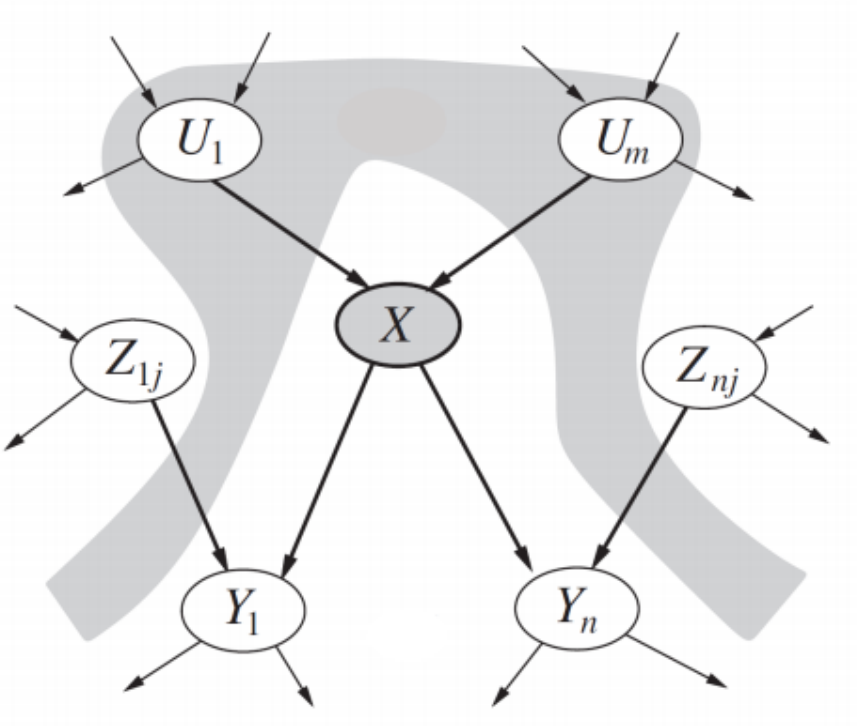
# Introduction to Bayesian Networks

Devin Soni Follow
Jun 8, 2018 · 5 min read




**Bayesian networks** are a type of probabilistic graphical model that uses Bayesian inference for probability computations. Bayesian

networks aim to model conditional dependence, and therefore causation, by representing conditional dependence by edges in a directed graph. Through these relationships, one can efficiently conduct inference on the random variables in the graph through the use of factors.

. . .

## Probability

Before going into exactly what a Bayesian network is, it is first useful to review probability theory.

First, remember that the joint probability distribution of random variables A_0, A_1, …, A_n, denoted as P(A_0, A_1, …, A_n), is equal to P(A_1 | A_2, …, A_n) * P(A_2 | A_3, …, A_n) * … * P(A_n) by the chain rule of probability. We can consider this a **factorized** representation of the distribution, since it is a product of N factors that are localized probabilities.

$$\mathrm{P}\left(\bigcap_{k=1}^{n} A_k\right) = \prod_{k=1}^{n} \mathrm{P}\left(A_k \,\middle|\, \bigcap_{j=1}^{k-1} A_j\right)$$

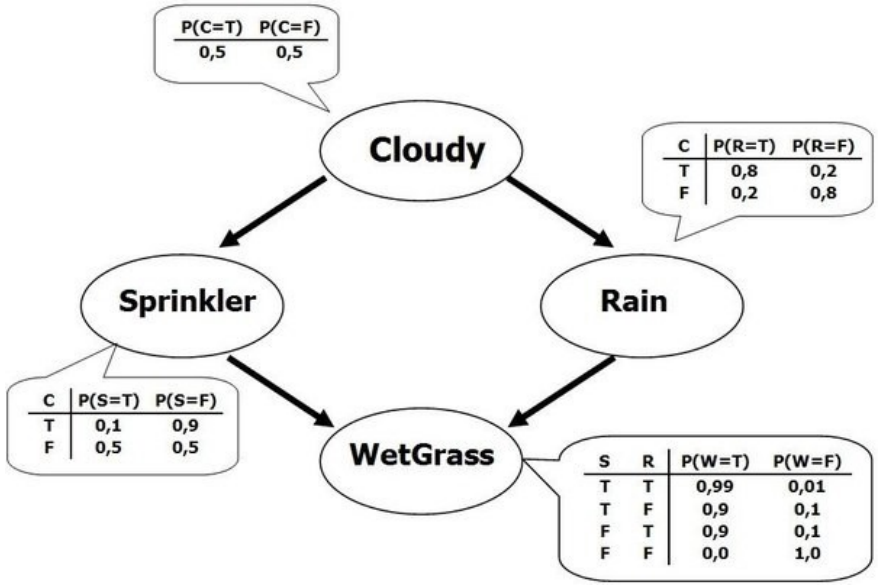
Next, recall that **conditional independence** between two random variables, A and B, given another random variable, C, is equivalent to satisfying the following property: P(A,B|C) = P(A|C) * P(B|C). In

other words, as long as the value of C is known and fixed, A and B are independent. Another way of stating this, which we will use later on, is that $P(A|B,C) = P(A|C)$.

