

COMP9418: Advanced Topics in Statistical Machine Learning

Learning Bayesian Networks Structure
with Maximum Likelihood

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Instructor: Gustavo Batista

University of New South Wales

Introduction

- So far, we have assumed we know the structure of a Bayesian network
 - We were mostly concerned with estimating its parameters
 - The main approach for this estimation has been the search for ML estimates
 - That is, the ones that maximize the probability of observing a given dataset
- Now, we assume the network structure is unknown
 - We want to learn the structure from a given dataset
- We adopt the same approach for parameter estimation
 - That is, we search for network structures that maximize the probability of observing a given dataset
 - We start with this approach and show it leads to a general class of scoring functions for network structures
- We focus on learning structures from complete data
 - Dealing with incomplete data is similar but much more demanding computationally

Introduction

- Consider the Bayesian network on the right

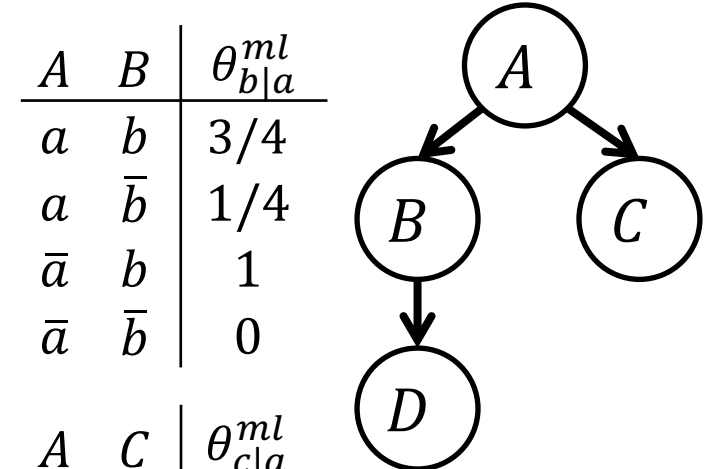
- Remind that in the previous lecture, we defined the log-likelihood of a structure G as

$$LL(G; \mathcal{D}) \stackrel{\text{def}}{=} \log L(\theta^{ml}, \mathcal{D})$$

- For the dataset \mathcal{D} , we can compute $LL(G; \mathcal{D})$ as

$$LL(G; \mathcal{D}) = \log \prod_{i=1}^5 P_{\theta^{ml}}(\mathbf{d}_i) = \sum_{i=1}^5 \log P_{\theta^{ml}}(\mathbf{d}_i)$$

\mathcal{D}	A	B	C	D	$P_{\theta^{ml}}(\mathbf{d}_i)$	$\log P_{\theta^{ml}}(\mathbf{d}_i)$
1	a	b	\bar{c}	d	$3/4 \times 3/4 \times 4/5 \times 1/4$.1125
2	a	b	\bar{c}	\bar{d}	$3/4 \times 3/4 \times 4/5 \times 3/4$.3375
3	a	\bar{b}	c	d	$1/4 \times 1/4 \times 4/5 \times 1$.05
4	\bar{a}	b	c	\bar{d}	$1 \times 1 \times 1/5 \times 3/4$.15
5	a	b	\bar{c}	\bar{d}	$3/4 \times 3/4 \times 4/5 \times 3/4$.3375
$LL(G; \mathcal{D}) =$						-13.35



A	B	$\theta_{b a}^{ml}$
a	b	$3/4$
a	\bar{b}	$1/4$
\bar{a}	b	1
\bar{a}	\bar{b}	0

A	C	$\theta_{c a}^{ml}$
a	c	$1/4$
a	\bar{c}	$3/4$
\bar{a}	c	1
\bar{a}	\bar{c}	0

A	θ_a^{ml}
a	$4/5$
\bar{a}	$1/5$

\mathcal{D}	A	B	C	D
1	a	b	\bar{c}	d
2	a	b	\bar{c}	\bar{d}
3	a	\bar{b}	c	d
4	\bar{a}	b	c	\bar{d}
5	a	b	\bar{c}	\bar{d}

B	D	$\theta_{b d}^{ml}$
b	d	$1/4$
b	\bar{d}	$3/4$
\bar{b}	d	1
\bar{b}	\bar{d}	0

Introduction

- Now, consider this alternate network structure G^* and ML estimates

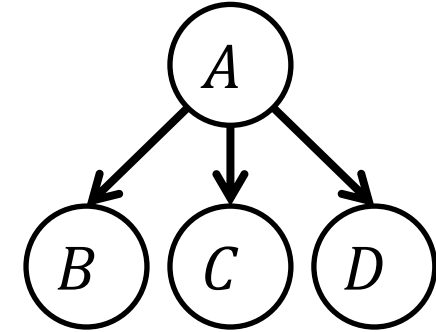
- Let us compute the log-likelihood of this network for the same dataset

Assignment Project Exam Help

\mathcal{D}	A	B	C	D	$P_{\theta^{ml}}(\mathbf{d}_i)$	$\log P_{\theta^{ml}}(\mathbf{d}_i)$
1	a	b	\bar{c}	d	$3/4 \times 3/4 \times 4/5 \times 1/2$.225
2	a	b	\bar{c}	\bar{d}	$3/4 \times 3/4 \times 4/5 \times 1/2$.225
3	a	\bar{b}	c	d	$1/4 \times 1/4 \times 4/5 \times 1/2$.025
4	\bar{a}	b	c	\bar{d}	$1 \times 1 \times 1/5 \times 1$.2
5	a	b	\bar{c}	\bar{d}	$3/4 \times 3/4 \times 4/5 \times 1/2$.225
$LL(G^*; \mathcal{D}) =$						-14.09

- As $LL(G^*; \mathcal{D}) < LL(G; \mathcal{D})$, we prefer G to G^*
 - Our goal is to search for a maximum likelihood structure

A	B	$\theta_{b a}^{ml}$
a	b	$3/4$
a	\bar{b}	$1/4$
\bar{a}	b	1
\bar{a}	\bar{b}	0



A	C	$\theta_{c a}^{ml}$
a	c	$1/4$
a	\bar{c}	$3/4$
\bar{a}	c	1
\bar{a}	\bar{c}	0

A	θ_a^{ml}
a	$4/5$
\bar{a}	$1/5$

\mathcal{D}	A	B	C	D
1	a	b	\bar{c}	d
2	a	b	\bar{c}	\bar{d}
3	a	\bar{b}	c	d
4	\bar{a}	b	c	\bar{d}
5	a	b	\bar{c}	\bar{d}

A	D	$\theta_{d a}^{ml}$
a	d	$1/2$
a	\bar{d}	$1/2$
\bar{a}	d	0
\bar{a}	\bar{d}	1

Learning Tree Structures

- We start exploring an algorithm based on a scoring measure for tree structures
 - Therefore, each node will have at most one parent
 - The scoring function is expressed in terms of mutual information
- Mutual information between variables X and U
 - It is a measure of dependence between these variables in a distribution
 - Our scoring measure is based on mutual information in the empirical distribution
- Given a tree structure G with edges $U \rightarrow X$
 - The scoring function is given by
 - The score is the addition of MI between every variable and its single parent in the tree structure

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

$$MI_{\mathcal{D}}(X, U) \stackrel{\text{def}}{=} \sum_{x, u} P_{\mathcal{D}}(x, u) \log \frac{P_{\mathcal{D}}(x, u)}{P_{\mathcal{D}}(x)P_{\mathcal{D}}(u)}$$

$$tScore(G; \mathcal{D}) \stackrel{\text{def}}{=} \sum_{U \rightarrow X} MI_{\mathcal{D}}(X, U)$$

Learning Tree Structures

- We can show that trees having a maximal likelihood are those trees that maximize *tScore*
 - That is, if G is a tree structure, and \mathcal{D} is a complete dataset, then
 $\text{argmax}_G tScore(G; \mathcal{D}) = \text{argmax}_G LL(G; \mathcal{D})$
- Therefore, we can find a maximum likelihood tree using an algorithm for computing maximum spanning trees
 - This is known as the Chow-Liu algorithm
 - We will discuss this algorithm shortly
 - But before that, let's understand the relationship between *tScore* and maximum log-likelihood

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

tScore and Maximum Log-likelihood

- Let us remind the definitions of
 - Entropy
 - Conditional entropy, and
 - Mutual information
- Also, remember from the previous lecture that the log-likelihood decomposes into components
 - Each component is a conditional entropy
- Expanding MI and substituting the definitions of entropy and conditional entropy leads to
 - $MI_{\mathcal{D}}(X, \mathbf{U}) = H_{\mathcal{D}}(X) - H_{\mathcal{D}}(X|\mathbf{U})$

$$H_{\mathcal{D}}(X) \stackrel{\text{def}}{=} - \sum_x P_{\mathcal{D}}(x) \log P_{\mathcal{D}}(x)$$

$$H_{\mathcal{D}}(X|\mathbf{U}) \stackrel{\text{def}}{=} - \sum_{x, \mathbf{u}} P_{\mathcal{D}}(x, \mathbf{u}) \log P_{\mathcal{D}}(x|\mathbf{u})$$

$$MI_{\mathcal{D}}(X, \mathbf{U}) \stackrel{\text{def}}{=} \sum_{x, \mathbf{u}} P_{\mathcal{D}}(x, \mathbf{u}) \log \frac{P_{\mathcal{D}}(x, \mathbf{u})}{P_{\mathcal{D}}(x)P_{\mathcal{D}}(\mathbf{u})}$$

$$LL(G; \mathcal{D}) = -N \sum_{x\mathbf{u}} H_{\mathcal{D}}(X|\mathbf{U})$$

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

tScore and Log ML

- Therefore,

$$LL(G; \mathcal{D}) = -N \sum_{XU} H_{\mathcal{D}}(X|U)$$

$$= -N \sum_{XU} (H_{\mathcal{D}}(X) - MI_{\mathcal{D}}(X, U))$$

$$= -N \sum_{XU} H_{\mathcal{D}}(X) + N \sum_{XU} MI_{\mathcal{D}}(X, U)$$

WeChat: cstutorcs

- Note that neither N nor the term $-N \sum_{XU} H_{\mathcal{D}}(X)$ depend on the tree structure G . Hence,

$$\operatorname{argmax}_G LL(G; \mathcal{D}) = \operatorname{argmax}_G \sum_{XU} MI_{\mathcal{D}}(X, U) = \operatorname{argmax}_G tScore(G; \mathcal{D})$$

$$H_{\mathcal{D}}(X) \stackrel{\text{def}}{=} - \sum_x P_{\mathcal{D}}(x) \log P_{\mathcal{D}}(x)$$

$$H_{\mathcal{D}}(X|U) \stackrel{\text{def}}{=} - \sum_{x,u} P_{\mathcal{D}}(x, u) \log P_{\mathcal{D}}(x|u)$$

$$MI_{\mathcal{D}}(X, U) \stackrel{\text{def}}{=} \sum_{x,u} P_{\mathcal{D}}(x, u) \log \frac{P_{\mathcal{D}}(x, u)}{P_{\mathcal{D}}(x)P_{\mathcal{D}}(u)}$$

