Statistical Machine Learning

https://cvml.ist.ac.at/courses/SML_W18

Christoph Lampert

IST AUSTRIA

Institute of Science and Technology

Winter Semester 2018/2019

Lecture 10

(lots of material courtesy of S. Nowozin, http://www.nowozin.net)

Overview (tentative)

Date		no.	Topic
Oct 08	Mon	1	A Hands-On Introduction
Oct 10	Wed	–	self-study (Christoph traveling)
Oct 15	Mon	2	Bayesian Decision Theory
			Generative Probabilistic Models
Oct 17	Wed	3	Discriminative Probabilistic Models
			Maximum Margin Classifiers
Oct 22	Mon	4	Generalized Linear Classifiers, Optimization
Oct 24	Wed	5	Evaluating Predictors; Model Selection
Oct 29	Mon	–	self-study (Christoph traveling)
Oct 31	Wed	6	Overfitting/Underfitting, Regularization
Nov 05	Mon	7	Learning Theory I: classical/Rademacher bounds
Nov 07	Wed	8	Learning Theory II: miscellaneous
Nov 12	Mon	9	Probabilistic Graphical Models I
Nov 14	Wed	10	Probabilistic Graphical Models II
Nov 19	Mon	11	Probabilistic Graphical Models III
Nov 21	Wed	12	Probabilistic Graphical Models IV
until Nov 25			final project

Probabilistic Inference

$$p(y_F|x)$$

Probabilistic Inference – Overview

Goal: for fixed model and x, compute $Z(x)$ or $p(y_F|x;w)$

Exact Inference

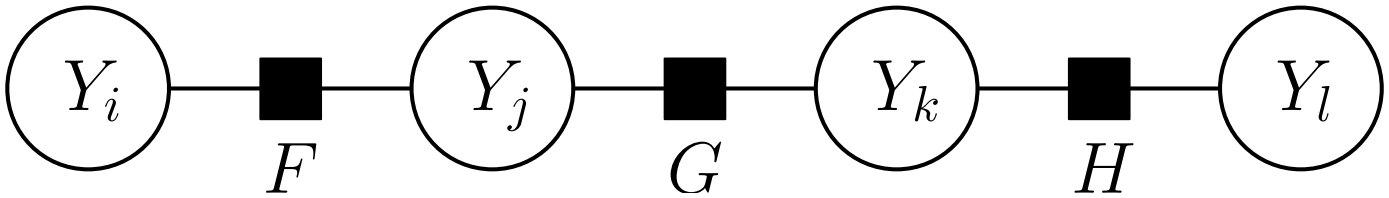
- Belief Propagation on chains
- Belief Propagation on trees
- Junction tree algorithm

Approximate Inference

- Loopy Belief Propagation
- Markov Chain Monte Carlo (MCMC) Sampling
- Variational Inference / Mean Field

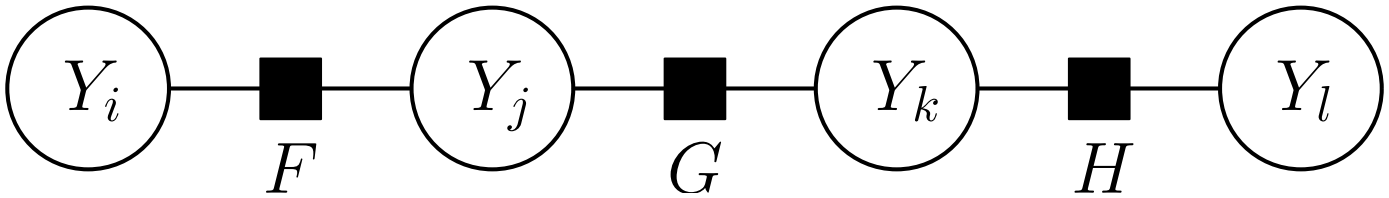
Probabilistic Inference – Belief Propagation

Assume $y = (y_i, y_j, y_k, y_l)$, $\mathcal{Y} = \mathcal{Y}_i \times \mathcal{Y}_j \times \mathcal{Y}_k \times \mathcal{Y}_l$, and an energy function $E(y; x)$ compatible with the following factor graph:



Probabilistic Inference – Belief Propagation

Assume $y = (y_i, y_j, y_k, y_l)$, $\mathcal{Y} = \mathcal{Y}_i \times \mathcal{Y}_j \times \mathcal{Y}_k \times \mathcal{Y}_l$, and an energy function $E(y; x)$ compatible with the following factor graph:

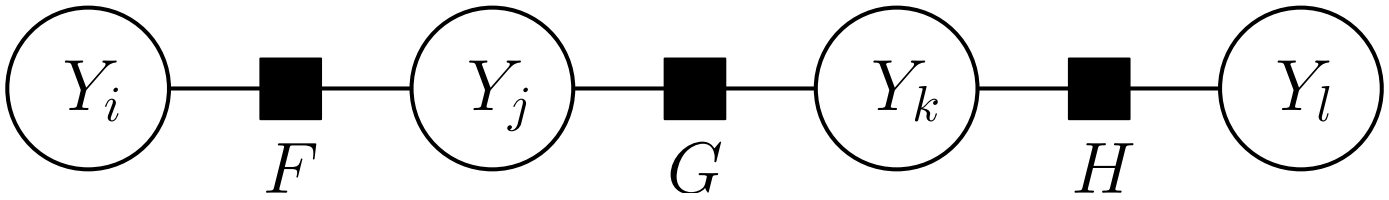


Task 1: for any $y \in \mathcal{Y}$, compute $p(y|x)$, using

$$p(y|x) = \frac{1}{Z(x)} e^{-E(y;x)}.$$

Probabilistic Inference – Belief Propagation

Assume $y = (y_i, y_j, y_k, y_l)$, $\mathcal{Y} = \mathcal{Y}_i \times \mathcal{Y}_j \times \mathcal{Y}_k \times \mathcal{Y}_l$, and an energy function $E(y; x)$ compatible with the following factor graph:



Task 1: for any $y \in \mathcal{Y}$, compute $p(y|x)$, using

$$p(y|x) = \frac{1}{Z(x)} e^{-E(y;x)}.$$

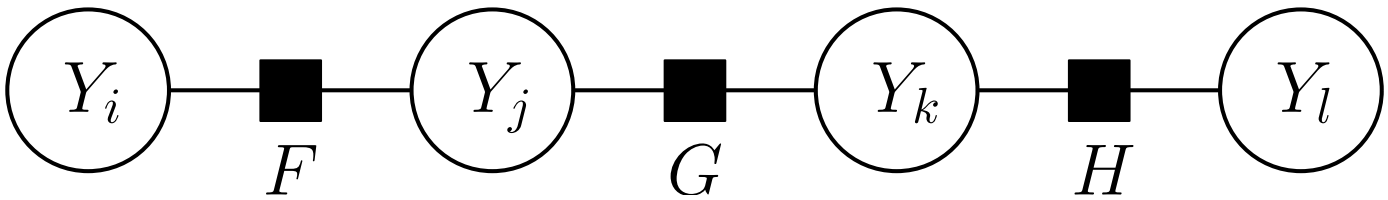
Problem: We don't know $Z(x)$, and computing it using

$$Z(x) = \sum_{y \in \mathcal{Y}} e^{-E(y;x)}$$

looks expensive (the sum has $|\mathcal{Y}_i| \cdot |\mathcal{Y}_j| \cdot |\mathcal{Y}_k| \cdot |\mathcal{Y}_l|$ terms).

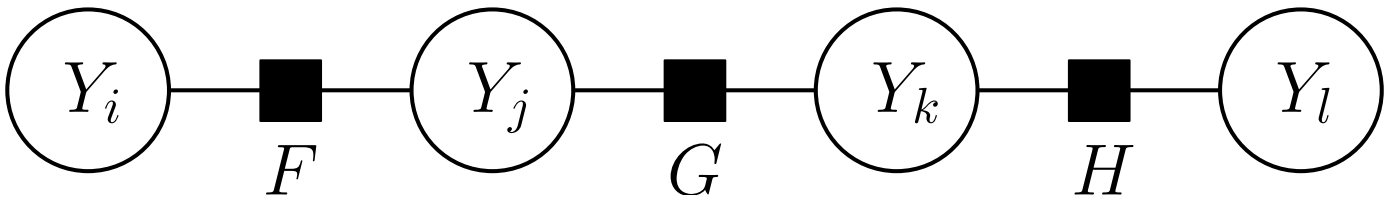
A lot research has been done on how to **efficiently compute** $Z(x)$.

Probabilistic Inference – Belief Propagation



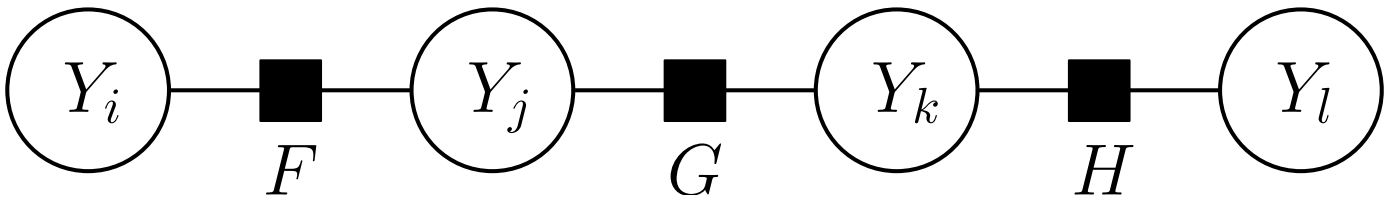
For notational simplicity, we drop the dependence on (fixed) x:

$$Z = \sum_{y \in \mathcal{Y}} e^{-E(y)}$$



For notational simplicity, we drop the dependence on (fixed) x:

