Soft Computing

Lecture Notes on Machine Learning

Matteo Matteucci

matteucci@elet.polimi.it

Department of Electronics and Information

Politecnico di Milano

Unsupervised Learning

– Bayesian Networks –

Beyond Independence . . .

We are thankful to the independence hypothesis because:

- It makes computation possible
- It yields optimal classifiers when satisfied
- It gives good enough generalization to our Bayes Classifiers

Beyond Independence . . .

We are thankful to the independence hypothesis because:

- It makes computation possible
- It yields optimal classifiers when satisfied
- It gives good enough generalization to our Bayes Classifiers

Seldom satisfied in practice, as attributes are often correlated!

Beyond Independence . . .

We are thankful to the independence hypothesis because:

- It makes computation possible
- It yields optimal classifiers when satisfied
- It gives good enough generalization to our Bayes Classifiers

Seldom satisfied in practice, as attributes are often correlated!

To overcome this limitation we can describe the probability distribution governing a set of variables by specifying:

- Conditional Independence Assumptions that apply on subsets of them
- A set of conditional probabilities

Beyond Independence . . .

We are thankful to the independence hypothesis because:

- It makes computation possible
- It yields optimal classifiers when satisfied
- It gives good enough generalization to our Bayes Classifiers

Seldom satisfied in practice, as attributes are often correlated!

To overcome this limitation we can describe the probability distribution governing a set of variables by specifying:

- Conditional Independence Assumptions that apply on subsets of them
- A set of conditional probabilities

These models are often referred also as **Graphical Models**

Bayesian Networks Intro

A **Bayesian Belief Networks**, or **Bayesian Network**, is a method to describe the joint probability distribution of a set of variables.

Bayesian Networks Intro

A **Bayesian Belief Networks**, or **Bayesian Network**, is a method to describe the joint probability distribution of a set of variables.

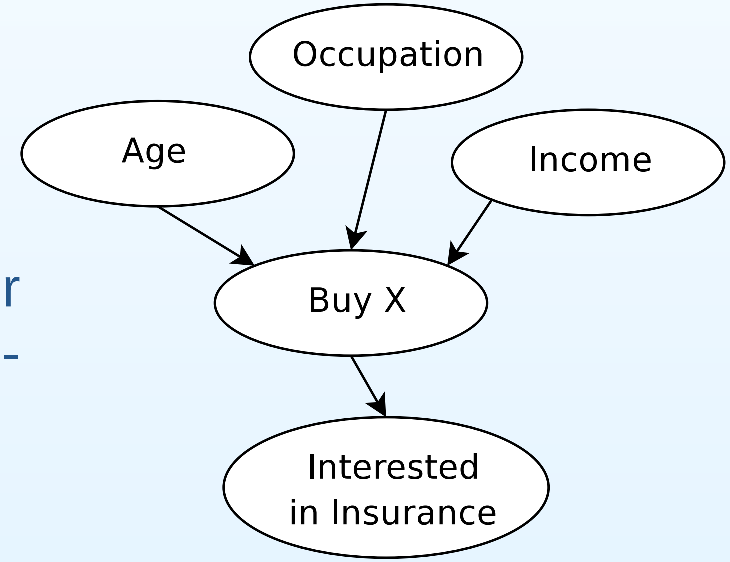
Let $x_1, x_2, \ldots, x_n$ be a set of variables or features. A Bayesian Network will tell us the probability of any combination of $x_1, x_2, \ldots, x_n$.

Bayesian Networks Intro

A **Bayesian Belief Networks**, or **Bayesian Network**, is a method to describe the joint probability distribution of a set of variables.

Let $x_1, x_2, \ldots, x_n$ be a set of variables or features. A Bayesian Network will tell us the probability of any combination of $x_1, x_2, \ldots, x_n$.

- Age, Occupation and Income determine if customer will buy this product.
- Given that customer buys product, whether there is interest in insurance is now independent of Age, Occupation, Income.

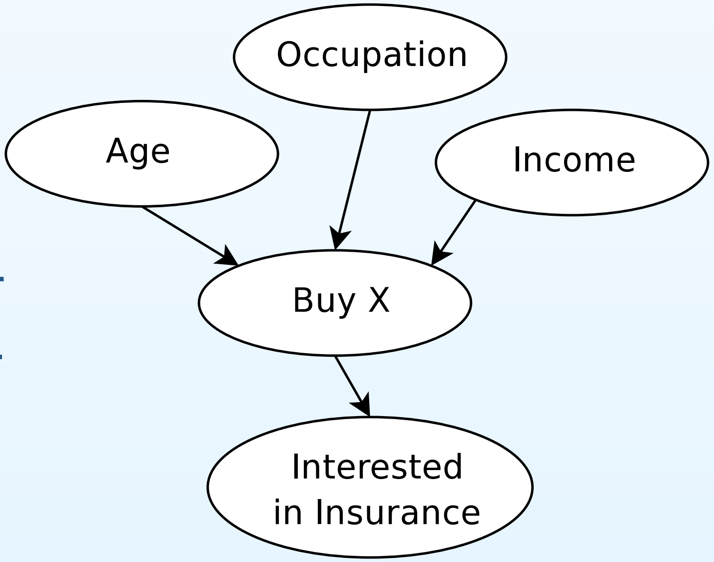


Bayesian Networks Intro

A **Bayesian Belief Networks**, or **Bayesian Network**, is a method to describe the joint probability distribution of a set of variables.

Let $x_1, x_2, \ldots, x_n$ be a set of variables or features. A Bayesian Network will tell us the probability of any combination of $x_1, x_2, \ldots, x_n$.

- Age, Occupation and Income determine if customer will buy this product.
- Given that customer buys product, whether there is interest in insurance is now independent of Age, Occupation, Income.



Similar to Naïve Bayes we will make some independence assumptions, but not as strong as the assumption of all variables being independent.

Bayesian Networks

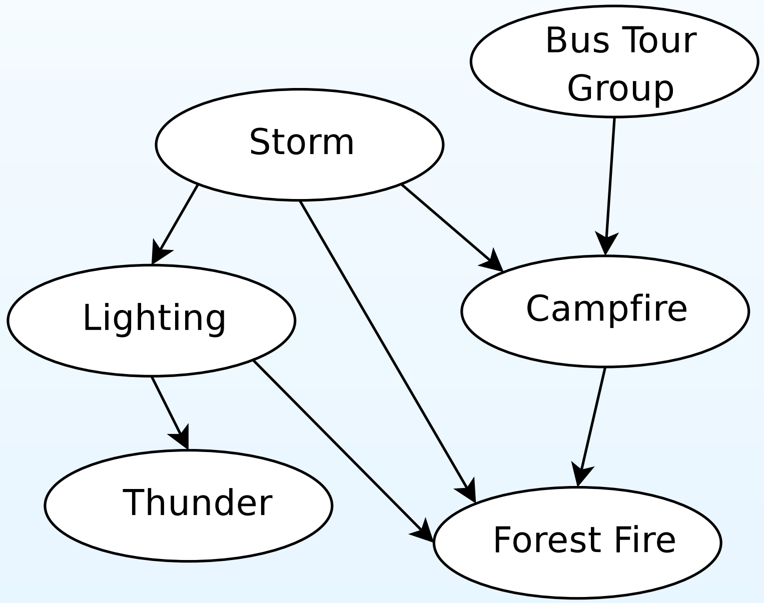
A Bayesian Network is a compact representation of the joint probability distribution of a set of variables by explicitly indicating the assumptions of conditional independence through the following:

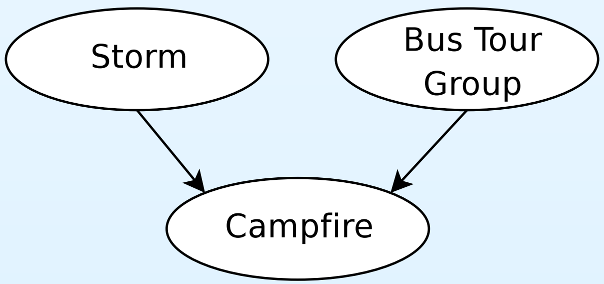
- A Directed Acyclic Graph (DAG)
  - Nodes - random variables
  - Edges - direct influence

Bayesian Networks

A Bayesian Network is a compact representation of the joint probability distribution of a set of variables by explicitly indicating the assumptions of conditional independence through the following:

- A Directed Acyclic Graph (DAG)
  - Nodes - random variables
  - Edges - direct influence
- Set of Conditional Probability Distributions (CPD) for "influenced" variables





	S, B	$S, \sim B$	$\sim S, B$	$\sim S, \sim B$
C	0.4	0.1	0.8	0.2
$\sim C$	0.6	0.9	0.2	0.8

Conditional Independence

We say X_1 is conditionally independent of X_2 given X_3 if the probability of X_1 is independent of X_2 given some knowledge about X_3:

$$P(X_1|X_2, X_3) = P(X_1|X_3)$$

Conditional Independence

We say X_1 is conditionally independent of X_2 given X_3 if the probability of X_1 is independent of X_2 given some knowledge about X_3:

$$P(X_1|X_2, X_3) = P(X_1|X_3)$$

The same can be said for a set of variables: X_1, X_2, X_3 is independent of Y_1, Y_2, Y_3 given Z_1, Z_2, Z_3:

$$P(X_1, X_2, X_3|Y_1, Y_2, Y_3, Z_1, Z_2, Z_3) = P(X_1, X_2, X_3|Z_1, Z_2, Z_3)$$

Conditional Independence

We say X_1 is conditionally independent of X_2 given X_3 if the probability of X_1 is independent of X_2 given some knowledge about X_3:

$$P(X_1|X_2, X_3) = P(X_1|X_3)$$

The same can be said for a set of variables: X_1, X_2, X_3 is independent of Y_1, Y_2, Y_3 given Z_1, Z_2, Z_3:

$$P(X_1, X_2, X_3|Y_1, Y_2, Y_3, Z_1, Z_2, Z_3) = P(X_1, X_2, X_3|Z_1, Z_2, Z_3)$$

Example: Martin and Norman toss the same coin. Let be A "Norman's outcome", and B "Martin's outcome". Assume the coin might be biased; in this case A and B are not independent: observing that B is Heads causes us to increase our belief in A being Heads.

Conditional Independence

We say X_1 is conditionally independent of X_2 given X_3 if the probability of X_1 is independent of X_2 given some knowledge about X_3:

$$P(X_1|X_2, X_3) = P(X_1|X_3)$$

The same can be said for a set of variables: X_1, X_2, X_3 is independent of Y_1, Y_2, Y_3 given Z_1, Z_2, Z_3:

$$P(X_1, X_2, X_3|Y_1, Y_2, Y_3, Z_1, Z_2, Z_3) = P(X_1, X_2, X_3|Z_1, Z_2, Z_3)$$

Example: Martin and Norman toss the same coin. Let be A "Norman's outcome", and B "Martin's outcome". Assume the coin might be biased; in this case A and B are not independent: observing that B is Heads causes us to increase our belief in A being Heads.

Variables A and B are both dependent on C, "the coin is biased towards Heads with probability θ". Once we know for certain the value of C then any evidence about B cannot change our belief about A.

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

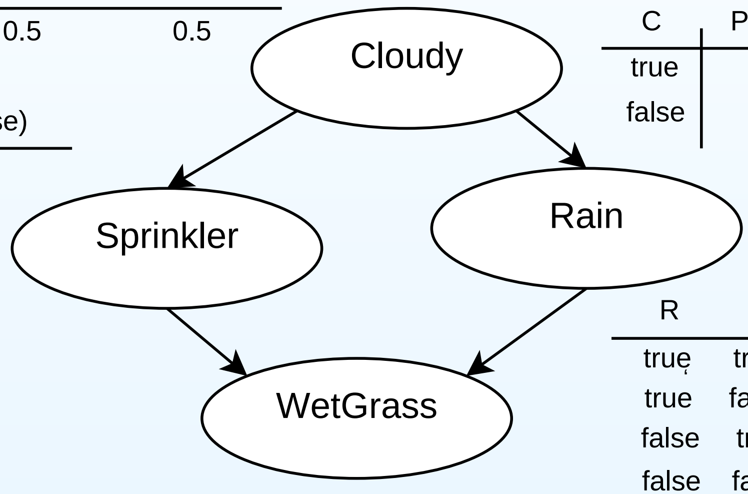
The Sprinkler Example: Modeling

The event “grass is wet” (`W=true`) has two possible causes: either the water *Sprinker* is on (`S=true`) or it is *Raining* (`R=true`).

