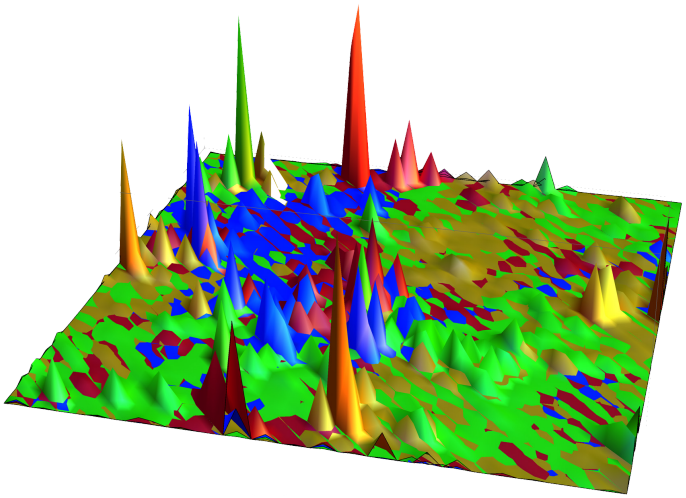
Introduction to Bayesian Networks Part 1: Theory and Algorithms

Phil Russell
Principal Big Data Software Engineer
AT&T Chief Data Office
pr144b@att.com

> *The theory of probabilities is at bottom nothing but common sense reduced to calculus; it enables us to appreciate with exactness that which accurate minds feel with a sort of instinct for which ofttimes they are unable to account.* [1]
>
> Pierre Simon Laplace, 1819

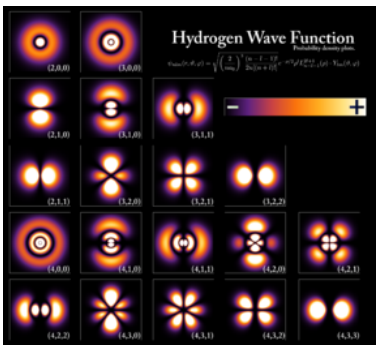
The stochastic nature of phenomena has always fascinated me. Some things appear to work deterministically, but there's so much that deterministic theories can't explain.

In my mind I sometimes visualize the world as a vast quilt of intersecting probability distributions. This picture, though about something else entirely, comes close to my mental image:



For a long time I searched for a formalism that would capture this idea, and when I encountered **Bayesian networks**, it seemed to fit my mental image in a very satisfying way.

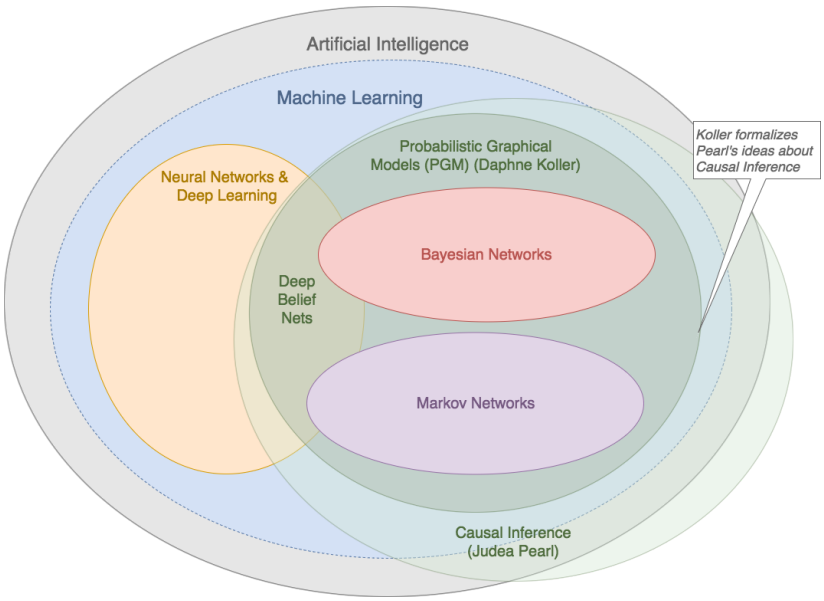
[1] Epigraph borrowed from Daphne Koller and Nir Friedman, *Probabilistic Graphical Models*.



From quantum mechanics we've learned that sub-atomic physics can be understood as probabilistic wave functions.
Source: *Wikipedia, Quantum mechanics*

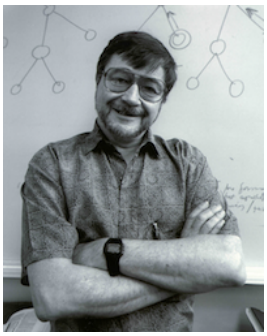
Context

Let's set Bayesian networks in the context of other kinds of network modeling.

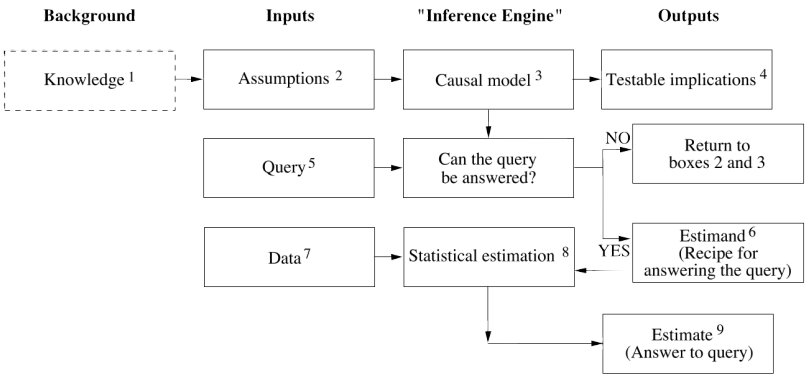


Bayesian networks are fundamentally different from **neural networks** in that the calculus that describes reasoning flows in the network is probabilistic, not deterministic. Inside the black box of a neural net are nodes organized into a multi-tier network. In a learned model, these nodes have immutable numeric weights that condition reasoning, and the network will always produce the same result for the same input.