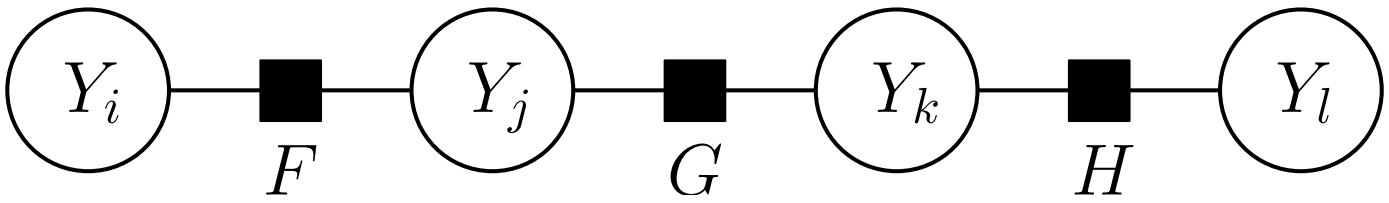
$$
\begin{aligned}
Z &= \sum_{y \in \mathcal{Y}} e^{-E(y)} \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} e^{-E(y_i, y_j, y_k, y_l)}
\end{aligned}
$$



For notational simplicity, we drop the dependence on (fixed) x:

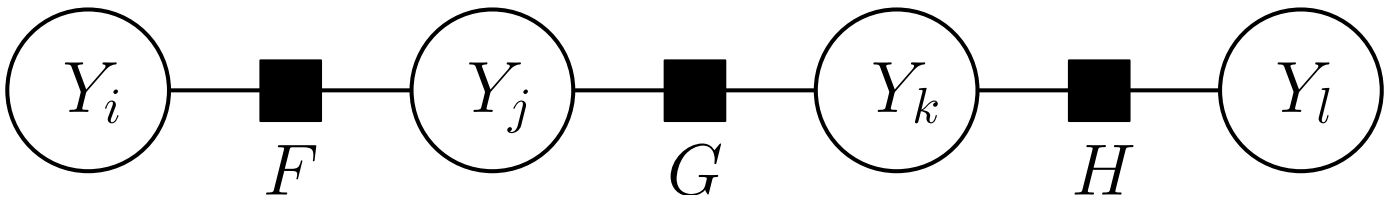
$$
\begin{aligned}
Z &= \sum_{y \in \mathcal{Y}} e^{-E(y)} \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} e^{-E(y_i, y_j, y_k, y_l)} \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} e^{-E_F(y_i, y_j) - E_G(y_j, y_k) - E_H(y_k, y_l)}
\end{aligned}
$$

Probabilistic Inference – Belief Propagation



$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} e^{-E_F(y_i, y_j) - E_G(y_j, y_k) - E_H(y_k, y_l)}$$

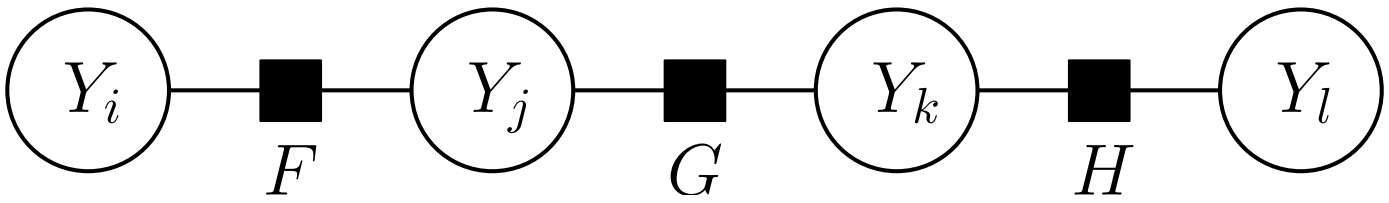
Probabilistic Inference – Belief Propagation



$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} e^{-E_F(y_i,y_j) - E_G(y_j,y_k) - E_H(y_k,y_l)}$$

$$= \sum_{y_i} \sum_{y_j} \sum_{y_k} \sum_{y_l} e^{-E_F(y_i,y_j)} e^{-E_G(y_j,y_k)} e^{-E_H(y_k,y_l)}$$

Probabilistic Inference – Belief Propagation

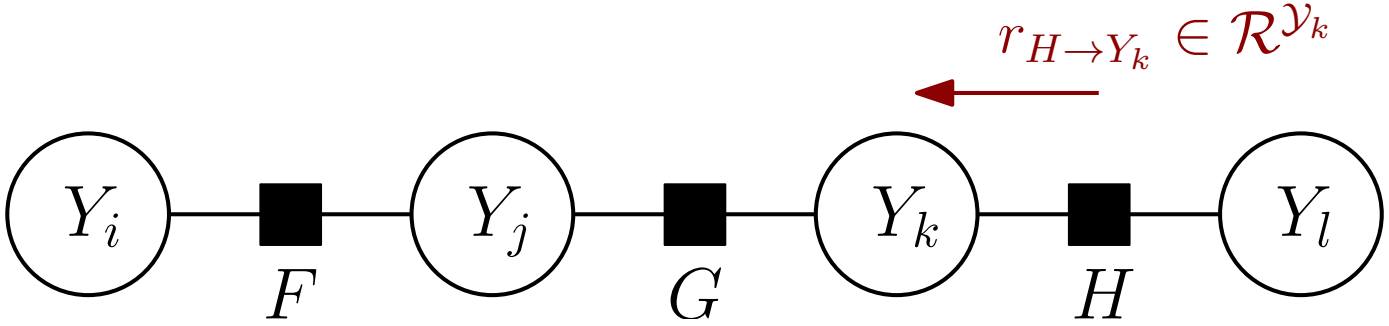


$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} e^{-E_F(y_i,y_j) - E_G(y_j,y_k) - E_H(y_k,y_l)}$$

$$= \sum_{y_i} \sum_{y_j} \sum_{y_k} \sum_{y_l} e^{-E_F(y_i,y_j)} e^{-E_G(y_j,y_k)} e^{-E_H(y_k,y_l)}$$

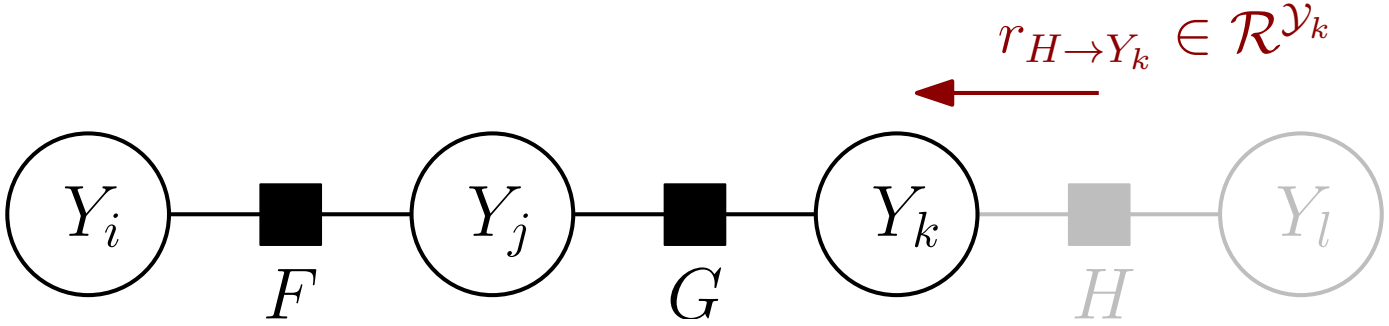
$$= \sum_{y_i} \sum_{y_j} e^{-E_F(y_i,y_j)} \sum_{y_k} e^{-E_G(y_j,y_k)} \sum_{y_l} e^{-E_H(y_k,y_l)}$$

Probabilistic Inference – Belief Propagation



$$Z = \sum_{y_i} \sum_{y_j} e^{-E_F(y_i, y_j)} \sum_{y_k} e^{-E_G(y_j, y_k)} \underbrace{\sum_{y_l} e^{-E_H(y_k, y_l)}}_{r_{H \to Y_k}(y_k)}$$

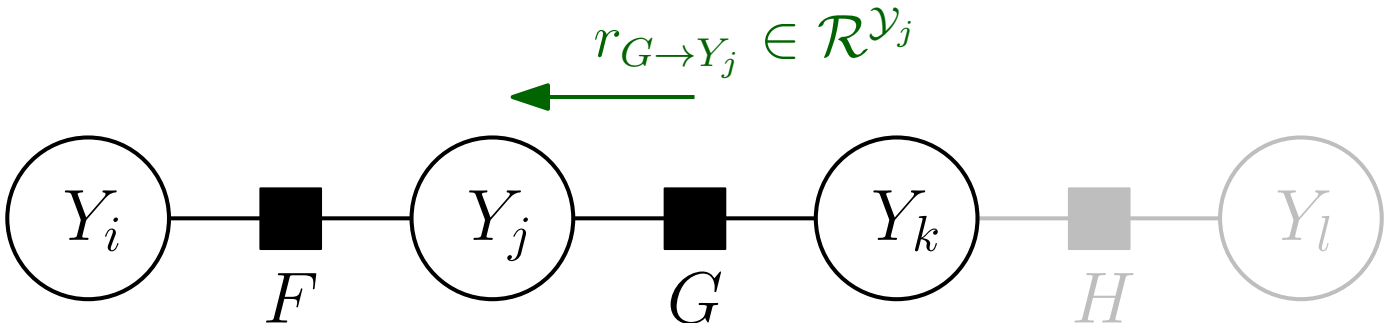
Probabilistic Inference – Belief Propagation



$$Z = \sum_{y_i} \sum_{y_j} e^{-E_F(y_i, y_j)} \sum_{y_k} e^{-E_G(y_j, y_k)} \underbrace{\sum_{y_l} e^{-E_H(y_k, y_l)}}_{r_{H \to Y_k}(y_k)}$$

