Graphical Models

Daniel Khashabi

Spring 2014
Last Update: March 19, 2015

1 Independence Rules

Here is a notation:

$$X - Y - Z$$

is equivalent to:

$$X \perp\!\!\!\perp Z|Y$$

which is also equivalent to

$$\mu(X, Y, Z) \propto f(X, Y)g(Y, Z) \text{ (for structly positive terms)}$$

Definition 1 (Graph Separation)**.** *C separates A and B if any path starting in A and terminating in B has at least one node in B.*

Definition 2 (Global Markov Property)**.** *A distribution $\mu(.)$ on $\mathcal{X}^V$ satisfies the Global Markov Property with respect to a graph G if for any partition (A, B, C) such that C separates A and B we have:*

$$\mu(x_A, x_B|x_C) \propto \mu(x_A|x_C)\mu(x_B|x_C)$$

Definition 3 (Local Markov Property)**.** *A distribution $\mu(.)$ on $\mathcal{X}^V$ satisfies the Global Markov Property with respect to a graph G if for any $i \in V$:*

$$\mu(x_i, x_{V\setminus i,\partial i}|x_{\partial i}) \propto \mu(x_i|x_{\partial i})\mu(x_{V\setminus i,\partial i}|x_{\partial i})$$

Definition 4 (Pairwise Markov Property)**.** *A distribution $\mu(.)$ on $\mathcal{X}^V$ satisfies the Global Markov Property with respect to a graph G if for any $i, j \in V$:*

$$\mu(x_i, x_j|x_{V\setminus i,j}) \propto \mu(x_i|x_{V\setminus i,j})\mu(x_j|x_{V\setminus i,j})$$

- Global Markov Property: **(G)**
- Local Markov Property: **(L)**
- Pairwise Markov Property: **(P)**

Lemma 1. *We have the following implications between the independence conditions:*

$$\textbf{\textit{(G)}} \Rightarrow \textbf{\textit{(L)}} \Rightarrow \textbf{\textit{(P)}}$$

The reverse implications are dependent on $\mu(.)$ being strictly positive.

Proof.

- Proof of **(G)** $\Rightarrow$ **(L)**: Easy to show, using just the definitions. Choose A and B to be i and ∂i.
- Proof of **(L)** $\Rightarrow$ **(P)**:

$$\begin{aligned} & x_i \perp\!\!\!\perp x_{V\setminus i,\partial i} | x_{\partial i} \\ \Rightarrow & x_i \perp\!\!\!\perp x_{V\setminus i,\partial i} | x_{V\setminus i,j} \\ \Rightarrow & x_i \perp\!\!\!\perp x_j | x_{V\setminus i,j} \end{aligned}$$

- Proof of **(P)** $\Rightarrow$ **(G)**: Proof with induction.

□

Remark 1. *In general **(P)** $\Rightarrow$ **(G)** does not hold. It holds if the following condition holds for <u>all</u> disjoint subsets A, B, C and $D \subset V$:*

$$x_A \perp\!\!\!\perp x_B | x_C, x_D \text{ and } x_A \perp\!\!\!\perp x_C | x_B \cup x_D \text{ then } x_A \perp\!\!\!\perp (x_B, x_C) | x_D$$

For strictly positive $\mu(.)$ the above holds.

Remark 2. *In general a distribution that is pairwise Markov with respect to G, does not imply that it is locally Markov with respect to the same graph.*

1.1 Independence Map (I-map)

Let

- $\mathcal{I}(G)$ denote the set of all conditional independencies implied by graph G.
- $\mathcal{I}(\mu)$ denote the set of all conditional independencies implied by a distribution $\mu(.)$.

Following the definition of the Local and Pairwise Markov property we can define $\mathcal{I}_L(\mu)$ and $\mathcal{I}_P(\mu)$. Again, similar to Lemma 1 for general distributions, $\mathcal{I}_P(\mu)$ is a strict subset of $\mathcal{I}_L(\mu)$, and $\mathcal{I}_L(\mu)$ is a strict subset of $\mathcal{I}_G(\mu) = \mathcal{I}(\mu)$. These sets are exactly the same if $\mu(.)$ is strictly positive.

G is an *I-map* of μ if μ satisfies all of the independencies of the graph G, i.e.

$$\mathcal{I}(G) \subset \mathcal{I}(\mu)$$

Remark 3. *A complete graph is always an I-map, since it does not express any independence in the graph. Therefore $\mathcal{I}(G) = \oslash$. It can then be observed that removing edges adds more independencies to $\mathcal{I}(G)$. In the case when the graph G has no elements, $\mathcal{I}(G)$ contains all possible independencies between vertecies without any conditioning.*

Following Remark 3 we define the graph G which has the biggest number of independencies, but still an I-map. A graph G is the minimal I-map for $\mu(x)$ if it is an I-map and removing a single edge from G causes the graph no longer be an I-map.

Example 1. *Consider the following distribution:*

$$\mathbb{P}(x_1 = x_2 = x_3 = x_4 = 1) = 0.5,\ \mathbb{P}(x_1 = x_2 = x_3 = x_4 = 0) = 0.5$$

The graphs in 1 (left and middle) are minimal I-maps. This can be proven by showing that, the graphs are I-maps and by removing any edges, we add independence rules which are not expressed by the distribution.

Here is a Directed Graph construction of minimal I-maps:

1. Choose an arbitrary ordering $x_1, \ldots, x_n$.
2. Choose a direct graph based on Bayes rule:

$$\mu(x) = \mu(x_1)\mu(x_2|x_1)\mu(x_3|x_1, x_2)...\mu(x_n|x_1, x_2...x_{n-1})$$

3. For each i choose its parent $\pi(i)$ to be the minimal subset of $\{x_1, \ldots, x_{i-1}\}$ such that

$$x_i - x_{\pi(i)} - \{x_1, ..., x_{i-1}\} \setminus x_{\pi(i)}$$

The following is the construction of an undirected graphical model. The construction gives a unique solution if $\mu(.) > 0$.

1. Create a complete graph.
2. Remove edge $e_{ij} \in E$ if i and j are conditionally independent given the rest of the nodes.

Example 2. *Consider the distribution in Example 1. If we construct a graph based on pairwise independence rule of the distribution we will end up with the graph in Figure 1 (right), which has is not an I-map. In other words, the construction does not give an I-map.*

The same example shows that pairwise Markov property does not imply local Markov. Although 1 (right) is locally Markov, but clearly, since no node has no neighbors, the pairwise independence (which is conditionals on neighbors) do not hold.

A graph G is called a *Perfect Map* for $\mu(x)$ if $\mathcal{I}(G) = \mathcal{I}(\mu)$.

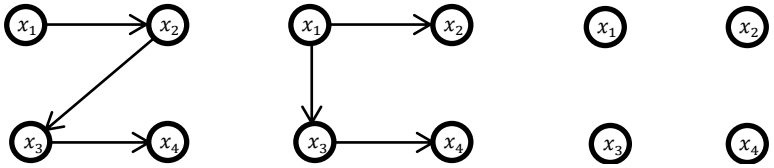


Figure 1: Example graphs.

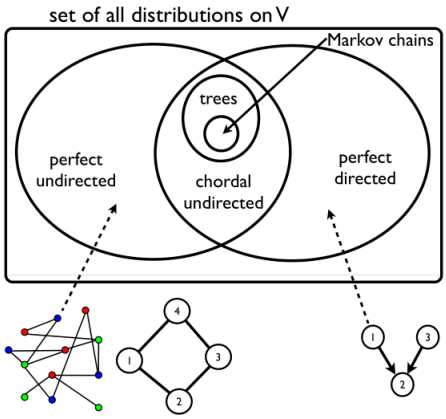


Figure 2: Different classes of perfect maps.

1.2 Moralization

Moralization is the procedure of converting a Bayesian network D to an MRF G such that $\mathcal{I}(G) \subset \mathcal{D}$. It can be proven that the resulting G is a minimal (undirected) I-map of $\mathcal{I}(D)$.

Lemma 2. *There exists a graph G such that $\mathcal{I}(D) = \mathcal{G}$ if and only if morelization does not add any edges.*

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

Figure 3: An example for Moralization of the a graph.

