6.825 Techniques in Artificial Intelligence

Bayesian Networks

Last time, we talked about probability, in general, and conditional probability. This time, I want to give you an introduction to Bayesian networks and then we'll talk about doing inference on them and then we'll talk about learning in them in later lectures.

6.825 Techniques in Artificial Intelligence

Bayesian Networks

- To do probabilistic reasoning, you need to know the joint probability distribution

The idea is that if you have a complicated domain, with many different propositional variables, then to really know everything about what's going on, you need to know the joint probability distribution over all those variables.

6.825 Techniques in Artificial Intelligence

Bayesian Networks

- To do probabilistic reasoning, you need to know the joint probability distribution
- But, in a domain with N propositional variables, one needs 2^N numbers to specify the joint probability distribution

But if you have N binary variables, then there are 2^n possible assignments, and the joint probability distribution requires a number for each one of those possible assignments.

6.825 Techniques in Artificial Intelligence

Bayesian Networks

- To do probabilistic reasoning, you need to know the joint probability distribution
- But, in a domain with N propositional variables, one needs 2^N numbers to specify the joint probability distribution
- We want to exploit independences in the domain

The intuition is that there's almost always some separability between the variables, some independence, so that you don't actually have to know all of those 2^n numbers in order to know what's going on in the world. That's the idea behind Bayesian networks.

6.825 Techniques in Artificial Intelligence

Bayesian Networks

- To do probabilistic reasoning, you need to know the joint probability distribution
- But, in a domain with N propositional variables, one needs 2^N numbers to specify the joint probability distribution
- We want to exploit independences in the domain
- Two components: structure and numerical parameters

Bayesian networks have two components. The first component is called the "causal component." It describes the structure of the domain in terms of dependencies between variables, and then the second part is the actual numbers, the quantitative part. So we'll start looking at the structural part and then we'll look at the quantitative part.

Icy Roads

Let's start by going through a couple of examples. Consider the following case:

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

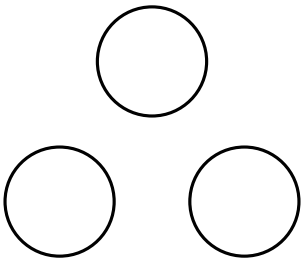
Inspector Smith is sitting in his study, waiting for Holmes and Watson to show up in order to talk to them about something, and it's winter and he's wondering if the roads are icy. He's worried that they might crash. Then his secretary comes in and tells him that Watson has had an accident. He says, "Hmm, Watson had an accident. Gosh, it must be that the roads really are icy. Ha! I bet Holmes is going to have an accident, too. They're never going to get here. I'll go have my lunch."

Then the secretary says to him, "No, no. The roads aren't icy. Look out the window. It's not freezing and they'd put sand on the roads anyway." So he says, "Oh, OK. I guess I better wait for Holmes to show up."

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

"Causal" Component

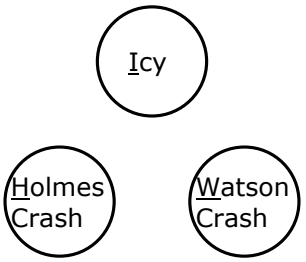


How could we model that using a little Bayesian network? The idea is that we have three propositions.

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

"Causal" Component

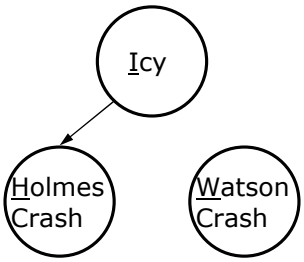


We have Are the roads icy? which we'll represent with the node labeled "Icy." We have "Holmes Crash." And we have "Watson Crash."

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

"Causal" Component

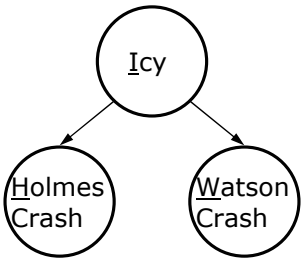


The inspector is thinking about the relationships between these variables, and he's got sort of a causal model of what's going on in the world. He thinks that if it's icy, it's more likely that Holmes is going to crash.

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

"Causal" Component

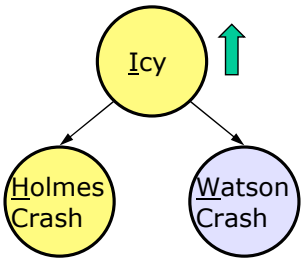


And also, if it's icy, it's more likely that Watson is going to crash.

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

"Causal" Component

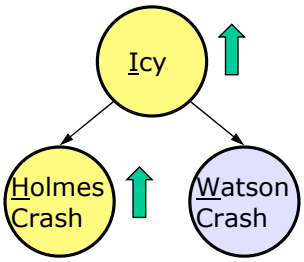


The inspector starts out with some initial beliefs about what's going on. Then, the secretary tells him that Watson crashed. And then he does some reasoning that says, "Well, if Icy is the cause of Watson crashing, and Watson really did crash, then it's more likely that it really is Icy outside."

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

"Causal" Component

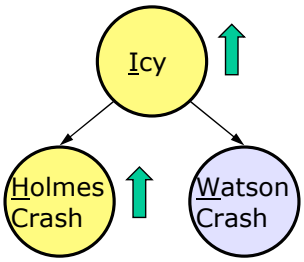


And, therefore, it's also more likely that Holmes will crash, too.

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

"Causal" Component



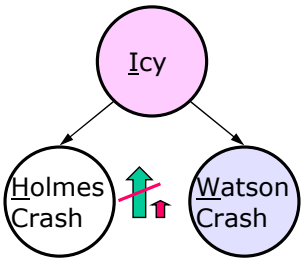
H and W are dependent,

So even though we had this picture with some sort of causal arrows going down from I to H and W, it seems like information can flow back up through the arrows in the other direction.

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

"Causal" Component



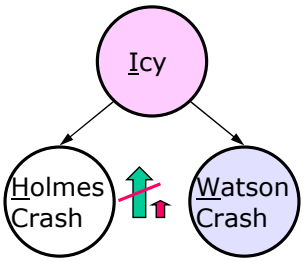
H and W are dependent,

Now, when the secretary says "No, the roads aren't icy," then knowing that Watson crashed, doesn't really have any influence on our belief that Holmes will crash, and our belief that it's true goes back down.

Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."

"Causal" Component



H and W are dependent, but conditionally independent given I

Using the concepts of last lecture, we can say that all of these variables are individually dependent on one another, but that H and W are conditionally independent given I. Once we know the value for I, W doesn't tell us anything about H.

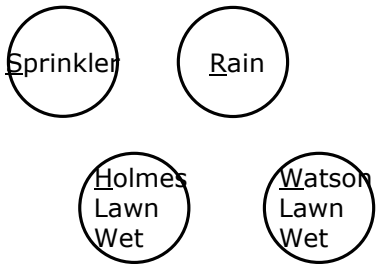
Holmes and Watson in LA

Holmes and Watson have moved to LA. He wakes up to find his lawn wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and he sees it is wet too. So, he concludes it must have rained.

Let's do another one. Holmes has moved to Los Angeles, and his grass is wet, which is a real surprise in L.A. And he wonders if it's because it rained or because he left the sprinkler on. Then he goes and he looks and he sees that his neighbor Watson's grass is also wet. So that makes him think it's been raining, because rain would cause them both to be wet. And it, then, decreases his belief that the sprinkler was on.

Holmes and Watson in LA

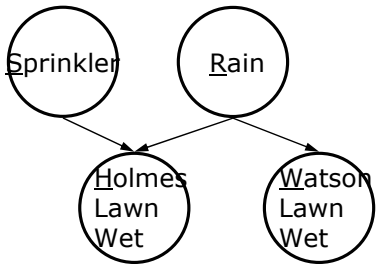
Holmes and Watson have moved to LA. He wakes up to find his lawn wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and he sees it is wet too. So, he concludes it must have rained.



Now we can draw a picture of that. You might have nodes for the 4 propositional random variables for: "Sprinkler on." "Rain". "Holmes' lawn wet." and "Watson lawn wet."

Holmes and Watson in LA

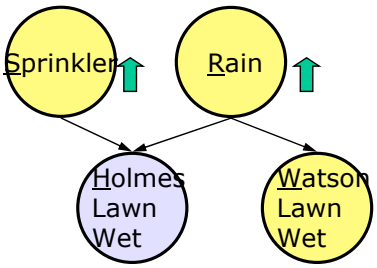
Holmes and Watson have moved to LA. He wakes up to find his lawn wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and he sees it is wet too. So, he concludes it must have rained.