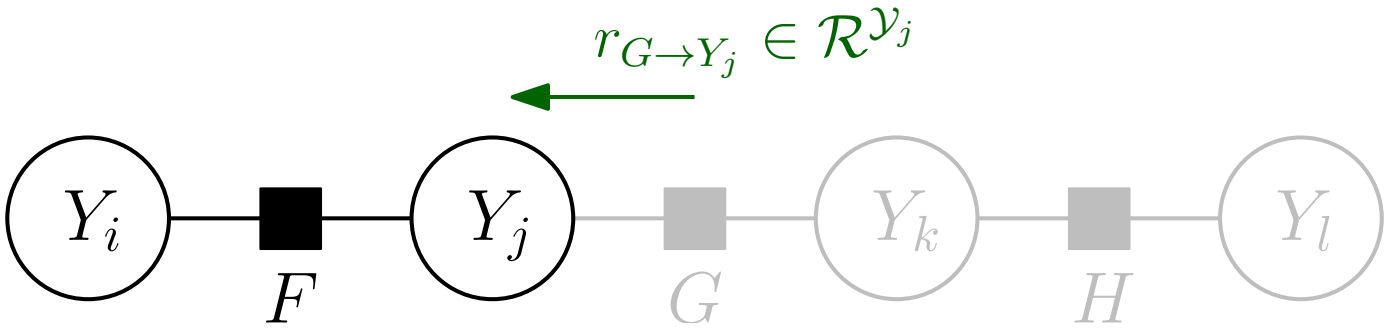
$$= \sum_{y_i} \sum_{y_j} e^{-E_F(y_i, y_j)} \sum_{y_k} e^{-E_G(y_j, y_k)} r_{H \to Y_k}(y_k)$$

Probabilistic Inference – Belief Propagation



$$Z = \sum_{y_i} \sum_{y_j} e^{-E_F(y_i, y_j)} \underbrace{\sum_{y_k} e^{-E_G(y_j, y_k)} r_{H \to Y_k}(y_k)}_{r_{G \to Y_j}(y_j)}$$

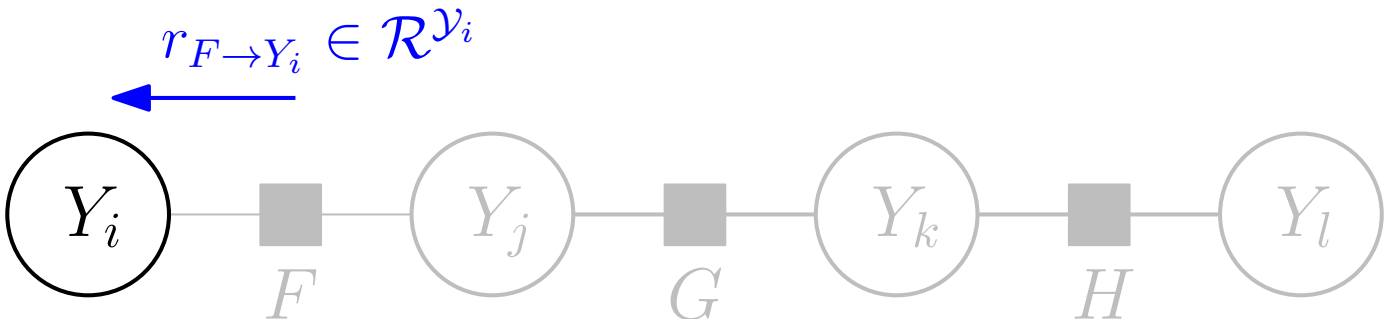
Probabilistic Inference – Belief Propagation



$$Z = \sum_{y_i} \sum_{y_j} e^{-E_F(y_i, y_j)} \underbrace{\sum_{y_k} e^{-E_G(y_j, y_k)} r_{H \to Y_k}(y_k)}_{r_{G \to Y_j}(y_j)}$$

$$= \sum_{y_i} \underbrace{\sum_{y_j} e^{-E_F(y_i, y_j)} r_{G \to Y_j}(y_j)}_{r_{F \to Y_i}(y_i)}$$

Probabilistic Inference – Belief Propagation

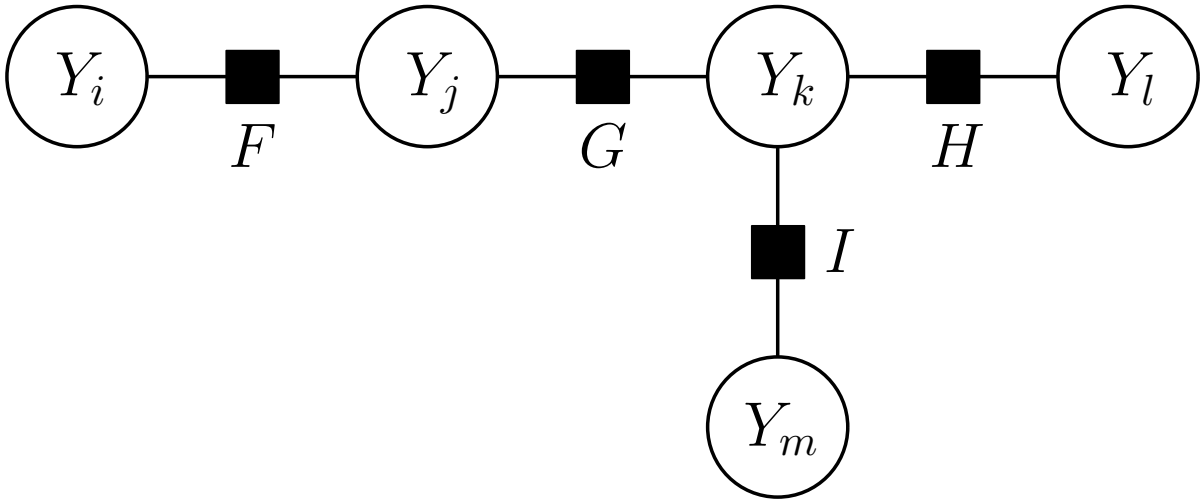


$$Z = \sum_{y_i} \sum_{y_j} e^{-E_F(y_i, y_j)} \underbrace{\sum_{y_k} e^{-E_G(y_j, y_k)} r_{H \to Y_k}(y_k)}_{r_{G \to Y_j}(y_j)}$$

$$= \sum_{y_i} \underbrace{\sum_{y_j} e^{-E_F(y_i, y_j)} r_{G \to Y_j}(y_j)}_{r_{F \to Y_i}(y_i)}$$

$$= \sum_{y_i} r_{F \to Y_i}(y_i)$$

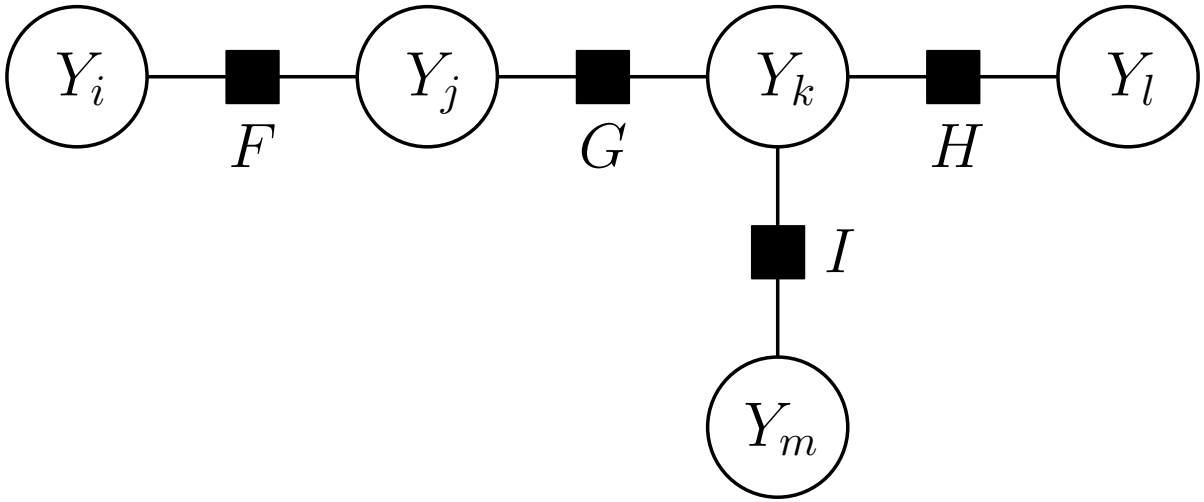
Example: Inference on Trees



1) pick a root (here: i)
2) and sort sums such that parents nodes are left of their children

$$
\begin{aligned}
Z &= \sum_{y \in \mathcal{Y}} e^{-E(y)} \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} \sum_{y_m \in \mathcal{Y}_m} e^{-E_F(y_i, y_j) - \ldots - E_I(y_k, y_m)}
\end{aligned}
$$

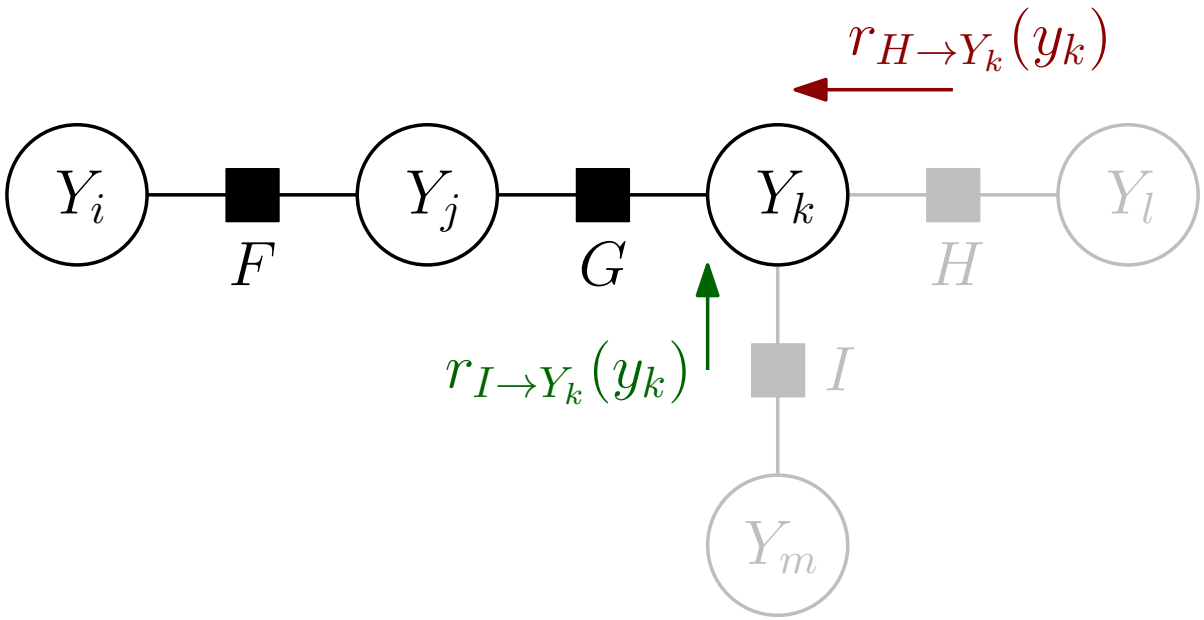
Example: Inference on Trees



1) pick a root (here: i)
2) and sort sums such that parents nodes are left of their children

