Part 3: Probabilistic Inference in Graphical Models

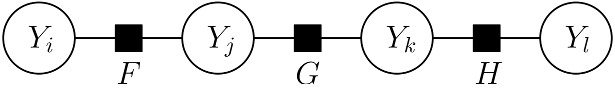
Sebastian Nowozin and Christoph H. Lampert

Providence, 21st June 2012

Belief Propagation

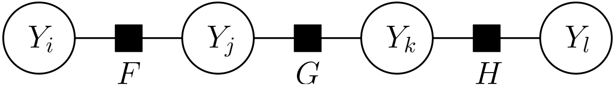
- "Message-passing" algorithm
- Exact and optimal for tree-structured graphs
- Approximate for cyclic graphs

Example: Inference on Chains



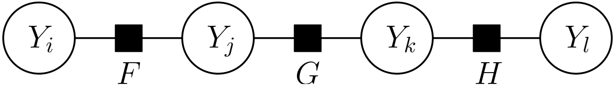
$$
\begin{aligned}
Z &= \sum_{y \in \mathcal{Y}} \exp(-E(y)) \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} \exp(-E(y_i, y_j, y_k, y_l)) \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k}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

Example: Inference on Chains



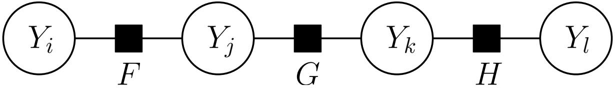
$$
\begin{aligned}
Z &= \sum_{y \in \mathcal{Y}} \exp(-E(y)) \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} \exp(-E(y_i, y_j, y_k, y_l)) \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k}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

Example: Inference on Chains



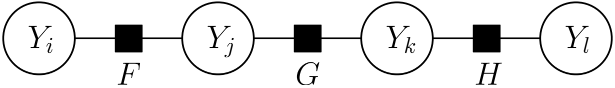
$$
\begin{aligned}
Z &= \sum_{y \in \mathcal{Y}} \exp(-E(y)) \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} \exp(-E(y_i, y_j, y_k, y_l)) \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k}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

Example: Inference on Chains



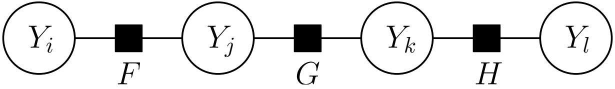
$$
\begin{aligned}
Z &= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} \exp(-(E_F(y_i, y_j) + E_G(y_j, y_k) + E_H(y_k, y_l))) \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in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

Example: Inference on Chains



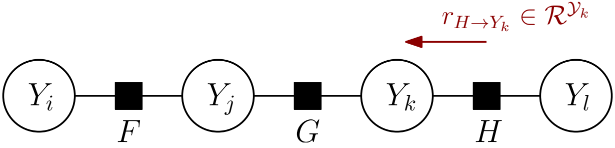
$$
\begin{aligned}
Z &= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} \exp(-(E_F(y_i, y_j) + E_G(y_j, y_k) + E_H(y_k, y_l))) \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in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

Example: Inference on Chains



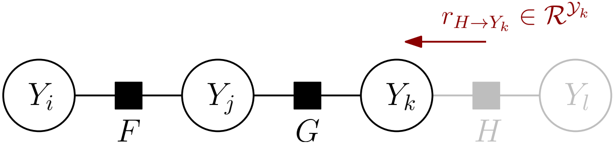
$$
\begin{aligned}
Z &= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in \mathcal{Y}_l} \exp(-(E_F(y_i, y_j) + E_G(y_j, y_k) + E_H(y_k, y_l))) \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \sum_{y_k \in \mathcal{Y}_k} \sum_{y_l \in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

Example: Inference on Chains



$$
\begin{aligned}
Z &= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \exp(-E_F(y_i, y_j)) \sum_{y_k \in \mathcal{Y}_k} \exp(-E_G(y_j, y_k)) \underbrace{\sum_{y_l \in \mathcal{Y}_l} \exp(-E_H(y_k, y_l))}_{r_{H \to Y_k}(y_k)} \\
&= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \exp(-E_F(y_i, y_j)) \sum_{y_k \in \mathcal{Y}_i} \exp(-E_G(y_j, y_k)) r_{H \to Y_k}(y_k)
\end{aligned}
$$

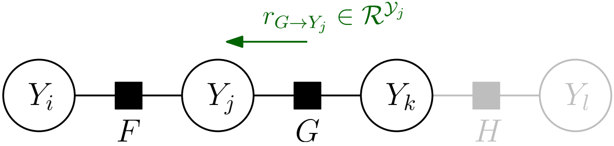
Example: Inference on Chains



$$Z = \sum_{y_i\in\mathcal{Y}_i}\sum_{y_j\in\mathcal{Y}_j}\exp(-E_F(y_i,y_j))\sum_{y_k\in\mathcal{Y}_k}\exp(-E_G(y_j,y_k))\underbrace{\sum_{y_l\in\mathcal{Y}_l}\exp(-E_H(y_k,y_l))}_{r_{H\to Y_k}(y_k)}$$

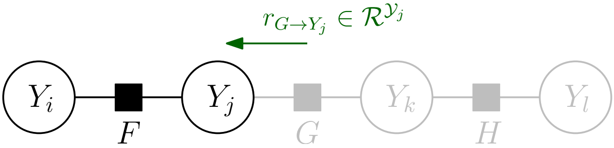
$$= \sum_{y_i\in\mathcal{Y}_i}\sum_{y_j\in\mathcal{Y}_j}\exp(-E_F(y_i,y_j))\sum_{y_k\in\mathcal{Y}_i}\exp(-E_G(y_j,y_k))r_{H\to Y_k}(y_k)$$

