Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.438 Algorithms for Inference
Fall 2014

15-16. Loopy Belief Propagation and Its Properties

Our treatment so far has been on *exact* algorithms for inference. We focused on trees first and uncovered efficient inference algorithms. To handle loopy graphs, we introduced the Junction Tree Algorithm, which could be computational intractable depending on the input graph. But many real-world problems involve loopy graphs, which demand efficient inference algorithms. Today, we switch gears to *approximate* algorithms for inference to tackle such problems. By tolerating some error in our solutions, many intractable problems suddenly fall within reach of efficient inference.

Our first stop will be to look at loopy graphs with at most pairwise potentials. Although we derived the Sum-Product algorithm so that it was exact on trees, nothing prevents us from running its message updates on a loopy graph with pairwise potentials. This procedure is referred to as *loopy belief propagation*. While mathematically unsound at a first glance, loopy BP performs surprisingly well on numerous real-world problems. On the flip side, loopy BP can also fail spectacularly, yielding nonsensical marginal distributions for certain problems. We want to figure out why such phenomena occur.

In guiding our discussion, we first recall the parallel Sum-Product algorithm, where we care about the parallel version rather than the serial version because our graph may not have any notion of leaves (e.g., a ring graph). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph over random variables $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_N \in \mathcal{X}$ distributed as follows:

$$p_{\mathbf{x}}(\mathbf{x}) \propto \prod_{i \in \mathcal{V}} \exp\left(\phi_i(x_i)\right) \prod_{(i,j) \in \mathcal{E}} \exp\left(\psi_{ij}(x_i, x_j)\right). \tag{1}$$

Remark: If $(i,j) \in \mathcal{E}$, then $(j,i) \notin \mathcal{E}$ since otherwise we would have ψ_{ij} and ψ_{ji} be two separate factors when we only want one of these; moreover, we define $\psi_{ji} \triangleq \psi_{ij}$.
Parallel Sum-Product is given as follows.
Initialization: For $(i,j) \in \mathcal{E}$ and $x_i, x_j \in \mathcal{X}$,

$$m^0_{i \to j}(x_j) \propto 1 \propto m^0_{j \to i}(x_i).$$

Message passing ($t = 0, 1, 2, \ldots$): For $i \in \mathcal{V}$ and $j \in N(i)$,

$$m^{t+1}_{i \to j}(x_j) \propto \sum_{x_i \in \mathcal{X}} \exp\left(\phi_i(x_i)\right) \exp\left(\psi_{ij}(x_i, x_j)\right) \prod_{k \in N(i) \setminus j} m^t_{k \to i}(x_i),$$

where we normalize our messages: $\sum_{x_j \in \mathcal{X}} m^t_{i \to j}(x_j) = 1$.

Computing node and edge marginals: For $(i,j) \in \mathcal{E}$,

$$b_i^t(x_i) \propto \exp\left(\phi_i(x_i)\right) \prod_{k \in N(i)} m_{k \to i}^t(x_i), \tag{2}$$

$$b_{ij}^t(x_i, x_j) \propto \exp\left(\phi_i(x_i) + \phi_j(x_j) + \psi_{ij}(x_i, x_j)\right) \prod_{k \in N(i) \setminus j} m_{k \to i}^t(x_i) \prod_{\ell \in N(j) \setminus i} m_{\ell \to j}^t(x_j). \tag{3}$$

But what exactly are these message-passing equations doing when the graph is not a tree? Let's denote $\mathbf{m}^t \in [0,1]^{2|\mathcal{E}||\mathcal{X}|}$ to be all messages computed in iteration t stacked up into a giant vector. Then we can view the message passing equations as applying some function $F : [0,1]^{2|\mathcal{E}||\mathcal{X}|} \to [0,1]^{2|\mathcal{E}||\mathcal{X}|}$ to $\mathbf{m}^t$ to obtain $\mathbf{m}^{t+1}$: $\mathbf{m}^{t+1} = F(\mathbf{m}^t)$. For loopy BP to converge, it must be that repeatedly applying the message-passing update F eventually takes us to what's called a *fixed point* $\mathbf{m}^*$ of F:

$$\mathbf{m}^* = F(\mathbf{m}^*).$$

This means that if we initialize our messages with $\mathbf{m}^*$, then loopy BP would immediately converge because repeatedly applying F to $\mathbf{m}^*$ will just keep outputting $\mathbf{m}^*$. Of course, in practice, we don't know where the fixed points are and the hope is that applying F repeatedly starting from some arbitrary point brings us to a fixed point, which would mean that loopy BP actually converges. This leads us naturally to the following questions:

- Does message-passing update F have a fixed point?
- If F has a fixed point or multiple fixed points, what are they?
- Does F actually converge to a fixed point?

In addressing these questions, we'll harness classical results from numerical analysis and interweave our inference story with statistical physics.

14.1 Brouwer's fixed-point theorem

To answer the first question raised, we note that the message-passing update equations are continous and so, collectively, the message-passing update function F is continuous and in fact maps a convex compact set[1] $[0,1]^{2|\mathcal{E}||\mathcal{X}|}$ to the same convex compact set. Then we can directly apply Brouwer's fixed-point theorem:

Theorem 1. *(Hadamard 1910, Brouwer 1912) Any continuous function mapping from a convex compact set to the same convex compact set has a fixed point.*

[1] In Euclidean space, a set is compact if and only if it is closed and bounded (Heine-Borel theorem) and a set is convex if the line segment connecting any two points in the set is also in the set.

This result has been popularized by a coffee cup analogy where we have a cup of coffee and we stir the coffee. The theorem states that after stirring, at least one point must be in the same position as it was prior to stirring! Note that the theorem is an existential statement with no known constructive proof that gives exact fixed points promised by the theorem. Moreover, as a historical note, this theorem can be used to prove the existence of Nash equilibria in game theory.

Applying Brouwer's fixed-point theorem to F, we conclude that in fact there does exist at least one fixed point to the message-passing equations. But what does this fixed point correspond to? It could be that this giant-vector-message fixed point does not correspond to the correct final messages that produce correct node and edge marginals. Our next goal is to characterize these fixed points by showing that they correspond to local extrema of an optimization problem.