$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} e^{-E_F(y_i, y_j)} \sum_{y_k \in \mathcal{Y}_k} e^{-E_G(y_j, y_k)} \cdot \underbrace{\left(\sum_{y_l \in \mathcal{Y}_l} e^{-E_H(y_k, y_l)} \right)}_{r_{H \to Y_k}(y_k)} \cdot \underbrace{\left(\sum_{y_m \in \mathcal{Y}_m} e^{-E_I(y_k, y_m)} \right)}_{r_{I \to Y_k}(y_k)}$$

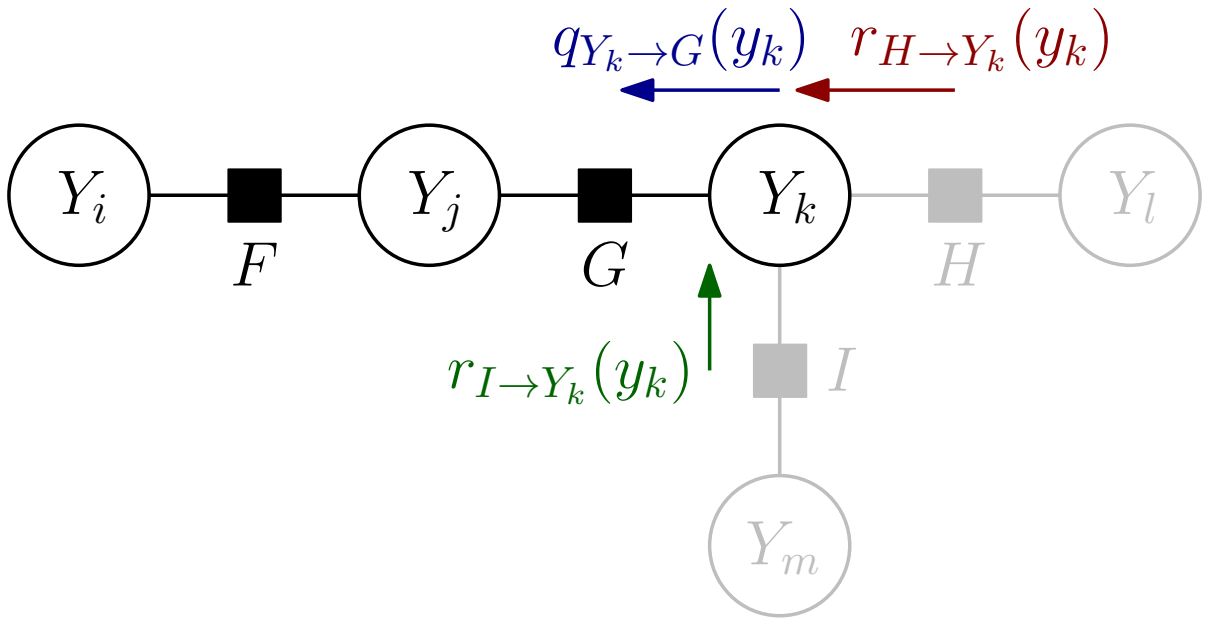
Example: Inference on Trees



1) pick a root (here: i)
2) and sort sums such that parents nodes are left of their children

$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} e^{-E_F(y_i, y_j)} \sum_{y_k \in \mathcal{Y}_k} e^{-E_G(y_j, y_k)} \cdot r_{H \to Y_k}(y_k) \cdot r_{I \to Y_k}(y_k)$$

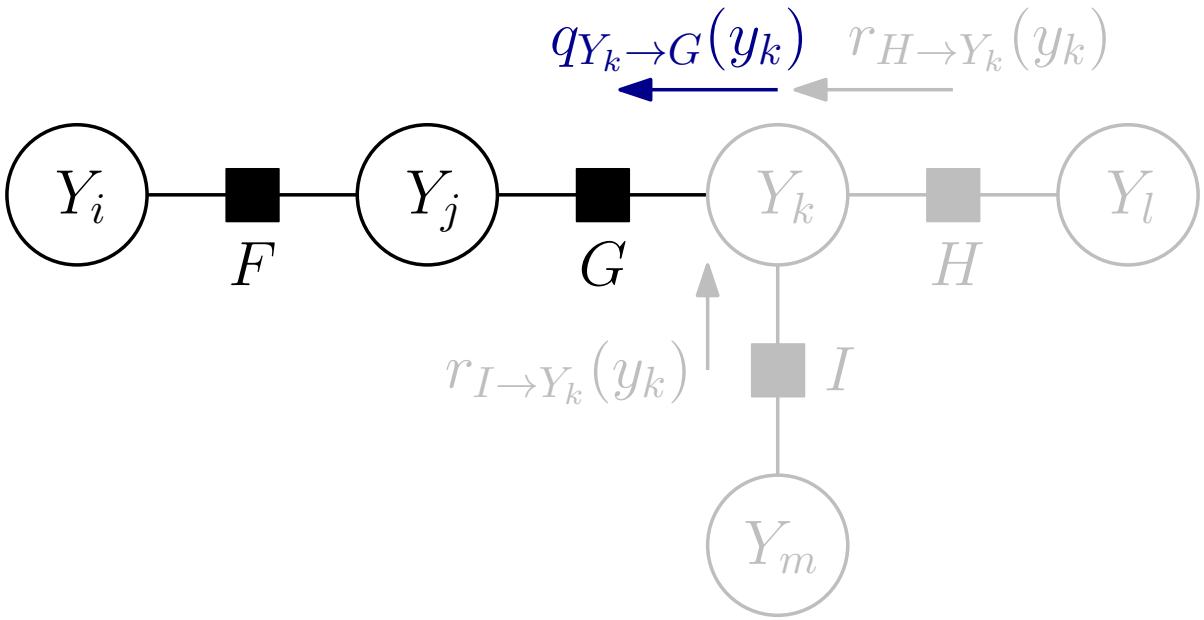
Example: Inference on Trees



1) pick a root (here: i)
2) and sort sums such that parents nodes are left of their children

$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} e^{-E_F(y_i, y_j)} \sum_{y_k \in \mathcal{Y}_k} e^{-E_G(y_j, y_k)} \underbrace{r_{H \to Y_k}(y_k) \cdot r_{I \to Y_k}(y_k)}_{q_{Y_k \to G}(y_k)}$$

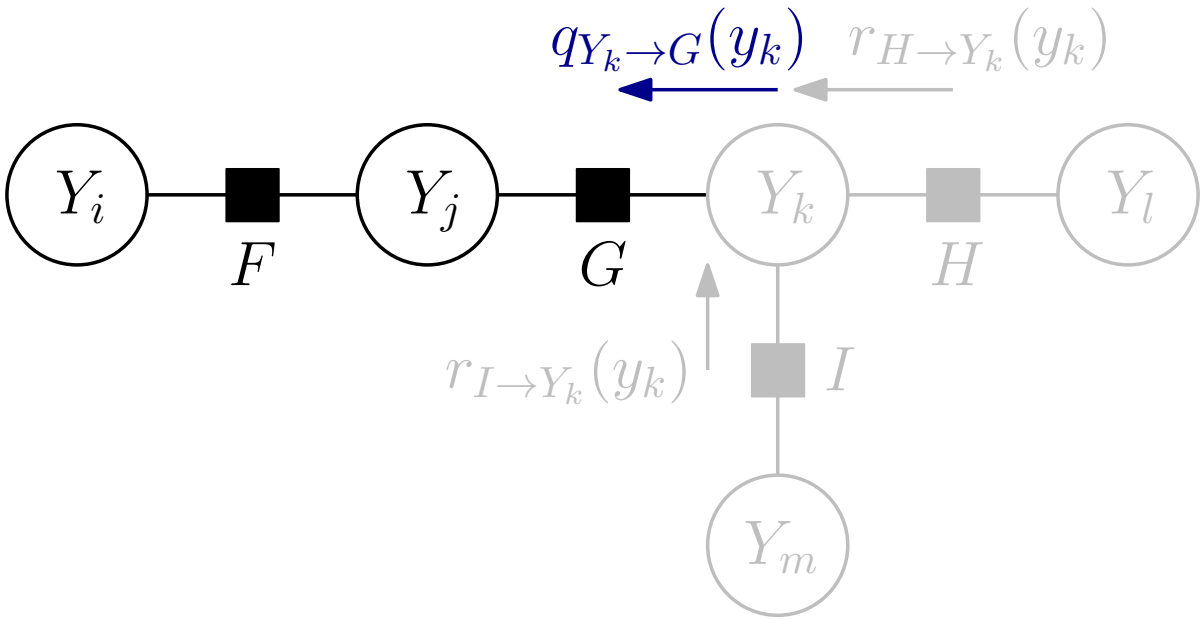
Example: Inference on Trees



1) pick a root (here: i)
2) and sort sums such that parents nodes are left of their children

$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} e^{-E_F(y_i, y_j)} \sum_{y_k \in \mathcal{Y}_k} e^{-E_G(y_j, y_k)} q_{Y_k \to G}(y_k)$$

Example: Inference on Trees



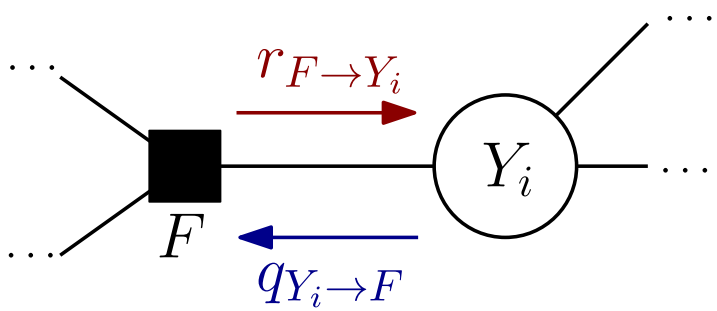
1) pick a root (here: i)
2) and sort sums such that parents nodes are left of their children

$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} e^{-E_F(y_i, y_j)} \sum_{y_k \in \mathcal{Y}_k} e^{-E_G(y_j, y_k)} q_{Y_k \to G}(y_k)$$

3) etc.

