

Graphical models

Kirsty McNaught

November 2016

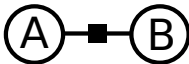
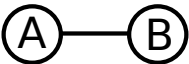
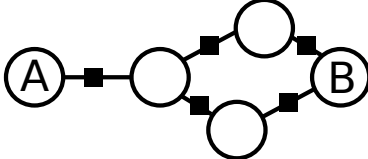
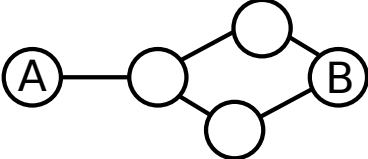
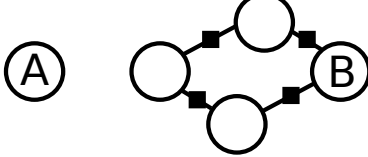
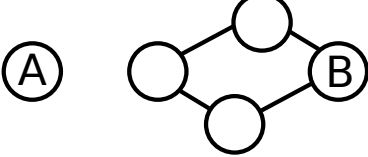
1 Finding independence in graphs

1.1 Marginal independence



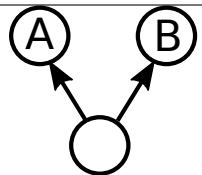
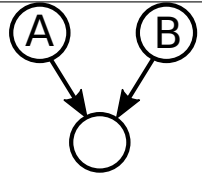
A and B are said to be marginally independent if observing something about B doesn't tell me anything about P(A). Recall the formal definition of independence:

$$A \perp\!\!\!\perp B \Leftrightarrow P(A, B) = P(A)P(B) \quad (1)$$

In any undirected graph (including factor graphs), two nodes are independent if and only if there is no path connecting them.

| | | |
|---|--|----------------------------|
|  |  | $A \not\perp\!\!\!\perp B$ |
|  |  | $A \not\perp\!\!\!\perp B$ |
|  |  | $A \perp\!\!\!\perp B$ |

In directed graphs, we have to be a little more careful. If the only path from A to B goes through a 'collider node', then A and B are independent - the only shared relationship is a shared descendant. In all other cases, the rules are as for an undirected graph.

| | |
|---|-----------------|
|  | $A \not\perp B$ |
|  | $A \not\perp B$ |
|  | $A \not\perp B$ |
|  | $A \perp B$ |

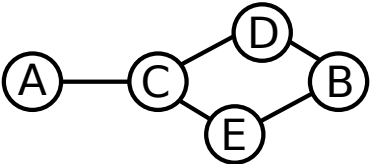
1.2 Conditional independence

A and B are said to be conditionally independent given C, if, assuming I already know C, observing something about B doesn't tell me anything more about P(A). The equivalent definition of independence is:

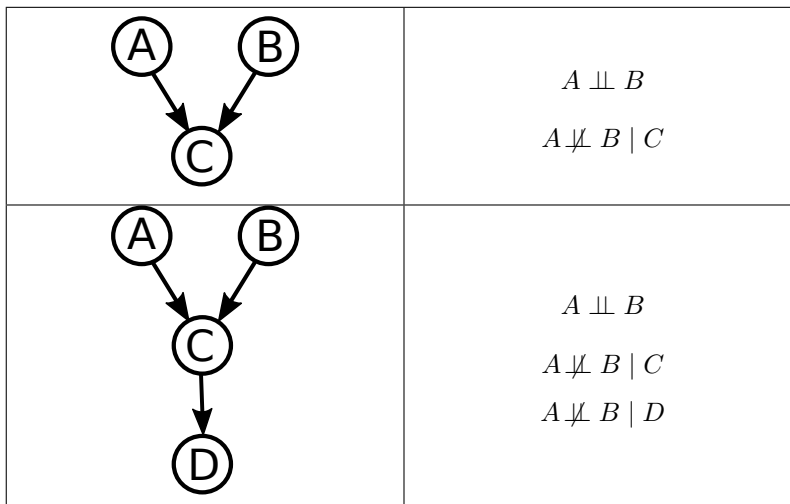
$$A \perp B \mid C \Leftrightarrow P(A, B \mid C) = P(A)P(B) \quad (2)$$

In this case, C may be a single node or a set of multiple nodes.