Example: Inference on Chains



$$\begin{aligned} Z &= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \exp(-E_F(y_i, y_j)) \underbrace{\sum_{y_k \in \mathcal{Y}_k} \exp(-E_G(y_j, y_k)) r_{H\to Y_k}(y_k)}_{r_{G\to Y_j}(y_j)} \\ &= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \exp(-E_F(y_i, y_j)) r_{G\to Y_j}(y_j) \end{aligned}$$

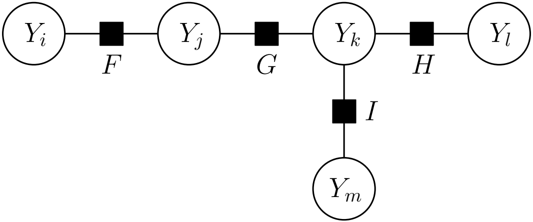
Example: Inference on Chains



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

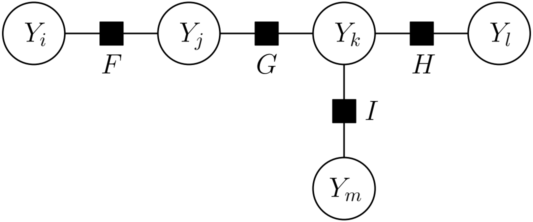
$$= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \exp(-E_F(y_i, y_j)) r_{G \to Y_j}(y_j)$$

Example: Inference on Trees



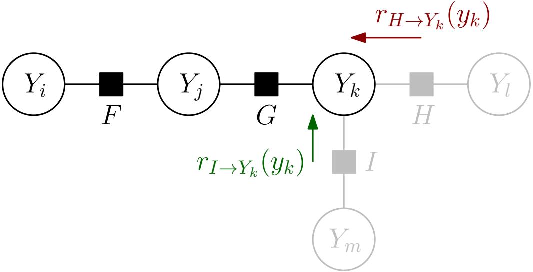
$$\begin{aligned} Z &= \sum_{y \in \mathcal{Y}} \exp(-E(y)) \\ &= \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_i} \sum_{y_k \in \mathcal{Y}_i} \sum_{y_l \in \mathcal{Y}_i} \sum_{y_m \in \mathcal{Y}_m} \exp(-(E_F(y_i, y_j) + \cdots + E_I(y_k, y_m))) \end{aligned}$$

Example: Inference on Trees



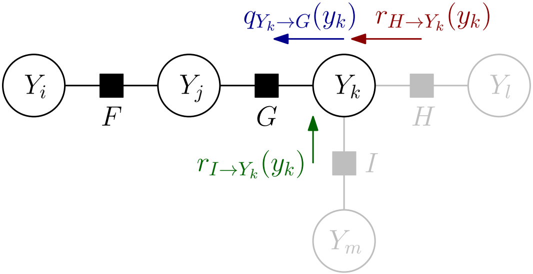
$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \exp(-E_F(y_i, y_j)) \sum_{y_k \in \mathcal{Y}_k} \exp(-E_G(y_j, y_k)) \cdot$$
$$\left(\underbrace{\left(\sum_{y_l \in \mathcal{Y}_l} \exp(-E_H(y_k, y_l)) \right)}_{r_{H \to Y_k}(y_k)} \cdot \underbrace{\left(\sum_{y_m \in \mathcal{Y}_m}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

Example: Inference on Trees



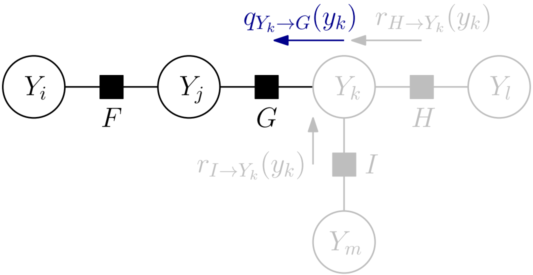
$$Z = \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} \exp(-E_F(y_i, y_j)) \sum_{y_k \in \mathcal{Y}_k} \exp(-E_G(y_j, y_k)) \cdot$$
$$(r_{H \to Y_k}(y_k) \cdot r_{I \to Y_k}(y_k))$$

Example: Inference on Trees



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

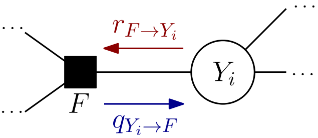
Example: Inference on Trees



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

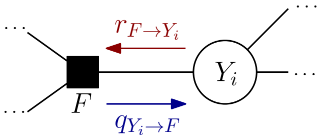
Factor Graph Sum-Product Algorithm

- "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 update messages
- After convergence: Z and μ_F can be obtained from the messages



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 update messages
- After convergence: Z and μ_F can be obtained from the messages



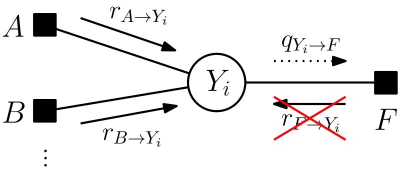
Sum-Product: Variable-to-Factor message

- Set of factors adjacent to variable i

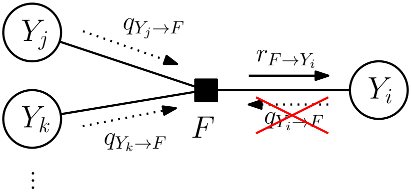
$$M(i) = \{F \in \mathcal{F} : (i, F) \in \mathcal{E}\}$$

- Variable-to-factor message

$$q_{Y_i \to F}(y_i) = \prod_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i)$$