We can connect them together to reflect the causal dependencies in the world. Both sprinkler and rain would cause Holmeses lawn to be wet. Just rain (or maybe some other un-modeled cause) would cause Watson's lawn to be wet.

Holmes and Watson in LA

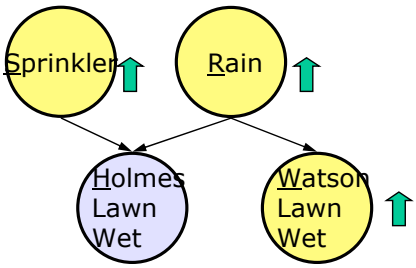
Holmes and Watson have moved to LA. He wakes up to find his lawn wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and he sees it is wet too. So, he concludes it must have rained.



Now, the way that this story goes is, we observe that Holmeses lawn is wet. We come out and we see that the lawn is wet and so from that, we believe that Sprinkler and Rain are both more likely.

Holmes and Watson in LA

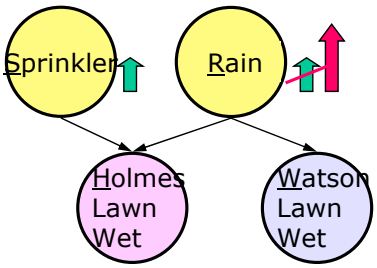
Holmes and Watson have moved to LA. He wakes up to find his lawn wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and he sees it is wet too. So, he concludes it must have rained.



Now, it also is true that without observing Watson's lawn, if Holmes sees that his own lawn is wet, he's going to believe that it's more likely that Watson's lawn is wet, too. That's because the cause, Rain, becomes more likely and, therefore, this symptom, Watson's lawn being wet, becomes more likely too.

Holmes and Watson in LA

Holmes and Watson have moved to LA. He wakes up to find his lawn wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and he sees it is wet too. So, he concludes it must have rained.

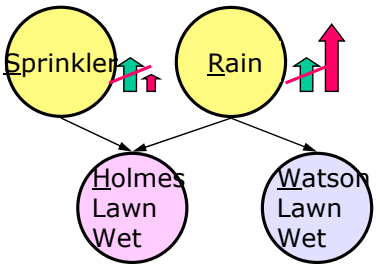


Given W, P(R) goes up

Now when he goes and observes Watson's lawn and sees that it is wet also, the probability of rain goes way up because, notice, there are two pieces of corroborating evidence.

Holmes and Watson in LA

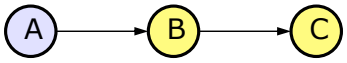
Holmes and Watson have moved to LA. He wakes up to find his lawn wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and he sees it is wet too. So, he concludes it must have rained.



Given W, P(R) goes up and P(S) goes down – "explaining away"

And, interestingly enough, the probability that the sprinkler was on comes down. This is a phenomenon called "explaining away." Later on we'll do this example with the numbers, so you can see how it comes out mathematically. But it's sort of intuitive because you have two potential causes, each of which becomes more likely when you see the symptom; but once you pick a cause, then the other cause's probability goes back down.

Forward Serial Connection



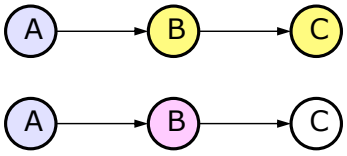
- Transmit evidence from A to C through unless B is instantiated (its truth value is known)

- Knowing about A will tell us something about C

Now we'll go through all the ways three nodes can be connected and see how evidence can be transmitted through them. First, let's think about a serial connection, in which a points to b which points to c. And let's assume that b is uninstantiated. Then, when we get evidence about A (either by instantiating it, or having it flow to A from some other node), it flows through B to C.

(In this and other slides in this lecture, I'll color a node light blue/gray if it's the one where evidence is arriving, and color other nodes yellow to show where that evidence propagates. As we'll see in a minute, I'll use pink for a node that is instantiated, or that has a child that is instantiated.)

Forward Serial Connection

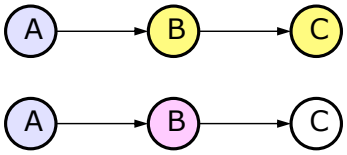


- Transmit evidence from A to C through unless B is instantiated (its truth value is known)

- Knowing about A will tell us something about C
- But, if we know B, then knowing about A will not tell us anything new about C.

If B is instantiated, then evidence does not propagate through from A to C.

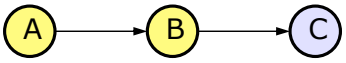
Forward Serial Connection



- Transmit evidence from A to C through unless B is instantiated (its truth value is known)
  - A = battery dead
  - B = car won't start
  - C = car won't move
- Knowing about A will tell us something about C
- But, if we know B, then knowing about A will not tell us anything new about C.

So what's an example of this? Well, the battery's dead, so the car won't start, so the car won't move. So finding out that the battery's dead gives you information about whether the car will move or not. But if you know the car won't start, then knowing about the battery doesn't give you any information about whether it will move or not.

Backward Serial Connection

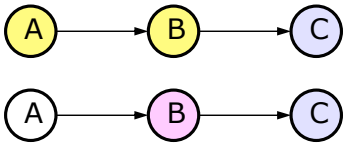


- Transmit evidence from C to A through unless B is instantiated (its truth value is known)

- Knowing about C will tell us something about A

What if we have the same set of connections, but our evidence arrives at node C rather than node A? The evidence propagates backward up serial links as long as the intermediate node is not instantiated.

Backward Serial Connection

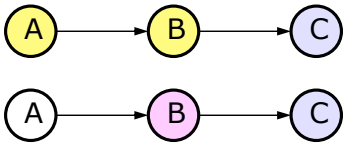


- Transmit evidence from C to A through unless B is instantiated (its truth value is known)

- Knowing about C will tell us something about A
- But, if we know B, then knowing about C will not tell us anything new about A

If the intermediate node is instantiated, then evidence does not propagate.

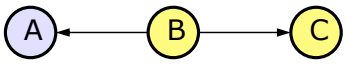
Backward Serial Connection



- Transmit evidence from C to A through unless B is instantiated (its truth value is known)
  - A = battery dead
  - B = car won't start
  - C = car won't move
- Knowing about C will tell us something about A
- But, if we know B, then knowing about C will not tell us anything new about A

So, finding out that the car won't move tells you something about whether it will start, which tells you something about the battery. But if you know the car won't start, then finding out that it won't move, doesn't give you any information about the battery.

Diverging Connection

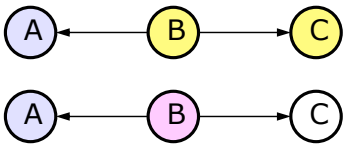


- Transmit evidence through B unless it is instantiated

- Knowing about A will tell us something about C
- Knowing about C will tell us something about A

In a diverging connection, there are arrows going from B to A and from B to C. Now, the question is, if we get evidence at A, what other nodes does it effect? If B isn't instantiated, it propagates through to C.

Diverging Connection

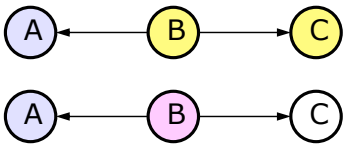


- Transmit evidence through B unless it is instantiated

- Knowing about A will tell us something about C
- Knowing about C will tell us something about A
- But, if we know B, then knowing about A will not tell us anything new about C, or vice versa

But if B is instantiated, as before, then the propagation is blocked.

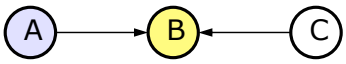
Diverging Connection



- Transmit evidence through B unless it is instantiated
  - A = Watson crash
  - B = Icy
  - C = Holmes crash
- Knowing about A will tell us something about C
- Knowing about C will tell us something about A
- But, if we know B, then knowing about A will not tell us anything new about C, or vice versa

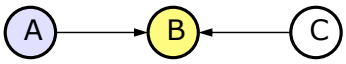
This is exactly the form of the icy roads example. Before we know whether the roads are icy, knowing that Watson crashed tells us something about whether Holmes will crash. But once we know the value of Icy, the information doesn't propagate.

Converging Connection



The tricky case is when we have a converging connection: A points to B and C points to B.