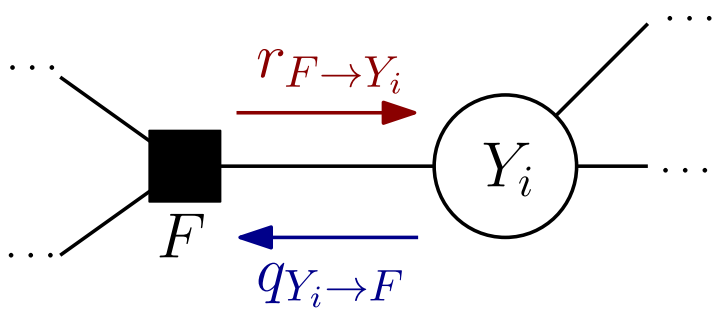
Factor Graph Sum-Product Algorithm

- "Message": pair of vectors at each factor graph edge $(i, F) \in \mathcal{E}$
  1. $r_{F \to Y_i} \in \mathbb{R}^{\mathcal{Y}_i}$: factor-to-variable message
  2. $q_{Y_i \to F} \in \mathbb{R}^{\mathcal{Y}_i}$: variable-to-factor message



Factor Graph Sum-Product Algorithm

- "Message": pair of vectors at each factor graph edge $(i, F) \in \mathcal{E}$
  1. $r_{F \to Y_i} \in \mathbb{R}^{\mathcal{Y}_i}$: factor-to-variable message
  2. $q_{Y_i \to F} \in \mathbb{R}^{\mathcal{Y}_i}$: variable-to-factor message



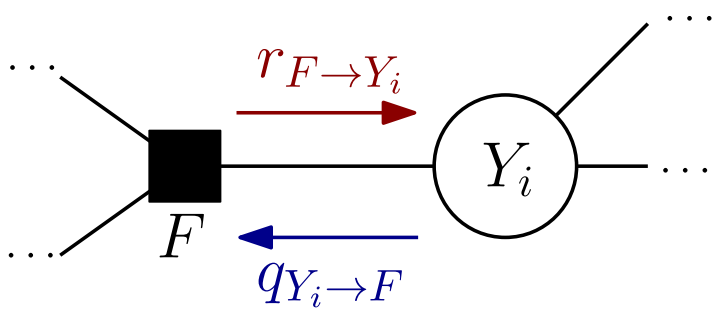
- Algorithm iteratively updates messages
- After convergence: Z and $p(y_F)$ can be obtained from the messages, e.g.

$$p(Y_i = y_i) \propto \prod_{F:(i,F)\in\mathcal{E}} r_{F \to Y_i}(y_i)$$

(Sum-Product) Belief Propagation

Factor Graph Sum-Product Algorithm

- "Message": pair of vectors at each factor graph edge $(i, F) \in \mathcal{E}$
  1. $r_{F \to Y_i} \in \mathbb{R}^{\mathcal{Y}_i}$: factor-to-variable message
  2. $q_{Y_i \to F} \in \mathbb{R}^{\mathcal{Y}_i}$: variable-to-factor message



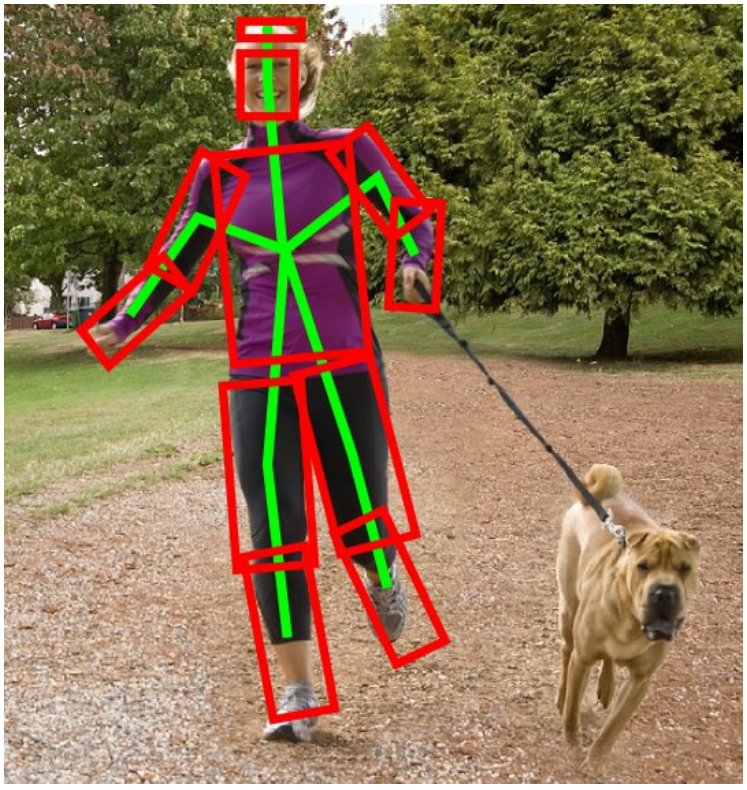
- Algorithm iteratively updates messages
- After convergence: Z and $p(y_F)$ can be obtained from the messages, e.g.

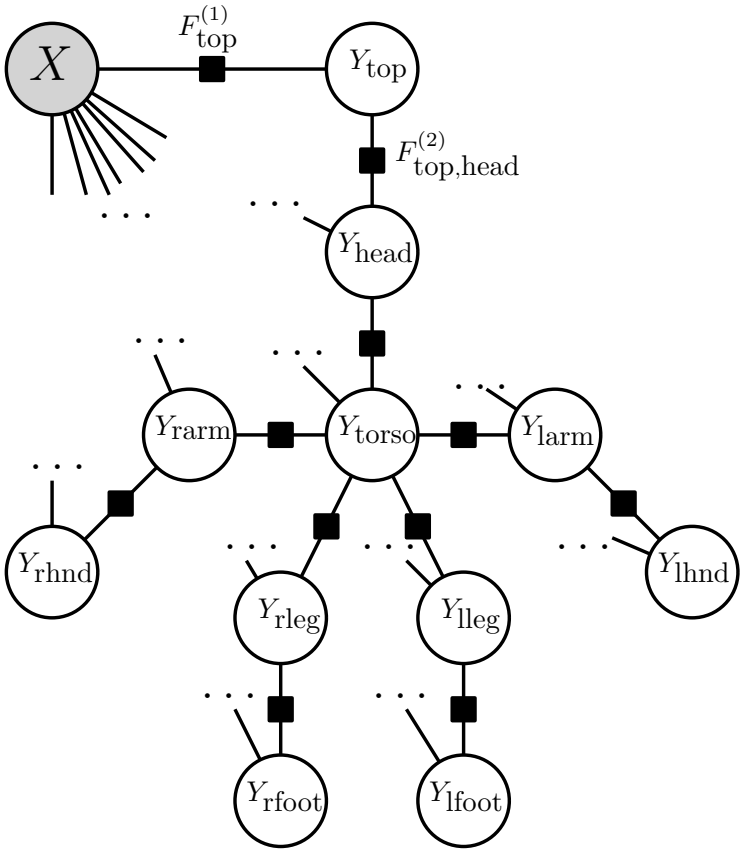
$$p(Y_i = y_i) \propto \prod_{F:(i,F)\in\mathcal{E}} r_{F \to Y_i}(y_i)$$

(Sum-Product) Belief Propagation

- Easier to implement than to explain...

Example: Pictorial Structures





- **Tree-structured model** for articulated pose
  (Felzenszwalb and Huttenlocher, 2000), (Fischler and Elschlager, 1973)
- Belief propagation is the state-of-the-art for prediction and inference

Example: Pictorial Structures

Probability of part states $\equiv$ body part locations:

estimated independently

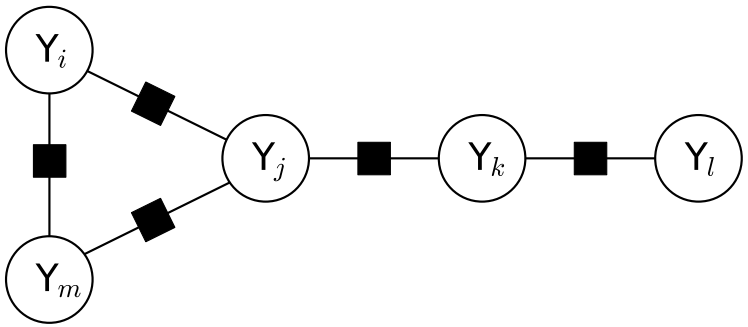
estimated from joint probability

- Marginal probabilities $p(y_i|x)$ provide
  - potential positions
  - uncertainty

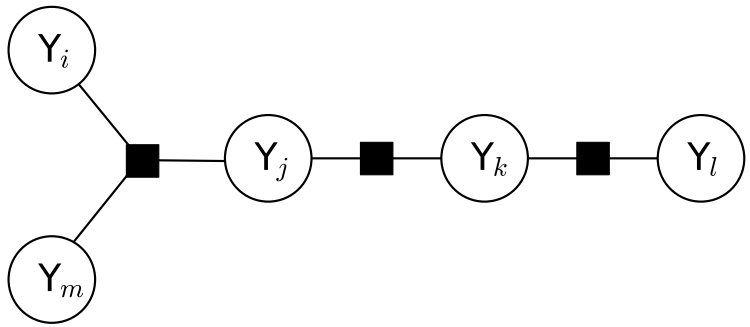
  of the body parts, taking into account also the other body parts.