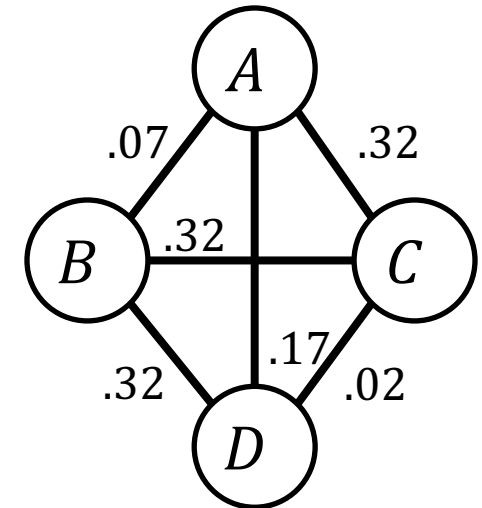
$$LL(G; \mathcal{D}) = -N \sum_{XU} H_{\mathcal{D}}(X|U)$$

$$MI_{\mathcal{D}}(X, U) = H_{\mathcal{D}}(X) - H_{\mathcal{D}}(X|U)$$

Learning Tree Structures

- Let us return to the algorithm for learning ML tree structures using *tScore*
 - We illustrate this algorithm for variables A, B, C and D and the dataset on the right
- The first step is constructing a complete, undirected graph over all variables
 - The cost of each edge is the mutual information between the two nodes connected by the edge
 - Let us detail the computation for edge $A - B$

\mathcal{D}	A	B	C	D
1	a	b	\bar{c}	d
2	a	b	\bar{c}	\bar{d}
3	a	\bar{b}	c	d
4	\bar{a}	b	c	\bar{d}
5	a	b	\bar{c}	\bar{d}

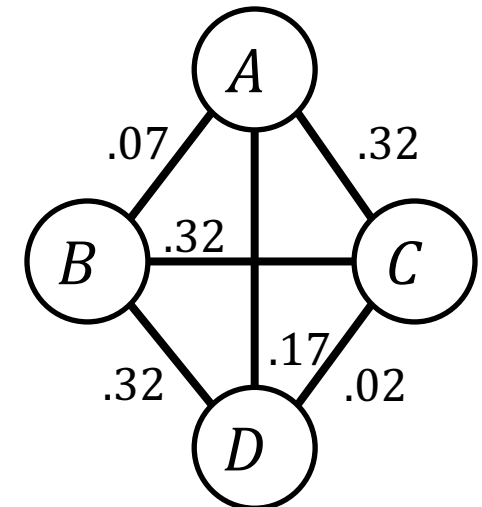


Learning Tree Structures

$$\begin{aligned}
 MI_{\mathcal{D}}(X, U) &\stackrel{\text{def}}{=} \sum_{x,u} P_{\mathcal{D}}(x, u) \log \frac{P_{\mathcal{D}}(x, u)}{P_{\mathcal{D}}(x)P_{\mathcal{D}}(u)} \\
 &= P_{\mathcal{D}}(a, b) \log \frac{P_{\mathcal{D}}(a, b)}{P_{\mathcal{D}}(a)P_{\mathcal{D}}(b)} + P_{\mathcal{D}}(a, \bar{b}) \log \frac{P_{\mathcal{D}}(a, \bar{b})}{P_{\mathcal{D}}(a)P_{\mathcal{D}}(\bar{b})} \\
 &= P_{\mathcal{D}}(\bar{a}, b) \log \frac{P_{\mathcal{D}}(\bar{a}, b)}{P_{\mathcal{D}}(\bar{a})P_{\mathcal{D}}(b)} + P_{\mathcal{D}}(\bar{a}, \bar{b}) \log \frac{P_{\mathcal{D}}(\bar{a}, \bar{b})}{P_{\mathcal{D}}(\bar{a})P_{\mathcal{D}}(\bar{b})}
 \end{aligned}$$

\mathcal{D}	A	B	C	D
1	a	b	\bar{c}	d
2	a	b	\bar{c}	\bar{d}
3	a	\bar{b}	c	d
4	\bar{a}	b	c	\bar{d}
5	a	b	\bar{c}	\bar{d}

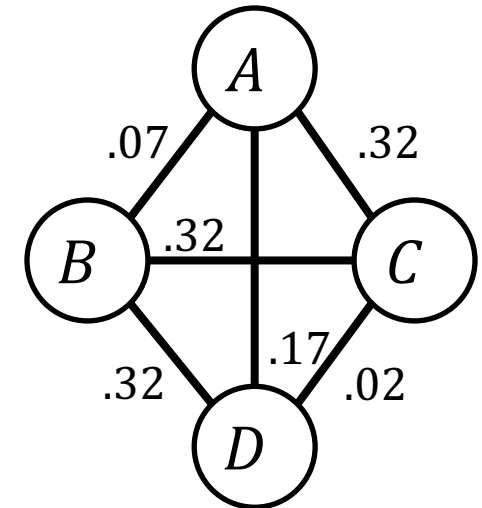
A	B	$P(A, B)$	$P(A)$	$P(B)$	$R(A, B)$	$P \log R(A, B)$
a	b	$3/5$	$4/5$	$4/5$.94	-.09
a	\bar{b}	$1/5$	$4/5$	$1/5$	1.25	.32
\bar{a}	b	$1/5$	$1/5$	$4/5$	1.25	.32
\bar{a}	\bar{b}	$0/5$	$1/5$	$1/5$	0	$-\infty$
$MI_{\mathcal{D}}(A, B) =$.074



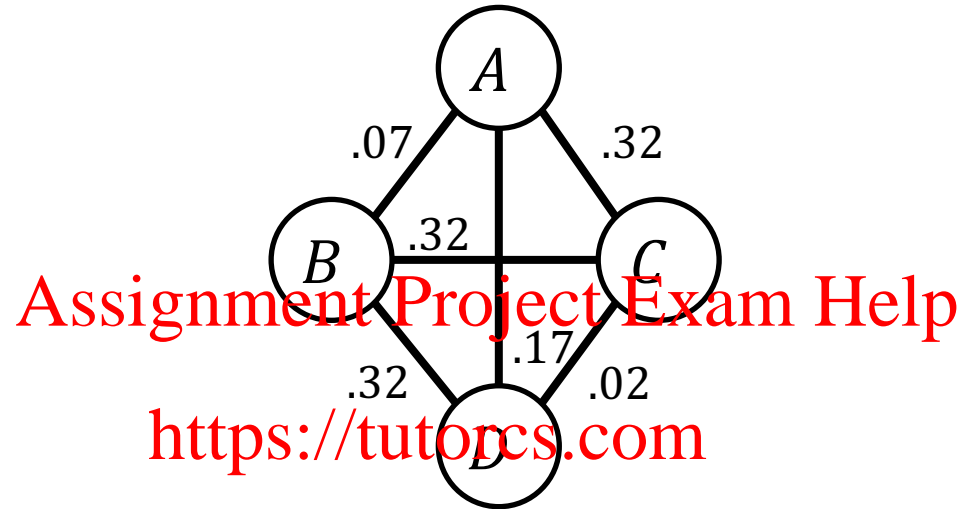
Learning Tree Structures

- The first step is constructing a complete, undirected graph over all variables
 - The cost of each edge is the mutual information between the two nodes connected by the edge
 - We compute the spanning tree with maximal cost
 - The cost of the tree is just the sum of costs associated with its edges
 - The algorithm for maximum spanning tree is a trivial adaptation of algorithms for minimum spanning tree such as Prim and Kruskal
- This method generates an undirected spanning tree
 - It coincides with several directed trees
 - We can choose any of these trees by selecting a node as root and directing edges away from the root
- The resulting trees are guaranteed to have a maximal likelihood among all tree structures

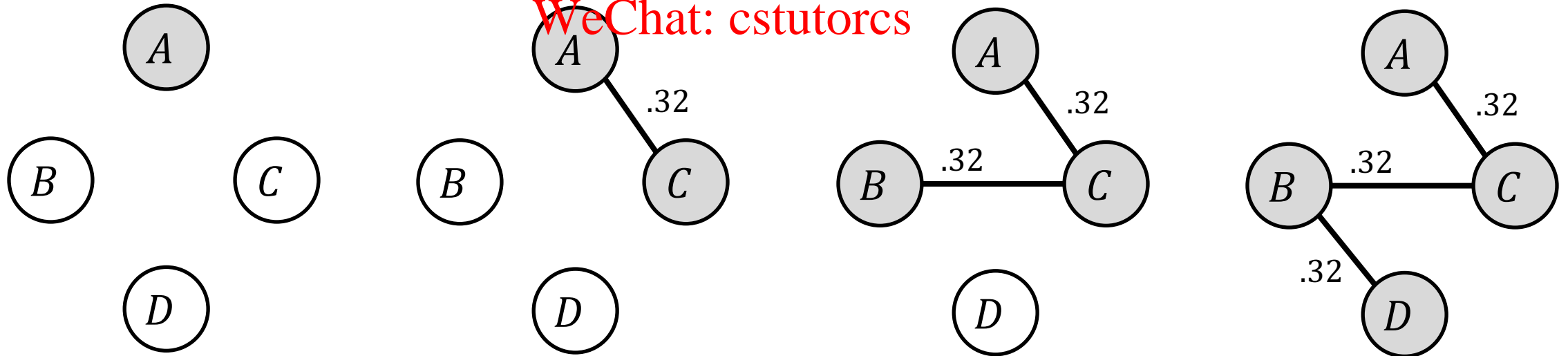
\mathcal{D}	A	B	C	D
1	a	b	\bar{c}	d
2	a	b	\bar{c}	\bar{d}
3	a	\bar{b}	c	d
4	\bar{a}	b	c	\bar{d}
5	a	b	\bar{c}	\bar{d}