Converging Connection

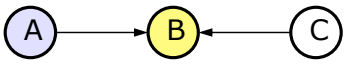


- Transmit evidence from A to C only if B or a descendant of B is instantiated

- Without knowing B, finding A does not tell us anything about B

Let's first think about the case when neither B nor any of its descendants is instantiated. In that case, evidence does not propagate from A to C.

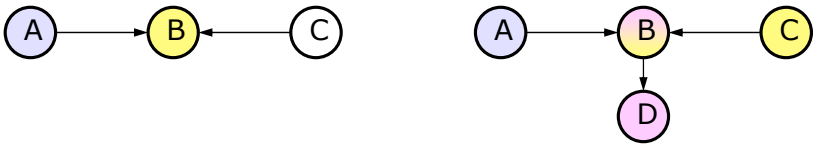
Converging Connection



- Transmit evidence from A to C only if B or a descendant of B is instantiated
  - A = Bacterial infection
  - B = Sore throat
  - C = Viral Infection
- Without knowing B, finding A does not tell us anything about B

This network structure arises when, for example, you have one symptom, say "sore throat", which could have multiple causes, for example, a bacterial infection or a viral infection. If you find that someone has a bacterial infection, it gives you information about whether they have a sore throat, but it doesn't affect the probability that they have a viral infection also.

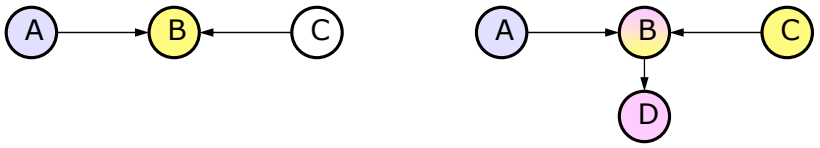
Converging Connection



- Transmit evidence from A to C only if B or a descendant of B is instantiated
  - A = Bacterial infection
  - B = Sore throat
  - C = Viral Infection
- Without knowing B, finding A does not tell us anything about B
- If we see evidence for B, then A and C become dependent (potential for "explaining away").

But when either node B is instantiated, or one of its descendants is, then we know something about whether B is true. And in that case, information **does** propagate through from A to C. I colored node B partly pink here to indicate that, although it's not instantiated, one of its descendants is.

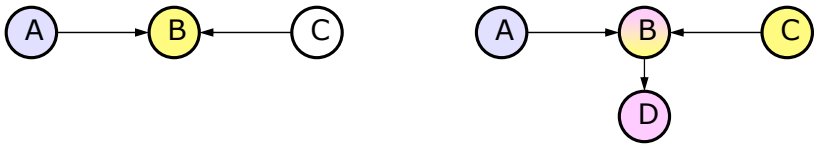
Converging Connection



- Transmit evidence from A to C only if B or a descendant of B is instantiated
  - A = Bacterial infection
  - B = Sore throat
  - C = Viral Infection
- Without knowing B, finding A does not tell us anything about B
- If we see evidence for B, then A and C become dependent (potential for "explaining away"). If we find bacteria in patient with a sore throat, then viral infection is less likely.

So, if we know that you have a sore throat, then finding out that you have a bacterial infection causes us to think it's less likely that you have a viral infection. This is the same sort of reasoning as we had in the wet lawn example, as well. We had a converging connection from Rain to Holmeses lawn to Sprinkler. We saw that Holmes's lawn was wet. So, then, when we saw that Watson's lawn was wet, the information propagated through the diverging connection through Rain to Holmes, and then through the converging connection through Holmes to Sprinkler. If we hadn't had evidence about Holmeses lawn, then seeing that Watson's lawn was wet wouldn't have affected our belief in the sprinkler's having been on.

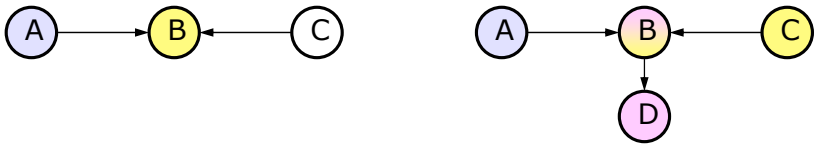
Converging Connection



- Transmit evidence from A to C only if B or a descendant of B is instantiated
  - A = Bacterial infection
  - B = Sore throat
  - C = Viral Infection
- Without knowing B, finding A does not tell us anything about B
- If we see evidence for B, then A and C become dependent (potential for "explaining away"). If we find bacteria in patient with a sore throat, then viral infection is less likely.

So, serial connections and diverging connections are essentially the same, in terms of evidence propagation, and you can generally turn the arrows around in a Bayesian network, as long as you never create or destroy any converging connections.

Converging Connection



- Transmit evidence from A to C only if B or a descendant of B is instantiated
  - A = Bacterial infection
  - B = Sore throat
  - C = Viral Infection
- Without knowing B, finding A does not tell us anything about B
- If we see evidence for B, then A and C become dependent (potential for "explaining away"). If we find bacteria in patient with a sore throat, then viral infection is less likely.

I've been using this language of causes and effects, of causes and symptoms. It gives us an interpretation of these arrows that makes them more intuitive for humans to specify. You can use these nodes and arrows to specify relationships that aren't causal but, a very intuitive way to use this notation is to try to make the arrows kind of correspond to causation.

D-separation

Now we're ready to define, based on these three cases, a general notion of how information can propagate or be blocked from propagation through a network. If two variables are d-separated, then changing the uncertainty on one does not change the uncertainty on the other.

D-separation

- Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either
  - The connection is serial or diverging and V is known
  - The connection is converging and neither V nor any descendant is instantiated

Two variables a and b are -"d-separated" if and only if for every path between them, there is an intermediate variable V such that either: the connection is (serial or diverging) and v is known; or the connection is converging and neither v nor any descendant has evidence.

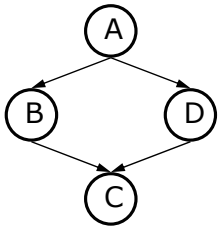
D-separation

- Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either
  - The connection is serial or diverging and V is known
  - The connection is converging and neither V nor any descendant is instantiated
  - Two variables are d-connected iff they are not d-separated

The opposite of "d-separated," not d-separated, we'll call "d-connected."

D-separation

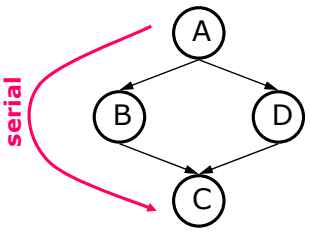
- Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either
  - The connection is serial or diverging and V is known
  - The connection is converging and neither V nor any descendant is instantiated
  - Two variables are d-connected iff they are not d-separated



We'll spend some time understanding the d-separation relations in this network.

D-separation

- Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either
  - The connection is serial or diverging and V is known
  - The connection is converging and neither V nor any descendant is instantiated
  - Two variables are d-connected iff they are not d-separated

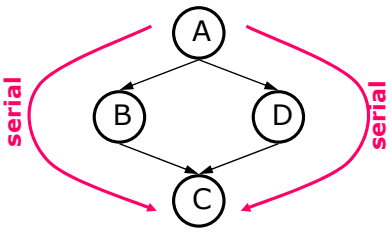


- A-B-C: serial, blocked when B is known, connected otherwise

The connection ABC is serial. It's blocked when B is known and connected otherwise. When it's connected, information can flow from A to C or from C to A.

D-separation

- Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either
  - The connection is serial or diverging and V is known
  - The connection is converging and neither V nor any descendant is instantiated
  - Two variables are d-connected iff they are not d-separated

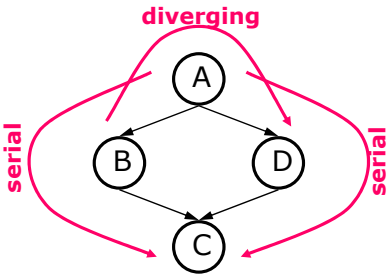


- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise

The connection ADC is the same.

D-separation

- Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either
  - The connection is serial or diverging and V is known
  - The connection is converging and neither V nor any descendant is instantiated
  - Two variables are d-connected iff they are not d-separated

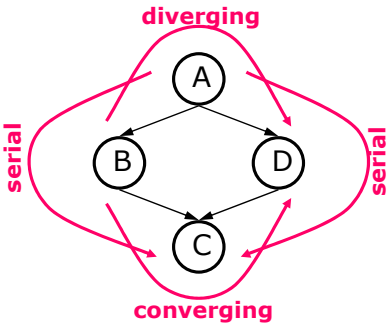


- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise

The connection BAD is diverging. It's blocked when A is known and connected otherwise.