Judea Pearl has been a thought leader and crusader in the field of **causal inference**. Pearl believes that the language of causality was unwelcome in scientific circles for far too long, and that we are now in the midst of a "causal revolution". He makes this case in the book *Causality: Models, Reasoning and Inference*, in which he also provides a basic mathematical framework for formulating and solving various problems of causal determination.



Judea Pearl, UCLA Computer Science Department, Cognitive Systems Lab



Pearl's framework includes this process for addressing causal questions.

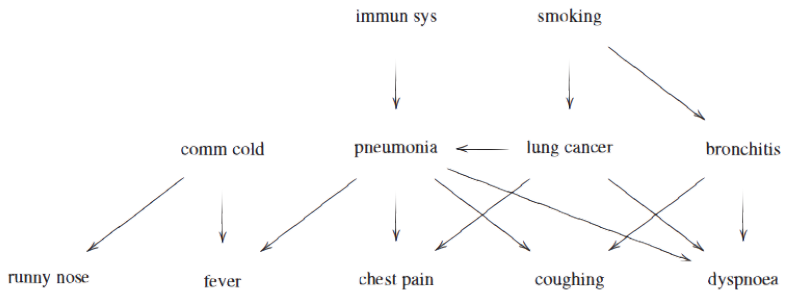
Daphne Koller and **Nir Friedman** build on Pearl's concepts and provide a comprehensive formalism in their book *Probabilistic Graphical Models:*

Principles and Techniques. Published in 2009, this book remains the seminal text on **probabilistic graphical models** (PGMs). It covers both **Markov networks** and Bayesian networks.

Markov networks differ from Bayesian networks in that the edges connecting the nodes of the network are non-directional, while Bayesian networks use directional nodes.

Daphne Koller, Stanford Computer Science Department

The Basics



A basic Bayesian network

Bayesian networks are a form of **knowledge representation**.

Bayesian network are a way of representing the *structure* of inter-related **joint probability distributions**.

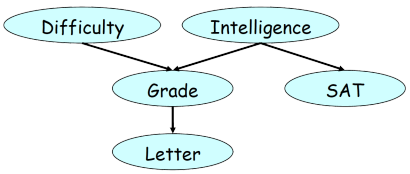
A Bayesian network is a graph in which the nodes represent **probabilistic (a.k.a. random) variables** and the edges represent **causal chains**. This network must be a **directed acyclic graph** (DAG).

Bayesian networks are an extension of **Bayesian inference**, in which the **posterior probability** of a **random variable** is derived from the **prior probabilities** of the variable and an antecedent variable, along with an observed value of the antecedent variable. In Bayesian networks this concept is expanded to describe how these probabilities work over a network of many related random variables.

The variables represent measures for which we have sufficient data to form a **probability distribution**. This distribution can be **continuous** or **discrete**. For simplicity, we'll stick to discrete distributions. This means that our data has finite cardinality or that the continuous values have been categorized into a finite number of buckets.

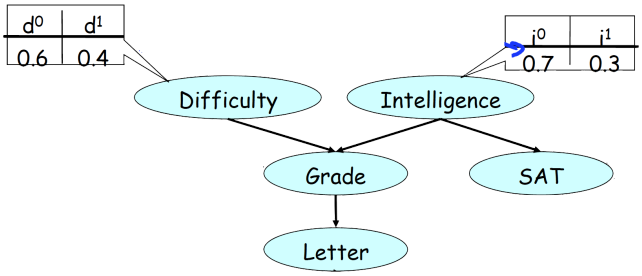
Of primary importance is how the nodes are arranged and connected with directional edges. Later we'll discuss methods for doing this. For now, let's just assume that our network accurately represents causality.

To the right we see a simple Bayesian network that Koller uses in her course. Here we want to reason about the interplay between various factors that affect and/or measure academic performance. In this model, course difficulty (*D*) and the student's intelligence (*I*) are thought to influence the grade (*G*) the student receives. Intelligence influences the SAT (*S*) score the student receives. And the student's grade influences the quality of the recommendation letters (*L*) she elicits.



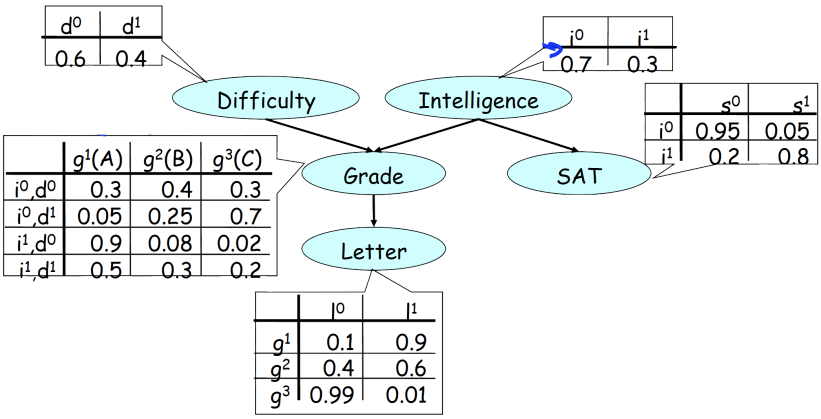
Bayesian network that models academic performance
Source: *Coursera: Probabilistic Graphical Models 1, Daphne Koller*

Now let's assign some probability distributions to these variables. The top two nodes, *D* and *I* aren't influenced by other variables so we can assign a simple discrete probability distribution to each. By querying the course data, we find that 60% of courses have a difficulty rating of d_0 and 40% have a difficulty rating of d_1. We'll call this a **simple probability distribution** (SPD) to distinguish it from other distribution types. We similarly determine the SPD for *I*.



A basic Bayesian network
Source: *Coursera: Probabilistic Graphical Models 1, Daphne Koller*

For the other nodes, *G*, *S*, and *L*, we want our probabilities to reflect the different possible values of *D* and *I*. For these nodes we use **conditional probability distributions** (CPDs).



A basic Bayesian network
Source: *Coursera: Probabilistic Graphical Models 1, Daphne Koller*

Look first at the CPD for *S*, the SAT score variable. Since *I* can have two possible values, we want to know the distribution of values *S* for each value, i_0 and i_1. So our CPD table has *two* rows instead of one. Each row gives the distribution of *S* for a different value of *I*. The *S* distributions are **conditioned** by the values of *I*. Thus the name, "conditional probability distribution."

