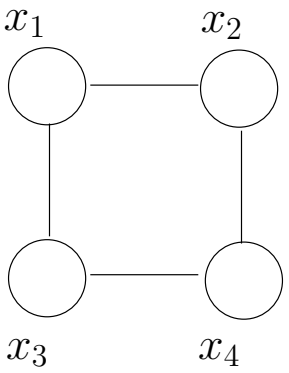
Lecture topics:

- Markov Random Fields
- Probabilistic inference

Markov Random Fields

We will briefly go over *undirected graphical models* or *Markov Random Fields* (MRFs) as they will be needed in the context of probabilistic inference discussed below (using the model to calculate various probabilities over the variables). The origin of these models is physics (e.g., spin glass) and they retain some of the terminology from the physics literature. The semantics of MRFs is similar but simpler than Bayesian networks. The graph again represents independence properties between the variables but the properties can be read off from the graph through simple graph separation rather than D-separation criterion. So, for example,



encodes two independence properties. First, x_1 is independent of x_4 given x_2 and x_3. In other words, if we remove x_2 and x_3 from the graph then x_1 and x_4 are no longer connected. The second property is that x_2 is independent of x_3 given x_1 and x_4. Incidentally, we couldn't define a Bayesian network over the same four variables that would explicate both of these properties (you can capture one while failing the other). So, in terms of their ability to explicate independence properties, MRFs and Bayesian networks are not strict subsets of each other.

By *Hammersley-Clifford theorem* we can specify the form that any joint distribution consistent with an undirected graph has to take. A distribution is consistent with the graph

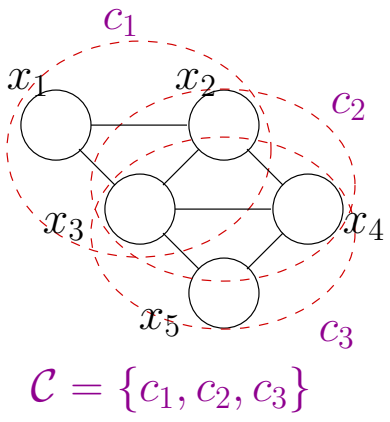
if it satisfies all the conditional independence properties we can read from the graph. The result is again in terms of how the distribution has to factor. For the above example, we can write the distribution as a product of (non-negative) *potential functions* over pairs of variables that specify how the variables depend on each other

$$P(x_1, x_2, x_3, x_4) = \frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)\psi_{24}(x_2, x_4)\psi_{34}(x_3, x_4) \tag{1}$$

where Z is a normalization constant (required since the potential functions can be any non-negative functions). The distribution is therefore *globally normalized*. More generally, an undirected graph places no constraints on how any fully connected subset of the variables, variables in a *clique*, depend on each other. In other words, we are free to associate any potential function with such variables. Without loss of generality we can restrict ourselves to *maximal cliques*, i.e., not consider separately cliques that are subsets of other cliques. In the above example, the maximal cliques where the pairs of connected variables. Now, in general, the Hammersley-Clifford theorem states that the joint distribution has to factor according to (maximal) cliques in the graph:

$$P(x_1, \ldots, x_n) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi_c(x_c) \tag{2}$$

where $c \in \mathcal{C}$ is a (maximal) clique in the graph and $x_c = \{x_i\}_{i\in c}$ denotes the set of variables in the clique. The normalization constant Z could be easily absorbed into one of the potential functions but we will write it explicitly here as a reminder that the model has to be normalized globally (is not automatically normalized as Bayesian networks). Figure below provides an example of a graph with three maximal cliques.



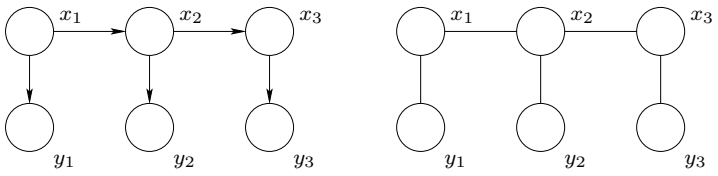
Bayesian networks as undirected models

We can always turn a Bayesian network into a MRF via *moralization*, i.e., connecting all the parents of a common child and dropping the directions on the edges. After the transformation we naturally still have the same probability distribution but may no longer capture all the independence properties explicitly in the graph. For example, in

$P(x_1, x_2, x_3) = P(x_1)P(x_2)P(x_3|x_1, x_2)$ $P(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$

where, clearly, $\psi(x_1, x_2, x_3) = P(x_1)P(x_2)P(x_3|x_1, x_2)$ so that the underlying distributions are the same (only the representation changed). The undirected graph is fully connected, however, and the marginal independence of x_1 and x_2 is no longer visible in the graph. In terms of probabilistic inference, i.e., calculating various probabilities, little is typically lost by turning a Bayesian network first into an undirected model. For example, we would often have some evidence pertaining to the variables, something would be known about x_3 and, as a result, x_1 and x_2 would become dependent. The advantage from the transformation is that the inference algorithms will run uniformly on both types of models.