D-separation

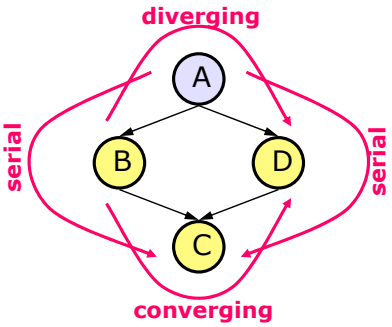
- Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either
  - The connection is serial or diverging and V is known
  - The connection is converging and neither V nor any descendant is instantiated
  - Two variables are d-connected iff they are not d-separated



- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise
- B-C-D: converging, blocked when C has no evidence, connected otherwise

The connection BCD is converging. Remember, this is the case that's different. It's blocked when C has no evidence, but connected otherwise.

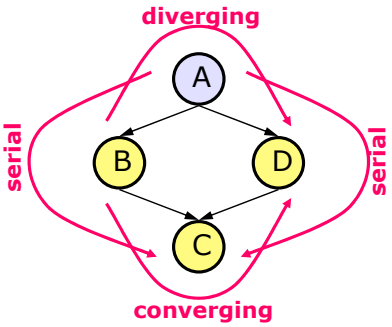
D-Separation Detail



- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise
- B-C-D: converging, blocked when C has no evidence, connected otherwise

The next few slides have a lot of detailed examples of d-separation. If you understand the concept well, you can just skip over them. Remember that we're using pink to mean a node is instantiated. The node where information is entering is colored blue, and the nodes to which it propagates are colored yellow.

D-Separation Detail

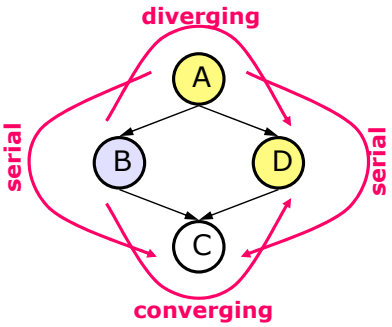


- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise
- B-C-D: converging, blocked when C has no evidence, connected otherwise

- No instantiation
  - A, C are d-connected (A-B-C connected, A-D-C connected)

What happens when none of the nodes are instantiated? Information at A flows through B to C and through D to C.

D-Separation Detail

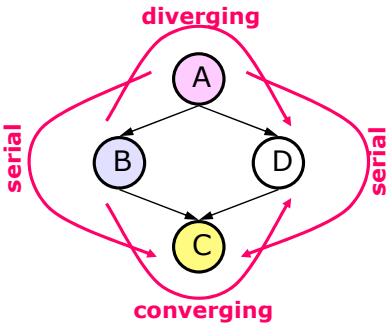


- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise
- B-C-D: converging, blocked when C has no evidence, connected otherwise

- No instantiation
  - A, C are d-connected (A-B-C connected, A-D-C connected)
  - B, D are d-connected (B-A-D connected, B-C-D blocked)

Information at B flows through A to D, but it does not flow through C to D (though it does give us information about C).

D-Separation Detail

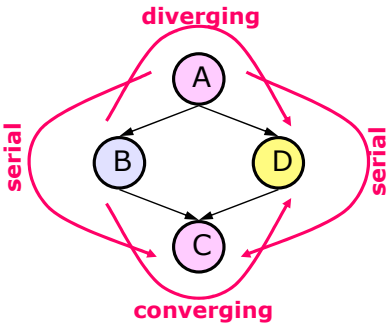


- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise
- B-C-D: converging, blocked when C has no evidence, connected otherwise

- No instantiation
  - A, C are d-connected (A-B-C connected, A-D-C connected)
  - B, D are d-connected (B-A-D connected, B-C-D blocked)
- A instantiated
  - B, D are d-separated (B-A-D blocked, B-C-D blocked)

Now, let's think about what happens when A is instantiated. B and D are d-separated. Information at B doesn't flow through A, because it's instantiated and it's a diverging connection. And it doesn't flow through C because it's not instantiated and it's a converging connection. So information about B gives us information about C, but it doesn't flow through to D.

D-Separation Detail

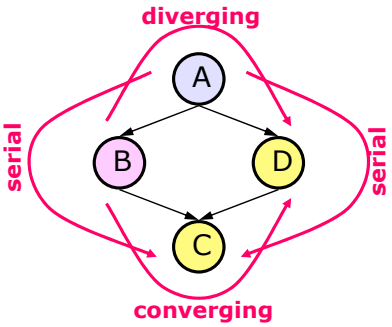


- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise
- B-C-D: converging, blocked when C has no evidence, connected otherwise

- No instantiation
  - A, C are d-connected (A-B-C connected, A-D-C connected)
  - B, D are d-connected (B-A-D connected, B-C-D blocked)
- A instantiated
  - B, D are d-separated (B-A-D blocked, B-C-D blocked)
- A and C instantiated
  - B, D are d-connected (B-A-D blocked, B-C-D connected)

When both A and C are instantiated, B and D become d-connected, because now information can flow through C.

D-Separation Detail

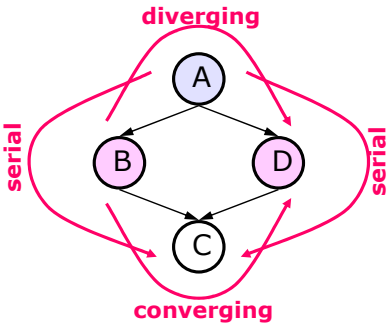


- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise
- B-C-D: converging, blocked when C has no evidence, connected otherwise

- No instantiation
  - A, C are d-connected (A-B-C connected, A-D-C connected)
  - B, D are d-connected (B-A-D connected, B-C-D blocked)
- A instantiated
  - B, D are d-separated (B-A-D blocked, B-C-D blocked)
- A and C instantiated
  - B, D are d-connected (B-A-D blocked, B-C-D connected)
- B instantiated
  - A, C are d-connected (A-B-C blocked, A-D-C connected)

When just B is instantiated, then evidence at A flows through D to C (and, of course, backwards from C through D to A).

D-Separation Detail



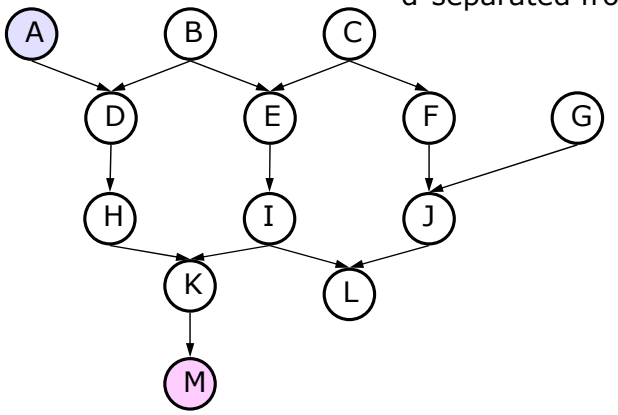
- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise
- B-C-D: converging, blocked when C has no evidence, connected otherwise

- No instantiation
  - A, C are d-connected (A-B-C connected, A-D-C connected)
  - B, D are d-connected (B-A-D connected, B-C-D blocked)
- A instantiated
  - B, D are d-separated (B-A-D blocked, B-C-D blocked)
- A and C instantiated
  - B, D are d-connected (B-A-D blocked, B-C-D connected)
- B instantiated
  - A, C are d-connected (A-B-C blocked, A-D-C connected)
- B and D instantiated
  - A, C are d-separated (A-B-C blocked, A-D-C blocked)

Finally, if B and D are both instantiated, then A and C are d-separated. There are two paths from A to C, but they're both blocked.

