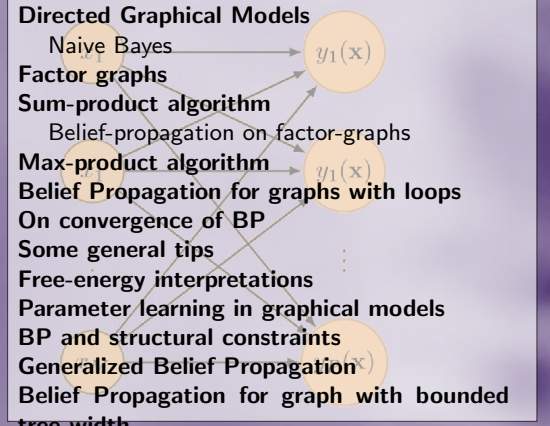


1 — Graphical Models

Daniel Khashabi [1]
KHASHAB2@ILLINOIS.EDU

1.1 Introduction

[TBW]

1.2 Undirected Graphical Models(Markov Random Fields)

Undirected graph $G = (V, E)$ with vertices on the set of variables $X = (X_1, \ldots, X_n)$. In this notation $n = |V|$ and $m = |E|$.

The graph shows the connection (interaction/dependence) between the variables in the model. Thus it is more accurate to model the set of variables connected to each other jointly. The most *accurate* modelling is defining a joint distribution for any variables in a connected graph. But such modelling strategy will create very big and cumbersome distribution which are hard to train, and do inference with. Thus we prefer to do approximations. One of such approximations is the *Markov property* in writing the joint distribution between the variables in the graph.

Definition 1.1 — Markov property in undirected graphical models. Let's say we have a graph $G = (V, E)$ in which the vertices represent the set of variables, and the edges represent the connection between the variables. The variables have the *Markov property* if for any two set of variables X_A and X_B (corresponding to the set of variables A and B respectively), if there is a set of variables X_S and S *separates* A and B, the variables X_A and X_B are independent, conditioned on X_S. Here "S *separates* A and B" means that, any path from any vertex in A to any vertex in B, needs to use at least one of the vertices in S. The

[1] This is part of my notes; to find the complete list of notes visit `http://web.engr.illinois.edu/~khashab2/learn.html`. This document is updated on November 26, 2013.

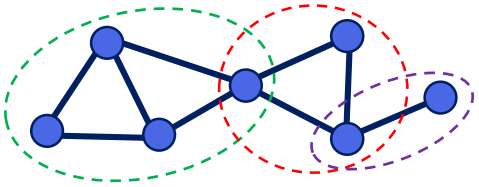


Figure 1.1: An undirected graphical models

mathematical notation for this definition is the following,

$$X_A \perp\!\!\!\perp X_B | X_S$$

Based on the above definition we can create the following two corollaries as a result of Makov property assumption.

Corollary 1.1 **Global Markov property:** For any two vertices which are not neighbours, given the other vertices they are independent:

$$X_a \perp\!\!\!\perp X_b | X_{V \setminus \{a,b\}} \Leftrightarrow (a, b) \notin E$$

Local Markov property: Any vertex is independent of all vertices, but its neighbours and itself, conditioned on all of its neighbours.

$$X_a \perp\!\!\!\perp X_{V \setminus \{a, \mathrm{ne}(a)\}} | X_{\mathrm{ne}(a)}$$

Note that the *local* Markov property is stronger that the *global* one.

To define proper factorization we need to be more careful. We need to define a factorization, as simple as possible, on the given input given graph, with respect to an input graph with its given edge connections, and obey the Markov independence property. Let's define some more mathematical definitions for graph properties. A *clique* C is a set of vertices, which are all connected to each other. For example in the Figure **??** there are three cliques are depicted (red, purple and green). The easiest way to define the factorization of distributions is to use cliques,

$$p(\mathbf{X}) = \prod_{C \in \mathrm{clique}(G)} p(\mathbf{X}_C). \tag{1.1}$$

This is a valid factorization since it has the *local* Markov property (proof?).

Definition 1.2 — Gibbs Random Field. When the probability factors defined for each of the factors of the Markov random field are strictly positive, we call it Gibbs random field.

Theorem 1.1 — HammersleyClifford theorem. A positive probability distribution with continuous density satisfies the pairwise Markov property with respect to an undirected graph G if and only if it factorizes according to G.

In other words, Hammersley-Clifford is saying that for any strictly positive distribution $p(.)$, the Markov property is equivalent to the factorization in Equation 1.1.