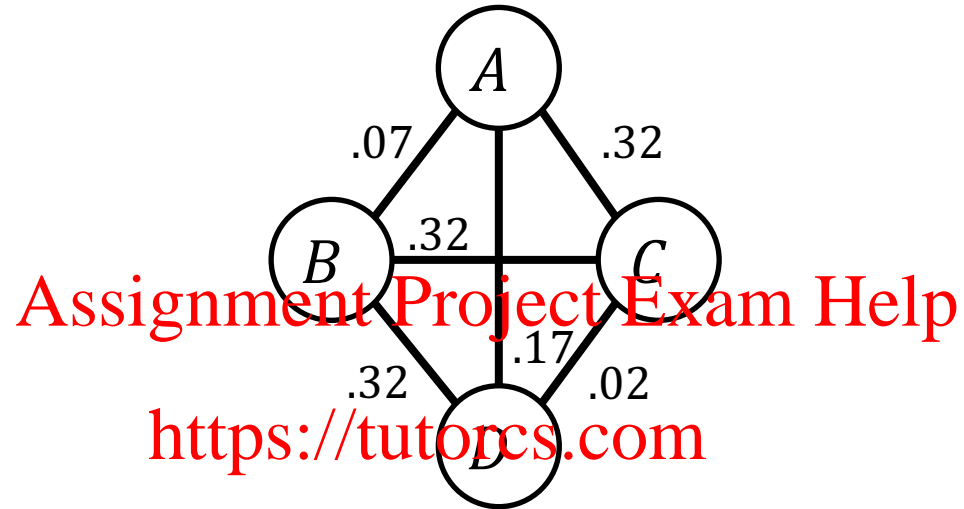
Learning Tree Structures: Prim



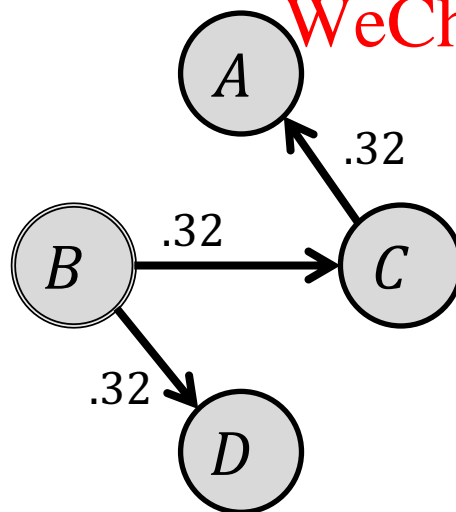
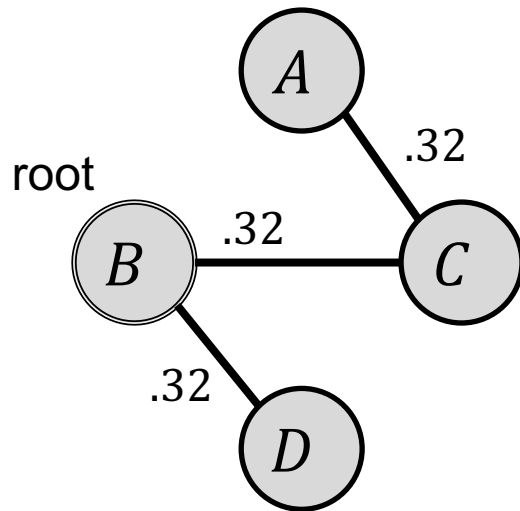
WeChat: cstutorcs



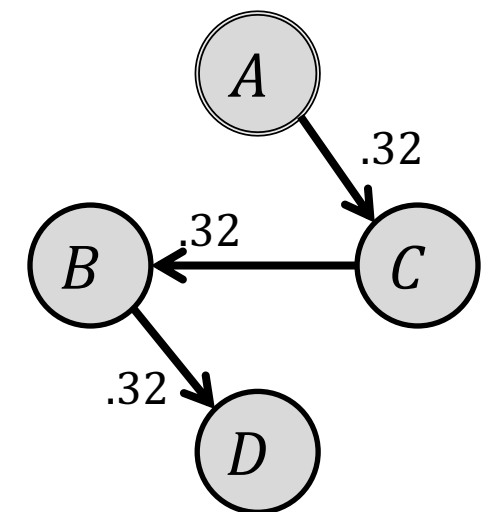
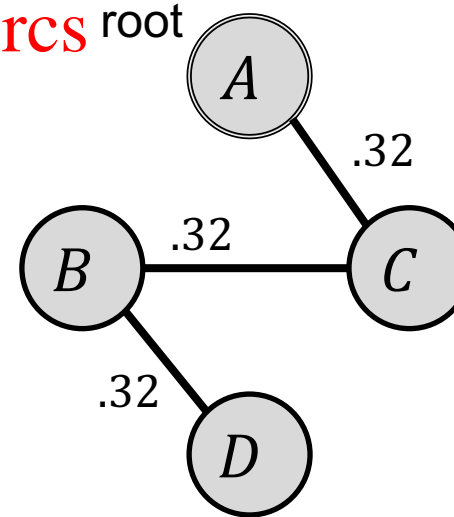
Learning Tree Structures: Prim



WeChat: cstutorcs



Log-likelihood: -12.1



Log-likelihood: -12.1

Learning Tree Structures

- Given a tree, we can compute the log-likelihood by
 - Computing the probability of each case in the dataset, as we did in slide 7
 - Or use the following equation that shows the log-likelihood corresponds to a sum of terms, one for each family in the network

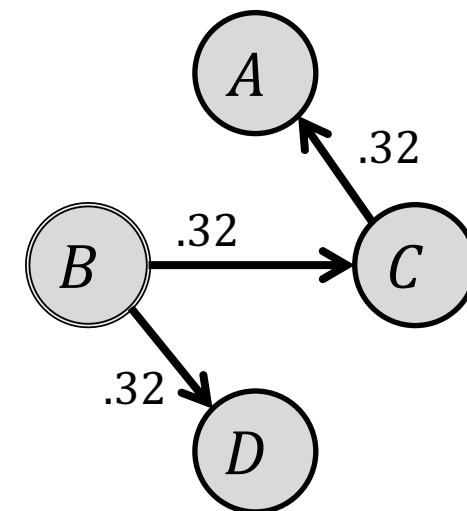
$$LL(G; \mathcal{D}) = -N \sum_{XU} H_{\mathcal{D}}(X|U)$$

- For this network, we have

$$\begin{aligned} LL(G; \mathcal{D}) &= -N(H_{\mathcal{D}}(A|C) + H_{\mathcal{D}}(B) + H_{\mathcal{D}}(C|B) + H_{\mathcal{D}}(D|B)) \\ &= -5(.400 + .722 + .649 + .649) \\ &= -12.1 \end{aligned}$$

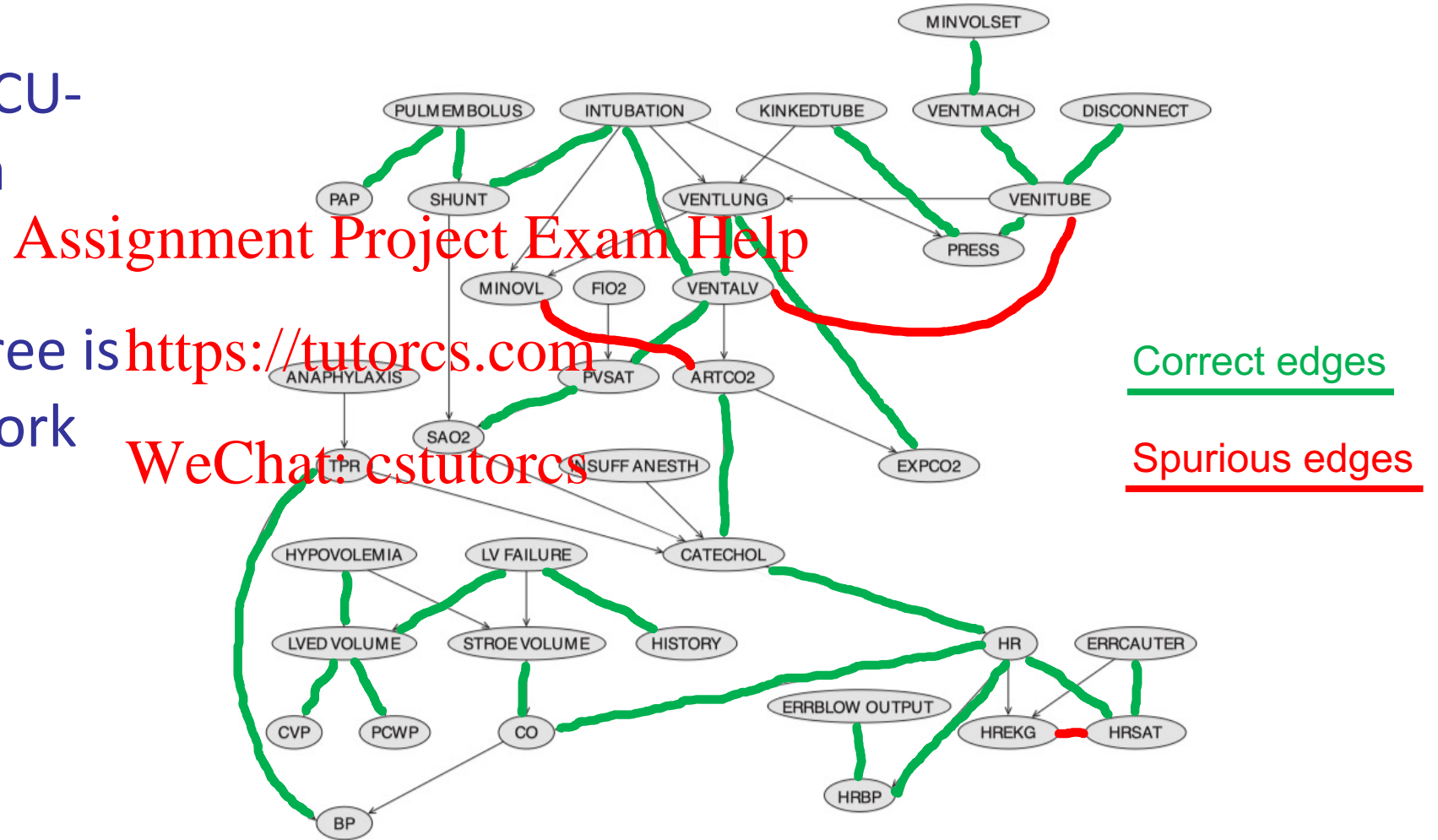
- Notice the terms correspond to families of the tree structure
 - $AC, B, CB,$ and DB