D-Separation Example

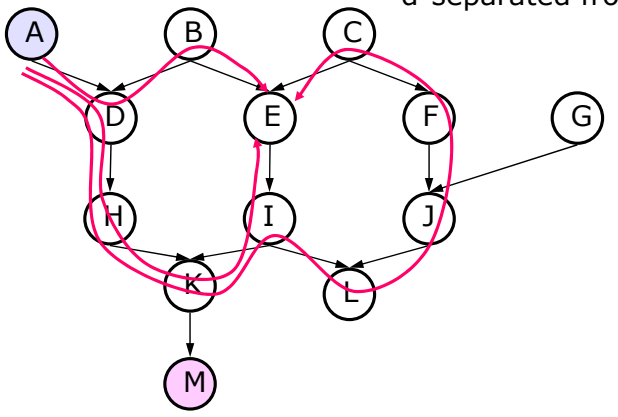
Given M is known, is A d-separated from E?



Okay. Here's a bigger Bayesian network. Let's consider a case in which M is known. Is A d-separated from E?

D-Separation Example

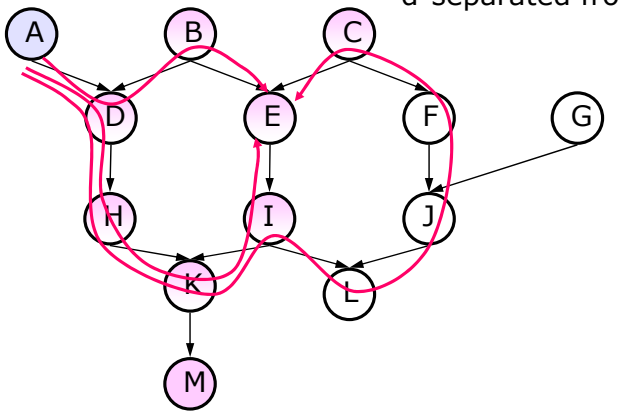
Given M is known, is A d-separated from E?



Remember, they are d-separated if all the paths between A and E are blocked. So, let's start by finding all the paths from A to E. There are three possible paths.

D-Separation Example

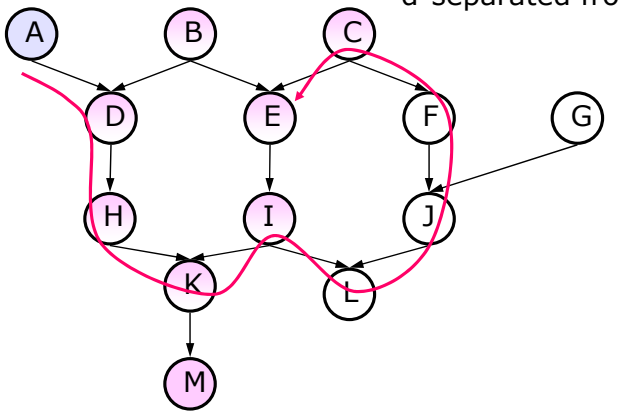
Given M is known, is A d-separated from E?



Now, let's remind ourselves which nodes have a descendent that's instantiated, by coloring them half pink.

D-Separation Example

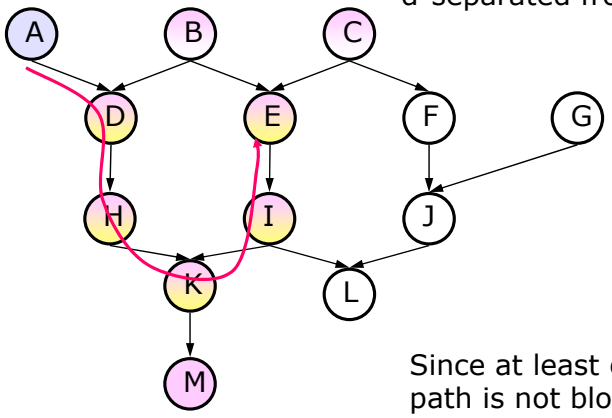
Given M is known, is A d-separated from E?



So, what about the path ADHKILJFCE. Is it blocked? Yes, because ILJ is a converging connection and L is has no evidence.

D-Separation Example

Given M is known, is A d-separated from E?

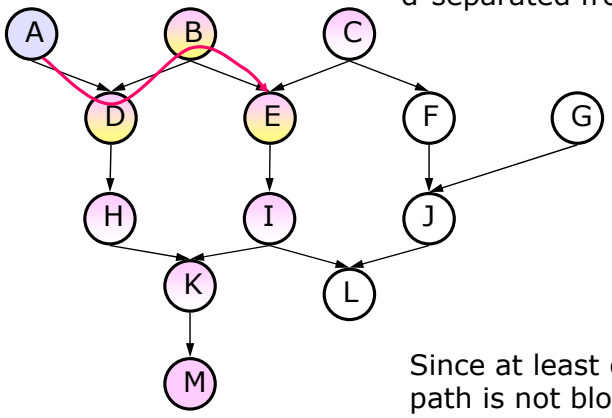


Since at least one path is not blocked, A is not d-separated from E

Okay. Now, what about path ADHKIE? Is it blocked? No. ADHK is all serial connections, and although those nodes have received some evidence from M, they are not instantiated, so information propagates. HKI is a converging connection, but because K has evidence, then information propagates to I. Finally, KIE is a (backward) serial connection, so information propagates all the way around. Once we find one non-blocked path from A to E, then we know they're d-connected.

D-Separation Example

Given M is known, is A d-separated from E?



Since at least one path is not blocked, A is not d-separated from E

For completeness' sake, let's look at the last path, ADBE. It is also not blocked. ADB is a converging connection, but because there's evidence at D, information goes through. And it goes through the diverging connection DBE because, although there's evidence at B, it's not instantiated.

Recitation Problems

Use the Bayesian network from the previous slides to answer the following questions:

- Are A and F d-separated if M is instantiated?
- Are A and F d-separated if nothing is instantiated?
- Are A and E d-separated if I is instantiated?
- Are A and E d-separated if B and H are instantiated?
- Describe a situation in which A and G are d-separated.
- Describe a situation in which A and G are d-connected.

Here are some practice problems on d-separation in the network from the previous slides.

Bayesian (Belief) Networks

Now we can describe a formal class of objects, called Bayesian Networks. They're also sometimes called belief networks or Bayesian belief networks. A Bayes net is made up of three components.

Bayesian (Belief) Networks

- Set of variables, each has a finite set of values

There's a finite set of variables, each of which has a finite domain (actually, it's possible to have continuous-valued variables, but we're not going to consider that case).

Bayesian (Belief) Networks

- Set of variables, each has a finite set of values
- Set of directed arcs between them forming acyclic graph

There's a set of directed arcs between the nodes, forming an acyclic graph.

Bayesian (Belief) Networks

- Set of variables, each has a finite set of values
- Set of directed arcs between them forming acyclic graph
- Every node A, with parents B_1, …, B_n, has P(A | B_1,…,B_n) specified

And every node A, with parents B1 through Bn has a conditional probability distribution, P(A | B1…Bn) specified (typically in a table indexed by values of the B variables, but sometimes stored in a more compact form, such as a tree).

Bayesian (Belief) Networks

- Set of variables, each has a finite set of values
- Set of directed arcs between them forming acyclic graph
- Every node A, with parents B_1, …, B_n, has P(A | B_1,…,B_n) specified

Theorem: If A and B are d-separated given evidence e, then P(A | e) = P(A | B, e)

The crucial theorem about Bayesian networks is that if A and B are d-separated given some evidence e, then A and B are conditionally independent given e; that is, then P(A | B, e)= P(A|e). We'll be able to exploit these conditional independence relationships to make inference efficient.

Chain Rule

Here's another important theorem, the chain rule of probabilities.

Chain Rule

- Variables: V_1, …, V_n

Let's say we have a whole bunch of variables, v1 through vn. I use big letters to stand for variables and little letters to stand for their values. I'm first going to write this out in complete detail and then I'll show you the shorthand I typically use.

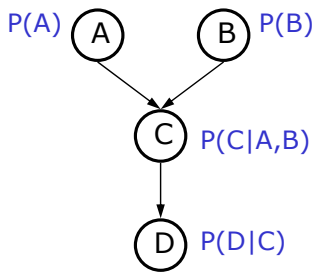
Chain Rule

- Variables: $V_1, \ldots, V_n$
- Values: $v_1, \ldots, v_n$
- $P(V_1=v_1, V_2=v_2, \ldots, V_n=v_n) = \prod_i P(V_i=v_i \mid \text{parents}(V_i))$

Let's assume that our V's are Boolean variables, OK? The probability of V1 = v1 and V2 = v2 and, etc., and Vn = vn, is equal to the product over all these variables, of the probability of Vi= vi given the values of the parents of vi. OK. So this is actually pretty cool and pretty important We're saying that the joint probability distribution is the product of all the individual probability distributions that are stored in the nodes of the graph. The parents of vi are just the nodes that have arcs into vi. This gives us a way to compute the probability of any possible assignment of values to variables; it lets us compute the value in any cell of the huge joint probability distribution.