■ **Example 1.1 — Discrete Markov Random Field.** Let $G = (E, V)$ be an undirected graph. Each vertex is associated with a random variable $x_s \in \{0, 1, \dots, d-1\}$ Assume the following exponential density,

$$p(\mathbf{x}; \boldsymbol{\theta}) \propto \exp\left\{ \sum_{v \in V} \theta_v(x_v) + \sum_{(v,u) \in E} \theta_{vu}(x_v, x_u) \right\},$$

where the set of factors used in the definition of the above function are,

$$\begin{cases} \theta_v(x_v) = \sum_{j=0}^{d-1} \alpha_{v;j} \mathbf{I}_j(x_v) \\ \theta_{vu}(x_v, x_u) = \sum_{j=0}^{d-1} \sum_{k=0}^{d-1} \alpha_{vu;jk} \mathbf{I}_j(x_v) \mathbf{I}_k(x_u) \end{cases}$$

where $\mathbf{I}_.(.)$ is the indicator function,

$$\mathbf{I}_j(x_v) = \begin{cases} 1 & \text{if } x_v = j \\ 0 & \text{otherwise} \end{cases}$$

and the parameters of the model are

$$\boldsymbol{\theta} = \{\alpha_{v;j} \in \mathbb{R}, v \in V, j \in \mathcal{X}\} \cup \{\alpha_{vu;jk} \in \mathbb{R}, (v,u) \in E, (j,k) \in \mathcal{X} \times \mathcal{X}\}$$

. The cumulant generating function (log normalization function) is,

$$A(\boldsymbol{\theta}) = \log \sum_{x \in \mathcal{X}} \exp\left\{ \sum_{v \in V} \theta_v(x_v) + \sum_{(v,u) \in E} \theta_{vu}(x_v, x_u) \right\},$$

■

■ **Example 1.2 — Ising Model.** Ising Model is an special case of Discrete Markov Random Field, when the variable can get only two values $x_s \in \{0, 1\}$ $(d = 2)$. The probability distribution can be shown in a more simpler form,

$$p(\mathbf{x}; \boldsymbol{\theta}) \propto \exp\left\{ \sum_{v \in V} \theta_v x_v + \sum_{(v,u) \in E} \theta_{vu} x_v x_u \right\},$$

and the parameters of the model are,

$$\boldsymbol{\theta} = \{\theta_v \in \mathbb{R}, v \in V\} \cup \{\theta_{vu} \in \mathbb{R}, (v,u) \in E\}$$

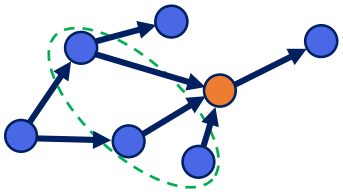


Figure 1.2: A directed graphical model; the parents of the node in orange is depicted in a green oval.

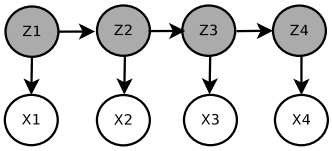


Figure 1.3: Structure of a Hidden Markov Model.

. In this exponential representation the vector is $\phi(\mathbf{x}) = [\dots x_v \dots x_{vu} \dots]^\top$, with length $|V| + |E|$. This exponential family is *regular* with $\Omega = \mathbb{R}^{|V|+|E|}$, and a *minimal* representation(if you don't know what each of these properties mean, refer to Wiki).

■

1.3 Directed Graphical Models

Directed graphical models are directed acyclic graphs (DAG), which accept factorization of distributions based on child-parent order,

$$p(\mathbf{X}) = \prod_{u \in V} p(u|\mathrm{pr}(u)),$$

where $\mathrm{pr}(u)$ is the set of parent vertices for u. An example directed graphical model is depicted in Figure **??**, in which is the set of parents for the orange vertex are denoted by a green oval.

■ **Example 1.3 — HMM.** An example for directed graphical models is Hidden Markov Models (HMM). The joint distribution for the observations and the latent variables shown in Figure 1.3 in an HMM is as following:

$$p(\mathbf{X}, \mathbf{Z}) = p(Z_1).p(Z_1|Z_1).p(Z_2|Z_1).p(X_2|Z_2).\dots. = p(X_1|Z_1)p(Z_1)\prod_{i=2}^{n} p(X_i|Z_i)p(Z_i|Z_{i-1}).$$

It should be clear that the above decomposition of probability distributions for an HMM exactly follows the definition of a directed graphical model. ■

1.3.1 Naive Bayes

In *Naive Bayes Classifier*, we assume that independence of input features, given the class, that is, we can write their joint distribution in the following way:

$$f(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n | G = g) = \prod_{i=1}^{n} f(\mathbf{X}_i | G = g)$$

Note that in the above formulation, $\mathbf{X}_i$ refer to each of input features, each of which could be of several dimension(and should not be confused with dimensions of one single feature, though in some cases they can be tough as the same thing). By applying this assumption in equation **??**, we have:

$$\begin{aligned}
\log \frac{\Pr(G = k | X = x)}{\Pr(G = l | X = x)} &= \log \frac{\pi_l f(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n | G = l)}{\pi_k f(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n | G = k)} \\
&= \log \frac{\pi_l \prod_{i=1}^{n} f(\mathbf{X}_i | G = l)}{\pi_k \prod_{i=1}^{n} f(\mathbf{X}_i | G = k)} \\
&= \log \frac{\pi_l}{\pi_k} + \sum_{i=1}^{n} \log \frac{f(\mathbf{X}_i | G = l)}{f(\mathbf{X}_i | G = k)}
\end{aligned}$$

