Outline

Overview of last class:

- Counterfactuals
- Defining and computing counterfactuals.
- The tree steps of computing counterfactuals (the deterministic case)

Nondeterministic counterfactuals.

- The 3-steps
- Do operators are limited and Expressing do by counterfactuals
- The graphical representation of counterfactuals
- Counterfactuals in Experimental Settings
- Practical use of counterfactuals

Counterfactual in Experimental Settings

So we can answer counterfactual question from a fully specified structural model.

But what to do when a model is not available, and we have only a finite sample of observed individuals?

Let's consider again the "encouragement design" model in which we analyzed the behavior of an individual named Joe. Assume that the experimenter observes a set of 10 individuals, with Joe being participant 1. Each, characterized by a distinct vector $U_i = (U_X, U_H, U_Y)$, as shown in the first 3 columns

Table 4.3 Potential and observed outcomes predicted by the structural model of Figure 4.1 units were selected at random, with each U_i uniformly distributed over $[0, 1]$

	Participant characteristics			Observed behavior			Predicted potential outcomes				
Participant	U_X	U_H	U_Y	X	Y	H	Y_0	Y_1	H_0	H_1	$Y_{00}\ldots$
1	0.5	0.75	0.75	0.5	1.50	1.0	1.05	1.95	0.75	1.25	0.75
2	0.3	0.1	0.4	0.3	0.71	0.25	0.44	1.34	0.1	0.6	0.4
3	0.5	0.9	0.2	0.5	1.01	1.15	0.56	1.46	0.9	1.4	0.2
4	0.6	0.5	0.3	0.6	1.04	0.8	0.50	1.40	0.5	1.0	0.3
5	0.5	0.8	0.9	0.5	1.67	1.05	1.22	2.12	0.8	1.3	0.9
6	0.7	0.9	0.3	0.7	1.29	1.25	0.66	1.56	0.9	1.4	0.3
7	0.2	0.3	0.8	0.2	1.10	0.4	0.92	1.82	0.3	0.8	0.8
8	0.4	0.6	0.2	0.4	0.80	0.8	0.44	1.34	0.6	1.1	0.2
9	0.6	0.4	0.3	0.6	1.00	0.7	0.46	1.36	0.4	0.9	0.3
10	0.3	0.8	0.3	0.3	0.89	0.95	0.62	1.52	0.8	1.3	0.3

$$X = U_X$$
$$H = a \cdot X + U_H$$
$$Y = b \cdot X + c \cdot H + U_Y$$
$$\sigma_{U_i U_j} = 0 \quad \text{for all } i, j \in \{X, H, Y\}$$

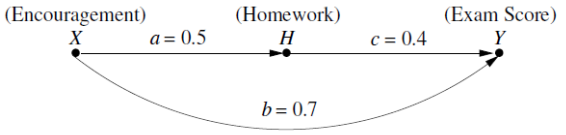


Figure 4.1: A model depicting the effect of Encouragement (X) on student's score

We use the model to fill the data from the U variables.

First item: Y_0 = 0.4 times 1 + 0.75= 1.05

Counterfactual in Experimental Settings

From this synthetic population, one can estimate the probability of every counterfactual query on variables X, Y,Z, assuming, of course, that we are in possession of all entries of the table.

Clearly the table is not available to us in either observational or experimental studies. This was deduced from the fully specified model from which we could infer the defining characteristics $\{U_X,U_H,U_Y\}$ of each participant, given the observations $\{X,H, Y\}$.

Without a parametric model, the observed behavior $\{X,H, Y\}$ tells very little of the potential outcome Y_1 or Y_0.

We know only the consistency rule: that $Y1$ must be equal to Y in case X = 1, and Y_0 must be equal to Y in case X = 0.

Yet we can say much at the population level estimating their probabilities or expectation. We can use
The adjustment formula of (4.16), where we were able to compute $E(Y\_1 - Y\_0)$ using the graph alone as we will see next.

Using Experimental Data

Assume that we have no information whatsoever about the underlying model. All we have are measurements on Y taken in an experimental study in which X is randomized over two levels, $X = 0$ and $X = 1$.

Table 4.4 Potential and observed outcomes in a randomized clinical trial with X randomized over $X = 0$ and $X = 1$

	Predicted potential outcomes		Observed outcomes	
Participant	Y_0	Y_1	Y_0	Y_1
1	1.05	1.95	1.05	■
2	0.44	1.34	■	1.34
3	0.56	1.46	■	1.46
4	0.50	1.40	■	1.40
5	1.22	2.12	1.22	■
6	0.66	1.56	0.66	■
7	0.92	1.82	■	1.82
8	0.44	1.34	0.44	■
9	0.46	1.36	■	1.36
10	0.62	1.52	0.62	■
	True average treatment effect: 0.90		Study average treatment effect: 0.68	

Randomized: participants 1, 5, 6, 8 and 10 assigned to X = 0, and the rest to X = 1. The first two columns give the true potential outcomes (taken from Table 4.3) while the last two columns describe the information available to the experimenter.

The difference between the observed means in the treatment and control groups will converge to the difference of the population averages, $E(Y\_1 - Y\_0) = 0.9$ due to randomization.

Under randomization, the adjustment formula (4.16) is applicable with Z = {empty}, yielding $E[Y\_x] = E[Y \mid X = x]$.
So, Table 4.4 helps us understand what is actually computed when we take sample averages in experimental settings and how those averages are related to the underlying counterfactuals, Y_1 and Y_0.

ATE (Average Treatment Effect)

- No information on the underlying model, we can run experiments
  - What does random X do?
  - Removes arrows into X
  - Estimates Y_0 and Y_1
  - $E[Y_x] = \sum_z E[Y|z,x] \cdot P(z)$
    - $Z = \emptyset$
  - $E[Y_x] = E[Y|x]$
- Estimate $E[Y_1 - Y_0]$
  - Average observations
  - $= \sum Y_1/n - \sum Y_0/n$
  - $= 0.68$
  - Should be 0.9, why isn't it?
  - Small sample size

	Predicted potential outcomes		Observed outcomes	
Participant	Y_0	Y_1	Y_0	Y_1
1	1.05	1.95	1.05	■
2	0.44	1.34	■	1.34
3	0.56	1.46	■	1.46
4	0.50	1.40	■	1.40
5	1.22	2.12	1.22	■
6	0.66	1.56	0.66	■
7	0.92	1.82	■	1.82
8	0.44	1.34	0.44	■
9	0.46	1.36	■	1.36
10	0.62	1.52	0.62	■
	True average treatment effect: 0.90		Study average treatment effect: 0.68	

Outline

Overview of last class:

- Counterfactuals
- Defining and computing counterfactuals.
- The tree steps of computing counterfactuals (the deterministic case)

Nondeterministic counterfactuals.

- The 3-steps
- Do operators are limited and Expressing do by counterfactuals
- The graphical representation of counterfactuals
- Counterfactuals in Experimental Settings
- Practical use of counterfactuals

Practical Uses of Counterfactuals

Recruitment program

Additive Interventions

Personal decision making

Sex discrimination in hiring

Mediation and path disabling

Recruitment Program Job Training Helps?

Example 4.4.1 A government is funding a job training program aimed at getting jobless people back into the workforce. A pilot randomized experiment shows that the program is effective; a higher percentage of people were hired among those who finished the program than among those who did not go through the program. As a result, the program is approved, and a recruitment effort is launched to encourage enrollment among the unemployed, by offering the job training program to any unemployed person who elects to enroll.

Enrollment is successful, and the hiring rate among the program's graduates turns out even higher than in the randomized pilot study. Success!!!

Critics say: Those who self-enroll, may be more intelligent, more resourceful, and more socially connected than the eligible who did not enroll and are more likely to have found a job regardless of the training.

The critics claim that what we need to estimate is the differential benefit of the program on those enrolled: the extent to which hiring rate has increased among the enrolled, ETT= compared to what it would have been had they not been trained.

$X = 1$ represent training and $Y = 1$ represent hiring, the quantity that needs to be evaluated is the effect of training on the trained (ETT, better known as “effect of treatment on the treated,”

$$ETT = E[Y_1 - Y_0 | X = 1] \quad (4.20)$$

Here the difference $Y_1 - Y_0$ represents the causal effect of training (X) on hiring (Y) for a randomly chosen individual, and the condition $X = 1$ limits the choice to those actually choosing the training program on their own initiative. As in our freeway example of Section 4.1, we are

Personal Decision Making

Example 4.4.3 Ms. Jones, a cancer patient, is facing a tough decision between two possible treatments: (i) lumpectomy alone, or (ii) lumpectomy plus irradiation. In consultation with her oncologist, she decides on (ii). Ten years later, Ms. Jones is alive, and the tumor has not recurred. She speculates: Do I owe my life to irradiation?
Mrs. Smith, on the other hand, had a lumpectomy alone, and her tumor recurred after a year. And she is regretting: I should have gone through irradiation.
Can these speculations ever be substantiated from statistical data? Moreover, what good would it do to confirm Ms. Jones's triumph or Mrs. Smith's regret?

Sex Discrimination in Hiring

Example 4.4.4 Mary files a law suit against the New York-based XYZ International, alleging discriminatory hiring practices. According to her, she has applied for a job with XYZ International, and she has all the credentials for the job, yet she was not hired, allegedly because she mentioned, during the course of her interview, that she is gay. Moreover, she claims, the hiring record of XYZ International shows consistent preferences for straight employees. Does she have a case? Can hiring records prove whether XYZ International was discriminating when declining her job application?
At the time of writing, U.S. law doesn't specifically prohibit employment discrimination on

276, Causal and Pobabilistic Reasoning

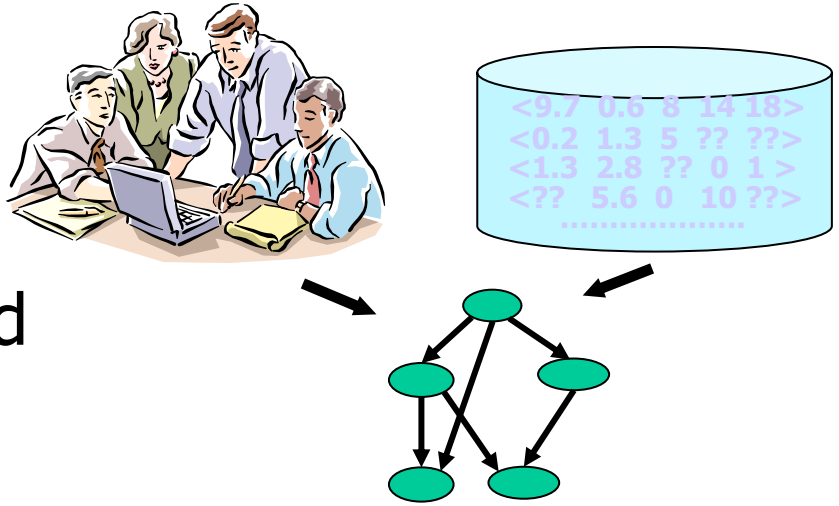
Rina Dechter, UCI

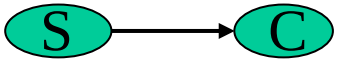
Lecture 14: Learning Bayesian Networks, the maximum likelihood approach

Darwiche chapters 17 and 18
Slides, Darwiche

Why Learn Bayesian Networks?