Let's consider one more example of turning Bayesian networks into MRFs. The figure below gives a simple HMM with the associated probability model and the same for the undirected version after moralization.



$P(x_1)P(x_2|x_1)P(x_3|x_2)\times$
$\times P(y_1|x_1)P(y_2|x_2)P(y_3|x_3)$

$\frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{23}(x_2, x_3)\times$
$\times\phi_1(x_1, y_1)\phi_2(x_2, y_2)\phi_3(x_3, y_3)$

Each conditional probability on the left can be assigned to any potential function that contains the same set of variables. For example, $P(x_1)$ could be included in $\psi_{12}(x_1, x_2)$ or in $\phi_1(x_1, y_1)$. The objective is merely to maintain the same distribution when we take the

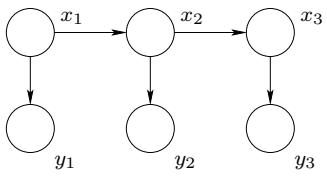
product (it doesn't matter how we reorder terms in a product). Here's a possible complete setting of the potential functions:

$$
\begin{aligned}
Z &= 1 \text{ (already normalized as we start with a Bayesian network)} && (3)\\
\psi_{12}(x_1, x_2) &= P(x_1)P(x_2|x_1) && (4)\\
\psi_{23}(x_2, x_3) &= P(x_3|x_2) && (5)\\
\phi_1(x_1, y_1) &= P(y_1|x_1) && (6)\\
\phi_1(x_2, y_2) &= P(y_2|x_2) && (7)\\
\phi_1(x_3, y_3) &= P(y_3|x_3) && (8)
\end{aligned}
$$

Probabilistic inference

Once we have the graph and the associated distribution (either learned from data or given to us), we would like to make use of this distribution. For example, in HMMs discussed above, we could try to compute $P(y_3|y_1, y_2)$, i.e., predict what we expect to see as the next observation having already seen y_1 and y_2. Note that the sequence of observations in an HMM does not satisfy the Markov property, i.e., y_3 is not independent of y_1 given y_1. This is easy to see from either Bayesian network or the undirected version via the separation criteria. You can also understand it by noting that y_1 may influence the state sequence, and therefore which value x_3 takes provided that y_2 does not fully constraint x_2 to take a specific value. We are also often interested in *diagnostic* probabilities such as $P(x_2|y_1, y_2, y_3)$, the posterior distribution over the states x_2 at $t = 2$ when we have already observed y_1, y_2, y_3. Or we may be interested in the most likely hidden state sequence and need to evaluate max-probabilities. All of these are probabilistic inference calculations.

If we can compute basic conditional probabilities such as $P(x_2|y_2)$, we can also evaluate various other quantities. For example, suppose we have an HMM



and we are interesting known which y's we should observe (query) so as to obtain the most information about x_2. To this end we have to define a value of new information. Consider,

for example, defining

$$\text{Value}(\{P(x_2|y_i)\}_{x_2=1,\ldots,m}) = -\text{Entropy}(x_2|y_i) \quad (9)$$

$$= \sum_{x_2} P(x_2|y_i) \log P(x_2|y_i) \quad (10)$$

In other words, if given a specific observation y_i, we can evaluate the value of the resulting conditional distribution $P(x_2|y_i)$ where y_i is known. There are many ways to define the value and, for simplicity, we defined it in terms of the uncertainty about x_2. The value is zero if we know x_2 perfectly and negative otherwise. Since we cannot know y_i prior to querying its value, we will have to evaluate its expected value assuming our HMM is correct: *the expected value of information* in response to querying a value for y_i is given by

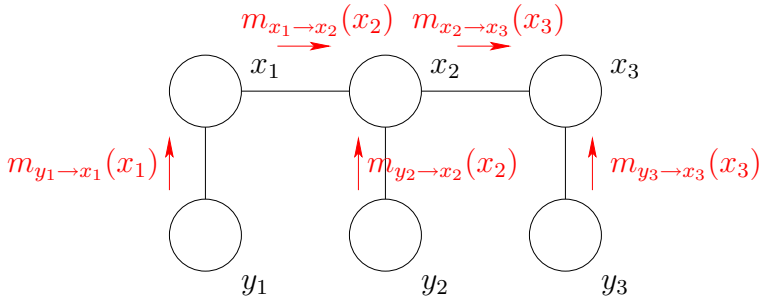
$$\sum_{y_i} P(y_i) \left[\sum_{x_2} P(x_2|y_i) \log P(x_2|y_i) \right] \quad (11)$$

where $P(y_i)$ is a marginal probability computed from the same HMM. We could now use the above criterion to find the observation most helpful in determining the value of x_2. Note that all the probabilities we needed were simple marginal and conditional probabilities. We still need to discuss how to evaluate such probabilities efficiently from a given distribution.