1.4 Factor graphs

Factor graphs, unifies the directed and undirected representation of graphical models Frey (2002). As it is shown in Figure 1.4, there are two types of nodes, factor nodes, and variable nodes. In addition to vertices that represent random variables (as it was the case in Directed and Undirected Graphical Models), we introduce rectangular vertices to the representation. Two examples are depicted in Figure 1.4. The nodes stand for random variables, and the rectangles are the joint functions among the variables which they are connected to. These representation can give more accurate decomposition of probability factors for the variables. The fact that factor graphs combine undirected and directed graphical models is because, the function/distribution in the factors could be anything, e.g. the joint distribution or a conditional distribution. To have a general representation we show each function/distribution defined on each factor by $f(\mathbf{x})$ where $\mathbf{x}$ is the set of variables which are connected to that factor. For example for the factor graph in Figure 1.4, the expansion of the distribution is as following,

$$p(x_1, x_2, x_3) \propto f(x_1, x_2) f(x_2, x_3) f(x_1, x_3)$$

■ **Example 1.4 — HMM as a factor graph.** In Figure 1.5 an HMM model is shown. Let's define the short hand $\mathbb{P}(Y = y | X = x) = \mathbb{P}(y|x)$. The probability of the

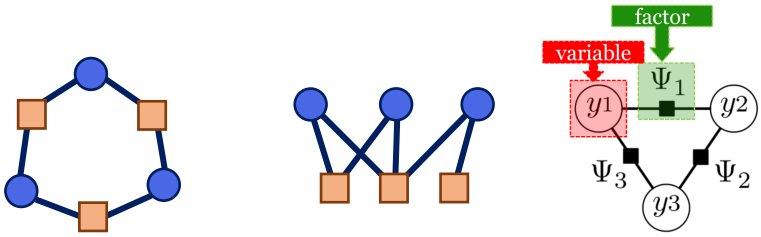


Figure 1.4: Representation of factor graphs. As it is shown in the right figure, there are two types of nodes, factor nodes, and variable nodes.

observations is,

$$\mathbb{P}(y|x) \propto \prod_i \mathbb{P}(y_i|x_i)\mathbb{P}(y_i|y_{i-1})$$

One can convert this into the factor graph, and eliminate the observed variables. The vertical factors $\mathbb{P}(y_i|x_i)$ represent the probability of observations and the horizontal factors $\mathbb{P}(y_i|y_{i-1})$ represent the probabilities of transitions. ■

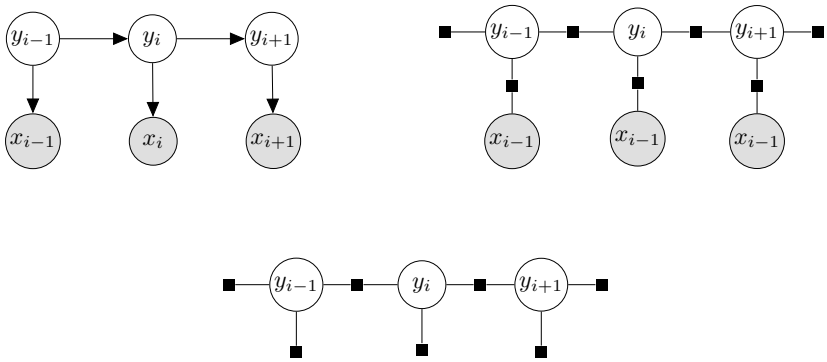


Figure 1.5: HMM in factor graph representation.

■ **Example 1.5 — Chain CRF as a factor graph.** In Figure 1.6 a Chain-CRF Lafferty et al. (2001) is represented. Assume the input space is $X = (X_1, \ldots, X_n)$. We know this CRF could be formulated as follows,

$$\mathbb{P}(y|x) \propto \exp\left(\sum_i w^\top f_i(y_i, x) + \sum_i w^\top f_i(y_i, y_{i-1}, x)\right)$$

The equivalent factor graph is also represented. The conversion is very simple. The vertical factor are $\exp\left(\sum_i w^\top f_i(y_i, x)\right)$ and the horizontal factors are $\exp\left(w^\top f_i(y_i, y_{i-1}, x)\right)$. In general any CRF can be shown as a factor graph. ■

■ **Example 1.6 — Dependency Parsing.** In McDonald et al. (2005) the dependency parsing problem is modelled as a factor graph. See Figure **??**. For each sentence, we have a triangle of variables, in which each variable can take 3 possible