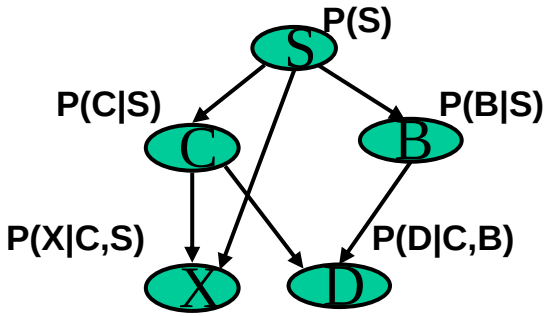
- Combining domain expert knowledge with data



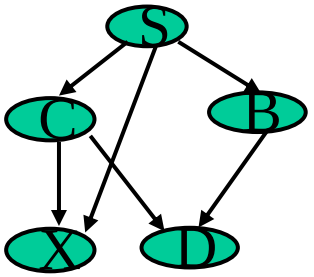
- Efficient representation and inference
- Incremental learning: P(H) ↗ or ↘
- Handling missing data: **<1.3 2.8 ?? 0 1 >**
- Learning causal relationships: 

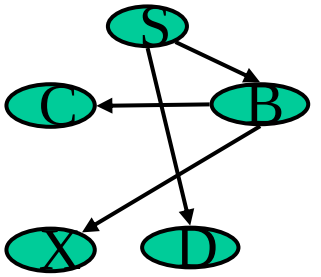
Learning Bayesian Networks

- Known graph – learn parameters
  - Complete data:
    parameter estimation (ML, MAP)
  - Incomplete data:
    non-linear parametric
    optimization (gradient descent, EM)



- Unknown graph – learn graph and parameters
  - Complete data:
    optimization (search
    in space of graphs)
  - Incomplete data:
    structural EM,
    mixture models





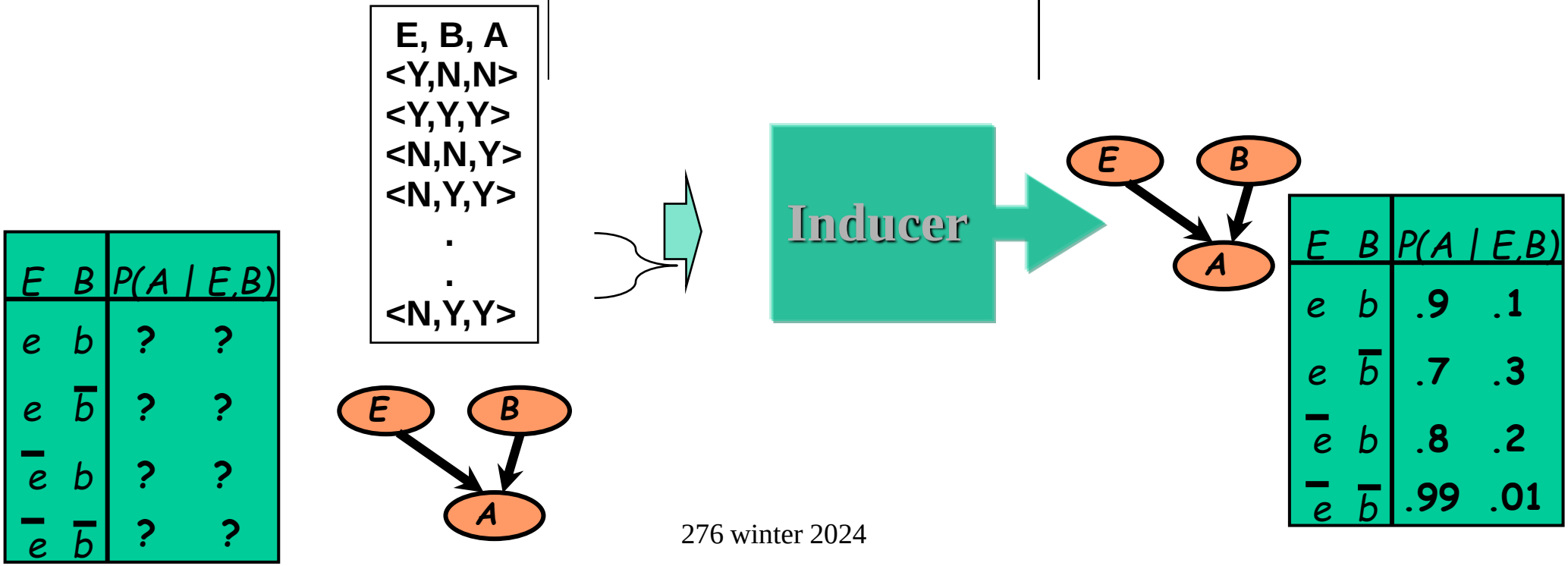
$$\hat{G} = \arg\max_{G} \text{Score}(G)$$

The Learning Problem

	Known Structure	**Unknown Structure**
Complete Data	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete Data	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)

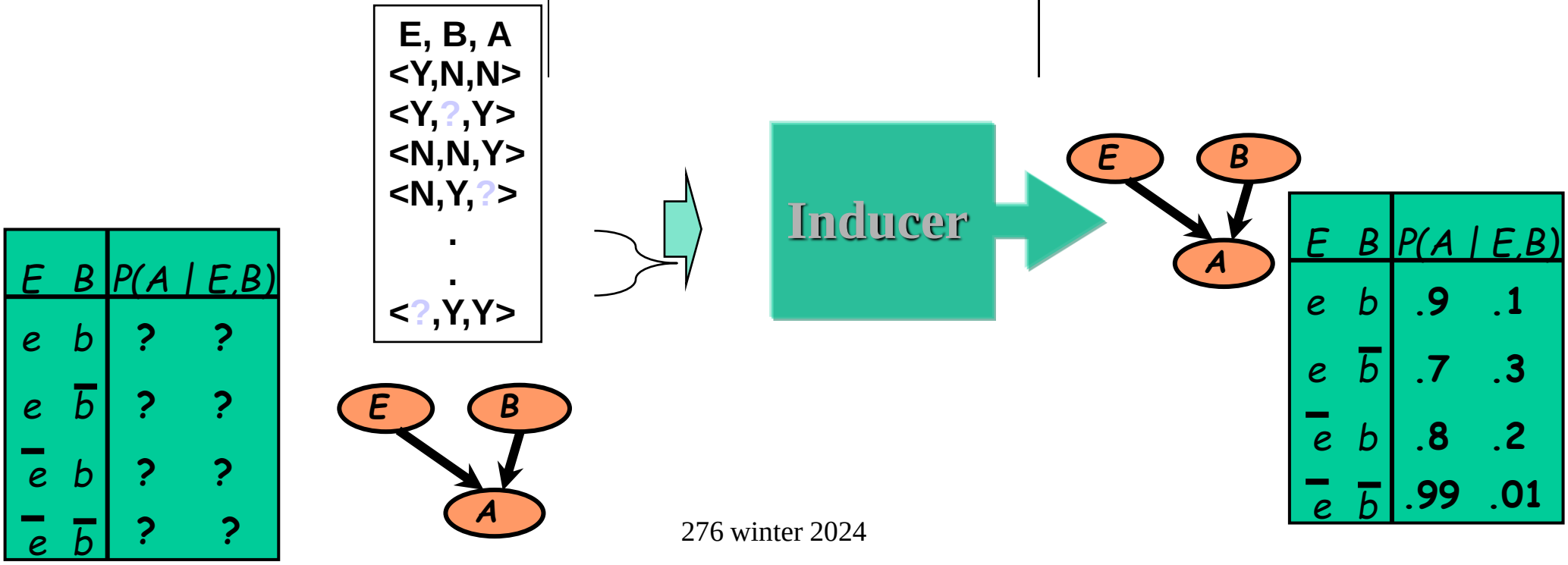
Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



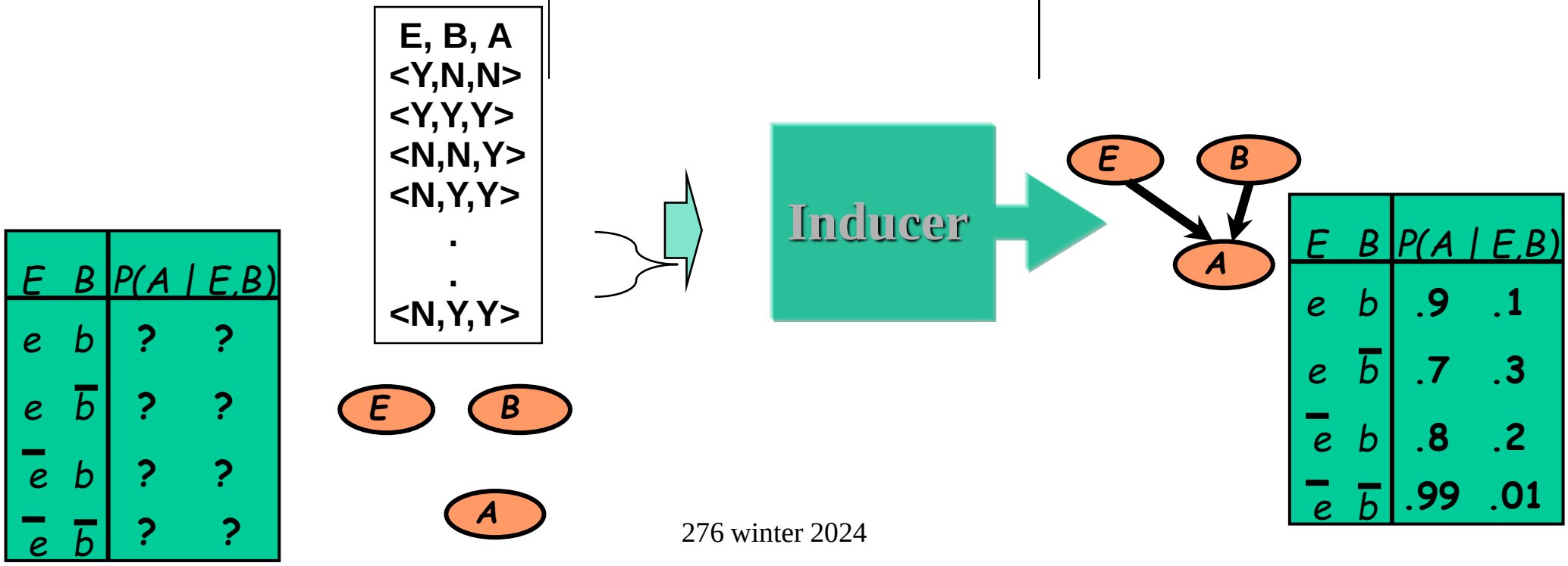
Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