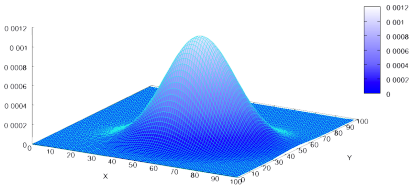
Things get a little more complicated when a node has two predecessors. For node *G* we now have *four* rows in our CPD table, representing each possible combination of *I* and *D* values.

The Chain Rule

A useful property of this network is the **chain rule**, which allows us to compute the **joint probability distribution** (JPD) of any combination of random variables by multiplying their CPDs.

Let's represent the probability distribution for our nodes like this:

Difficulty	$P(D)$
Intelligence	$P(I)$
Grade	$P(G \mid I,D)$
SAT	$P(S \mid I)$
Letter	$P(L \mid G)$



If we were working with continuous variables, the JPD for two variables (here *X* and *Y*) might look something like this. Compare this with my "mental image" at the beginning of this article. Now consider what a JPD might look like for four or five random variables! Difficult to visualize, but the chain rule still holds.

The vertical bar in these expressions can be read "given" or "conditioned on." So the JPD for the Grade node *G* reads "probability of *G* given *I* and *D*."

So mathematically, we can say things like:

$$P(D,G) = P(D)\,P(G \mid I,D)$$

which reads "the joint probability of *D* and *G* is equal to the product of the *D*'s simple probability distribution and *G*'s conditional probability distribution, conditioned on *I* and *D*"; and

$$P(D,I,G,S,L) = P(D)\,P(I)\,P(G \mid I,D)\,P(S \mid I)\,P(L \mid G).$$

In the literature, the JPDs to the right of the equals signs are often referred to as **factors**, even before we start multiplying them together.

Bayesian Network Reasoning

Now for the fun part. And this is the *most important* aspect of your understanding and use of Bayesian networks.

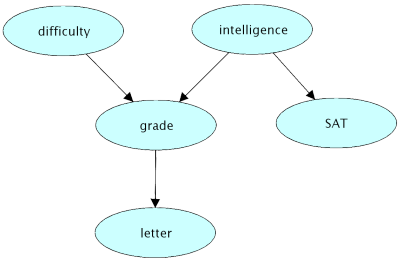
Causal Reasoning

Let's say we've trained our Bayesian network using historical data from our database. Each of our nodes has a probability distribution that approximates reality.

Now suppose some new data comes in. A new student walks in the door and tells you her IQ, which is what we're using as a measure of intelligence. When we plug this value into the model, we change one of the random variables into a known constant. How does this affect our reasoning?

To explore this, we'll turn to a tool called **SamIAm**, a Java app developed by UCLA's Automated Reasoning Group. Here we'll show some screen shots from the app but we won't get into the details of how to use it.

To the right is our initial model setup in SamIAm. We've created and arranged our five nodes and connected them with directional arrows.



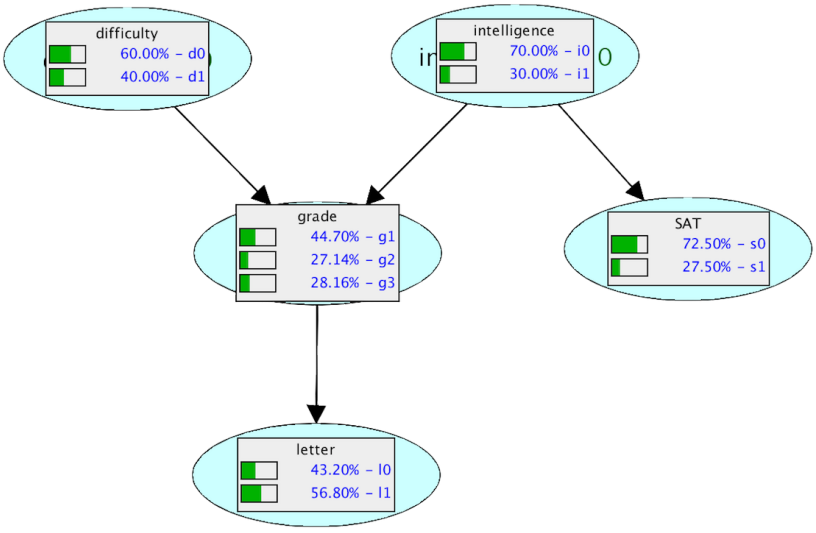
Initial model setup in SamIAm

For each node we input the conditional probability values. Based on the network topology, SamIAm creates a **conditional probability table** (CPT) with all of the right value combinations. Here's the conditional probability table for the Grade variable *G*:

difficulty	d0		d1	
intelligence	i0	i1	i0	i1
g1	0.3	0.05	0.9	0.5
g2	0.4	0.25	0.08	0.3
g3	0.3	0.7	0.02	0.2

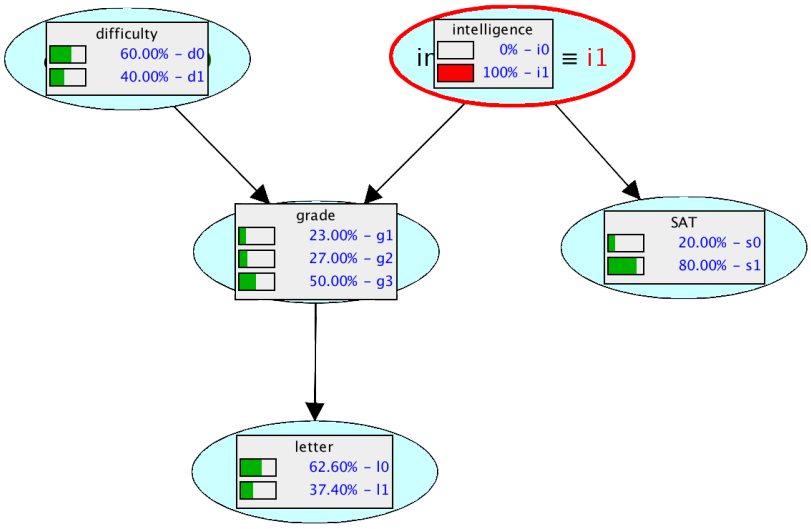
We enter the values from the picture of conditional probability distributions in a previous section. Observe that the CPT is similar to the one in the previous picture, except that the horizontal and vertical labels are swapped.