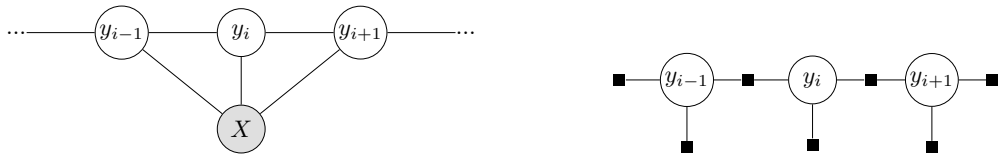


Figure 1.6: Chain-CRF as factor graph.

values:

$$y_i \in \{left, right, off\}$$

■

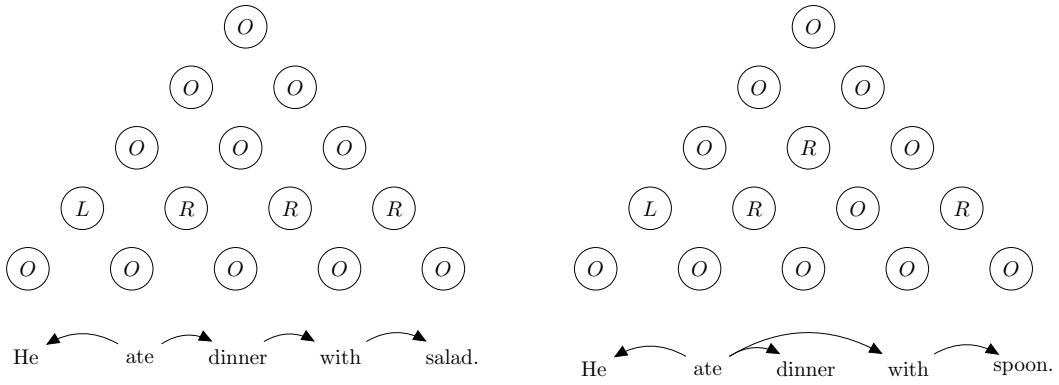


Figure 1.7: Dependency Parse for two example sentences.

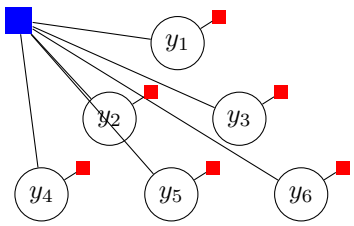


Figure 1.8: Modelling dependency parsing with a factor graph.

■ **Example 1.7 — Joint inference: sequence labelling.** In Figure 1.10 a joint sequence labelling is shown with a factor graph. ■

■ **Example 1.8 — Sentence Alignment.** 1 ■

1.5 Sum-product algorithm

Let's say we have a acyclic graph and we want to calculate the marginals. Let's say we have the following factorization on a an arbitrary acyclic graph,

Figure 1.9: Joint inference with factor graph.

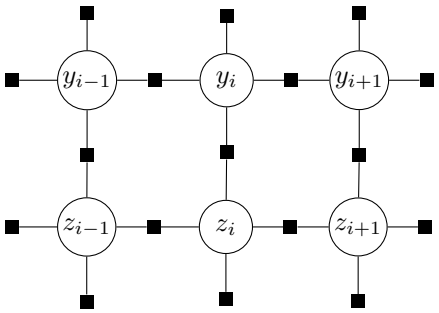


Figure 1.10: Joint inference with factor graph.

$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}} \prod_{s \in V} \psi(\mathbf{x}_s) \prod_{(u,v) \in E} \psi(\mathbf{x}_u, \mathbf{x}_v) \tag{1.2}$$

Let's say we are given a graph and we want to marginalize over a leaf variable $\mathbf{x}_v$ (figure 1.12(left)),

$$p(\mathbf{x}_{\setminus v}) \propto \sum_{\mathbf{x}_v} \prod_{s \in V} \psi(\mathbf{x}_s) \prod_{(a,b) \in E} \psi(\mathbf{x}_a, \mathbf{x}_b),$$

which can be simplified as,

$$p(\mathbf{x}_{\setminus v}) \propto \left(\prod_{s \in V \setminus \{v\}} \psi(\mathbf{x}_s) \prod_{(a,b) \in E,\ v \neq t} \psi(\mathbf{x}_a, \mathbf{x}_b) \right) . \sum_{\mathbf{x}_v} \left(\psi(\mathbf{x}_v) \prod_{(a,v) \in E} \psi(\mathbf{x}_a, \mathbf{x}_v) \right) .$$

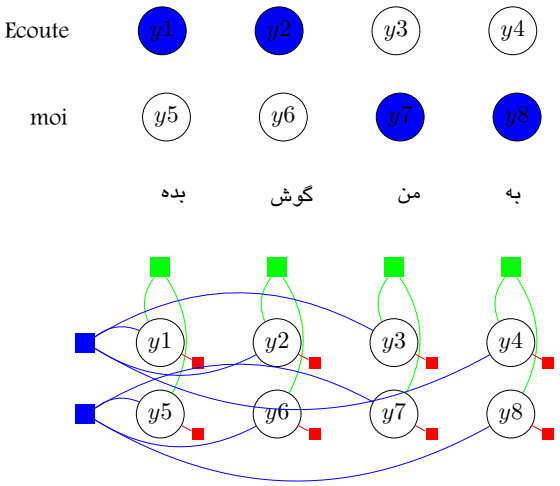


Figure 1.11: Alignment of sentences with factor graphs.

