Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Introduction to Probabilistic Machine Learning

Ralf Herbrich

Graphical Models: Inference

Overview

1. Factor Graphs
2. The Sum-Product Algorithm
3. Practical Considerations in Message Passing
4. Approximate Message Passing

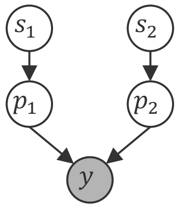
Overview

1. **Factor Graphs**
2. The Sum-Product Algorithm
3. Practical Considerations in Message Passing
4. Approximate Message Passing

Inference in Probabilistic Models

- **Learning**: In order to learn from data for most data models, we need to marginalize ("sum-out") all non-observed variables given the observed variables (i.e., data).
  - **Example**: Two player game with one winner

$$p(\boldsymbol{s}|y) \propto p(\boldsymbol{s}) \cdot \int \mathcal{N}(p_1; s_1, \beta^2) \cdot \mathcal{N}(p_2; s_2, \beta^2) \cdot \mathbb{I}(y(p_1 - p_2) > 0)\ dp_1\, dp_2$$



- **Problem**: Naïve summation scales exponentially because we have a sum of products (i.e., product of conditional disitrubtions of all latent variables)!
  - **Example**: Consider an example of n Bernoulli variables $x_1, \ldots, x_n$

$$p(x_1) = \underbrace{\sum_{x_2=0}^{1} \sum_{x_3=0}^{1} \cdots \sum_{x_n=0}^{1}}_{2^{n-1}\text{ summations}} p(x_1, x_2, \ldots, x_n)$$

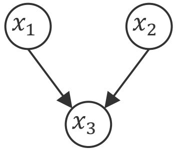
- **Idea**: We exploit the product structure of the probabilisitic model of our data because not every variable depends on all variables before them
  - **Example (ctd)**. Consider $p(x_1, x_2, \ldots, x_n) = \prod_i p(x_i)$: then there are only $O(n)$ sums!

Factor Graphs

- **Factor Graph (Frey, 1998)**. *Given a product of m functions $f_1, f_2, \dots, f_m$, each over a subset of n variables $x_1, x_2, \dots, x_n$, a factor graph if a bipartite graphical model with m factor nodes and n variable nodes where an undirected edge connects f_i and x_j if and only if the function f_i depends on x_j.*
- Factor graphs are more expressive than a Bayesian network!

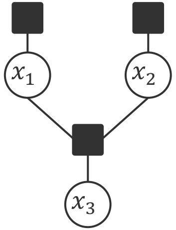
Brendan Frey
(1968 –)

Bayesian network



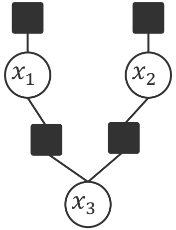
$p(x_1, x_2, x_3) = p(x_1) \cdot p(x_2) \cdot p(x_3 | x_1, x_2)$

Corresponding factor graph



$p(x_1, x_2, x_3) = f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3, x_1, x_2)$

Factor graph with more structure



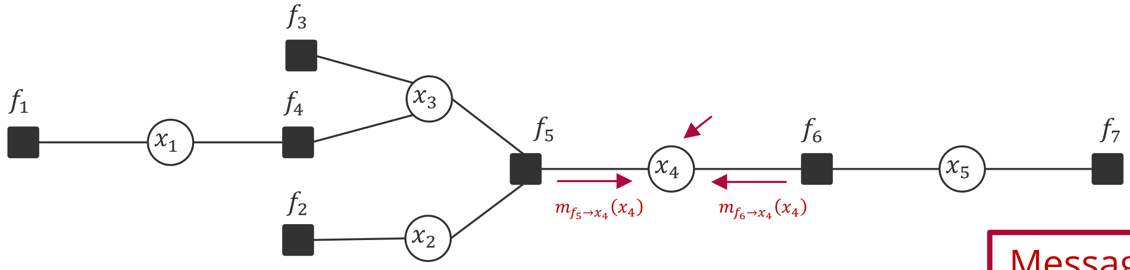
$p(x_1, x_2, x_3) = f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_1, x_3) \cdot f_4(x_2, x_3)$