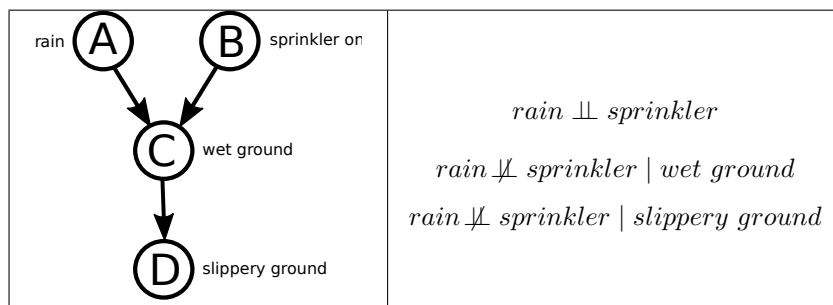
In an undirected graph (including factor graphs), two nodes are conditionally independent if all paths between them are blocked by a node in the conditioning set.

| | |
|---|--|
|  | $A \perp B \mid C$ $A \perp B \mid \{D, E\}$ $A \perp B \mid \{C, D\}$ $A \not\perp B \mid D$ |
|---|--|

In a directed graph, we again have similar rules, with a special case for collider nodes. Let's revisit the collider node on it's own and check the intuition:



To understand this scenario more clearly, let's look at a concrete example.



In this model, there are two independent causes of wet ground (we assume that the sprinklers are not weather-dependent). If we have no observations, the state of the sprinklers and the rain are independent. If, however, we observe that the ground is wet, then finding out that the sprinklers are on reduces the likelihood that it is raining (the sprinklers have 'explained away' the cause of the wetness). If we merely observe that the ground is slippery, that increases our likelihood that the ground is wet, and the rest follows.

With this in mind, we can write down some rules for testing independence in a general directed graph. First of all, remember that in a case *without* collider nodes, independence arises when an observed node *blocks* the path from A to B. In the case of a collider node, we have the opposite effect - a collider node already induces independence purely by existing (unobserved) on the path from A to B. If however we observe the collider node - or any of its descendants - then we lose this independence.

A path is **blocked** by a set of nodes \mathcal{V} if either:

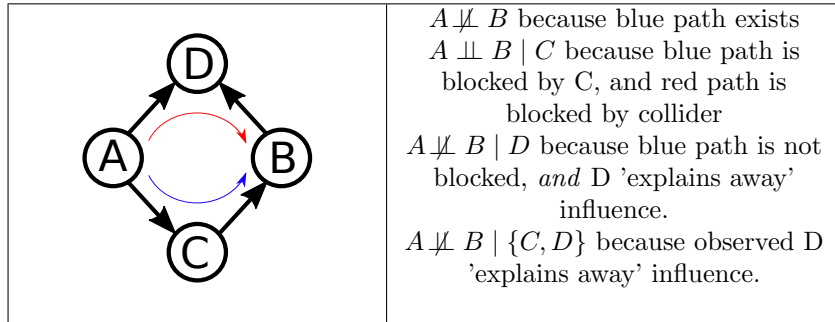
- There is a collider node Z on the path, and neither Z nor any of its

descendants $\in \mathcal{V}$; or

- The path does *not* have any collider nodes, and there is an observed node on the path, $Z \in \mathcal{V}$

A and B are conditionally independent on \mathcal{V} if all paths from A to B are blocked.

Let's look at an example with multiple paths:



2 Joint probabilities over graphs

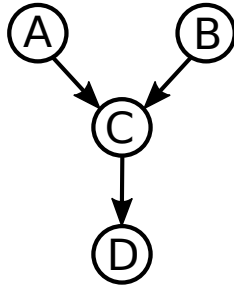
As well as telling us about independent relationships in the model, graphs also tell us how the joint probability over the whole model can be factorised.