Chain Rule

- Variables: $V_1, \ldots, V_n$
- Values: $v_1, \ldots, v_n$
- $P(V_1=v_1, V_2=v_2, \ldots, V_n=v_n) = \prod_i P(V_i=v_i \mid \text{parents}(V_i))$

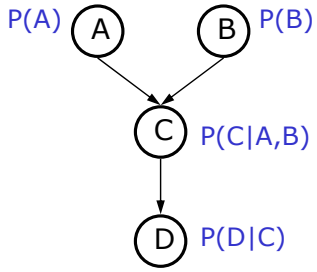


Let's illustrate this using an example. Here's a graph. We'll have local probability tables that have P(A), P(B), P(C|A,B), and P(D|C).

Chain Rule

- Variables: $V_1, \ldots, V_n$
- Values: $v_1, \ldots, v_n$
- $P(V_1=v_1, V_2=v_2, \ldots, V_n=v_n) = \prod_i P(V_i=v_i \mid \text{parents}(V_i))$

P(ABCD) = P(A=true, B=true,
C=true, D=true)



Now, we'd like to compute the probability that A,B,C,and D are all true. I'll write it using the shorthand P(ABCD).

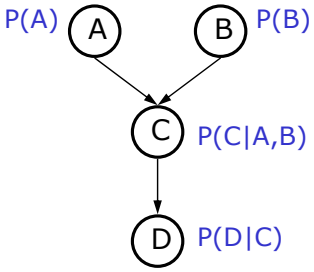
Chain Rule

- Variables: V_1, …, V_n
- Values: v_1, …, v_n
- $P(V_1=v_1, V_2=v_2, \ldots, V_n=v_n) = \prod_i P(V_i=v_i \mid \text{parents}(V_i))$

P(ABCD) = P(A=true, B=true, C=true, D=true)

P(ABCD) =

P(D|ABC)P(ABC)



We can use conditioning to write that as the product of P(D|ABC) and P(ABC).

Chain Rule

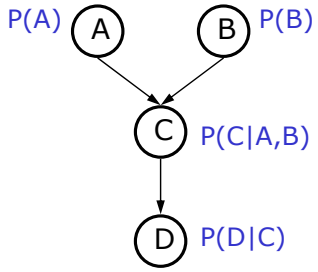
- Variables: $V_1, \ldots, V_n$
- Values: $v_1, \ldots, v_n$
- $P(V_1=v_1, V_2=v_2, \ldots, V_n=v_n) = \prod_i P(V_i=v_i \mid \text{parents}(V_i))$

P(ABCD) = P(A=true, B=true, C=true, D=true)

P(ABCD) =

P(D|ABC)P(ABC) =

P(D|C) P(ABC) =



A d-separated from D given C

B d-separated from D given C

Now, we can simplify P(D|ABC) to P(D|C), because, given C, D is d-separated from A and B. And we have P(D|C) stored directly in a local probability table, so we're done with this term.

Chain Rule

- Variables: V_1, …, V_n
- Values: v_1, …, v_n
- $P(V_1=v_1, V_2=v_2, \ldots, V_n=v_n) = \prod_i P(V_i=v_i \mid \text{parents}(V_i))$

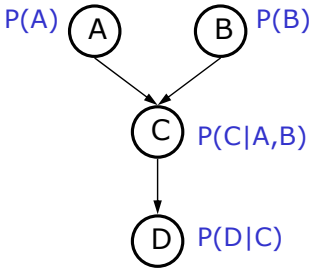
P(ABCD) = P(A=true, B=true, C=true, D=true)

P(ABCD) =

P(D|ABC)P(ABC) =

P(D|C) P(ABC) =

P(D|C) P(C|AB) P(AB) =



A d-separated from D given C

B d-separated from D given C

Now, we can use conditioning to write P(ABC) as P(C|AB) times P(AB). We have P(C|AB) in our table, so that's done.

Chain Rule

- Variables: V_1, …, V_n
- Values: v_1, …, v_n
- $P(V_1=v_1, V_2=v_2, \ldots, V_n=v_n) = \prod_i P(V_i=v_i \mid \text{parents}(V_i))$

P(ABCD) = P(A=true, B=true, C=true, D=true)

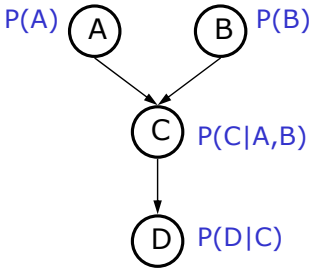
P(ABCD) =

P(D|ABC)P(ABC) =

P(D|C) P(ABC) =

P(D|C) P(C|AB) P(AB) =

P(D|C) P(C|AB) P(A)P(B)



A d-separated from D given C

B d-separated from D given C

A d-separated from B

All that's left to deal with is P(AB). We can write this as P(A) times P(B) because A and B are independent (they are d-separated given nothing).

Chain Rule

- Variables: $V_1, \ldots, V_n$
- Values: $v_1, \ldots, v_n$
- $P(V_1=v_1, V_2=v_2, \ldots, V_n=v_n) = \prod_i P(V_i=v_i \mid \text{parents}(V_i))$

P(ABCD) = P(A=true, B=true, C=true, D=true)

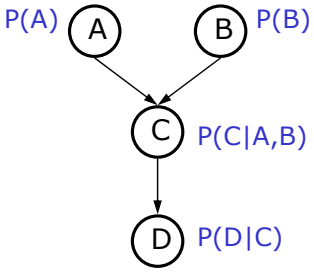
P(ABCD) =

P(D|ABC)P(ABC) =

P(D|C) P(ABC) =

P(D|C) P(C|AB) P(AB) =

P(D|C) P(C|AB) P(A)P(B)



A d-separated from D given C

B d-separated from D given C

A d-separated from B

So, this element of the joint distribution is a product of terms, one for each node, expressing the probability it takes on that value given the values of the parents.

Key Advantage

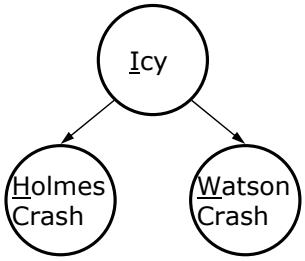
- The conditional independencies (missing arrows) mean that we can store and compute the joint probability distribution more efficiently

For each variable, we just have to condition on its parents. And we multiply those together and we get the joint. So what that means is that if you have any independencies -- if you have anything other than all the arrows in your graph, in some sense, then you have to do less work to compute the joint distribution. You have to store fewer number in your table, you have to do less work. Now, it's true that there are some probability distributions for which you have to have all the arrows in there, there's no other way to represent them. But there are plenty of other ones that do have some amount of d-separation and therefore give us some efficiency in calculation.

Icy Roads with Numbers

Let’s finish by doing the numerical calculations that go with the examples.

Icy Roads with Numbers

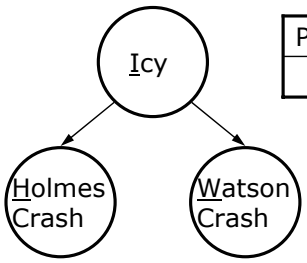


Here's the Bayesian Network for the icy roads problem.

Icy Roads with Numbers

t= true

f= false



P(I=t)	P(I=f)
0.7	0.3

The right-hand column in these tables is redundant, since we know the entries in each row must add to 1.

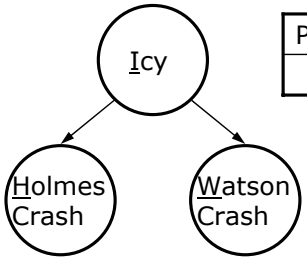
NB: the columns need NOT add to 1.

The node Icy has no parents, so it's conditional probability table is really just the probability that Icy is true. Let's say it's 0.7.

Icy Roads with Numbers

t= true

f= false



P(I=t)	P(I=f)
0.7	0.3

	P(W=t | I)	P(W=f | I)
I=t	0.8	0.2
I=f	0.1	0.9

The right-hand column in these tables is redundant, since we know the entries in each row must add to 1.