P(C=true)	P(C=false)
0.5	0.5

C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1

C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8



R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

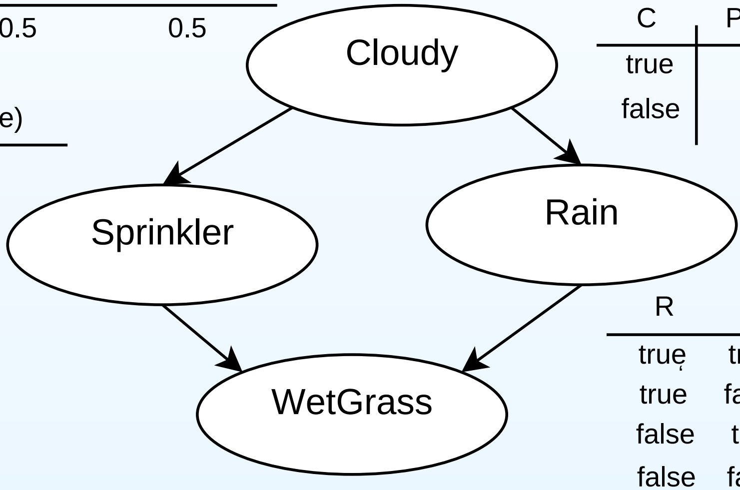
The Sprinkler Example: Modeling

The event “grass is wet” (`W=true`) has two possible causes: either the water *Sprinker* is on (`S=true`) or it is *Raining* (`R=true`).

P(C=true)	P(C=false)
0.5	0.5

C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1

C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8



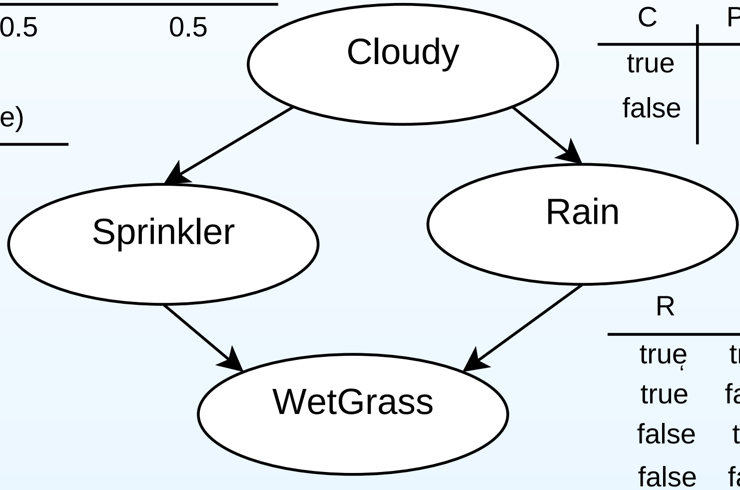
R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

The strength of this relationship is shown in the tables. For example, on second row, $P(W = true|S = true, R = false) = 0.9$, and, since each row sums up to one, $P(W = false|S = true, R = false) = 1 - 0.9 = 0.1$.

The Sprinkler Example: Modeling

The event “grass is wet” (`W=true`) has two possible causes: either the water *Sprinker* is on (`S=true`) or it is *Raining* (`R=true`).

P(C=true)	P(C=false)
0.5	0.5



C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1

C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8

R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

The strength of this relationship is shown in the tables. For example, on second row, $P(W = true|S = true, R = false) = 0.9$, and, since each row sums up to one, $P(W = false|S = true, R = false) = 1 - 0.9 = 0.1$.

The C node has no parents, its Conditional Probability Table (CPT) simply specifies the prior probability that it is *Cloudy* (in this case, 0.5).

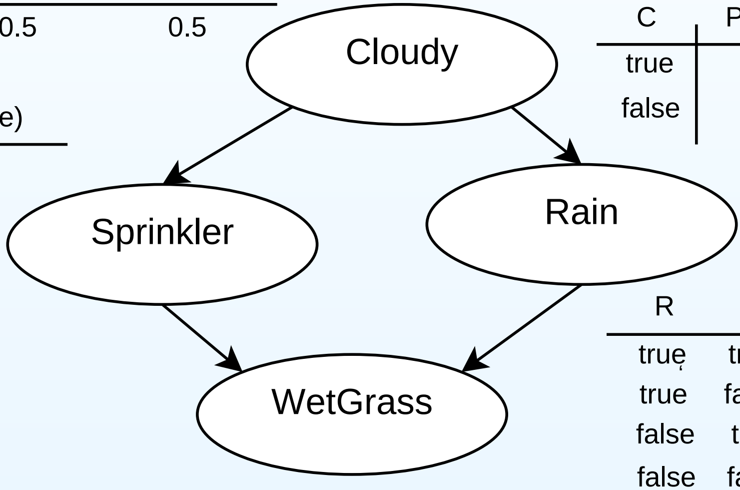
The Sprinkler Example: Joint Probability

The simplest conditional independence encoded in a Bayesian network can be stated as: "a node is independent of its ancestors given its parents."

P(C=true)	P(C=false)
0.5	0.5

C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1

C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8



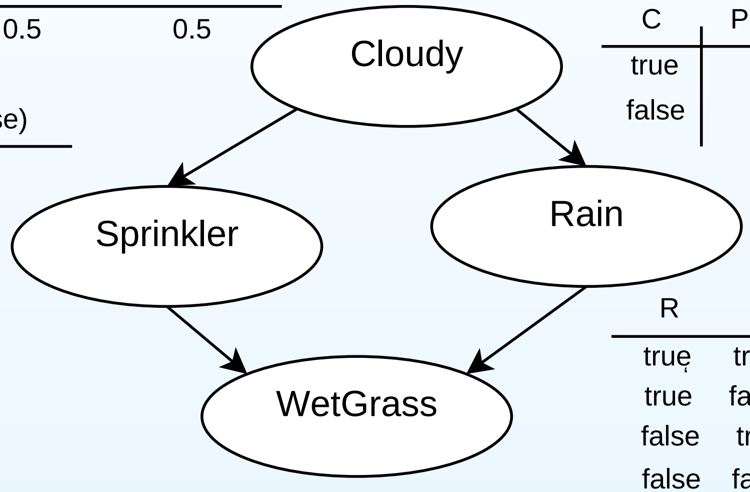
R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

The Sprinkler Example: Joint Probability

The simplest conditional independence encoded in a Bayesian network can be stated as: “a node is independent of its ancestors given its parents.”

P(C=true)	P(C=false)
0.5	0.5

C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1



C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8

R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

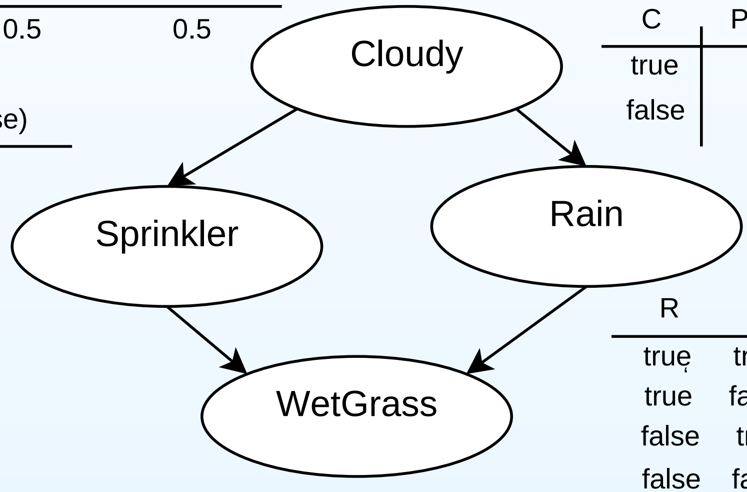
Using the chain rule we get the joint probability of nodes in the graph

$$\begin{aligned} P(C,S,R,W) &= P(W|C,S,R)P(R|C,S)P(S|C)P(C) \\ &= P(W|S,R)P(R|C)P(S|C)P(C). \end{aligned}$$

The Sprinkler Example: Joint Probability

The simplest conditional independence encoded in a Bayesian network can be stated as: “a node is independent of its ancestors given its parents.”

P(C=true)	P(C=false)
0.5	0.5



C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1

C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8

R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

Using the chain rule we get the joint probability of nodes in the graph

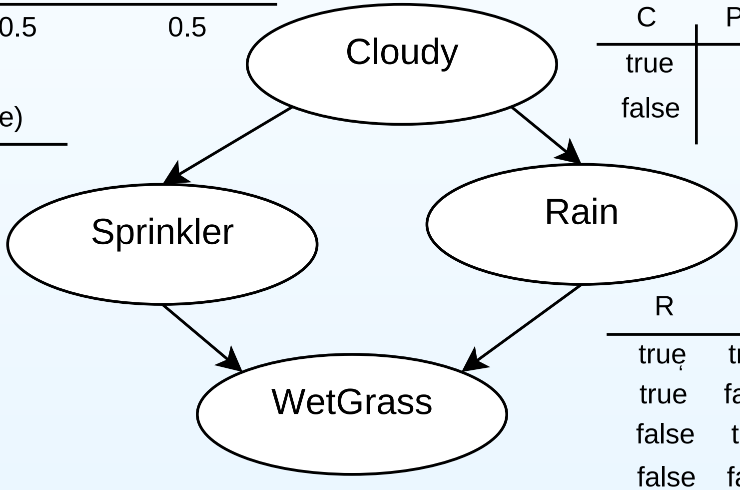
$$\begin{aligned} P(C,S,R,W) &= P(W|C,S,R)P(R|C,S)P(S|C)P(C) \\ &= P(W|S,R)P(R|C)P(S|C)P(C). \end{aligned}$$

In general, with N binary nodes and being k the maximum node fan-in, the full joint requires $O(2^N)$ parameters while the factored one $O(N \cdot 2^k)$.

The Sprinkler Example: Making Inference

We observe the fact that the grass is wet. There are two possible causes for this: (a) the sprinkler is on or (b) it is raining. Which is more likely?

P(C=true)	P(C=false)
0.5	0.5



C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1

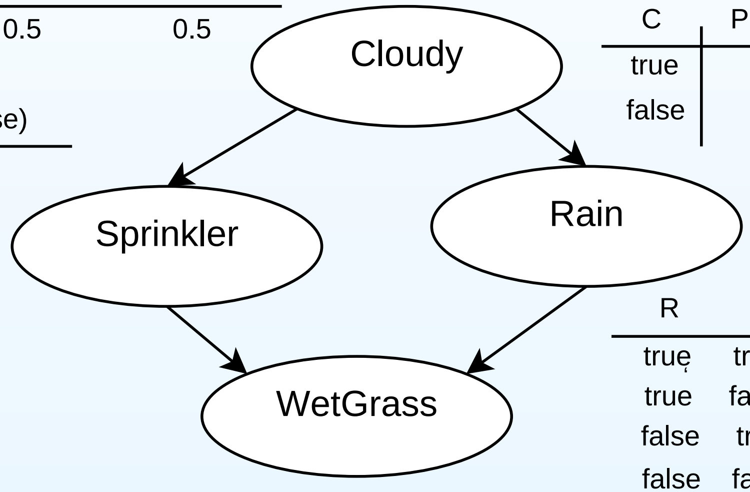
C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8

R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

The Sprinkler Example: Making Inference

We observe the fact that the grass is wet. There are two possible causes for this: (a) the sprinkler is on or (b) it is raining. Which is more likely?

P(C=true)	P(C=false)
0.5	0.5



C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1

C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8

R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

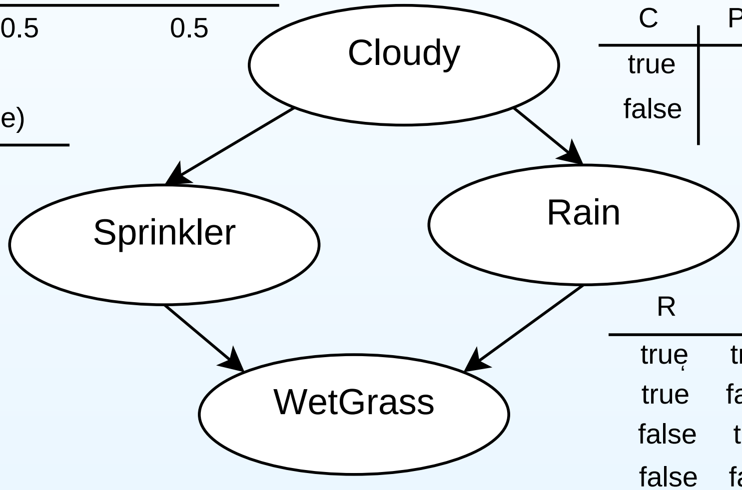
Use Bayes' rule to compute the posterior probability of each explanation:

$$P(S|W) = P(S,W)/P(W) = \sum_{c,r} P(C,S,R,W)/P(W) = 0.430$$

The Sprinkler Example: Making Inference

We observe the fact that the grass is wet. There are two possible causes for this: (a) the sprinkler is on or (b) it is raining. Which is more likely?