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

Overview

1. Factor Graphs
2. **The Sum-Product Algorithm**
3. Practical Considerations in Message Passing
4. Approximate Message Passing

Sum-Product Algorithm: Marginals



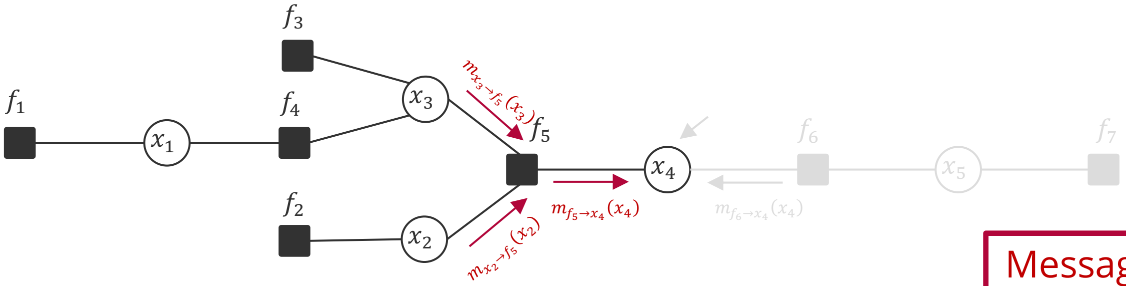
Message $m_{f_j \to x_i}(x_i)$ is the sum over all variables in the subtree rooted at f_j

$$p(x_4) = \sum_{\{x_1\}} \sum_{\{x_2\}} \sum_{\{x_3\}} \sum_{\{x_5\}} f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_2) \cdot f_4(x_1, x_3) \cdot f_5(x_2, x_3, x_4) \cdot f_6(x_4, x_5) \cdot f_7(x_5)$$

$$= \underbrace{\left[\sum_{\{x_1\}} \sum_{\{x_2\}} \sum_{\{x_3\}} f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3) \cdot f_4(x_1, x_3) \cdot f_5(x_2, x_3, x_4) \right]}_{m_{f_5 \to x_4}(x_4)} \cdot \underbrace{\left[\sum_{\{x_5\}} f_6(x_4, x_5) \cdot f_7(x_5) \right]}_{m_{f_6 \to x_4}(x_4)}$$

Marginals are the product of all incoming messages from neighbouring factors!

Sum-Product Algorithm: Message from Factor to Variable



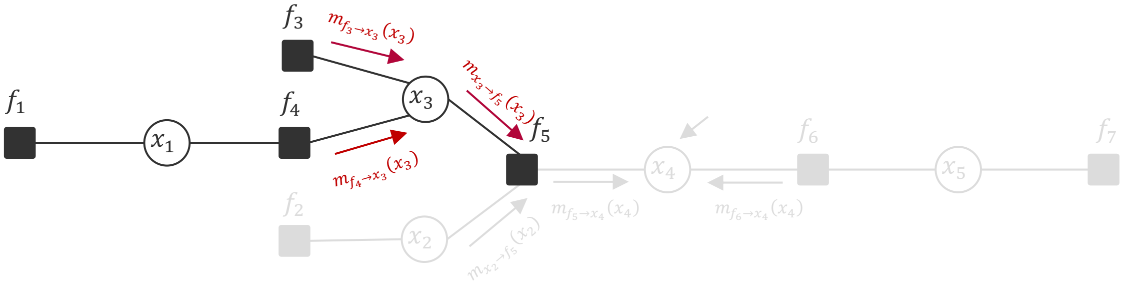
Message $m_{x_i \to f_j}(x_i)$ is the sum over all variables in the subtree rooted at x_i

$$m_{f_5 \to x_4}(x_4) = \sum_{\{x_1\}} \sum_{\{x_2\}} \sum_{\{x_3\}} f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3) \cdot f_4(x_1, x_3) \cdot f_5(x_2, x_3, x_4)$$

$$= \sum_{\{x_2\}} \sum_{\{x_3\}} f_5(x_2, x_3, x_4) \cdot \underbrace{[f_2(x_2)]}_{m_{x_2 \to f_5}(x_2)} \cdot \underbrace{\left[\sum_{\{x_1\}} f_1(x_1) \cdot f_3(x_3) \cdot f_4(x_1, x_3)\right]}_{m_{x_3 \to f_5}(x_3)}$$