\mathcal{D}	A	B	C	D
1	a	b	\bar{c}	d
2	a	b	\bar{c}	\bar{d}
3	a	\bar{b}	c	d
4	\bar{a}	b	c	\bar{d}
5	a	b	\bar{c}	\bar{d}



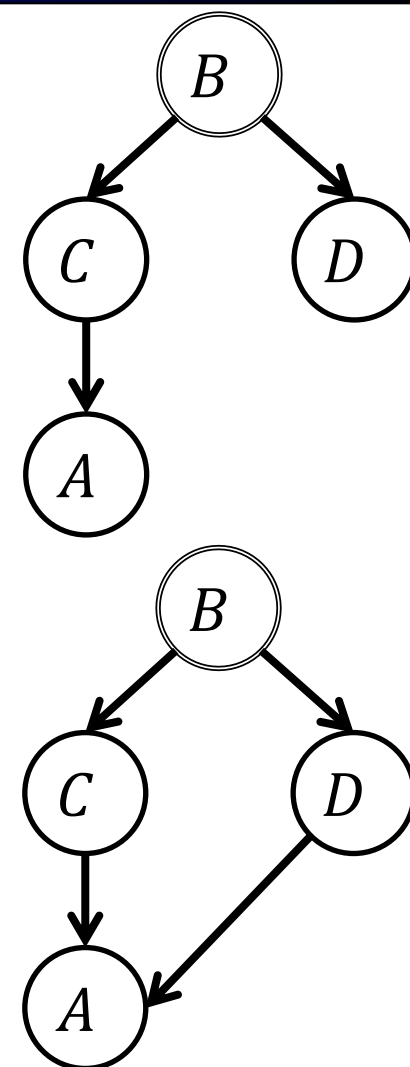
Learning Tree Structures: Example

- Tree learned from ICU-Alarm network data
- Not every edge in tree is in the original network
- Inferred edges are undirected



Learning DAG Structures

- Suppose now our goal is to find a maximum likelihood structure
 - Without restricting to tree structures
- For example, consider the tree structure obtained in the previous slide plus the edge $D \rightarrow A$
 - The log-likelihood of this DAG is
$$\begin{aligned} LL(G; \mathcal{D}) &= -N(H_{\mathcal{D}}(A|C, D) + H_{\mathcal{D}}(B) + H_{\mathcal{D}}(C|B) + H_{\mathcal{D}}(D|B)) \\ &= -5(0 + .722 + .649 + .649) \\ &= -10.1 \end{aligned}$$
 - Which is larger than the log-likelihood of the tree



Learning DAG Structures

- Notice the only difference between the two likelihoods is the entropy for variable A

- This is the only variable with different families in the two structures
- The family of A is AC in the tree and ACD in the DAG

$$H_{\mathcal{D}}(A|C, D) < H_{\mathcal{D}}(A|C)$$

- Hence

$$-H_{\mathcal{D}}(A|C, D) > -H_{\mathcal{D}}(A|C)$$

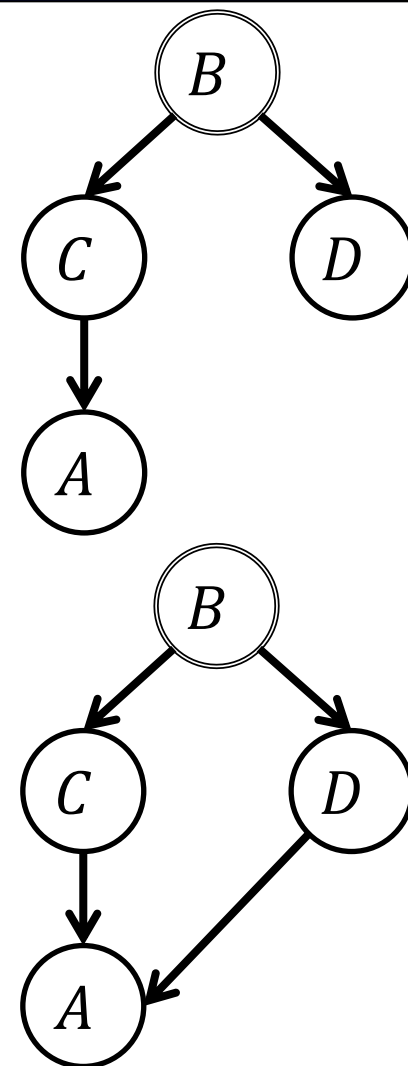
- Which is why the DAG has larger log-likelihood than the tree

- However, this result is not completely accidental

- We can show that if $\mathbf{U} \subseteq \mathbf{U}^*$, then $H_{\mathcal{D}}(X|\mathbf{U}) \geq H_{\mathcal{D}}(X|\mathbf{U}^*)$

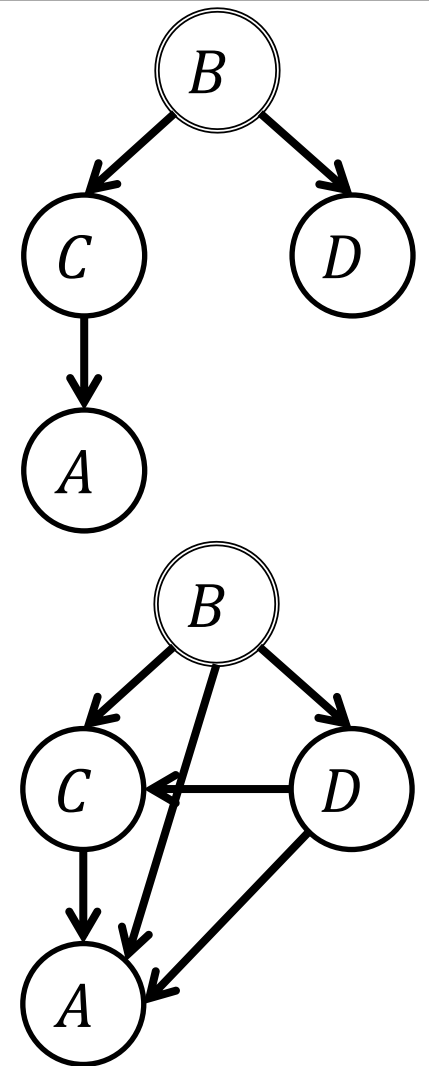
- Adding more parents to a variable never increases the entropy term

- Therefore, never decreases the log-likelihood of the resulting structure



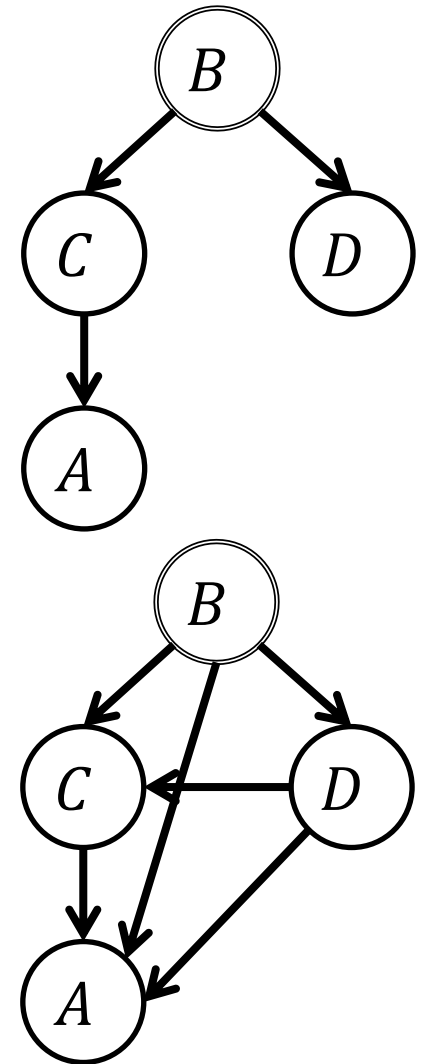
Learning DAG Structures and Overfitting

- Therefore, if a DAG G^* is the result of adding edges to a DAG G , then
$$LL(G^*; \mathcal{D}) \geq LL(G; \mathcal{D})$$
- If we simply search for a network structure with maximal likelihood
 - We end up choosing a complete network structure
 - That is, a DAG to which no more edges can be added without introducing cycles
- Complete DAGs are undesirable for several reasons
 - They make no assertions of conditional independence
 - Their topology does not reveal any properties of the distribution they induce
 - A complete DAG over n variables has treewidth of $n - 1$
 - Complete DAGs suffer of the problem of *overfitting*: the use of a model that has too many parameters compared to the available data



Overfitting

- In summary, the problem of overfitting occurs when
 - We focus on learning a model that fits the data well
 - Without constraining enough the number of free model parameters
- The result is that we end up adopting models that are more complex than necessary
 - Such models tend to have poor generalization performance
 - That is, they perform poorly on cases that are not part of the dataset
- There is no agreed upon solution for overfitting
 - However, the available solutions are based on the principle of *Occam's Razor*
 - Which states we should prefer simpler models, other things being equal



Model Complexity: DAG Dimension

- To realize Occam's razor principle, we need
 - A measure of model complexity
 - A method for balancing model complexity and data fit
- A common choice for model complexity is the number of independent parameters in the model
 - This gives us the following measure of complexity
- Let G be a DAG over variables X_1, \dots, X_N with corresponding parents $\mathbf{U}_1, \dots, \mathbf{U}_N$ and let $Y^\#$ denote the number of instantiations for variables Y
 - The *dimension of G* is defined as
 - Therefore, the dimension of a DAG is equal to the number of independent parameters in its CPTs