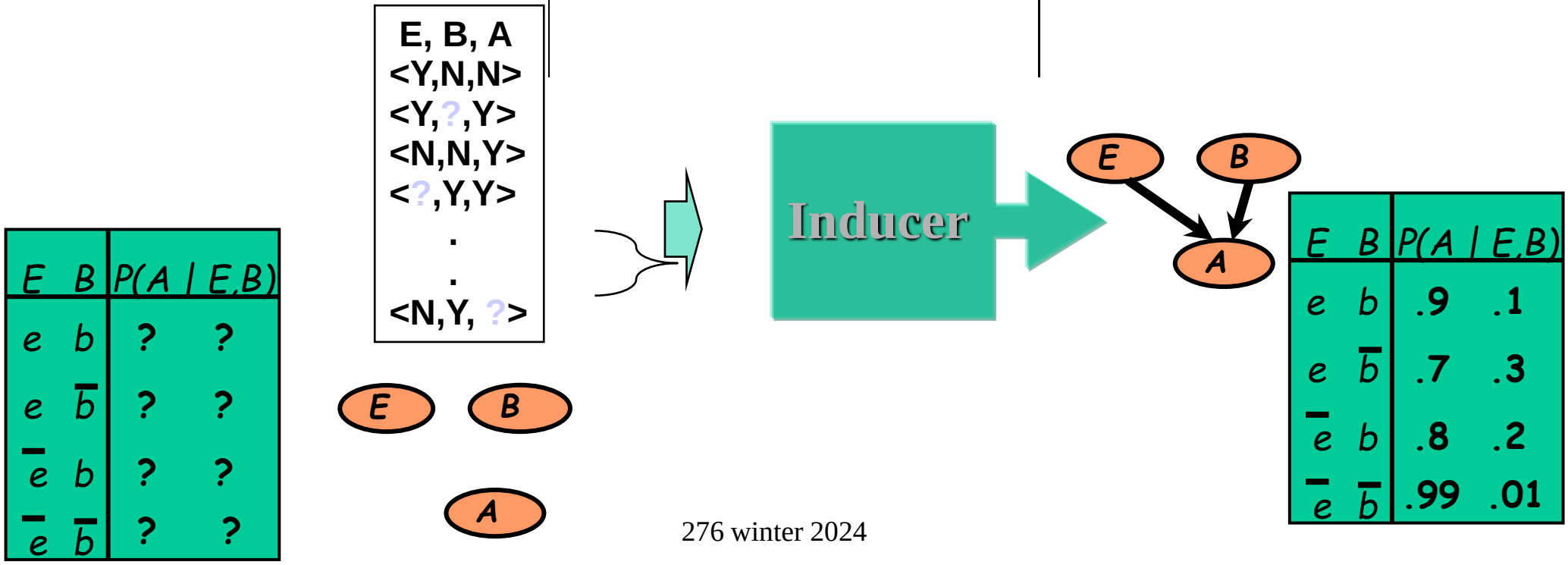
Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



Introduction

One can distinguish between three general approaches to the learning problem.

The first approach is based on the likelihood principle

which favors those estimates that have a maximal likelihood, i.e., ones that maximize the probability of observing the given data set. This approach is therefore known as the maximum likelihood approach to learning.

This is the approach treated in this chapter.

Introduction

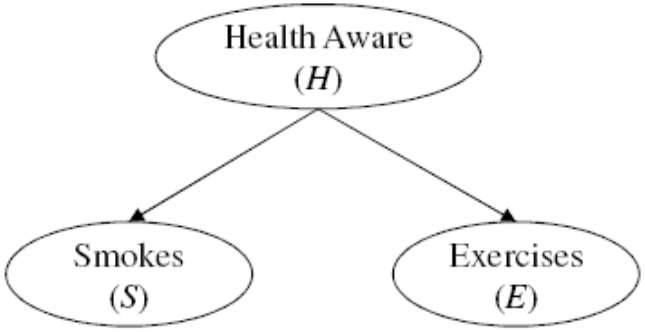
The second approach requires more input to the learning process

as it demands one to define a meta distribution over network structures and parameters. It then reduces the problem of learning to a problem of classical inference in which the data set is viewed as evidence. In particular, it first conditions the meta distribution on the given data set, and then uses the posterior meta distribution as a criterion for defining estimates.

This approach is known as the Bayesian approach to learning and will be treated in Chapter 18.

Estimating Parameters from Complete Data

Assume each data is generated independently from the true distribution



(a) network structure

Case	H	S	E
1	T	F	T
2	T	F	T
3	F	T	F
4	F	F	T
5	T	F	F
6	T	F	T
7	F	F	F
8	T	F	T
9	T	F	T
10	F	F	T
11	T	F	T
12	T	T	T
13	T	F	T
14	T	T	T
15	T	F	T
16	T	F	T

(b) complete data

H	S	E	$\Pr_{\mathcal{D}}(.)$
T	T	T	2/16
T	T	F	0/16
T	F	T	9/16
T	F	F	1/16
F	T	T	0/16
F	T	F	1/16
F	F	T	2/16
F	F	F	1/16

(c) empirical distribution

Estimating Parameters from Complete Data

Assumption

Data simulated from the true Bayesian network: the cases generated independently according to their true probabilities.

Empirical distribution summarizes data set.

H	S	E	$\Pr_{\mathcal{D}}(.)$
T	T	T	$2/16$
T	T	F	$0/16$
T	F	T	$9/16$
T	F	F	$1/16$
F	T	T	$0/16$
F	T	F	$1/16$
F	F	T	$2/16$
F	F	F	$1/16$

The empirical probability of instantiation h, s, e

is its frequency of occurrence in the data set:

$$\Pr_{\mathcal{D}}(h, s, e) = \frac{\mathcal{D}\#(h, s, e)}{N},$$

where $\mathcal{D}\#(h, s, e)$ is the number of cases in the data set $\mathcal{D}$ that satisfy instantiation h, s, e, and N is the data set size.

Estimating Parameters from Complete Data

Estimate parameters based on the empirical distribution

Consider the parameter $\theta_{s|h}$ for example, which corresponds to the probability that a person will smoke given that they are health aware, $\Pr(s|h)$. Our estimate for this parameter is now given by:

$$\Pr_{\mathcal{D}}(s|h) = \frac{\Pr_{\mathcal{D}}(s,h)}{\Pr_{\mathcal{D}}(h)} = \frac{2/16}{12/16} = 1/6$$

Estimating Parameters from Complete Data

Basic definitions

A data set $\mathcal{D}$ for variables $\mathbf{X}$ is a vector $\mathbf{d}_1, \ldots, \mathbf{d}_N$, where each $\mathbf{d}_i$ is called a case and represents a partial instantiation of variables $\mathbf{X}$. The data set is complete if each case is a complete instantiation of variables $\mathbf{X}$; otherwise, the data set is incomplete. The empirical distribution for a complete data set $\mathcal{D}$ is defined as follows:

$$\Pr_{\mathcal{D}}(\alpha) \stackrel{def}{=} \frac{\mathcal{D}\#(\alpha)}{N},$$

where $\mathcal{D}\#(\alpha)$ is the number of cases $\mathbf{d}_i$ in the data set $\mathcal{D}$ that satisfy event α, that is, $\mathbf{d}_i \models \alpha$.

$\mathcal{D}\#(\alpha) = N$ when α is a valid event (α satisfied by every case $\mathbf{d}_i$)

Estimating Parameters from Complete Data

Case	H	S	E
1	T	F	T
2	T	F	T
3	F	T	F
4	F	F	T
5	T	F	F
6	T	F	T
7	F	F	F
8	T	F	T
9	T	F	T
10	F	F	T
11	T	F	T
12	T	T	T
13	T	F	T
14	T	T	T
15	T	F	T
16	T	F	T

$$\begin{aligned}
\mathfrak{D}\#(\alpha) &= 9, \text{ when } \alpha \text{ is } (H=T) \wedge (S=F) \wedge (E=T);\\
\mathfrak{D}\#(\alpha) &= 12, \text{ when } \alpha \text{ is } (H=T);\\
\mathfrak{D}\#(\alpha) &= 14, \text{ when } \alpha \text{ is } (H=T) \vee (E=T).
\end{aligned}$$

Estimating Parameters from Complete Data

We estimate the parameter $\theta_{x|\mathbf{u}}$ by the empirical probability

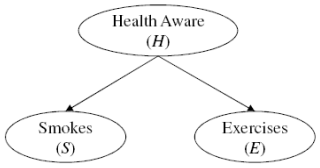
$$\theta^{ml}_{x|\mathbf{u}} \stackrel{def}{=} \Pr_{\mathcal{D}}(x|\mathbf{u}) = \frac{\mathcal{D}\#(x,\mathbf{u})}{\mathcal{D}\#(\mathbf{u})}$$

The count $\mathcal{D}\#(x,\mathbf{u})$ is called a sufficient statistic in this case.

More generally though, any function of the data is called a statistic. Moreover, a sufficient statistic is a statistic that contains all of the information in the data set that is needed for a particular estimation task.