After entering all of the CPT values, we compile the model and switch to query mode. Here we can see the calculated probabilities for each node.

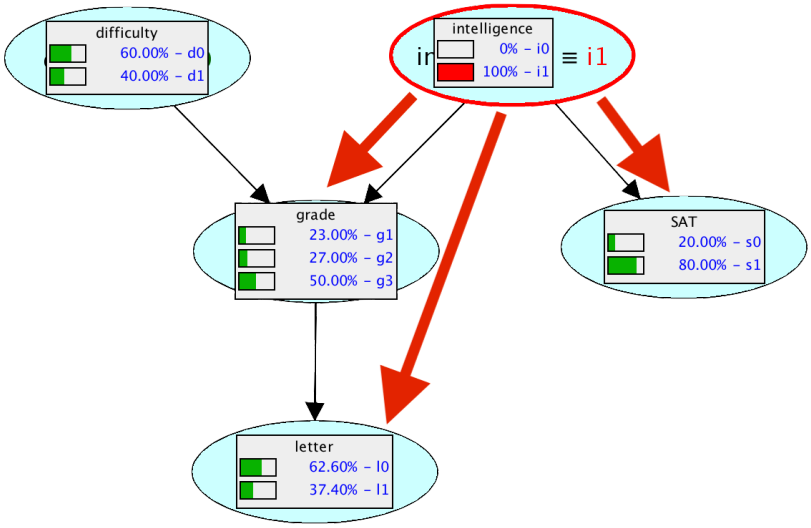


For the three nodes that have conditional probability distributions, *G*, *L*, and *S*, we see the calculated probabilities rather than the CPT. So, for example, *before* a new student walks in the door, we can say that the probability of that student receiving a grade of g_1 is 44.7%.

Now let's see what happens when our new student actually shows up. The IQ number she gave us falls into the category we call g_1. How does this new information affect the probabilities?



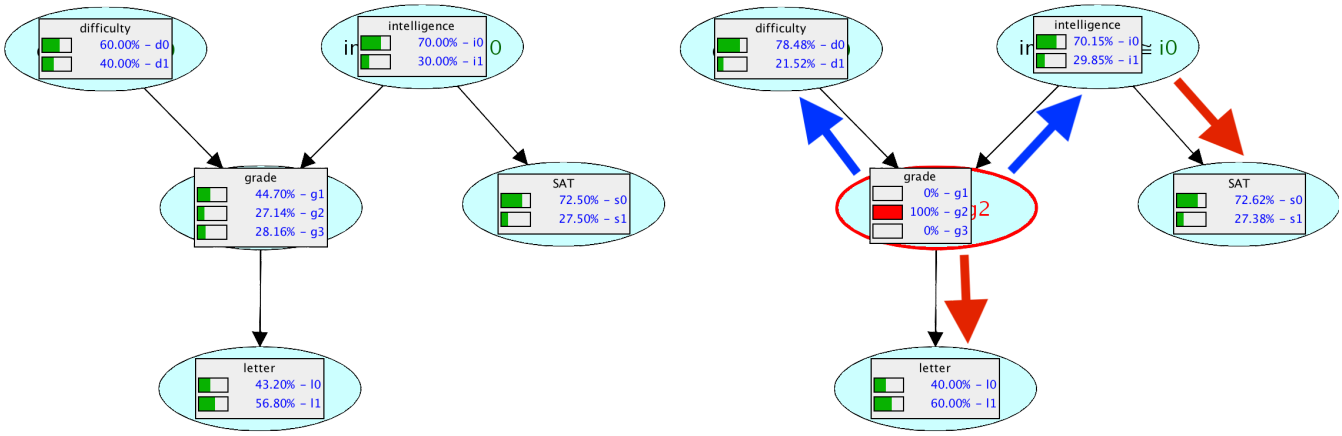
The probabilities have changed in every node that is downstream of *I*. We've just witnessed an example of the **flow of probabilistic influence**. Selecting a value for *I* influenced the probability distributions for *G*, *L*, and *S*. We refer to these paths of influence as **active trails**. In this case, the active trails are the paths shown here in red:



Probabilistic reasoning that flows in this direction is called **causal reasoning**. It flows in the same direction as our black arrows, the network edges that indicate our view of causal influences.

Evidential Reasoning

Suppose another student walks in the door and shows us a report card where having a grade of g_2. We know nothing else about this student. What happens to the other nodes' distributions? Le't compare the new distributions with the original ones.



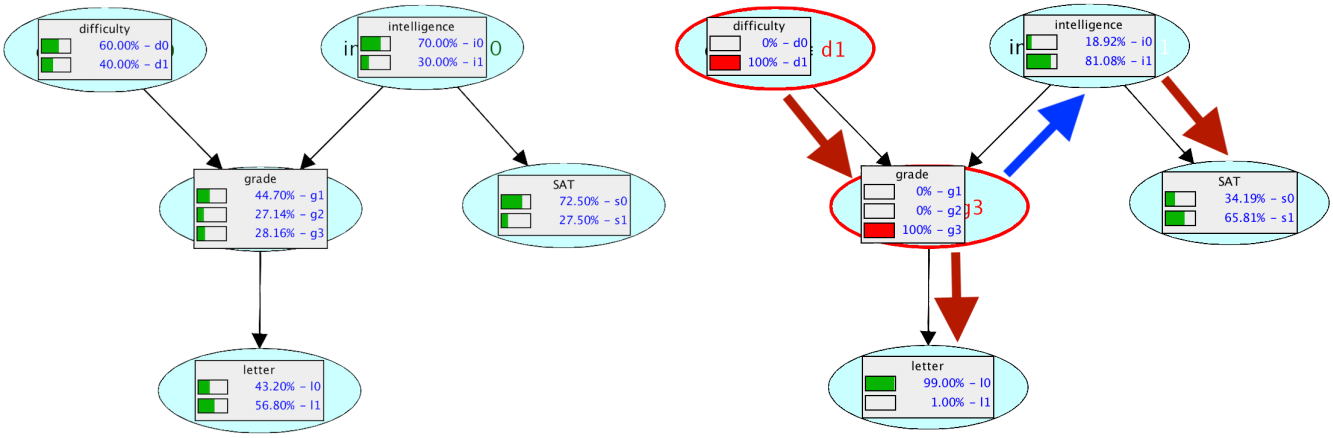
All of the other nodes' distributions change. We observe a new kind of reasoning that flows "uphill", against the direction of the causal influences. This is called **evidential reasoning** and is indicated on our picture with blue arrows.