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 5 (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 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. ***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.*

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

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 2 there are three cliques are depicted (red, purple and green). The easiest way to define the factorization of distributions is to use cliques,

$$p(\mathbf{X}) = \prod_{C \in \text{clique}(G)} p(\mathbf{X}_C). \tag{1}$$

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

Definition 6 (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 (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.

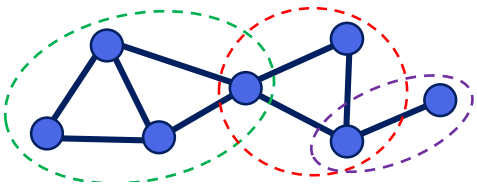


Figure 4: An undirected graphical models

Example 3 (Discrete Markov Random Field)**.** *Let $G = (E, V)$ be an undirected graph. Each vertex is associated with a random variable $x_s \in \{0, 1, \ldots, d-1\}$ 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 $\mathbf{I}_{\cdot}(.)$ 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 4 (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,

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

. In this exponential representation the vector is $\phi(\mathbf{x}) = [\ldots x_v \ldots x_{vu} \ldots]^\top$, 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).

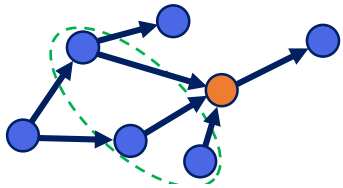


Figure 5: A directed graphical model; the parents of the node in orange is depicted in a green oval.

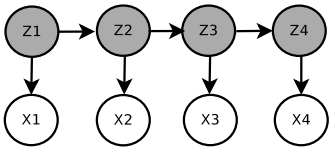


Figure 6: Structure of a Hidden Markov Model.

3 Directed Graphical Models

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

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

where $\text{pr}(u)$ is the set of parent vertices for u. An example directed graphical model is depicted in Figure 3, in which is the set of parents for the orange vertex are denoted by a green oval.

Example 5 (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 6 in an HMM is as following:*

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

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

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:

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

Example 6. *We know one intuition behind Gaussian BP computing the minimum of a quadratic function:*

$$\hat{x} = \arg\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\langle x, Qx\rangle + \langle b, x\rangle\right\} \tag{2}$$

for some positive definite $Q \in \mathbb{R}^{n\times n}$.

This can be extended to a more general case, when Q is not positive definite but full rank. In this case we can still define the inverse and therefore inversion $\hat{x} = -Q^{-1}x$ is well-defined (which is stationary/saddle point of the above quadratic program). The claim is that BP can still compute the correct result in this case.

We consider a specific model:

$$y = As_0 + w_0$$

where $s_0 \in \mathbb{R}^n$ is an unknown signal, $y \in \mathbb{R}^m$ is the vector of observations, $A \in \mathbb{R}^{m\times n}$ is the vector of observations, and $w_0 \in \mathbb{R}^m$ is the vector of i.i.d. Gaussian noise with entries $w_{0,i} \sim \mathcal{N}(0,\sigma^2)$. The goal is to reconstruct s_0 and w_0 given A and y.

A popular way to do this using Ridge Regression:

$$\hat{s} = \arg\min_{s\in\mathbb{R}^n}\left\{\frac{1}{2}\|y-As\|_2^2 + \frac{1}{2}\lambda\|s\|_2^2\right\} \tag{3}$$

Drawing inspiration from the Ridge model, we define the following loss function:

$$C_{A,y}(x=(z,s)) = -\frac{1}{2}\|z\|_2^2 + \frac{1}{2}\lambda\|s\|_2^2 + \langle z, y-As\rangle \tag{4}$$

1. *We simplify the cost in Eq 4 and write in the form of Eq 2:*

$$C_{A,y}(x=(z,s)) = -\frac{1}{2}\begin{bmatrix}z^\top, s^\top\end{bmatrix} Q \begin{bmatrix}z\\ s\end{bmatrix} + b^\top\begin{bmatrix}z\\ s\end{bmatrix}$$

where

$$Q = \begin{bmatrix} I_m & A \\ A^\top & \lambda I_n \end{bmatrix}, \quad b = \begin{bmatrix} y \\ 0_{n\times 1} \end{bmatrix} \tag{5}$$

2. *Since we have proven that have proven that 4 could be written in the form of 2,*

$$\hat{x} = (\hat{z}^\top, \hat{s}^\top)^\top = -Q^{-1}b$$

where Q and b are defined in 5. Thus, we just replace their definition and find $\hat{s}$:

$$\hat{x} = \begin{bmatrix} \hat{z} \\ \hat{s} \end{bmatrix} = -\begin{bmatrix} I_m & A \\ A^\top & -\lambda I_n \end{bmatrix}^{-1} \begin{bmatrix} y \\ 0_{n\times 1} \end{bmatrix}$$

Since we want $\hat{s}$, we can use use blockwise inversion trick:

$$\hat{s} = (\lambda I_n + A^\top A)^{-1} A^\top y = \lambda^{-1} A^\top (I_m + \lambda^{-1} A A^\top)^{-1} y$$

To verify that it corresponds to the solution of the Ridge regression it is enough to find the minimizer of the Ridge directly:

$$f(s) = \frac{1}{2}\|y - As\|_2^2 + \frac{1}{2}\lambda \|s\|_2^2$$

Then

$$\nabla_s f(s) = A^\top (y - As) + \lambda s = 0 \Rightarrow (A^\top A + \lambda I_n) s = A^\top y \Rightarrow s_{ridge} = (A^\top A + \lambda I_n)^{-1} A^\top y$$

which proves the equivalence between $\hat{s}$ and s_{ridge}.

3. *To solve this we can use the similar updates for Gaussian BP. Suppose dimension of Q is $n \times n$. Then the updates are:*

$$\begin{cases} b_{i\to j} = b_i - \sum_{k\in[n]\setminus\{j\}} \frac{Q_{ik} b_{k\to i}}{Q_{k\to i}} \\ Q_{i\to j} = Q_{ii} - \sum_{k\in[n]\setminus\{j\}} \frac{Q_{ik}^2}{Q_{k\to i}} \end{cases}$$

4. *The proof of convergence is followed by the proof of convergence of standard Gaussian BP which is done on a computation tree, by showing that that BP convergences with at most two times the tree-width of the computation tree iterations (if it converges to any fixed point).*

4 Factor graphs

Factor graphs, unifies the directed and undirected representation of graphical models [3]. As it is shown in Figure 7, 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 7. 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 7, the expansion of the distribution is as following,

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

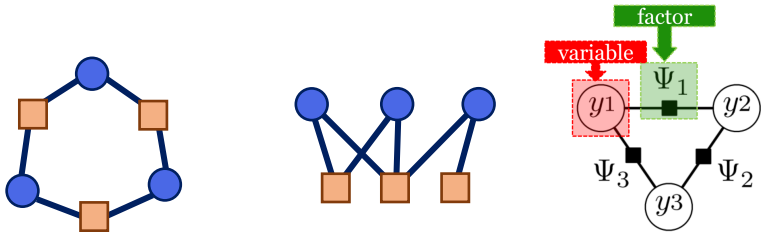


Figure 7: Representation of factor graphs. As it is shown in the right figure, there are two types of nodes, factor nodes, and variable nodes.

Example 7 (HMM as a factor graph)**.** *In Figure 8 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 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.*

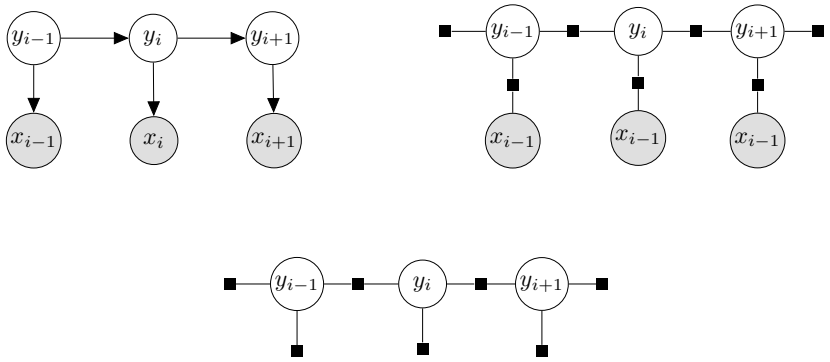


Figure 8: HMM in factor graph representation.