Sum-Product: Factor-to-Variable message



- Factor-to-variable message

$$r_{F \to Y_i}(y_i) = \sum_{\substack{y'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \left(\exp\left(-E_F(y'_F)\right) \prod_{j \in N(F) \setminus \{i\}} q_{Y_j \to F}(y'_j) \right)$$

Message Scheduling

$$
\begin{aligned}
q_{Y_i \to F}(y_i) &= \prod_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i) \\
r_{F \to Y_i}(y_i) &= \sum_{\substack{y'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \left(\exp\left(-E_F(y'_F)\right) \prod_{j \in N(F) \setminus \{i\}} q_{Y_j \to F}(y'_j) \right)
\end{aligned}
$$

- Problem: message updates depend on each other
- No dependencies if product is empty ($= 1$)
- For tree-structured graphs we can resolve all dependencies

Message Scheduling

$$q_{Y_i \to F}(y_i) = \prod_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i)$$

$$r_{F \to Y_i}(y_i) = \sum_{\substack{y'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \left(\exp\left(-E_F(y'_F)\right) \prod_{j \in N(F) \setminus \{i\}} q_{Y_j \to F}(y'_j) \right)$$

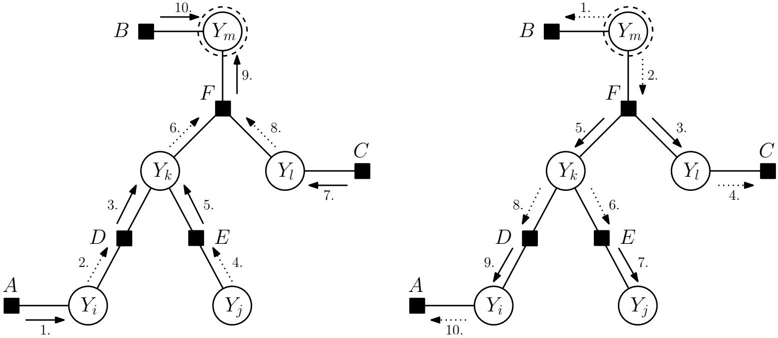
- Problem: message updates depend on each other
- No dependencies if product is empty ($= 1$)
- For tree-structured graphs we can resolve all dependencies

Message Scheduling

$$
\begin{aligned}
q_{Y_i \to F}(y_i) &= \prod_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i) \\
r_{F \to Y_i}(y_i) &= \sum_{\substack{y'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \left(\exp\left(-E_F(y'_F)\right) \prod_{j \in N(F) \setminus \{i\}} q_{Y_j \to F}(y'_j) \right)
\end{aligned}
$$

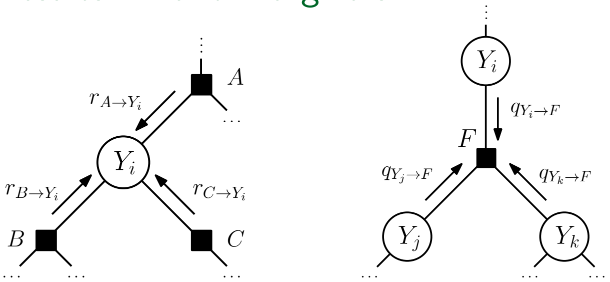
- Problem: message updates depend on each other
- No dependencies if product is empty ($= 1$)
- For tree-structured graphs we can resolve all dependencies

Message Scheduling: Trees



1. Select one variable node as *tree root*
2. Compute leaf-to-root messages
3. Compute root-to-leaf messages

Inference Results: Z and marginals



- Partition function, evaluated at root

$$Z = \sum_{y_r \in \mathcal{Y}_r} \prod_{F \in M(r)} r_{F\to Y_r}(y_r)$$

- Marginal distributions, for each factor

$$\mu_F(y_F) = p(Y_F = y_F) = \frac{1}{Z} \exp\left(-E_F(y_F)\right) \prod_{i \in N(F)} q_{Y_i \to F}(y_i)$$

Max-Product/Max-Sum Algorithm

- Belief Propagation for MAP inference

$$y^* = \operatorname*{argmax}_{y \in \mathcal{Y}} p(Y = y | x, w)$$

- Exact for trees
- For cyclic graphs: not as well understood as sum-product algorithm

$$\begin{aligned} q_{Y_i \to F}(y_i) &= \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i) \\ r_{F \to Y_i}(y_i) &= \max_{\substack{y'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \left(-E_F(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{Y_j \to F}(y'_j) \right) \end{aligned}$$

Sum-Product/Max-Sum comparison

Sum-Product

$$q_{Y_i \to F}(y_i) = \prod_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i)$$

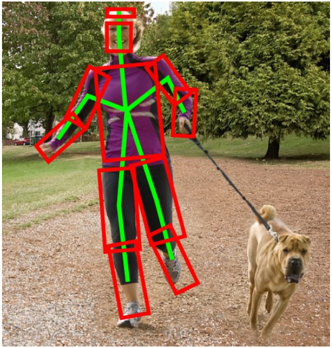
$$r_{F \to Y_i}(y_i) = \sum_{\substack{y'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \left(\exp\left(-E_F(y'_F)\right) \prod_{j \in N(F) \setminus \{i\}} q_{Y_j \to F}(y'_j) \right)$$

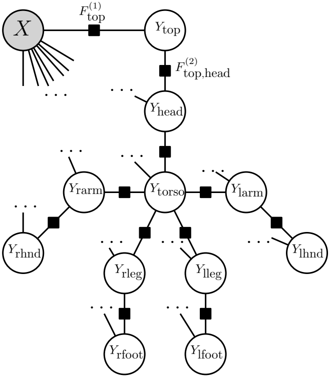
Max-Sum

$$q_{Y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i)$$

$$r_{F \to Y_i}(y_i) = \max_{\substack{y'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \left(-E_F(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{Y_j \to F}(y'_j) \right)$$