An important distinction to make is that *we haven't changed the direction of causal influence*. The black arrows remain as they were. What *has* changed is our view of the upstream probability distributions. The reasoning goes like this: we believe, for example, that course difficulty D influences the student's grade G. Knowing that the student has a grade of g_2 is *evidence* that leads us to believe that it is more likely that the course was difficult.

We now believe that the difficulty D distribution is different from what it was before. Knowing the student got a g_2 on his report card makes us think that the likelihood that the course was difficult is 78.48% instead of what we originally thought, which was 60%.

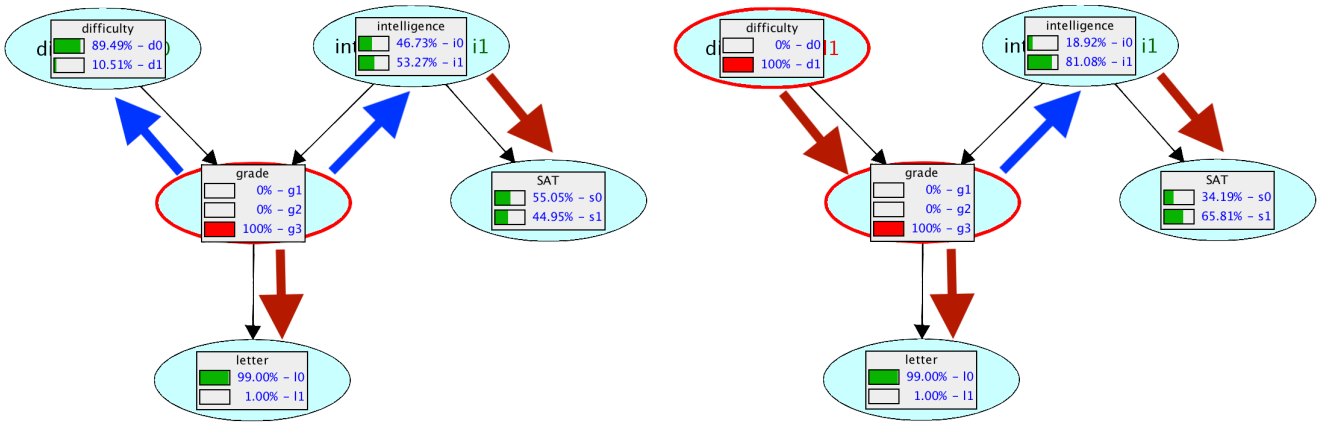
Intercausal Reasoning

Now let's consider what happens when our hypothetical student provides us more than one piece of information. This student brings in a report card showing a poor grade in a particular course, and says, "I don't know how this could have happened; the course was really easy". The grade was a "C", which in our model is denoted by the value g_3. We represent an easy course using a value of d_1. Let's select these values in our model and see what happens.



The values of D and G are fixed, so those nodes aren't the objects of reasoning flows. Fixing these values changes the distributions for all of the other variables. Remarkably, S changes even though it isn't an ancestor or descendent of either D or G. As in the previous example, fixing G influences S because of the evidential reasoning from G to I and the causal reasoning from I to S.

To understand this further, we compare how this scenario works with and without a fixed value of D. In the left hand network below, we don't believe the student's assessment of course difficulty, so we don't fix D. The network on the right is the same as the previous example.

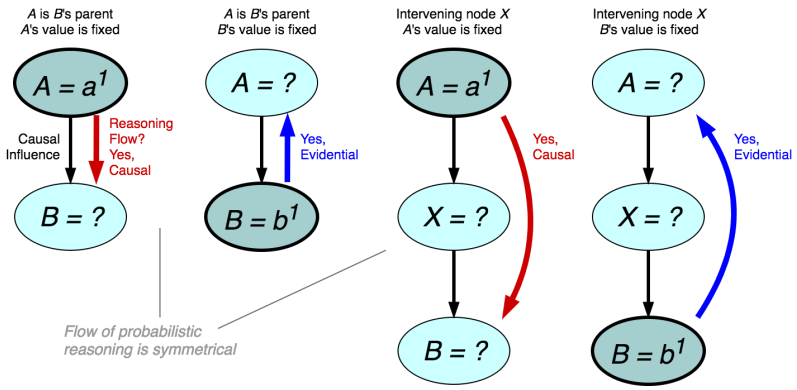


This clearly shows the indirect influence that D has on S. Even with a fixed value of G, fixing D changes the probability distribution of S! So fixing the value of G doesn't interrupt the flow of probabilistic reasoning. The path between D and S is an active trail.

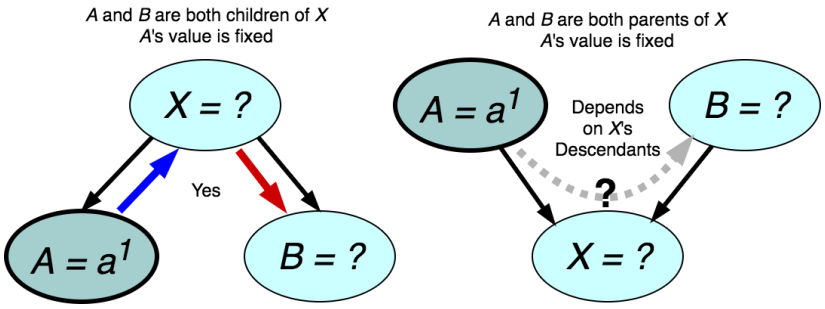
Intuition tells us that this result is reasonable. In the left image above, without fixing the value of D, our default assessment of the course's difficulty was an 89.49% likelihood that the course was difficult. Knowing the student did poorly in an *easy* course lowers our opinion about the probability that they received a good SAT score. Once we stipulate that the course was indeed easy, we now think differently about the SAT score probability. We think that the likelihood of the student having received a good SAT score is 34.19%, not 55% that we thought was an accurate probability before we knew how easy the course was.

