Message passing and expectation propagation

Christoph Dehner

Technische Universität München
Department of Informatics
Data Mining and Analytics
kdd.in.tum.de

29 May 2017

Overview

Graphical models: Bayesian networks and Markov random fields

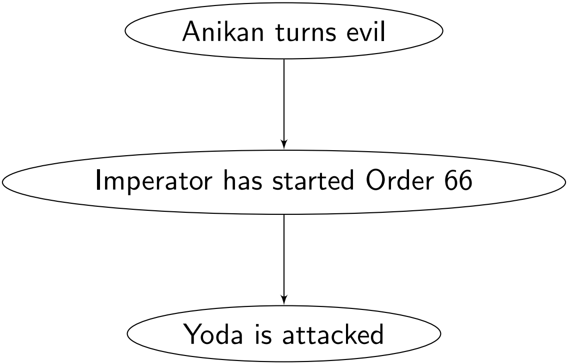
$$p(X) = \prod_s f_s(X_s)$$

Inference in graphical models:

- Marginalization
- Maximum aposteriori extimation
- Posterior approximation

Motivation for marginalization

Simple (but cool) example:



Factor graph

More formally:

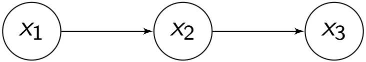
$$p(X) = p(x_1)p(x_2|x_1)p(x_3|x_1)$$

$$x_1 \longrightarrow x_2 \longrightarrow x_3$$

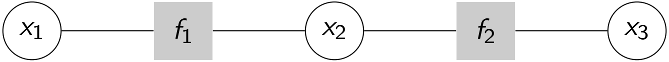
Factor graph

More formally:

$$p(X) = p(x_1)p(x_2|x_1)p(x_3|x_1)$$



A corresponding factor graph: $f_1 = p(x_1)p(x_2|x_1)$, $f_2 = p(x_3|x_2)$

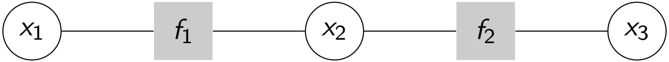


$$p(X) = \prod_s f_s(X_s)$$

Message passing

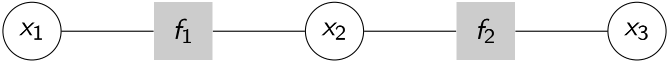
Message passing: Idea

$$p(X) = \prod_s f_s(X_s)$$



Message passing: Idea

$$p(X) = \prod_s f_s(X_s)$$

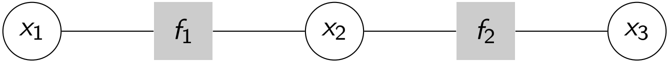


Marginalization naive:

$$p(x_2) = \sum_{x_1} \sum_{x_3} p(X) \in \mathcal{O}(k^n)$$

Message passing: Idea

$$p(X) = \prod_s f_s(X_s)$$



Marginalization naive:

$$p(x_2) = \sum_{x_1} \sum_{x_3} p(X) \in \mathcal{O}(k^n)$$

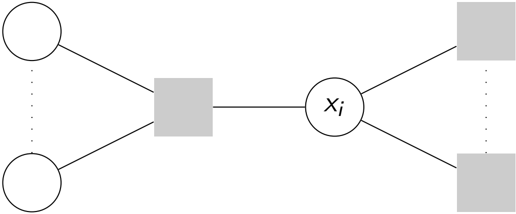
Marginalization advanced:

$$\begin{aligned} p(x_2) &= \sum_{x_1} \sum_{x_3} p(x_1) p(x_2|x_1) p(x_3|x_2) \\ &= \underbrace{\left[\sum_{x_1} p(x_1) p(x_2|x_1) \right]}_{\mu_{f_1 \to x_2}} \cdot \underbrace{\left[\sum_{x_3} p(x_3|x_2) \right]}_{\mu_{f_2 \to x_2}} \in \mathcal{O}(n \cdot k^2) \end{aligned}$$

Sum-product algorithm 1

General marginalization:

$$p(x_i) = \prod_{s \in ne(x_i)} \mu_{f_s \to x_i}(x_i)$$

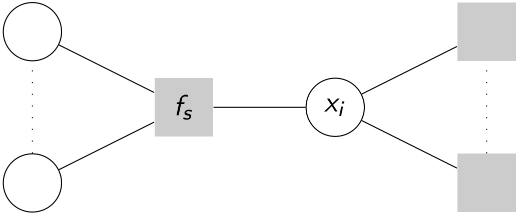


Sum-product algorithm 2

Local messages through factor graph:

$$\mu_{f_s \to x_i}(x_i) = \sum_{\mathbf{x_s} \setminus x_i} f_s(x_i, X_s) \prod_{m \in ne(f_s) \setminus x_i} \mu_{x_m \to f_s}(x_m)$$

