph** is a *bipartite* graph $G = (V, F, E)$ of *variables* $v_i \in V$, *factors* $f_i \in F$ and *edges*, such that each edge connects a factor to a variable.

images: Kschischang: U Toronto; Frey: Toronto Star; Loeliger: ETH Zürich

Example

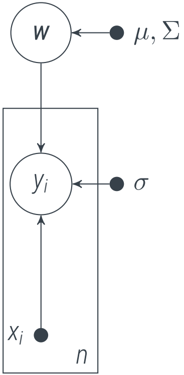
Parametric Regression

$$p(\boldsymbol{y}, \boldsymbol{w}) = \prod_{i=1}^{n} \mathcal{N}(y_i; \phi(x_i)^\intercal \boldsymbol{w}, \sigma^2)\mathcal{N}(\boldsymbol{w}; \mu, \Sigma)$$

To construct a factor graph from a directed graph

$$p(\boldsymbol{x}) = \prod_{c \in C} p_c(\boldsymbol{x}_c \mid x_{\mathrm{pa}(c)})$$

- draw a circle for each variable x_i
- draw a box for each conditional p_c
- connect each p_c to the variables in it

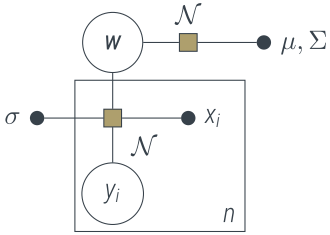


$$p(\mathbf{y}, \mathbf{w}) = \prod_{i=1}^{n} \mathcal{N}(y_i; \phi(x_i)^\intercal \mathbf{w}, \sigma^2)\mathcal{N}(\mathbf{w}; \mu, \Sigma)$$

To construct a factor graph from a directed graph

$$p(\mathbf{x}) = \prod_{c \in C} p_c(\mathbf{x}_c \mid x_{\mathrm{pa}(c)})$$

- draw a circle for each variable x_i
- draw a box for each conditional p_c
- connect each p_c to the variables in it

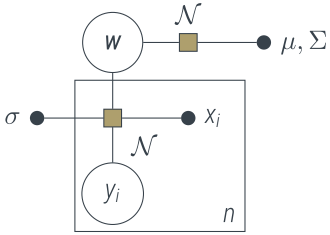


$$p(\boldsymbol{y}, \boldsymbol{w}) = \prod_{i=1}^{n} \mathcal{N}(y_i; \phi(x_i)^\intercal \boldsymbol{w}, \sigma^2)\mathcal{N}(\boldsymbol{w}; \mu, \Sigma)$$

To construct a factor graph from a MRF

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{c \in C} \psi_c(\boldsymbol{x}_c)$$

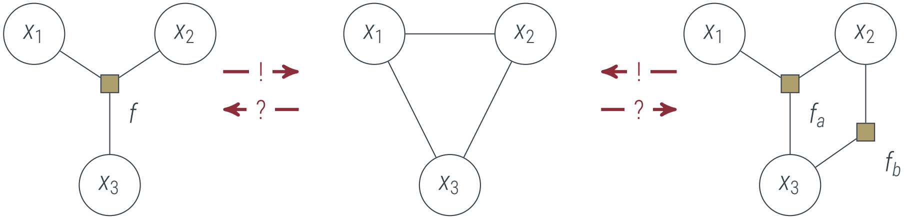
- draw a circle for each variable x_i
- draw a box for each factor (potential) ψ
- connect each ψ to the variables used in the factor



Explicit Functional Relationships Reveal Structure

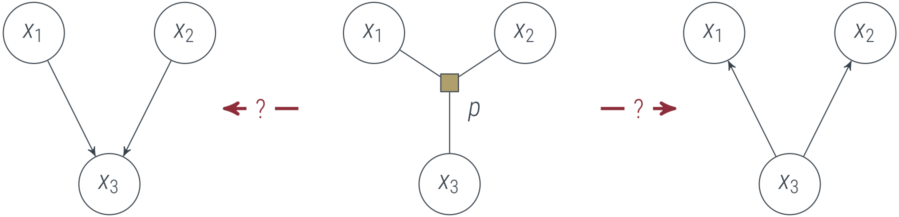
Factor Graphs can express structure not visible in Undirected Graphs

$$p(x_1, x_2, x_3) = f_a(x_1, x_2, x_3) \cdot f_b(x_2, x_3)$$



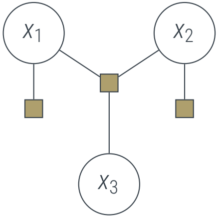
Functional relationships have no direction, but they identify parents

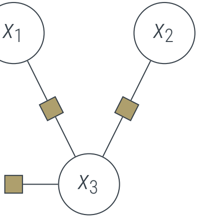
Factor graphs can capture DAGs if we're careful



$$p(x_1, x_2, x_3) = p(x_3 \mid x_1, x_2)p(x_1)p(x_2)$$

$$p(x_1, x_2, x_3) = p(x_1 \mid x_3)p(x_2 \mid x_3)p(x_3)$$





What do we have to do, in general, to compute a **marginal** distribution

$$p(x_i) = \int p(x_1, \dots, x_i, \dots, x_n) \, dx_{j \neq i}$$

If the joint $p(x_1, \dots, x_n)$ is given by a factor graph?