14.2 Variational form of the log partition function

Describing the optimization problem that message-passing update equations actually solve involves a peculiar detour: We will look at a much harder optimization problem that in general we can't efficiently solve. Then we'll relax this hard optimization problem by imposing constraints to make the problem substantially easier, which will directly relate to our loopy BP message update equations.

The hard optimization problem is to solve for the log partition function of distribution (1). Recall from Lecture 1 that, in general, solving for the partition function is hard. Its logarithm is no easier to compute. But why should we care about the partition function though? The reason is that if we have a black box that computes partition functions, then we can get any marginal of the distribution that we want! We'll show this shortly but first we'll rewrite factorization (1) in the form of a Boltzmann distribution:

$$p_{\mathbf{x}}(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j) \right) = \frac{1}{Z} e^{-E(\mathbf{x})}, \tag{4}$$

where Z is the partition function and $E(\mathbf{x}) \triangleq -\sum_{i \in \mathcal{V}} \phi_i(x_i) - \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j)$ refers to energy. Then suppose that we wanted to know $p_{x_i}(x_i)$. Then letting $\mathbf{x}_{\setminus i}$ denote

the collection $(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N)$, we have

$$\begin{aligned} p_{\mathsf{x}_i}(x_i) &= \sum_{\mathbf{x}_{\setminus i}} p_{\mathbf{x}}(\mathbf{x}) \\ &= \sum_{\mathbf{x}_{\setminus i}} \frac{1}{Z} \exp\left(\sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j) \right) \\ &= \frac{\sum_{\mathbf{x}_{\setminus i}} \exp\left(\sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j) \right)}{Z} \\ &= \frac{Z(\mathsf{x}_i = x_i)}{Z} \end{aligned}$$

where $Z(\mathsf{x}_i = x_i) = \sum_{\mathbf{x}_{\setminus i}} \exp(\sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j))$ is just the partition function once we fix random variable x_i to take on value x_i. So if computing partition functions were easy, then we could compute any marginal—not just for nodes—easily. Thus, intuitively, computing the log partition function should be hard.

We now state what the hard optimization problem actually is:

$$\log Z = \sup_{\mu \in \mathcal{M}} \left\{ - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) E(\mathbf{x}) - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log \mu(\mathbf{x}) \right\}, \tag{5}$$

where $\mathcal{M}$ is the set of all distributions over $\mathcal{X}^N$. This expression is a *variational characterization* of the log partition function, which is a fancy way of saying that we wrote some expression as an optimization problem. We'll show why the right-hand side optimization problem actually does yield $\log Z$ momentarily. First, we lend some physical insight for what's going on by parsing the objective function $\mathcal{F}$ of the maximization problem in terms of energy:

$$\begin{aligned} \mathcal{F}(\mu) &= - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) E(\mathbf{x}) - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log \mu(\mathbf{x}) \\ &= - \underbrace{\mathbb{E}_\mu[E(\mathbf{x})]}_{\substack{\text{average energy} \\ \text{w.r.t. } \mu}} + \underbrace{\mathbb{E}_\mu[-\log \mu(\mathbf{x})]}_{\text{entropy of } \mu}. \end{aligned} \tag{6}$$

So by maximizing $\mathcal{F}$ over μ, we are finding μ that minimizes average energy while maximizing entropy, which makes sense from a physics perspective.

We could also view the optimization problem through the lens of information theory because as we'll justify shortly, any solution to optimization problem (5) is also a solution to the following optimization problem:

$$\min_{\mu \in \mathcal{M}} D(\mu \parallel p_{\mathbf{x}}), \tag{7}$$

where $D(\cdot \parallel \cdot)$ is the *Kullback-Leibler divergence* also called the *information divergence* between two distributions over the same alphabet:

$$D(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)}.$$

KL divergence is a way to measure how far apart two distributions are; however, it is not a proper distance because it is not symmetric. It offers a key property that we shall use though: $D(p \parallel q) \geq 0$ for all distributions p and q defined on the same alphabet, where equality occurs if and only if p and q are the same distribution.

We now show why the right-hand side maximization in equation (5) yields $\log Z$ and, along the way, establish that the maximization in (5) has the same solution as the minimization in (7). We begin by rearranging terms in (4) to obtain

$$E(\mathbf{x}) = -\log Z - \log p_{\mathbf{x}}(\mathbf{x}). \tag{8}$$

Plugging (8) into (6), we obtain

$$\begin{aligned}
\mathcal{F}(\mu) &= -\sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) E(\mathbf{x}) - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log \mu(\mathbf{x}) \\
&= \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \left(\log Z + \log p_{\mathbf{x}}(\mathbf{x})\right) - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log \mu(\mathbf{x}) \\
&= \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log Z + \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log p_{\mathbf{x}}(\mathbf{x}) - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log \mu(\mathbf{x}) \\
&= \log Z + \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log p_{\mathbf{x}}(\mathbf{x}) - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log \mu(\mathbf{x}) \\
&= \log Z - \sum_{\mathbf{x} \in \mathcal{X}^N} \mu(\mathbf{x}) \log \frac{\mu(\mathbf{x})}{p_{\mathbf{x}}(\mathbf{x})} \\
&= \log Z - D(\mu \parallel p_{\mathbf{x}}) \\
&\leq \log Z,
\end{aligned}$$

where the last step is because $D(\mu \parallel p_{\mathbf{x}}) \geq 0$, with equality if and only if $\mu \equiv p_{\mathbf{x}}$. In fact, we could set μ to be $p_{\mathbf{x}}$, so the upper bound established is attained with $\mu \equiv p_{\mathbf{x}}$, justifying why the maximization in (5) yields $\log Z$. Moreover, from the second-to-last line in the derivation above, since $\log Z$ does not depend on μ, maximizing $\mathcal{F}(\mu)$ over μ is equivalent to minimizing $D(\mu \parallel p_{\mathbf{x}})$, as done in optimization problem (7).