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

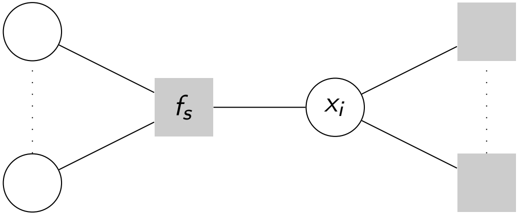


Sum-product algorithm 2

Local messages through factor graph:

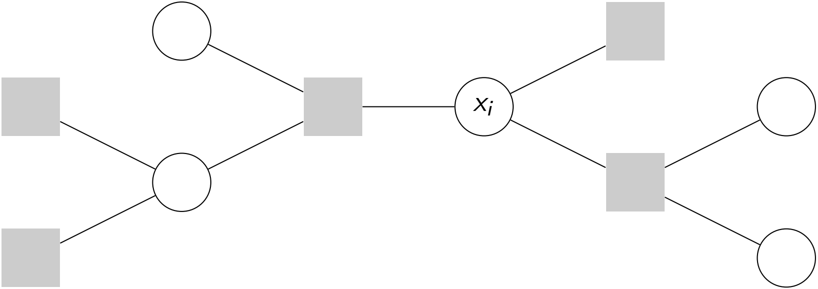
$$\mu_{f_s \to x_i}(x_i) = \sum_{\mathbf{x_s} \setminus x_i} f_s(x_i, X_s) \prod_{m \in ne(f_s) \setminus x_i} \mu_{x_m \to f_s}(x_m)$$

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

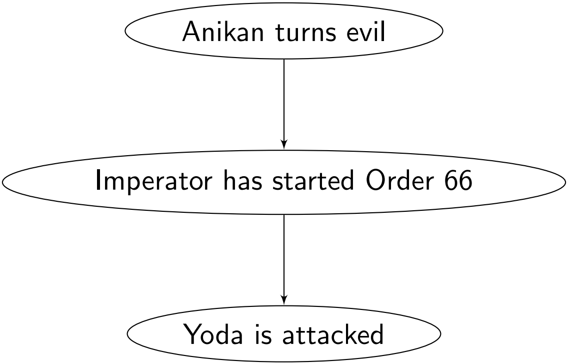


Initialization at leaf nodes: $\mu_{x \to f}(x) = 1$, $\mu_{f \to x}(x) = f(x)$

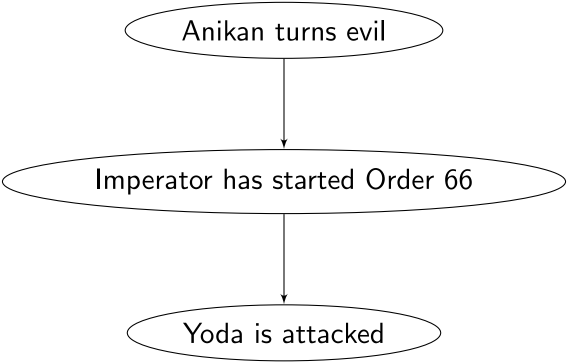
Message passing in action



Maximum aposteriori estimation



Maximum aposteriori estimation



$$X^* = \arg\max_X p(X)$$

Max-sum algorithm

Maximum aposteriori estimation:

$$X^* = \arg\max_X \; p(X) = \arg\max_X \; \ln(p(X))$$

Max-sum algorithm

Maximum aposteriori estimation:

$$X^* = \arg\max_X \; p(X) = \arg\max_X \; \ln(p(X))$$

Infer $p(X^*)$ by adapting sum-product algorithm:

$$\mu_{f_s \to x_i}(x_i) = \max_{X_s \backslash x_i} \left[f_s(x_i, X_s) \sum_{m \in ne(f_s) \backslash x_i} \mu_{x_m \to f_s}(x_m) \right]$$

$$\mu_{x_m \to f_s}(x_m) = \sum_{l \in ne(x_m) \backslash f_s} \mu_{f_l \to x_m}(x_m)$$

Max-sum algorithm

Maximum aposteriori estimation:

$$X^* = \arg\max_X \; p(X) = \arg\max_X \; \ln(p(X))$$

Infer $p(X^*)$ by adapting sum-product algorithm:

$$\mu_{f_s \to x_i}(x_i) = \max_{X_s \setminus x_i} \left[f_s(x_i, X_s) \sum_{m \in ne(f_s) \setminus x_i} \mu_{x_m \to f_s}(x_m) \right]$$

$$\mu_{x_m \to f_s}(x_m) = \sum_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

Initialization at leaf nodes: $\mu_{x \to f}(x) = 0$, $\mu_{f \to x}(x) = \ln f(x)$

Max-sum algorithm

Maximum aposteriori estimation:

$$X^* = \arg\max_X \, p(X) = \arg\max_X \, \ln(p(X))$$

Infer $p(X^*)$ by adapting sum-product algorithm:

$$\mu_{f_s \to x_i}(x_i) = \max_{X_s \setminus x_i} \left[f_s(x_i, X_s) \sum_{m \in ne(f_s) \setminus x_i} \mu_{x_m \to f_s}(x_m) \right]$$

$$\mu_{x_m \to f_s}(x_m) = \sum_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

Initialization at leaf nodes: $\mu_{x \to f}(x) = 0$, $\mu_{f \to x}(x) = \ln f(x)$

Keep track of maximal X values!

