Factor Graphs, the Sum-Product Algorithm and TrueSkill™

Kuhwan Jeong [1]

[1] Department of Statistics, Seoul National University, South Korea

July, 2018

Introduction

- x_i : a variable taking on values in some domain A_i, $i = 1, \ldots, n$
- $g(x_1, \ldots, x_n)$: a function of $x_1, \ldots, x_n$
- $g_i(x_i)$: the marginal function w.r.t. x_i

$$
\begin{aligned}
g_i(x_i) &= \sum_{x_1 \in A_1} \cdots \sum_{x_{i-1} \in A_{i-1}} \sum_{x_{i+1} \in A_{i+1}} \sum_{x_n \in A_n} g(x_1, \ldots, x_n) \\
&= \sum_{\sim x_i} g(x_1, \ldots, x_n)
\end{aligned}
$$

- The sum-product algorithm is a efficient procedure for computing marginal functions that
  - a) exploit the way in which the global function factors, and
  - b) reuses intermediate values.
- It is a simple way to understand a large number of seemingly different algorithms that have been developed.

Factor Graphs

- Suppose that $g(x_1, \ldots, x_n)$ factors into a product of several local functions,

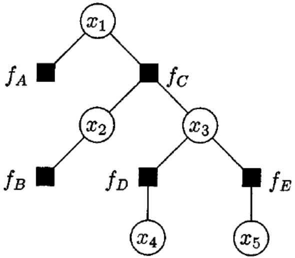
$$g(x_1, \ldots, x_n) = \prod_{j \in J} f_j(X_j)$$

  where J is a discrete index set, X_j is a subset of $\{x_1, \ldots, x_n\}$, and $f_j(X_j)$ is a function having the elements of X_j as arguments.

- A factor graph is a bipartite graph that expresses the structure of the factorization. It has
  - a *variable node* for each variable x_i,
  - a *factor node* for each local function f_j,
  - and an edge between a variable node x_i and a factor node f_j if and only if x_i is an argument of f_j.

Example.

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5)$$



Example (continued).

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5)$$

$$\begin{aligned}
g_1(x_1) &= f_A(x_1) \left[\sum_{x_2} f_B(x_2) \left\{ \sum_{x_3} f_C(x_1, x_2, x_3) \left(\sum_{x_4} f_D(x_3, x_4) \right) \left(\sum_{x_5} f_E(x_3, x_5) \right) \right\} \right] \\
&= f_A(x_1) \sum_{\sim x_1} \left\{ f_B(x_2) f_C(x_1, x_2, x_3) \left(\sum_{\sim x_3} f_D(x_3, x_4) \right) \left(\sum_{\sim x_3} f_E(x_3, x_5) \right) \right\} \\
g_3(x_3) &= \left(\sum_{\sim x_3} f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) \right) \left(\sum_{\sim x_3} f_D(x_3, x_4) \right) \left(\sum_{\sim x_3} f_E(x_3, x_5) \right)
\end{aligned}$$

Computing a Single Marginal Function

Single-i Sum-Product algorithm.

- Take x_i as the root vertex.
- Each leaf node sends a message to its parent.
- Each vertex waits for messages from all of its children before computing the message to be sent to its parent.
- A variable node simply sends the product of messages received from its children

  $$\mu_{x \to f} = \prod_{h \in n(x) \setminus f} \mu_{h \to x}(x)$$

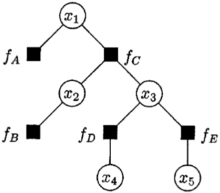
  where $n(x)$ is the set of functions of which x is an argument.
- A factor node f with a parent x forms the product of f with the messages received from its children, and then operates the summation $\sum_{-x}$ on the result

  $$\mu_{f \to x}(x) = \sum_{\sim x} \left(f(X) \prod_{y \in n(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

  where $X = n(f)$ is the set of arguments of f.
- $g_i(x_i)$ is obtained as the product of all messages received at x_i.

Computing a Single Marginal Function

Example (continued).

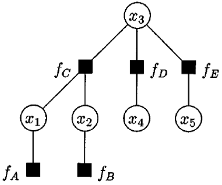


$$
\begin{aligned}
\mu_{x_4 \to f_D} &= \mu_{x_5 \to f_E} = 1, \\
\mu_{f_D \to x_3} &= \sum_{\sim x_3} f_D(x_3, x_4),\ \mu_{f_E \to x_3} = \sum_{\sim x_3} f_E(x_3, x_5), \\
\mu_{x_3 \to f_C} &= \left(\sum_{\sim x_3} f_D(x_3, x_4) \right) \left(\sum_{\sim x_3} f_E(x_3, x_5) \right), \\
\mu_{f_B \to x_2} &= \mu_{x_2 \to f_C} = f_B(x_2),
\end{aligned}
$$