Belief Propagation in Cyclic Graphs

Belief propagation does not work for



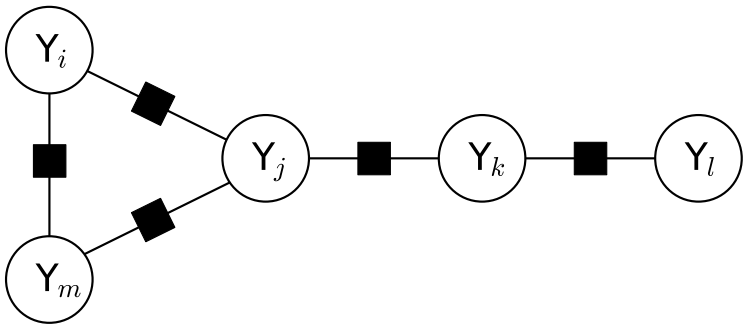
graph with cycles



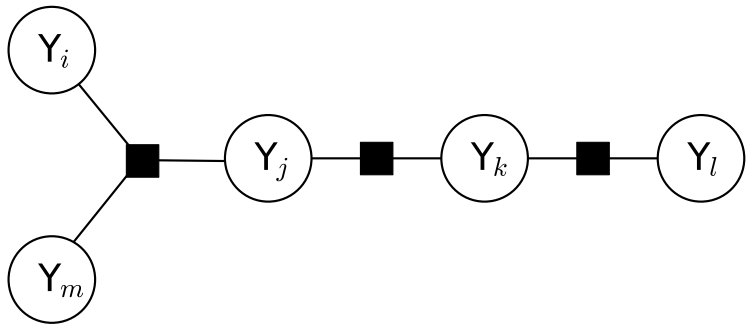
graph with factors of size more than 2

Belief Propagation in Cyclic Graphs

Belief propagation does not work for

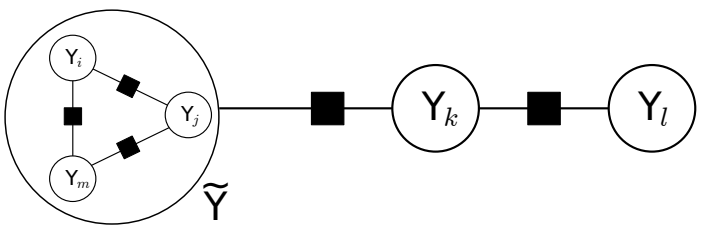


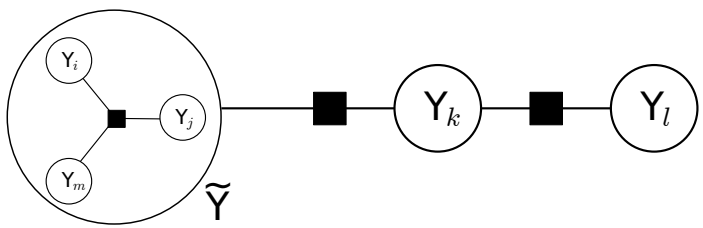
graph with cycles



graph with factors of size more than 2

We can construct equivalent chain/tree models:





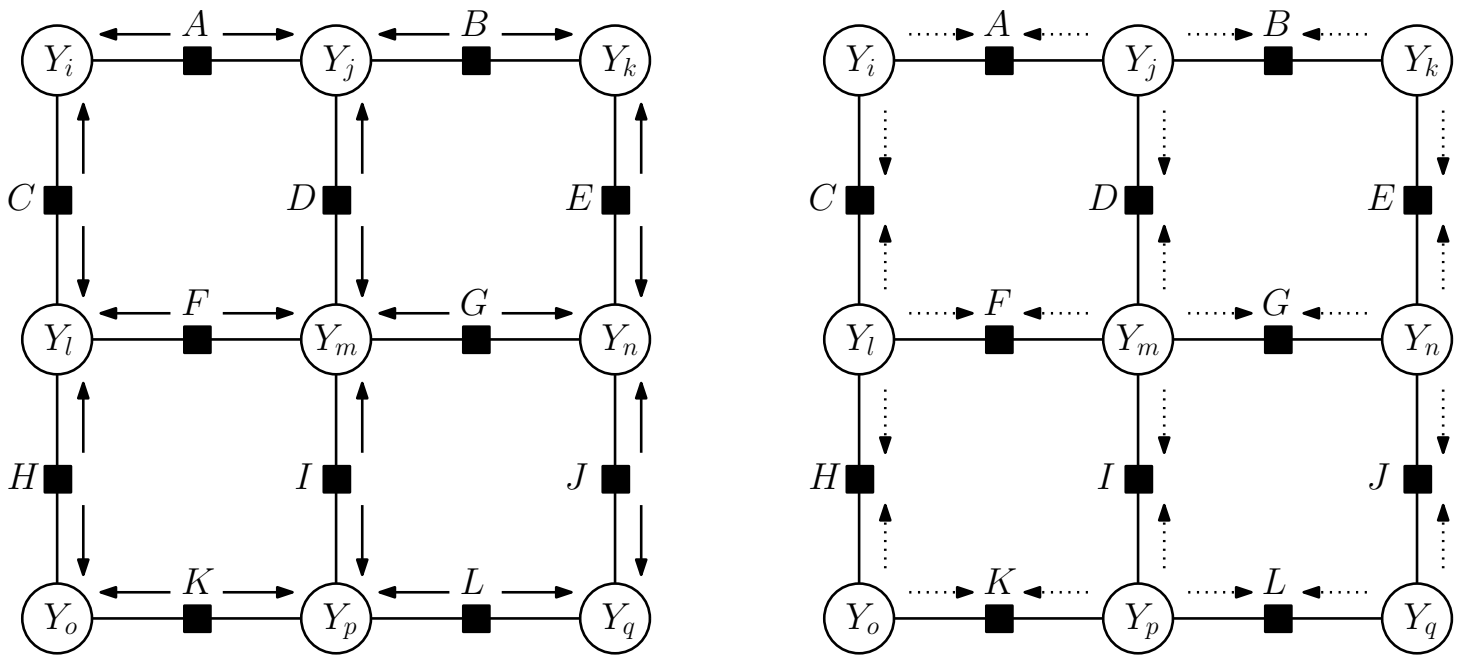
$$\tilde{Y} = (Y_i, Y_j, Y_m) \text{ with state space } \tilde{\mathcal{Y}} = \mathcal{Y}_i \times \mathcal{Y}_j \times \mathcal{Y}_m$$

General procedure: **junction tree algorithm**

Problem: exponentially growing state space $\rightarrow$ BP inefficient

Belief Propagation in Cyclic Graphs

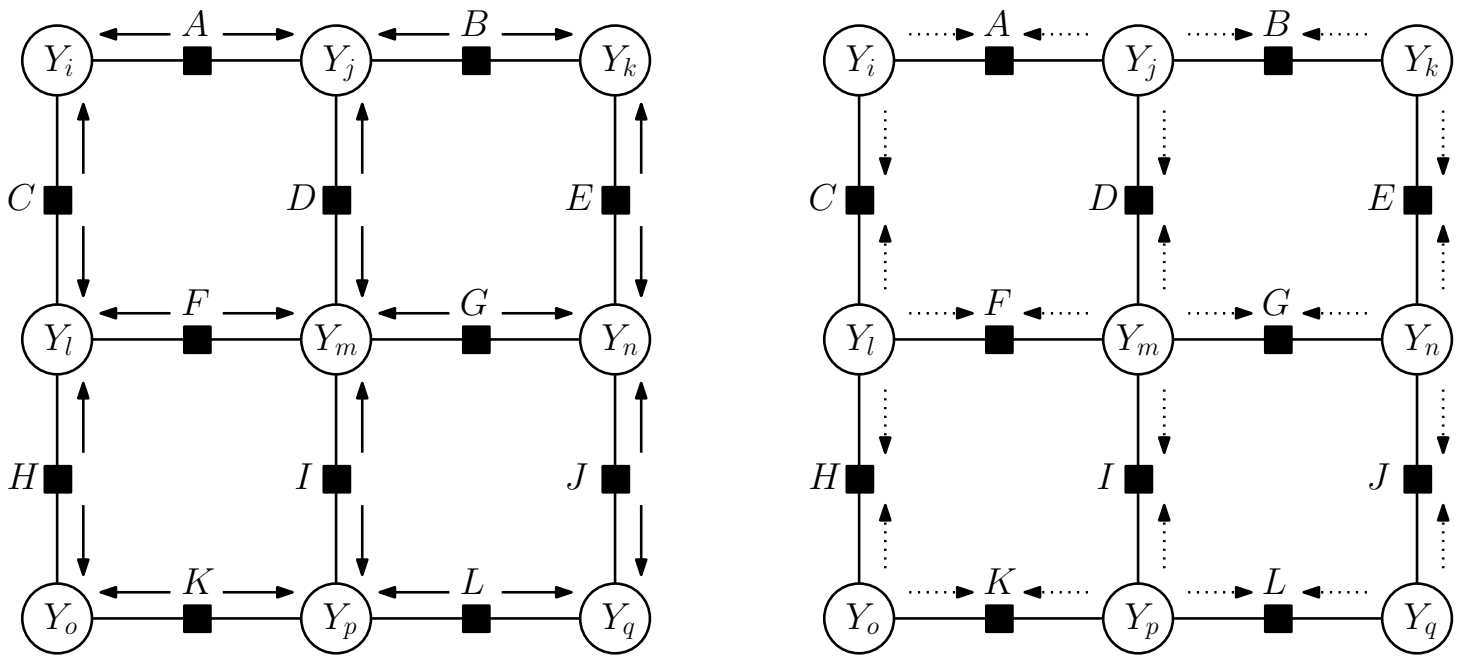
What if we do belief propagation even though the graphs has cycles?



Problem: There is no well-defined *leaf–to–root* order → where to start?

Belief Propagation in Cyclic Graphs

What if we do belief propagation even though the graphs has cycles?

