# Variable Elimination and Belief Propagation in Graphical Models

Karl Stratos

We write $X = (X_1 \dots X_n) \in \mathcal{X}^n$ to denote $n$ discrete random variables. Let $K = |\mathcal{X}|$. We write $x = (x_1 \dots x_n)$ to mean a specific configuration of $X$. Similarly, if $X'$ is a subset of $X$, then we write $x'$ to mean a specific configuration of that subset. **Graphical models** express a distribution over $X$ in terms of nodes and edges.

## 1 Types of Graphical Models

A **directed graphical model (DGM)**, or **Bayesian network**, is a directed acyclic graph (DAG) that represents the chain rule applied on $p(X)$, optionally with Markov assumptions (e.g., HMMs, generative probabilistic neural models). If $\mathbf{pa}(X_i) \subseteq X$ denotes the parent nodes of $X_i$ under a DGM, then it is parameterized by local distributions $p(X_i|\mathbf{pa}(X_i))$ to define

$$p(X) = \prod_{i=1}^{n} p(X_i|\mathbf{pa}(X_i))$$

An **undirected graphical model (UGM)**, or **Markov random field (MRF)**, is an undirected graph on $X$ that establishes certain local conditional independence assumptions with edges. By the Hammersley-Clifford theorem, a UGM is equivalently characterized by its *maximal* cliques $C$. It is parameterized by nonnegative **potential functions** $\psi_c(c(X)) \geq 0$ for all $c \in C$. The "unnormalized energy" of a particular configuration $x$ is given by $\prod_{c \in C} \psi_c(c(x))$. Using the normalization factor $Z = \sum_x \prod_{c \in C} \psi_c(c(x))$, the UGM defines

$$p(X) = \frac{1}{Z} \prod_{c \in C} \psi_c(c(X))$$

## 2 Generalized Marginalization

We focus on the problem of **marginalizing** $X' \subseteq X$ in an UGM. Marginalizing can be summation, maximization, or something else (see below). This also handles DGMs since we can write $\prod_{i=1}^{n} P(X_i|\mathbf{pa}(X_i))$ into an equivalent UGM, $\prod_{i=1}^{n} \psi_i(c_i(X))$, where each of the $n$ cliques $c_i(X) = \{X_i\} \cup \mathbf{pa}(X_i)$ has potential $P(X_i|\mathbf{pa}(X_i))$ (with the normalization factor $Z = 1$). The creation of new undirected edges between parents is called **moralization**.

### 2.1 Setup

We consider any operation $\oplus$ and $\otimes$ that form a **commutative semiring** (i.e., they are commutative and distributive with identity elements). Given $X' \subseteq X$ of size $m$,

the marginalization problem is posed as

$$\bigoplus_{x'} \bigotimes_{c \in C} \psi_c(c(X : X' = x')) \tag{1}$$

Note that a naive calculation looping through all possible configurations of $X'$ takes $O(K^m)$.

One common use of marginalization is to calculate a **marginal distribution**. In this case, $a \oplus b = a + b$ (identity 0) and $a \otimes b = ab$ (identity 1). Assuming $Z = 1$ for simplicity, the distribution over a subset $Y \subseteq X$ is given by summing over all possible configurations of $X' = X \backslash Y$.

$$p(Y) = \sum_{x'} p(Y, x') = \sum_{x'} \prod_{c \in C} \psi_c(c(X : X' = x'))$$

A slight variant of this problem can be used to calculate the log normalization factor $\log Z$ in log space. In this case, $a \oplus b = \text{logsumexp}(a, b) := \log(\exp(a) + \exp(b))$ (identity $-\infty$) and $a \otimes b = a + b$ (identity 0). Then

$$\log Z = \log \sum_{x} \prod_{c \in C} \psi_c(c(x)) = \operatorname*{logsumexp}_{x} \sum_{c \in C} \log \psi_c(c(x))$$