So we now even know the value of μ that maximizes the right-hand side of (5). Unfortunately, plugging in μ to be $p_{\mathbf{x}}$, we're left with having to sum over all possible configurations $\mathbf{x} \in \mathcal{X}^N$, which in general is intractable. We next look at a way to relax this problem called *Bethe approximation*.

14.3 Bethe approximation

The guiding intuition we use for relaxing the log partition optimization is to look at what happens when $p_{\mathbf{x}}$ has a tree graph. By what we argued earlier, the solution to the log partition optimization is to set $\mu \equiv p_{\mathbf{x}}$, so if $p_{\mathbf{x}}$ has a tree graph, then μ must factorize as a tree as well. Hence, we need not optimize over all possible distributions in $\mathcal{X}^N$ for μ; we only need to look at distributions with tree graphs!

With this motivation, we look at how to parameterize μ to simplify our optimization problem, which involves imposing constraints on μ. We'll motivate the contraints we're about to place by way of example. Supposing that $p_{\mathbf{x}}$ has the tree graph in Figure 1, then $p_{\mathbf{x}}$ factorizes as:

$$\begin{aligned} p_{\mathbf{x}}(\mathbf{x}) &= p_{x_1}(x_1)p_{x_2|x_1}(x_2|x_1)p_{x_3|x_1}(x_3|x_1) \\ &= p_{x_1}(x_1)\frac{p_{x_1,x_2}(x_1,x_2)}{p_{x_1}(x_1)}\frac{p_{x_1,x_3}(x_1,x_3)}{p_{x_1}(x_1)} \\ &= p_{x_1}(x_1)p_{x_2}(x_2)p_{x_3}(x_3)\frac{p_{x_1,x_2}(x_1,x_2)}{p_{x_1}(x_1)p_{x_2}(x_2)}\frac{p_{x_1,x_3}(x_1,x_3)}{p_{x_1}(x_1)p_{x_3}(x_3)}. \end{aligned}$$

A pattern emerges in the last step, which in fact holds in general: if $p_{\mathbf{x}}$ has tree graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, then its factorization is given by

$$p_{\mathbf{x}}(\mathbf{x}) = \prod_{i \in \mathcal{V}} p_{x_i}(x_i) \prod_{(i,j) \in \mathcal{E}} \frac{p_{x_i,x_j}(x_i,x_j)}{p_{x_i}(x_i)p_{x_j}(x_j)}.$$

Using this observation, we make the crucial step for relaxing the log partition optimization by parameterizing μ by:

$$\mu(\mathbf{x}) = \prod_{i \in \mathcal{V}} \mu_i(x_i) \prod_{(i,j) \in \mathcal{E}} \frac{\mu_{ij}(x_i,x_j)}{\mu_i(x_i)\mu_j(x_j)}, \tag{9}$$

where we have introduced new distributions μ_i and μ_{ij} that need to behave like node marginals and pairwise marginals. By forcing μ to have the above factorization, we arrive at the following *Bethe variational problem* for the log partition function:

$$\log Z_{\text{bethe}} = \max_{\mu} \mathcal{F}(\mu) \tag{10}$$

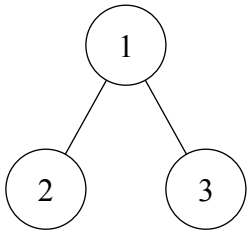

Figure 1

subject to the constraints:

$$\begin{aligned}
\mu(\mathbf{x}) &= \prod_{i\in\mathcal{V}} \mu_i(x_i) \prod_{(i,j)\in\mathcal{E}} \frac{\mu_{ij}(x_i,x_j)}{\mu_i(x_i)\mu_j(x_j)} && \text{for all } \mathbf{x}\in\mathcal{X}^N \\
\mu_i(x_i) &\geq 0 && \text{for all } i\in\mathcal{V}, x_i\in\mathcal{X} \\
\sum_{x_i\in\mathcal{X}} \mu_i(x_i) &= 1 && \text{for all } i\in\mathcal{V} \\
\mu_{ij}(x_i,x_j) &\geq 0 && \text{for all } (i,j)\in\mathcal{E}, x_i, x_j\in\mathcal{X} \\
\sum_{x_j\in\mathcal{X}} \mu_{ij}(x_i,x_j) &= \mu_i(x_i) && \text{for all } (i,j)\in\mathcal{E}, x_i\in\mathcal{X} \\
\sum_{x_i\in\mathcal{X}} \mu_{ij}(x_i,x_j) &= \mu_j(x_j) && \text{for all } (i,j)\in\mathcal{E}, x_j\in\mathcal{X}
\end{aligned}$$

The key takeaway here is that *if $p_{\mathbf{x}}$ has a tree graph*, then the optimal μ will factor as a tree and so this new Bethe variational problem is equivalent to our original log partition optimization problem and, furthermore, $\log Z = \log Z_{\text{bethe}}$.

However, if $p_{\mathbf{x}}$ does not have a tree graph but still has the original pairwise potential factorization given in equation (4), then the above optimization may no longer be exact for computing $\log Z$, i.e., we may have $\log Z_{\text{bethe}} \neq \log Z$. Constraining μ to factor into μ_i's and μ_{ij}'s as in equation (9), where $\mathcal{E}$ is the set of edges in $p_{\mathbf{x}}$ and could be loopy, is referred to as a *Bethe approximation*, named after physicist Hans Bethe. Note that since edge set $\mathcal{E}$ is fixed and comes from $p_{\mathbf{x}}$, we are not optimizing over the space of all tree distributions!

We're now ready to state a landmark result.

Theorem 2. *(Yedidia, Freeman, Weiss 2001) The fixed points of Sum-Product message updates result in node and edge marginals that are local extrema of the Bethe variational problem* (10)*.*

We set the stage for the proof by first massaging the objective function a bit to make it clear exactly what we're maximizing once we plug in the equality constraint for μ factorizing as μ_i's and μ_{ij}'s. This entails a fair bit of algebra. We first compute simplified expressions for $\log p_{\mathbf{x}}$ and $\log \mu$, which we'll then use to derive reasonably simple expressions for the average energy and entropy terms of objective function $\mathcal{F}$.