Estimating Parameters from Complete Data

Estimating Parameters from Complete Data



(a) network structure

Case	H	S	E
1	T	F	T
2	T	F	T
3	F	T	F
4	F	F	T
5	T	F	F
6	T	F	T
7	F	F	F
8	T	F	T
9	T	F	T
10	F	F	T
11	T	F	T
12	T	T	T
13	T	F	T
14	T	T	T
15	T	F	T
16	T	F	T

(b) complete data

H	S	E	$\Pr_{\mathcal{D}}(.)$
T	T	T	2/16
T	T	F	0/16
T	F	T	9/16
T	F	F	1/16
F	T	T	0/16
F	T	F	1/16
F	F	T	2/16
F	F	F	1/16

(c) empirical distribution

We have the following parameter estimates:

H	θ^{ml}_{H}
h	$3/4$
$\bar{h}$	$1/4$

H	S	$\theta^{ml}_{S\mid H}$
h	s	$1/6$
h	$\bar{s}$	$5/6$
$\bar{h}$	s	$1/4$
$\bar{h}$	$\bar{s}$	$3/4$

H	E	$\theta^{ml}_{E\mid H}$
h	e	$11/12$
h	$\bar{e}$	$1/12$
$\bar{h}$	e	$1/2$
$\bar{h}$	$\bar{e}$	$1/2$

Estimating Parameters from Complete Data

- Estimate $\theta^{ml}_{x|\mathbf{u}}$ will have different values depending on the given data set.
- The variance of this estimate will decrease as the data set increases in size.

If data set $\mathcal{D}$ is a sample of size N simulated from distribution $\Pr$

The distribution of estimate $\theta^{ml}_{x|\mathbf{u}}$ is asymptotically Normal and can be approximated by a Normal distribution with mean $\Pr(x|\mathbf{u})$ and variance:

$$\frac{\Pr(x|\mathbf{u})(1-\Pr(x|\mathbf{u}))}{N \cdot \Pr(\mathbf{u})}$$

Estimating Parameters from Complete Data

Likelihood of parameter estimates

Let θ be the set of all parameter estimates for network structure G, and let $\Pr_\theta(.)$ be the probability distribution induced by structure G and estimates θ. The likelihood of these estimates is:

$$\mathrm{L}(\theta|\mathcal{D}) \stackrel{def}{=} \prod_{i=1}^{N} \Pr_\theta(\mathbf{d}_i)$$

Likelihood of estimates θ is the probability of observing the data set $\mathcal{D}$ under these estimates.

Estimating Parameters from Complete Data

Likelihood of parameter estimates

Let θ be the set of all parameter estimates for network structure G, and let $\Pr_\theta(.)$ be the probability distribution induced by structure G and estimates θ. The likelihood of these estimates is:

$$\mathrm{L}(\theta|\mathcal{D}) \stackrel{def}{=} \prod_{i=1}^{N} \Pr_\theta(\mathbf{d}_i)$$

Likelihood of estimates θ is the probability of observing the data set $\mathcal{D}$ under these estimates.

Estimating Parameters from Complete Data

Let $\mathcal{D}$ be a complete data set

The parameter estimates defined earlier are the only estimates that maximize the likelihood function:[a]

$$\theta^{\star} = \underset{\theta}{\operatorname{argmax}}\, \mathrm{L}(\theta|\mathcal{D}) \text{ iff } \theta^{\star}_{x|\mathbf{u}} = \Pr_{\mathcal{D}}(x|\mathbf{u})$$

[a] Assumes $\Pr_{\mathcal{D}}(\mathbf{u}) > 0$ for every instantiation $\mathbf{u}$ of every parent set $\mathbf{U}$

It is for this reason that these estimates are called maximum likelihood (ML) estimates and are denoted by θ^{ml}:

$$\theta^{ml} = \underset{\theta}{\operatorname{argmax}}\, \mathrm{L}(\theta|\mathcal{D})$$

Estimating Parameters from Complete Data

Another property of our ML estimates is that they minimize the KL–divergence between the learned Bayesian network and the empirical distribution.

Let $\mathcal{D}$ be a complete data set over variables $\mathbf{X}$

$$\operatorname*{argmax}_{\theta} \mathrm{L}(\theta|\mathcal{D}) = \operatorname*{argmin}_{\theta} \mathrm{KL}(\Pr_{\mathcal{D}}(\mathbf{X}), \Pr_{\theta}(\mathbf{X}))$$

$$KL(P,Q) = \sum_{X} P(X) \log \frac{P(x)}{Q(X)}$$

Estimating Parameters from Complete Data

Since ML estimates are unique for a given structure G and complete data set $\mathcal{D}$

the likelihood of these parameters is then a function of the structure G and data set $\mathcal{D}$

We will therefore define the likelihood of structure G given data set $\mathcal{D}$ as follows:

$$\mathrm{L}(G|\mathcal{D}) \stackrel{def}{=} \mathrm{L}(\theta^{ml}|\mathcal{D}),$$

where θ^{ml} are the ML estimates for structure G and data set $\mathcal{D}$

Estimating Parameters from Complete Data

More convenient to work with the logarithm of likelihood

$$\mathrm{LL}(\theta|\mathcal{D}) \stackrel{def}{=} \log \mathrm{L}(\theta|\mathcal{D}) = \sum_{i=1}^{N} \log \Pr_{\theta}(\mathbf{d}_i)$$

The log-likelihood of structure G is defined similarly:

$$\mathrm{LL}(G|\mathcal{D}) \stackrel{def}{=} \log \mathrm{L}(G|\mathcal{D})$$

Likelihood is ≥ 0 while log-likelihood is ≤ 0

Maximizing likelihood is equivalent to maximizing log-likelihood.

We will use $\log_2$ but write log.

Estimating Parameters from Complete Data

Log-likelihood decomposes into family-based components

Let G be a network structure and $\mathcal{D}$ be a complete data set of size N. If $X\mathbf{U}$ ranges over the families of structure G, then

$$\mathrm{LL}(G|\mathcal{D}) = -N \sum_{X\mathbf{U}} \mathrm{ENT}_{\mathcal{D}}(X|\mathbf{U}),$$

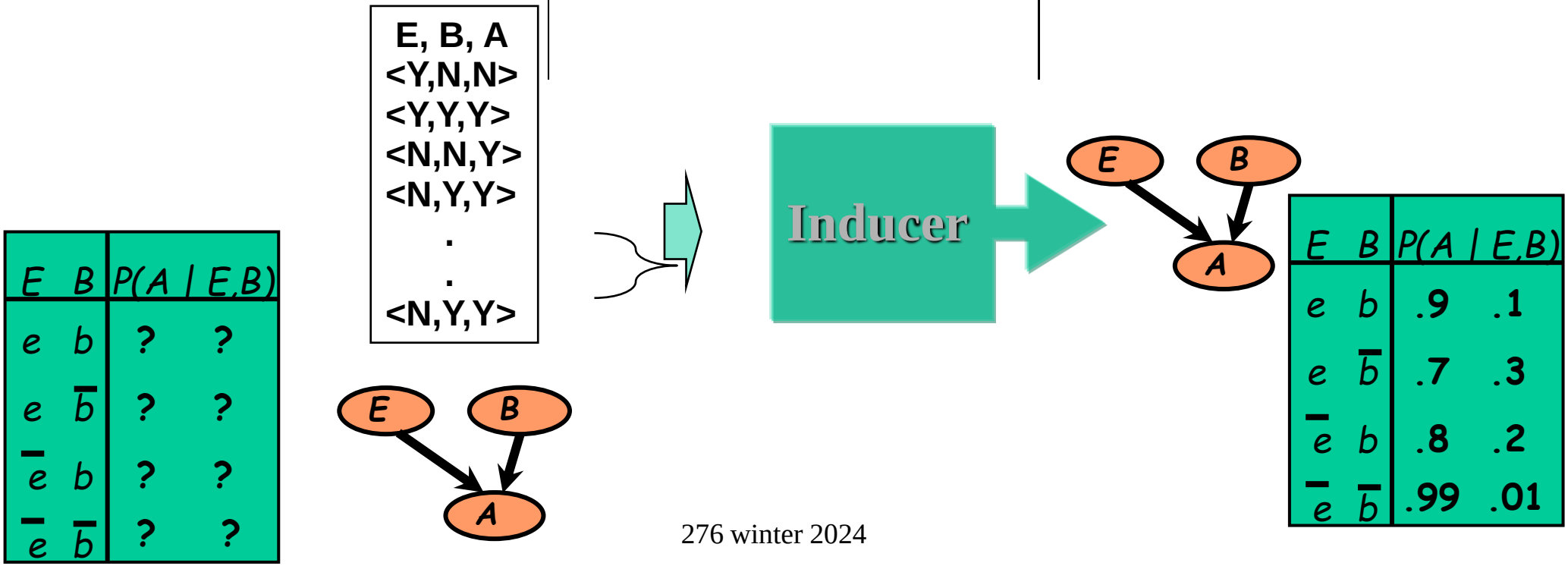
where $\mathrm{ENT}_{\mathcal{D}}(X|\mathbf{U})$ is the conditional entropy defined as follows:

$$\mathrm{ENT}_{\mathcal{D}}(X|\mathbf{U}) = -\sum_{x\mathbf{u}} \Pr_{\mathcal{D}}(x\mathbf{u}) \log_2 \Pr_{\mathcal{D}}(x|\mathbf{u})$$

Decomposition is critical when learning network structure.