P(C=true)	P(C=false)
0.5	0.5



C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1

C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8

R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

Use Bayes' rule to compute the posterior probability of each explanation:

$$P(S|W) = P(S,W)/P(W) = \sum_{c,r} P(C,S,R,W)/P(W) = 0.430$$

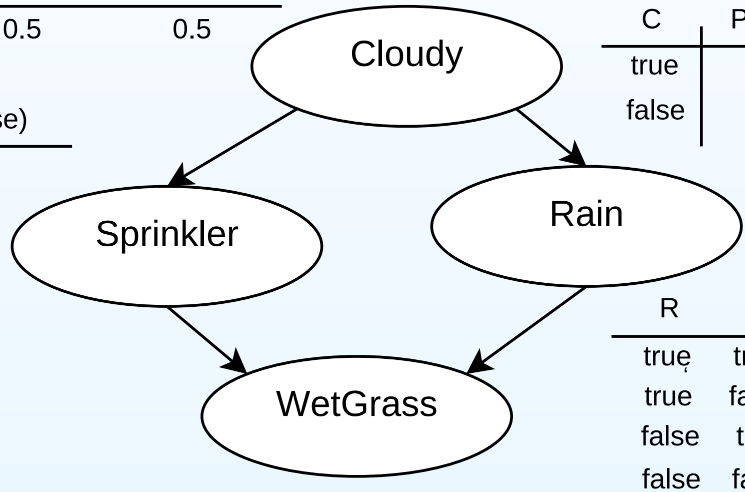
$$P(R|W) = P(R,W)/P(W) = \sum_{c,s} P(C,S,R,W)/P(W) = 0.708$$

The Sprinkler Example: Making Inference

We observe the fact that the grass is wet. There are two possible causes for this: (a) the sprinkler is on or (b) it is raining. Which is more likely?

P(C=true)	P(C=false)
0.5	0.5

C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1



C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8

R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

Use Bayes' rule to compute the posterior probability of each explanation:

$$P(S|W) = P(S,W)/P(W) = \sum_{c,r} P(C,S,R,W)/P(W) = 0.430$$

$$P(R|W) = P(R,W)/P(W) = \sum_{c,s} P(C,S,R,W)/P(W) = 0.708$$

$P(W)$ is a normalizing constant, equal to the probability (likelihood) of the data; the likelihood ratio is $0.7079/0.4298 = 1.647$.

The Sprinkler Example: Explaining Away

In Sprinkler Example the two causes *"compete"* to *"explain"* the observed data. Hence S and R become conditionally dependent given that their common child, W, is observed.

Example: Suppose the grass is wet, but we know that it is raining. Then the posterior probability of sprinkler being on goes down: $P(S|W, R) = 0.1945$

The Sprinkler Example: Explaining Away

In Sprinkler Example the two causes *“compete”* to *“explain”* the observed data. Hence S and R become conditionally dependent given that their common child, W, is observed.

Example: Suppose the grass is wet, but we know that it is raining. Then the posterior probability of sprinkler being on goes down: $P(S|W, R) = 0.1945$

This is phaenomenon is called ***Explaining Away***, and, in statistics, it is also known as *Berkson’s Paradox*, or *Selection Bias*.

The Sprinkler Example: Explaining Away

In Sprinkler Example the two causes *"compete"* to *"explain"* the observed data. Hence S and R become conditionally dependent given that their common child, W, is observed.

Example: Suppose the grass is wet, but we know that it is raining. Then the posterior probability of sprinkler being on goes down: $P(S|W, R) = 0.1945$

This is phaenomenon is called ***Explaining Away***, and, in statistics, it is also known as *Berkson's Paradox*, or *Selection Bias*.

Example: Consider a college which admits students who are either *Brainy* or *Sporty* (or both!). Let C denote the event that someone is admitted to *College*, which is made true if they are either Brainy (B) or Sporty (S). Suppose in the general population, B and S are independent.

The Sprinkler Example: Explaining Away

In Sprinkler Example the two causes *“compete”* to *“explain”* the observed data. Hence S and R become conditionally dependent given that their common child, W, is observed.

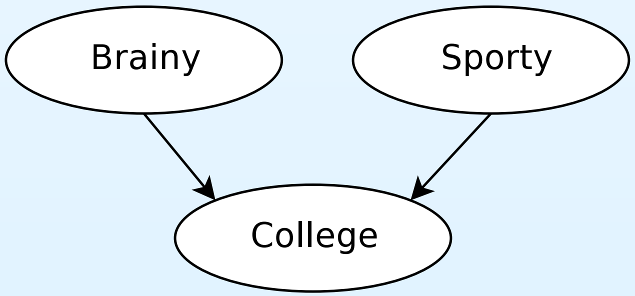
Example: Suppose the grass is wet, but we know that it is raining. Then the posterior probability of sprinkler being on goes down: $P(S|W, R) = 0.1945$

This is phaenomenon is called ***Explaining Away***, and, in statistics, it is also known as *Berkson’s Paradox*, or *Selection Bias*.

Example: Consider a college which admits students who are either *Brainy* or *Sporty* (or both!). Let C denote the event that someone is admitted to *College*, which is made true if they are either Brainy (B) or Sporty (S). Suppose in the general population, B and S are independent.

In College population, being Brainy makes you less likely to be Sporty and vice versa, because either property alone is sufficient to explain evidence on C

$$P(S = 1|C = 1, B = 1) \leq P(S = 1|C = 1)$$



Bottom-up and Top-down Reasoning

Looking at the simple Sprinkler Example we can already see the two kind of reasoning we can make with Bayesian Networks:

- “bottom up” reasoning: we had evidence of an effect (Wet Grass), and inferred the most likely cause; it goes from effects to causes and it is a common task in expert systems or diagnostic;

Bottom-up and Top-down Reasoning

Looking at the simple Sprinkler Example we can already see the two kind of reasoning we can make with Bayesian Networks:

- “bottom up” reasoning: we had evidence of an effect (Wet Grass), and inferred the most likely cause; it goes from effects to causes and it is a common task in expert systems or diagnostic;
- “top down”, reasoning: we can compute the probability that the grass will be wet given that it is cloudy; for this reason Bayesian Networks are often called “generative” models.

Bottom-up and Top-down Reasoning

Looking at the simple Sprinkler Example we can already see the two kind of reasoning we can make with Bayesian Networks:

- “bottom up” reasoning: we had evidence of an effect (Wet Grass), and inferred the most likely cause; it goes from effects to causes and it is a common task in expert systems or diagnostic;
- “top down”, reasoning: we can compute the probability that the grass will be wet given that it is cloudy; for this reason Bayesian Networks are often called “generative” models.

The most interesting property of Bayesian Networks is that they can be used to reason about causality on a solid mathematical basis:

Question: Can we distinguish causation from mere correlation? So we don’t need to make experiments to infer causality.

Bottom-up and Top-down Reasoning

Looking at the simple Sprinkler Example we can already see the two kind of reasoning we can make with Bayesian Networks:

- "bottom up" reasoning: we had evidence of an effect (Wet Grass), and inferred the most likely cause; it goes from effects to causes and it is a common task in expert systems or diagnostic;
- "top down", reasoning: we can compute the probability that the grass will be wet given that it is cloudy; for this reason Bayesian Networks are often called "generative" models.

The most interesting property of Bayesian Networks is that they can be used to reason about causality on a solid mathematical basis:

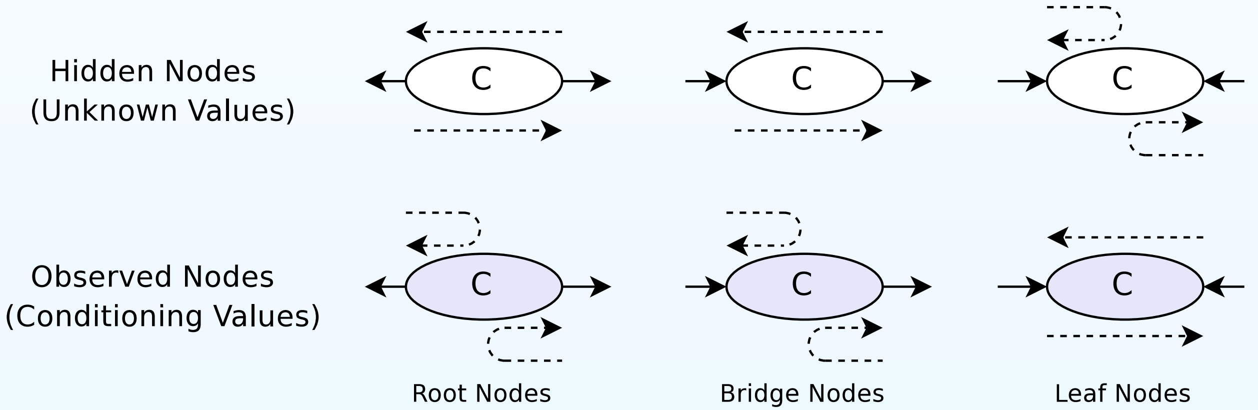
Question: Can we distinguish causation from mere correlation? So we don't need to make experiments to infer causality.

Answer: Yes, "sometimes", but we need to measure the relationships between at least three variables.

For details refer to Causality: Models, Reasoning and Inference,Judea Pearl, 2000.

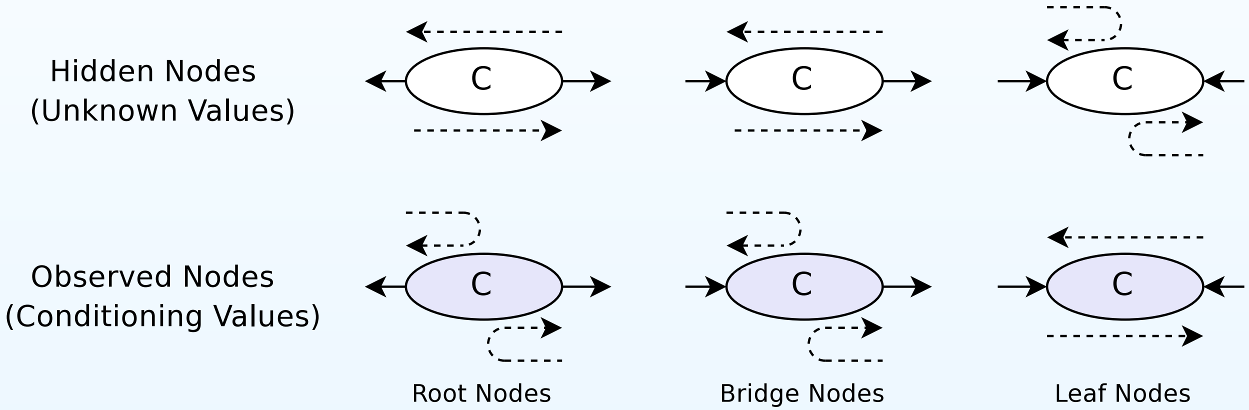
Conditional Independence in Bayesian Networks

Two (sets of) nodes A and B are conditionally independent (d-separated) given C if and only iif all the path from A to B are shielded by C.



Conditional Independence in Bayesian Networks

Two (sets of) nodes A and B are conditionally independent (d-separated) given C if and only iif all the path from A to B are shielded by C.