Without further ado, denoting d_i to be the degree of node i, we take the log of equation (4):

$$\begin{aligned}
\log p_{\mathbf{x}} &= -\log Z + \sum_{i\in\mathcal{V}} \phi_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i,x_j) \\
&= -\log Z + \sum_{i\in\mathcal{V}} \phi_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \left(\psi_{ij}(x_i,x_j) + \phi_i(x_i) + \phi_j(x_j)\right) - \sum_{i\in\mathcal{V}} d_i\phi_i(x_i) \\
&= -\log Z + \sum_{i\in\mathcal{V}} (1-d_i)\phi_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \left(\psi_{ij}(x_i,x_j) + \phi_i(x_i) + \phi_j(x_j)\right).
\end{aligned}$$

Meanwhile, taking the log of the tree factorization imposed for μ given by equation (9), we get:

$$\begin{aligned}\log \mu(\mathbf{x}) &= \sum_{i \in \mathcal{V}} \log \mu_i(x_i) + \sum_{(i,j) \in \mathcal{E}} (\log \mu_{ij}(x_i, x_j) - \log \mu_i(x_i) - \log \mu_j(x_j)) \\ &= \sum_{i \in \mathcal{V}} (1 - d_i) \log \mu_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \log \mu_{ij}(x_i, x_j).\end{aligned}$$

Now, recall from equations (6) and (8) that

$$\mathcal{F}(\mu) = \underbrace{-\mathbb{E}_\mu[E(\mathbf{x})]}_{\substack{\text{average energy} \\ \text{w.r.t. } \mu}} + \underbrace{\mathbb{E}_\mu[-\log \mu(\mathbf{x})]}_{\text{entropy of } \mu} = \mathbb{E}_\mu[\log Z + \log p_{\mathbf{x}}(\mathbf{x})] + \mathbb{E}_\mu[-\log \mu(\mathbf{x})].$$

Gladly, we've computed $\log p_{\mathbf{x}}$ and $\log \mu$ already, so we plug these right in to determine what $\mathcal{F}$ is equal to *specifically when μ factorizes like a tree*:

$$\begin{aligned}\mathcal{F}(\mu) &= \mathbb{E}_\mu[\log Z + \log p_{\mathbf{x}}(\mathbf{x})] + \mathbb{E}_\mu[-\log \mu(\mathbf{x})] \\ &= \mathbb{E}_\mu \left[\sum_{i \in \mathcal{V}} (1 - d_i) \phi_i(\mathsf{x}_i) + \sum_{(i,j) \in \mathcal{E}} (\psi_{ij}(\mathsf{x}_i, \mathsf{x}_j) + \phi_i(\mathsf{x}_i) + \phi_j(\mathsf{x}_j)) \right] \\ &\quad - \mathbb{E}_\mu \left[\sum_{i \in \mathcal{V}} (1 - d_i) \log \mu_i(\mathsf{x}_i) + \sum_{(i,j) \in \mathcal{E}} \log \mu_{ij}(\mathsf{x}_i, \mathsf{x}_j) \right] \\ &= \sum_{i \in \mathcal{V}} (1 - d_i) \mathbb{E}_\mu[\phi_i(\mathsf{x}_i)] + \sum_{(i,j) \in \mathcal{E}} \mathbb{E}_\mu[\psi_{ij}(\mathsf{x}_i, \mathsf{x}_j) + \phi_i(\mathsf{x}_i) + \phi_j(\mathsf{x}_j)] \\ &\quad + \sum_{i \in \mathcal{V}} (1 - d_i) \mathbb{E}_\mu[-\log \mu_i(\mathsf{x}_i)] + \sum_{(i,j) \in \mathcal{E}} \mathbb{E}_\mu[-\log \mu_{ij}(\mathsf{x}_i, \mathsf{x}_j)] \\ &= \sum_{i \in \mathcal{V}} (1 - d_i) \mathbb{E}_{\mu_i}[\phi_i(\mathsf{x}_i)] + \sum_{(i,j) \in \mathcal{E}} \mathbb{E}_{\mu_{ij}}[\psi_{ij}(\mathsf{x}_i, \mathsf{x}_j) + \phi_i(\mathsf{x}_i) + \phi_j(\mathsf{x}_j)] \\ &\quad + \sum_{i \in \mathcal{V}} (1 - d_i) \underbrace{\mathbb{E}_{\mu_i}[-\log \mu_i(\mathsf{x}_i)]}_{\text{entropy of } \mu_i} + \sum_{(i,j) \in \mathcal{E}} \underbrace{\mathbb{E}_{\mu_{ij}}[-\log \mu_{ij}(\mathsf{x}_i, \mathsf{x}_j)]}_{\text{entropy of } \mu_{ij}} \\ &\triangleq \mathcal{F}_{\text{bethe}}(\mu),\end{aligned}$$

where $\mathcal{F}_{\text{bethe}}$ is the *negative Bethe free energy*. Consequently, the Bethe variational problem (10) can be viewed as minimizing the Bethe free energy. Mopping up $\mathcal{F}_{\text{bethe}}$

a little more yields:

$$\begin{aligned}
\mathcal{F}_{\text{bethe}}(\mu) &= \sum_{i \in \mathcal{V}} (1 - d_i) \mathbb{E}_{\mu_i}[\phi_i(\mathsf{x}_i) - \log \mu_i(\mathsf{x}_i)] \\
&\quad + \sum_{(i,j) \in \mathcal{E}} \mathbb{E}_{\mu_{ij}}[\psi_{ij}(\mathsf{x}_i, \mathsf{x}_j) + \phi_i(\mathsf{x}_i) + \phi_j(\mathsf{x}_j) - \log \mu_{ij}(\mathsf{x}_i, \mathsf{x}_j)] \\
&= \sum_{i \in \mathcal{V}} (1 - d_i) \sum_{x_i \in \mathcal{X}} \mu_i(x_i)[\phi_i(x_i) - \log \mu_i(x_i)] \\
&\quad + \sum_{(i,j) \in \mathcal{E}} \sum_{x_i, x_j \in \mathcal{X}} \mu_{ij}(x_i, x_j)[\psi_{ij}(x_i, x_j) + \phi_i(x_i) + \phi_j(x_j) - \log \mu_{ij}(x_i, x_j)].
\end{aligned} \tag{11}$$