Messages from a factor to a variable sum out all neighboring variables weighted by their incoming message

Sum-Product Algorithm: Message from Variable to Factor



$$m_{x_3 \to f_5}(x_3) = \sum_{\{x_1\}} f_1(x_1) \cdot f_3(x_3) \cdot f_4(x_1, x_3)$$

$$= \underbrace{[f_3(x_3)]}_{m_{f_3 \to x_3}(x_3)} \cdot \underbrace{\left[\sum_{\{x_1\}} f_1(x_1) \cdot f_4(x_1, x_3)\right]}_{m_{f_4 \to x_3}(x_3)}$$

Messages from a variable to a factor multiply incoming message from neighboring factors

Sum-Product Algorithm

- **Sum-Product Algorithm (Aji-McEliece, 1997).** Putting it all together, we have

$$p(x) = \prod_{f \in \mathrm{ne}(x)} m_{f \to x}(x)$$

$$m_{f \to x}(x) = \sum_{\{x' \in \mathrm{ne}(f) \setminus \{x\}\}} \cdots \sum_{\{x'' \in \mathrm{ne}(f) \setminus \{x\}\}} f(x, x', \dots, x'') \prod_{x' \in \mathrm{ne}(f) \setminus \{x\}} m_{x' \to f}(x')$$

$$m_{x \to f}(x) = \prod_{f' \in \mathrm{ne}(x) \setminus \{f\}} m_{f' \to x}(x)$$

- **Basis**: Generalized distributive law (which also holds for max-product)
- **Efficiency**: By storing messages, we
  - Only have to compute local summations in $O(2^T)$ where degree $T = \max_f |\mathrm{ne}(f)|$!
  - All marginals can be computed recursively in $O(E \cdot 2^T)$ vs $O(2^n)$ (where E is the number of edges of the factor graph)!

Robert McEliece
(1942 – 2019)

Introduction to Probabilistic Machine Learning

Unit 4 – Graphical Models: Inference

Overview

Even more efficiency

- **Redundancies.** By the very definition of messages and marginals

$$p(x) = \prod_{f \in \mathrm{ne}(x)} m_{f \to x}(x) = m_{f' \to x}(x) \cdot \underbrace{\prod_{f \in \mathrm{ne}(x) \setminus \{f'\}} m_{f \to x}(x)}_{m_{x \to f'}(x)}$$

  - **Interpretation**. Application of Bayes' rule at a variable x at factor f

$$\underbrace{p(x)}_{\text{posterior}} = \underbrace{m_{f \to x}(x)}_{\text{Likelihood} \times \text{normalization}} \cdot \underbrace{m_{x \to f}(x)}_{\text{prior}}$$

- **Storage Efficiency**. We only store the marginals $p(x)$ and $m_{f \to x}(x)$ because

$$m_{x \to f}(x) = \frac{p(x)}{m_{f \to x}(x)}$$

- **Exponential Family**. If all the messages from factors to variables are in the exponential family, then the marginals and messages from the variable to factors are simply additions and subtraction of natural parameters!
  - **Example**: If $p(x) = \mathcal{G}(x; \tau_1, \rho_1)$ and $m_{f \to x}(x) = \mathcal{G}(x; \tau_2, \rho_2)$ then $m_{x \to f}(x) \propto \mathcal{G}(x; \tau_1 - \tau_2, \rho_1 - \rho_2)$