Belief propagation

Belief propagation is a simple message passing algorithm that generalizes the forward-backward algorithm. It is exact on undirected graphs that are trees (a graph is a tree if any pair of nodes have a unique path connecting them, i.e., has no loops). In case of more general graphs, we can cluster variables together so as to obtain a tree of clusters, and apply the same algorithm again, now on the level of clusters.

We will begin by demonstrating how messages are computed in the belief propagation algorithm. Consider the problem of evaluating the marginal probability of x_3 in an undirected HMM



$$\frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{23}(x_2, x_3)\times$$
$$\times\phi_1(x_1, y_1)\phi_2(x_2, y_2)\phi_3(x_3, y_3)$$

We can perform the required marginalizations (summing over the other variables) in order: first y_1, then x_1, then y_2, and so on. Each of such operations will affect the variables they interact with. This effect is captured in terms of messages that are shown in red in the above figure. For example, $m_{y_1 \to x_1}(x_1)$ is a message, a function of x_1, that summarizes the effect of marginalizing over y_1. It is computed as

$$m_{y_1 \to x_1}(x_1) = \sum_{y_1} \phi_1(\mathbf{x}_1, y_1) \tag{12}$$

Note that in the absence of any evidence about y_1, $\phi(x_1, y_1) = P(y_1|x_1)$ and we would simply get a constant function of x_1 as the message. Suppose, instead, that we had already observed the value of y_1 and denote this value as $\hat{y}_1$. We can incorporate this observation (evidence about y_1) into the potential function $\phi_1(x_1, y_1)$ by redefining it as

$$\phi_1(x_1, y_1) = \delta(y_1, \hat{y}_1)P(y_1|x_1) \tag{13}$$

This way the message would be calculated as before but the value of the message as a function of x_1 would certainly change:

$$m_{y_1 \to x_1}(x_1) = \sum_{y_1} \phi_1(\mathbf{x}_1, y_1) = \sum_{y_1} \delta(y_1, \hat{y}_1)P(y_1|x_1) = P(\hat{y}_1|x_1) \tag{14}$$

After completing the marginalization over y_1, we turn to x_1. This marginalization results in a message $m_{x_1 \to x_2}(x_2)$ as x_2 relates to x_1. In calculating this message, we will have to take into account the message from y_1 so that

$$m_{x_1 \to x_2}(x_2) = \sum_{x_1} m_{y_1 \to x_1}(x_1)\psi_{12}(x_1, x_2) \tag{15}$$

More generally, in evaluating such messages, we incorporate (take a product of) all the incoming messages except the one coming from the variable we are marginalizing towards. Thus, x_2 will send the following message to x_3

$$m_{x_2 \to x_3}(x_3) = \sum_{x_2} m_{x_1 \to x_2}(x_2) m_{y_2 \to x_2}(x_2) \psi_{23}(x_2, x_3) \tag{16}$$

Finally, the probability of x_3 is obtained by collecting all the messages into x_3

$$P(x_3, D) = m_{x_2 \to x_3}(x_3) m_{y_3 \to x_3}(x_3) \tag{17}$$

where D refers to any data or observations incorporated into the potential functions (as illustrated above). The distribution we were after is then

$$P(x_3|D) = \frac{P(x_3, D)}{\sum_{x'_3} P(x'_3, D)} \tag{18}$$

It might seem that to evaluate similar distributions for all the other variables we would have to perform the message passing operations (marginalizations) in a particular order in each case. This is not necessary, actually. We can simply initialize all the messages to all one functions, and carry out the message passing operations asynchronously. For example, we can pick x_2 and calculate its message to x_1 based on the other available incoming messages (that may be wrong at this time). The messages will converge to the correct ones provided that the undirected model is a tree, and we repeatedly update each message (i.e., send messages from each variable in all directions). The asynchronous message updates will propagate the necessary information across the graph. This exchange of information between the variables is a bit more efficient to carry out synchronously, however. In other words, we designate a root variable and imagine the edges oriented outwards from this variable (this orientation has nothing to do with Bayesian networks). Then "collect" all the messages toward the root, starting from the leaves. Once the root has all its incoming messages, the direction is switched and we send (or "distribute") the messages outwards from the root, starting with the root. These two passes suffice to get the correct incoming messages for all the variables.