We can now rewrite the Bethe variational problem (10) as

$$\max_{\mu} \mathcal{F}_{\text{bethe}}(\mu)$$

subject to the constraints:

$$\begin{aligned}
\mu_i(x_i) &\geq 0 && \text{for all } i \in \mathcal{V}, x_i \in \mathcal{X} \\
\sum_{x_i \in \mathcal{X}} \mu_i(x_i) &= 1 && \text{for all } i \in \mathcal{V} \\
\mu_{ij}(x_i, x_j) &\geq 0 && \text{for all } (i,j) \in \mathcal{E}, x_i, x_j \in \mathcal{X} \\
\sum_{x_j \in \mathcal{X}} \mu_{ij}(x_i, x_j) &= \mu_i(x_i) && \text{for all } (i,j) \in \mathcal{E}, x_i \in \mathcal{X} \\
\sum_{x_i \in \mathcal{X}} \mu_{ij}(x_i, x_j) &= \mu_j(x_j) && \text{for all } (i,j) \in \mathcal{E}, x_j \in \mathcal{X}
\end{aligned}$$

Basically all we did was simplify the objective function by plugging in the factorization for μ. As a result, the stage is now set for us to prove Theorem 2.

Proof. Our plan of attack is to introduce Lagrange multipliers to solve the now simplified Bethe variational problem. We'll look at where the gradient of the Lagrangian is zero, corresponding to local extrema, and we'll see that at these local extrema, Lagrange multipliers relate to Sum-Product messages that have reached a fixed-point and the μ_i's and μ_{ij}'s correspond exactly to Sum-Product's node and edge marginals.

So first, we introduce Lagrange multipliers. We won't introduce any for the non-negativity constraints because it'll turn out that these constraints aren't active; i.e., the other constraints will already yield local extrema points that always have non-negative μ_i and μ_{ij}. Let's assign Lagrange multipliers to the rest of the constraints:

Constraints	**Lagrange multipliers**
$\sum_{x_i \in \mathcal{X}} \mu_i(x_i) = 1$	λ_i
$\sum_{x_j \in \mathcal{X}} \mu_{ij}(x_i, x_j) = \mu_i(x_i)$	$\lambda_{j \to i}(x_i)$
$\sum_{x_i \in \mathcal{X}} \mu_{ij}(x_i, x_j) = \mu_j(x_j)$	$\lambda_{i \to j}(x_j)$

Collectively calling all the Lagrange multipliers λ, the Lagrangian is thus

$$\begin{aligned}\mathcal{L}(\mu,\lambda) = \mathcal{F}_{\text{bethe}}(\mu) &+ \sum_{i\in\mathcal{V}} \lambda_i \left(\sum_{x_i\in\mathcal{X}} \mu_i(x_i) - 1 \right) \\ &+ \sum_{(i,j)\in\mathcal{E}} \sum_{x_i\in\mathcal{X}} \lambda_{j\to i}(x_i) \left(\sum_{x_j\in\mathcal{X}} \mu_{ij}(x_i,x_j) - \mu_i(x_i) \right) \\ &+ \sum_{(i,j)\in\mathcal{E}} \sum_{x_j\in\mathcal{X}} \lambda_{i\to j}(x_j) \left(\sum_{x_i\in\mathcal{X}} \mu_{ij}(x_i,x_j) - \mu_j(x_j) \right).\end{aligned}$$

The local extrema of $\mathcal{F}_{\text{bethe}}$ subject to the constraints we imposed on μ are precisely the points where the gradient of $\mathcal{L}$ with respect to μ is zero and all the equality constraints in the table above are met.

Next, we take derivatives. We'll work with the form of $\mathcal{F}_{\text{bethe}}$ given in equation (11), and we'll use the result that $\frac{d}{dx} x\log x = \log x + 1$. First, we differentiate with respect to $\mu_k(x_k)$:

$$\begin{aligned}\frac{\partial\mathcal{L}(\mu,\lambda)}{\partial\mu_k(x_k)} &= (1-d_k)(\phi_k(x_k) - \log\mu_k(x_k) - 1) + \lambda_k - \sum_{\ell\in N(k)} \lambda_{\ell\to k}(x_k) \\ &= -(d_k-1)\phi_k(x_k) + (d_k-1)(\log\mu_k(x_k)+1) + \lambda_k - \sum_{\ell\in N(k)} \lambda_{\ell\to k}(x_k).\end{aligned}$$

Setting this to 0 gives

$$\log\mu_k(x_k) = \phi_k(x_k) + \frac{1}{d_k-1}\left(\sum_{\ell\in N(k)} \lambda_{\ell\to k}(x_k) - \lambda_k \right) - 1,$$

so

$$\mu_k(x_k) \propto \exp\left\{ \phi_k(x_k) + \frac{1}{d_k-1} \sum_{\ell\in N(k)} \lambda_{\ell\to k}(x_k) \right\}. \tag{12}$$

Next, we differentiate with respect to $\mu_{k\ell}(x_k,x_\ell)$:

$$\frac{\partial\mathcal{L}(\mu,\lambda)}{\partial\mu_{k\ell}(x_k,x_\ell)} = \psi_{k\ell}(x_k,x_\ell) + \phi_k(x_k) + \phi_\ell(x_\ell) - \log\mu_{k\ell}(x_k,x_\ell) - 1 + \lambda_{\ell\to k}(x_k) + \lambda_{k\to\ell}(x_\ell).$$