Flow Rules

We could continue creating many such examples that illustrate the way probabilistic reasoning can flow in a Bayesian Network. After trying many combinations of fixed and non-fixed values, we can formulate some rules about the circumstances under which a flow of probabilistic reasoning (a.k.a. an active trail) can exist.

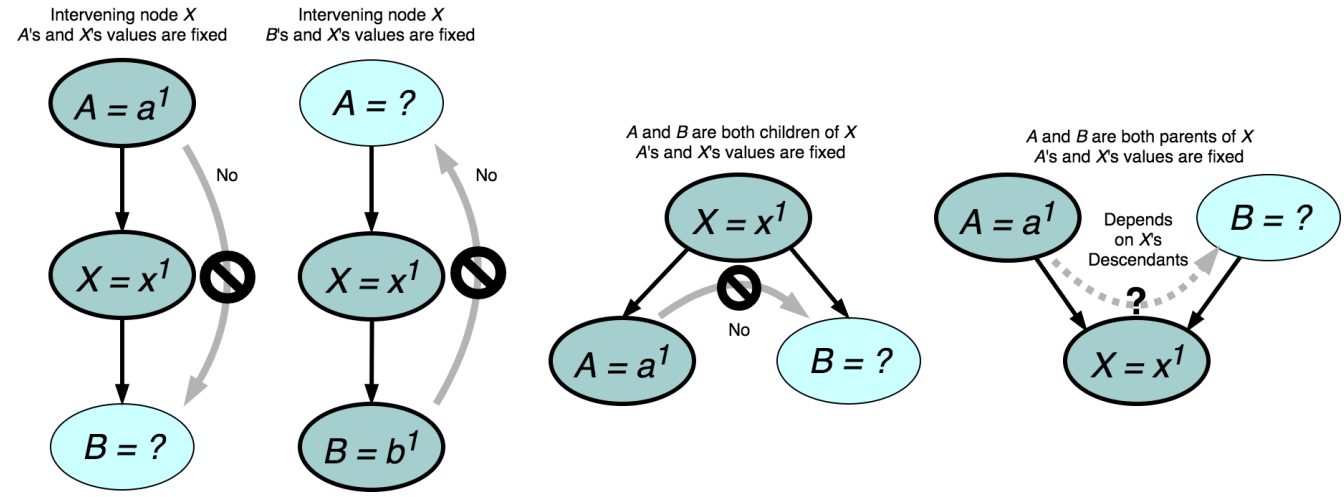


These first four cases depicted above involve direct descendants. Probabilistic reasoning can flow between A and B regardless of whether there is an intervening node X. The flow is symmetrical and can just as readily flow from B to A.



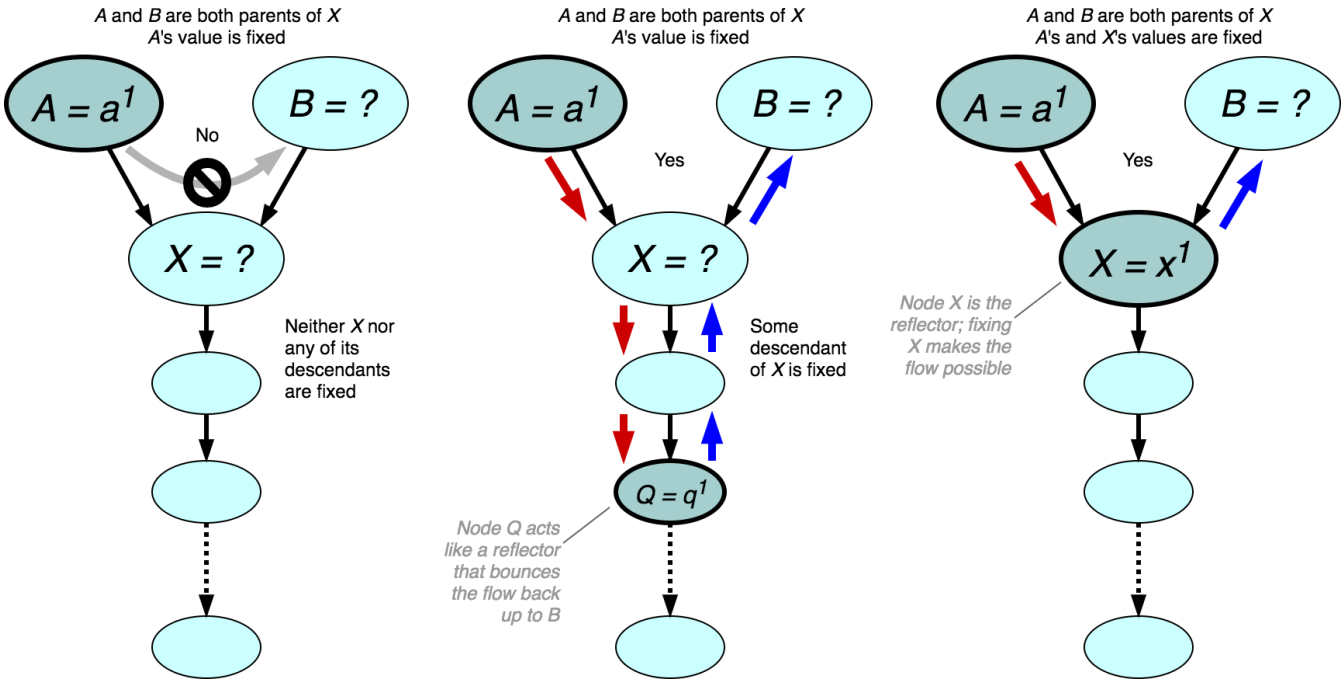
Things get a little more interesting when A and B are both children or both parents of X. The diagram at the right is a **V-structure** which we will discuss in greater detail momentarily.

Now let's see what happens when we fix the values for two nodes:



In the first three cases we see that fixing X *blocks* the flow of probabilistic reasoning. In each case the flow now starts and node X with its fixed value, and thus cannot start at A or B.

