Example 1 Ising Model
The variables here are \( x_i \in \{-1; 1\} \) for \( i \in [n] \). Again, we call \( \mathbf{x} = (x_1, ..., x_n) \) the vector of all variables.

We are given a graph \( G \).

Define the energy function
\[
H(x) = - \sum_{i,j \in G} J_{ij} x_i x_j - \sum_i h_i x_i. \tag{1}
\]

This leads to the following distribution
\[
\mu(x) = \frac{1}{Z} e^{-H(x)}
\]

If we neglect the term corresponding to the \( h_i \)'s, we can write this as a factor graph:

- 1 variable node per original node in the graph \( G \).
- 1 function node per edge in the graph \( G \).

This defines a factor graph \( G(V,F,E') \). Thus we can write \( \mu(x) \) in the usual way
\[
\mu(x) = \frac{1}{Z} \prod_{a \in F} \psi_a(x_a)
\]

In this specific case, it is easy to find what the functions \( \psi_a \) are. It is important to know that computing \( Z \) is, in general, \#P-complete, though there are instances where it is easy to calculate (say for planar graphs).

Example 2 Source Coding using LDPC
Here, \( x_i \in \{0, 1\} \) and \( i \in [n] \). Again, we call \( \mathbf{x} = (x_1, ..., x_n) \) the vector of all variables. Assume that all \( x_i \)'s are IID copies of a given Bernoulli random variable with parameter \( p \in [0, 1] \).

The question we want to answer is how to compress \( \mathbf{x} = (x_1, ..., x_n) \). Let \( H \) be a \( \{0, 1\} \times n \) matrix with \( m < n \). We perform a mod (2) vector matrix multiplication to get
\[
y = Hx.
\]

Let \( m = rn \). Given that \( m < n \), we have that \( r \in (0, 1) \), and we call \( r \) the rate. Let us now describe the underlying graphical model.

- 1 variable node per entry in \( \mathbf{x} \).
- 1 function node per entry in \( \mathbf{y} \).
- An edge \((i,a) \in G\) if and only if \( H_{ia} = 1 \).

In other words, \( H \) is the adjacency matrix. Now, given \( \mathbf{y} \), how do we recover \( \mathbf{x} \)? If we set
\[
\hat{x}_i = \underset{x_i}{\text{argmax}} P(X_i = x_i | Y = y)
\]
then we minimize the expected number of errors. We need to calculate the posterior probability and then take the max. We first calculate \( \mu(x) = P(X = \mathbf{x} | Y = \mathbf{y}) \) and then we get the marginals from here. This is quite complicated. Using Bayes’ theorem, we get
\[
\mu(x) = \frac{P(y|x) P(x)}{P(y)} \propto \prod_{a \in \{Y_a = (Hx)_a\}} \prod_i Q(x_i)
\]
where \( Q(x_i) = 1 - p \) when \( x_i = 0 \) and \( Q(x_i) = p \) if \( x_i = 1 \).

What is crucial is that out of \( H \) and its underlying graph, we are computing marginals of a graphical model.

We now turn our attention to more abstract material. We start with message passing algorithms. These are not exact but they are very efficient.

1 Message Passing

Message passing have the following characteristics:

- Iterative. Time is indexed by \( t \in \mathbb{N} \)
- Messages are basic quantities. They are associated with directed edges. We define two different set of variables related to messages:
  - \( \{ \nu^{(t)}_{i \rightarrow a} : i \in V, a \in F \} \)
  - \( \{ \hat{\nu}^{(t)}_{a \rightarrow i} : i \in V, a \in F \} \)

where \( \nu \) and \( \hat{\nu} \) are in some alphabet \( M \).
- Initialization. We will not linger on the initialization, just assume that there is an element of \( M \) called “0”, and that \( \forall i, a, \nu^{(0)}_{i \rightarrow a} = \hat{\nu}^{(0)}_{a \rightarrow i} = 0. \)
- Update. We are given two sets of functions to update both types of messages:

\[
\nu_{i \rightarrow a}^{(t+1)} = \Phi_{i \rightarrow a}(\{ \hat{\nu}^{(t)}_{b \rightarrow i} : b \in \partial_i \setminus a \})
\]
\[
\hat{\nu}_{a \rightarrow i}^{(t+1)} = \Phi_{a \rightarrow i}(\{ \nu^{(t)}_{j \rightarrow a} : j \in \partial_a \setminus i \})
\]

Now we have that \( \nu_{i \rightarrow a}^{(t)}(x_i) = \Phi_{i \rightarrow a}(\{ \hat{\nu}^{(t)}_{a \rightarrow i} : a \in \partial_i \setminus i \}) \).

NB: We exclude information from the node we are sending a message to to avoid self reinforcing loops. We can still run into cases where these self reinforcing loops exist. That represents the weak point of these algorithms. We’ll see through an example that such exclusions make these algorithms exact on trees.