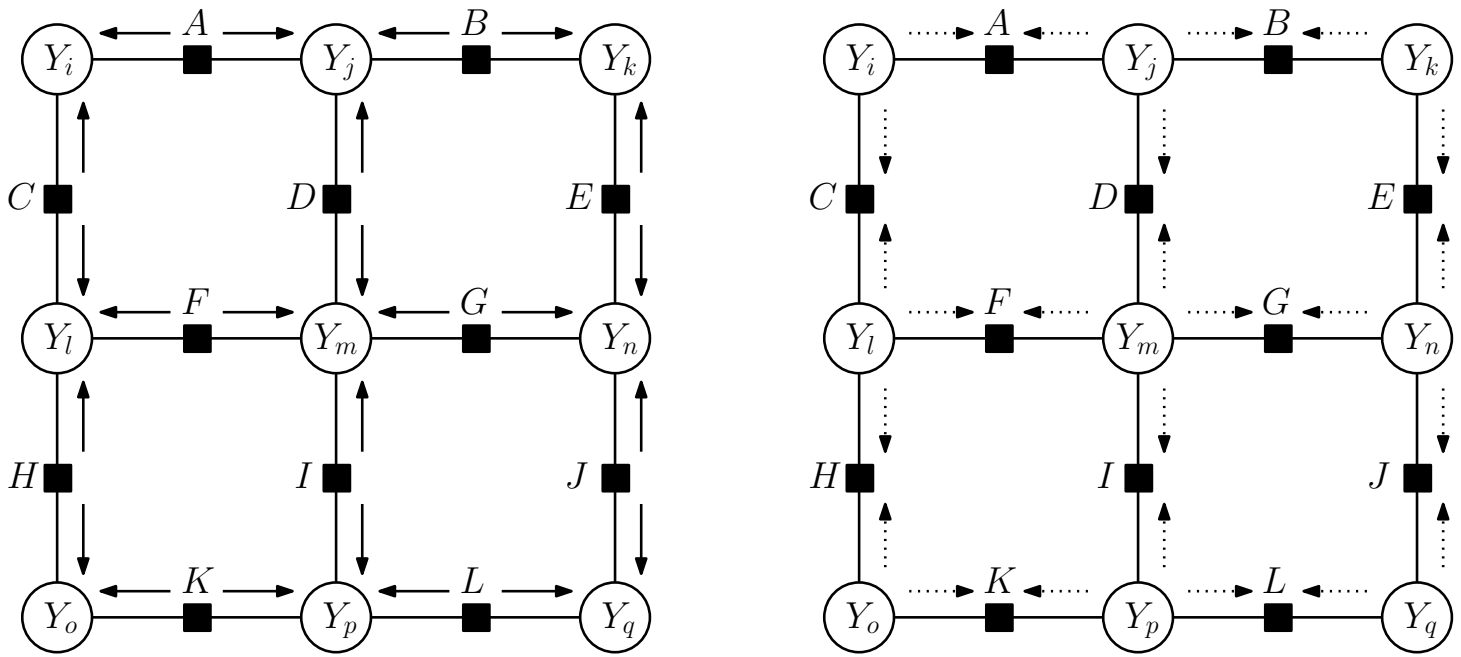


Problem: There is no well-defined *leaf–to–root* order $\rightarrow$ where to start?

Loopy Belief Propagation (LBP)

- initialize all messages as constant 1
- pass messages using rules of BP until a stop criterion

Belief Propagation in Cyclic Graphs



Problems:

- loopy BP might not converge (e.g. it can oscillate)
- even if it does, the computed probabilities are only *approximate*.

Several improved schemes exist, some even convergent (but approximate)

Exact inference in general cyclic graph is **#P-hard**.

Task: Compute marginals $p(y_F|x)$ for general $p(y|x)$

Idea: Rephrase as computing the *expected value of a function*:

$$\mathbb{E}_{y\sim p(y|x,w)}[h(x,y)],$$

for some (well-behaved) function $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.

For probabilistic inference, this step is easy.

$$h_{F,z}(x,y) := [\![y_F = z]\!],$$

Then

$$\begin{aligned}\mathbb{E}_{y\sim p(y|x,w)}[h_{F,z}(x,y)] &= \sum_{y\in\mathcal{Y}} p(y|x)[\![y_F = z]\!] \\ &= \sum_{y_F\in\mathcal{Y}_F} p(y_F|x)[\![y_F = z]\!] = p(y_F = z|x).\end{aligned}$$

Probabilistic Inference – Sampling / Markov-Chain Monte Carlo

Expectations can be computed/approximated by **sampling**:

- For fixed x, let $y^{(1)}, y^{(2)}, \ldots$ be i.i.d. samples from $p(y|x)$, then

$$\mathbb{E}_{y \sim p(y|x)}[h(x,y)] \approx \frac{1}{S} \sum_{s=1}^{S} h(x, y^{(s)}).$$

- The law of large numbers guarantees convergence for $S \to \infty$,
- For S independent samples, approximation error is $O(1/\sqrt{S})$, independent of the size of $\mathcal{Y}$.

Probabilistic Inference – Sampling / Markov-Chain Monte Carlo

Expectations can be computed/approximated by **sampling**:

- For fixed x, let $y^{(1)}, y^{(2)}, \ldots$ be i.i.d. samples from $p(y|x)$, then

$$\mathbb{E}_{y\sim p(y|x)}[h(x,y)] \approx \frac{1}{S}\sum_{s=1}^{S} h(x, y^{(s)}).$$

- The law of large numbers guarantees convergence for $S \to \infty$,
- For S independent samples, approximation error is $O(1/\sqrt{S})$, independent of the size of $\mathcal{Y}$.

Problem:

- Producing i.i.d. samples, $y^{(s)}$, from $p(y|x)$ is hard.

Solution:

- We can get away with a sequence of dependent samples

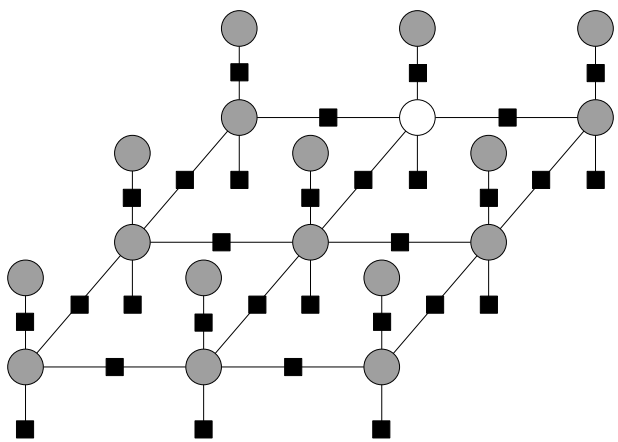
Monte-Carlo Markov Chain (MCMC) sampling

One example how to do MCMC sampling: **Gibbs sampler**

- Initialize $y^{(1)} = (y_1, \ldots, y_d)$ arbitrarily
- For $s = 1, \ldots, S$:
  1. Select an index i,
  2. Re-sample $y_i \sim p(y_i | y^{(s)}_{V \setminus \{i\}}, x)$.
  3. Output sample $y^{(s+1)} = (y^{(s)}_1, \ldots, y^{(s)}_{i-1},\ y_i,\ y^{(s)}_{i+1}, \ldots, y^{(s)}_d)$

$$p(y_i | y^{(s)}_{V \setminus \{i\}}, x) = \frac{p(y_i, y^{(s)}_{V \setminus \{i\}} | x)}{\sum_{y_i \in \mathcal{Y}_i} p(y_i, y^{(s)}_{V \setminus \{i\}} | x)}$$

$$= \frac{e^{-E(y_i, y^{(s)}, x)}}{\sum_{y_i \in \mathcal{Y}_i} e^{-E(y_i, y^{(s)}, x)}}$$



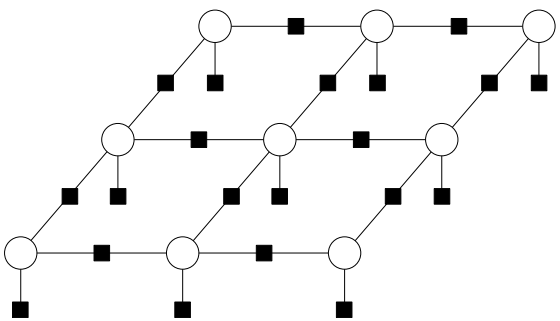
Task: Compute marginals $p(y_F|x)$ for general $p(y|x)$

Idea: Approximate $p(y|x)$ by simpler $q(y)$ and use marginals from that.

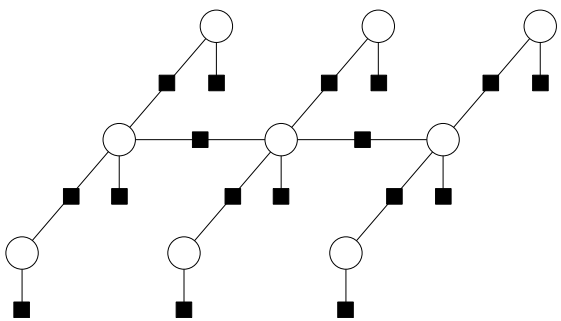
$$q^* = \underset{q \in \mathcal{Q}}{\mathbf{argmin}} \; D_{KL}(q(y) \| p(y|x))$$

$$p(y_F|x) \approx q^*(y_F)$$

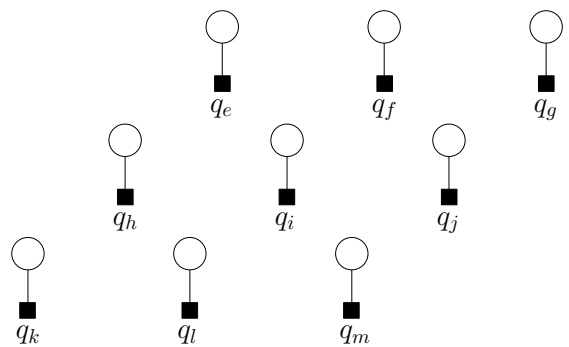
For example



original model



tree approximation



product of unary factors

Special case: **(Naive) Mean Field** for $p(y|x) = \frac{1}{Z(x)} e^{-E(y,x)}$

$$p(y|x) \approx q(y) = \prod_{i \in V} q_i(y_i)$$

No closed for expression for q^*, but optimality condition:

$$q_i^*(y_i) \propto e^{-\mathbb{E}_{y \setminus \{y_i\} \sim Q}\{E(y,x)\}}$$

$$\text{for} \quad Q(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n) = \prod_{j \neq i} q_j^*(y_j)$$

Iterative scheme:

- initialize q_i (e.g. uniform)
- repeat until convergence
  - for $i \in V$ in any order:
    - update q_i while keeping the others fixed

Probabilistic Inference – Summary

Task: compute (marginal) probabilities $p(y_F|x)$

Exact Probabilistic Inference

Only possible for certain models:

- trees/forests: sum-product belief propagation
- general graphs: junction chain algorithm (if tractable)

Approximate Probabilistic Inference

Many techniques with different properties and guarantees:

- loopy belief propagation
- MCMC sampling (e.g. Gibbs sampling)
- variational inference (e.g. mean field)
- ...