. . .

## The Bayesian Network

Using the relationships specified by our Bayesian network, we can obtain a compact, factorized representation of the joint probability distribution by taking advantage of conditional independence.



A Bayesian network is a **directed acyclic graph** in which each edge corresponds to a conditional dependency, and each node corresponds to a unique random variable. Formally, if an edge (A, B) exists in the graph connecting random variables A and B, it means that P(B|A) is a **factor** in the joint probability distribution, so we must know P(B|A) for all values of B and A in order to conduct inference. In the above example, since Rain has an edge going into WetGrass, it means that P(WetGrass|Rain) will be a factor, whose probability values are specified next to the WetGrass node in a conditional probability table.

Bayesian networks satisfy the **local Markov property**, which states that a node is conditionally independent of its non-descendants given its parents. In the above example, this means that P(Sprinkler|Cloudy, Rain) = P(Sprinkler|Cloudy) since Sprinkler is conditionally independent of its non-descendant, Rain, given Cloudy. This property allows us to simplify the joint distribution, obtained in the previous section using the chain rule, to a smaller form. After simplification, the joint distribution for a Bayesian network is equal to the product of P(node|parents(node)) for all nodes, stated below:

$$P(X_1,\ldots,X_n)=\prod_{i=1}^{n}P\left(X_i \mid X_1,\ldots,X_{i-1}\right)=\prod_{i=1}^{n}P\left(X_i \mid Parents(X_i)\right)$$

In larger networks, this property allows us to greatly reduce the amount of required computation, since generally, most nodes will have few parents relative to the overall size of the network.

. . .

## Inference

Inference over a Bayesian network can come in two forms.

The first is simply evaluating the joint probability of a particular assignment of values for each variable (or a subset) in the network. For this, we already have a factorized form of the joint distribution, so we simply evaluate that product using the provided conditional probabilities. If we only care about a subset of variables, we will need to marginalize out the ones we are not interested in. In many cases, this may result in underflow, so it is common to take the logarithm of that product, which is equivalent to adding up the individual logarithms of each term in the product.

The second, more interesting inference task, is to find P(x|e), or, to find the probability of some assignment of a subset of the variables (x) given assignments of other variables (our evidence, e). In the above example, an example of this could be to find P(Sprinkler, WetGrass | Cloudy), where {Sprinkler, WetGrass} is our x, and {Cloudy} is our e. In order to calculate this, we use the fact that P(x|e) = P(x, e) / P(e) = αP(x, e), where α is a normalization constant that we will calculate at the end such that P(x|e) + P(¬x | e) = 1. In order to calculate P(x, e), we must marginalize the joint probability distribution over the variables that do not appear in x or e, which we will denote as Y.

$$P(x|e) = \alpha \sum_{\forall y \in Y} P(x, e, Y)$$

For the given example, we can calculate P(Sprinkler, WetGrass | Cloudy) as follows:

$$P(Sprinkler, WetGrass \mid Cloudy) = \alpha \sum_{Rain} P(WetGrass|Sprinkler, Rain)P(Sprinkler|Cloudy)P(Rain|Cloudy)P(Cloudy)$$
$$= \alpha P(WetGrass|Sprinkler, Rain)P(Sprinkler|Cloudy)P(Rain|Cloudy)P(Cloudy) + \alpha P(WetGrass|Sprinkler, \neg Rain)P(Sprinkler|Cloudy)P(\neg Rain|Cloudy)P(Cloudy)$$

We would calculate P(¬x | e) in the same fashion, just setting the value of the variables in x to false instead of true. Once both P(x | e) and P(¬x | e) are calculated, we can solve for α, which equals 1 / (P(x | e) + P(¬x | e)).

Note that in larger networks, Y will most likely be quite large, since most inference tasks will only directly use a small subset of the variables. In cases like these, exact inference as shown above is very computationally intensive, so methods must be used to reduce the amount of computation. One more efficient method of exact inference is through variable elimination, which takes advantage of the fact that each factor only involves a small number of variables. This means that the summations can be rearranged such that only factors involving a given variable are used in the marginalization of that variable. Alternatively, many networks are too large even for this method, so approximate inference methods such as MCMC are instead used; these

provide probability estimations that require significantly less computation than exact inference methods.

. . .

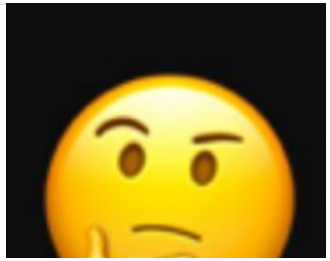
Make sure you give this post **50 claps** and **follow me** if you enjoyed this post and want to see more!

Devin Soni — Medium

Read writing from Devin Soni on Medium. crypto markets, data science ☞ twitter @devin_soni ☞ website...

medium.com

You can also follow me on **Twitter**!



Devin Soni (@devin_soni) | Twitter

The latest Tweets from Devin Soni (@devin_soni).

https://t.co/Q3YFB0nWiL

twitter.com