Message passing: Discussion

- Linear complexity in the number of involved variables

Message passing: Discussion

- Linear complexity in the number of involved variables
- Only exact for trees

Message passing: Discussion

- Linear complexity in the number of involved variables
- Only exact for trees
- In general graphs: Loopy belief propagation

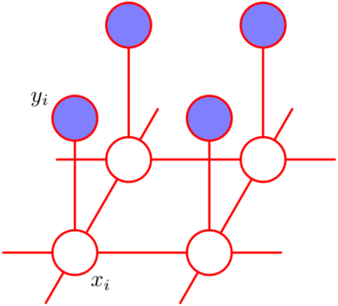
Message passing: Discussion

- Linear complexity in the number of involved variables
- Only exact for trees
- In general graphs: Loopy belief propagation
- Linearized message passing

Expectation propagation

Motivation for approximate inference

Image de-noising:



X true image, **Y** noisy observation

Figure taken from Bishop: Pattern Recognition and Machine Learning.

Expectation propagation: Methodology

Approximate posterior distribution:

$$p(\mathbf{X}|\mathbf{Y}) = \frac{1}{p(\mathbf{Y})} p(\mathbf{X}, \mathbf{Y}) = \frac{1}{Z} \prod_i f_i(\theta)$$

Expectation propagation: Methodology

Approximate posterior distribution:

$$p(\mathbf{X}|\mathbf{Y}) = \frac{1}{p(\mathbf{Y})} p(\mathbf{X}, \mathbf{Y}) = \frac{1}{Z} \prod_i f_i(\theta)$$

Approximating function from exponential family:

$$q(\theta) = \frac{1}{Z} \prod_i \tilde{f}_i(\theta)$$

Expectation propagation: Methodology

Approximate posterior distribution:

$$p(\mathbf{X}|\mathbf{Y}) = \frac{1}{p(\mathbf{Y})} p(\mathbf{X}, \mathbf{Y}) = \frac{1}{Z} \prod_i f_i(\theta)$$

Approximating function from exponential family:

$$q(\theta) = \frac{1}{Z} \prod_i \tilde{f}_i(\theta)$$

Minimize KL divergence $KL(p||q)$: Moment matching!

Expectation propagation: Algorithm

Approximating function:

$$q(\theta) = \frac{1}{Z} \prod_i \tilde{f}_i(\theta)$$

Expectation propagation: Algorithm

Approximating function:

$$q(\theta) = \frac{1}{Z} \prod_i \tilde{f}_i(\theta)$$

Iteratively refine one factor at a time:

$$q^{\setminus j}(\theta) = \prod_{i \neq j} \tilde{f}_i(\theta)$$

$$q^{new}(\theta) \propto f_j \; q^{\setminus j}(\theta)$$

$$\tilde{f}_j = \frac{1}{Z_j} \frac{q^{new}(\theta)}{q^{\setminus j}(\theta)}$$

Expectation propagation: Discussion

- Expectation propagation is not guaranteed to converge

Expectation propagation: Discussion

- Expectation propagation is not guaranteed to converge
- If EP converges, it often outperforms VI

EP vs. VI

Expectation propagation: Minimize $KL(p||q) = \int p \ln \frac{p}{q}$

Variational inference: Minimize $KL(q||p) = \int q \ln \frac{q}{p}$

EP vs. VI

Expectation propagation: Minimize $KL(p||q) = \int p \ln \frac{p}{q}$

Variational inference: Minimize $KL(q||p) = \int q \ln \frac{q}{p}$

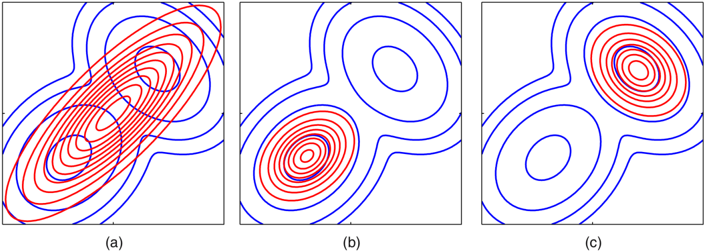


Figure taken from Bishop: Pattern Recognition and Machine Learning.

Thanks for your attention!
Questions?