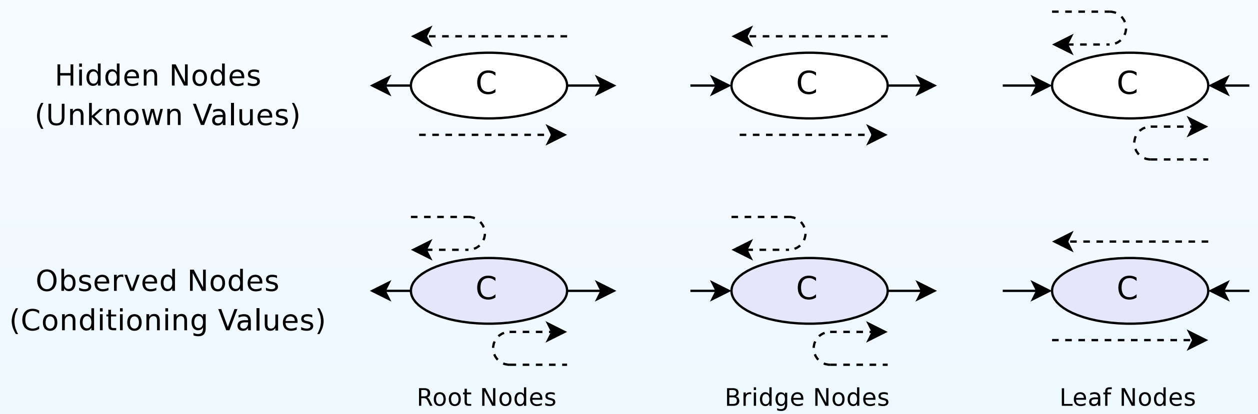


The dotted arcs indicate direction of flow in the path:

- C is a “root”: if C is hidden, children are dependent due to a hidden common cause. If C is observed, they are conditionally independent;

Conditional Independence in Bayesian Networks

Two (sets of) nodes A and B are conditionally independent (d-separated) given C if and only iif all the path from A to B are shielded by C.

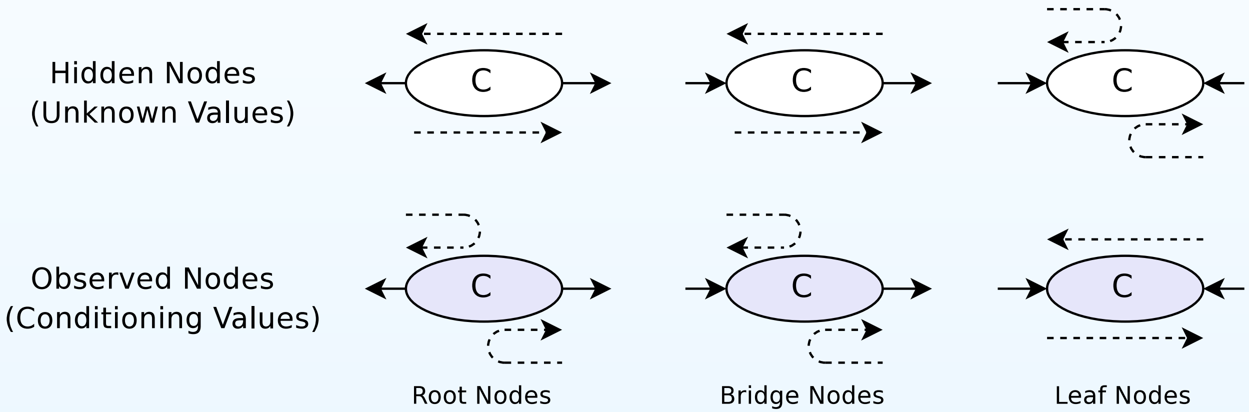


The dotted arcs indicate direction of flow in the path:

- C is a "root": if C is hidden, children are dependent due to a hidden common cause. If C is observed, they are conditionally independent;
- C is a "leaf": if C is hidden, its parents are marginally independent, but if c is observed, the parents become dependent (Explaining Away);

Conditional Independence in Bayesian Networks

Two (sets of) nodes A and B are conditionally independent (d-separated) given C if and only iif all the path from A to B are shielded by C.



The dotted arcs indicate direction of flow in the path:

- C is a “root”: if C is hidden, children are dependent due to a hidden common cause. If C is observed, they are conditionally independent;
- C is a “leaf”: if C is hidden, its parents are marginally independent, but if c is observed, the parents become dependent (Explaining Away);
- C is a “bridge”: nodes upstream and downstream of C are dependent iff C is hidden, because conditioning breaks the graph at that point.

Undirected Bayesian Networks

Undirected graphical models, also called **Markov Random Fields** (MRFs) or **Markov Networks**, are more popular with the Physics and Computer Vision communities, and have a simpler definition of independence:

Two (sets of) nodes A and B are conditionally independent given set C, if all paths between the nodes in A and B are separated by a node in C.

Undirected Bayesian Networks

Undirected graphical models, also called **Markov Random Fields** (MRFs) or **Markov Networks**, are more popular with the Physics and Computer Vision communities, and have a simpler definition of independence:

Two (sets of) nodes A and B are conditionally independent given set C, if all paths between the nodes in A and B are separated by a node in C.

Directed graph independence is more complex independence, but:

- We can regard an arc from A to B as indicating that A “causes” B;

Undirected Bayesian Networks

Undirected graphical models, also called **Markov Random Fields** (MRFs) or **Markov Networks**, are more popular with the Physics and Computer Vision communities, and have a simpler definition of independence:

Two (sets of) nodes A and B are conditionally independent given set C, if all paths between the nodes in A and B are separated by a node in C.

Directed graph independence is more complex independence, but:

- We can regard an arc from A to B as indicating that A "causes" B;
- Causality can be used to construct the graph structure;

Undirected Bayesian Networks

Undirected graphical models, also called **Markov Random Fields** (MRFs) or **Markov Networks**, are more popular with the Physics and Computer Vision communities, and have a simpler definition of independence:

Two (sets of) nodes A and B are conditionally independent given set C, if all paths between the nodes in A and B are separated by a node in C.

Directed graph independence is more complex independence, but:

- We can regard an arc from A to B as indicating that A "causes" B;
- Causality can be used to construct the graph structure;
- We can encode deterministic relationships too;

Undirected Bayesian Networks

Undirected graphical models, also called **Markov Random Fields** (MRFs) or **Markov Networks**, are more popular with the Physics and Computer Vision communities, and have a simpler definition of independence:

Two (sets of) nodes A and B are conditionally independent given set C, if all paths between the nodes in A and B are separated by a node in C.

Directed graph independence is more complex independence, but:

- We can regard an arc from A to B as indicating that A "causes" B;
- Causality can be used to construct the graph structure;
- We can encode deterministic relationships too;
- They are easier to learn (fit to data).

Undirected Bayesian Networks

Undirected graphical models, also called **Markov Random Fields** (MRFs) or **Markov Networks**, are more popular with the Physics and Computer Vision communities, and have a simpler definition of independence:

Two (sets of) nodes A and B are conditionally independent given set C, if all paths between the nodes in A and B are separated by a node in C.

Directed graph independence is more complex independence, but:

- We can regard an arc from A to B as indicating that A "causes" B;
- Causality can be used to construct the graph structure;
- We can encode deterministic relationships too;
- They are easier to learn (fit to data).

When converting a directed graph to an undirected graph, we must add links between "unmarried" parents who share a common child (i.e., "**moralize**" the graph) to prevent reading incorrect independences.

Graphical Models with Real Values

We can have Bayesian Networks with real valued nodes:

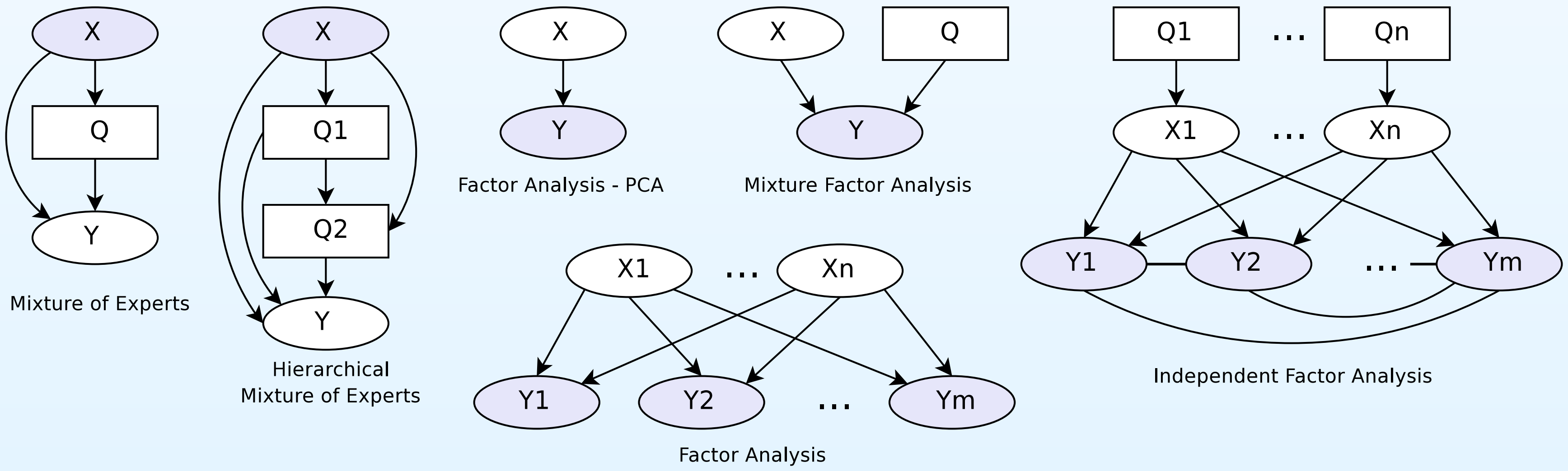
- For discrete nodes with continuous parents, we can use the logistic/softmax distribution;
- The most common distribution for real nodes is Gaussian.

Graphical Models with Real Values

We can have Bayesian Networks with real valued nodes:

- For discrete nodes with continuous parents, we can use the logistic/softmax distribution;
- The most common distribution for real nodes is Gaussian.

Using these nodes we can obtain a rich toolbox for complex probabilistic modeling (circle=real, square=discrete, clear=hidden, shaded=observed):

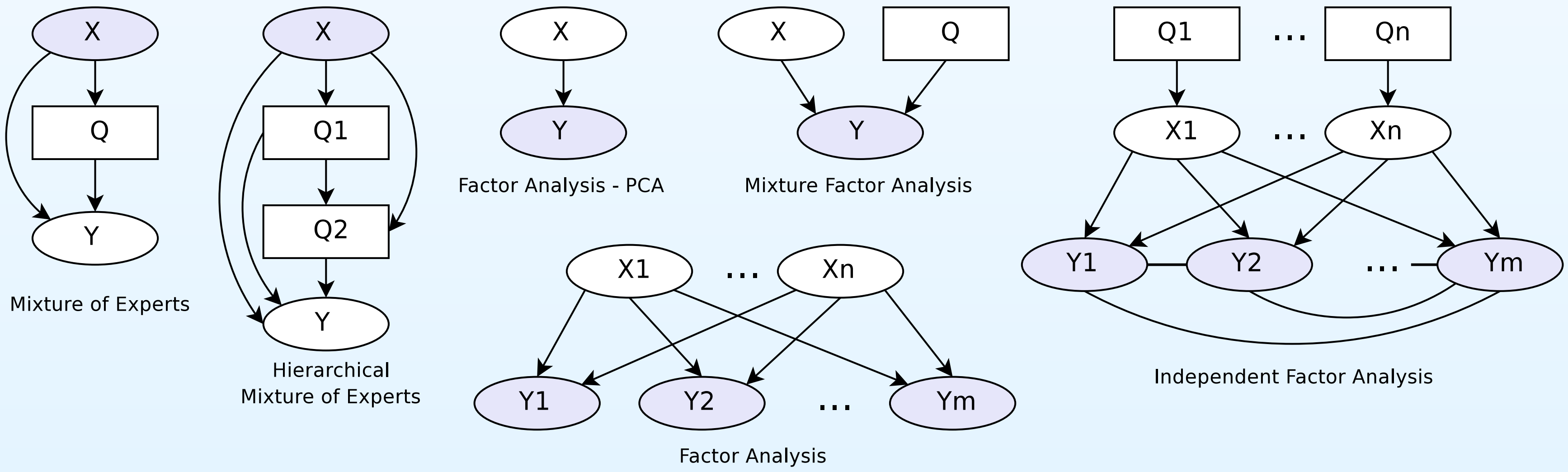


Graphical Models with Real Values

We can have Bayesian Networks with real valued nodes:

- For discrete nodes with continuous parents, we can use the logistic/softmax distribution;
- The most common distribution for real nodes is Gaussian.