Example 8 (Chain CRF as a factor graph)**.** *In Figure 9 a Chain-CRF [5] 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.*

Figure 9: Chain-CRF as factor graph.

Example 9 (Dependency Parsing)**.** *In [7] 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 values:*

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

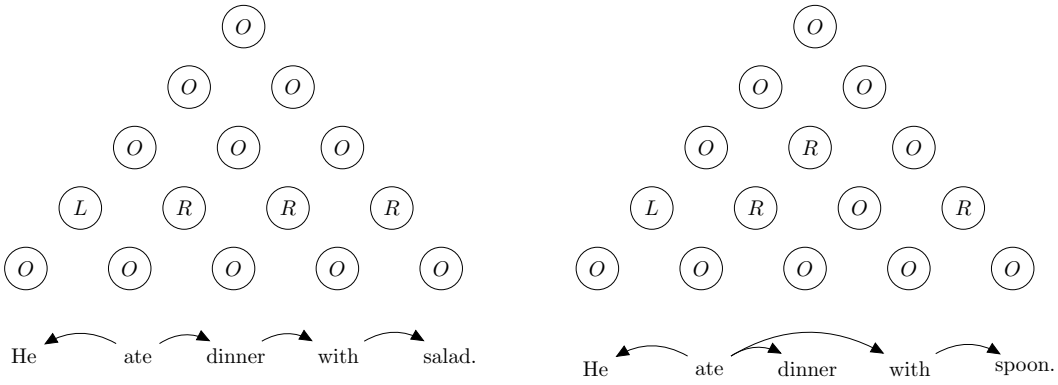


Figure 10: Dependency Parse for two example sentences.

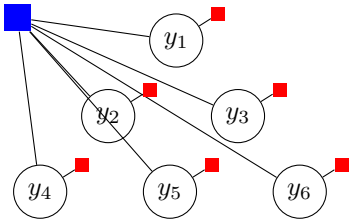


Figure 11: Modelling dependency parsing with a factor graph.

Example 10 (Joint inference: sequence labelling)**.** *In Figure 12 a joint sequence labelling is shown with a factor graph.*

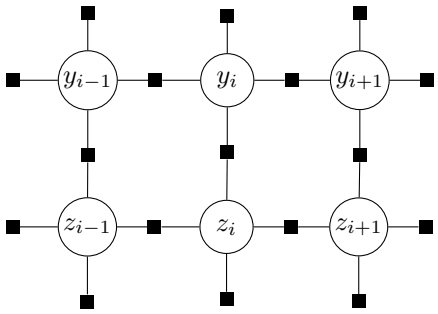


Figure 12: Joint inference with factor graph.

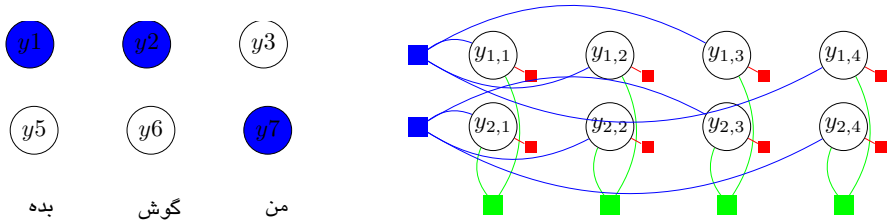


Figure 13: Alignment of sentences with factor graphs.

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,

$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}} \prod_{s \in V} \psi(\mathbf{x}_s) \prod_{(u,v) \in E} \psi(\mathbf{x}_u, \mathbf{x}_v) \tag{6}$$

Let's say we are given a graph and we want to marginalize over a leaf variable $\mathbf{x}_v$ (figure 14(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,

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

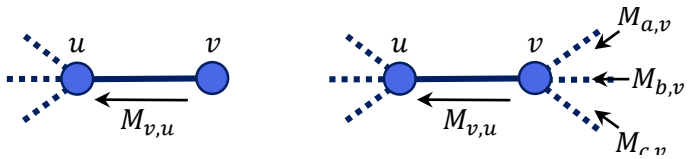


Figure 14: The elimination of a leaf node.

Like the case of Figure 14 if we assume that v (the leaf) has only one parent u we can simplify the equation more,

$$p(\mathbf{x}_{\setminus v}) \propto \left(\prod_{s \in V \setminus \{v\}} \psi(\mathbf{x}_s) \prod_{(a,b) \in E,\ v \neq t} \psi(\mathbf{x}_a, \mathbf{x}_b) \right) . \sum_{\mathbf{x}_v} \psi(\mathbf{x}_v) \psi(\mathbf{x}_u, \mathbf{x}_v). \tag{7}$$

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 14(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 7 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 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 15. 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 $\mathcal{A}_1$).

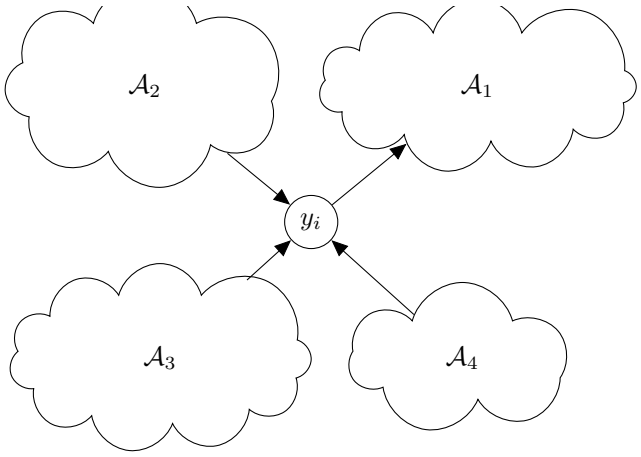


Figure 15: Transmission of belief from subgraphs.

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 [4]. 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,

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

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 16.

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,

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

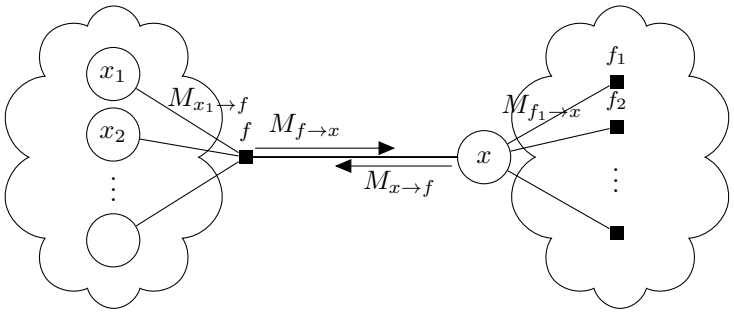


Figure 16: Message passing on a factor graph.

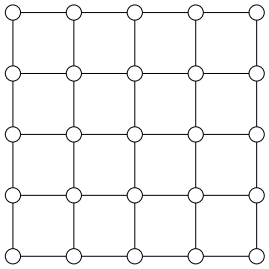


Figure 17: A grid of size $l \times l$. An Ising model is define on such grid.

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.

Example 11 (BP for Ising Model)**.** *For $l \in \mathbb{N}$ let graph $G_l = (V_l, E_l)$ be an $l \times l$ grid (Figure 17). The parameters for this Ising model is $\Theta = \{\theta_{ij}, \theta_i : (i,j) \in E_l, i \in V_l\}$. The joint probability distribution over $x \in \{-1, +1\}^{|V_l|}$ is defined as:*

$$\mu(x) = \frac{1}{Z_G} \exp\left\{ \sum_{(i,j) \in E_l} \theta_{ij} x_i x_j + \sum_{i \in V_l} \theta_i x_i \right\}$$

The belief update (up to a constant) is:

$$\nu_{i \to j}^{(t+1)}(x_i) \propto \exp\{\theta_i x_i\} \prod_{k \in \partial i \setminus j} \left\{ \sum_{x_k} \exp\{\theta_{ik} x_i x_j\} \, \nu_{k \to i}^{(t)}(x_k) \right\}$$