Another common use of marginalization is for **maximum a posteriori probability (MAP)** estimate. In this case, $a \oplus b = \max(a, b)$ (identity 0) and $a \otimes b = ab$ (identity 1). If $O$ and $H$ partition $X$ into observed and hidden variables, calculating the most probable configuration of $H$ with $O = o$ boils down to calculating

$$\max_h p(O = o, H = h) = \max_h \prod_{c \in C} \psi_c(c(O = o, H = h))$$

## 2.2 The Variable Elimination Algorithm

The variable elimination (VE) algorithm uses the fact that for any functions $f, g$ over discrete variables,

$$\bigoplus_a \bigoplus_b (f(a) \otimes g(b)) = \left( \bigoplus_a f(a) \right) \otimes \left( \bigoplus_b g(b) \right)$$

This follows from the commutative and distributive properties of $\oplus$ and $\otimes$. VE solves the generalized marginalization problem (1) in a potentially efficient way. Given an **elimination ordering** $X'_1 \dots X'_m$ of variables in $X'$, at each step it views the function $\bigotimes_{c \in C} \psi_c(c(X : X' = x'))$ as a product of a function $f$ of $X'_i$ and a function $g$ of $\neg X'_i = X' \backslash X'_i$. Then it uses the above fact and sums $f$ over $X'_i$:

$$\bigoplus_{x'_i} \bigoplus_{\neg x'_i} (f(x'_i) \otimes g(\neg x'_i)) = \left( \bigoplus_{x'_i} f(x'_i) \right) \otimes \left( \bigoplus_{\neg x'_i} g(\neg x'_i) \right)$$

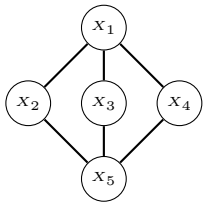
Importantly, $f(X'_i)$ can involve variables other than $X'_i$. For example, if $f(X'_i) = \psi(\{X'_i, X_1\}) \otimes \psi(\{X'_i, X_2\}) \otimes \psi(\{X'_i, X_3\})$, then eliminating $X'_i$ creates a new three-dimensional table $\bigoplus_{x'_i} f(x'_i)$ over all possible configurations of $X_1, X_2, X_3$.

**VE**
**Input**: UGM over $X$ with maximal cliques $C$ and potential functions $\psi_c(c(X))$, commutative semiring $(\oplus, \otimes)$, subset $X' \subseteq X$ of size $m$, elimination ordering $X'_1 \dots X'_m$ of variables in $X'$
**Output**: $\bigoplus_{x'} \bigotimes_{c \in C} \psi_c(c(X : X' = x'))$

1. For $i = 1 \dots m-1$,
   (a) Let $C_i$ denote the set of all current cliques that include $X'_i$ and let $D_i = C_i \setminus \{X'_i\}$.
   (b) Fully connect $D_i$ into a clique with potential
   $$\psi_i(D_i) := \bigoplus_{x'_i} \bigotimes_{c \in C_i} \psi_c(c(X : X'_i = x'_i))$$
   (c) Eliminate $X'_i$ from the graph.
2. Return $\bigoplus_{x'_m} \psi_c(c(X : X'_m = x'_m))$.

The asymptotic runtime of VE is $O(mK^d)$ where $d$ is the size of the largest clique induced in the elimination process. This is simply because it creates a table of $K^d$ entries (see the example below). The **induced width** of a UGM given an elimination ordering $X'_1 \dots X'_m$ is the size of the largest induced clique minus 1 (hence "width"). Unfortunately, finding an elimination ordering that has the minimum induced width is generally NP-hard (Arnborg et al., 1987).