**Example 3 Belief Propagation (SUM-PRODUCT algorithm in coding theory)**

Messages are a distribution over \( \mathcal{X} \). \( \nu_{i \rightarrow a}^{(t)}(x_i) \) is a guess on the distribution of \( x_i \) given \( G(i \rightarrow a) \).

![Figure 1: Graph G(i→a)](image)

And \( M \) is defined as follows: \( M = \{ (\nu(1), ..., \nu(\mathcal{X})) | \nu(i) \geq 0 \text{ and } \sum_i \nu(i) = 1 \} \). We can see that

\[
\nu_{i \rightarrow a}^{(t+1)}(x_i) \propto \prod_{b \in \partial_i \setminus a} \hat{\nu}_{b \rightarrow i}^{(t)}(x_i)
\]
\[
\hat{\nu}_{a \rightarrow i}^{(t+1)}(x_i) \propto \sum_{\{x_j : j \in \partial_a \setminus i \}} \psi_a(x_a) \prod_{j \in \partial_a \setminus i} \nu_{j \rightarrow a}^{(t)}(x_j)
\]
where the “proportional to” is needed because of a normalization factor.

In the end, we get that

\[ \bar{\nu}^{(t)}(x_i) \propto \prod_{a \in \partial_i} \rho^{(t)}_{a \rightarrow i}(x_i). \]

We are now ready to prove that belief propagation is exact on trees. In order to state rigorously and prove such result, we need some notation and definitions. Assume that \( G \) is a tree, and let \( u \) and \( v \) be two of its nodes. We define:

- \( G(u \rightarrow v) \) is the subtree rooted at \( v \) and containing \( u \) as \( v \)'s only child and the subtree rooted at \( u \). See figure 2 for an illustration.
- \( \tau(u \rightarrow v) \) as the depth of \( G(u \rightarrow v) \).
- \( \mu^{u \rightarrow v}(.) \) as the graphical model on \( G(u \rightarrow v) \) induced by the original graphical model.

**Figure 2:** Subtree \( G(u \rightarrow v) \) in red

**Proposition 1** (Belief Propagation is Exact on Trees). Let \( i \in V \) and \( a \in F \) be such that \((i, a) \in E\). Then we have that:

1. \( \forall t \geq \tau(i \rightarrow a) + 2, \)
   \[ \nu^{(t)}_{i \rightarrow a}(x_i) = \mu^{i \rightarrow a}(x_i) \]

2. \( \forall t \geq \max\{\tau(a \rightarrow i), a \in \partial_i\} + 1, \)
   \[ \bar{\nu}^{(t)}(x_i) = \mu(x_i) \]

**Proof** By induction on the depth of the tree.

Assume that the tree is the one represented in Figure 3. Then we have that

**Figure 3:** Initialization of Induction
\[ \mu_{i \rightarrow a}(x_i) = \sum_{\{x_j, j \neq i\}} \mu_{j \rightarrow a}(x_j) \]

\[ \propto \sum_{\{x_j, j \neq i\}} \prod_{b \in \mathcal{G}(i \rightarrow a)} \psi_b(x_{\partial_b}) \]

\[ = \sum_{\{x_j, j \neq i\}} \prod_{b \in \partial_i \setminus \partial_a} \prod_{c \in \mathcal{G}(b \rightarrow i)} \psi_c(x_{\partial_c}) \]

\[ = \prod_{b \in \partial_i \setminus \partial_a} \left( \sum_{\{x_j, j \in \mathcal{G}(b \rightarrow i)\} \cap \mathcal{G}(b \rightarrow i)} \mu_{b \rightarrow i}(x) \right) \]

\[ \propto \prod_{b \in \partial_i \setminus \partial_a} \mu_{b \rightarrow i}(x_i). \]

We can perform the same analysis for the other base case, namely that represented in Figure 4 and get

![Figure 4: Initialization of Induction](image)

\[ \nu_{a \rightarrow i}(x_i) \propto \sum_{\{x_j, j \in \partial_a \setminus \partial_i\}} \psi_a(x_{\partial_a}) \prod \mu_{j \rightarrow a}(x_j). \]

Now assume it true for depth up to \( t \). We have that

\[ \nu_{i \rightarrow a}^{(t+1)}(x_i) \propto \prod_{b} \nu_{b \rightarrow i}^{(t)}(x_i), \]

by the induction hypothesis, we have that

\[ \propto \prod_{b} \sum_{j \in \partial_b \setminus \partial_i} \psi_b(x_{\partial_b}) \mu_{j \rightarrow b}(x_j) \]

\[ \propto \prod_{b} \sum_{j \in \partial_b \setminus \partial_i} \psi_b(x_{\partial_b}) \mu_{j \rightarrow b}(x_j). \]

This concludes the first section of this course. We now turn our attention to variational methods. We want to understand how computing marginals is related to relaxations of optimization problems.