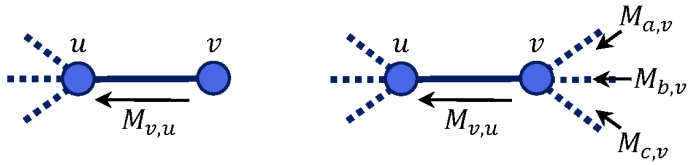


Figure 1.12: The elimination of a leaf node.

Like the case of Figure 1.12 if we assume that v (the leaf) has only one parent u we can simplify the equation more,

$$p(\mathbf{x}_{\setminus v}) \propto \left(\prod_{s \in V \setminus \{v\}} \psi(\mathbf{x}_s) \prod_{(a,b) \in E,\, v \neq t} \psi(\mathbf{x}_a, \mathbf{x}_b) \right) . \sum_{\mathbf{x}_v} \psi(\mathbf{x}_v) \psi(\mathbf{x}_u, \mathbf{x}_v). \tag{1.3}$$

Thus the marginalization over the leaf variables is equivalent to marginalization over the factors of the leaf variables, and the mutual factors with its parents, and multiplying the result to the rest of the product. This idea makes the problem of inference on acyclic graphs much easier. Instead of global summations (integrations) we only need local integrations by carefully ordering the variables. Now we make the notation more concrete by defining *message*. We define $M_{v,u}(\mathbf{x}_u)$ as the message from v(a leaf) to u, which is the marginalization over the sender's and mutual factors, and is only a function of the recipient.

$$M_{v \to u}(\mathbf{x}_u) = \sum_{\mathbf{x}_v} \psi(\mathbf{x}_v) \psi(\mathbf{x}_u, \mathbf{x}_v).$$

So, we can say that, when marginalization of leaf variables, we only need to keep the messages and through anything else away.

Let's say we have a more general case, as in the Figure 1.12(right) where we are marginalizing over a set of variables, including v which is not a leaf. For simplicity we can assume that the others are three leaf nodes, a, b, c. Since we want to marginalize over $\{v, a, b, c\}$, we can start by calculating the messages from the leaves $M_{a,v}(\mathbf{x}_v), M_{b,v}(\mathbf{x}_v)$ and $M_{c,v}(\mathbf{x}_v)$ and multiplying them into the joint distribution of the remaining variables. Now we only need to marginalize over v:

$$p(\mathbf{x}_{\setminus v}) \propto \left(\prod_{s \in V \setminus \{v\}} \psi(\mathbf{x}_s) \prod_{(a,b) \in E,\, v \neq t} \psi(\mathbf{x}_a, \mathbf{x}_b) \right) . \sum_{\mathbf{x}_v} \psi(\mathbf{x}_v) \psi(\mathbf{x}_u, \mathbf{x}_v) \underbrace{[M_{a \to v}(\mathbf{x}_v) M_{b \to v}(\mathbf{x}_v) M_{c \to v}(\mathbf{x}_v)]}_{\text{Messages from children}}.$$

The only difference between the above equation and Equation 1.3 is the last term which is the *messages from children*. In other words, to marginalize over a non-leaf variable v, we only need to calculate the messages from its children, then sum over,

- The messages from its children (here a, b and c)
- The factor on itself, v
- The joint factor of it with its parent u

Now we can give the the general definition of a message,

$$M_{v\to u}(\mathbf{x}_u) = \sum_{\mathbf{x}_v} \psi(\mathbf{x}_v)\psi(\mathbf{x}_u, \mathbf{x}_v) \prod_{a\in N(v)\setminus\{u\}} M_{a\to v}(\mathbf{x}_v).$$

In the above definition $N(t)$ is the set of all vertices which neighbour v. Based on the above message formula we can calculate the marginals as following,

$$p(\mathbf{x}_s) \propto \psi_t(\mathbf{x}_t) \prod_{t\in N(s)} M_{t\to s}(\mathbf{x}_s)$$

Corollary 1.2 Since any tree is a acyclic graph, the marginals can be calculated in one-time swiping the nodes of the tree, with proper ordering of the vertices. In other words, the sum-product algorithm can give the exact marginal distributions of every node in a tree in $O(n = |V|)$ complexity.

Consider Figure 1.13. In generals, for any vertex like y_i the beliefs come from the neighbouring subgraphs (in this case $\mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$) and are passed to another subgraph (in this cases $\mathcal{A}_1$).