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

$$\|G\| \stackrel{\text{def}}{=} \sum_i^N \|X_i \mathbf{U}_i\|$$
$$\|X_i \mathbf{U}_i\| \stackrel{\text{def}}{=} (X_i^\# - 1) \mathbf{U}_i^\#$$

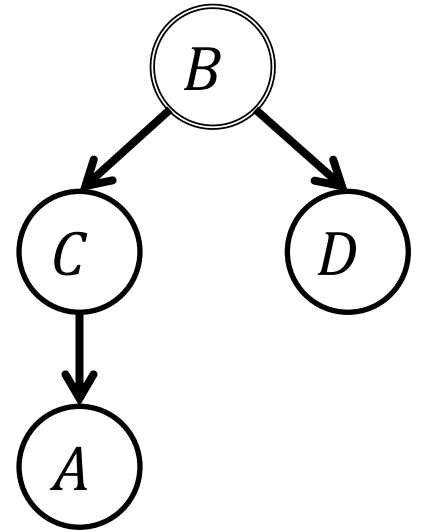
Model Complexity: DAG Dimension

- For example, the dimension of this DAG considering all binary variables is
 - $\|B\| = 1, \|CB\| = 2, \|DB\| = 2, \|AC\| = 2$. Total = 7

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs



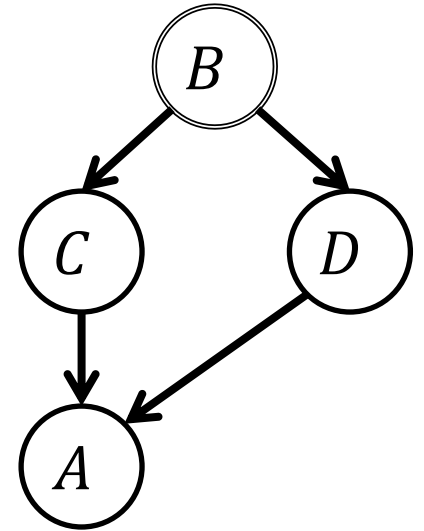
$$\|G\| \stackrel{\text{def}}{=} \sum_i^N \|X_i \mathbf{U}_i\|$$
$$\|X_i \mathbf{U}_i\| \stackrel{\text{def}}{=} (X_i^\# - 1) \mathbf{U}_i^\#$$

Model Complexity: DAG Dimension

- For example, the dimension of this DAG considering all binary variables is
 - $\|B\| = 1, \|CB\| = 2, \|DB\| = 2, \|AC\| = 2$. Total = 7
 - $\|B\| = 1, \|CB\| = 2, \|DB\| = 2, \|ACD\| = 4$. Total = 9

<https://tutorcs.com>

WeChat: cstutorcs



$$\|G\| \stackrel{\text{def}}{=} \sum_i^N \|X_i \mathbf{U}_i\|$$
$$\|X_i \mathbf{U}_i\| \stackrel{\text{def}}{=} (X_i^\# - 1) \mathbf{U}_i^\#$$

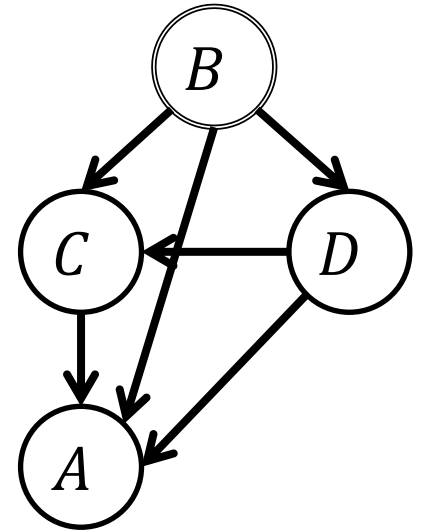
Model Complexity: DAG Dimension

- For example, the dimension of this DAG considering all binary variables is

- $\|B\| = 1, \|CB\| = 2, \|DB\| = 2, \|AC\| = 2$. Total = 7
- $\|B\| = 1, \|CB\| = 2, \|DB\| = 2, \|ACD\| = 4$. Total = 9
- $\|B\| = 1, \|CBD\| = 4, \|DB\| = 2, \|ABCD\| = 8$. Total = 15

<https://tutorcs.com>

WeChat: cstutorcs



$$\|G\| \stackrel{\text{def}}{=} \sum_i^N \|X_i \mathbf{U}_i\|$$
$$\|X_i \mathbf{U}_i\| \stackrel{\text{def}}{=} (X_i^\# - 1) \mathbf{U}_i^\#$$

Model Complexity: DAG Dimension

- For example, the dimension of this DAG considering all binary variables is

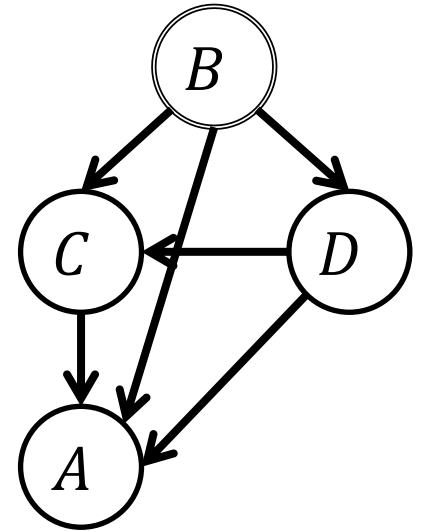
- $\|B\| = 1, \|CB\| = 2, \|DB\| = 2, \|AC\| = 2$. Total = 7
- $\|B\| = 1, \|CB\| = 2, \|DB\| = 2, \|ACD\| = 4$. Total = 9
- $\|B\| = 1, \|CBD\| = 4, \|DB\| = 2, \|ABCD\| = 8$. Total = 15

<https://tutorcs.com>

- Using this notion of model complexity, we can define the following class of scoring measures

$$Score(G; \mathcal{D}) \stackrel{\text{def}}{=} LL(G; \mathcal{D}) - \psi(N) \cdot \|G\|$$

- The first component is the log-likelihood of the graph G
- The second one $\psi(N) \cdot \|G\|$ is a penalty term that favours simple models
- The penalty term has a weight, $\psi(N) \geq 0$, that is a function of the dataset size N



$$\|G\| \stackrel{\text{def}}{=} \sum_i^N \|X_i \mathbf{U}_i\|$$

$$\|X_i \mathbf{U}_i\| \stackrel{\text{def}}{=} (X_i^\# - 1) \mathbf{U}_i^\#$$

Scoring Measures

- When the penalty weight $\psi(N)$ is a constant independent of N
 - We obtain a score in which the model's complexity is a secondary issue
 - The log-likelihood function $LL(G; \mathcal{D})$ grows linearly in the dataset size N
 - Therefore, the log-likelihood will quickly dominate the penalty term
 - The model complexity will be used to distinguish between models that have relatively equal log-likelihood terms
 - This scoring measure is known as *Akaike information criterion* (AIC)
- A more common choice is $\psi(N) = \log_2 N / 2$, which leads to a more influential term
 - However, this term grows logarithmically in N , while the log-likelihood term grows linearly
 - The influence of model complexity decreases as N grows, allowing the log-likelihood eventually dominate
 - This penalty weight gives rise to the *minimum description length* (MDL) score

Minimum Description Length (MDL)

- Minimum Description Length (MDL) score

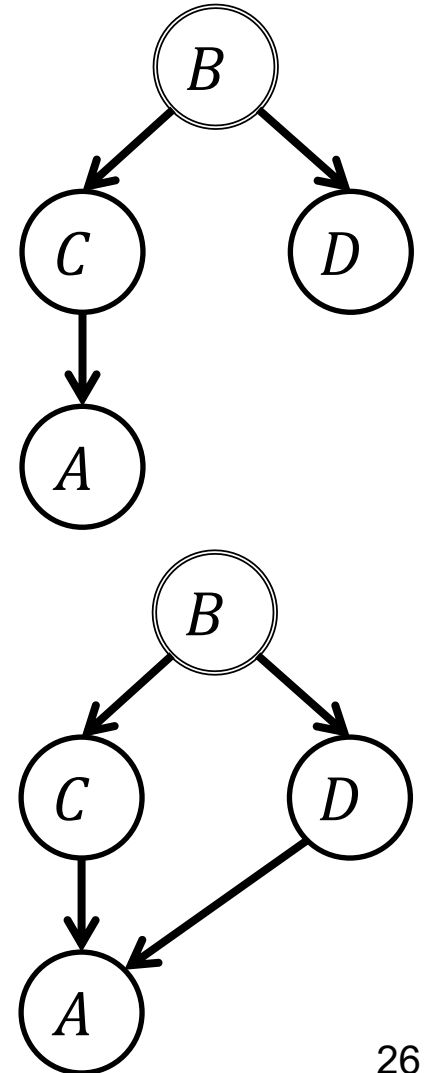
$$MDL(G; \mathcal{D}) \stackrel{\text{def}}{=} LL(G; \mathcal{D}) - \left(\frac{\log_2 N}{2} \right) \|G\|$$

- For example, the network on the top has the following MDL score

$$\begin{aligned} &= -12.1 - \left(\frac{\log_2 5}{2} \right) 7 \\ &= -12.1 - 8.1 \\ &= -20.2 \end{aligned}$$

- For example, the network on the bottom has the following MDL score

$$\begin{aligned} &= -10.1 - \left(\frac{\log_2 5}{2} \right) 9 \\ &= -10.1 - 10.4 \\ &= -20.5 \end{aligned}$$

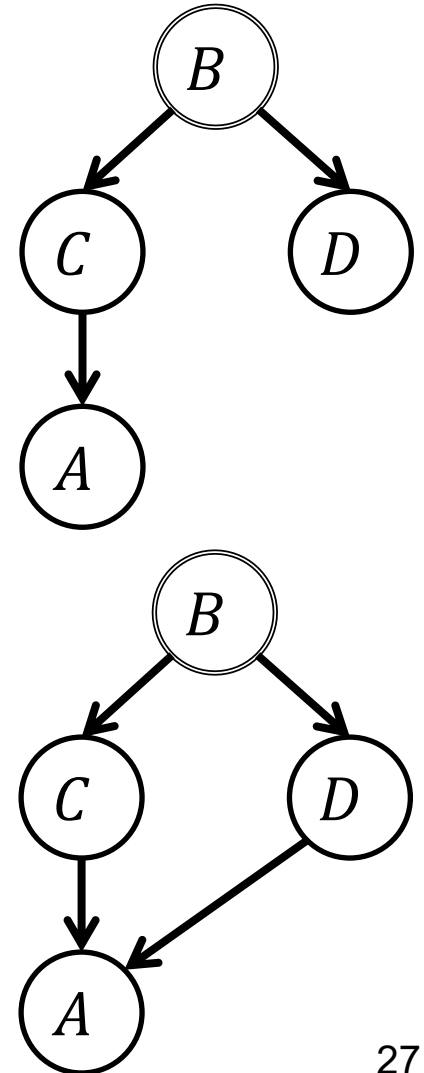


Minimum Description Length (MDL)