**Example.** Consider the UGM




Each clique $(X_i, X_j)$ has potential $\psi_{ij}(X_i, X_j)$: this is just a table with $K^2$ entries. Say we want to calculate the normalization factor

$$Z = \sum_{x_1 \dots x_5} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4)\psi_{25}(x_2, x_5)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5)$$

which would take $O(K^5)$ time to naively enumerate all configurations. In contrast, applying VE on the elimination ordering $X_2, X_3, X_4, X_5, X_1$ looks like

$$\begin{aligned}
Z &= \sum_{x_3, x_4, x_5, x_1} \psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_2} \psi_{12}(x_1, x_2)\psi_{25}(x_2, x_5)\right)}_{\phi^2(x_1, x_5)} \\
&= \sum_{x_4, x_5, x_1} \psi_{14}(x_1, x_4)\psi_{45}(x_4, x_5)\phi^2(x_1, x_5) \underbrace{\left(\sum_{x_3} \psi_{13}(x_1, x_3)\psi_{35}(x_3, x_5)\right)}_{\phi^3(x_1, x_5)} \\
&= \sum_{x_5, x_1} \phi^2(x_1, x_5)\phi^3(x_1, x_5) \underbrace{\left(\sum_{x_4} \psi_{14}(x_1, x_4)\psi_{45}(x_4, x_5)\right)}_{\phi^4(x_1, x_5)}
\end{aligned}$$

which involves $K^3 + K^3 + K^3 + K^2 = O(K^3)$ operations. But applying VE on the elimination ordering $X_1, X_2, X_3, X_4, X_5$ looks like

$$
\begin{aligned}
Z &= \sum_{x_2,x_3,x_4,x_5} \psi_{25}(x_2,x_5)\psi_{35}(x_3,x_5)\psi_{45}(x_4,x_5) \underbrace{\left(\sum_{x_1} \psi_{12}(x_1,x_2)\psi_{13}(x_1,x_3)\psi_{14}(x_1,x_4)\right)}_{\psi^1(x_2,x_3,x_4)} \\
&= \sum_{x_3,x_4,x_5} \psi_{35}(x_3,x_5)\psi_{45}(x_4,x_5) \underbrace{\left(\sum_{x_2} \psi_{25}(x_2,x_5)\psi^1(x_2,x_3,x_4)\right)}_{\psi^2(x_3,x_4,x_5)} \\
&= \sum_{x_4,x_5} \psi_{45}(x_4,x_5) \underbrace{\left(\sum_{x_3} \psi_{35}(x_3,x_5)\psi^2(x_3,x_4,x_5)\right)}_{\psi^3(x_4,x_5)}
\end{aligned}
$$

which involves $K^4 + K^4 + K^3 + K^2 = O(K^4)$ operations. We can generalize this example to have $N$ nodes each with pairwise connections to $X_1$ and $X_5$: the first elimination ordering takes $O(NK^3)$ whereas the second elimination ordering takes $O(NK^{N+1})$.

**VE on trees (including chains).** VE has guaranteed runtime $O(mK^2)$ on trees because we can use a bottom-up traversal of nodes as our elimination ordering. This ordering ensures that each elimination step never introduces a new clique of size bigger than one, thus the size of the largest induced clique is $d = 2$. The $O(mK^2)$ forward algorithm in first-order HMMs is simply VE with the left-to-right elimination ordering of $m$ state nodes. The forward algorithm in second-order HMMs takes $O(mK^3)$ because the size of the largest induced clique is $d = 3$ (between $(H_{t-2}, H_{t-1}, H_t)$ after the DGM-to-UGM conversion).

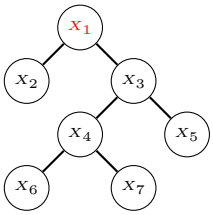
# 3 Belief Propagation

**Belief propagation (BP) on trees** is nothing but VE on trees while *caching* intermediate tables (which are always one-dimensional) in elimination steps and calling them **messages**. Once caching is done, we can easily obtain various marginal distributions from the messages.