Since the updates can take two possible values, writing the update in the form of log-likelihood ratio

might make sens:

$$
\begin{aligned}
L_{i\to j}^{(t+1)} &= \frac{1}{2}\log\left(\frac{\nu_{i\to j}^{(t)}(+1)}{\nu_{i\to j}^{(t)}(-1)}\right) \\
&= \frac{1}{2}\log\left(\frac{\exp\{+\theta_i\}\prod_{k\in\partial i\setminus j}\left\{\sum_{x_k}\exp\{+\theta_{ik}x_k\}\,\nu_{k\to i}^{(t)}(x_k)\right\}}{\exp\{-\theta_i\}\prod_{k\in\partial i\setminus j}\left\{\sum_{x_k}\exp\{-\theta_{ik}x_k\}\,\nu_{k\to i}^{(t)}(x_k)\right\}}\right) \\
&= \theta_i + \frac{1}{2}\sum_{k\in\partial i\setminus j}\log\left(\frac{\sum_{x_k}\exp\{+\theta_{ik}x_k\}\,\nu_{k\to i}^{(t)}(x_k)}{\sum_{x_k}\exp\{-\theta_{ik}x_k\}\,\nu_{k\to i}^{(t)}(x_k)}\right) \\
&= \theta_i + \frac{1}{2}\sum_{k\in\partial i\setminus j}\log\left(\frac{\exp\{+\theta_{ik}\}\,\nu_{k\to i}^{(t)}(+1)+\exp\{-\theta_{ik}\}\,\nu_{k\to i}^{(t)}(-1)}{\exp\{-\theta_{ik}\}\,\nu_{k\to i}^{(t)}(+1)+\exp\{+\theta_{ik}\}\,\nu_{k\to i}^{(t)}(-1)}\right) \\
&= \theta_i + \frac{1}{2}\sum_{k\in\partial i\setminus j}\log\left(\frac{\exp\{+\theta_{ik}\}\exp 2L_{k\to i}^{(t)}+\exp\{-\theta_{ik}\}}{\exp\{-\theta_{ik}\}\exp 2L_{k\to i}^{(t)}+\exp\{+\theta_{ik}\}}\right) \\
&= \theta_i + \frac{1}{2}\sum_{k\in\partial i\setminus j}\log\left(\frac{\exp\left\{+\theta_{ik}+L_{k\to i}^{(t)}\right\}+\exp\left\{-\left(\theta_{ik}+L_{k\to i}^{(t)}\right)\right\}}{\exp\left\{-\left(\theta_{ik}-L_{k\to i}^{(t)}\right)\right\}+\exp\left\{\theta_{ik}-L_{k\to i}^{(t)}\right\}}\right)
\end{aligned}
$$

To simplify the final result, we use the following facts:

$$
\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},\ \frac{1}{2}\log z = \arctan\left(\frac{z-1}{z+1}\right)
$$

If we define:

$$
z \triangleq \frac{\exp\left\{+\theta_{ik}+L_{k\to i}^{(t)}\right\}+\exp\left\{-\left(\theta_{ik}+L_{k\to i}^{(t)}\right)\right\}}{\exp\left\{-\left(\theta_{ik}-L_{k\to i}^{(t)}\right)\right\}+\exp\left\{\theta_{ik}-L_{k\to i}^{(t)}\right\}}
$$

Then it can be shown that:

$$
\begin{aligned}
\frac{z-1}{z+1} &= \frac{\exp\{+\theta_{ik}\}-\exp\{-\theta_{ik}\}}{\exp\{+\theta_{ik}\}+\exp\{-\theta_{ik}\}}\cdot\frac{\exp\left\{+L_{k\to i}^{(t)}\right\}-\exp\left\{-L_{k\to i}^{(t)}\right\}}{\exp\left\{+L_{k\to i}^{(t)}\right\}+\exp\left\{-L_{k\to i}^{(t)}\right\}} \\
&= \tanh(\theta_{ik})\tanh(L_{k\to i}^{(t)})
\end{aligned}
$$

Which would give the following result:

$$
L_{i\to j}^{(t+1)} = \theta_i + \sum_{k\in\partial i\setminus j}\arctan\left(\tanh(\theta_{ik})\tanh(L_{k\to i}^{(t)})\right)
$$

Here we do some experiments on these updates. Suppose the following is our criterion for convergence:

$$
\Delta(t) = \frac{1}{|\overrightarrow{E}_l|}\sum_{(i,j)\in E_l}\left|\nu_{i\to j}^{(t+1)}(+1)-\nu_{i\to j}^{(t)}(+1)\right|
$$

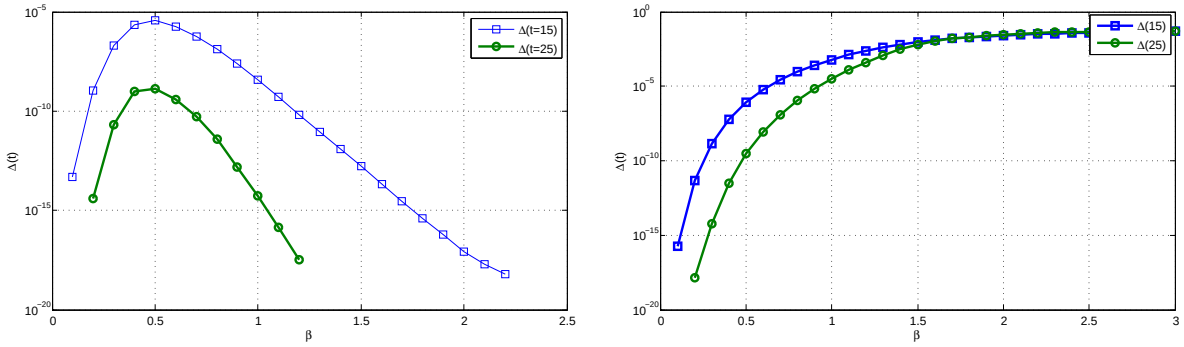


Figure 18: Convergence of BP for Ising model. Left: Initialization with random samples from $[0, \beta]$. Right: Initialization with random samples from $[-\beta, \beta]$.

where $\overrightarrow{E}_l$ *the set of directed edges and* $|\overrightarrow{E}_l| = 2|E_l|$. *We plot the convergence for* $\Delta(t = 15)$ *and* $\Delta(t = 25)$, *with uniformly random initialization of parameters in* $[0, \beta]$, *in Figure 18 (left), as function of* β. *It can be observed that for medium values of* β *the initialization is slower.*

If we initialize the parameters randomly from $[-\beta, \beta]$ *the convergence is depicted in Figure 18 (right). In this case the convergence for small values of* β *is relatively fast. Although for big* β*s the convergence gets very slow.*

Lemma 3. *The (parallel) sum-product algorithm converges at most in diameter of the graph iterations. (Note: diameter of the graph is the length of the longest path in a graph.)*

Proof. The goal is to inductively prove that the (parallel) sum-product algorithm will not iterate more than D, the diameter of the graph (where diameter of a graph, is defined to be the length of the maximum path in the graph.)

The proof is done by induction on the size of the diameter and a few additional steps at the end. Basically, given a tree T, we construct another tree T' which has diameter one less than T. To construct T', we strip down the leaves of the tree T, and replace the unary potentials with modified potentials:

$$\psi'_i(x_i) = \psi_i(x_i) \prod_{k \in \partial i \cap L} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu^{(0)}_{k \to i}(x_k) \right\} \tag{8}$$

and the rest of the potential functions remain the same (Figure 19). The goal is to prove that, if we run two separate BPs, one on T and one on T', with initialization $\nu'^{(0)}_{i \to j}(x_i) = \nu^{(1)}_{i \to j}(x_i)$, then:

$$\nu'^{(t)}_{i \to j}(x_i) = \nu^{(t+1)}_{i \to j}(x_i), \text{ for all } (i, j) \in E', \forall t \geq 0 \tag{9}$$

We know that the sum-product is

$$\nu^{(t+1)}_{i \to j}(x_i) = \psi_i(x_i) \prod_{k \in \partial i \setminus j} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu^{(t)}_{k \to i}(x_k) \right\}$$

The most important thing to observe is that, for any outgoing message from leave i in T', $\nu'^{(t)}_{i\to j}(x_i)$ will be the same as the corresponding message on the same edge in T, $\nu^{(t+1)}_{i\to j}(x_i)$, given that all of the incoming messages are the same. This can be proved with induction on time.