Using these nodes we can obtain a rich toolbox for complex probabilistic modeling (circle=real, square=discrete, clear=hidden, shaded=observed):



More details in the illuminating paper by Sam Roweis & Zoubin Ghahramani:
A Unifying Review of Linear Gaussian Models, Neural Computation 11(2) (1999) pp.305-345.

Inference Algorithms in Bayesian Networks

A graphical model specifies a complete joint probability distribution:

- Given the joint probability, we can answer all possible inference queries by marginalization (i.e., summing out irrelevant variables);

Inference Algorithms in Bayesian Networks

A graphical model specifies a complete joint probability distribution:

- Given the joint probability, we can answer all possible inference queries by marginalization (i.e., summing out irrelevant variables);
- The joint probability distribution has size $O(2^n)$, where n is the number of nodes, and we have assumed each node can have 2 states.

Inference Algorithms in Bayesian Networks

A graphical model specifies a complete joint probability distribution:

- Given the joint probability, we can answer all possible inference queries by marginalization (i.e., summing out irrelevant variables);
- The joint probability distribution has size $O(2^n)$, where n is the number of nodes, and we have assumed each node can have 2 states.

Hence inference on Bayesian Networks takes exponential time!

Inference Algorithms in Bayesian Networks

A graphical model specifies a complete joint probability distribution:

- Given the joint probability, we can answer all possible inference queries by marginalization (i.e., summing out irrelevant variables);
- The joint probability distribution has size $O(2^n)$, where n is the number of nodes, and we have assumed each node can have 2 states.

Hence inference on Bayesian Networks takes exponential time!

Lots of work have been done to overcome this issue:

- Variable Elimination

Inference Algorithms in Bayesian Networks

A graphical model specifies a complete joint probability distribution:

- Given the joint probability, we can answer all possible inference queries by marginalization (i.e., summing out irrelevant variables);
- The joint probability distribution has size $O(2^n)$, where n is the number of nodes, and we have assumed each node can have 2 states.

Hence inference on Bayesian Networks takes exponential time!

Lots of work have been done to overcome this issue:

- Variable Elimination
- Dynamic Programming (message passing & junction trees)

Inference Algorithms in Bayesian Networks

A graphical model specifies a complete joint probability distribution:

- Given the joint probability, we can answer all possible inference queries by marginalization (i.e., summing out irrelevant variables);
- The joint probability distribution has size $O(2^n)$, where n is the number of nodes, and we have assumed each node can have 2 states.

Hence inference on Bayesian Networks takes exponential time!

Lots of work have been done to overcome this issue:

- Variable Elimination
- Dynamic Programming (message passing & junction trees)
- Approximated methods:
  - Sampling (Monte Carlo) methods
  - Variational approximation
  - ...

Inference Algorithms in Bayesian Networks

A graphical model specifies a complete joint probability distribution:

- Given the joint probability, we can answer all possible inference queries by marginalization (i.e., summing out irrelevant variables);
- The joint probability distribution has size $O(2^n)$, where n is the number of nodes, and we have assumed each node can have 2 states.

Hence inference on Bayesian Networks takes exponential time!

Lots of work have been done to overcome this issue:

- Variable Elimination
- Dynamic Programming (message passing & junction trees)
- Approximated methods:
  - Sampling (Monte Carlo) methods
  - Variational approximation
  - ...

We'll see just a few of them ... don't worry!

Inference in Bayes Nets: Variable Elimination

We can sometimes use the factored representation of Joit Probability to do marginalisation efficiently. The key idea is to “push sums in” as far as possible when summing (marginalizing) out irrelevant terms:

$$\begin{aligned} p(W) &= \sum_c \sum_s \sum_r P(C,S,R,W) \\ &= \sum_c \sum_s \sum_r P(W|S,R)P(R|C)P(S|C)P(C) \\ &= \sum_c P(C) \sum_s P(S|C) \sum_r P(W|S,R)P(R|C) \end{aligned}$$

Inference in Bayes Nets: Variable Elimination

We can sometimes use the factored representation of Joit Probability to do marginalisation efficiently. The key idea is to “push sums in” as far as possible when summing (marginalizing) out irrelevant terms:

$$
\begin{aligned}
p(W) &= \sum_c \sum_s \sum_r P(C,S,R,W) \\
&= \sum_c \sum_s \sum_r P(W|S,R)P(R|C)P(S|C)P(C) \\
&= \sum_c P(C) \sum_s P(S|C) \sum_r P(W|S,R)P(R|C)
\end{aligned}
$$

As we perform the innermost sums we create new terms to be summed:

- $T_1(C,W,S) = \sum_r P(W|S,R)P(R|C)$;
- $T_2(C,W) = \sum_s P(S|C)T_1(C,W,S)$;
- $P(W) = \sum_c P(C)T_2(C,W)$.

Inference in Bayes Nets: Variable Elimination

We can sometimes use the factored representation of Joit Probability to do marginalisation efficiently. The key idea is to “push sums in” as far as possible when summing (marginalizing) out irrelevant terms:

$$\begin{aligned} p(W) &= \sum_c \sum_s \sum_r P(C,S,R,W) \\ &= \sum_c \sum_s \sum_r P(W|S,R)P(R|C)P(S|C)P(C) \\ &= \sum_c P(C) \sum_s P(S|C) \sum_r P(W|S,R)P(R|C) \end{aligned}$$

As we perform the innermost sums we create new terms to be summed:

- $T_1(C,W,S) = \sum_r P(W|S,R)P(R|C)$;
- $T_2(C,W) = \sum_s P(S|C)T_1(C,W,S)$;
- $P(W) = \sum_c P(C)T_2(C,W)$.

Complexity is bounded by the size of the largest term. Finding the optimal order is NP-hard, although greedy algorithms work well in practice.

Inference in Bayes Nets: Local Message Passing (I)

If the underlying undirected graph of the BN is acyclic (i.e., a tree), we can use a local message passing algorithm:

- Suppose we want $P(X_i|E)$ where E is some set of evidence variables

Inference in Bayes Nets: Local Message Passing (I)

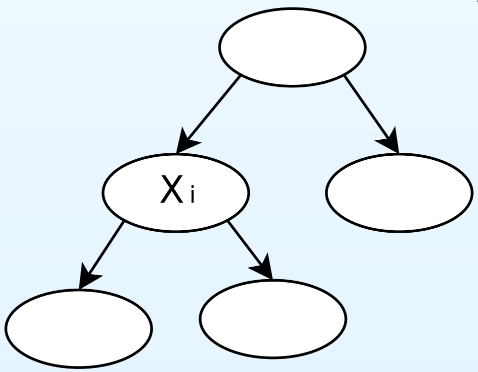
If the underlying undirected graph of the BN is acyclic (i.e., a tree), we can use a local message passing algorithm:

- Suppose we want $P(X_i|E)$ where E is some set of evidence variables
- Let's Split E into two parts:
  - E_i^- assignments to variables in the subtree rooted at X_i;
  - E_i^+ the rest of E

Inference in Bayes Nets: Local Message Passing (I)

If the underlying undirected graph of the BN is acyclic (i.e., a tree), we can use a local message passing algorithm:

- Suppose we want $P(X_i|E)$ where E is some set of evidence variables
- Let's Split E into two parts:
  - E_i^- assignments to variables in the subtree rooted at X_i;
  - E_i^+ the rest of E



$$\begin{aligned} P(X_i|E) &= P(X_i|E_i^-, E_i^+) \\ &= \frac{P(E_i^-|X_i, E_i^+)P(X_i|E_i^+)}{P(E_i^-|E_i^+)} \\ &= \frac{P(E_i^-|X_i)P(X_i|E_i^+)}{P(E_i^-|E_i^+)} \\ &= \alpha\pi(X_i)\lambda(X_i) \end{aligned}$$

With α independent from X_i, $\pi(X_i) = P(X_i|E_i^+), \lambda(X_i) = P(E_i^-|X_i)$.

Inference in Bayes Nets: Local Message Passing (II)

We can exploit such decomposition to compute $\lambda(X_i) = P(E_i^- | X_i)$ for all X_i recursively as follows:

Inference in Bayes Nets: Local Message Passing (II)

We can exploit such decomposition to compute $\lambda(X_i) = P(E_i^-|X_i)$ for all X_i recursively as follows:

- `if` X_i is a leaf
  - $X_i \in E$: `then` $\lambda(X_i) = 1$ if X_i matches E, 0 otherwise;
  - $X_i \notin E$: E_i^- is the empty set so $\lambda(X_i) = 1$

Inference in Bayes Nets: Local Message Passing (II)

We can exploit such decomposition to compute $\lambda(X_i) = P(E_i^-|X_i)$ for all X_i recursively as follows:

- `if` X_i is a leaf
  - $X_i \in E$: `then` $\lambda(X_i) = 1$ if X_i matches E, 0 otherwise;
  - $X_i \notin E$: E_i^- is the empty set so $\lambda(X_i) = 1$
- `if` X_i has one child X_c

$$\begin{aligned}\lambda(X_i) &= P(E_i^-|X_i) = \sum_j P(E_i, X_c = j|X_i) \\ &= \sum_j P(X_c = j|X_i)P(E_i^-|X_i, X_c = j) = \sum_j P(X_c = j|X_i)\lambda(X_c = j)\end{aligned}$$

Inference in Bayes Nets: Local Message Passing (II)

We can exploit such decomposition to compute $\lambda(X_i) = P(E_i^-|X_i)$ for all X_i recursively as follows:

- `if` X_i is a leaf
  - $X_i \in E$: `then` $\lambda(X_i) = 1$ if X_i matches E, 0 otherwise;
  - $X_i \notin E$: E_i^- is the empty set so $\lambda(X_i) = 1$
- `if` X_i has one child X_c

$$\begin{aligned}\lambda(X_i) &= P(E_i^-|X_i) = \sum_j P(E_i, X_c = j|X_i) \\ &= \sum_j P(X_c = j|X_i) P(E_i^-|X_i, X_c = j) = \sum_j P(X_c = j|X_i)\lambda(X_c = j)\end{aligned}$$

- `if` X_i has a set of children C since X_i d-separates them

$$\lambda(X_i) = P(E_i^-|X_i) = \prod_{X_j \in C} \lambda_j(X_j) = \prod_{X_j \in C} \left(\sum_{X_j} P(X_j|X_i)\lambda(X_j) \right)$$

where $\lambda_j(X_j)$ is the conribution to $P(E_i^-|X_i)$ of subtree rooted at X_j.

Inference in Bayes Nets: Local Message Passing (III)

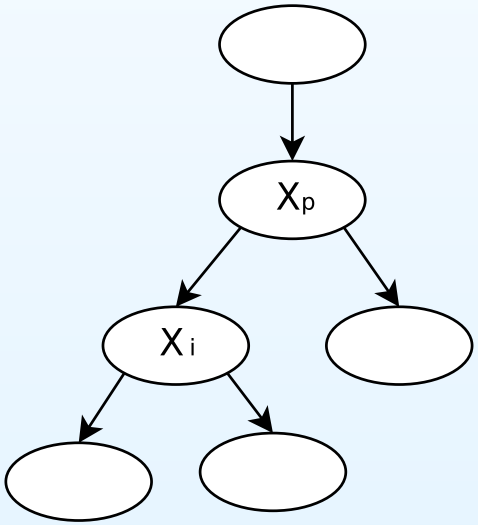
We can now compute the rest of our inference: $\pi(X_i) = P(X_i|E_i^+)$

- For the root of the tree X_r we have E_i^+ is empty thus $\pi(X_i) = P(X_i)$

Inference in Bayes Nets: Local Message Passing (III)

We can now compute the rest of our inference: $\pi(X_i) = P(X_i|E_i^+)$

- For the root of the tree X_r we have E_i^+ is empty thus $\pi(X_i) = P(X_i)$
- For an arbitrary X_i with parent X_p knowing X_p and/or $P(X_p|E)$:

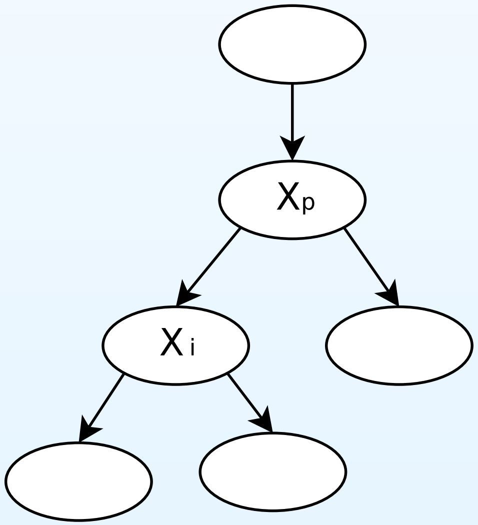


$$\begin{aligned}
\pi(X_i) &= P(X_i|E_i^+) = \sum_j P(X_i, X_p = j|E_i^+) \\
&= \sum_j P(X_i|X_p = j, E_i^+) P(X_p = j|E_i^+) \\
&= \sum_j P(X_i|X_p = j) P(X_p = j|E_i^+) \\
&= \sum_j P(X_i|X_p = j) \frac{P(X_p = j|E)}{\lambda_i(X_p = j)} \\
&= \sum_j P(X_i|X_p = j) \pi_i(X_p = j).
\end{aligned}$$

Inference in Bayes Nets: Local Message Passing (III)

We can now compute the rest of our inference: $\pi(X_i) = P(X_i|E_i^+)$

- For the root of the tree X_r we have E_i^+ is empty thus $\pi(X_i) = P(X_i)$
- For an arbitrary X_i with parent X_p knowing X_p and/or $P(X_p|E)$:



$$\begin{aligned}
\pi(X_i) &= P(X_i|E_i^+) = \sum_j P(X_i, X_p = j|E_i^+) \\
&= \sum_j P(X_i|X_p = j, E_i^+) P(X_p = j|E_i^+) \\
&= \sum_j P(X_i|X_p = j) P(X_p = j|E_i^+) \\
&= \sum_j P(X_i|X_p = j) \frac{P(X_p = j|E)}{\lambda_i(X_p = j)} \\
&= \sum_j P(X_i|X_p = j) \pi_i(X_p = j).
\end{aligned}$$

Having defined $\pi_i(X_p = j)$ equal to $\frac{P(X_p=j|E)}{\lambda_i(X_p=j)}$ we can now compute all $\pi(X_i)$s and then all the $P(X_i|E)$s!

Inference in Bayes Nets: Local Message Passing (IV)

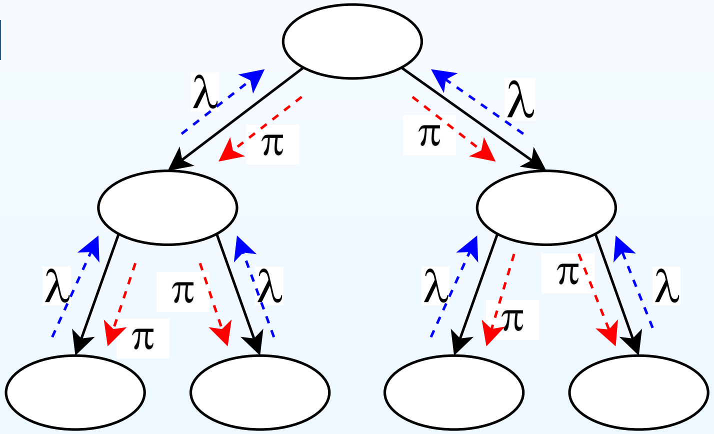
In the message passing algorithm, we can thing nodes as autonomous processors passing λ and π messages to their neighbors.

Inference in Bayes Nets: Local Message Passing (IV)

In the message passing algorithm, we can thing nodes as autonomous processors passing λ and π messages to their neighbors.

If we want $P(A, B|C)$ instead of just marginal distrtibutions $P(A|C)$ and $P(B|C)$?

- Apply the chain rule: $P(A, B|C) = P(A|B, C)P(B|C)$;
- Apply Local Message Passing twice.

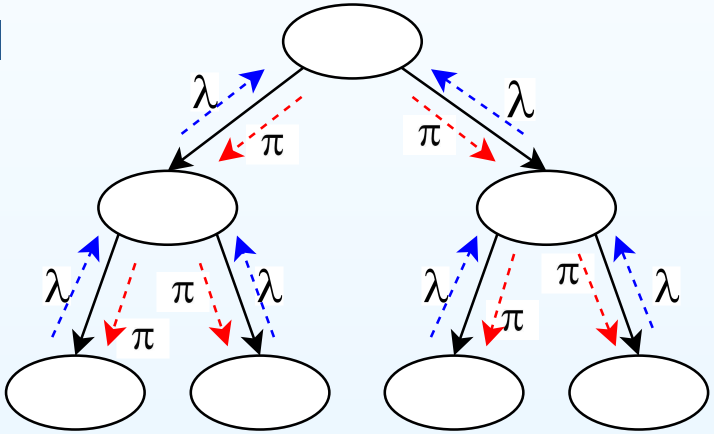


Inference in Bayes Nets: Local Message Passing (IV)

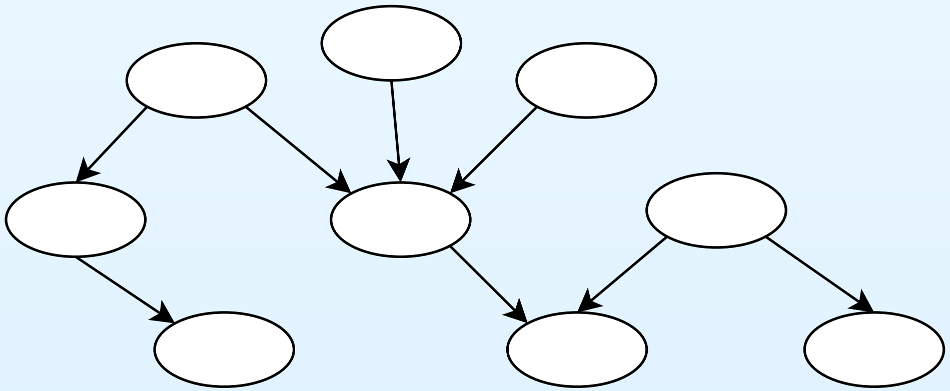
In the message passing algorithm, we can thing nodes as autonomous processors passing λ and π messages to their neighbors.

If we want $P(A,B|C)$ instead of just marginal distrtibutions $P(A|C)$ and $P(B|C)$?

- Apply the chain rule: $P(A,B|C) = P(A|B,C)P(B|C)$;
- Apply Local Message Passing twice.



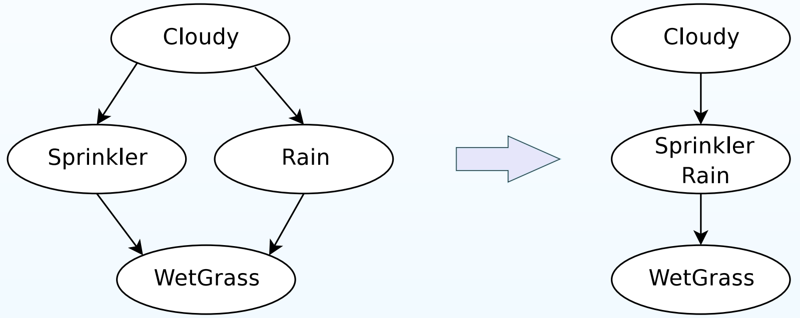
This technique can be generalized to *polytrees*:



Inference in Bayes Nets: Message Passing with Cycles

The Local Message Passing algorithm can deal also with cycles:

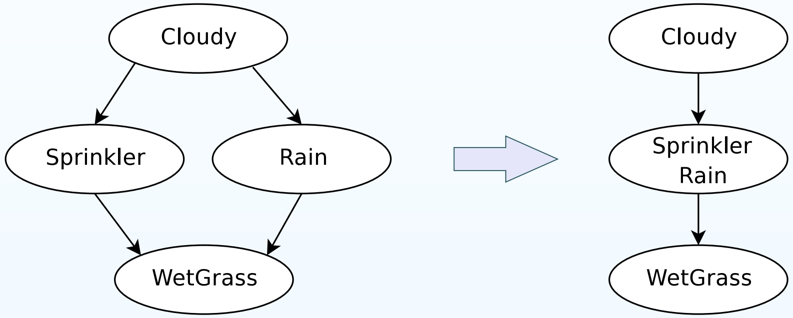
- Clustering variables together:



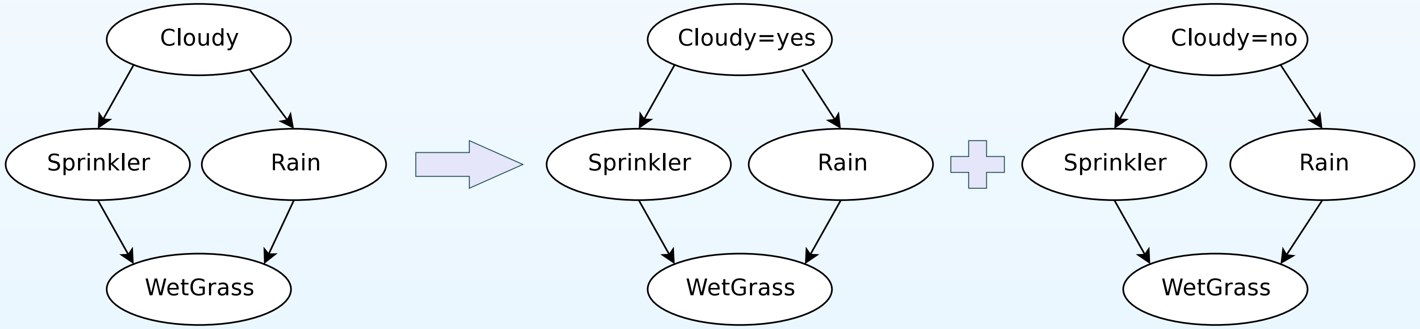
Inference in Bayes Nets: Message Passing with Cycles

The Local Message Passing algorithm can deal also with cycles:

- Clustering variables together:



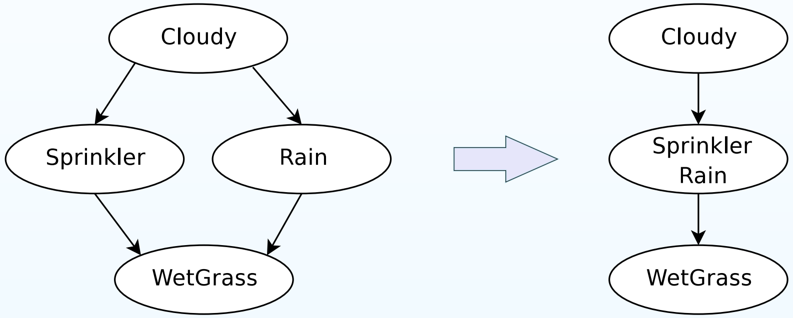
- Conditioning:



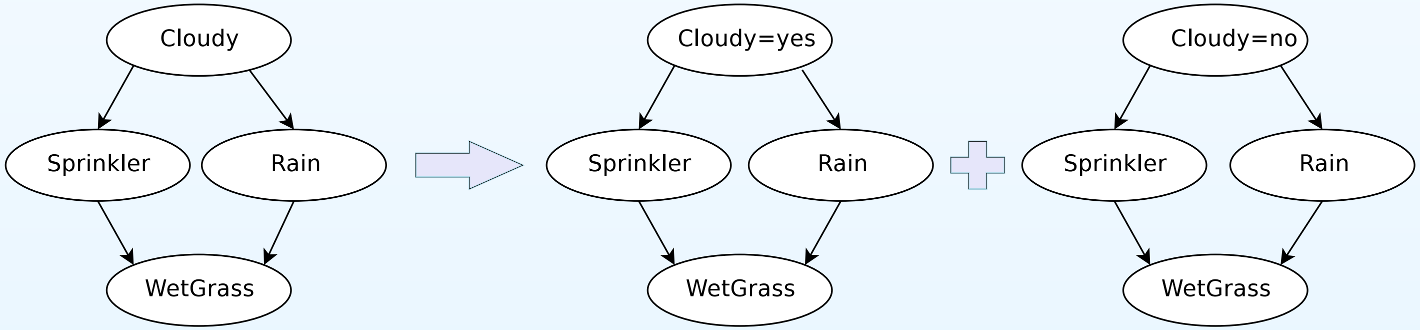
Inference in Bayes Nets: Message Passing with Cycles

The Local Message Passing algorithm can deal also with cycles:

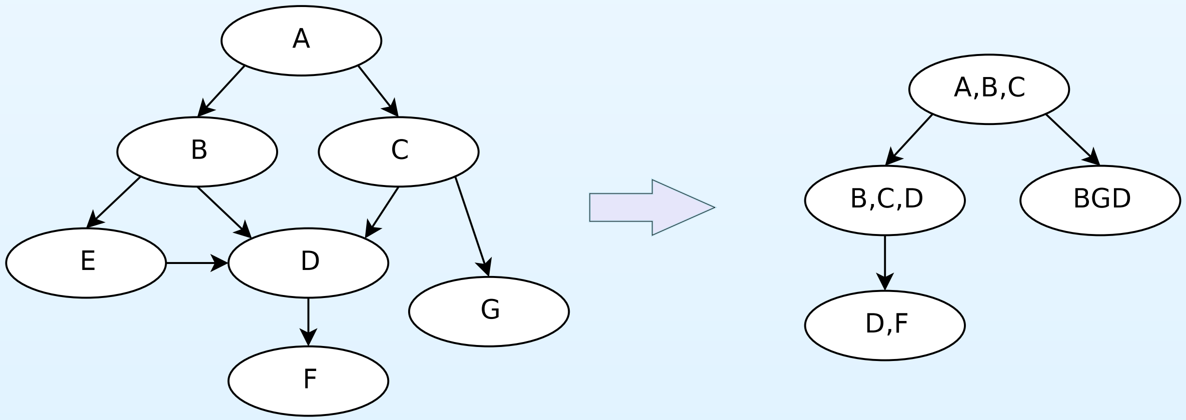
- Clustering variables together:



- Conditioning:



- Join Tree:



Inference in Bayes Nets: Simulation & Sampling (I)

We can sample from the Bayesian Network a set of assignments with the same probability as the underlying joint distribution:

Inference in Bayes Nets: Simulation & Sampling (I)

We can sample from the Bayesian Network a set of assignments with the same probability as the underlying joint distribution:

1. Randomly choose a sample c, $c = \texttt{true}$ with prob $P(C)$

Inference in Bayes Nets: Simulation & Sampling (I)

We can sample from the Bayesian Network a set of assignments with the same probability as the underlying joint distribution:

1. Randomly choose a sample c, $c = \texttt{true}$ with prob $P(C)$
2. Randomly choose a sample s, $s = \texttt{true}$ with prob $P(S|c)$

Inference in Bayes Nets: Simulation & Sampling (I)

We can sample from the Bayesian Network a set of assignments with the same probability as the underlying joint distribution:

1. Randomly choose a sample c, $c = \texttt{true}$ with prob $P(C)$
2. Randomly choose a sample s, $s = \texttt{true}$ with prob $P(S|c)$
3. Randomly choose a sample r, $r = \texttt{true}$ with prob $P(R|c)$

Inference in Bayes Nets: Simulation & Sampling (I)

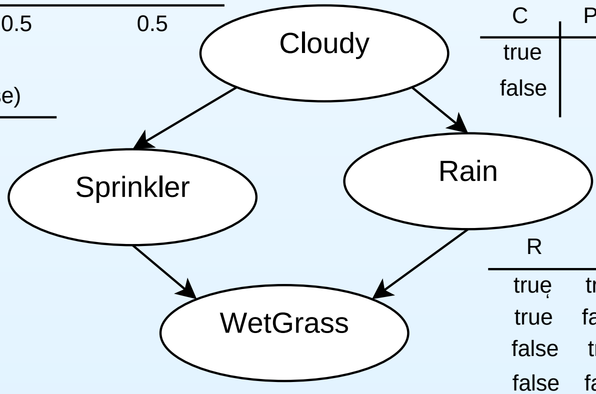
We can sample from the Bayesian Network a set of assignments with the same probability as the underlying joint distribution:

1. Randomly choose a sample c, $c = \texttt{true}$ with prob $P(C)$
2. Randomly choose a sample s, $s = \texttt{true}$ with prob $P(S|c)$
3. Randomly choose a sample r, $r = \texttt{true}$ with prob $P(R|c)$
4. Randomly choose a sample w, $w = \texttt{true}$ with prob $P(W|s, r)$

The sample c, s, r, w is a sample from the joint distribution of C, S, R, W.

P(C=true)	P(C=false)
0.5	0.5

C	P(S=true)	P(S=false)
true	0.5	0.5
false	0.9	0.1



C	P(R=true)	P(R=false)
true	0.8	0.2
false	0.2	0.8

R	S	P(W=true)	P(W=false)
true	true	0.99	0.01
true	false	0.9	0.1
false	true	0.9	0.1
false	false	0	1

Inference in Bayes Nets: Simulation & Sampling (II)

Suppose we are interested in knowing $P(E_1|E_2)$, now we have a simple mechanism to estimate this:

Inference in Bayes Nets: Simulation & Sampling (II)

Suppose we are interested in knowing $P(E_1|E_2)$, now we have a simple mechanism to estimate this:

- Take a lot of random samples from the joint distributiona and count:
  - N_c: the number of sample in which E_2 is verified.
  - N_s: the number of sample in which both E_1 and E_2 are verified.
  - N:the total number of samples.

Inference in Bayes Nets: Simulation & Sampling (II)

Suppose we are interested in knowing $P(E_1|E_2)$, now we have a simple mechanism to estimate this:

- Take a lot of random samples from the joint distributiona and count:
  - N_c: the number of sample in which E_2 is verified.
  - N_s: the number of sample in which both E_1 and E_2 are verified.
  - N:the total number of samples.
- When N is big enough we have:
  - $P(E_2) \approx N_c/N$
  - $P(E_1, E_2) \approx N_s/N$
  - $P(E_1|E_2) = P(E_1, E_2)/P(E_2) \approx N_s/N_c$

Inference in Bayes Nets: Simulation & Sampling (II)

Suppose we are interested in knowing $P(E_1|E_2)$, now we have a simple mechanism to estimate this:

- Take a lot of random samples from the joint distributiona and count:
  - N_c: the number of sample in which E_2 is verified.
  - N_s: the number of sample in which both E_1 and E_2 are verified.
  - N:the total number of samples.
- When N is big enough we have:
  - $P(E_2) \approx N_c/N$
  - $P(E_1, E_2) \approx N_s/N$
  - $P(E_1|E_2) = P(E_1, E_2)/P(E_2) \approx N_s/N_c$

With lots of constraints or unlikely events in E the most of the simulation thrown away (no effect on N_c and N_s).

Inference in Bayes Nets: Simulation & Sampling (II)

Suppose we are interested in knowing $P(E_1|E_2)$, now we have a simple mechanism to estimate this:

- Take a lot of random samples from the joint distributiona and count:
  - N_c: the number of sample in which E_2 is verified.
  - N_s: the number of sample in which both E_1 and E_2 are verified.
  - N:the total number of samples.
- When N is big enough we have:
  - $P(E_2) \approx N_c/N$
  - $P(E_1, E_2) \approx N_s/N$
  - $P(E_1|E_2) = P(E_1, E_2)/P(E_2) \approx N_s/N_c$

With lots of constraints or unlikely events in E the most of the simulation thrown away (no effect on N_c and N_s).

We should use Likelihood Sampling!

Inference in Bayes Nets: Simulation & Sampling (III)

We can exploit a simple idea to improve our sampling strategy

- Suppose in E_2 we have the constraint $X_i = v$
- We are about generating a random value with $P(X_i = v|parents) = w$;
- Generate always $X_i = v$ but weight the final answer by w.

Inference in Bayes Nets: Simulation & Sampling (III)

We can exploit a simple idea to improve our sampling strategy

- Suppose in E_2 we have the constraint $X_i = v$
- We are about generating a random value with $P(X_i = v|parents) = w$;
- Generate always $X_i = v$ but weight the final answer by w.

This turns into the following algorithm (initalize $N_c = 0$ and $N_s = 0$):

1. Generate a random assignment for all variables matching E_2;
2. Define w to be the probability that this assignment would have been generated instead of an unmatching assignment (w is the product of all likelihood factors involved in its generation);
3. $N_c = N_c + w$;
4. if our sample matches E_1 the $N_s = N_s + w$;
5. Go to 1.

Inference in Bayes Nets: Simulation & Sampling (III)

We can exploit a simple idea to improve our sampling strategy

- Suppose in E_2 we have the constraint $X_i = v$
- We are about generating a random value with $P(X_i = v|parents) = w$;
- Generate always $X_i = v$ but weight the final answer by w.

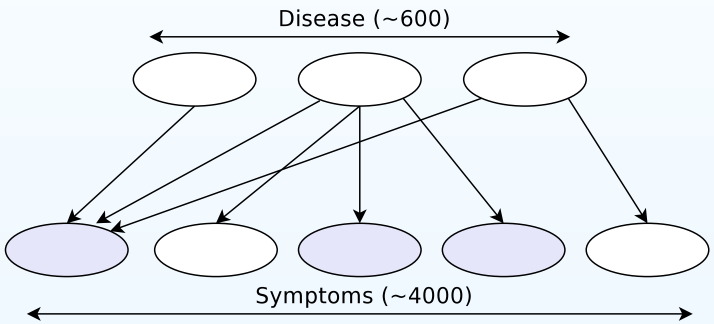
This turns into the following algorithm (initalize $N_c = 0$ and $N_s = 0$):

1. Generate a random assignment for all variables matching E_2;
2. Define w to be the probability that this assignment would have been generated instead of an unmatching assignment (w is the product of all likelihood factors involved in its generation);
3. $N_c = N_c + w$;
4. if our sample matches E_1 the $N_s = N_s + w$;
5. Go to 1.

Again the ratio N_s/N_c estimates our query $P(E_1|E_2)$

Bayesian Networks Applications

They originally arose to add probabilities in expert systems; a famous example is the reformulation of the Quick Medical Reference model.

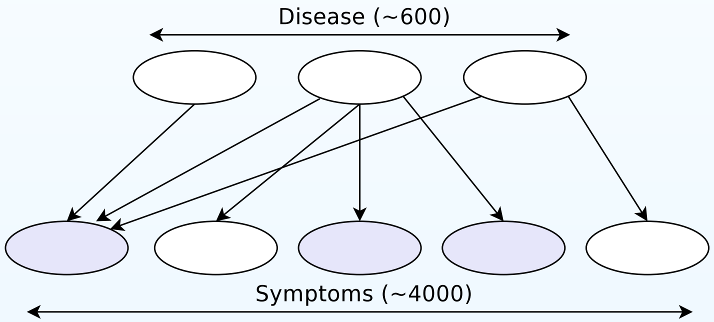


- Top layer represents hidden disease;
- Bottom layer represents observed symptoms;
- QMR-DT is so densely connected that exact inference is impossible.

The goal is to infer the posterior probability of each disease given all the symptoms (which can be present, absent or unknown).

Bayesian Networks Applications

They originally arose to add probabilities in expert systems; a famous example is the reformulation of the Quick Medical Reference model.

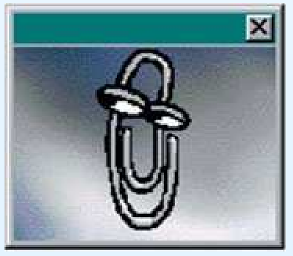


- Top layer represents hidden disease;
- Bottom layer represents observed symptoms;
- QMR-DT is so densely connected that exact inference is impossible.

The goal is to infer the posterior probability of each disease given all the symptoms (which can be present, absent or unknown).

The most widely used Bayes Nets are embedded in Microsoft's products:

- Answer Wizard in Office 95;
- Office Assistant in Office 97;
- Over 30 Technical Support Troubleshooters.



Check the Economist article (22/3/01) about Microsoft's application of BNs.

Learning Bayesian Networks

In order to define a Bayesian Network we need to specify:

- The graph topology (structure)
- The parameters of each Conditional Probability Densisty.

Learning Bayesian Networks

In order to define a Bayesian Network we need to specify:

- The graph topology (structure)
- The parameters of each Conditional Probability Densisty.

We can specify both of them with the help of experts or it is possible to learn both of these from data; however remember that:

- Learning the structure is much harder than learning parameters
- Learning when some of the nodes are hidden, or we have missing data, is much harder than when everything is observed

Learning Bayesian Networks

In order to define a Bayesian Network we need to specify:

- The graph topology (structure)
- The parameters of each Conditional Probability Densisty.