- Minimum Description Length (MDL) score

$$MDL(G; \mathcal{D}) \stackrel{\text{def}}{=} LL(G; \mathcal{D}) - \left(\frac{\log_2 N}{2} \right) \|G\|$$

- For example, the network on the top has the following MDL score
 $= -20.2$
- For example, the network on the bottom has the following MDL score
 $= -20.5$
- Therefore, MDL score prefers the top network even though it has smaller log-likelihood
- The MDL score is also known as *Bayesian information criterion* (BIC)
 - It is sometimes expressed as the negative of the score above
 - The goal becomes to minimize the score instead of maximizing it



Searching for Network Structure

- Searching for a network structure can be quite expensive
 - Due to the very large number of structures we need to consider
 - Greedy algorithms tend to be more practical when learning structures
 - Systematic search algorithms can be practical under some conditions
- Both classes of algorithms rely for their efficiency on a property of scoring functions
 - Most score functions are *decomposable* or *modular*
 - They allow to decompose the score into an aggregate of local scores
 - One for each network family

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Searching for Network Structure

- For instance, the class of scores

$$Score(G; \mathcal{D}) \stackrel{\text{def}}{=} LL(G; \mathcal{D}) - \psi(N) \cdot \|G\|$$

- can be decomposed as

$$Score(G; \mathcal{D}) = \sum_{XU} Score(X, U; G)$$

- where

$$Score(X, U; \mathcal{D}) \stackrel{\text{def}}{=} -N \cdot H_{\mathcal{D}}(X|U) - \psi(N) \cdot \|XU\|$$

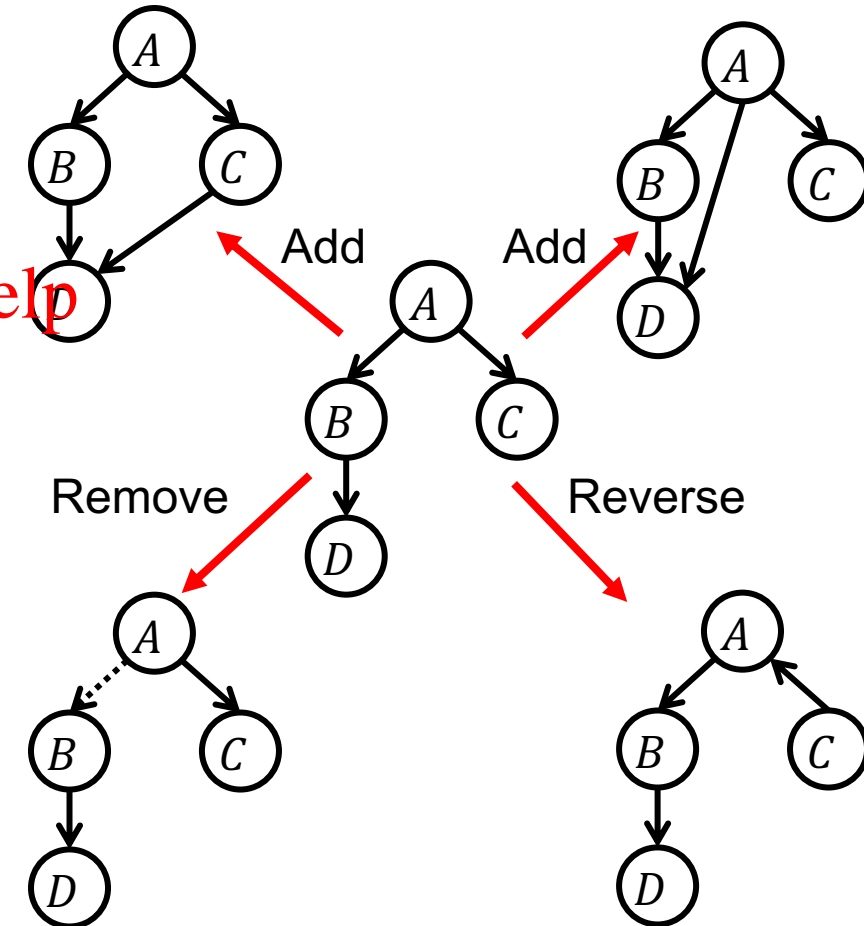
- Note how the score is a sum of local scores, each for some family XU

- The contribution of each family is split into two parts

- One resulting from a decomposition of the log-likelihood component of the score
- The other resulting from a decomposition of the penalty component
- This decomposition enables several efficient implementations of heuristic and optimal search algorithms

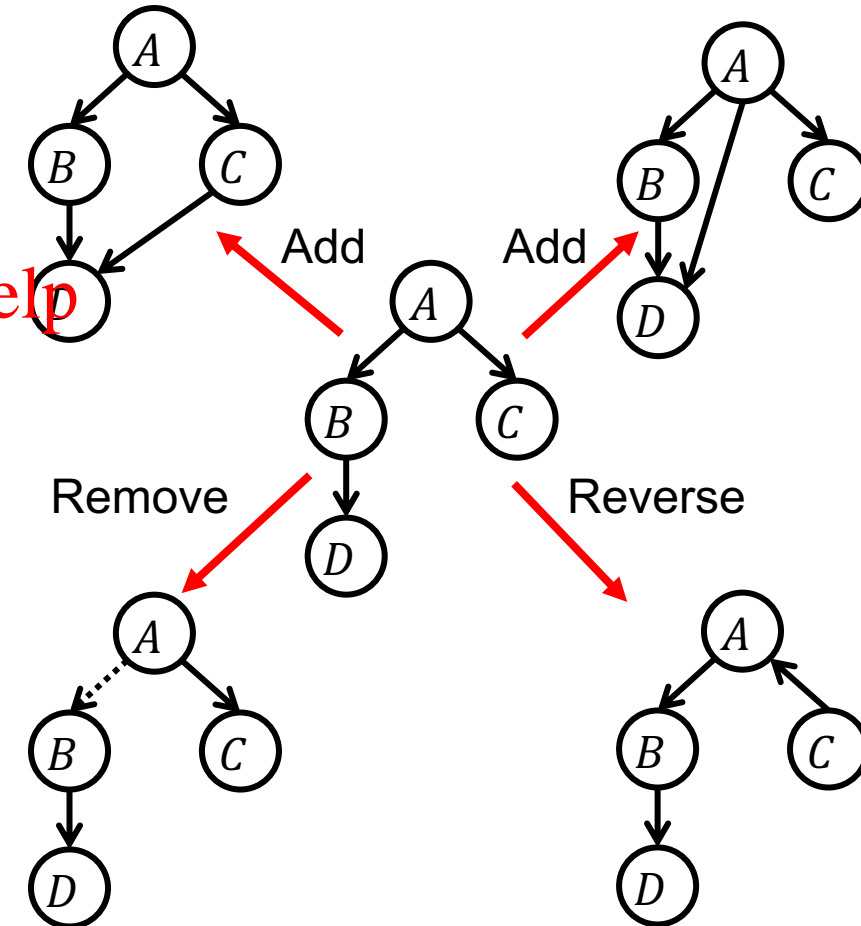
Local Search

- We can search for network structure by starting with some initial structure
 - Then modify it locally to increase its score
 - The initial structure can be chosen randomly, based on some prior knowledge, or can be a tree structure as discussed previously
- The local modifications to the structure are constrained to
 - Adding an edge
 - Removing an edge
 - Reversing an edge while ensuring the structure remains a DAG
- These local changes to the network structure also change the score
 - Possibly increasing or decreasing it
 - However, the goal is to commit to the change that increases the score the most
 - If none of the local changes can increase the score, the algorithm terminates and returns the current structure



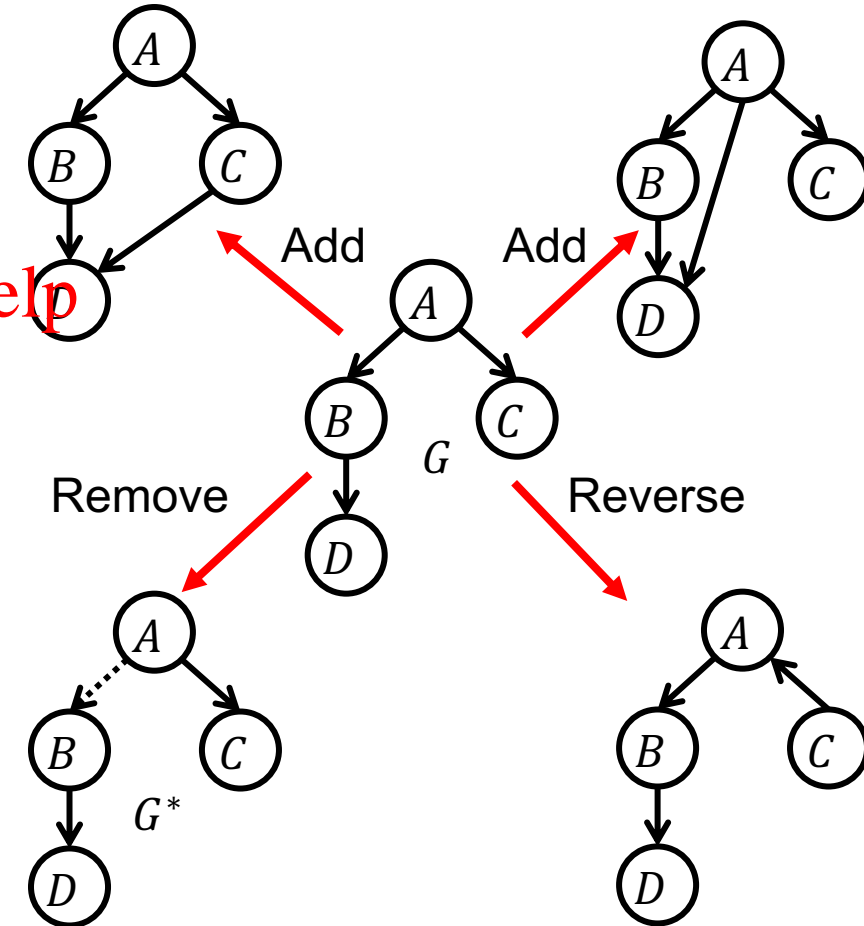
Local Search

- Some observations about this local search algorithm
 - It is not guaranteed to return an optimal network structure
 - The only guarantee provided by the algorithm is that the returned structure is locally optimal
 - That is, no local change can improve its score
- This suboptimal behaviour can be improved by techniques such as *random restarts*
 - We repeat the local search multiple times, each time starting with a different network
 - Return the network with the best score across all repetitions