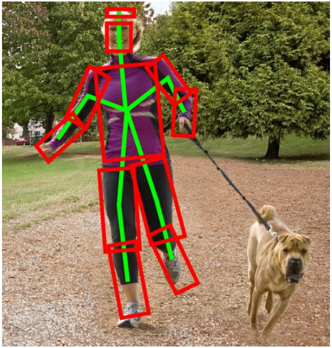
Example: Pictorial Structures

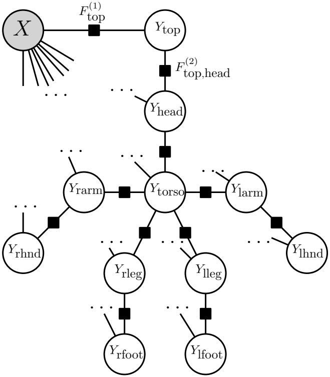




- Tree-structured model for articulated pose (Felzenszwalb and Huttenlocher, 2000), (Fischler and Elschlager, 1973)
- Body-part variables, states: discretized tuple (x, y, s, θ)
- (x, y) position, s scale, and θ rotation

Example: Pictorial Structures





- Tree-structured model for articulated pose (Felzenszwalb and Huttenlocher, 2000), (Fischler and Elschlager, 1973)
- Body-part variables, states: discretized tuple (x, y, s, θ)
- (x, y) position, s scale, and θ rotation

Example: Pictorial Structures (cont)

$$r_{F \to Y_i}(y_i) \quad = \max_{\substack{(y_i', y_j') \in \mathcal{Y}_i \times \mathcal{Y}_j, \\ y_i' = y_i}} \left(-E_F(y_i', y_j') + \sum_{j \in N(F) \setminus \{i\}} q_{Y_j \to F}(y_j') \right) \quad (1)$$

- Because $\mathcal{Y}_i$ is large ($\approx 500k$), $\mathcal{Y}_i \times \mathcal{Y}_j$ is too big
- (Felzenszwalb and Huttenlocher, 2000) use special form for $E_F(y_i, y_j)$ so that (1) is computable in $O(|\mathcal{Y}_i|)$

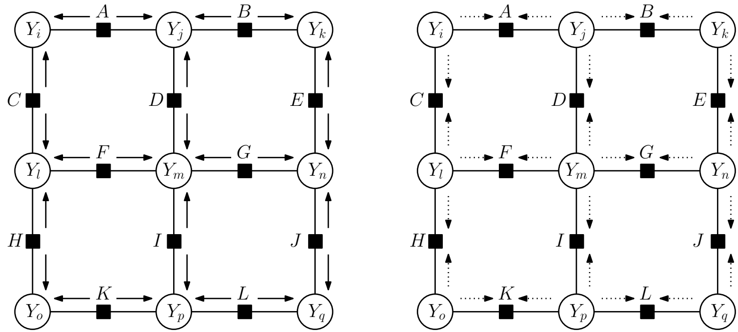
Loopy Belief Propagation

- Key difference: no schedule that removes dependencies
- But: message computation is still well defined
- Therefore, classic loopy belief propagation (Pearl, 1988)
  1. Initialize message vectors to 1 (sum-product) or 0 (max-sum)
  2. Update messages, hoping for convergence
  3. Upon convergence, treat beliefs μ_F as approximate marginals
- Different messaging schedules (synchronous/asynchronous, static/dynamic)
- Improvements: generalized BP (Yedidia et al., 2001), convergent BP (Heskes, 2006)

Loopy Belief Propagation

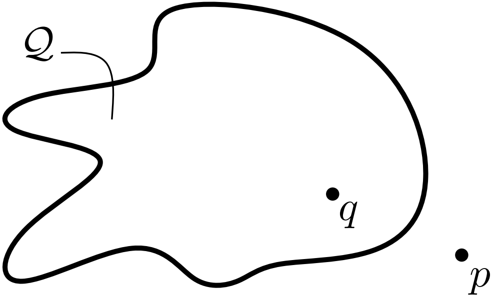
- Key difference: no schedule that removes dependencies
- But: message computation is still well defined
- Therefore, classic loopy belief propagation (Pearl, 1988)
  1. Initialize message vectors to 1 (sum-product) or 0 (max-sum)
  2. Update messages, hoping for convergence
  3. Upon convergence, treat beliefs μ_F as approximate marginals
- Different messaging schedules (synchronous/asynchronous, static/dynamic)
- Improvements: generalized BP (Yedidia et al., 2001), convergent BP (Heskes, 2006)

Synchronous Iteration



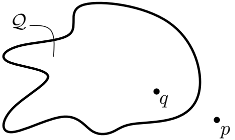
Mean field methods

- Mean field methods (Jordan et al., 1999), (Xing et al., 2003)
- Distribution $p(y|x, w)$, inference intractable
- Approximate distribution $q(y)$
- Tractable family $\mathcal{Q}$



Mean field methods (cont)

$$q^* = \operatorname*{argmin}_{q \in \mathcal{Q}} D_{KL}(q(y) \| p(y|x,w))$$

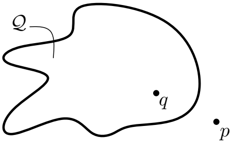


$$\begin{aligned} & D_{KL}(q(y) \| p(y|x,w)) \\ = & \sum_{y \in \mathcal{Y}} q(y) \log \frac{q(y)}{p(y|x,w)} \\ = & \sum_{y \in \mathcal{Y}} q(y) \log q(y) - \sum_{y \in \mathcal{Y}} q(y) \log p(y|x,w) \\ = & -H(q) + \sum_{F \in \mathcal{F}} \sum_{y_F \in \mathcal{Y}_F} \mu_{F,y_F}(q) E_F(y_F; x_F, w) + \log Z(x,w), \end{aligned}$$

where $H(q)$ is the *entropy* of q and μ_{F,y_F} are the marginals of q. (The form of μ depends on $\mathcal{Q}$.)