Best choice depends on model and requirements.

Training Conditional Random Fields in Practice

Gradient of the CRF training objective:

$$\nabla_w \mathcal{L}(w) = \lambda w + \sum_{n=1}^{N} \left[\phi(x^n, y^n) - \mathop{\mathbb{E}}_{y \sim p(y|x^n;w)} \phi(x^n, y) \right]$$

Feature function decompose over factors:

$$\phi(x, y) = \Big(\phi_F(x, y_F) \Big)_{F \in \mathcal{F}}$$

$$\mathop{\mathbb{E}}_{y_F \sim p(y_F|x;w)} \phi_F(x, y_F) = \sum_{y_F \in \mathcal{Y}_F} \underbrace{p(y_F|x;w)}_{\text{factor marginals}} \phi_F(x, y_F)$$

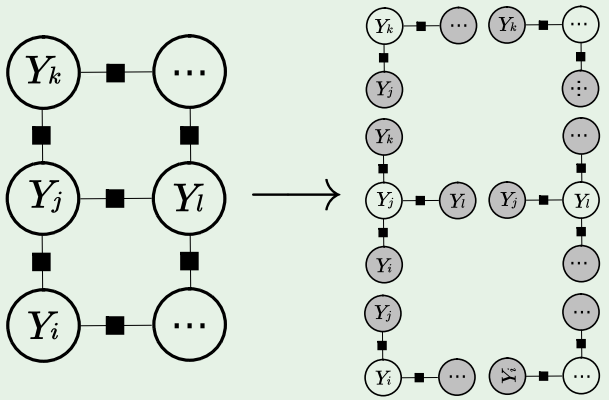
Problem: what if factor marginals $\mu_F = p(y_F|x;w)$ are intractable?

- approximate inference $\rightarrow$ approximate gradient
- convengence of gradient descent not guaranteed ☹

Alternative: optimize a simpler quantity instead of conditional likelihood

Pseudolikelihood [Besag, 1987]

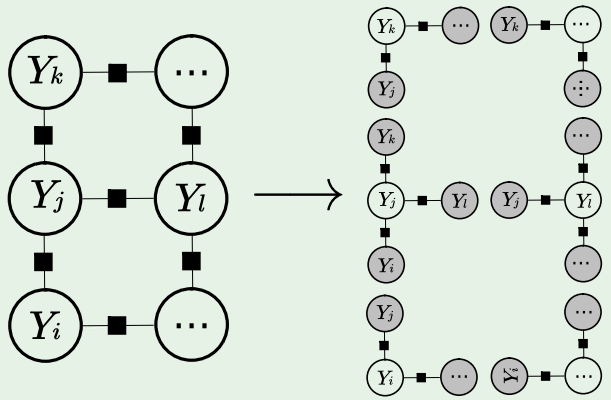
$$p(y|x) \approx \prod_{i \in V} p(y_i | y_{V \setminus \{i\}}, x)$$
$$= \prod_{i \in V} p(y_i | y_{N(i)}, x)$$



Alternative: optimize a simpler quantity instead of conditional likelihood

Pseudolikelihood [Besag, 1987]

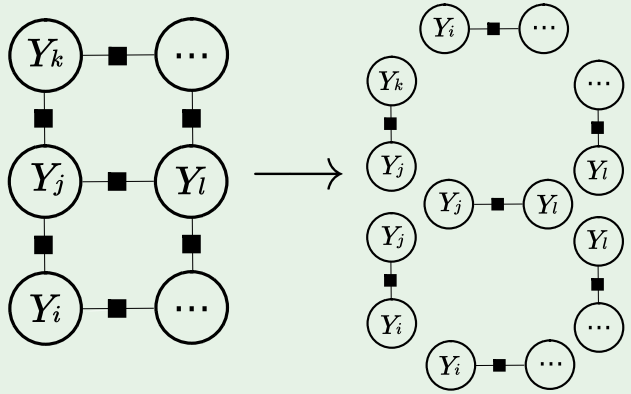
$$p(y|x) \approx \prod_{i \in V} p(y_i | y_{V \setminus \{i\}}, x)$$
$$= \prod_{i \in V} p(y_i | y_{N(i)}, x)$$



Piecewise Training [Sutton, McCallum, 2005]

$$p(y|x) \approx \prod_{F \in \mathcal{F}} p_F(y_F|x) \quad \text{for}$$

$$p_F(y_F|x) = \frac{1}{Z_F(x;w)} e^{-\langle w_F, \phi_F(y_F, x) \rangle}$$



Training with Approximate Likelihood – Pseudolikelihood (PL)

$$p(y|x) \approx p_{\mathsf{PL}}(y|x) = \prod_{i \in V} p(y_i | y_{N(i)}, x; w)$$

For training data $\{(x^1, y^1), \ldots, (x^N, y^N)\}$:

$$\begin{aligned}
\mathcal{L}_{PL}(w) &= \log \prod_{n=1}^{N} p_{\mathsf{PL}}(y^n | x^n; w) \\
&= \sum_{n=1}^{N} \sum_{i \in V} \log p(y_i^n | y_{N(i)}^n, x) \\
&= \sum_{n=1}^{N} \sum_{i \in V} \left[\langle w, \phi(y^n, x^n) \rangle - \log \sum_{\boldsymbol{k} \in \mathcal{Y}_i} e^{\langle w, \phi(y_1^n, \ldots, y_{i-1}^n, k, y_{i+1}^n, \ldots, y_{|V|}^n, x^n) \rangle} \right]
\end{aligned}$$

Training with Approximate Likelihood – Pseudolikelihood (PL)

$$p(y|x) \approx p_{\mathsf{PL}}(y|x) = \prod_{i \in V} p(y_i | y_{N(i)}, x; w)$$

For training data $\{(x^1, y^1), \ldots, (x^N, y^N)\}$:

$$\begin{aligned}
\mathcal{L}_{PL}(w) &= \log \prod_{n=1}^{N} p_{\mathsf{PL}}(y^n | x^n; w) \\
&= \sum_{n=1}^{N} \sum_{i \in V} \log p(y_i^n | y_{N(i)}^n, x) \\
&= \sum_{n=1}^{N} \sum_{i \in V} \left[\langle w, \phi(y^n, x^n) \rangle - \log \sum_{\boldsymbol{k} \in \mathcal{Y}_i} e^{\langle w, \phi(y_1^n, \ldots, y_{i-1}^n, \boldsymbol{k}, y_{i+1}^n, \ldots, y_{|V|}^n, x^n) \rangle} \right]
\end{aligned}$$

Partition functions sum only over one variable at a time $\rightarrow$ tractable

Training with Approximate Likelihood – Piecewise Training (PW)

$$p(y|x) \approx \prod_{F \in \mathcal{F}} p_F(y_F|x; w_F) \quad \text{for} \quad p_F(y_F|x) \propto e^{-\langle w_F, \phi_F(y_F, x)\rangle}$$

For training data $\{(x^1, y^1), \ldots, (x^N, y^N)\}$:

$$\mathcal{L}_{PW}(w) = \log \prod_{n=1}^{N} p_{\mathsf{PW}}(y_F^n|x^n; w) = \sum_{n=1}^{N} \sum_{F \in \mathcal{F}} \log p_F(y_F^n|x)$$

$$= \sum_{n=1}^{N} \sum_{F \in \mathcal{F}} \left[\langle w_F, \phi_F(y_F^n, x^n)\rangle - \log \sum_{\bar{y}_F \in \mathcal{Y}_F} e^{\langle w_F, \phi_F(\bar{y}_F, x^n)\rangle} \right]$$

Training with Approximate Likelihood – Piecewise Training (PW)

$$p(y|x) \approx \prod_{F \in \mathcal{F}} p_F(y_F|x; w_F) \quad \text{for} \quad p_F(y_F|x) \propto e^{-\langle w_F, \phi_F(y_F, x)\rangle}$$

For training data $\{(x^1, y^1), \ldots, (x^N, y^N)\}$:

$$\mathcal{L}_{PW}(w) = \log \prod_{n=1}^{N} p_{\mathsf{PW}}(y_F^n|x^n; w) = \sum_{n=1}^{N} \sum_{F \in \mathcal{F}} \log p_F(y_F^n|x)$$

$$= \sum_{n=1}^{N} \sum_{F \in \mathcal{F}} \left[\langle w_F, \phi_F(y_F^n, x^n)\rangle - \log \sum_{\bar{y}_F \in \mathcal{Y}_F} e^{\langle w_F, \phi_F(\bar{y}_F, x^n)\rangle} \right]$$

Partition functions sum over $|F|$ variables at a time $\rightarrow$ usually tractable

Optimization decomposes into a sum over the w_F $\rightarrow$ easy to parallelize

Training with Approximate Likelihood – Piecewise Training (PW)

$$p(y|x) \approx \prod_{F \in \mathcal{F}} p_F(y_F|x; w_F) \quad \text{for} \quad p_F(y_F|x) \propto e^{-\langle w_F, \phi_F(y_F, x)\rangle}$$

For training data $\{(x^1, y^1), \ldots, (x^N, y^N)\}$:

$$\mathcal{L}_{PW}(w) = \log \prod_{n=1}^{N} p_{\mathsf{PW}}(y_F^n|x^n; w) = \sum_{n=1}^{N} \sum_{F \in \mathcal{F}} \log p_F(y_F^n|x)$$

$$= \sum_{F \in \mathcal{F}} \sum_{n=1}^{N} \left[\langle w_F, \phi_F(y_F^n, x^n)\rangle - \log \sum_{\bar{y}_F \in \mathcal{Y}_F} e^{\langle w_F, \phi_F(\bar{y}_F, x^n)\rangle} \right]$$

Partition functions sum over $|F|$ variables at a time $\rightarrow$ usually tractable

Optimization decomposes into a sum over the w_F $\rightarrow$ easy to parallelize