2.1 Directed graphs

A directed acyclic graph represents a factorisation of the joint probability distribution in terms of conditionals. In a general case, we have the following definition via the chain rule, so long as A, B, C, D have an appropriate ordering:

$$\begin{aligned}
 P(A, B, C, D) &= P(D \mid C, A, B)P(C, A, B) \\
 &= P(D \mid C, A, B)P(C \mid A, B)P(A, B) \\
 &= P(D \mid C, A, B)P(C \mid A, B)P(A \mid B)P(B) \tag{3}
 \end{aligned}$$

If we have a graph to tell us about marginal independences, we can remove a lot of the conditionals in the joint, as below:



$$P(A, B, C, D) = P(D | C, A, B)P(C | A, B)P(A|B)P(B)$$

$$P(A, B, C, D) = P(D | C)P(C | A, B)P(A)P(B) \quad (4)$$

You should be able to see that this joint can be expressed more concisely as:

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{all parents}) \quad (5)$$

2.2 Undirected graphs

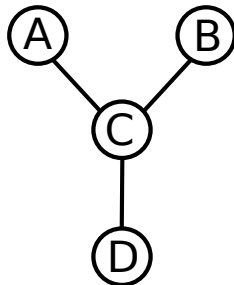
In the case of an undirected graph, we do not know about the order of causation - we just know that some variables interact. We express the joint over the whole graph via clique potentials.

product of potentials over maximal clique

clique is maximally connected subgraph

tree width = number of nodes in largest clique of junction tree minus 1.

we have to marginalise over this many variables in our final node in the JT algorithm, so the tree width determines the efficiency of the search.



$$P(X_1, X_2, \dots, X_n) = \frac{1}{Z} \{ \psi_{A,C}(A, C) \psi_{B,C}(B, C) \psi_{C,D}(C, D) \} \quad (6)$$

$$P(X_1, X_2, \dots, X_n) = \frac{1}{Z} \prod_{\text{cliques } c} \psi_c(X_c) \quad (7)$$

2.3 Factor graph

Factor: product of factors

2.4 Junction tree

Undirected graphs; edges tell us dependencies, which factorise our joint. Also express as product over cliques.

Factor graphs; factors give us probabilities, joint = product over factors.

Directed graphs;

traingulated: every cycle of four or larger has a chord.

Junction tree - require tree structure for belief propagation, but with each super-node you'll still need to marginalise out stuff you don't care about.

"markov equivalent" - has same conditional and marginal relationships, but factorisation may be different.

3 Inference over graphs

3.1 Reminder of useful probability manipulations

Marginalisation

$$P(A) = \sum_{B'} P(A, B') = \sum_{B'} P(A | B')P(B') \quad (8)$$

Bayes

$$P(A | B) = \frac{P(B, A)}{P(B)} = \frac{P(B | A)P(A)}{P(B)} \quad (9)$$

Marginal independence

$$A \perp\!\!\!\perp B \Leftrightarrow P(A, B) = P(A)P(B) \quad (10)$$

$$A \perp\!\!\!\perp B \Leftrightarrow P(A | B) = P(A) \quad (11)$$

$$A \perp\!\!\!\perp B \Leftrightarrow P(C | A, B) = P(C | A)P(C | A) \quad (12)$$

Conditional independence

$$A \perp\!\!\!\perp B | C \Leftrightarrow P(A, B | C) = P(A | C)P(B | C) \quad (13)$$

$$A \perp\!\!\!\perp B | C \Leftrightarrow P(A | B, C) = P(A | C) \quad (14)$$

$$A \perp\!\!\!\perp B | C \Leftrightarrow P(A, B, C) = P(A, C)P(B, C) \quad (15)$$

4 Marginalising over a variable in a graph

4.1 Undirected graph

This is easy. Just fully connect all neighbours of the node being conditioned on, and erase the node. This is what we do when we use variable elimination to triangulate a graph.

4.2 DAG

Who knows? Is it exactly the same as an undirected graph? Probably...

5 Conditioning on a variable in a graph

Sometimes we want to redraw a graph, conditioned on one or more variables. In any case, the conditioned graph must exhibit the marginal independences implied by the conditional independences of the original graph, so if in doubt, refer back to the rules for conditional independence structures.

5.1 Undirected graph

This is easy. Just erase the node and all its edges.

5.2 DAG

This is slightly trickier. To ensure we don't lose conditional independences, we need to make sure we add moralising edges to any parents or ancestors of the conditioning node.