$$
\begin{aligned}
\mu_{f_C \to x_1} &= \sum_{\sim x_1} \left\{ f_B(x_2) f_C(x_1, x_2, x_3) \left(\sum_{\sim x_3} f_D(x_3, x_4) \right) \left(\sum_{\sim x_3} f_E(x_3, x_5) \right) \right\}, \\
\mu_{f_A \to x_1} &= f_A(x_1), \\
g_1(x_1) &= f_A(x_1) \sum_{\sim x_1} \left\{ f_B(x_2) f_C(x_1, x_2, x_3) \left(\sum_{\sim x_3} f_D(x_3, x_4) \right) \left(\sum_{\sim x_3} f_E(x_3, x_5) \right) \right\}.
\end{aligned}
$$

Computing a Single Marginal Function

Example (continued).



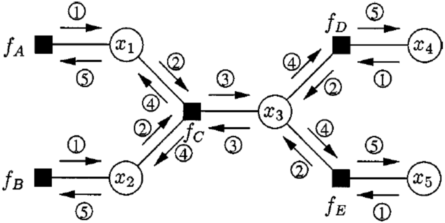
$$
\begin{aligned}
\mu_{f_A \to x_1} &= \mu_{x_1 \to f_C} = f_A(x_1), \\
\mu_{f_B \to x_2} &= \mu_{x_2 \to f_C} = f_B(x_2), \\
\mu_{f_C \to x_3} &= \sum_{\sim x_3} f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3), \\
\mu_{x_4 \to f_D} &= \mu_{x_5 \to f_E} = 1, \\
\mu_{f_D \to x_3} &= \sum_{\sim x_3} f_D(x_3, x_4), \; \mu_{f_E \to x_3} = \sum_{\sim x_3} f_E(x_3, x_5),
\end{aligned}
$$

$$
g_3(x_3) = \left(\sum_{\sim x_3} f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) \right) \left(\sum_{\sim x_3} f_D(x_3, x_4) \right) \left(\sum_{\sim x_3} f_E(x_3, x_5) \right).
$$

Computing All Marginal Functions

Sum-Product algorithm.

- As in the single-i algorithm, message passing is initiated at the leaves.
- Each vertex v remains idle until messages have arrived on all but one of the edges incident on v.



Example : TrueSkill

- TrueSkill ranking system is a skill based ranking system for Xbox Live developed at Microsoft Research.

- The purpose is to both identify and track the skills of gamers in order to be able to match them into competitive matches.

- TrueSkill ranking system only uses the final standings of all teams in a game in order to update the skill estimates of all gamers playing in this game.

- Ranking systems have been proposed for many sports but possibly the most prominent ranking system in use today is the *Elo system*.

Elo System

- In 1959, Arpad Elo developed a statistical rating system for Chess, which was adopted by the World Chess Federation FIDE in 1970.
- It models the probability of the possible game outcomes as a function of the two players' skill ratings s_1 and s_2.
- In a game each player i exhibits performance $p_i \sim \mathcal{N}(s_i, \beta^2)$.
- The probability that player 1 wins is given by

$$P(p_1 > p_2 | s_1, s_2) = \Phi\left(\frac{s_1 - s_2}{\sqrt{2}\beta}\right).$$

- Let $y = 1$ if player 1 wins, $y = -1$ if player 2 wins and $y = 0$ if a draw occurs.
- After the game, the skill ratings s_1 and s_2 are updated by

$$\begin{aligned} s_1 &\leftarrow s_1 + y\Delta \\ s_2 &\leftarrow s_2 - y\Delta \end{aligned}$$

where

$$\Delta = \alpha\beta\sqrt{\pi}\left(\frac{y+1}{2} - \Phi\left(\frac{s_1 - s_2}{\sqrt{2}\beta}\right)\right).$$

TrueSkill

- Assume an independent normal prior $p(s) = \prod_{i=1}^{n} \mathcal{N}(s_i; \mu_i, \sigma_i^2)$.
- Each player i exhibits a performance $p_i \sim \mathcal{N}(p_i; s_i, \beta^2)$.
- From among a population of n players $\{1, \ldots, n\}$ in a game let k teams compete a match.
- The team assignments are specified by k non-overlapping subsets

$$A_j \subset \{1, \ldots, n\}, A_i \cap A_j = \text{ if } i \neq j.$$

- The performance t_j of team j is modeled as the sum of the performances of its members

$$t_j = \sum_{i \in A_j} p_i.$$

TrueSkill

- The outcome $\mathbf{r} = (r_1, \ldots, r_k) \in \{1, \ldots, k\}^k$ is specified by a rank r_j for each team j.