Setting this to 0 gives:

$$\mu_{k\ell}(x_k,x_\ell) \propto \exp\left\{ \psi_{k\ell}(x_k,x_\ell) + \phi_k(x_k) + \phi_\ell(x_\ell) + \lambda_{\ell\to k}(x_k) + \lambda_{k\to\ell}(x_\ell) \right\}. \tag{13}$$

We now introduce variables $m_{i\to j}(x_j)$ for $i\in\mathcal{V}$ and $j\in N(i)$, which are cunningly named as they will turn out to be the same as Sum-Product messages, but right now,

we do not make such an assumption! The idea is that we'll write our $\lambda_{i\to j}$ multipliers in terms of $m_{i\to j}$'s. But how do we do this? Pattern matching between equation (13) and the edge marginal equation (3) suggests that we should set

$$\lambda_{i\to j}(x_j) = \sum_{k\in N(j)\setminus i} \log m_{k\to j}(x_j) \qquad \text{for all } i \in \mathcal{V}, j \in N(i), x_j \in \mathcal{X}.$$

With this substitution, equation (13) becomes

$$\mu_{k\ell}(x_k, x_\ell) \propto \exp\left(\psi_{k\ell}(x_k, x_\ell) + \phi_k(x_k) + \phi_\ell(x_\ell)\right) \prod_{i\in N(k)\setminus \ell} m_{i\to k}(x_k) \prod_{j\in N(\ell)\setminus k} m_{j\to \ell}(x_\ell), \tag{14}$$

which means that at a local extremum of $\mathcal{F}_{\text{bethe}}$, the above equation is satisfied. Of course, this is the same as the edge marginal equation for Sum-Product. We verify a similar result for μ_i. The key observation is that

$$\sum_{\ell\in N(k)} \lambda_{\ell\to k}(x_k) = \sum_{\ell\in N(k)} \sum_{i\in N(k)\setminus \ell} \log m_{i\to k}(x_k) = (d_k - 1) \sum_{j\in N(k)} \log m_{j\to k}(x_k),$$

which follows from a counting argument. Plugging this result directly into equation (12), we obtain

$$\begin{aligned}
\mu_k(x_k) &\propto \exp\left\{\phi_k(x_k) + \frac{1}{d_k - 1} \sum_{\ell\in N(k)} \lambda_{\ell\to k}(x_k)\right\} \\
&= \exp\left\{\phi_k(x_k) + \frac{1}{d_k - 1}(d_k - 1) \sum_{j\in N(k)} \log m_{j\to k}(x_k)\right\} \\
&= \exp(\phi_k(x_k)) \prod_{j\in N(k)} m_{j\to k}(x_k),
\end{aligned} \tag{15}$$

matching the Sum-Product node marginal equation.

What we've shown so far is that any local extremum μ of $\mathcal{F}_{\text{bethe}}$ subject to the constraints we imposed must have μ_i and μ_{ij} take on the forms given in Equations (15) and (14), which just so happen to match node and edge marginal equations for Sum-Product. But we have not yet shown that the messages themselves are at a fixed point. To show this last step, we examine our equality constraints on μ_{ij}. It suffices to just look at what happens for one of them:

$$\begin{aligned}
&\sum_{x_\ell\in\mathcal{X}} \mu_{k\ell}(x_k, x_\ell) \\
&\propto \sum_{x_\ell\in\mathcal{X}} \exp\left(\psi_{k\ell}(x_k, x_\ell) + \phi_k(x_k) + \phi_\ell(x_\ell)\right) \prod_{i\in N(k)\setminus \ell} m_{i\to k}(x_k) \prod_{j\in N(\ell)\setminus k} m_{j\to \ell}(x_\ell) \\
&= \left(\exp(\phi_k(x_k)) \prod_{i\in N(k)\setminus \ell} m_{i\to k}(x_k)\right) \sum_{x_\ell\in\mathcal{X}} \exp\left(\psi_{k\ell}(x_k, x_\ell) + \phi_\ell(x_\ell)\right) \prod_{j\in N(\ell)\setminus k} m_{j\to \ell}(x_\ell),
\end{aligned}$$

at which point, comparing the above equation and the μ_k equation (15), perforce, we have

$$m_{\ell \to k}(x_k) = \sum_{x_\ell} \exp\left(\psi_{k\ell}(x_k, x_\ell) + \phi_\ell(x_\ell)\right) \prod_{j \in N(\ell) \setminus k} m_{j \to \ell}(x_\ell).$$

The above equation is just the Sum-Product message-passing update equation! Moreover, the above is in fact satisfied at a local extremum of $\mathcal{F}_{\text{bethe}}$ for all messages $m_{i \to j}$. Note that there is absolutely no dependence on iterations. This equation say that once we're at a local extremum of $\mathcal{F}_{\text{bethe}}$, the above equation must hold for all $x_k \in \mathcal{X}$. This same argument holds for all the other $m_{i \to j}$'s, which means that we're at a fixed-point. And such a fixed-point is precisely for the Sum-Product message-update equations. □

We've established that Sum-Product message updates on loopy graphs do have at least one fixed point and, moreover, all possible fixed points are local extrema of $\mathcal{F}_{\text{bethe}}$, the negative Bethe free energy. Unfortunately, Theorem 2 is only a statement about local extrema and says nothing about whether a fixed point of Sum-Product message updates is a local maximum or a local minimum for $\mathcal{F}_{\text{bethe}}$. Empirically, it has been found that fixed points corresponding to maxima of the negative Bethe free energy, i.e., minima of Bethe free energy, tend to be more stable. This empirical result intuitively makes sense since for loopy graphs, we can view local maxima of $\mathcal{F}_{\text{bethe}}$ as trying to approximate $\log Z$, which was what the original hard optimization problem was after in the first place.

Now that we have characterized the fixed points of Sum-Product message-passing equations for loopy (as well as not-loopy) graphs, it remains to discuss whether the algorithm actually converges to any of these fixed points.

14.4 Loopy BP convergence

Our quest ends in the last question we had asked originally, which effectively asks whether loopy BP converges to any of the fixed points. To answer this, we will sketch two different methods of analysis and will end by stating a few results on Gaussian loopy BP and an alternate approach to solving the Bethe variational problem.