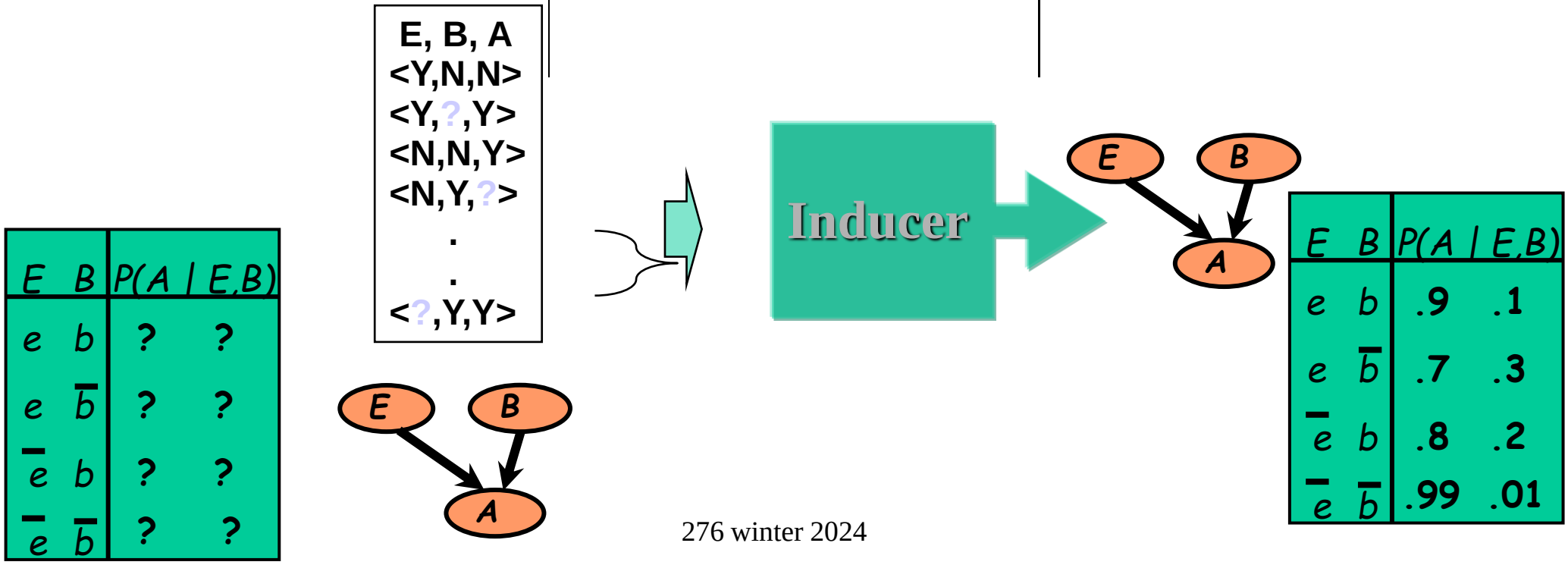
Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



Estimating Parameters from Incomplete Data

We still seek the maximum likelihood objective

Consider a network structure $C \rightarrow T$, where C represents a medical condition and T represents a test for detecting this condition:

True parameters

C	θ_c
yes	.25
no	.75

C	T	$\theta_{t\|c}$
yes	+ve	.80
yes	–ve	.20
no	+ve	.40
no	–ve	.60

Note: $\Pr(T = +\text{ve}) = \Pr(T = -\text{ve}) = 1/2$

Estimating Parameters from Incomplete Data

$\mathfrak{D}^1$	C	T
1	?	+ve
2	?	+ve
3	?	−ve
4	?	−ve
5	?	−ve
6	?	+ve
7	?	+ve
8	?	−ve

Values of variable C are missing in all cases of the first data set, perhaps because we can never determine this condition directly. We will say in this situation that variable C is hidden or latent.

Estimating Parameters from Incomplete Data

$\mathfrak{D}^1$	C	T
1	?	+ve
2	?	+ve
3	?	−ve
4	?	−ve
5	?	−ve
6	?	+ve
7	?	+ve
8	?	−ve

Values of variable C are missing in all cases of the first data set, perhaps because we can never determine this condition directly. We will say in this situation that variable C is hidden or latent.

Estimating Parameters from Incomplete Data

$\mathcal{D}^2$	C	T
1	yes	+ve
2	yes	+ve
3	yes	–ve
4	no	?
5	yes	–ve
6	yes	+ve
7	no	?
8	no	–ve

Variable C is always observed, while variable T has some missing values, but is not hidden.

Estimating Parameters from Incomplete Data

$\mathcal{D}^3$	C	T
1	yes	+ve
2	yes	+ve
3	?	–ve
4	no	?
5	yes	–ve
6	?	+ve
7	no	?
8	no	–ve

Both variables have some missing values, but neither is hidden.

Estimating Parameters from Incomplete Data

$\mathfrak{D}^1$	C	T
1	?	+ve
2	?	+ve
3	?	–ve
4	?	–ve
5	?	–ve
6	?	+ve
7	?	+ve
8	?	–ve

Cases are split equally between the +ve and –ve values of T. We expect this to be true in the limit, given the distribution generating this data.

ML estimates are characterized by

$$\theta_{T=+\text{ve}|C=\text{yes}} \cdot \theta_{C=\text{yes}} + \theta_{T=+\text{ve}|C=\text{no}} \cdot \theta_{C=\text{no}} = \frac{1}{2}$$

Estimating Parameters from Incomplete Data

ML estimates are characterized by

$$\theta_{T=+\text{ve}|C=\text{yes}} \cdot \theta_{C=\text{yes}} + \theta_{T=+\text{ve}|C=\text{no}} \cdot \theta_{C=\text{no}} = \frac{1}{2}$$

The true parameter values satisfy the above equation. But the following estimates do as well:

$$\theta_{C=\text{yes}} = 1, \quad \theta_{T=+\text{ve}|C=\text{yes}} = 1/2,$$

with $\theta_{T=+\text{ve}|C=\text{no}}$ taking any value.

ML estimates are not unique.

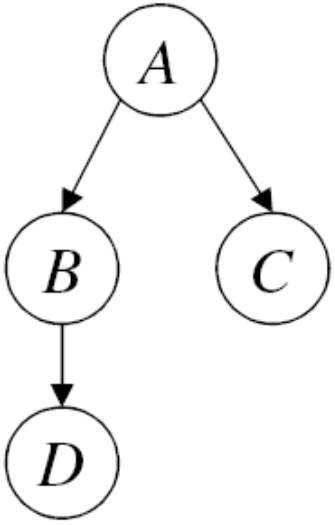
Expectation Maximization

Our first local search method, called Expectation Maximization (EM), is based on the method of complete data we discussed earlier.

This method will first complete the data set, inducing an empirical distribution, and then use it to estimate parameters as we did earlier.

The new set of parameters are guaranteed to have no less likelihood than the initial parameters, so this process can be repeated until some convergence condition is met.

Expectation Maximization

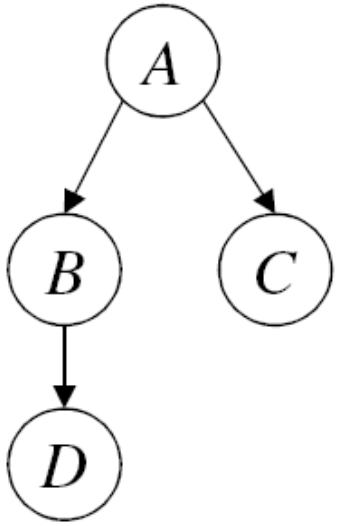


$\mathcal{D}$	A	B	C	D
$\mathbf{d}_1$	?	b_1	c_2	?
$\mathbf{d}_2$	?	b_1	?	d_2
$\mathbf{d}_3$	?	b_2	c_1	d_1
$\mathbf{d}_4$	?	b_2	c_1	d_1
$\mathbf{d}_5$	?	b_1	?	d_2

Our goal is to find ML estimates for the given data set.

Expectation Maximization

Let's assume these initial parameters



A	θ^0_a
a_1	.20
a_2	.80

A	B	$\theta^0_{b\|a}$
a_1	b_1	.75
a_1	b_2	.25
a_2	b_1	.10
a_2	b_2	.90

A	C	$\theta^0_{c\|a}$
a_1	c_1	.50
a_1	c_2	.50
a_2	c_1	.25
a_2	c_2	.75

B	D	$\theta^0_{d\|b}$
b_1	d_1	.20
b_1	d_2	.80
b_2	d_1	.70
b_2	d_2	.30

A Bayesian network inducing a probability distribution $\Pr_{\theta^0}(.)$

Expectation Maximization

The initial estimates θ^0 have the following likelihood:

$$\begin{aligned} \mathrm{L}(\theta^0|\mathcal{D}) &= \prod_{i=1}^{5} \mathrm{Pr}_{\theta^0}(\mathbf{d}_i) \\ &= \mathrm{Pr}_{\theta^0}(b_1, c_2)\mathrm{Pr}_{\theta^0}(b_1, d_2)\mathrm{Pr}_{\theta^0}(b_2, c_1, d_1)\mathrm{Pr}_{\theta^0}(b_2, c_1, d_1)\mathrm{Pr}_{\theta^0}(b_1, d_2) \\ &= (.135)(.184)(.144)(.144)(.184) \\ &= 9.5 \times 10^{-5} \end{aligned}$$

Evaluating the terms in the above product would generally require inference on the Bayesian network.

Expectation Maximization

To illustrate the process of completing a data set, consider again the data set:

$\mathcal{D}$	A	B	C	D
$\mathbf{d}_1$	?	b_1	c_2	?
$\mathbf{d}_2$	?	b_1	?	d_2
$\mathbf{d}_3$	?	b_2	c_1	d_1
$\mathbf{d}_4$	?	b_2	c_1	d_1
$\mathbf{d}_5$	?	b_1	?	d_2

The first case in this data set has two variables with missing values, A and D. Hence, there are four possible completions for this case. Although we do not know which one of these completions is the correct one, we can compute the probability of each completion based on the initial set of parameters we have.

Expectation Maximization

$\mathcal{D}$	A	B	C	D	$\Pr_{\theta^0}(\mathbf{C}_i \mid \mathbf{d}_i)$
$\mathbf{d}_1$	?	b_1	c_2	?	
	a_1	b_1	c_2	d_1	$.111 = \Pr_{\theta^0}(a_1, d_1 \mid b_1, c_2)$
	a_1	b_1	c_2	d_2	.444
	a_2	b_1	c_2	d_1	.089
	a_2	b_1	c_2	d_2	.356
$\mathbf{d}_2$	?	b_1	?	d_2	
	a_1	b_1	c_1	d_2	$.326 = \Pr_{\theta^0}(a_1, c_1 \mid b_1, d_2)$
	a_1	b_1	c_2	d_2	.326
	a_2	b_1	c_1	d_2	.087
	a_2	b_1	c_2	d_2	.261
$\mathbf{d}_3$	?	b_2	c_1	d_1	
	a_1	b_2	c_1	d_1	$.122 = \Pr_{\theta^0}(a_1 \mid b_2, c_1, d_1)$
	a_2	b_2	c_1	d_1	.878
$\mathbf{d}_4$	?	b_2	c_1	d_1	
	a_1	b_2	c_1	d_1	$.122 = \Pr_{\theta^0}(a_1 \mid b_2, c_1, d_1)$
	a_2	b_2	c_1	d_1	.878
$\mathbf{d}_5$	?	b_1	?	d_2	
	a_1	b_1	c_1	d_2	$.326 = \Pr_{\theta^0}(a_1, c_1 \mid b_1, d_2)$
	a_1	b_1	c_2	d_2	.326
	a_2	b_1	c_1	d_2	.087
	a_2	b_1	c_2	d_2	.261

(a) completed data set, with expected values of completed cases

A	B	C	D	$\Pr_{\mathcal{D},\theta^0}(.)$
a_1	b_1	c_1	d_1	0
a_1	b_1	c_1	d_2	.130
a_1	b_1	c_2	d_1	.022
a_1	b_1	c_2	d_2	.219
a_1	b_2	c_1	d_1	.049
a_1	b_2	c_1	d_2	0
a_1	b_2	c_2	d_1	0
a_1	b_2	c_2	d_2	0
a_2	b_1	c_1	d_1	0
a_2	b_1	c_1	d_2	.035
a_2	b_1	c_2	d_1	.018
a_2	b_1	c_2	d_2	.176
a_2	b_2	c_1	d_1	.351
a_2	b_2	c_1	d_2	0
a_2	b_2	c_2	d_1	0
a_2	b_2	c_2	d_2	0

(b) expected empirical distribution

Expectation Maximization

There are three occurrences of the instantiation a_1, b_1, c_2, d_2 in the completed data set, which result from completing the cases $\mathbf{d}_1, \mathbf{d}_2$ and $\mathbf{d}_5$.