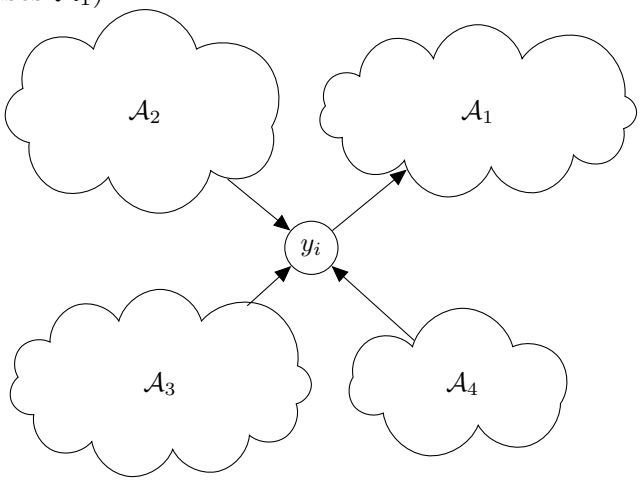


Figure 1.13: Transmission of belief from subgraphs.

1.5.1 Belief-propagation on factor-graphs

It is easy to generalize the belief-propagation scheme mentioned in the previous section to factor-graphs. There is a nice discussion of factor graphs could be found at Kschischang et al. (2001). The messages from variables to factors is,

$$M_{x_i\to f}(x_i) \triangleq \prod_{f'\in N(x_i)\setminus\{f\}} M_{f'\to x_i}(x_i).$$

And messages from factors to variables,

$$M_{f\to x_i}(x_i) \triangleq \sum_{\mathbf{x}\setminus x_i} f(\mathbf{x}) \prod_{x'\in N(f)\setminus\{x\}} M_{x'\to f}(x').$$

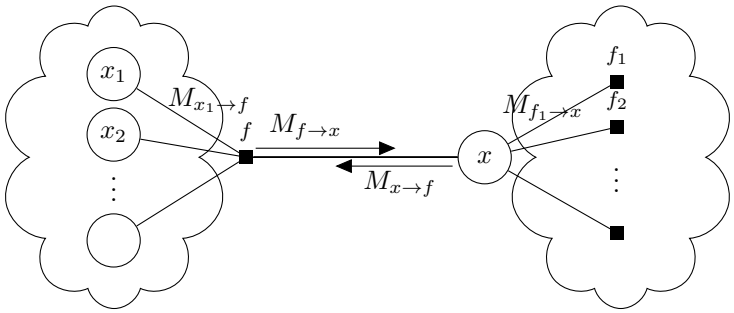


Figure 1.14: Message passing on a factor graph.

Note that in the above notation, $N(f)$ is the set of all *variables* adjacent to factor f, and $N(x)$ is the set of all *factors* adjacent to variable x. The belief propagation for factor graphs is depicted in Figure 1.14.

After the propagation of the *beliefs* is done, the beliefs (the probability distributions of each outcome of each variable) can be calculated by,

$$b(\mathbf{x}) \propto f(\mathbf{x}) \prod_{x' \in N(f)} M_{x' \to f}(x')$$

The marginalization can simply be written as follows,

$$b(x_i) = \sum_{\mathbf{x} \setminus x_i} b(\mathbf{x})$$

Usually people initialize $M_{f \to x_i} = 1$ and $M_{x_i \to f} = 1$, though if the graph has tree structure, the initialization does not have a lot of effect.

1.6 Max-product algorithm

Before anything, we show the necessity of finding max-probability for a set of random variables.

■ **Example 1.9 — Inference in graphical models.** Let's consider a general undirected graphical model with set of cliques $\mathcal{C}$,

$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}} \prod_{C \in \mathcal{C}} \psi(\mathbf{x}_C)$$

with a set of observation variables $\mathbf{x}_E$. We want to maximize the posterior probability of variables given some observations O ,

$$\theta = \arg\max_{\theta} p(\mathbf{x} = O)$$

■

The max-product algorithm is so similar to the sum-product algorithm we discussed. In the max-product algorithm we assume that we have a joint distribution of a group of variables, and we want to find its mode with respect to several variables. Let's say we have the same factorization as in Equation 1.2. It it easy to observe that *maximization* has distributive property on factors, like summation in the sum-product algorithm. Thus, if in the sum-product we substitute the summation with maximization, we will end-up with the desired algorithm,

$$M_{v\to u}(\mathbf{x}_u) = \max_{x'_v \in \mathcal{X}_v} \left\{ \psi(\mathbf{x}'_v)\psi(\mathbf{x}_u, \mathbf{x}'_v) \prod_{a \in N(v)\setminus\{u\}} M_{a\to v}(\mathbf{x}_v) \right\}.$$

And the *max-marginal* can be found using the following,

$$\tilde{p}(\mathbf{x}_u) \propto \psi_u(\mathbf{x}_u) \prod_{v \in N(u)} M_{v\to u}(\mathbf{x}_u)$$

Corollary 1.3 Since any tree is a acyclic graph, the max-marginals can be calculated in one-time swiping the nodes of the tree. In other words, the max-product algorithm can give the exact max-marginal distributions of every node in a tree in $O(n = |V|)$ complexity.