Mean field methods (cont)

$$q^* = \operatorname*{argmin}_{q \in \mathcal{Q}} D_{KL}(q(y) \| p(y|x,w))$$



$$\begin{aligned}
& D_{KL}(q(y) \| p(y|x,w)) \\
= & \sum_{y \in \mathcal{Y}} q(y) \log \frac{q(y)}{p(y|x,w)} \\
= & \sum_{y \in \mathcal{Y}} q(y) \log q(y) - \sum_{y \in \mathcal{Y}} q(y) \log p(y|x,w) \\
= & -H(q) + \sum_{F \in \mathcal{F}} \sum_{y_F \in \mathcal{Y}_F} \mu_{F,y_F}(q) E_F(y_F; x_F, w) + \log Z(x,w),
\end{aligned}$$

where $H(q)$ is the *entropy* of q and μ_{F,y_F} are the marginals of q. (The form of μ depends on $\mathcal{Q}$.)

Mean field methods (cont)

$$q^* = \operatorname*{argmin}_{q \in \mathcal{Q}} D_{KL}(q(y) \| p(y|x, w))$$

- When $\mathcal{Q}$ is rich: q^* is close to p
- Marginals of q^* approximate marginals of p
- *Gibbs inequality*

$$D_{KL}(q(y) \| p(y|x, w)) \geq 0$$

- Therefore, we have a lower bound

$$\log Z(x, w) \geq H(q) - \sum_{F \in \mathcal{F}} \sum_{y_F \in \mathcal{Y}_F} \mu_{F, y_F}(q) E_F(y_F; x_F, w).$$

Mean field methods (cont)

$$q^* = \operatorname*{argmin}_{q \in \mathcal{Q}} D_{KL}(q(y) \| p(y|x, w))$$

- When $\mathcal{Q}$ is rich: q^* is close to p
- Marginals of q^* approximate marginals of p
- *Gibbs inequality*

$$D_{KL}(q(y) \| p(y|x, w)) \geq 0$$

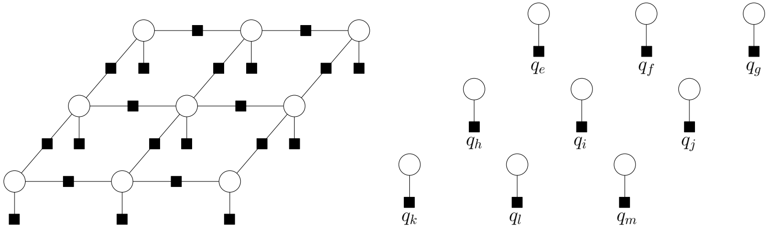
- Therefore, we have a lower bound

$$\log Z(x, w) \geq H(q) - \sum_{F \in \mathcal{F}} \sum_{y_F \in \mathcal{Y}_F} \mu_{F, y_F}(q) E_F(y_F; x_F, w).$$

Naive Mean Field

- Set $\mathcal{Q}$ all distributions of the form

$$q(y) = \prod_{i \in V} q_i(y_i).$$



Naive Mean Field

- Set $\mathcal{Q}$ all distributions of the form

$$q(y) = \prod_{i \in V} q_i(y_i).$$

- Marginals μ_{F,y_F} take the form

$$\mu_{F,y_F}(q) = \prod_{i \in N(F)} q_i(y_i).$$

- Entropy $H(q)$ decomposes

$$H(q) = \sum_{i \in V} H_i(q_i) = -\sum_{i \in V} \sum_{y_i \in \mathcal{Y}_i} q_i(y_i) \log q_i(y_i).$$

Naive Mean Field (cont)

Putting it together,

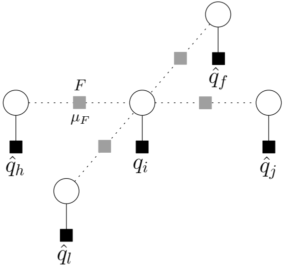
$$
\begin{aligned}
& \operatorname*{argmin}_{q \in \mathcal{Q}} \quad D_{KL}(q(y) \| p(y|x,w)) \\
= & \operatorname*{argmax}_{q \in \mathcal{Q}} \quad H(q) - \sum_{F \in \mathcal{F}} \sum_{y_F \in \mathcal{Y}_F} \mu_{F,y_F}(q) E_F(y_F; x_F, w) - \log Z(x,w) \\
= & \operatorname*{argmax}_{q \in \mathcal{Q}} \quad H(q) - \sum_{F \in \mathcal{F}} \sum_{y_F \in \mathcal{Y}_F} \mu_{F,y_F}(q) E_F(y_F; x_F, w) \\
= & \operatorname*{argmax}_{q \in \mathcal{Q}} \quad \Bigg[- \sum_{i \in V} \sum_{y_i \in \mathcal{Y}_i} q_i(y_i) \log q_i(y_i) \\
& \qquad - \sum_{F \in \mathcal{F}} \sum_{y_F \in \mathcal{Y}_F} \Bigg(\prod_{i \in N(F)} q_i(y_i) \Bigg) E_F(y_F; x_F, w) \Bigg].
\end{aligned}
$$

Optimizing over $\mathcal{Q}$ is optimizing over $q_i \in \Delta_i$, the probability simplices.