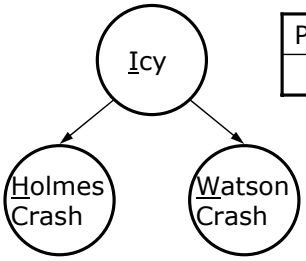
NB: the columns need NOT add to 1.

Now, the variable W depends on I, so it's going to have to have a more complicated table. For each possible value of I, we're going to have to give the probability of W given I. Note that the rows add up to 1, because, for a given value for I, they specify the probabilities of W being true and false. But there's no need for the columns to add up (or have any other relationship to one another). When I is true, W has one distribution. When I is false, it can have a completely different distribution. (If I and W are independent, then both rows of the table will be the same). Watson is really a terrible driver!

Icy Roads with Numbers

t= true

f= false



P(I=t)	P(I=f)
0.7	0.3

	P(H=t | I)	P(H=f | I)
I=t	0.8	0.2
I=f	0.1	0.9

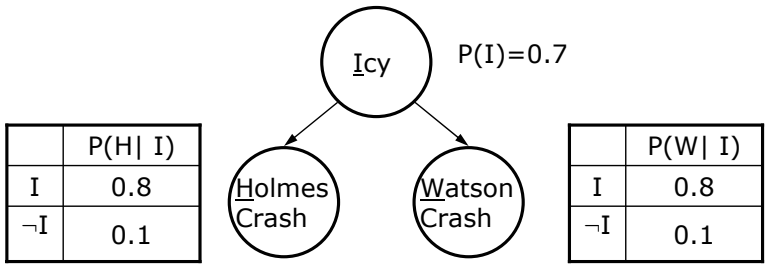
	P(W=t | I)	P(W=f | I)
I=t	0.8	0.2
I=f	0.1	0.9

The right-hand column in these tables is redundant, since we know the entries in each row must add to 1.

NB: the columns need NOT add to 1.

We're going to assume that Holmes and Watson are indistinguishable in their bad driving skills. And so we'll use the same probabilities for the H node (though it's in no way necessary).

Numerical Example: Shorthand

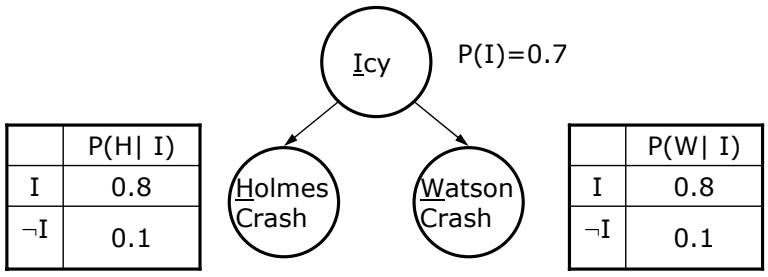


	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

Here's a more compact way of saying what we wrote out in complete detail on the previous slide. We don't need to provide P(not icy), for example, because we know it's 1 – P(icy).

Probability that Watson Crashes



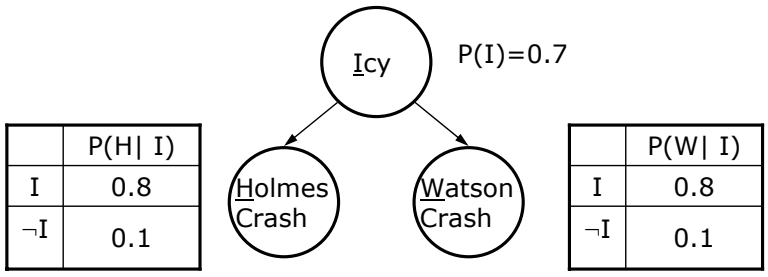
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

P(W) =

Okay. Now let's compute the probability that Watson crashes. This is before we have any evidence at all; this is our **prior** probability of Watson crashing.

Probability that Watson Crashes



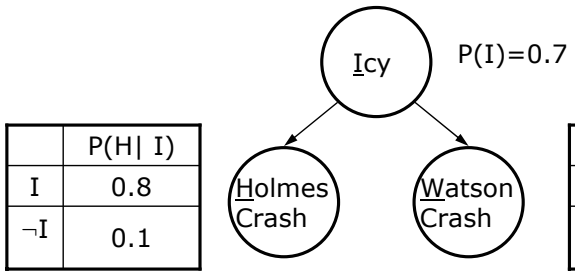
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

$$P(W) = P(W|\ I)\ P(I) + P(W|\ \neg I)\ P(\neg I)$$

We don't know anything directly about P(W), but we do know P(W|I). So, let's use conditioning to write P(W) as P(W|I) P(I) + P(W|not I) P(not I).

Probability that Watson Crashes



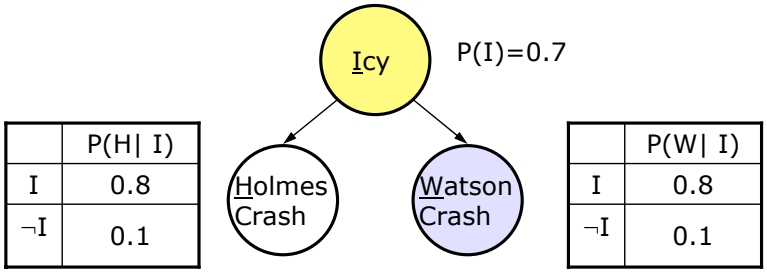
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

$$
\begin{aligned}
P(W) &= P(W|\,I)\,P(I) + P(W|\,\neg I)\,P(\neg I) \\
&= 0.8 \cdot 0.7 + 0.1 \cdot 0.3 \\
&= 0.56 + 0.03 \\
&= 0.59
\end{aligned}
$$

Now, we have all of those quantities directly in our tables and we can do the arithmetic to get 0.59. So, Watson is such a bad driver, that, even without any evidence, we think there's almost a .6 chance that he's going to crash.

Probability of Icy given Watson



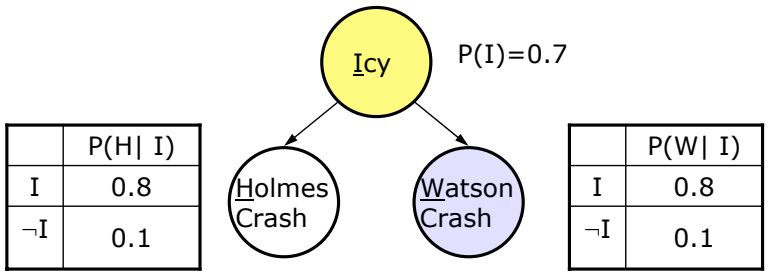
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

P(I | W) =

Now we find out that W is true. Watson has crashed. So let's figure out what that tells us about whether it's icy, and what that tells us about Holmes's situation. So we want to know what's the probability that it's icy, given that Watson crashed.

Probability of Icy given Watson



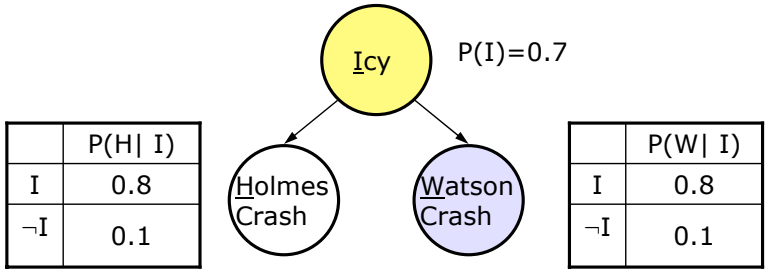
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

P(I | W) =

Well, our arrows don't go in that direction, and so we don't have tables that tell us the probabilities in that direction. So what do you do when you have conditional probability that doesn't go in the direction you want to be going? Bayes' Rule.

Probability of Icy given Watson



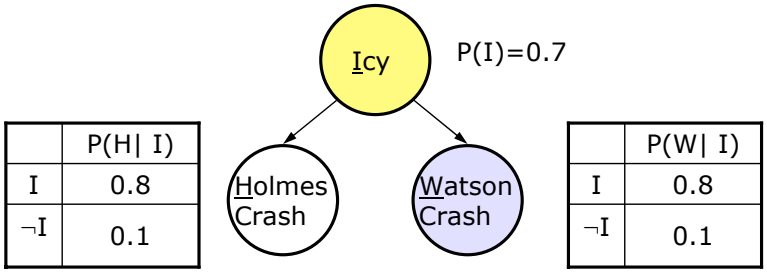
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

$P(I \mid W) = P(W \mid I)\, P(I) / P(W)$

So, P(I | W) is P(W | I) P(I) / P(W). Now we're in good shape, because P(W|I) is in our table, as is P(I). And we just computed P(W).