When it is $t = 0$ based on the initialization, the outgoing message (i, j) (Figure 19) is:

$$\nu^{(1)}_{i\to j}(x_i) = \psi_i(x_i) \prod_{k\in\partial i\setminus j} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k)\nu^{(0)}_{k\to i}(x_k) \right\}$$

and the corresponding message $\nu'^{(1)}_{i\to j}(x_i)$ in T' is the same as its unary potential, which based on the definition of the modified unary potential in Equation 8 is the same as $\nu^{(0)}_{i\to j}(x_i)$. Note that the equality between $\nu'^{(1)}_{i\to j}(x_i)$ and $\nu^{(0)}_{i\to j}(x_i)$ for non-leaf vertices trivially holds, as their potentials are not changed.

Now suppose (9) holds for t and we want to prove it for $t + 1$. With a similar argument as the previous case, the outgoing message from $i \in L'$ is the same as its counterpart message in T: $\nu'^{(t)}_{i\to j}(x_i) = \nu^{(t+1)}_{i\to j}(x_i)$. Also note that, based on the assumption of the base case for $t = 1$, the equality of (9) holds for non-leaf messages.

Another step of the argument is to argue that the diameter of T' is strictly less than $D - 1$ (D is diameter of T). The point to be noticed is that, the longest path in a tree, essentially passes through a leaf node (proof by contradiction. Suppose it is not the case. Specifically there is a non-leaf node at one end of the longest path. In that case, the path can be extended to one of the leaves connected to this node.)

Based on the construction, at each step we remove the leaves of the tree and therefore, essentially the diameter of the new tree is (at least) one less than the diameter of the original tree.

Since each time that we construct a modified tree, the diameter of the tree decreases by at least one, this construction can be repeated at most $D - 1$ times. To state it more accurately, if the sum-product in T converges in $d < D$ iterations, the sum-product in T' will converge in $d - 1 < D - 1$ steps. The proof is simply by observing that, the time index of the messages in T' are one ahead of the corresponding messages in T (which can also be written in the form of an induction).

Also, note that adding back the leaves to T' will add at most one more iteration to updates of BP (at the beginning; equivalent to modifying the leaf potentials), until convergence to fixed (if it ever converges). Therefore the BP converges at least in $d < D$ iterations.

□

Example 12 (Forward-Backward Algorithm for HMM)**.** *It be shown that the forward-backward algorithm is a special case of the sum-product algorithm.*

Remark 4. *The forwardbackward algorithm can be used to find the most likely state for any point in time. It cannot, however, be used to find the most likely sequence of states.*

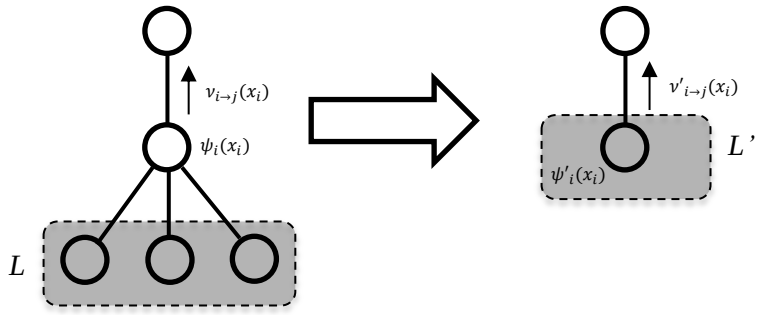


Figure 19: Stripping leaves of a given tree.

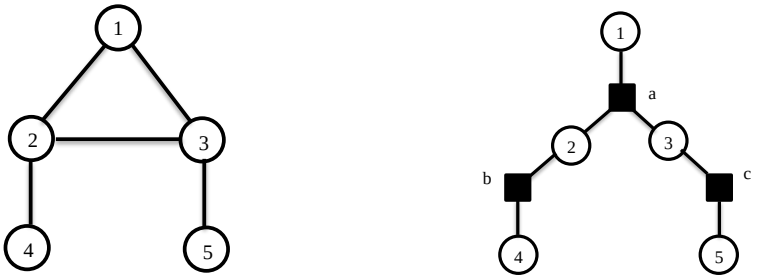


Figure 20: An undirected graphical model and its corresponding factor graph.

Example 13. *An undirected graphical model has been shown in Figure 20(left), along with its factor graph representation. Running the BP updates in the undirected graph is not guaranteed to result in a correct answer since the graph contains a loop. However in the factor graph representation the loop has been collapsed into one factor, and therefore there is no loop in the factor graph and BP updates are guaranteed to converge to the correct answer.*

Here we write the message updates in the factor graph.

Messages from factor a:

$$\begin{cases} \nu_{a\to 1}(x_1) = \sum_{x_2,x_3} \psi_{1,2,3}(x_1,x_2,x_3)\nu_{2\to a}(x_2)\nu_{3\to a}(x_3) \\ \nu_{a\to 2}(x_2) = \sum_{x_1,x_3} \psi_{1,2,3}(x_1,x_2,x_3)\nu_{1\to a}(x_1)\nu_{3\to a}(x_3) \\ \nu_{a\to 3}(x_3) = \sum_{x_1,x_2} \psi_{1,2,3}(x_1,x_2,x_3)\nu_{1\to a}(x_1)\nu_{2\to a}(x_2) \end{cases}$$

Messages from factor b:

$$\begin{cases} \nu_{b\to 2}(x_2) = \sum_{x_4} \psi_{2,4}(x_2,x_4)\nu_{4\to b}(x_4) \\ \nu_{b\to 4}(x_4) = \sum_{x_2} \psi_{2,4}(x_2,x_4)\nu_{2\to b}(x_2) \end{cases}$$

Messages from factor c:

$$\begin{cases} \nu_{c\to 3}(x_3) = \sum_{x_5} \psi_{3,5}(x_3,x_5)\nu_{5\to b}(x_5) \\ \nu_{c\to 5}(x_5) = \sum_{x_3} \psi_{3,5}(x_3,x_5)\nu_{3\to b}(x_3) \end{cases}$$

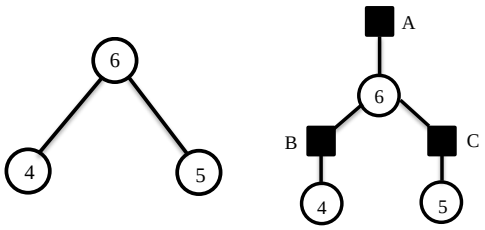


Figure 21: The undirected graph in Figure 20 after grouping the variables x_1, x_2, x_3, and its corresponding factor graph.

Messages from variable:

$$\begin{cases} \nu_{2\to b}(x_2) = \nu_{a\to 2}(x_2) \\ \nu_{2\to a}(x_2) = \nu_{b\to 2}(x_2) \\ \nu_{3\to c}(x_3) = \nu_{a\to 3}(x_3) \\ \nu_{3\to a}(x_3) = \nu_{c\to 3}(x_3) \\ \nu_{1\to a}(x_1) = \frac{1}{|x|} \\ \nu_{4\to b}(x_4) = \frac{1}{|x|} \\ \nu_{5\to c}(x_5) = \frac{1}{|x|} \end{cases}$$

By grouping the variables x_1, x_2, x_3 into one variable x_6 we get a tree graphical model in Figure 21 in which BP updates are guaranteed to converge to the correct fixed point. The new potential functions are:

$$\begin{cases} \tilde{\psi}_6(x_1, x_2, x_3) = \psi_{1,2,3}(x_1, x_2, x_3) \\ \tilde{\psi}_{65}(x_1, x_2, x_3, x_5) = \psi_{3,5}(x_3, x_5) \\ \tilde{\psi}_{64}(x_1, x_2, x_3, x_4) = \psi_{2,4}(x_2, x_4) \end{cases}$$

Messages from factor A:

$$\begin{cases} \nu_{As\to 6}(x_1, x_2, x_3) = \tilde{\psi}_6(x_1, x_2, x_3) \end{cases}$$

Messages from factor B:

$$\begin{cases} \nu_{B\to 6}(x_1, x_2, x_3) = \sum_{x_4} \tilde{\psi}_{64}(x_1, x_2, x_3, x_4)\nu_{4\to B}(x_4) \\ \nu_{B\to 4}(x_4) = \sum_{x_1,x_2,x_3} \tilde{\psi}_{64}(x_1, x_2, x_3, x_4)\nu_{6\to B}(x_1, x_2, x_3) \end{cases}$$

Messages from factor C:

$$\begin{cases} \nu_{C\to 6}(x_1, x_2, x_3) = \sum_{x_5} \tilde{\psi}_{65}(x_1, x_2, x_3, x_5)\nu_{5\to C}(x_5) \\ \nu_{C\to 5}(x_5) = \sum_{x_1,x_2,x_3} \tilde{\psi}_{65}(x_1, x_2, x_3, x_5)\nu_{6\to C}(x_1, x_2, x_3) \end{cases}$$

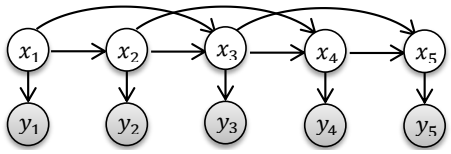


Figure 22: A second order HMM.