Naive Mean Field (cont)

$$\operatorname*{argmax}_{q \in \mathcal{Q}} \Bigg[-\sum_{i \in V} \sum_{y_i \in \mathcal{Y}_i} q_i(y_i) \log q_i(y_i) - \sum_{F \in \mathcal{F}} \sum_{y_F \in \mathcal{Y}_F} \Big(\prod_{i \in N(F)} q_i(y_i) \Big) E_F(y_F; x_F, w) \Bigg].$$

- Non-concave maximization problem $\rightarrow$ hard. (For general E_F and pairwise or higher-order factors.)
- Block coordinate ascent: closed-form update for each q_i



Naive Mean Field (cont)

Closed form update for q_i:

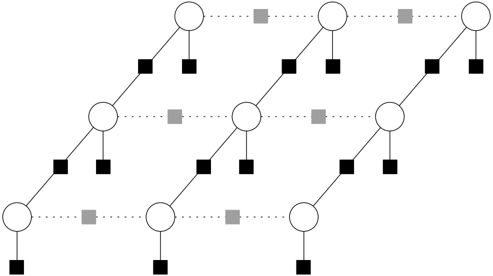
$$q_i^*(y_i) = \exp\left(- \sum_{\substack{F \in \mathcal{F}, \\ i \in N(F)}} \sum_{\substack{y_F \in \mathcal{Y}_F, \\ [y_F]_i = y_i}} \left(\prod_{j \in N(F) \setminus \{i\}} \hat{q}_j(y_j) \right) E_F(y_F; x_F, w) - \lambda \right)$$

$$\lambda = \log \left(\sum_{y_i \in \mathcal{Y}_i} \exp \left(- \sum_{\substack{F \in \mathcal{F}, \\ i \in N(F)}} \sum_{\substack{y_F \in \mathcal{Y}_F, \\ [y_F]_i = y_i}} \left(\prod_{j \in N(F) \setminus \{i\}} \hat{q}_j(y_j) \right) E_F(y_F; x_F, w) \right) \right)$$

- Look scary, but very easy to implement
- Interpretation in terms of mean field expectation, see (Koller and Friedman, 2009)

Structured Mean Field

- Naive mean field approximation can be poor
- Structured mean field (Saul and Jordan, 1995) uses factorial approximations with larger tractable subgraphs
- Block coordinate ascent: optimizing an entire subgraph using exact probabilistic inference on trees



Sampling

Probabilistic inference and related tasks require the computation of

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

where $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is an arbitary but well-behaved function.

- Inference: $h_{F,z_F}(x,y) = I(y_F = z_F)$,

$$\mathbb{E}_{y \sim p(y|x,w)}[h_{F,z_F}(x,y)] = p(y_F = z_F | x, w),$$

  the marginal probability of factor F taking state z_F.
- Parameter estimation: *feature map* $h(x,y) = \phi(x,y)$, $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$,

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

  "expected *sufficient statistics* under the model distribution".

Sampling

Probabilistic inference and related tasks require the computation of

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

where $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is an arbitary but well-behaved function.

- Inference: $h_{F,z_F}(x,y) = I(y_F = z_F)$,

  $$\mathbb{E}_{y \sim p(y|x,w)}[h_{F,z_F}(x,y)] = p(y_F = z_F|x,w),$$

  the marginal probability of factor F taking state z_F.
- Parameter estimation: *feature map* $h(x,y) = \phi(x,y)$, $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$,

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

  "expected *sufficient statistics* under the model distribution".

Sampling

Probabilistic inference and related tasks require the computation of

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

where $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is an arbitary but well-behaved function.

- Inference: $h_{F,z_F}(x,y) = I(y_F = z_F)$,

$$\mathbb{E}_{y \sim p(y|x,w)}[h_{F,z_F}(x,y)] = p(y_F = z_F | x, w),$$

  the marginal probability of factor F taking state z_F.
- Parameter estimation: *feature map* $h(x,y) = \phi(x,y)$, $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$,

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

  "expected *sufficient statistics* under the model distribution".

Monte Carlo

- Sample approximation from $y^{(1)}, y^{(2)}, \ldots$

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

- When the expectation exists, then 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 dimension d.

Problem

- Producing exact samples $y^{(s)}$ from $p(y|x)$ is *hard*

Monte Carlo

- Sample approximation from $y^{(1)}, y^{(2)}, \ldots$

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

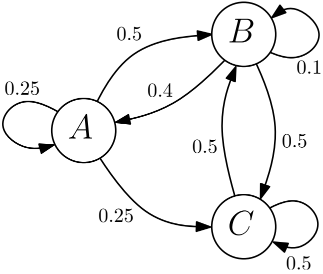
- When the expectation exists, then the *law of large numbers* guarantees convergence for $S \to \infty$,
- For S independent samples, approximation error is $O(1/\sqrt{S})$, *independent* of the dimension d.