The Sum-Product Algorithm

Automated Inference in Factor Graphs



- J. Pearl. Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, 1988.
- S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. J. R. Stat. Soc., 50:157–224, 1988.



- F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum- product algorithm. IEEE Transactions on Information Theory, 47(2):498–519, February 2001.

The Sum-Product-Algorithm we are about to develop unifies many historically separate ideas (as listed by H.A. Loeliger, 2008):

Statistical physics:

- Markov random fields (Ising 1925)

Signal processing:

- linear state-space models and Kalman filtering: Kalman 1960…
- recursive least-squares adaptive filters
- Hidden Markov models: Baum et al. 1966…
- unification: Levy et al. 1996…

Error correcting codes:

- Low-density parity check codes: Gallager 1962; Tanner 1981; MacKay 1996; Luby et al. 1998…
- Convolutional codes and Viterbi decoding: Forney 1973…
- Turbo codes: Berrou et al. 1993…

Machine learning:

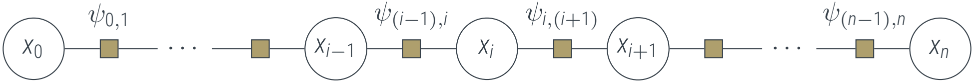
- Bayesian networks: Pearl 1988; Shachter 1988; Lauritzen and Spiegelhalter 1988; Shafer and Shenoy 1990…

Base Case: Markov Chains

Filtering and Smoothing are special cases of the sum-product algorithm

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

[exposition based on Bishop, PRML, 2006]



Assume discrete $x_i \in [1,\dots,k]$ for the moment. What is the **marginal** $p(x_i)$?

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{0,1}(x_0,x_1)\cdots\psi_{i-1,i}(x_{i-1},x_i)\cdot\psi_{i,i+1}(x_i,x_{i+1})\cdot\psi_{n-1,n}(x_{n-1},x_n)$$

$$p(x_i) = \sum_{x_{\neq i}} p(\mathbf{x}) = \frac{1}{Z}\underbrace{\left(\sum_{x_{i-1}}\psi_{i-1,i}(x_{i-1},x_i)\cdots\left(\sum_{x_1}\psi_{1,2}(x_1,x_2)\left(\sum_{x_0}\psi(x_0,x_1)\right)\right)\right)}_{=:\mu_{\rightarrow}(x_i)}$$

$$\cdot\underbrace{\left(\sum_{x_{i+1}}\psi_{i,i+1}(x_i,x_{i+1})\cdots\left(\sum_{x_n}\psi_{n-1,n}(x_{n-1},x_n)\right)\right)}_{=:\mu_{\leftarrow}(x_i)} = \frac{1}{Z}\mu_{\rightarrow}(x_i)\cdot\mu_{\leftarrow}(x_i).$$

$$p(x_i) = \frac{1}{Z} \underbrace{\left(\sum_{x_{i-1}} \psi_{i-1,i}(x_{i-1}, x_i) \cdots \left(\sum_{x_1} \psi_{1,2}(x_1, x_2) \left(\sum_{x_0} \psi(x_0, x_1) \right) \right) \right)}_{=: \mu_{\rightarrow}(x_i)} \cdot \underbrace{\left(\sum_{x_{i+1}} \psi_{i,i+1}(x_i, x_{i+1}) \cdots \left(\sum_{x_n} \psi_{n-1,n}(x_{n-1}, x_n) \right) \right)}_{=: \mu_{\leftarrow}(x_i)}.$$

- Marginal can be computed **locally**

$$p(x_i) = \frac{1}{Z} \mu_{\rightarrow}(x_i) \cdot \mu_{\leftarrow}(x_i) \quad \text{with} \quad Z = \sum_{x_i} \mu_{\rightarrow}(x_i) \cdot \mu_{\leftarrow}(x_i)$$

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

$$p(x_i) = \frac{1}{Z} \underbrace{\left(\sum_{x_{i-1}} \psi_{i-1,i}(x_{i-1},x_i) \cdots \left(\sum_{x_1} \psi_{1,2}(x_1,x_2) \left(\sum_{x_0} \psi(x_0,x_1)\right)\right)\right)}_{=:\mu_{\rightarrow}(x_i)} \cdot \underbrace{\left(\sum_{x_{i+1}} \psi_{i,i+1}(x_i,x_{i+1}) \cdots \left(\sum_{x_n} \psi_{n-1,n}(x_{n-1},x_n)\right)\right)}_{=:\mu_{\leftarrow}(x_i)}.$$