Probability of Icy given Watson



	P(H| I)
I	0.8
¬I	0.1

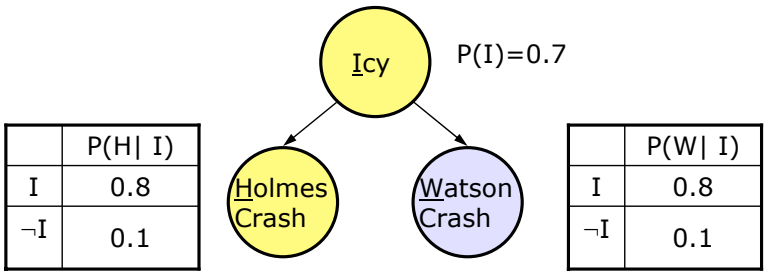
	P(W| I)
I	0.8
¬I	0.1

$$P(I \mid W) = P(W \mid I)\, P(I) / P(W)$$
$$= 0.8 \cdot 0.7 / 0.59$$
$$= 0.95$$

We started with P(I) = 0.7; knowing that Watson crashed raised the probability to 0.95

Now we can do the arithmetic to get 0.95. So, we started out thinking that it was icy with probability 0.7, but now that we know that Watson crashed, we think it's 0.95 likely that it's icy.

Probability of Holmes given Watson



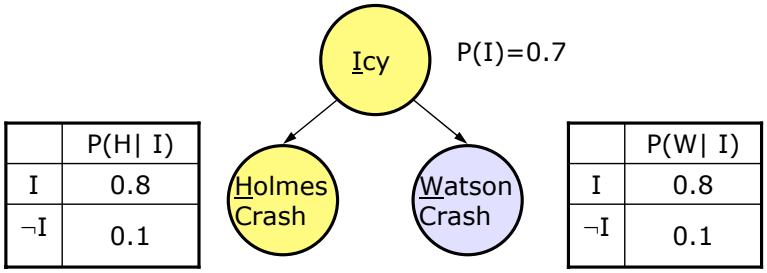
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

P(H|W) =

Now let's see what we think about Holmes, given only that we know that Watson crashed. We need P(H | W), but Bayes' rule won't help us directly here. We're going to have to do conditioning again, summing over all possible values of I.

Probability of Holmes given Watson



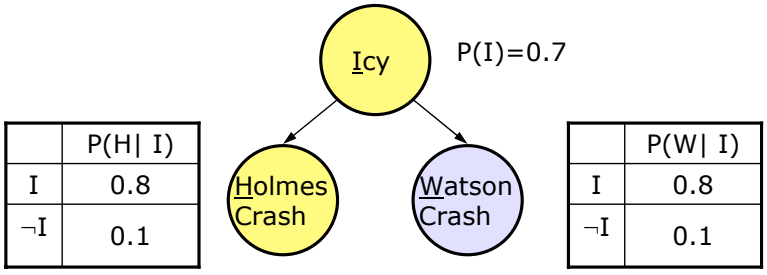
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

$$P(H|W) = P(H|W,I)P(I|W) + P(H|W,\neg I)\,P(\neg I|\,W)$$

So, we have P(H|W,I) times P(I|W) + P(H|W,~I) times P(~I | W). This is a version of conditioning that applies when you already have one variable on the right side of the bar. You can verify it as an exercise.

Probability of Holmes given Watson



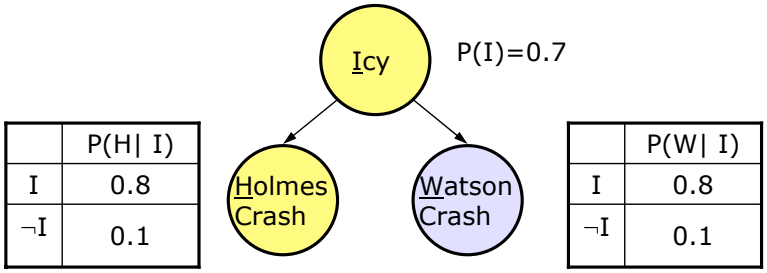
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

$$P(H|W) = P(H|W,I)P(I|W) + P(H|W,\neg I)\,P(\neg I|\,W)$$
$$= P(H|I)P(I|W) + P(H|\neg I)\,P(\neg I|\,W)$$

Now, because H is conditionally independent of W given I (which is true because H is d-separated from W given I), we can simplify P(H|W,I) to P(H|I). And the same for P(H|~I).

Probability of Holmes given Watson



	P(H| I)
I	0.8
¬I	0.1

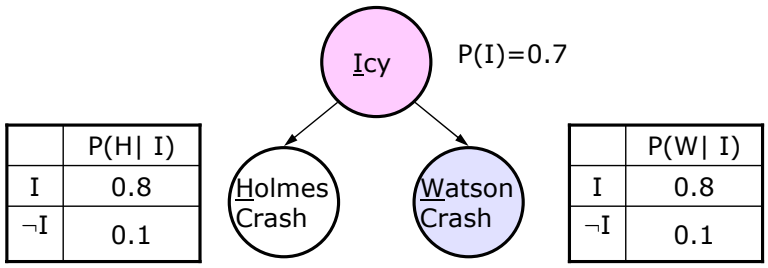
	P(W| I)
I	0.8
¬I	0.1

$$P(H|W) = P(H|W,I)P(I|W) + P(H|W,\neg I)\,P(\neg I|\,W)$$
$$= P(H|I)P(I|W) + P(H|\neg I)\,P(\neg I|\,W)$$
$$= 0.8 \cdot 0.95 + 0.1 \cdot 0.05$$
$$= 0.765$$

We started with P(H) = 0.59; knowing that Watson crashed raised the probability to 0.765

Now, we know all of these values. P(H|I) is in the table, and P(I|W) we just computed in the previous exercise. So, doing the arithmetic, we get 0.765. So, we started with P(H) = 0.59, but knowing that Watson crashed has increased the probability that Holmes has crashed to 0.765.

Prob of Holmes given Icy and Watson



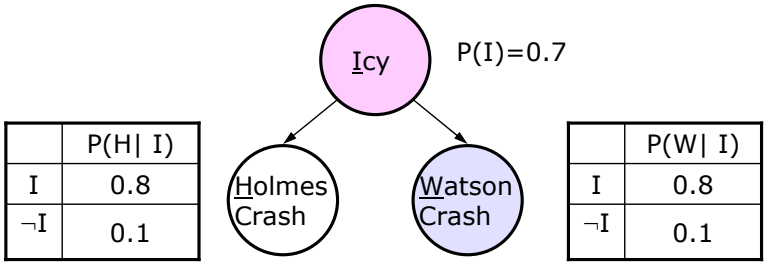
	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

$P(H|W, \neg I) = P(H|\neg I)$

Now we want to compute one more thing. Now the secretary -- the voice of reason, says "Look out the window. It's not icy." So now we know not I. So now we're interested in, one more time, trying to decide whether Holmes is going to come or whether we can sneak out and go have lunch. So we want to know the probability of Holmes coming given that Watson crashed and it's not icy. So this is just the probability of H given not I. Why is that? Because H and W are d-separated, given knowledge of I. So for that reason, they're conditionally independent, and for that reason, we can leave the W out here.

Prob of Holmes given Icy and Watson



	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

$P(H|W, \neg I) = P(H|\neg I) = 0.1$

H and W are d-separated given I, so H and W are conditionally independent given I

So probability H given not I is 0.1. We cut off that whole line of reasoning from Watson to Holmes and all that matters is that it's not icy.

Recitation Problems II

In the Watson and Holmes visit LA network, use the following conditional probability tables.

P(R) = 0.2

P(S) = 0.1

	P(W| R)
R	1.0
¬R	0.2

	P(H| R,S)
R,S	1.0
R,¬S	1.0
¬ R,S	0.9
¬ R,¬S	0.1

Calculate:

P(H), P(R|H), P(S|H), P(W|H), P(R|W,H), P(S|W,H)

Use the following conditional probabilities in the network about Holmes and Watson in LA. Compute the following probabilities, as specified by the network.