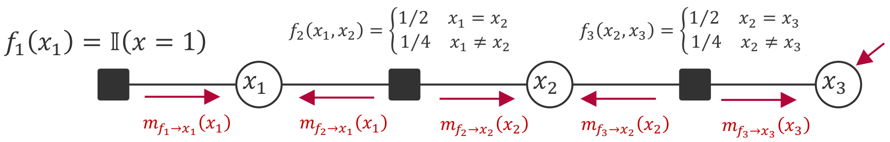
A Practical Implementation

1. Initialize all messages $m_{f\to x}(x)$ and marginals $p(x)$ with a constant function (i.e., uniform distribution)

2. Pick an arbitrary root (say, x_3)

3. Update all messages $m_{f\to x}(x)$ from the leaves of the tree rooted at x_3 **upwards**

4. Update all messages $m_{f\to x}(x)$ from the root x_3 to the leaves **downwards**



Update	$p(x_1)$	$p(x_2)$	$p(x_3)$	$m_{f_1\to x_1}(x_1)$	$m_{f_2\to x_1}(x_1)$	$m_{f_2\to x_2}(x_2)$	$m_{f_3\to x_2}(x_2)$	$m_{f_3\to x_3}(x_3)$
Initial	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$
$m_{f_1\to x_1}(x_1)$	$[1,0,0]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$[1,0,0]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$
$m_{f_2\to x_2}(x_2)$	$[1,0,0]$	$\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$[1,0,0]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$
$m_{f_3\to x_3}(x_3)$	$[1,0,0]$	$\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]$	$\left[\frac{3}{8},\frac{5}{16},\frac{5}{16}\right]$	$[1,0,0]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{3}{8},\frac{5}{16},\frac{5}{16}\right]$
$m_{f_3\to x_2}(x_2)$	$[1,0,0]$	$\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]$	$\left[\frac{3}{8},\frac{5}{16},\frac{5}{16}\right]$	$[1,0,0]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{3}{8},\frac{5}{16},\frac{5}{16}\right]$
$m_{f_2\to x_1}(x_1)$	$[1,0,0]$	$\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]$	$\left[\frac{3}{8},\frac{5}{16},\frac{5}{16}\right]$	$[1,0,0]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]$	$\left[\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]$	$\left[\frac{3}{8},\frac{5}{16},\frac{5}{16}\right]$

Inference

$$m_{f_2\to x_2}(x_2) = \sum_{x_1=1}^{3} f_2(x_1,x_2)\cdot \boxed{\frac{p(x_1)}{m_{f_2\to x_1}(x_1)}} \longleftarrow m_{x_1\to f_2}(x_1)$$

Normal Distributions and the Product Rule

- **Theorem (Multiplication)**. *Given two one-dimensional Gaussian distributions* $\mathcal{G}(x;\tau_1,\rho_1)$ *and* $\mathcal{G}(x;\tau_2,\rho_2)$ *we have*

Gaussian density
(as normalization constant)

$$\mathcal{G}(x;\tau_1,\rho_1)\cdot\mathcal{G}(x;\tau_2,\rho_2)=\mathcal{G}(x;\tau_1+\tau_2,\rho_1+\rho_2)\cdot\mathcal{N}(\mu_1;\mu_2,\sigma_1^2+\sigma_2^2)$$

Additive updates!

- **Theorem (Division)**. *Given two one-dimensional Gaussian distributions* $\mathcal{G}(x;\tau_1,\rho_1)$ *and* $\mathcal{G}(x;\tau_2,\rho_2)$ *we have*

Gaussian density
(as normalization constant)

$$\frac{\mathcal{G}(x;\tau_1,\rho_1)}{\mathcal{G}(x;\tau_2,\rho_2)}=\mathcal{G}(x;\tau_1-\tau_2,\rho_1-\rho_2)\cdot\frac{1}{\mathcal{N}\left(\frac{\tau_1-\tau_2}{\rho_1-\rho_2};\frac{\tau_2}{\rho_2},\frac{1}{\rho_1-\rho_2}+\frac{1}{\rho_2}\right)}$$

Subtractive updates!