This is best explained by example. Recall that a tree (undirected) is simply an acyclic connected graph $G = (V, E)$. Consider the following tree on 7 variables

$$
\begin{aligned}
p(X) = \frac{1}{Z}&\psi_1(X_1)\psi_2(X_2)\psi_3(X_3)\psi_4(X_4)\psi_5(X_5)\psi_6(X_6)\psi_7(X_7) \\
&\psi_{12}(X_1,X_2)\psi_{13}(X_1,X_3)\psi_{34}(X_3,X_4)\psi_{35}(X_3,X_5)\psi_{46}(X_4,X_6)\psi_{47}(X_4,X_7)
\end{aligned}
$$

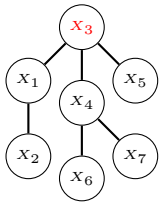
Suppose we want to calculate $p(X_1)$. We can treat $X_1$ as the "root" and run VE bottom-up as described above to marginalize out all "children" of $X_1$.



The VE algorithm looks like

$$
\begin{aligned}
p(X_1) &= \frac{1}{Z} \sum_{x_3, x_4} \psi_1(X_1)\psi_3(x_3)\psi_4(x_4)\psi_{13}(X_1, x_3)\psi_{34}(x_3, x_4) \\
&\quad \underbrace{\left(\sum_{x_2} \psi_2(x_2)\psi_{12}(X_1, x_2)\right)}_{\mu_{2\to 1}(X_1)} \underbrace{\left(\sum_{x_5} \psi_5(x_5)\psi_{35}(x_3, x_5)\right)}_{\mu_{5\to 3}(x_3)} \underbrace{\left(\sum_{x_6} \psi_6(x_6)\psi_{46}(x_4, x_6)\right)}_{\mu_{6\to 4}(x_4)} \underbrace{\left(\sum_{x_7} \psi_7(x_7)\psi_{47}(x_4, x_7)\right)}_{\mu_{7\to 4}(x_4)} \\
&= \frac{1}{Z} \sum_{x_3} \psi_1(X_1)\psi_3(x_3)\psi_{13}(X_1, x_3)\mu_{2\to 1}(X_1)\mu_{5\to 3}(x_3) \underbrace{\left(\sum_{x_4} \psi_{34}(x_3, x_4)\psi_4(x_4)\mu_{6\to 4}(x_4)\mu_{7\to 4}(x_4)\right)}_{\mu_{4\to 3}(x_3)} \\
&= \frac{1}{Z}\psi_1(X_1)\mu_{2\to 1}(X_1) \underbrace{\left(\sum_{x_3} \psi_3(x_3)\psi_{13}(X_1, x_3)\mu_{5\to 3}(x_3)\mu_{4\to 3}(x_3)\right)}_{\mu_{3\to 1}(X_1)} \\
&= \frac{1}{Z}\psi_1(X_1)\mu_{2\to 1}(X_1)\mu_{3\to 1}(X_1) \qquad (2)
\end{aligned}
$$

We call intermediate tables $\mu_{i\to j}(x_j)$ by the **message from** $i$ **to** $j$ which is a function of the subtree rooted at $i$ (away from $j$) over possible values of $X_j$. What if we now want to calculate $p(X_3)$? We can certainly treat $X_3$ as the root and run VE bottom-up again:




Doing VE yields

$$
p(X_3) = \frac{1}{Z}\psi_3(X_3)\mu_{5\to 3}(X_3)\mu_{4\to 3}(X_3) \underbrace{\left(\sum_{x_1} \psi_1(x_1)\mu_{2\to 1}(x_1)\psi_{13}(x_1, X_3)\right)}_{\mu_{1\to 3}(X_3)} \qquad (3)
$$

Note that we *reuse* much of the calculation in $p(X_1)$ when we calculate $p(X_3)$. The only new calculation is the message from 1 to 3, since we didn't need to pass message in that direction before.