Problem

- Producing exact samples $y^{(s)}$ from $p(y|x)$ is *hard*

Markov Chain Monte Carlo (MCMC)

- *Markov chain* with $p(y|x)$ as stationary distribution
- Here: $\mathcal{Y} = \{A, B, C\}$
- Here: $p(y) = (0.1905, 0.3571, 0.4524)$

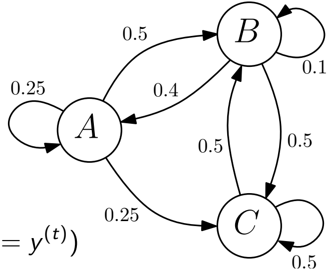


Markov Chains

Definition (Finite Markov chain)

Given a finite set $\mathcal{Y}$ and a matrix $P \in \mathbb{R}^{\mathcal{Y} \times \mathcal{Y}}$, then a random process $(Z_1, Z_2, \dots)$ with Z_t taking values from $\mathcal{Y}$ is a *Markov chain with transition matrix P*, if

$$
\begin{aligned}
& p(Z_{t+1} = y^{(j)} | Z_1 = y^{(1)}, Z_2 = y^{(2)}, \dots, Z_t = y^{(t)}) \\
= \; & p(Z_{t+1} = y^{(j)} | Z_t = y^{(t)}) \\
= \; & P_{y^{(t)}, y^{(j)}}
\end{aligned}
$$



MCMC Simulation (1)

Steps

1. Construct a Markov chain with stationary distribution $p(y|x, w)$
2. Start at $y^{(0)}$
3. Perform random walk according to Markov chain
4. After sufficient number S of steps, stop and treat $y^{(S)}$ as sample from $p(y|x, w)$

- Justified by *ergodic theorem*.
- In practise: discard a fixed number of initial samples ("burn-in phase") to forget starting point
- In practise: afterwards, use between 100 to $100,000$ samples

MCMC Simulation (1)

Steps

1. Construct a Markov chain with stationary distribution $p(y|x, w)$
2. Start at $y^{(0)}$
3. Perform random walk according to Markov chain
4. After sufficient number S of steps, stop and treat $y^{(S)}$ as sample from $p(y|x, w)$

- Justified by *ergodic theorem*.
- In practise: discard a fixed number of initial samples ("burn-in phase") to forget starting point
- In practise: afterwards, use between 100 to 100,000 samples

MCMC Simulation (1)

Steps

1. Construct a Markov chain with stationary distribution $p(y|x, w)$
2. Start at $y^{(0)}$
3. Perform random walk according to Markov chain
4. After each step, output $y^{(i)}$ as sample from $p(y|x, w)$

- Justified by *ergodic theorem*.
- In practise: discard a fixed number of initial samples ("burn-in phase") to forget starting point
- In practise: afterwards, use between 100 to $100,000$ samples

MCMC Simulation (1)

Steps

1. Construct a Markov chain with stationary distribution $p(y|x, w)$
2. Start at $y^{(0)}$
3. Perform random walk according to Markov chain
4. After each step, output $y^{(i)}$ as sample from $p(y|x, w)$

- Justified by *ergodic theorem*.
- In practise: discard a fixed number of initial samples ("burn-in phase") to forget starting point
- In practise: afterwards, use between 100 to $100,000$ samples

Gibbs sampler

How to construct a suitable Markov chain?

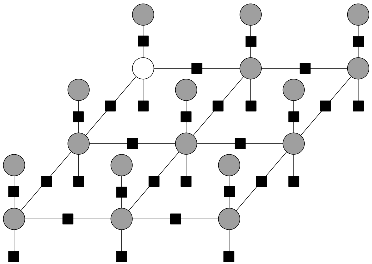
- *Metropolis-Hastings chain*, almost always possible
- Special case: *Gibbs sampler* (Geman and Geman, 1984)
  1. Select a variable y_i,
  2. Sample $y_i \sim p(y_i | y_{V \setminus \{i\}}, x)$.

Gibbs sampler

How to construct a suitable Markov chain?

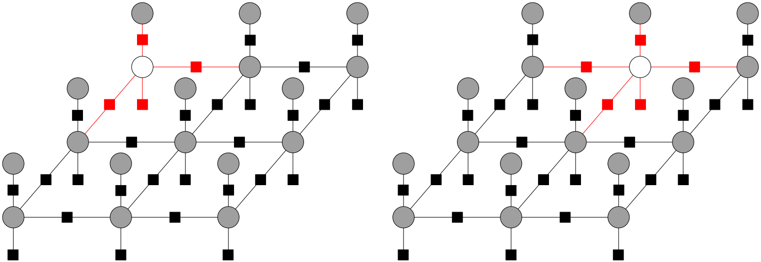
- *Metropolis-Hastings chain*, almost always possible
- Special case: *Gibbs sampler* (Geman and Geman, 1984)
  1. Select a variable y_i,
  2. Sample $y_i \sim p(y_i | y_{V \setminus \{i\}}, x)$.

Gibbs Sampler



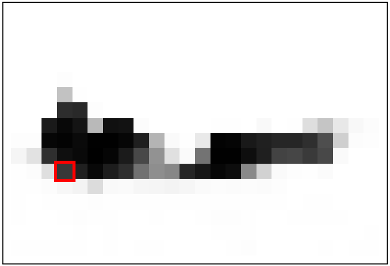
$$p(y_i|y^{(t)}_{V\setminus\{i\}}, x, w) = \frac{p(y_i, y^{(t)}_{V\setminus\{i\}}|x, w)}{\sum_{y_i \in \mathcal{Y}_i} p(y_i, y^{(t)}_{V\setminus\{i\}}|x, w)} = \frac{\tilde{p}(y_i, y^{(t)}_{V\setminus\{i\}}|x, w)}{\sum_{y_i \in \mathcal{Y}_i} \tilde{p}(y_i, y^{(t)}_{V\setminus\{i\}}|x, w)}$$