The probability of seeing these completions is given by:

$$\begin{aligned}
\Pr_{\mathcal{D},\theta^0}(a_1, b_1, c_2, d_2) &= \frac{\Pr_{\theta^0}(a_1, d_2|\mathbf{d}_1) + \Pr_{\theta^0}(a_1, c_2|\mathbf{d}_2) + \Pr_{\theta^0}(a_1, c_2|\mathbf{d}_5)}{N} \\
&= \frac{.444 + .326 + .326}{5} \\
&= .219
\end{aligned}$$

Note here that we are using $\Pr_{\mathcal{D},\theta^0}(.)$ to denote the expected empirical distribution based on parameters θ^0

Expectation Maximization

The expected empirical distribution of data set $\mathcal{D}$ under parameters θ^k is defined as follows

$$\Pr_{\mathcal{D},\theta^k}(\alpha) \stackrel{def}{=} \frac{1}{N} \sum_{\mathbf{d}_i, \mathbf{c}_i \models \alpha} \Pr_{\theta^k}(\mathbf{c}_i|\mathbf{d}_i),$$

where α is an event and $\mathbf{C}_i$ are the variables with missing values in case $\mathbf{d}_i$.

Recall that $\mathbf{d}_i, \mathbf{c}_i \models \alpha$ means that event α is satisfied by complete case $\mathbf{d}_i, \mathbf{c}_i$. Hence, we are summing $\Pr_{\theta^k}(\mathbf{c}_i|\mathbf{d}_i)$ for all cases $\mathbf{d}_i$ and their completions $\mathbf{c}_i$ that satisfy event α.

Expectation Maximization

When the data set is complete, $\Pr_{\mathfrak{D},\theta^k}(.)$ reduces to the empirical distribution $\Pr_{\mathfrak{D}}(.)$ which is independent of parameters θ^k.

Moreover, $N \cdot \Pr_{\mathfrak{D},\theta^k}(\mathbf{x})$ is called the expected count of instantiation $\mathbf{x}$ in data set $\mathfrak{D}$, just as $N \cdot \Pr_{\mathfrak{D}}(\mathbf{x})$ represents the count of instantiation $\mathbf{x}$ in a complete data set $\mathfrak{D}$.

Expectation Maximization

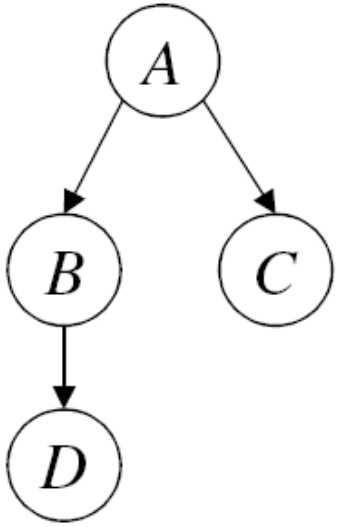
We can now use this expected empirical distribution to estimate parameters, just as we did for complete data.

For example, we have the following estimate for parameter $\theta_{c_1|a_2}$:

$$\theta^1_{c_1|a_2} = \Pr_{\mathcal{D},\theta^0}(c_1|a_2) = \frac{\Pr_{\mathcal{D},\theta^0}(c_1, a_2)}{\Pr_{\mathcal{D},\theta^0}(a_2)} = .666$$

Expectation Maximization

Parameters after one iteration



A	θ^1_a
a_1	.420
a_2	.580

A	B	$\theta^1_{b\|a}$
a_1	b_1	.883
a_1	b_2	.117
a_2	b_1	.395
a_2	b_2	.605

A	C	$\theta^1_{c\|a}$
a_1	c_1	.426
a_1	c_2	.574
a_2	c_1	.666
a_2	c_2	.334

B	D	$\theta^1_{d\|b}$
b_1	d_1	.067
b_1	d_2	.933
b_2	d_1	1.00
b_2	d_2	0.00

A Bayesian network inducing a probability distribution $\Pr_{\theta^1}(.)$

Expectation Maximization

The new estimates θ^1 have likelihood:

$$\begin{aligned} \mathrm{L}(\theta^1|\mathfrak{D}) &= \prod_{i=1}^{5} \mathrm{Pr}_{\theta^1}(\mathbf{d}_i) \\ &= (.290)(.560)(.255)(.255)(.560) \\ &= 5.9 \times 10^{-3} \\ &> \mathrm{L}(\theta^0|\mathfrak{D}) \end{aligned}$$

The new estimates have a higher likelihood than the initial ones we started with. This holds more generally as we will now show.

Expectation Maximization

The EM estimates for data set $\mathcal{D}$ and parameters θ^k

$$\theta^{k+1}_{x|\mathbf{u}} \stackrel{def}{=} \Pr_{\mathcal{D},\theta^k}(x|\mathbf{u})$$

EM estimates are based on the expected empirical distribution, just as our estimates for complete data were based on the empirical distribution. We now have the following key result.

Expectation Maximization

EM estimates satisfy the following property

$$\mathrm{LL}(\theta^{k+1}|\mathcal{D}) \geq \mathrm{LL}(\theta^{k}|\mathcal{D})$$

This is a corollary of Theorems to be discussed later, which characterize the EM algorithm and also explain its name

Expectation Maximization

EM estimates can be computed without constructing the expected empirical distribution.

The expected empirical distribution of data set $\mathcal{D}$ given parameters θ^k can be computed as follows

$$\Pr_{\mathcal{D},\theta^k}(\alpha) = \frac{1}{N}\sum_{i=1}^{N}\Pr_{\theta^k}(\alpha|\mathbf{d}_i)$$

That is, we simply iterate over the data set cases, while computing the probability of α given each case (i.e., no need to explicitly consider the completion of each case).

Expectation Maximization

The EM estimates for data set $\mathcal{D}$ and parameters θ^k can now be computed as follows:

$$\theta^{k+1}_{x|\mathbf{u}} = \frac{\sum_{i=1}^{N} \Pr_{\theta^k}(x\mathbf{u}|\mathbf{d}_i)}{\sum_{i=1}^{N} \Pr_{\theta^k}(\mathbf{u}|\mathbf{d}_i)}$$

Does not reference the expected empirical distribution.

Equation computes EM estimates by performing inference on a Bayesian network parameterized by the previous parameter estimates θ^k

For example,

$$\theta^{1}_{c_1|a_2} = \frac{\sum_{i=1}^{5} \Pr_{\theta^0}(c_1, a_2|\mathbf{d}_i)}{\sum_{i=1}^{5} \Pr_{\theta^0}(a_2|\mathbf{d}_i)} = \frac{0 + .087 + .878 + .878 + .087}{.444 + .348 + .878 + .878 + .348} = .666$$

Expectation Maximization

EM may converge to different parameters, with different likelihoods, depending on the initial estimates θ^0 that it starts with.

Each iteration of the EM algorithm will have to perform inference on a Bayesian network.

In each iteration, the algorithm computes the probability of each instantiation $x\mathbf{u}$ given each case $\mathbf{d}_i$ as evidence.

All of these computations correspond to posterior marginals over network families.

Expectation Maximization

Recall the log-likelihood function:

$$\mathrm{LL}(\theta|\mathcal{D}) = \sum_{i=1}^{N} \log \Pr_\theta(\mathbf{d}_i)$$

We have seen earlier how one can maximize this function for a complete data set by choosing parameter estimates based on the empirical distribution:

$$\theta_{x|\mathbf{u}} = \Pr_{\mathcal{D}}(x|\mathbf{u})$$

Expectation Maximization

Following result draws a parallel between the two cases of log-likelihood and expected log-likelihood.

EM parameter estimates are the only estimates that maximize the expected log-likelihood function

$$\theta^{k+1} = \underset{\theta}{\text{argmax}}\, \text{ELL}(\theta|\mathcal{D}, \theta^k) \text{ iff } \theta^{k+1}_{x|\mathbf{u}} = \Pr_{\mathcal{D},\theta^k}(x|\mathbf{u})$$

Hence, EM is indeed searching for estimates that maximize the expected log-likelihood function, which also explains its name.

Expectation Maximization

Following result draws a parallel between the two cases of log-likelihood and expected log-likelihood.

EM parameter estimates are the only estimates that maximize the expected log-likelihood function

$$\theta^{k+1} = \underset{\theta}{\text{argmax}}\, \text{ELL}(\theta|\mathcal{D}, \theta^k) \text{ iff } \theta^{k+1}_{x|\mathbf{u}} = \Pr_{\mathcal{D},\theta^k}(x|\mathbf{u})$$

Hence, EM is indeed searching for estimates that maximize the expected log-likelihood function, which also explains its name.

Expectation Maximization

Parameters that maximize the expected log-likelihood function cannot decrease the log-likelihood function

If $\theta^{k+1} = \underset{\theta}{\text{argmax}}\, \text{ELL}(\theta|\mathcal{D}, \theta^k)$, then $\text{LL}(\theta^{k+1}|\mathcal{D}) \geq \text{LL}(\theta^k|\mathcal{D})$

Expectation Maximization

EM is capable of converging to every local maxima of the log-likelihood function

The fixed points of EM are precisely the stationary points of the log-likelihood function.

The EM algorithm is known to converge very slowly if the fraction of missing data is quite large.

Gradient Ascent

Another approach for maximizing the log-likelihood function is to view the problem as one of optimizing a continuous nonlinear function.