We can specify both of them with the help of experts or it is possible to learn both of these from data; however remember that:

- Learning the structure is much harder than learning parameters
- Learning when some of the nodes are hidden, or we have missing data, is much harder than when everything is observed

This gives rise to 4 approaches:

Structure/Oservability	Full	Partial
Known	Maximum Likelihood Estimation	EM (or gradient ascent)
Unknown	Search through model space	EM + search through model space

Learning: Known Structure & Full Observability

Learning has to find the values of the parameters of each Coditional Probability Distribution which maximizes the likelihood of the training data:

$$\mathcal{L} = \sum_{i=1}^{m} \sum_{r=1}^{R} \log P(X_i | Pa(X_i), \mathcal{D}_r)$$

Log-likelihood decomposes according to the structure of the graph; we can maximize the contribution of each node independently.

Learning: Known Structure & Full Observability

Learning has to find the values of the parameters of each Coditional Probability Distribution which maximizes the likelihood of the training data:

$$\mathcal{L} = \sum_{i=1}^{m} \sum_{r=1}^{R} \log P(X_i | Pa(X_i), \mathcal{D}_r)$$

Log-likelihood decomposes according to the structure of the graph; we can maximize the contribution of each node independently.

- Sparse data problems can be solved by using (mixtures of) Dirichlet priors (pseudo counts), Wishart prior with Gaussians, etc.

Learning: Known Structure & Full Observability

Learning has to find the values of the parameters of each Coditional Probability Distribution which maximizes the likelihood of the training data:

$$\mathcal{L} = \sum_{i=1}^{m} \sum_{r=1}^{R} \log P(X_i | Pa(X_i), \mathcal{D}_r)$$

Log-likelihood decomposes according to the structure of the graph; we can maximize the contribution of each node independently.

- Sparse data problems can be solved by using (mixtures of) Dirichlet priors (pseudo counts), Wishart prior with Gaussians, etc.
- For Gaussian nodes, we can compute the sample mean and variance, and use linear regression to estimate the weight matrix;

Learning: Known Structure & Full Observability

Learning has to find the values of the parameters of each Coditional Probability Distribution which maximizes the likelihood of the training data:

$$\mathcal{L} = \sum_{i=1}^{m} \sum_{r=1}^{R} \log P(X_i | Pa(X_i), \mathcal{D}_r)$$

Log-likelihood decomposes according to the structure of the graph; we can maximize the contribution of each node independently.

- Sparse data problems can be solved by using (mixtures of) Dirichlet priors (pseudo counts), Wishart prior with Gaussians, etc.
- For Gaussian nodes, we can compute the sample mean and variance, and use linear regression to estimate the weight matrix;

Example: For the $WetGrass$ node, from a set of training data, we can just count the number of times the “grass is wet” when it is “raining” and the “sprinler” is on, $N(W = 1, S = 1, R = 1)$, and so on:

$$P(W|S, R) = \frac{N(W,S,R)}{N(S,R)} = \frac{N(W,S,R)}{N(\overline{W},S,R)+N(W,S,R)}$$

Learning: Known Structure & Partial Observability

When some of the nodes are hidden, we can use the *Expectation Maximization* (EM) algorithm to find a (locally) optimal estimate:

- E-Step: compute expected values using an inference algorithm, and then treat these expected values as observed;

Learning: Known Structure & Partial Observability

When some of the nodes are hidden, we can use the *Expectation Maximization* (EM) algorithm to find a (locally) optimal estimate:

- E-Step: compute expected values using an inference algorithm, and then treat these expected values as observed;
- M-Step: consider the model as fully observable and apply the previous algorithm.

Given the expected counts, maximize parameters, and recompute the expected counts iteratively. EM converges to a likelihood local maximum.

Learning: Known Structure & Partial Observability

When some of the nodes are hidden, we can use the *Expectation Maximization* (EM) algorithm to find a (locally) optimal estimate:

- E-Step: compute expected values using an inference algorithm, and then treat these expected values as observed;
- M-Step: consider the model as fully observable and apply the previous algorithm.

Given the expected counts, maximize parameters, and recompute the expected counts iteratively. EM converges to a likelihood local maximum.

Inference becomes a subroutine called by learning: <u>it should be fast!</u>

Learning: Known Structure & Partial Observability

When some of the nodes are hidden, we can use the *Expectation Maximization* (EM) algorithm to find a (locally) optimal estimate:

- E-Step: compute expected values using an inference algorithm, and then treat these expected values as observed;
- M-Step: consider the model as fully observable and apply the previous algorithm.

Given the expected counts, maximize parameters, and recompute the expected counts iteratively. EM converges to a likelihood local maximum.

Inference becomes a subroutine called by learning: it should be fast!

Example: in the case of *WetGrass* node, we replace the observed counts of the events with the number of times we expect to see each event:

$$P(W|S,R) = \frac{E[N(W,S,R)]}{E[N(S,R)]}$$

where $E[N(x)]$ is the expected number of times x occurs in the training set, given the current guess of the parameters: $E[N(.)] = \sum_k P(.|\mathcal{D}_k)$.

Learning: Unknown Structure & Full Observability (I)

The maximum likelihood model G_{MLE} will be a complete graph:

- It has the largest number of parameters and can fit the data the best
- This is a joint distribution, it will overfit for sure!

Learning: Unknown Structure & Full Observability (I)

The maximum likelihood model G_{MLE} will be a complete graph:

- It has the largest number of parameters and can fit the data the best
- This is a joint distribution, it will overfit for sure!

To avoid overfitting we can use MAP:

$$P(G|\mathcal{D}) = \frac{P(\mathcal{D}|G)P(G)}{P(\mathcal{D})}$$

Learning: Unknown Structure & Full Observability (I)

The maximum likelihood model G_{MLE} will be a complete graph:

- It has the largest number of parameters and can fit the data the best
- This is a joint distribution, it will overfit for sure!

To avoid overfitting we can use MAP:

$$P(G|\mathcal{D}) = \frac{P(\mathcal{D}|G)P(G)}{P(\mathcal{D})}$$

taking logs, we find:

$$\log P(G|D) = \log P(\mathcal{D}|G) + \log P(G) + c.$$

Learning: Unknown Structure & Full Observability (I)

The maximum likelihood model G_{MLE} will be a complete graph:

- It has the largest number of parameters and can fit the data the best
- This is a joint distribution, it will overfit for sure!

To avoid overfitting we can use MAP:

$$P(G|\mathcal{D}) = \frac{P(\mathcal{D}|G)P(G)}{P(\mathcal{D})}$$

taking logs, we find:

$$\log P(G|D) = \log P(\mathcal{D}|G) + \log P(G) + c.$$

Last term $c = -\log P(\mathcal{D})$ does not depend on G. We could use structure prior $P(G)$ to penalizes overly complex models, however, this is not necessary since the marginal likelihood term

$$P(\mathcal{D}|G) = \int_\theta P(\mathcal{D}|G, \theta)$$

has already a similar effect; it embodies the Bayesian Occam's razor.

Learning: Unknown Structure & Full Observability (II)

The goal of structure learning is to learn a DAG that best explains the data:

- it is an NP-hard problem, since the number of DAGs on M variables is super-exponential in M
  - there are 543 DAGs on 4 nodes;
  - there are $O(10^{18})$ DAGSs on 10 nodes.

Learning: Unknown Structure & Full Observability (II)

The goal of structure learning is to learn a DAG that best explains the data:

- it is an NP-hard problem, since the number of DAGs on M variables is super-exponential in M
  - there are 543 DAGs on 4 nodes;
  - there are $O(10^{18})$ DAGSs on 10 nodes.
- if we know the ordering of the nodes, we can learn the parent set for each node independently
  - there are at most $\sum_{k=0}^{M} \binom{M}{k} = 2^M$ sets of possible parents.

Learning: Unknown Structure & Full Observability (II)

The goal of structure learning is to learn a DAG that best explains the data:

- it is an NP-hard problem, since the number of DAGs on M variables is super-exponential in M
  - there are 543 DAGs on 4 nodes;
  - there are $O(10^{18})$ DAGSs on 10 nodes.
- if we know the ordering of the nodes, we can learn the parent set for each node independently
  - there are at most $\sum_{k=0}^{M} \binom{M}{k} = 2^M$ sets of possible parents.

We can start with an initial guess of the model structure, and then perform local search, evaluating the score of neighboring structures and move to the best one, until we reach a local optimum.

Learning: Unknown Structure & Full Observability (II)

The goal of structure learning is to learn a DAG that best explains the data:

- it is an NP-hard problem, since the number of DAGs on M variables is super-exponential in M
  - there are 543 DAGs on 4 nodes;
  - there are $O(10^{18})$ DAGSs on 10 nodes.
- if we know the ordering of the nodes, we can learn the parent set for each node independently
  - there are at most $\sum_{k=0}^{M} \binom{M}{k} = 2^M$ sets of possible parents.

We can start with an initial guess of the model structure, and then perform local search, evaluating the score of neighboring structures and move to the best one, until we reach a local optimum.

- use the Tabu Search algorithm;

Learning: Unknown Structure & Full Observability (II)

The goal of structure learning is to learn a DAG that best explains the data:

- it is an NP-hard problem, since the number of DAGs on M variables is super-exponential in M
  - there are 543 DAGs on 4 nodes;
  - there are $O(10^{18})$ DAGSs on 10 nodes.
- if we know the ordering of the nodes, we can learn the parent set for each node independently
  - there are at most $\sum_{k=0}^{M} \binom{M}{k} = 2^M$ sets of possible parents.

We can start with an initial guess of the model structure, and then perform local search, evaluating the score of neighboring structures and move to the best one, until we reach a local optimum.

- use the Tabu Search algorithm;
- use Genetic Algorithms to fing a global optimmum;

Learning: Unknown Structure & Full Observability (II)

The goal of structure learning is to learn a DAG that best explains the data:

- it is an NP-hard problem, since the number of DAGs on M variables is super-exponential in M
  - there are 543 DAGs on 4 nodes;
  - there are $O(10^{18})$ DAGSs on 10 nodes.
- if we know the ordering of the nodes, we can learn the parent set for each node independently
  - there are at most $\sum_{k=0}^{M} \binom{M}{k} = 2^M$ sets of possible parents.

We can start with an initial guess of the model structure, and then perform local search, evaluating the score of neighboring structures and move to the best one, until we reach a local optimum.

- use the Tabu Search algorithm;
- use Genetic Algorithms to fing a global optimmum;
- use multiple restarts to try to find the global optimum, and to learn an ensemble of models.

Learning: Unknown Structure & Partial Observability

Here come the tough part! We have that

- the structure is unknown;
- there are hidden variables and/or missing data.

Learning: Unknown Structure & Partial Observability

Here come the tough part! We have that

- the structure is unknown;
- there are hidden variables and/or missing data.

This is usually intractable; we can use an approximation of the posterior called Bayesian Information Criterion (BIC):

$$\log P(D|G) \approx \log P(D|G, \hat{\Theta}_G) - \frac{N}{2} \log R$$

where R is the number of samples, $\hat{\Theta}_G$ is the ML estimate of model parameters, and N is the dimension of the model:

Learning: Unknown Structure & Partial Observability

Here come the tough part! We have that

- the structure is unknown;
- there are hidden variables and/or missing data.

This is usually intractable; we can use an approximation of the posterior called Bayesian Information Criterion (BIC):

$$\log P(D|G) \approx \log P(D|G, \hat{\Theta}_G) - \frac{N}{2} \log R$$

where R is the number of samples, $\hat{\Theta}_G$ is the ML estimate of model parameters, and N is the dimension of the model:

- in the fully observable case, dimension of a model is the number of free parameters; in models with hidden variables, it might be less;

Learning: Unknown Structure & Partial Observability

Here come the tough part! We have that

- the structure is unknown;
- there are hidden variables and/or missing data.

This is usually intractable; we can use an approximation of the posterior called Bayesian Information Criterion (BIC):

$$\log P(D|G) \approx \log P(D|G, \hat{\Theta}_G) - \frac{N}{2} \log R$$

where R is the number of samples, $\hat{\Theta}_G$ is the ML estimate of model parameters, and N is the dimension of the model:

- in the fully observable case, dimension of a model is the number of free parameters; in models with hidden variables, it might be less;
- BIC score decomposes into a sum of local terms, but local search is still expensive, because we need to run EM at each step to compute $\hat{\Theta}_G$. We can do local search inside the M-Step (Structural EM).