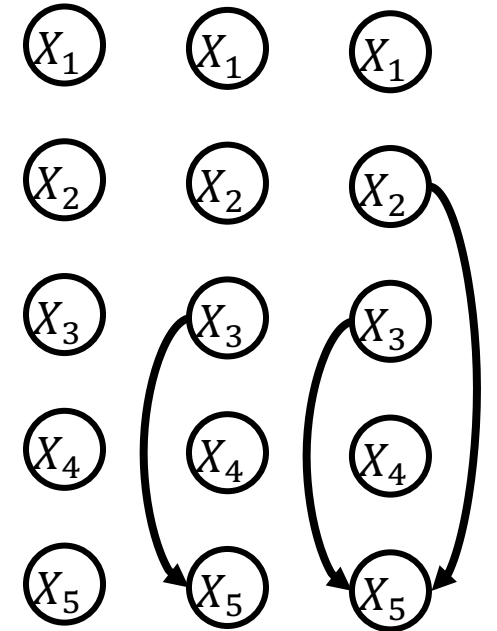
Local Search

- Another observation relates to score updating
 - Consider the networks G (centre) and G^* (after removing edge $A \rightarrow B$)
 - Since this change affects only the family of node B , we have
$$Score(G^*; \mathcal{D}) = Score(G; \mathcal{D}) - Score(B; A; \mathcal{D}) + Score(B; \mathcal{D})$$
 - That is, we discount the contribution of B 's old family and add the contribution of B 's new family
- More generally, adding or removing one edge changes only one family, while reversing an edge changes only two families
 - The score can always be updated locally as a result of the local network change



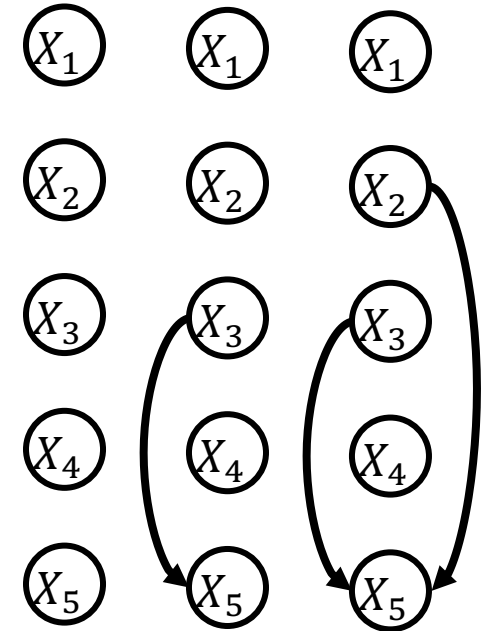
Constraining the Search Space

- A technique for reducing the search space size is to assume a total ordering on variables
 - Search only among network structures that are consistent with the chosen order
 - For instance, we use variable order X_1, \dots, X_n
 - The search process tries to find for each variable X_i a set of parents $U_i \subseteq X_1, \dots, X_{i-1}$
- This technique also allows us to decompose the search into n independent problems
 - Each concerned with finding a set of parents for some network variable
 - That is, the search problem reduces to considering each variable X_i independently
 - And then finding a set of parents $U_i \subseteq X_1, \dots, X_{i-1}$ that maximize $Score(X_i, U_i; \mathcal{D})$
- We discuss greedy and optimal methods for maximizing these local scores



Greedy Search

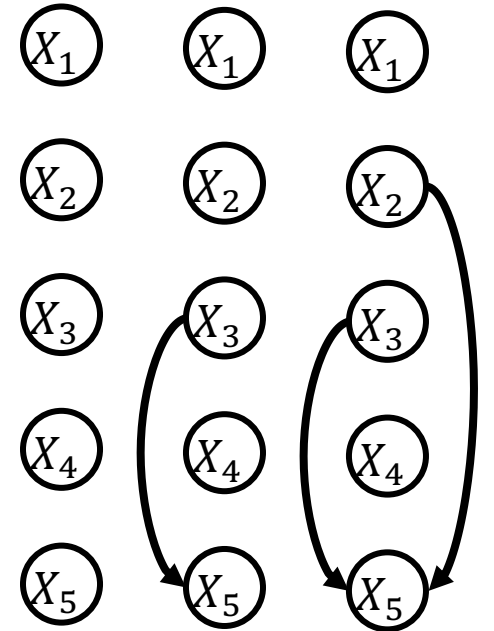
- A well-known heuristic algorithm for optimizing a family score is known as K3
 - It starts with an empty set of parents
 - Successively add variables to the set one at a time
 - Until such additions no longer increase the score
- For example, we want to find a set of parents for X_5 from X_1, \dots, X_4
 - We start with $U_5 = \emptyset$, and try to find X_i that maximizes
$$Score(X_5, X_i; \mathcal{D}) \geq Score(X_5; \mathcal{D})$$
 - Suppose X_3 is such a variable, then $U_5 = \{X_3\}$ and we search for another variable X_i that maximizes
$$Score(X_5, X_3, X_i; \mathcal{D}) \geq Score(X_5, X_3; \mathcal{D})$$
 - Suppose X_2 happens to be such a variable, then $U_5 = \{X_2, X_3\}$
 - If no other variable can improve the score, then K3 terminates with $U_5 = \{X_2, X_3\}$ as the parent set for X_5



Greedy Search

- K3 is not guaranteed to identify the optimal set of parents
 - That is, the one that maximizes $Score(X_i, \mathbf{U}_i; \mathcal{D})$

- Therefore, it is common to use the output of this algorithm as a starting point for other algorithms
 - Such as the local search algorithm discussed previously
 - Or the optimal search algorithm we discuss next

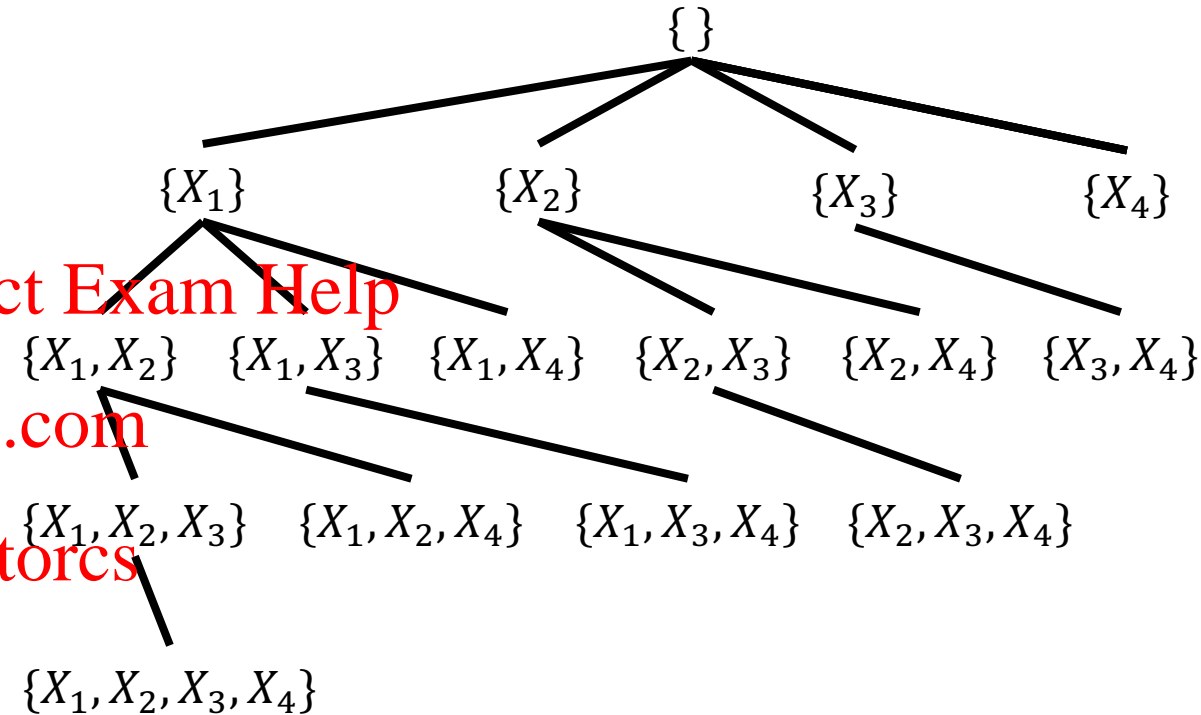


Optimal Search

- We discuss an optimal search algorithm for network structures

- Based on branch-and-bound depth-first search
- Like K3, it assumes a total order of variables X_1, \dots, X_n
- The search is restricted among structures consistent with this order
- This allow us to decompose the search into n independent search problems

- For instance, consider the search for parents of X_5
 - The first level of the tree represents an empty set of parents $U_5 = \{\}$
 - Each additional level corresponds by adding a single variable to each parent set while avoiding duplicate sets



- The search tree has 2^{i-1} nodes, corresponding to the number of subsets we can choose from X_1, \dots, X_{i-1}