The rightmost case above is the V-structure which we will now examine in greater detail:

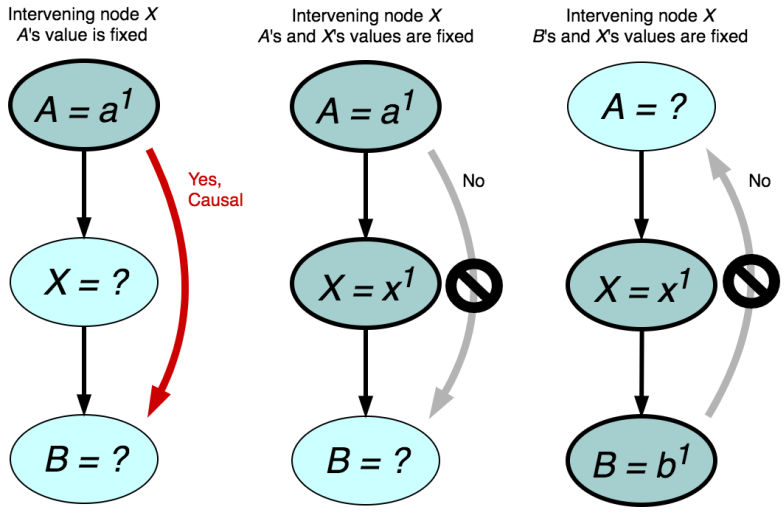


In a V-structure the reasoning can flow only if node X or one of its descendants is fixed.

Independencies

The nodal relationships depicted above can also be described in terms of **independence** and **conditional independence**. Nodes that cannot be connected by flows of red and blue arrows are considered to be independent. If this independence depends on whether the value of some intervening node is fixed, it is considered to be conditional. Consider these three examples, repeated from earlier diagrams:

For those of you who prefer abstract cerebration, independencies can be represented using symbolic mathematical expressions building on the **chain rule**, and involving concepts such as **factoring**, **d-separation** and **I-maps**. For the sake of brevity we won't cover that here.



When the value of X is not fixed, a dependency exists between A and B. Fixing X interrupts the flow of probabilistic reasoning, making A and B independent from each other. In this case, A and B are considered to be conditionally independent.

IT IS IMPORTANT TO REMEMBER that a Bayesian network simultaneously hosts two different logical flows: the *flow of causal influence*, depicted in our diagrams by the black arrows, and the *flow of probabilistic reasoning* depicted by the red and blue arrows.

Think of this as being analogous to the **ethernet over power** technology that is sometimes used in home computer networks. These devices are able to send and receive Ethernet signals over the same electrical wires used for the wall outlets. So on the same wiring (or network, if you will) two different things are happening at the same time: electrical power is being distributed throughout the house and Ethernet communication between computer network devices is taking place. In a Bayesian network, flows of causal influence and probabilistic reasoning both operate over the same network topology at the same time.

Ethernet over power devices
Source: *amazon.com*

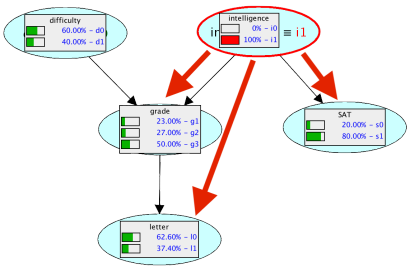
Inference

The examples and patterns we've seen so far all have to do with the Bayesian network's central question: what happens to a node's probability distribution when evidence is provided for another node in the network? In other words, what is a node's *inferred* probability distribution given a known value for another node? The act of determining this is called **inference**. This kind of inference is called **posterior probability**. More formally expressed: posterior probability inference is the *act of updating a random variable's probability distribution based on evidence, that is, fixing the value, of a different, causally-related random variable's probability distribution*.

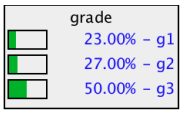
Another type of inference, **most likely explanation**, lets us ask our network which value of some other random variable is most likely to have caused a given value for a given random variable. For purposes of this introduction, we'll limit our discussion to posterior probability.

In the *Bayesian Network Reasoning* section above the inference calculations were performed for us by the SamIAm application. Now let's look at how applications like SamIAm do inference.

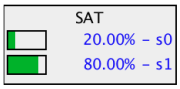
First let's mathematically express the relationships in the example network we've been using, shown to the right. We'll call the random variables *D, I, G, S,* and *L* using the first letter of each node's label. In this example we fix the value of *I*, which changes the distributions for dependent nodes *G, S,* and *L*. We need to calculate the inferred distributions, $P(G \mid I=i_1)$, $P(S \mid I=i_1)$, and $P(L \mid I=i_1)$, for which we've already seen these results from SamIAm:



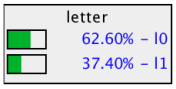
$P(G \mid I{=}i_1) =$



$P(S \mid I{=}i_1) =$



$P(L \mid I{=}i_1) =$

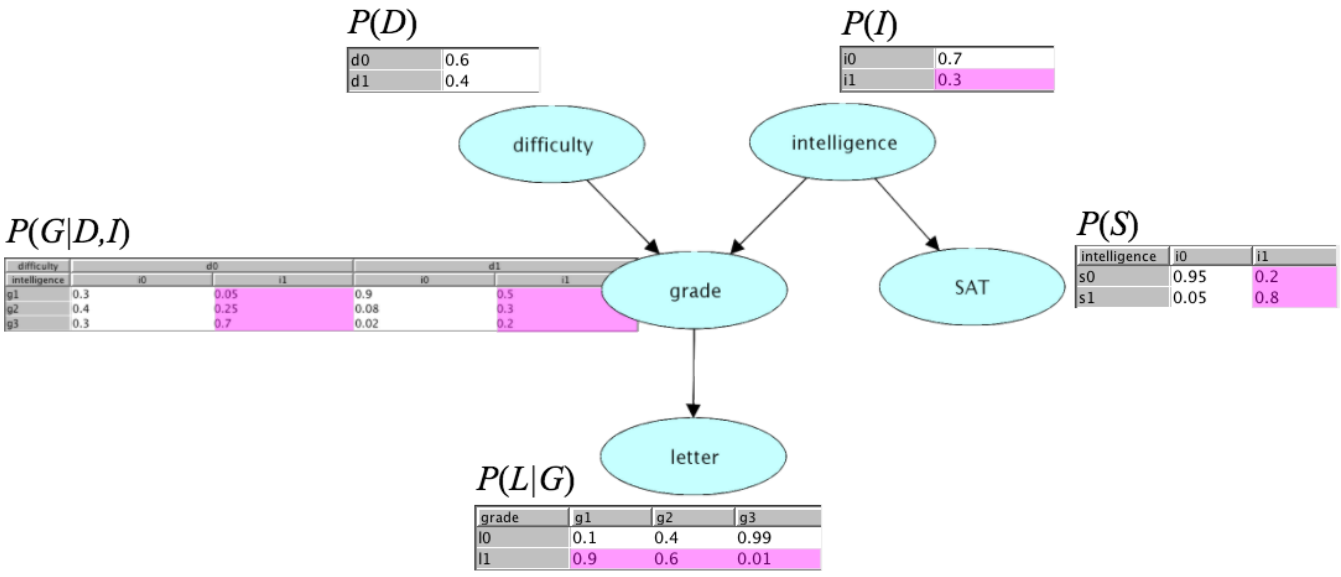


Let's focus on $P(G)$. In this example, G is our **query variable**, I is our **evidence variable**, and D, S, and L are called **hidden variables**, which simply means that they are neither query nor evidence variables.

Inference by Exact Enumeration

The brute force approach to calculating $P(G \mid I{=}i_1)$ is to enumerate every possible combination of values of the other random variables and take the product of those probabilities. Our goal is to update the $P(G)$ conditional probability table, so that we have rows for each of g_1, g_2, *and* g_3 that contain new entries.

In order to visualize the formulas we use, we'll refer to this example. The picture shows the prior probability tables for the example network we've been studying. Here we've highlighted in pink the rows and columns of the probability tables corresponding to the fixed value i_1.



Per the definition of conditional probability:

$$P(G|I) = \frac{P(G, I)}{P(I)}.$$

For a fixed value of I:

$$P(G|I = i_1) = \frac{P(G, i_1)}{P(i_1)}.$$

In order to calculate $P(G)$, we'll need to calculate the conditional probabilities for each row of the table, g_1, g_2, or g_3. We'll use the term g_n to represent any such row, and also shorten our notation to use "$P(x_n)$" instead of "$P(X=x_n)$" for any random variable X. Fixing $G=g_n$,

$$P(G = g_n|I = i_1) = P(g_n|i_1) = \frac{P(g_n, i_1)}{P(i_1)}.$$

Let's separately consider the numerator and the denominator of the right-hand side of this equation.

To get the value of the numerator, $P(g_n,i_1)$, we sum the probabilities of every combination of every *other* random variable in our network.

$$P(g_n, i_1) = \sum_{d=d_0}^{d_1} \sum_{s=s_0}^{s_1} \sum_{l=l_0}^{l_1} P(g_n, i_1, d, s, l).$$

Since I has no conditional probabilities, the denominator, $P(i_1)$ doesn't require further reduction, but if it were the case that I were conditioned by G, we would say,

$$P(i_1) = \sum_{n=1}^{3} P(g_n, i_1).$$

Looking at the numerator formula above with its triple summation, you can begin to see the computational scope of exact inference calculations. The problem is **NP-hard**, and for a network with m random variables that each have n possible values, the computational cost is $O(m^n)$.

Less Expensive Inference Algorithms

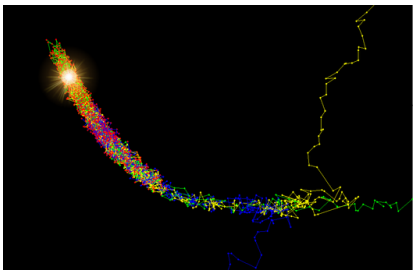
Fortunately there are many ways to reduce this computational cost. Present scope precludes us from providing theoretical depth on these, so here we'll only provide brief descriptions.

Variable Elimination The exact enumeration method is improved by eliminating random variables that are irrelevant to the result. It is further improved by storing and re-using interim results.

Approximation by Direct Sampling Because a Bayesian network is a *generative* model, it allows us to efficiently generate samples from its joint distribution. These samples can be used to approximate conditional probability distributions. A sample result for a given random variable is calculated by randomly selecting values from its probability table. For each sample, a choice is made by classifying the randomizer result according to the distribution. A full sample set contains a single sample result for each of the network's non-fixed random variables. Over a large number of random samples, the frequency of the sample values should approximate the joint probability distribution. These frequencies can be used to approximate the inferred conditional probability for a random variable. If we have N samples, the approximation for the example used above would be

$$P(g_n|i_1) = \frac{P(g_n, i_1)}{P(i_1)} \approxeq \frac{\text{count}(g_n \wedge i_1)}{\text{count}(i_1)}.$$

Markov Chain Monte Carlo (MCMC) Sampling In a Bayesian network with a very large number of nodes, the cost of direct sampling across the highly-dimensional space becomes prohibitive. MCMC algorithms can be used to direct the sampling sequence, which is expressed as a Markov Chain. In this method you start from a randomly-selected point and generate a sample for that point. You then proceed to another point in some randomly-selected direction and generate another sample. An **acceptance function** determines whether to accept or reject the new point. If it is rejected, you return to the previous point and try a different direction. Over a large number of iterations the samples obtained will approximate the actual probability density.



This picture shows three Markov chains running on the 3D Rosenbrock function using the **Metropolis-Hastings** MCMC algorithm. You can see that each sampling sequence has a "burn-in" phase during which it searches for the region of greatest density. After that, the acceptance function guides the navigation sequence to remain close to that region. Source: *Wikipedia*

MCMC sampling can be performed using a number of different algorithms. The **Metropolis-Hastings** algorithm can be used in situations where you can compute the value of a function that is *proportional* to the desired probability density function. A special case of Metroplis-Hastings is **Gibbs sampling**, which takes advantage of the fact that conditional distributions are proportional to the desired joint distribution function, but have a lower cost of sample calculation.

In Part 2 of this series we'll discuss the steps needed to create a Bayesian network model and look at some software libraries that can be used for that purpose.