Message from variables:

$$\begin{cases} \nu_{5\to C}(x_5) = \frac{1}{|x|} \\ \nu_{4\to B}(x_4) = \frac{1}{|x|} \\ \nu_{6\to A}(x_1, x_2, x_3) = \nu_{6\to B}(x_1, x_2, x_3)\nu_{6\to C}(x_1, x_2, x_3) \\ \nu_{6\to B}(x_1, x_2, x_3) = \nu_{6\to A}(x_1, x_2, x_3)\nu_{6\to C}(x_1, x_2, x_3) \\ \nu_{6\to C}(x_1, x_2, x_3) = \nu_{6\to B}(x_1, x_2, x_3)\nu_{6\to A}(x_1, x_2, x_3) \end{cases}$$

We can continue the collapsing further and concatenate all the variables into one variable. In this case BP will not do anything. In case we are interested in any marginal, it could be calculated with looping through all possible configurations, which is exponential. This basically shows that, although one can gain accuracy by smart concatenation of variables, it increases the computational burden. Specifically, the case when everything is grouped into one variable, there is exponential computation needed to calculate the marginals.

Example 14. *Consider a random process that transitions on a finite set of states $s_1, \ldots, s_k$. At each times step $i \in [N]$[1], X_i represents the state of the system. Here is the full generative description:*

- *Sample the initial state X_1 from an initial distribution p_1.*
  - *Sample duration from a duration distribution p_D over integer set $[M]$.*
  - *Remain in the current state for the next d time steps.*
  - *Sample a successor state from a transition distribution $p_T(s'|s)$ over states $s' \neq s$.*

With that description, we can write the probability of observing state sequence $s_1, s_1, s_1, s_2, s_3, s_3$ is:

$$p_1(s_1)p_D(3)p_T(s_2|s_1)p_D(1)p_T(s_3|s_2)p_D(2)$$

The observations are the set of vectors y_i sampled from distribution $p(Y_i = y_i|X_i = s)$

We assume $M = 2$ and $p_D(d) = \begin{cases} 0.6 & d = 1 \\ 0.4 & d = 2 \end{cases}$ and each X_i takes on values from $\{a, b\}$.

1. *The dependence described in the model can be represented as Fig. 22.*
2. *We define the augmented state representation as (x, t) where x represents the state in the system and t the time elapsed with that state. For example for our previous example, the*

[1] $[N] = \{1, \ldots, N\}$

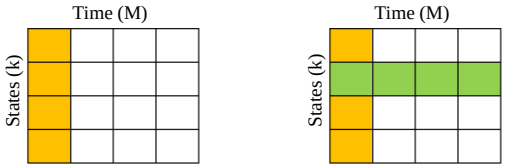


Figure 23: Smart state updates.

augmented representation is $(s_1,1),(s_1,2),(s_1,3),(s_2,1),(s_3,1),(s_3,2)$. *We define an HMM on the augmented state representation:*

$$\tilde{p}_{X_{i+1},T_{i+1}|X_i,T_i}(x_{i+1},t_{i+1}|x_i,t_i)=\begin{cases}\phi(x_i,x_{i+1},t_i) & t_{i+1}=1 \text{ and } x_{i+1}\neq x_i\\ \xi(x_i,t_i) & t_{i+1}=t_i+1 \text{ and } x_{i+1}=x_i\end{cases}$$

$$\xi(x_i,t)=\mathbb{P}\left(D\geq t_i+1|D\geq t_i\right)=\frac{\sum_{d\geq t_i+1}p_D(d)}{\sum_{d\geq t_i}p_D(d)}$$

$$\phi(x_i,x_{i+1},d)=\mathbb{P}\left(D=t_i|D\geq t_i\right)p_T(x_{i+1}|x_i)=\frac{p_D(t_i)}{\sum_{d\geq t_t}p_D(d)}p_T(x_{i+1}|x_i)$$

The observation distribution:

$$\tilde{p}_{Y_i|X_i,T_i}(y_i|x_i,t_i)=p(y_i|x_i)$$

3. *Suppose* $\mathcal{S}=\{s_1,\ldots,s_k\}$. *Here are the naive forward-backward updates:*

$$\begin{cases}\nu_{i\to(i+1)}(x_i,t_i)\propto\sum_{\substack{x_{i-1}\in\mathcal{S},\\ t_{i-1}\in[M]}}\tilde{p}_{X_i,T_i|X_{i-1},T_{i-1}}(x_i,t_i|x_{i-1},t_{i-1})\tilde{p}_{Y_{i-1}|X_{i-1},T_{i-1}}(y_{i-1}|x_{i-1},t_{i-1})\nu_{(i-1)\to i}(x_{i-1},t_{i-1})\\ \nu_{(i+1)\to i}(x_i,t_i)\propto\sum_{\substack{x_{i+1}\in\mathcal{S},\\ t_{i+1}\in[M]}}\tilde{p}_{X_{i+1},T_{i+1}|X_i,T_i}(x_{i+1},t_{i+1}|x_i,t_i)\tilde{p}_{Y_i|X_i,T_i}(y_i|x_i,t_i)\nu_{(i+2)\to(i+1)}(x_{i+1},t_{i+1})\end{cases}$$

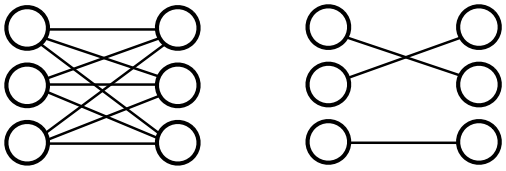
The complexity of such updates is $O(N(kM)^2)$. *However the structure of the problem could be exploited to make the computations faster. In other words, rather than repeating the above updates for any possible* (x_i,t_i) *we do the updates in smarter way. First we can fix* $t_i=1$ *and update all states (just like the traditional HMM). This is shown in the Figure 23 (left; yellow column). This has complexity* $O(Nk^2)$ *Then, for a fixed state* x_i, *we can fill in the states corresponding to different time steps* $(x_i,1\leq t\leq M)$ *(Figure 23 (right; green row)). Since in the original model, the delays are not dependent on the state, we can use the result for any state (thus copying the green state to any other rows). This will have an additional* $O(NkM)$ *complexity. In total complexity is* $O(N(k^2+kM))$.

6 Max-product algorithm

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

Example 15 (Inference in graphical models)**.** *Let's consider a general undirected graphical model with set of cliques* $\mathcal{C}$,

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



a complete bipartite graph a perfect matching

Figure 24: Example graphs for matching.