Gibbs Sampler

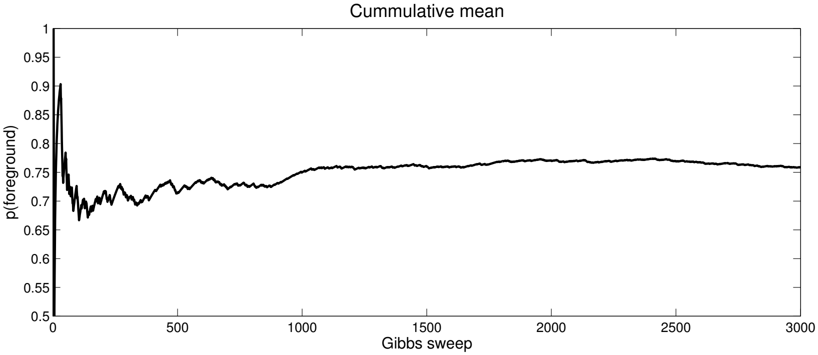


$$p(y_i | y^{(t)}_{V \setminus \{i\}}, x, w) = \frac{\sum_{F \in M(i)} \exp(-E_F(y_i, y^{(t)}_{F \setminus \{i\}}, x_F, w))}{\sum_{y_i \in \mathcal{Y}_i} \sum_{F \in M(i)} \exp(-E_F(y_i, y^{(t)}_{F \setminus \{i\}}, x_F, w))}$$

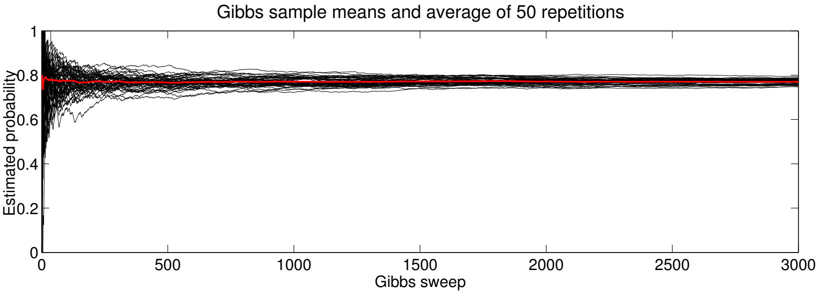
Example: Gibbs sampler



Example: Gibbs sampler

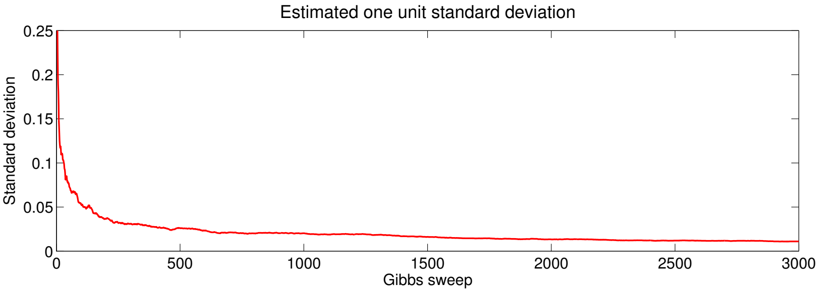


Example: Gibbs sampler



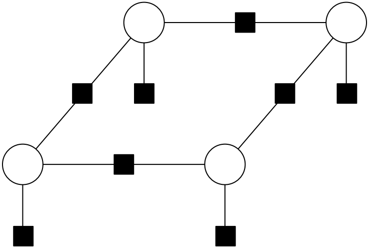
- $p(y_i = \text{“}foreground\text{''}) \approx 0.770 \pm 0.011$

Example: Gibbs sampler

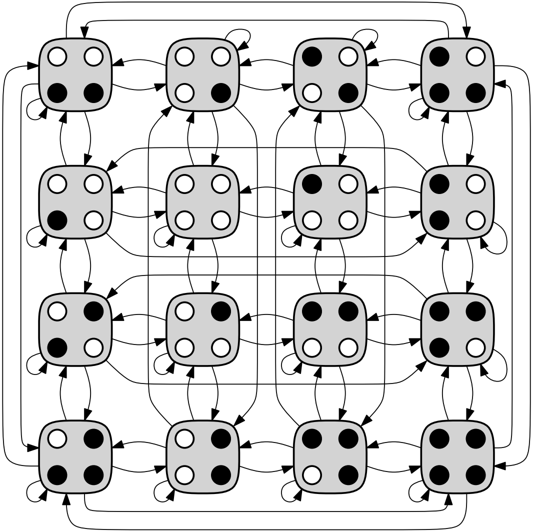


- Standard deviation $O(1/\sqrt{S})$

Gibbs Sampler Transitions



Gibbs Sampler Transitions

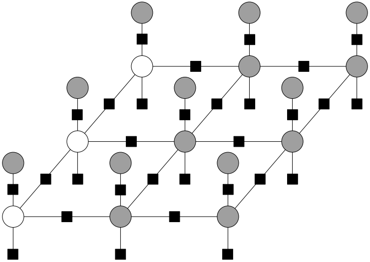


Block Gibbs Sampler

Extension to larger groups of variables

1. Select a block y_I
2. Sample $y_I \sim p(y_I | y_{V \setminus I}, x)$

$\rightarrow$ Tractable if sampling from blocks is tractable.



Summary: Sampling

Two families of Monte Carlo methods

1. Markov Chain Monte Carlo (MCMC)
2. Importance Sampling

Properties

- Often simple to implement, general, parallelizable
- (Cannot compute partition function Z)
- Can fail without any warning signs

References

- (Häggström, 2000), introduction to Markov chains
- (Liu, 2001), excellent Monte Carlo introduction

Coffee Break

Coffee Break

Continuing at 10:30am

Slides available at
`http://www.nowozin.net/sebastian/cvpr2011tutorial/`