5.2.1 Recipe for conditioning on node C

- If C is a collider, moralise it (join all pairs of parents).
- For any ancestor of C which is a collider node, moralise all pairs of parents.
- Erase C and all its edges.

6 Junction trees

6.1 Creating a junction tree

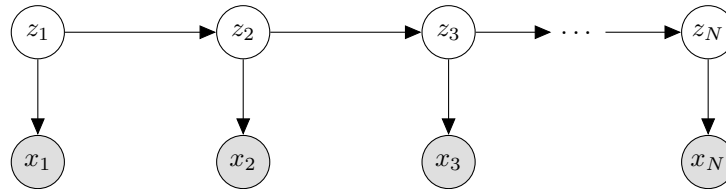
- Create undirected graph by moralising undirected graph and replacing all directed edges with undirected one.
- Triangulate by eliminating nodes and connecting all their parents, until no nodes are left.
- Create clique graph by identifying maximal cliques, separators are given by nodes shared between cliques.
- Remove weaker edges from junction graph until we're left with a max weight spanning tree (weights = number of nodes in separator).
- Double-check running intersection property holds

6.2 Inference on Junction trees

- Assign potentials to each clique node, and to each separator:
 - All separators start with potential 1
 - All clique nodes are assigned at least one conditional or marginal probability (factor) from the original directed graph, where each factor goes to one of the possible cliques containing the variable. E.g. given $p(a | b)p(b | c)p(c)$, you'd assign $p(a | b)$ to a clique containing a , $p(b | c)$ to a clique containing b and $p(c)$ to a clique containing c . If you assign more than one probability to a clique, multiply them. Note that this step probably has more than one possible solution.
- something

7 Message passing on a HMM

A Hidden Markov model is a Markovian time series model with discrete latent variables. Typically the variables x_k are observed and we wish to infer the latent states z_k .



7.1 Forward algorithm

The forward algorithm computes $P(z_k, x_{1:k})$ using the message recursion. This is called *Bayesian filtering*:

$$\alpha_1(z_1) = p(z_1)p(x_1 | z_1)$$
$$\alpha_k(z_k) \ (k > 1) = \sum_{z_{k-1}} p(z_k | z_{k-1})p(x_k | z_k)\alpha_{k-1}(z_{k-1})$$

7.2 Backward algorithm

The backward algorithm computes $P(x_{k+1:n} | z_k)$ using the message recursion:

$$\beta_N(z_N) = 1$$
$$\beta_k(z_k) \ (k < N) = \sum_{z_{k+1}} p(z_{k+1} | z_k)p(x_{k+1} | z_{k+1})\beta_{k+1}(z_{k+1})$$

7.3 Forward-backward algorithm

A full pass of the forward-backward algorithm allows us to compute marginal posteriors (given *all* the observations) by multiplying the incoming messages. This is called *Bayesian smoothing*:

$$\begin{aligned} P(z_k | x_{1:N}) &\propto P(z_k, x_{1:N}) = P(z_k | x_{1:k})P(x_{k+1:n} | z_k) \\ &= \alpha_k(z_k)\beta_k(z_k) \end{aligned}$$

If we do this for every node, and pick the most likely state at each node, we get a “best” state path. This is the path with the *maximum expected number of correct states*. But this is **not** the single path with the highest probability of generating the data. It may even be a path of probability zero, if one of the transitions is impossible.

7.4 Viterbi algorithm

All of the above recursions hold for computations of the max, rather than sums (this is fine so long as the things being maximised are non-negative, like probabilities). This is what we do if we want to find the *most likely path of hidden states*, i.e. $\operatorname{argmax}_{y_{1:N}} P(y_{1:N}|x_{1:N})$. For example, let’s take the forward messages and swap the sums for max operations, which will give us the maximum value associated with the best path:

$$\begin{aligned} \alpha_1(z_1) &= p(z_1)p(x_1 | z_1) \\ \alpha_k(z_k) \text{ (} k > 1 \text{)} &= \max_{z_{k-1}} p(z_k | z_{k-1})p(x_k | z_k)\alpha_{k-1}(z_{k-1}) \end{aligned}$$