This is a widely studied problem, where most of the solutions are based on local search, which starts by assuming some initial value $\theta^0_{x|\mathbf{u}}$ for each parameter $\theta_{x|\mathbf{u}}$, and then move through the parameter space in steps of the form $\theta^{k+1}_{x|\mathbf{u}} = \theta^k_{x|\mathbf{u}} + \delta^k_{x|\mathbf{u}}$

Different algorithms will use different values for the increment $\delta^k_{x|\mathbf{u}}$, yet most of them will use gradient information for determining this increment.

Recall that for a function $f(v_1, \ldots, v_n)$, the gradient is the vector of partial derivatives $\partial f/\partial v_1, \ldots, \partial f/\partial v_n$

When evaluated at a particular point $(v_1, \ldots, v_n)$, the gradient gives the direction of the greatest increase in the value of f

Hence, a direct use of the gradient, called gradient ascent, suggests that we move in the direction of the gradient by incrementing each variable v_i with $\eta \frac{\partial f}{\partial v_i}(v_1, \ldots, v_n)$, where η is a constant known as the learning rate.

For more read in Darwiche book, chapter 17

Learning Network Structure

Our main approach for estimating network parameters has been to search for ML estimates, that is, ones that maximize the probability of observing the given data set.

We will now assume that the structure itself is unknown, and suggest methods for learning it from the given data set.

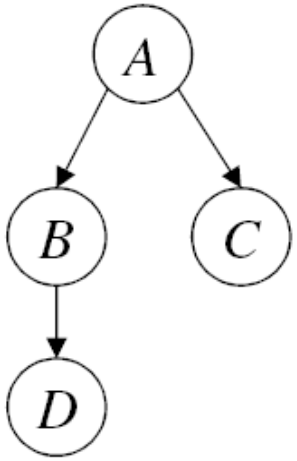
It is natural here to adopt the same approach we adopted for parameter estimation, that is, search for network structures that maximize the probability of observing the given data set.

We will indeed start with this approach first, and then show that it needs some further refinements, leading to a general class of scoring functions for network structures.

Learning Network Structure

Consider the ML estimates for the following structure and data set.

Structure G

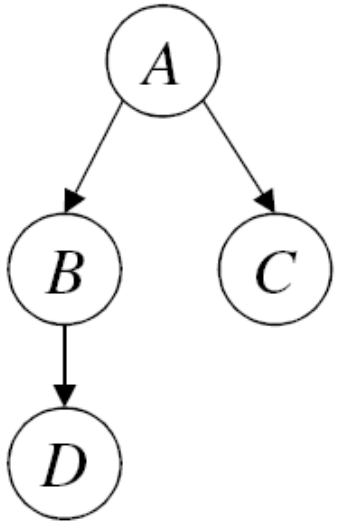


$\mathcal{D}$	A	B	C	D
$\mathbf{d}_1$	a_1	b_1	c_2	d_1
$\mathbf{d}_2$	a_1	b_1	c_2	d_2
$\mathbf{d}_3$	a_1	b_2	c_1	d_1
$\mathbf{d}_4$	a_2	b_1	c_1	d_2
$\mathbf{d}_5$	a_1	b_1	c_2	d_2

The log-likelihood of this network structure is given by:

$$\mathrm{LL}(G|\mathcal{D}) = -13.3$$

Learning Network Structure



A	θ^{ml}_{a}
a_1	$4/5$
a_2	$1/5$

A	B	$\theta^{ml}_{b\|a}$
a_1	b_1	$3/4$
a_1	b_2	$1/4$
a_2	b_1	1
a_2	b_2	0

A	C	$\theta^{ml}_{c\|a}$
a_1	c_1	$1/4$
a_1	c_2	$3/4$
a_2	c_1	1
a_2	c_2	0

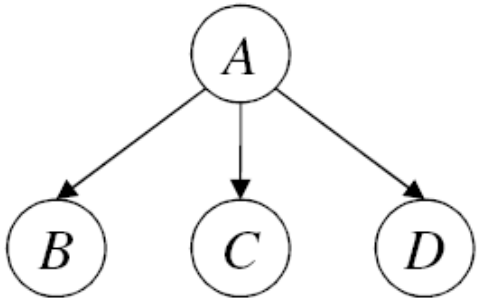
B	D	$\theta^{ml}_{d\|b}$
b_1	d_1	$1/4$
b_1	d_2	$3/4$
b_2	d_1	1
b_2	d_2	0

A network structure with its maximum likelihood parameters.
The log-likelihood of this structure is -13.3

Learning Network Structure

Consider the ML estimates for the following structure and data set.

Structure $G^\star$



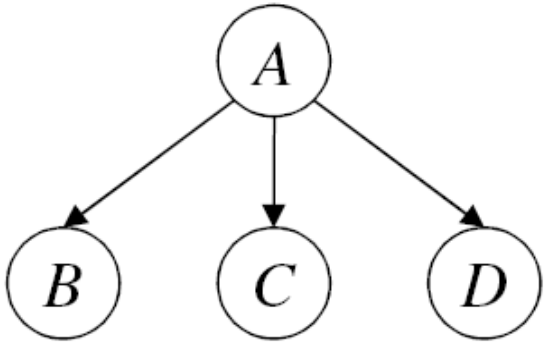
$\mathcal{D}$	A	B	C	D
$\mathbf{d}_1$	a_1	b_1	c_2	d_1
$\mathbf{d}_2$	a_1	b_1	c_2	d_2
$\mathbf{d}_3$	a_1	b_2	c_1	d_1
$\mathbf{d}_4$	a_2	b_1	c_1	d_2
$\mathbf{d}_5$	a_1	b_1	c_2	d_2

The log-likelihood of this network structure is given by:

$$\mathrm{LL}(G^\star|\mathcal{D}) = -14.1,$$

which is smaller than the likelihood for structure G.

Learning Network Structure



A	θ^{ml}_{a}
a_1	$4/5$
a_2	$1/5$

A	B	$\theta^{ml}_{b\|a}$
a_1	b_1	$3/4$
a_1	b_2	$1/4$
a_2	b_1	1
a_2	b_2	0

A	C	$\theta^{ml}_{c\|a}$
a_1	c_1	$1/4$
a_1	c_2	$3/4$
a_2	c_1	1
a_2	c_2	0

A	D	$\theta^{ml}_{d\|a}$
a_1	d_1	$1/2$
a_1	d_2	$1/2$
a_2	d_1	0
a_2	d_2	1

A network structure with its maximum likelihood parameters.
The log-likelihood of this structure is -14.1

Learning Tree Structures

We will next present an algorithm for finding ML tree structures in time and space that are quadratic in the number of nodes in the structure.

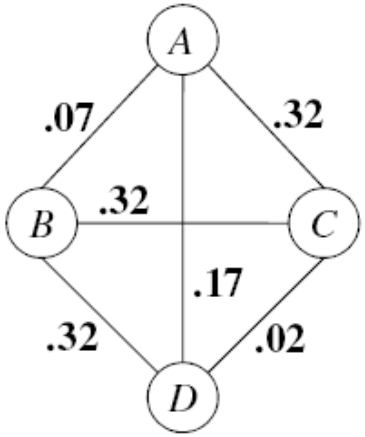
Consider the mutual information between two variables in the empirical distribution:

$$\mathrm{MI}_{\mathfrak{D}}(X,U) \stackrel{def}{=} \sum_{x,u} \Pr_{\mathfrak{D}}(x,u) \log \frac{\Pr_{\mathfrak{D}}(x,u)}{\Pr_{\mathfrak{D}}(x)\Pr_{\mathfrak{D}}(u)}$$

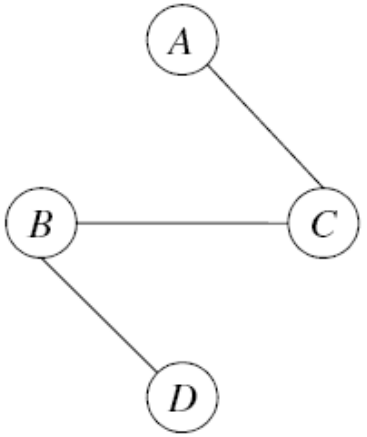
Given a tree structure G with edges $U \rightarrow X$, its score is given by

$$\mathrm{tScore}(G|\mathfrak{D}) \stackrel{def}{=} \sum_{U \rightarrow X} \mathrm{MI}_{\mathfrak{D}}(X,U)$$

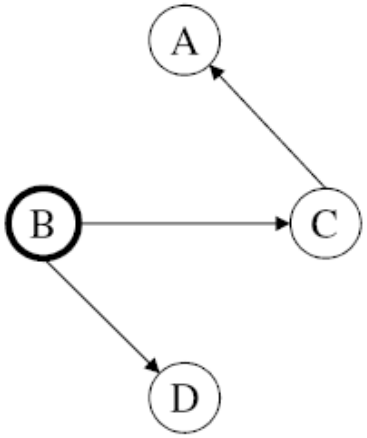
Learning Tree Structures



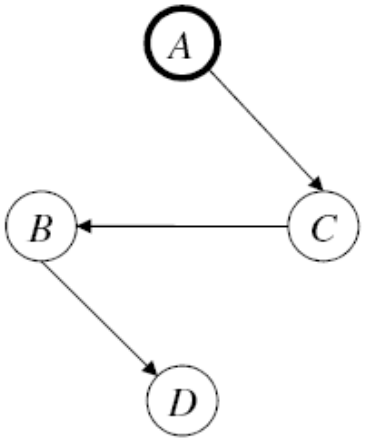
(a) mutual information graph



(b) maximum spanning tree



(c) maximum likelihood tree



(d) maximum likelihood tree

Estimating Parameters from Complete Data

reminder

Log-likelihood decomposes into family-based components

Let G be a network structure and $\mathcal{D}$ be a complete data set of size N. If $X\mathbf{U}$ ranges over the families of structure G, then

$$\mathrm{LL}(G|\mathcal{D}) = -N \sum_{X\mathbf{U}} \mathrm{ENT}_{\mathcal{D}}(X|\mathbf{U}),$$

where $\mathrm{ENT}_{\mathcal{D}}(X|\mathbf{U})$ is the conditional entropy defined as follows:

$$\mathrm{ENT}_{\mathcal{D}}(X|\mathbf{U}) = -\sum_{x\mathbf{u}} \Pr_{\mathcal{D}}(x\mathbf{u}) \log_2 \Pr_{\mathcal{D}}(x|\mathbf{u})$$

Decomposition is critical when learning network structure.

Learning Tree Structures

We can obtain log-likelihood by computing the probability of each case in the data set using any of these tree structures (and its corresponding ML estimates).