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 6. 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 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.*

Example 16. *We want to solve the Maximum Weighted Matching problem in a bipartitte graph. Given a bipartitte graph* $G(X, Y, E)$ *the variables* X *and* Y *are vertices on the left and right side, respectively, and their size* $|X| = |Y| = n$*. Any variable in* X *can be connected to only variables in* Y*. For any edge* $e_{ij} \in E$*, there is a weight* w_{ij} *associated. A perfect matching is the case when each variables in* X *is mapped to exactly one variable in* Y*, which can be captured with a permutation function on the nodes* $\pi = (\pi(1), \ldots, \pi(n))$ *and the total weight of the matching is* $W_\pi = \sum_i w_{i,pi(i)}$*. The problem of maximum weighted matching is defined as* $\arg\max_\pi W_\pi$*. We want to solve this by inducing a graphical model with weight proportional to the weight of matching.*

$$\mu(\pi) \propto e^{W_\pi} \mathbf{I}(\pi \text{ is a bipartite matching})$$

1. *One way to realize the distribution is the following form:*

$$\mu(x, y) = \prod_{(i,j)\in E} \psi_{i,j}(x_i, y_j) \prod_i e^{w_{i,x_i}} \prod_j e^{w_{y_j,j}}$$

where $x, y \in \{1, \dots, n\}^n$ encode the assignment of the nodes and

$$\psi_{i,j}(x_i, y_j) = \begin{cases} 0 & x_i = j \text{ and } y_j \neq i \\ 0 & x_i = j \text{ and } y_j \neq i \\ 1 & \text{otherwise} \end{cases}$$

To see the correctness, it is easy to see that having a perfect matching is equivalent to having $x_i = j$ and $y_j = i$ for any i, j, since in this case each node in the X side is connected to only one node in the Y side. Also it is easy to observe that, whens have a valid assignment (i.e. perfect matching) the value of the probability distribution is $\exp\left(\sum_i w_i, x_i + \sum_j w_j, y_j\right) = \exp 2W_\pi$.

2. We have shown that $\mu(x, y)$ is nonzero iff the assignment is a valid perfect matching. To find the maximum weighted matching it is enough to find $(x^*, y^*) = \arg\max_{x,y} \mu(x, y)$. This would essentially give the correct answer since $\mu(x, y)$ is proportional to W_π, when the assignment is correct. Also note that such assignment is essentially a perfect matching (by definition of $\mu(x, y)$).

3. We want to derive an algorithm based on Max-Product for finding the maximum of $\mu(x, y)$. Let's denote the variables on the left side by l_i and the variables on the right side by r_j. For example, a message from right to left is denoted with $v_{r_j \to l_i}(y_j)^{(t)}$ and the message from left to right is denoted by $v_{l_i \to r_j}(x_i)^{(t)}$. We initialize the messages as following:

$$v_{r_j \to l_i}(y_j)^{(0)} = e^{w_{i,x_i}}, v_{l_i \to r_j}(x_i)^{(0)} = e^{w_{y_j,j}}$$

The update rules for messages are the followings:

$$\begin{cases} v_{r_j \to l_i}(y_j)^{(t+1)} = \prod_{k \in [n] \setminus \{j\}} \max_{x_k} \psi_{k,j}(x_k, y_j) v_{l_k \to r_j}(x_k)^{(t+1)} \\ v_{l_i \to r_j}(x_i)^{(t+1)} = \prod_{k \in [n] \setminus \{i\}} \max_{y_k} \psi_{i,k}(x_i, y_k) v_{r_k \to l_i}(y_k)^{(t+1)} \end{cases}$$

To find the marginals we can do

$$\begin{cases} \mu_i(x_i) = \prod_{k \in [n]} \left\{ \max_{y_k} \psi_{i,k}(x_i, y_k) v_{r_k \to l_i}(y_k)^{(t_{max})} \right\} \\ \mu_i(y_j) = \prod_{k \in [n]} \left\{ \max_{x_k} \psi_{k,j}(x_k, y_j) v_{r_i \to l_k}(x_k)^{(t_{max})} \right\} \end{cases}$$

The correct configuration could be found by back-tracking the maximums (back-pointers).

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.

The loopy BP works better if the graph has shorter and smaller loops. It is known that if it has only one loop, the answers will be exact (reference?).

Example 17 (An example on loopy BP)**.** *See the loopy binary graphical model in Figure 25. 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[2]*,*

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

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:*

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

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

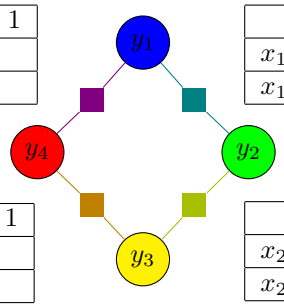
We don't continue further.

$(x_4, x_1) =$

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

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

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



$(x_3, x_4) =$

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

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

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

Figure 25: A loopy graphical model in factor-graph notation.

[2] 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.

8 Gaussian Graphical Models

First a brief on the properties of the Gaussian distribution:

Two representation of the Gaussian distribution

1. **Covariance form:** If $x \sim \mathcal{N}(m, \Lambda)$, then

$$\mu(x) = \frac{1}{(2\pi)^{n/2}\|\Lambda\|^{1/2}} \exp\left\{-\frac{1}{2}(x-m)^\top \Lambda^{-1}(x-m)\right\}$$

2. **Information form:** If $x \sim \mathcal{N}^{-1}(h, J)$, then

$$\mu(x) \propto \exp\left\{-\frac{1}{2}x^\top Jx + h^\top x\right\}$$

where h is the potential vector and J is the precision/information matrix.

A few important points:

- The equivalence between the two representation:

$$\begin{cases} J = \Lambda^{-1} \\ h = \Lambda^{-1}m = Jm \end{cases}$$

- Marginalizing: Given the following two joint distributions of (x_1, x_2) forms, we want to find the the distribution of x_1.

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} m_1 \\ m_2 \end{bmatrix}, \begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix}\right) = \mathcal{N}^{-1}\left(\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}, \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}\right)$$

  If the marginal form is $x_1 \sim \mathcal{N}(\hat{m}, \hat{\Lambda}) = \mathcal{N}(\hat{h}, \hat{J})$.

  - $\hat{m} = m_1$ and $\hat{\Lambda} = \Lambda_{11}$
  - $\hat{J} = \Lambda_{11}^{-1} = J_{11} - J_{12}J_{22}^{-1}J_{21}$ and $\hat{h} = \hat{J}m_1 = h_1 - J_{12}J_{22}^{-1}h_2$

- Conditioning: The goal is to find:

$$x_1|x_2 \sim \mathcal{N}(m', \Lambda') = \mathcal{N}^{-1}(h', J')$$

  - $h' = h_1 - J_{12}x_2$ and $J' = J_{11}$.
  - $m' = m_1 + \Lambda_{12}\Lambda_{22}^{-1}(x_2 - m_2)$ and $\Lambda' = \Lambda_{11} - \Lambda_{12}\Lambda_{12}^{-1}\Lambda_{21}$.

- Independent in the covariance form: In $x \sim \mathcal{N}(m, \Lambda)$, x_i and x_j are independent if and only if $\Lambda_{ij} = 0$.

- Pairwise independence in the information form: if $x \sim \mathcal{N}^{-1}(h, J)$, we have the pairwise independence $x_1 - x_{V\setminus\{i,j\}} - x_j$, if and only if $J_{ij} = 0$.

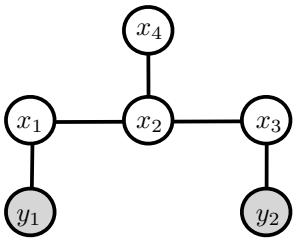


Figure 26: A sample graphical model.

If the messages are of the information form:

$$\nu_{k\to i}(x_k) = \mathcal{N}^{-1}(h_{k\to i}, J_{k\to i})$$

then the Gaussian belief updates are the followings:

$$\begin{cases} h_{i\to j} = h_i - \sum_{k\in\partial i\setminus j} J_{ik}J_{k\to i}^{-1}h_{k\to i} \\ J_{i\to j} = J_{ii} - \sum_{k\in\partial i\setminus j} J_{ik}J_{k\to i}^{-1}J_{ki} \end{cases}$$

And the marginals can be computed as $x_i \sim \mathcal{N}^{-1}(\hat{h}_i, \hat{J}_i)$ where

$$\begin{cases} \hat{h}_i = h_i - \sum_{k\in\partial i} J_{ik}J_{k\to i}^{-1}h_{k\to i} \\ \hat{J}_i = J_{ii} - \sum_{k\in\partial i} J_{ik}J_{k\to i}^{-1}J_{ki} \end{cases}$$

The complexity of these update are $O(n.d^3)$. Compare this to the complexity of of naively marginalization which is $O((n.d)^3)$.