1.7 Belief Propagation for graphs with loops

In general in a graph with cycles, there might be cliques that create groups of vertices that are all connected to each other. In this case, unlike sampling, running the belief updates is not guaranteed to converge to the optimal answer. Also, unlike mean-field approaches, it is not guaranteed to converge to a value (even if that is not optimal). But in practice it might give good results, though it is mostly dependent on the structure of the graph.

■ **Example 1.10 — An example on loopy BP.** See the loopy binary graphical model in Figure 1.15. We want to perform BP step by step in this graph. Since the whole graph is a loop, there is no specific ordering. Since, it might take many iterations until convergence, we just perform several initial steps. We initialize the beliefs with[a],

$$\begin{cases} M_{f\to x_i}(x_i) = [0.5, \quad 0.5]^\top \\ M_{x_i\to f}(x_i) = [0.5, \quad 0.5]^\top \end{cases}$$

1. From x_1 to x_4:

$$M_{x_1\to f_{4,1}}(x_1) = M_{f_{1,2}\to x_1} = [0.5, \quad 0.5]^\top$$

$$M_{f_{4,1}\to x_4}(x_4) = \sum_{x_1} f(x_4, x_1) M_{x_1\to f_{4,1}}(x_1) = \left[\frac{4}{12}, \quad \frac{2}{12}\right]^\top$$

2. From x_4 to x_1:

$$M_{x_4\to f_{4,1}}(x_4) = M_{f_{3,4}\to x_4} = [0.5, \quad 0.5]^\top$$

$$M_{f_{4,1}\to x_1}(x_1) = \sum_{x_4} f(x_4, x_1) M_{x_4\to f_{4,1}}(x_4) = \left[\frac{4}{12}, \quad \frac{2}{12}\right]^\top$$

3. From x_1 to x_2:

$$M_{x_1\to f_{1,2}}(x_1) = M_{f_{4,1}\to x_1} = \left[\frac{4}{12}, \quad \frac{2}{12}\right]^\top$$

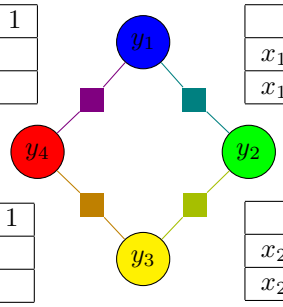
$$M_{f_{1,2}\to x_2}(x_2) = \sum_{x_1} f(x_1, x_2) M_{x_1\to f_{1,2}}(x_1) = \left[\frac{1}{22}, \quad \frac{5}{33}\right]^\top$$

We don't continue further. ∎

[a] A note on the notation: since the variables are binary, the function of one binary variable, could take only two possible value, which could be represented in a vector.

$f_{4,1}(x_4, x_1) =$

	$x_1 = 0$	$x_1 = 1$
$x_4 = 0$	$\frac{7}{12}$	$\frac{1}{12}$
$x_4 = 1$	$\frac{1}{12}$	$\frac{3}{12}$



	$x_2 = 0$	$x_2 = 1$
$x_1 = 0$	$\frac{1}{11}$	$\frac{1}{11}$
$x_1 = 1$	$\frac{1}{11}$	$\frac{8}{11}$

$= f_{1,2}(x_1, x_2)$

$f_{3,4}(x_3, x_4) =$

	$x_4 = 0$	$x_4 = 1$
$x_3 = 0$	$\frac{6}{12}$	$\frac{1}{12}$
$x_3 = 1$	$\frac{2}{12}$	$\frac{3}{12}$

	$x_3 = 0$	$x_3 = 1$
$x_2 = 0$	$\frac{1}{19}$	$\frac{9}{19}$
$x_2 = 1$	$\frac{8}{19}$	$\frac{1}{19}$

$= f_{2,3}(x_2, x_3)$

Figure 1.15: A loopy graphical model in factor-graph notation.

1.8 On convergence of BP

In general there is no guarantee for BP to converge. In Mooij and Kappen (2007, 2005) there is discussion on sufficient conditions for BP to converge.

1.9 Some general tips

Here are some general tips:

1. In a real-world problem, usually some of the factors are hard to compute. If there is a no order in computing the beliefs, try to compute the heavy beliefs (slowest message) less frequently.
2. In some structured problems when there needs to be some structure between the outcomes of the variables, doing the BP updates might be inefficient.

Instead one can use the structure of the problem, and perform some intermediary steps in the dynamic programming, to make the BP updates more efficient. There are nice examples of this in Smith and Eisner (2008); Martins et al. (2010); Cromieres and Kurohashi (2009); Burkett and Klein (2012).

3. You don't usually need to run until convergence. Many people previously have shown than, the results, even before the convergence, could be promising.