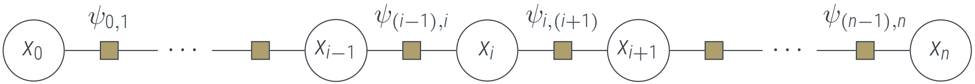
- Messages are recursive, thus computational complexity is $\mathcal{O}(n \cdot k^2)$

$$\mu_{\rightarrow}(x_i) = \sum_{i-1} \psi_{i-1,i}(x_{i-1},x_i)\mu_{\rightarrow}(x_{i-1}) \qquad \mu_{\leftarrow}(x_i) = \sum_{x_{i+1}} \psi_{i,i+1}(x_i,x_{i+1})\mu_{\leftarrow}(x_{i+1}).$$

- By storing local messages, **all marginals** can be computed in $\mathcal{O}(n \cdot k^2)$ (cf. filtering and smoothing)
- To compute one message from the preceding one, take the **sum** over the preceding variable in (the **product** of) the local factors incoming message(s). To compute a local marginal, take the **sum** of the **product** of the incoming messages.

How about the most probable State?

The **Viterbi** Algorithm



Assume discrete $x_i \in [1, \dots, k]$ for the moment. Where is the **maximum** $\max p(\mathbf{x})$?

$$p(\boldsymbol{x}) = \frac{1}{Z}\psi_{0,1}(x_0, x_1) \cdots \psi_{i-1,i}(x_{i-1}, x_i) \cdot \psi_{i,i+1}(x_i, x_{i+1}) \cdot \psi_{n-1,n}(x_{n-1}, x_n)$$

$$\max_{\boldsymbol{x}} p(\boldsymbol{x}) = \frac{1}{Z} \max_{x_0} \cdots \max_{x_N} \psi_{0,1}(x_0, x_1) \cdots \psi_{n-1,n}(x_{n-1}, x_n)$$

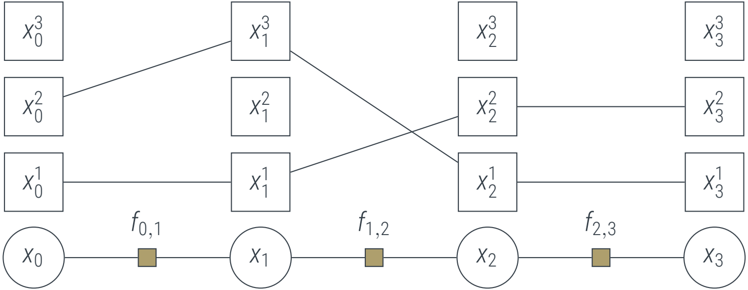
$$= \frac{1}{Z} \max_{x_0, x_1} \left(\psi_{0,1}(x_0, x_1) \left(\cdots \max_{x_n} \psi_{n-1,n}(x_{n-1}, x_n) \right) \right)$$

$$\log \max_{\boldsymbol{x}} p(\boldsymbol{x}) = \max_{x_0, x_1} \left(\log \psi_{0,1}(x_0, x_1) + \left(\cdots \max_{x_n} \log \psi_{n-1,n}(x_{n-1}, x_n) \right) \right) - \log Z$$

$$\arg\max_{\boldsymbol{x}} p(\boldsymbol{x}) = \arg\max_{x_0, x_1} \left(\log \psi_{0,1}(x_0, x_1) + \left(\cdots + \arg\max_{x_n} \log \psi_{n-1,n}(x_{n-1}, x_n) \right) \right)$$

The Viterbi Algorith

On a **trellis** (="Spalier") [based on Fig.8.53 in Bishop, PRML, 2006]



$$\mu_{x_0 \to f_{01}} = 0$$

$$\mu_{f_{i-1,i} \to x_i}(x_i) = \max_{x_{i-1}} \left(\log f_{i-1,i}(x_{i-1}, x_i) + \mu_{x_{i-1} \to f_{i-1,i}}(x_{i-1})\right)$$

$$\phi(x_i) = \arg\max_{x_{i-1}} \left(\log f_{i-1,i}(x_{i-1}, x_i) + \mu_{x_{i-1} \to f_{i-1,i}}(x_i)\right)$$

$$\mu_{x_i \to f_{i,i+1}}(x_i) = \mu_{f_{i-1,i} \to x_i}(x_i)$$

$$x_{i-1}^{\max} = \phi(x_i^{\max})$$

Factor Graphs

- are a tool to directly represent an entire computation in a formal language (which also includes the functions in question themselves)
- both directed and undirected graphical models can be mapped onto factor graphs.

Inference on Chains

- separates into **local messages** being sent forwards and backwards along the factor graph
- both the local marginals and the *most-probable state* can be inferred in this way. For the most probable state, we need to additionally keep track of its identity, which requires an additional data structure (a *trellis*).
- more fundamentally, both algorithms utilize the *distributive* property of sum and max:

$$+(ab, ac) = ab + ac = a(b + c) = a \cdot +(b, c)$$
$$\max(ab, ac) = a \cdot \max(b, c)$$
$$\max(a + b, a + c) = a + \max(b, c)$$