Overview

1. Factor Graphs
2. The Sum-Product Algorithm
3. Practical Considerations in Message Passing
4. **Approximate Message Passing**

Approximate Message Passing

- **Message update from factors to variables**. For general factors f, the sum-product algorithm is not closed under the application of

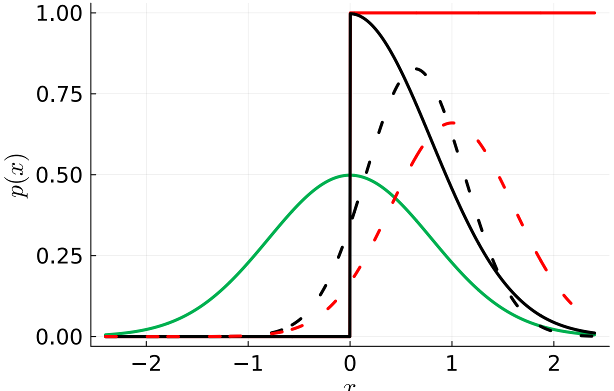
$$m_{f\to x}(x) = \sum_{\{x'\in \mathrm{ne}(f)\setminus\{x\}\}} \cdots \sum_{\{x''\in \mathrm{ne}(f)\setminus\{x\}\}} f(x, x', \dots, x'') \prod_{x'\in \mathrm{ne}(f)\setminus\{x\}} m_{x'\to f}(x')$$

  - **Example**: Truncating a 1D-Gaussian distribution

- **Idea**: Find the "best" approximation $\hat{p}(x)$ for the marginal $p(x)$ and approximate $m_{f\to x}(x)$ by

$$\widehat{m}_{f\to x}(x) = \frac{\hat{p}(x)}{m_{x\to f}(x)}$$

  - **Example**: Truncating a 1D-Gaussian distribution



Information Theoretic Approximation: KL Divergence

- **Problem**. We have a non-Gaussian posterior distribution $p(x)$ and would like to approximate it by a Gaussian $q(x) = \mathcal{N}(x; \mu, \sigma^2)$.
- **Idea**. The best approximation $\mu^*, {\sigma^2}^*$ minimizes the Kullback-Leibler divergence

$$\mathrm{KL}\big(p(\cdot)|\mathcal{N}(\cdot; \mu, \sigma^2)\big) = \int p(x) \cdot \log_2 \left(\frac{p(x)}{\mathcal{N}(x; \mu, \sigma^2)} \right) dx$$

- **Theorem (Moment Matching)**. *Given any distribution $p(x)$ the minimizer $\mu^*, {\sigma^2}^*$ of the KL divergence $\mathrm{KL}\big(p(\cdot)|\mathcal{N}(\cdot; \mu, \sigma^2)\big)$ to a Gaussian distribution is*

$$\mu^* = E_{x \sim p(x)}[x] \quad \text{and} \quad {\sigma^2}^* = E_{x \sim p(x)}[x^2] - (\mu^*)^2$$

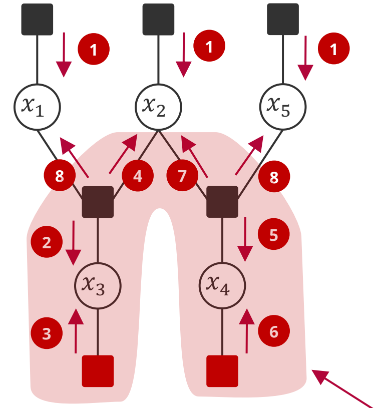
Solomon Kullback
(1909 – 1994)

Richard Leibler
(1914 – 2003)

Expectation Propagation

- **Idea**: If we have factors in the factor graph that require approximate messages, we keep iterating on the whole path between them until convergence minimizing $\mathrm{KL}\left(p(\cdot)|\mathcal{N}(\cdot;\mu,\sigma^2)\right)$ locally for the affected marginals of the approximate factor.
- **Theorem (Minka, 2003)**: Approximate message passing will converge if the approximating distribution is in the exponential family!