1.10 Free-energy interpretations

We know in tree-graphs, the BP procedure is exact. But how can we analyze its performance in general graphs?
In Yedidia et al. (2005, 2003) there is a good discussion on Free-Energy Interpretations of BP. To be more clear, they create an energy function for the whole graph and show that, the distributed BP updates, minimize this function. When the graph is tree, the updates will find its global minimum, but when it is not a tree, there is no guarantee to find the global minimum. But instead, this gives a very good tool to analyze the convergence properties of BP.

The energy needs to get smaller as we get better approximation for the original distribution. Consider that the original distribution is denoted by $p(\mathbf{x})$ and the approximate marginals are $b(\mathbf{x})$. To measure the distance between the distributions we can use the KL-divergence. One can assume the following form for the original distribution[2] :

$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}} \exp\{-E(\mathbf{x})/T\}$$

which is know as the Boltzman energy, which is minimized during time (or as the temperature T decreases). Here we just assume that $T = 1$ is a constant[3].

[To be written later!]

1.11 Parameter learning in graphical models

[TBW]

1.12 BP and structural constraints

[TBW]

1.13 Generalized Belief Propagation

[TBW]

1.14 Belief Propagation for graph with bounded tree-width

Even of the graphical model is not a tree, one can use its tree-like properties. in the large scale, to bypass the cyclicity of the graph. One way to do this, is to create a tree graph, from the original graph based on its tree-width. If we find

[2] This formalism is first introduced by physicists, and of course, they like their own stuff!
[3] In practice this parameter could be used for changing the scale of the the units for energy.

all the cliques in a graph, and group the vertices inside them together we will the *clique tree* of a graph, in which nodes are cliques of the graph. The separator sets are formed by the intersections of cliques adjacent in the graph.

Definition 1.3 — Running intersection property. A graph has the running intersection property if for any two cliques C_1 and C_2 for any unique paths joining them include the intersection nodes $C_1 \cap C_2$.

A clique tree with running intersection property is called a *junction tree*. This property is important because it ensures the probabilistic consistency of the calculations in a clique tree.

Definition 1.4 — Triangulated graph. A graph is triangulated (or chordal, rigid circuit) if each of its cycles of four or more nodes has an edge joining two nodes that are not adjacent in the cycle. Such an edge is called a *chord*.

Proposition 1.1 — ?. A graph has a junction tree if and only if it is triangulated (proof?).

The junction tree for exact inference is as following,

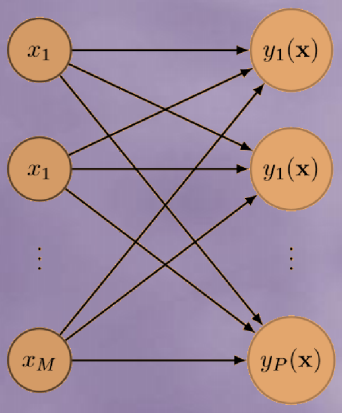
1. Given an undirected graph G form a triangulated graph $\tilde{G}$ by adding edges.
2. Form the clique graph (in which nodes are cliques of the triangulated graph).
3. Extract a junction tree (using a maximum weight spanning tree algorithm on weighted clique graph).
4. Run sum/max-product on the resulting junction tree.

Proposition 1.2 In a triangulated graph, there is an elimination ordering for G that does not lead to any added edges (proof?).

Theorem 1.2 If in a triangulated graph, for any adjacent cliques A and B, we weight all edges of a cliques with $|A \cap B|$, then the junction tree of the clique graph could be find by maximum-weight spanning tree in linear time (proof?).

1.15 Bibliographical notes

In preparation of this notes I have used Martin J. Wainwright's slides, and David Burkett's ACL tutorial . The nice paper Yedidia et al. (2005) is a very important resource for modelling BP in factor-graphs. Thanks to `https://github.com/jluttine/tikz-bayesnet`, many graphical models are drawn with this package.



Bibliography

David Burkett and Dan Klein. Fast inference in phrase extraction models with belief propagation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 29–38. Association for Computational Linguistics, 2012.

Fabien Cromieres and Sadao Kurohashi. An alignment algorithm using belief propagation and a structure-based distortion model. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 166–174. Association for Computational Linguistics, 2009.

Brendan J Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 257–264. Morgan Kaufmann Publishers Inc., 2002.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2): 498–519, 2001.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics, 2010.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005.

Joris M. Mooij and Hilbert J. Kappen. Sufficient conditions for convergence of loopy belief propagation. In F. Bacchus and T. Jaakkola, editors, *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 396–403, Corvallis, Oregon, 2005. AUAI Press.

Joris M. Mooij and Hilbert J. Kappen. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Transactions on Information Theory*, 53(12): 4422–4437, December 2007. doi: 10.1109/TIT.2007.909166.

David A Smith and Jason Eisner. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 145–156. Association for Computational Linguistics, 2008.

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003.

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005.