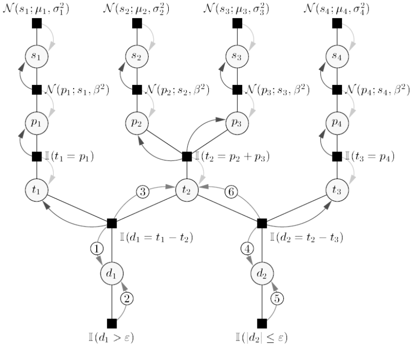
- Disregarding draws, the probability of a game outcome r is modeled as

$$P(\mathbf{r}|t_1, \ldots, t_k) = P(t_{r_{(1)}} > t_{r_{(2)}} > \cdots > t_{r_{(k)}}).$$

- If draws are permitted the wining outcome $r_{(j)} < r_{(j+1)}$ requires $t_{r_{(j)}} > t_{r_{(j+1)}} + \epsilon$ and the draw outcome $r_{(j)} = r_{(j+1)}$ requires $|t_{r_{(j)}} - t_{r_{(j+1)}}| \leq \epsilon$, where ϵ is a draw margin.

TrueSkill

- Consider a game with 3 teams with $A_1 = \{1\}, A_2 = \{2, 3\}$ and $A_3 = 4$.
- Assume that team 1 is the winner and that teams 2 and 3 draw, i.e., $\mathbf{r} = (1, 2, 2)$.
- The factor graph representing the joint distribution $P(\mathbf{s}, \mathbf{p}, \mathbf{t}|\mathbf{r}, A)$ is depicted.



TrueSkill

- The quantities of interest are the posterior distribution $P(s_i|\mathbf{r}, A)$.

- $P(s_i|\mathbf{r}, A)$ is calculated from the joint distribution integrating out the individual performances $\{p_i\}$ and the team performances $\{t_i\}$,

$$P(s_i|\mathbf{r}, A) = \int \int P(\mathbf{s}, \mathbf{p}, \mathbf{t}|\mathbf{r}, A) d\mathbf{p} d\mathbf{t}$$

Approximate Message Passing

- The message passing is characterized by the following equations:

$$\begin{aligned} P(v_k) &= \prod_{f \in n(v_k)} \mu_{f \to v_k}(v_k), \\ \mu_{f \to v_j}(v_j) &= \int f(\mathbf{v}) \prod_{v_i \in n(f) \setminus \{v_j\}} m_{v_i \to f}(v_i) d\mathbf{v}_{\sim j}, \\ \mu_{v_k \to f}(v_k) &= \prod_{h \in n(v_k) \setminus \{f\}} \mu_{h \to v_k}(v_k). \end{aligned}$$

- The TrueSkill factor graph is acyclic and the majority of messages can be represented compactly as 1-dimensional Gaussians.

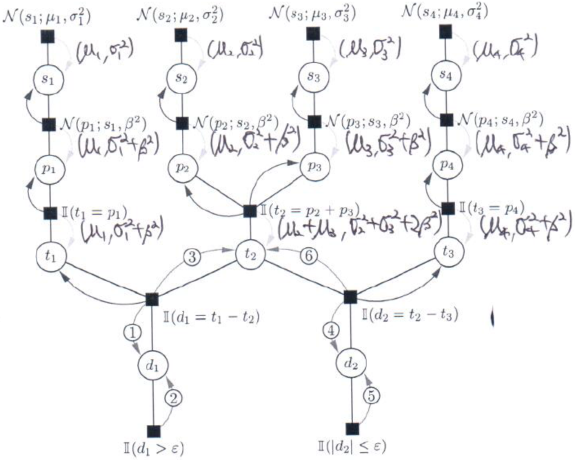
- However, messages from the comparison factors $I(\cdot > \epsilon)$ or $I(|\cdot| \leq \epsilon)$ to the performance differences d_i are non Gaussian.

- We approximate these messages by approximating the marginal $P(d_i)$ via moment matching resulting in a Gaussian $\hat{P}(d_i)$ with the same mean and variance as $P(d_i)$.

- Then, we have

$$\hat{\mu}_{f \to d_i}(d_i) = \frac{\hat{P}(d_i)}{\mu_{d_i \to f}(d_i)}.$$

Approximate Message Passing



- Since the messages 2 and 5 are approximate, iterate over all messages that are on the shortest path between any two approximate marginals $\hat{P}(d_i)$ until the convergence of marginals.

References

- Kschischang, F. R., Frey, B. J., & Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. IEEE Transactions on information theory, 47(2), 498-519.

- Herbrich, R., Minka, T., & Graepel, T. (2007). TrueSkill$^{\mathrm{TM}}$: a Bayesian skill rating system. In Advances in neural information processing systems (pp. 569-576).