Tom Minka



iterate until convergence

Introduction to Probabilistic Machine Learning

Unit 4 – Graphical Models: Inference

Approximating the Normalization Constant

- **Problem**: Given a factor graph $\prod_{i=1}^{m} f_i\left(\boldsymbol{x}_{\mathrm{ne}(f_i)}\right)$ we would like to also compute

$$Z = \sum_{\{x_1\}} \cdots \sum_{\{x_n\}} f_1\left(\boldsymbol{x}_{\mathrm{ne}(f_1)}\right) \cdot f_2\left(\boldsymbol{x}_{\mathrm{ne}(f_2)}\right) \cdots f_m\left(\boldsymbol{x}_{\mathrm{ne}(f_m)}\right) = \sum_{\{\boldsymbol{x}\}} \prod_{i=1}^{m} f_i\left(\boldsymbol{x}_{\mathrm{ne}(f_i)}\right)$$

- **Observation**: Each factor is replaced by $f_i\left(\boldsymbol{x}_{\mathrm{ne}(f_i)}\right) = Z_{f_i} \cdot \prod_{j \in \mathrm{ne}(f_i)} \widehat{m}_{f_i \to x_j}(x_j)$
- **Conclusion**: Z is product of factor and variable marginal normalization

$$\sum_{\{\boldsymbol{x}\}} \prod_{i=1}^{m} f_i\left(\boldsymbol{x}_{\mathrm{ne}(f_i)}\right) = \sum_{\{\boldsymbol{x}\}} \prod_{i=1}^{m} \prod_{j \in \mathrm{ne}(f_i)} Z_{f_i} \cdot \widehat{m}_{f_i \to x_j}(x_j) = \left[\prod_{i=1}^{m} Z_{f_i}\right] \cdot \left[\prod_{j=1}^{n} \sum_{\{x_j\}} \prod_{f \in \mathrm{ne}(x_j)} \widehat{m}_{f \to x_j}(x_j)\right]$$

One normalization per factor

- **Idea**: Approximate each Z_{f_i} by matching the zeroth moment (i.e. sum over all $\boldsymbol{x}_{\mathrm{ne}(f_i)}$):

$$\sum_{\{\boldsymbol{x}_{\mathrm{ne}(f_i)}\}} f_i\left(\boldsymbol{x}_{\mathrm{ne}(f_i)}\right) \prod_{j \in \mathrm{ne}(f_i)} \widehat{m}_{x_j \to f_i}(x_j) = Z_{f_i} \cdot \sum_{\{\boldsymbol{x}_{\mathrm{ne}(f_i)}\}} \prod_{j \in \mathrm{ne}(f_i)} \widehat{m}_{f_i \to x_j}(x_j) \cdot \prod_{j \in \mathrm{ne}(f_i)} \widehat{m}_{x_j \to f_i}(x_j)$$

$$Z_{f_i} = \frac{\sum_{\{\boldsymbol{x}_{\mathrm{ne}(f_i)}\}} f_i\left(\boldsymbol{x}_{\mathrm{ne}(f_i)}\right) \prod_{j \in \mathrm{ne}(f_i)} \widehat{m}_{x_j \to f_i}(x_j)}{\sum_{\{\boldsymbol{x}_{\mathrm{ne}(f_i)}\}} \prod_{j \in \mathrm{ne}(f_i)} \widehat{m}_{f_i \to x_j}(x_j) \cdot \prod_{j \in \mathrm{ne}(f_i)} \widehat{m}_{x_j \to f_i}(x_j)}$$

Influence of all other factors on the zeroth moment approximation

Summary

1. **Factor Graphs**
   - Generalization of Bayesian networks specifically designed for fast inference
   - Date back to coding algorithms
2. **Sum-Product Algorithm**
   - Application of generalized distributive law
   - Trades memory (“messages”) for computation (“sums”)
   - Reduces the computational complexity to exponential in the largest out-degree of a factor rather than exponential in the number of variables
3. **Approximate Message Passing and Expectation Propagation**
   - Approximations will always be done on the marginals, **not** the messages!
   - When the Kullback-Leibler divergence is used as distance, all moments get preserved!

See you next week!