14.4.1 Computation trees

In contrast to focusing on fixed points, computation trees provide more of a dynamic view of loopy BP through visualizing how messages across iterations contribute to the computation of a node marginal. We illustrate computation trees through an example. Consider a probability distribution with the loopy graph in Figure 2a.

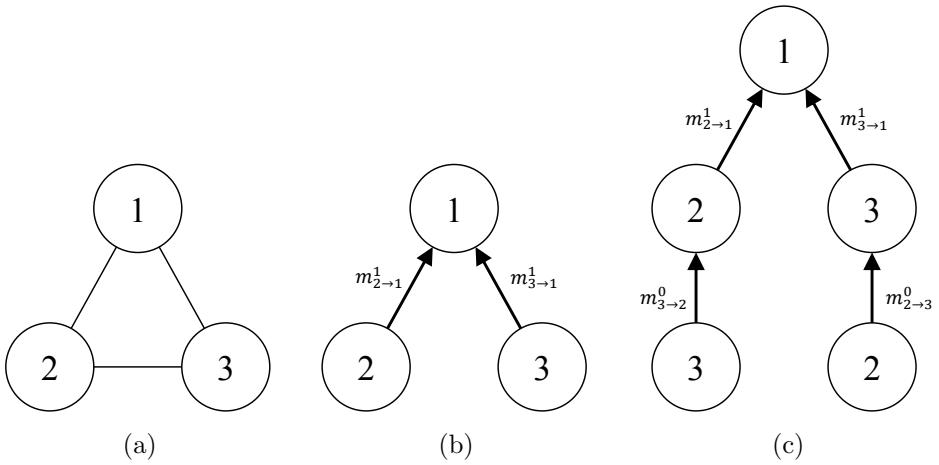

Figure 2

Suppose we ran loopy BP for exactly one iteration and computed the marginal at node 1. Then this would require us to look at messages $m^1_{2\to1}$ and $m^1_{3\to1}$, where the superscripts denote the iteration number. We visualize these two dependencies in Figure 2b. To compute $m^1_{2\to1}$, we needed to look at message $m^0_{3\to2}$, and to compute $m^1_{3\to1}$, we needed to look at $m^0_{2\to3}$. These dependencies are visualized in Figure 2c.

These dependency diagrams are called computation trees. For the example graph in Figure 2a, we can also draw out the full computation tree up to iteration t, which is shown in Figure 3. Note that loopy BP is operating as if the underlying graph is the computation tree (with the arrows on the edges removed), not realizing that some nodes are duplicates!

But how can we use the computation tree to reason about loopy BP convergence? Observe that in the full computation tree up to iteration t, the left and right chains that descend from the top node each have a repeating pattern of three nodes, which are circled in Figure 3. Thus, the initial uniform message sent up the left chain will pass through the repeated block over and over again. In fact, assuming messages are renormalized at each step, then each time the message passes through the repeated block of three nodes, we can actually show that it's as if the message was left-multiplied by a state transition matrix $\mathbf{M}$ for a Markov chain!

So if the initial message $\mathbf{v}$ on the left chain is from node 2, then after $3t$ iterations, the resulting message would be $\mathbf{M}^t\mathbf{v}$. This analysis suggests that loopy BP converges to a unique solution if as $t \to \infty$, $\mathbf{M}^t\mathbf{v}$ converges to a unique vector and something similar happens for the right-hand side chain. In fact, from Markov chain theory, we know that this uniqueness result does occur if our alphabet size is finite and $\mathbf{M}$ is the transition matrix for an ergodic Markov chain. This result follows from the Perron-Frobenius theorem and can be used to show loopy BP convergence for a variety of graphs with single loops. Complications arise when there are multiple loops.

14.4.2 Attractive fixed points

Another avenue for analyzing convergence is a general fixed-point result in numerical analysis that basically answers: if we're close to a fixed point of some iterated function, does applying the function push us closer toward the fixed point? One way to answer this involves looking at the gradient of the iterated function, which for our case is the Sum-Product message update F defined back in Section for giant vectors containing all messages.

We'll look at the 1D case first to build intuition. Suppose $f : \mathbb{R} \to \mathbb{R}$ has fixed point $x^* = f(x^*)$. Let x^t be our guess of fixed point x^* at iteration t, where x^0 is our initial guess and we have $x^{t+1} = f(x^t)$. What we would like is that at each iteration, the error decreases, i.e., error $|x^{t+1} - x^*|$ should be smaller than error $|x^t - x^*|$. More formally, we want

$$|x^{t+1} - x^*| \leq \rho |x^t - x^*| \qquad \text{for some } \rho < 1. \tag{16}$$

In fact, we could chain these together to bound our total error at iteration $t+1$ in terms of our initial error:

$$\begin{aligned} |x^{t+1} - x^*| &\leq \rho |x^t - x^*| \leq \rho(\rho |x^{t-1} - x^*|) \\ &\leq \rho(\rho(\rho |x^{t-2} - x^*|)) \cdots \\ &\leq \rho^{t+1} |x^0 - x^*|, \end{aligned} \tag{17}$$

which implies that as $t \to \infty$, the error is driven to 0.

We're now going to relate this back to f, but first, we'll need to recall the Mean-Value Theorem: Let $a < b$. If $f : [a, b] \to \mathbb{R}$ is continuous on $[a, b]$ and differentiable on (a, b), then $f(a) - f(b) = f'(c)(a - b)$ for some $c \in (a, b)$. Then

$$\begin{aligned} |x^{t+1} - x^*| &= |f(x^t) - f(x^*)| \\ &= |f'(y)(x^t - x^*)| \quad \text{(via Mean-Value Theorem)} \\ &= |f'(y)||x^t - x^*|, \end{aligned}$$

for some y in an open interval (a, b) with $a < b$ and $x^t, x^* \in [a, b]$. Note that if $|f'(y)| \leq \rho$ for all y close to fixed point x^*, then we'll exactly recover condition (16), which will drive our error over time to 0, as shown in inequality (17). Phrased another way, if $|f'(y)| < 1$ for all y close to fixed point x^*, then if our initial guess is close enough to x^*, we will actually converge to x^*.