Belief propagation on trees means precomputing

$$
\mu_{i\to j}(x_j) := \sum_{x_i} \psi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in \mathbf{neighbor}(i)\setminus\{j\}} \mu_{k\to i}(x_i)
$$

for every edge $(i, j) \in E$ (both ways) where $\mathbf{neighbor}(i) := \{k : (i, k) \in E\}$. This assumes that when $i$ is messaging $j$, $i$ already has messages from all neighbors $k$ except $j$. It is easy to see that this is guaranteed if we compute messages bottom-up and then top-down. Another way to achieve this effect is to arbitrarily initialize all messages and *in parallel* compute $\mu_{i\to j}(x_j)$ for every edge $(i, j) \in E$ (both ways) for

$D(G)$ times where $D(G)$ is the largest distance between any pair of nodes in $G$ (called the **diameter of** $G$).

Once all messages are precomputed, for any node $i \in V$ we can calculate

$$p(X_i) = \frac{1}{Z}\psi_i(X_i) \prod_{j \in \textbf{neighbor}(i)} \mu_{j \to i}(X_i)$$

For example, see (2) and (3). Since this is supposed to be a probability distribution, we can easily calculate the normalization factor by $Z = \sum_{x'_i} \psi_i(x'_i) \prod_{j \in \textbf{neighbor}(i)} \mu_{j \to i}(x'_i)$. We discuss two variants of this basic BP algorithm.

**Locally normalized messages.** We precompute messages with local normalization:

$$Z_{ij} := \sum_{x'_j} \left( \sum_{x_i} \psi_i(x_i)\psi_{ij}(x_i, x'_j) \prod_{k \in \textbf{neighbor}(i) \setminus \{j\}} \bar{\mu}_{k \to i}(x_i) \right)$$

$$\bar{\mu}_{i \to j}(x_j) := \frac{1}{Z_{ij}} \sum_{x_i} \psi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in \textbf{neighbor}(i) \setminus \{j\}} \bar{\mu}_{k \to i}(x_i)$$

Thus $\bar{\mu}_{i \to j} \in \Delta^{K-1}$ defines a probability distribution over the $K$ values of $X_j$ representing "prediction" of the value at $j$ based on the subtree rooted at $i$ (away from $j$). After precomputation, we can again calculate the marginal distribution at any $i \in V$ by locally normalizing with $\bar{\mu}$ since

$$p(X_i) = \frac{1}{Z}\psi_i(X_i) \left( \prod_{j \in \textbf{neighbor}(i)} \bar{\mu}_{j \to i}(X_i) \right) \underbrace{\left( \prod_{j \in \textbf{neighbor}(i)} Z_{ji} \right)}_{\text{constant wrt. } X_i} \propto \psi_i(X_i) \prod_{j \in \textbf{neighbor}(i)} \bar{\mu}_{j \to i}(X_i)$$

**Locally normalized messages in log space.** Note that

$$\nu_{i \to j}(x_j) = \operatorname*{logsumexp}_{x_i} \left( \log \psi_i(x_i) + \log \psi_{ij}(x_i, x_j) + \sum_{k \in \textbf{neighbor}(i) \setminus \{j\}} \log \bar{\mu}_{k \to i}(x_i) \right)$$

$$\log \bar{\mu}_{i \to j}(x_j) = \nu_{i \to j}(x_j) - \operatorname*{logsumexp}_{x'_j} \nu_{i \to j}(x'_j)$$

and

$$p(X_i) \propto \exp \left( \log \psi_i(x_i) + \sum_{j \in \textbf{neighbor}(i)} \log \bar{\mu}_{j \to i}(x_i) \right)$$

Thus we can do BP entirely in log probability space.

## 3.1 Loopy Belief Propagation

**Loopy belief propagation (LBP)** is BP applied on a non-tree graph (i.e., it has cycles) to approximate marginal probabilities. All messages are initialized to ones or random values in $[0, 1]$, and we keep updating messages until convergence (which may not happen). After enough iterations of LBP, we approximate $p(X_i)$ by normalizing $\psi_i(X_i) \prod_{j \in \textbf{neighbor}(i)} \mu_{j \to i}(X_i)$.

**References.**

- Concise slides on BP
- Lecture note with good BP examples