Example 18. *If $x \sim \mathcal{N}^{-1}(h_x, J_x)$ and $y = Cx + v$ where $v \sim \mathcal{N}(0, R)$.*
a. We want to find the potential vector $h_{y|x}$ and the information matrix $J_{y|x}$ of $p(y|x)$.

$$\begin{aligned} p(x,y) = p(y|x)p(x) &\propto \exp\left\{-\frac{1}{2}\left(y-Cx\right)^\top R^{-1}\left(y-Cx\right)\right\}.\exp\left\{-\frac{1}{2}x^\top J_x x + h_x^\top x\right\} \\ &= \exp\left\{-\frac{1}{2}y^\top R^{-1}y + \frac{1}{2}x^\top C^\top R^{-1}y + \frac{1}{2}y^\top R^{-1}Cx - \frac{1}{2}x^\top\left(J_x + CR^{-1}C^\top\right)x + h_x^\top x\right\} \end{aligned}$$

Then

$$h_{x,y} = \left[h_x, 0\right]^\top, \quad J_{x,y} = \begin{bmatrix} R^{-1} & -C^\top R^{-1} \\ -R^{-1}C & J_x + CR^{-1}C^\top \end{bmatrix}$$

b. Given the joint distribution we can find the conditional $x|y$ we have:

$$\begin{cases} h_{x|y} = h_x + C^\top R^{-1}y \\ J_{x|y} = J_x + C^\top R^{-1}C \end{cases} \tag{10}$$

c. Suppose we have $R = I$ and

$$\begin{cases} y_1 = x_1 + v_1 \\ y_2 = x_3 + v_2 \end{cases}$$

The corresponding graphical model is shown in Figure 26. The C can be written as:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Based on the message updates in Equation 10, if we replace the definition of C we will have:

$$\begin{cases} h_{x_3|y} = h_{x_3} + y_2 \\ J_{x_3|y} = J_{x_3} + 1 \end{cases}$$

d. Similar to the previous section we can replace C in Equation 10.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Which gives the following message updates:

$$\begin{cases} h_{x_3|y} = h_{x_3} + y_2 + y_3 \\ J_{x_3|y} = J_{x_3} + 2 \end{cases}$$

e. Naturally, as there is more observation is received, the distributions (of messages) are expected to lean towards the empirical mean with decreasing variance (of the observation). We can verify this by checking the message parameters of $3 \to 2$ for one observation y_2 and two observations y_2 and y_3.

One observation:

$$\mathcal{N}(\frac{h_{x3} + y_2}{J_{x_3,x_3} + 1}, \frac{1}{J_{x_3,x_3} + 1})$$

Two observations:

$$\mathcal{N}(\frac{h_{x3} + y_2 + y_3}{J_{x_3,x_3} + 2}, \frac{1}{J_{x_3,x_3} + 2})$$

And in the case of n observations, we expect the mean to get closer to $\frac{1}{n}\sum_i y_i$.

9 Submodularity and tractable inference

We want to investigate the connection between min-cut problem and pair-wise graphical models. Consider a graphical model defined on $G(V, E)$:

$$\mu(x) \propto \exp\left\{ -\sum_{i \in V} \phi_i(x_i) - \sum_{(i,j) \in E} \phi_{ij}(x_i, x_j) \right\}$$

for $x_i \in \{0, 1\}^n$. Also we assume that $\phi_{ij}(0, 0) = \phi_{ij}(1, 1) = 0$ for all $(i, j) \in E$:

$$\phi_i(.) = \begin{bmatrix} \phi_i(0) \\ \phi_i(1) \end{bmatrix}, \quad \phi_{ij}(.) = \begin{bmatrix} 0 & \phi_{ij}(0,1) \\ \phi_{ij}(1,0) & 0 \end{bmatrix},$$

The goal is to find the assignment which maximizes the above distribution, which as we will show, can be cast into min-cut problem. To this end, given a pairwise MRF on $G(V, E)$ and compatibility function $\phi_{ij}(., .)$'s we create a directed and weighted graph $\tilde{G}(\tilde{V}, \tilde{E})$ as follow:

1. For any node in V add one node to $\tilde{V}$.
2. Add one node for source s and one node for sink t to $\tilde{V}$.
3. For any undirected edge $(i, j) \in E$, add two directed edges (i, j) and (j, i) to $\tilde{E}$.
4. Add directed edge from source s to any node (except the sink) to $\tilde{E}$.
5. Add directed edge from any node (except the source) to the sink t to $\tilde{E}$.

1. If any of the potentials is negativem we add a positive value to the edges to that they are not negative anymore, but the solution to the min-cut (and therefore the energy function) is not changed. For example, if $\phi_1(0) < 0$, we add big enough $\alpha > 0$ to the two edges out of the source $\phi_1(0)$ and the edge into the sink $\phi_1(1)$. This corresponds to adding a constant α to the energy function:

$$\begin{aligned} E'(x) &= (\alpha + \phi_0(x_i)) + \sum_{i \in V \setminus \{0\}} \phi_i(x_i) + \sum_{(i,j) \in E} \phi_{ij}(x_i, x_j) \\ &= \alpha + \sum_{i \in V} \phi_i(x_i) + \sum_{(i,j) \in E} \phi_{ij}(x_i, x_j) \end{aligned}$$

2. If the diagonal elements of ϕ (i.e. $\phi_{ij}(0, 0)$ and $\phi_{ij}(1, 1)$) are not zero, we can add them into the graph and still get correct result from the min-cut algorithm. For any $(i, j) \in E$, such that $\phi_{ij}(0, 0)$ or $\phi_{ij}(1, 1)$ are nonzero,
   - We add the extra weight $\phi_{ab}(0, 0)$ to the edge $(s, a) \in \tilde{E}$. This is to ensure that when the cut is contained both (s, a) and (s, b) (i.e. both assigned zero), it will also take $\phi_{ab}(0, 0)$ into account.
   - We add the extra weight $\phi_{ab}(1, 1)$ to the edge $(a, t) \in \tilde{E}$. Similar to the previous part, the reason is that, if the cut contains both (a, t) and (b, t) (both assigned 1), it takes the cost $\phi_{ab}(1, 1)$ into account.
   - What if the cut contains only a and not b? We subtract $\phi_{ab}(1, 1) + \phi_{ab}(0, 0)$ from the edge $(a, b) \in \tilde{E}$ to remove extra costs.
3. In order to solve the problem the max-flow algorithm we need to have a graph with positive weights. The only place that we might have negative weights are the pairwise edges, where the weights are

$$\begin{cases} \phi_{ab}(1, 0) - \phi_{ab}(0, 0) - \phi_{ab}(1, 1) \\ \phi_{ab}(0, 1) \end{cases} \tag{11}$$

   If any of the weights here are negative, we can use the previous trick of adding positive constants (part a), but with slight difference.

$$\begin{aligned} E'(x) &= \alpha + \sum_{i \in V} \phi_i(x_i) + \sum_{(i,j) \in E} \phi_{ij}(x_i, x_j) \\ &= (\alpha + \phi_{ab}(x_a, x_b)) + \sum_{i \in V} \phi_i(x_i) + \sum_{(i,j) \in E \setminus \{(a,b)\}} \phi_{ij}(x_i, x_j) \end{aligned}$$

Basically the extra constant is added to each of $\phi_{ab}(1,0)$, $\phi_{ab}(0,0)$, $\phi_{ab}(1,1)$ and $\phi_{ab}(0,1)$ in 11, which gives the following modified weights:

$$\begin{cases} \phi_{ab}(1,0) - \phi_{ab}(0,0) - \phi_{ab}(1,1) - \alpha \\ \phi_{ab}(0,1) + \alpha \end{cases}$$

Setting each of the weights to be bigger than zero we get:

$$-\phi_{ab}(0,1) \leq \alpha \leq \phi_{ab}(1,0) - \phi_{ab}(0,0) - \phi_{ab}(1,1)$$

$$\Rightarrow \phi_{ab}(1,0) + \phi_{ab}(0,1) \leq \phi_{ab}(0,0) + \phi_{ab}(1,1)$$

which is known as *submodularity* property.

10 On convergence of BP

In general there is no guarantee for BP to converge. In [9, 8] there is discussion on sufficient conditions for BP to converge.

11 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 [10, 6, 2, 1].
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.

12 Bibliographical notes

In preparation of this notes I have used Martin J. Wainwright's slides, and David Burkett's ACL tutorial . The nice paper [11] 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. Some of the images are from Sewoong Oh's course on Graphical Models.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.