Optimal Search: Pruning

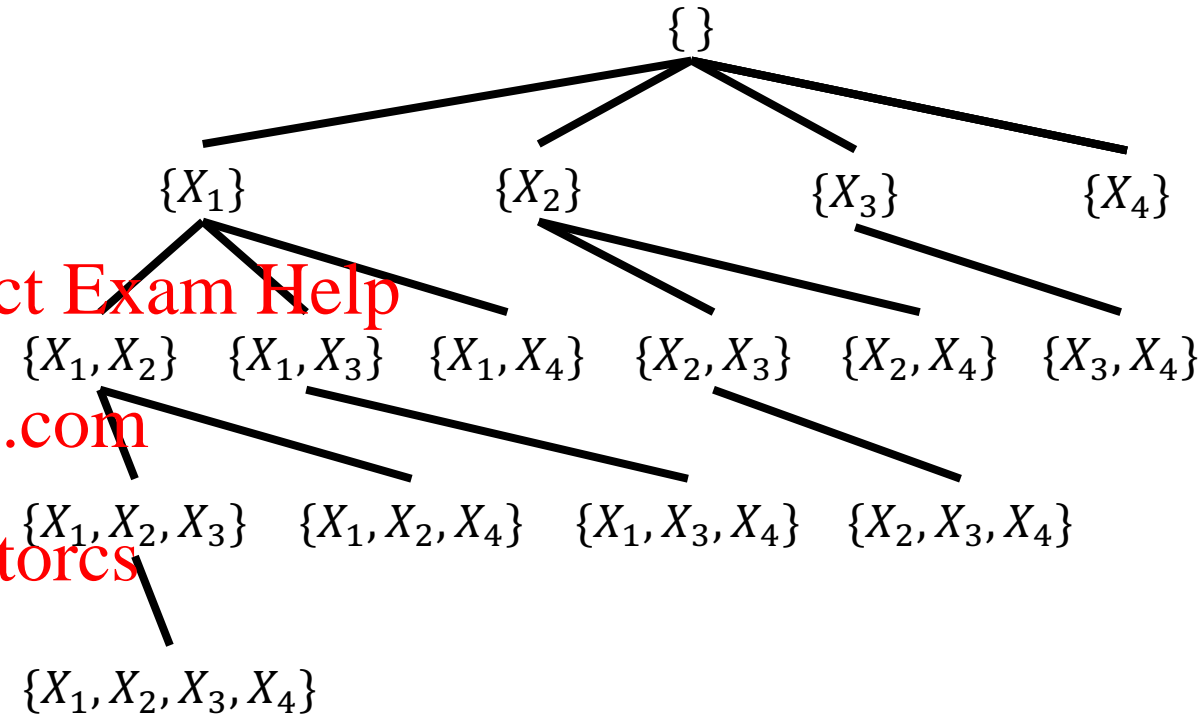
- We can search this tree using depth-first search
 - Maintaining the score s of the best parent visited so far
 - When visiting node \mathbf{U}_i , we evaluate $Score(X_i, \mathbf{U}_i; \mathcal{D})$ and check if it is better than s
 - Depth-first guarantees every parent set is visited, leading us to identify the optimal parent set
 - However, such optimality comes at the expense of exponential complexity

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

- The complexity can be improved on average
 - If we use an upper bound on $Score(X_i, \mathbf{U}_i^*; \mathcal{D})$, where $\mathbf{U}_i \subseteq \mathbf{U}_i^*$
 - If the upper bound at node \mathbf{U}_i is not better than the best score s , then we prune \mathbf{U}_i and all nodes below it
 - The extend of pruning depends on the quality of the upper bound used



Optimal Search: Pruning

- For the MDL score, we can use the following upper bound
 - Let \mathbf{U}_i be the parent set and let \mathbf{U}_i^+ be the largest parent set appearing below \mathbf{U}_i in the search tree. If \mathbf{U}_i^* is a parent set in the tree rooted at \mathbf{U}_i , then

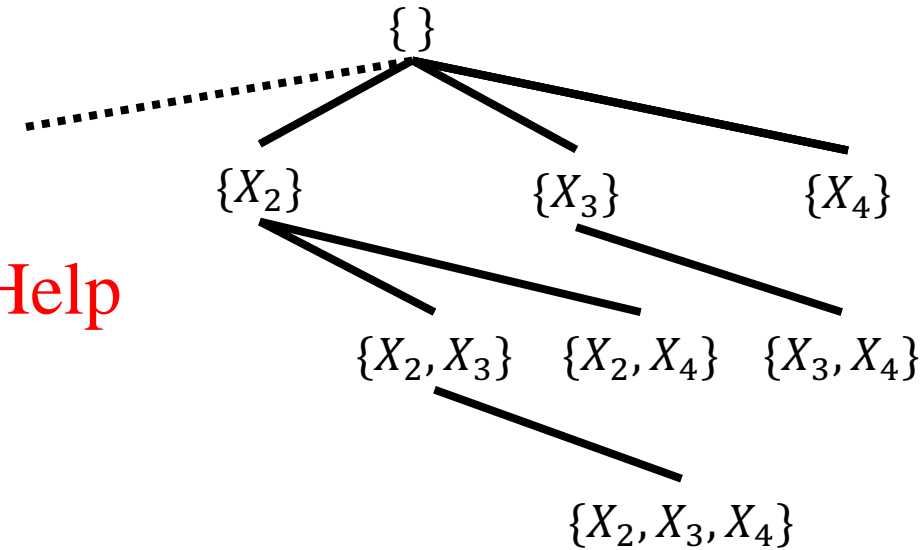
$$MDL(X_i, \mathbf{U}_i^*; \mathcal{D}) \leq -N \cdot H_{\mathcal{D}}(X_i | \mathbf{U}_i^+) - \psi(N) \cdot \|\mathbf{U}_i\|$$

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

- This bound needs to be computed at each node \mathbf{U}_i
 - For example, at search node $\mathbf{U}_5 = \{X_2\}$, we get $\mathbf{U}_5^+ = \{X_2, X_3, X_4\}$
 - Moreover, \mathbf{U}_5^* ranges over parent sets $\{X_2\}$, $\{X_2, X_3\}$, $\{X_2, X_4\}$ and $\{X_2, X_3, X_4\}$



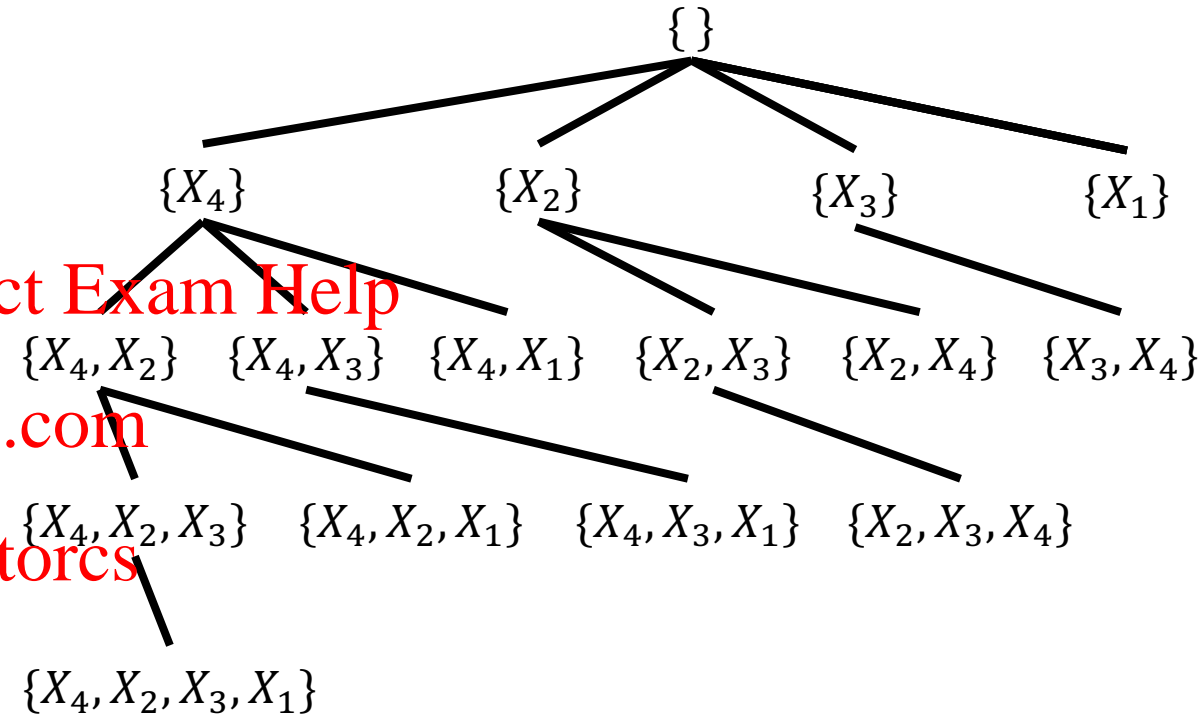
Optimal Search: Variable Order

- Consider this search tree with different variable orders

- Now variable X_4 appears more frequently on the first branch
- Suppose variable X_4 tends to reduce the entropy of X_5 more than does X_1
- Then we expect this search tree to visit parent sets with higher scores first
- This may lead to more aggressive pruning

- Therefore, we prefer a tree that is expanded according to variable order $X_{k_1}, \dots, X_{k_{i-1}}$, where

$$H(X_i|X_{k_1}) \leq \dots \leq H(X_i|X_{k_{i-1}})$$



Structure Learning and Incomplete Data

- We have restricted our discussion on the search for network structures to complete datasets
 - The main reason is computational
 - The likelihood of a network structure does not admit a closed form when the dataset is incomplete
 - Also, it does not decompose into components
- Hence, algorithms for learning structures with incomplete data involve two searches
 - An outer search in the space of network structures
 - An inner search in the space of network parameters

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Conclusion

- In this lecture, we studied approaches to learn the network structure from data
 - We focused on approaches that search for a maximum likelihood structure
- The first approach was restricted to tree structures
 - Therefore, each nodes had at most one parent
 - Our approach could find an optimal structure using maximum spanning trees
- The second approach extended structures to DAGs
 - Adding edges never decreases the log-likelihood of the structure
 - Therefore, unrestricted search to minimize conditional entropy leads to complex structures and overfitting
 - We reviewed a class of scoring functions that ally data fitness and model complexity
- When searching for a DAG, we studied two main strategies
 - Local search based on hill climbing
 - Constrained search methods that assume a variable order to reduce the search space
- Task
 - Read chapter 17

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: estutores