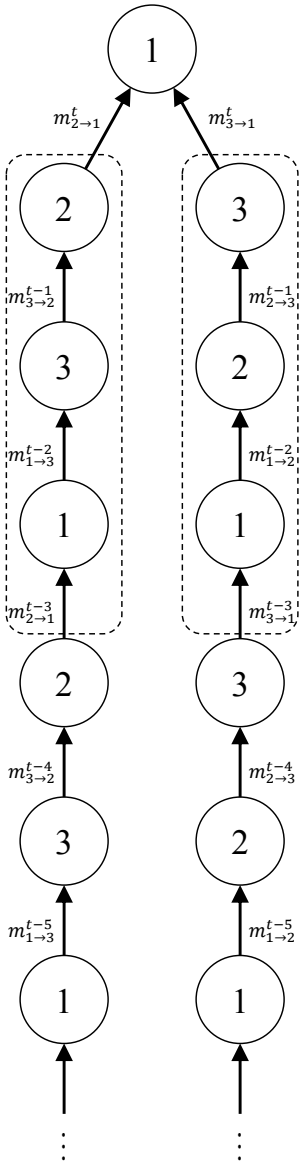

Figure 3

Extending this to the n-dimensional case is fairly straight-forward. We'll basically apply a multi-dimensional version of the Mean-Value Theorem. Let $f : \mathbb{R}^n \to \mathbb{R}^n$ have fixed point $\mathbf{x}^* = f(\mathbf{x}^*)$. We update $\mathbf{x}^{t+1} = f(\mathbf{x}^t)$. Let f_i denote the i-th component of the output of f, i.e.,

$$\mathbf{x}^{t+1} = f(\mathbf{x}^t) = \begin{pmatrix} f_1(\mathbf{x}^t) \\ f_2(\mathbf{x}^t) \\ \vdots \\ f_n(\mathbf{x}^t) \end{pmatrix}.$$

Then

$$\begin{aligned} |x_i^{t+1} - x_i^*| &= |f_i(\mathbf{x}^t) - f_i(\mathbf{x}^*)| \\ &= |\nabla f_i(\mathbf{y}_i)^{\mathrm{T}}(\mathbf{x}^t - \mathbf{x}^*)| \qquad \text{(multi-dimensional Mean-Value Theorem)} \\ &\le \|f_i'(\mathbf{y}_i)\|_1 \|\mathbf{x}^t - \mathbf{x}^*\|_\infty, \end{aligned}$$

for some $\mathbf{y}_i$ in an neighborhood whose closure includes $\mathbf{x}^t$ and $\mathbf{x}^*$, and where the inequality[2] is because for any $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$,

$$|\mathbf{a}^{\mathrm{T}}\mathbf{b}| \le \left| \sum_{k=1}^n a_k b_k \right| \le \sum_{k=1}^n |a_k||b_k| \le \underbrace{\left(\sum_{k=1}^n |a_k| \right)}_{\triangleq \|\mathbf{a}\|_1} \underbrace{\left(\max_{\ell=1,\ldots,n} |b_\ell| \right)}_{\triangleq \|\mathbf{b}\|_\infty}.$$

Lastly, taking the max of both sides across all $i = 1, 2, \ldots, n$, we get

$$\|\mathbf{x}^{t+1} - \mathbf{x}^*\|_\infty \le \left(\max_{i=1,\ldots,n} \|\nabla f_i(\mathbf{y}_i)\|_1 \right) \|\mathbf{x}^t - \mathbf{x}^*\|_\infty.$$

Then if $\|\nabla f_i(\mathbf{z})\|_1$ is strictly less than positive constant S for all $\mathbf{z}$ in an open ball whose closure includes $\mathbf{x}^t$ and $\mathbf{x}^*$ and for all i, then

$$\|\mathbf{x}^{t+1} - \mathbf{x}^*\|_\infty \le \left(\max_{i=1,\ldots,n} \|\nabla f_i(\mathbf{y}_i)\|_1 \right) \|\mathbf{x}^t - \mathbf{x}^*\|_\infty \le S\|\mathbf{x}^t - \mathbf{x}^*\|_\infty,$$

and provided that $S < 1$ and $\mathbf{x}^0$ is sufficiently close to $\mathbf{x}^*$, iteratively applying f will indeed cause $\mathbf{x}^t$ to converge to $\mathbf{x}^*$.

In fact, if $\|\nabla f_i(\mathbf{x}^*)\|$ is strictly less than 1 for all i, then by continuity and differentiability of f, there will be some neighborhood $\mathcal{B}$ around $\mathbf{x}^*$ for which $\|\nabla f_i(\mathbf{z})\| < 1$ for all $\mathbf{z} \in \mathcal{B}$ and for all i. In this case, $\mathbf{x}^*$ is called an *attractive fixed point*. Note that this condition is sufficient but not necessary for convergence.

This gradient condition can be used on the Sum-Product message update F and its analysis is dependent on the node and edge potentials.

[2] In fact, this is a special case of what's called *Hölder's inequality*, which generalizes Cauchy-Schwarz by upper-bounding absolute values of dot products by dual norms.

14.4.3 Final remarks

We've discussed existence of at least one loopy BP fixed point and we've shown that any such fixed point is a local extremum encountered in the Bethe variational problem. Whether loopy BP converges is tricky; we sketched two different approaches one could take. Luckily, some work has already been done for us: Weiss and Freeman (2001) showed that if Gaussian loopy BP converges, then we are guaranteed that the means are correct but not necessarily the covariances; additional conditions ensure that the covariances are correct as well. And if loopy BP fails to converge, then a recent result by Shih (2011) gives an algorithm with running time $O(n^2(1/\varepsilon)^n)$ that is guaranteed to solve the Bethe variational problem with error at most ε.

6.438 Algorithms for Inference

Fall 2014