We can also use an earlier result, which shows that the log-likelihood corresponds to a sum of terms, one term for each family in the network.

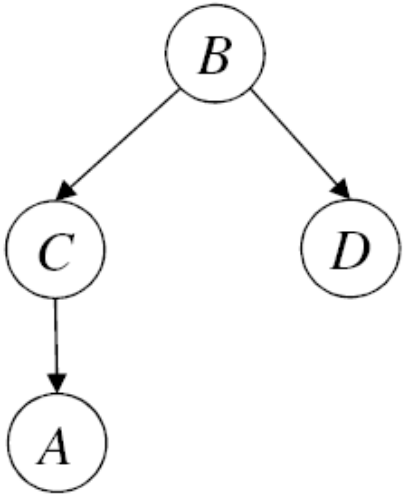
If we consider the tree structure G in (c) above, this result gives:

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

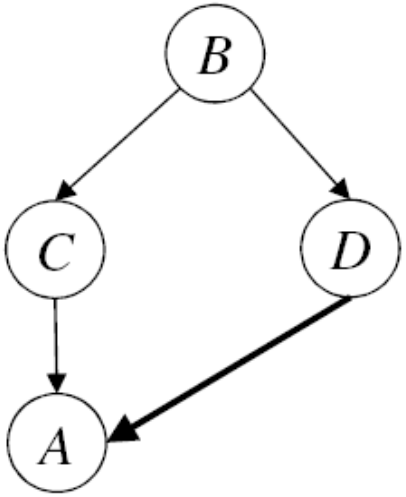
The terms correspond to the families of given tree structure: AC, B, CB and DB.

Learning DAG Structures

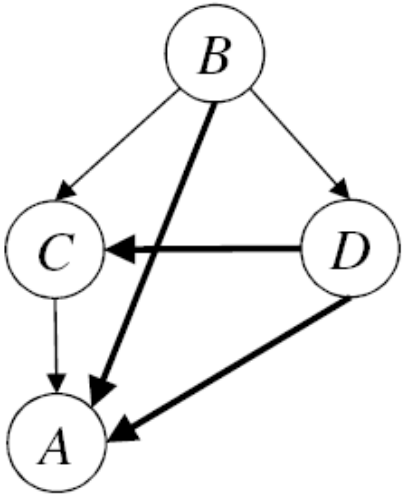
Suppose now that our goal is to find a maximum likelihood structure, but without restricting ourselves to tree structures.



(a) tree



(b) DAG



(c) complete DAG

Learning DAG Structures

Consider the DAG structure in (b) earlier, which is obtained by adding an edge $D \longrightarrow A$ to the tree structure in (a).

The log-likelihood of this DAG is given by:

$$
\begin{aligned}
&\mathrm{LL}(G|\mathcal{D}) \\
&\quad = \quad -N \times (\mathrm{ENT}_{\mathcal{D}}(A|C,D) + \mathrm{ENT}_{\mathcal{D}}(B) + \mathrm{ENT}_{\mathcal{D}}(C|B) + \mathrm{ENT}_{\mathcal{D}}(D|B)) \\
&\quad = \quad -5 \times (0 + .722 + .649 + .649) \\
&\quad = \quad -10.1
\end{aligned}
$$

which is larger than the log-likelihood of the tree in (a).

Learning DAG Structures

Only difference between two likelihoods is the entropy term for variable A, since this is the only variable with different families.

The family of A is AC in the tree, and it is ACD in the DAG. Moreover,

$$\mathrm{ENT}_{\mathfrak{D}}(A|C,D) < \mathrm{ENT}_{\mathfrak{D}}(A|C),$$

and, hence,

$$-\mathrm{ENT}_{\mathfrak{D}}(A|C,D) > -\mathrm{ENT}_{\mathfrak{D}}(A|C),$$

which is why the DAG has a larger log-likelihood than the tree.

More generally

If $\mathbf{U} \subseteq \mathbf{U}^{\star}$, then $\mathrm{ENT}(X|\mathbf{U}) \geq \mathrm{ENT}(X|\mathbf{U}^{\star})$

By adding more parents to a variable, we will never increase its entropy term and, hence, will never decrease the log-likelihood of resulting structure.

Learning DAG Structures

If DAG $G^\star$ is the result of adding edges to DAG G, then

$$\mathrm{LL}(G^\star|\mathcal{D}) \geq \mathrm{LL}(G|\mathcal{D}).$$

If we simply search for a network structure with maximal likelihood, we will end up choosing a complete network structure; that is, a DAG to which no more edges can be added (without introducing directed cycles).[2]

[2] Recall that there are $n!$ complete DAGs over n variables. Each of these DAGs corresponds to a total variable ordering $X_1, \ldots, X_n$ in which variable X_i has $X_1, \ldots, X_{i-1}$ as its parents.

Learning DAG Structures

Complete DAGs are undesirable for a number of reasons:

1. They make no assertions of conditional independence and, hence, their topology does not reveal any properties of the distribution they induce.
2. A complete DAG over n variables has a treewidth of $n - 1$ and is therefore impossible to work with practically.
3. Complete DAGs suffer from the problem of overfitting, which refers to the use of a model that has too many parameters compared to the available data.

Learning DAG Structures

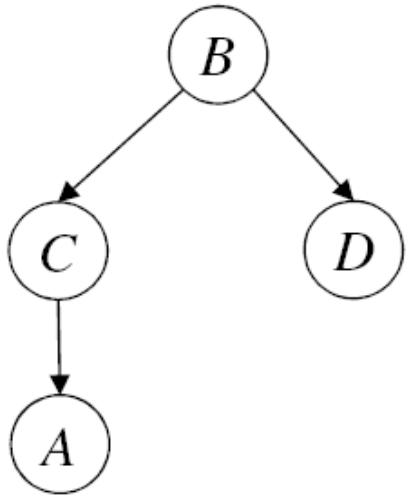
Even though there is no agreed upon solution to the problem of overfitting, all available solutions tend to be based on a common principle known as Occam's razor, which says that one should prefer simpler models over more complex models, others things being equal.

To realize this principle, one needs a measure of model complexity, and a method for balancing the complexity of a model with its data fit.

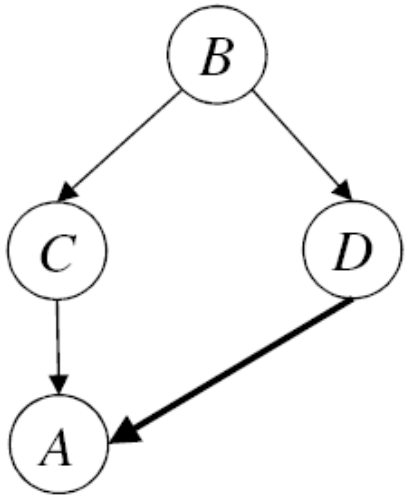
For Bayesian networks (and many other modeling frameworks), model complexity is measured using the number of independent parameters in the model.

Learning DAG Structures

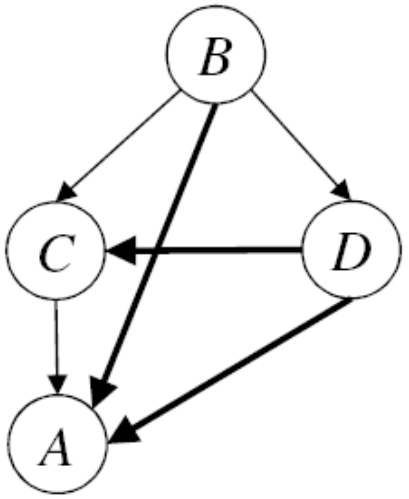
The dimension is the number of free parameters:



(a) dimension 7



(b) dimension 9



(c) dimension 15

Learning DAG Structures

Scoring measures for structure G and data set $\mathcal{D}$ of size N:

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

Note: Score is ≤ 0

The first component of this score, $\text{LL}(G|\mathcal{D})$, is the log-likelihood function we considered before.

The second component, $\psi(N) \cdot ||G||$, is a penalty term that favors simpler models, i.e., ones with a smaller number of independent parameters.

Penalty term has a weight, $\psi(N) \geq 0$, which is a function of the data set size N

Learning DAG Structures

When the penalty weight $\psi(N)$ is a constant that is independent of N, one gets score in which model complexity is a secondary issue.

Log-likelihood function $\mathrm{LL}(G|\mathcal{D})$ grows linearly in the data set size N and will quickly dominate the penalty term.

Model complexity will only be used to distinguish between models that have relatively equal log-likelihood terms.

Scoring measure is known as the Akaike Information Criterion (AIC).

Learning DAG Structures

Another, yet more common, choice of the penalty weight is $\psi(N) = \frac{\log_2 N}{2}$, which leads to a more influential penalty term.

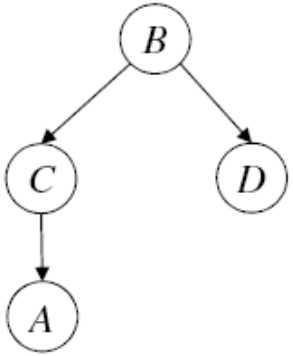
This term grows logarithmically in N, while the log-likelihood term grows linearly in N.

The influence of model complexity will decrease as N grows, allowing the log-likelihood term to eventually dominate the score.

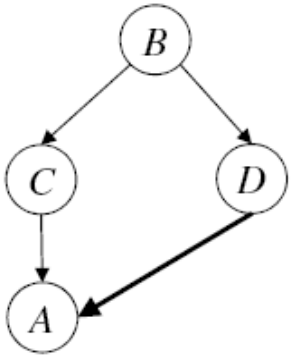
This penalty weight gives rise to the Minimum Description Length (MDL) score:

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

Learning DAG Structures



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



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

MDL prefers first structure even though it has smaller log-likelihood.

Learning DAG Structures

The MDL score is also known as the Bayesian Information Criterion (BIC).

It is sometimes expressed as the negative of the given score, where the goal is to minimize the score instead of maximizing it.

Searching for Network Structure

Searching for a network structure that optimizes a particular score can be quite expensive due to the very large number of structures one may need to consider.

Greedy algorithms tend to be of more practical use when learning network structures.

Systematic search algorithms can also be practical, but only under some conditions.

Both classes of algorithms rely for their efficient implementation on a property that most scoring functions have.

decomposability or modularity: allows one to decompose the score into an aggregate of local scores, one for each network family.

Searching for Network Structure

Score for structure G and data set $\mathcal{D}$ of size N

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

Let $X\mathbf{U}$ range over the families of DAG G

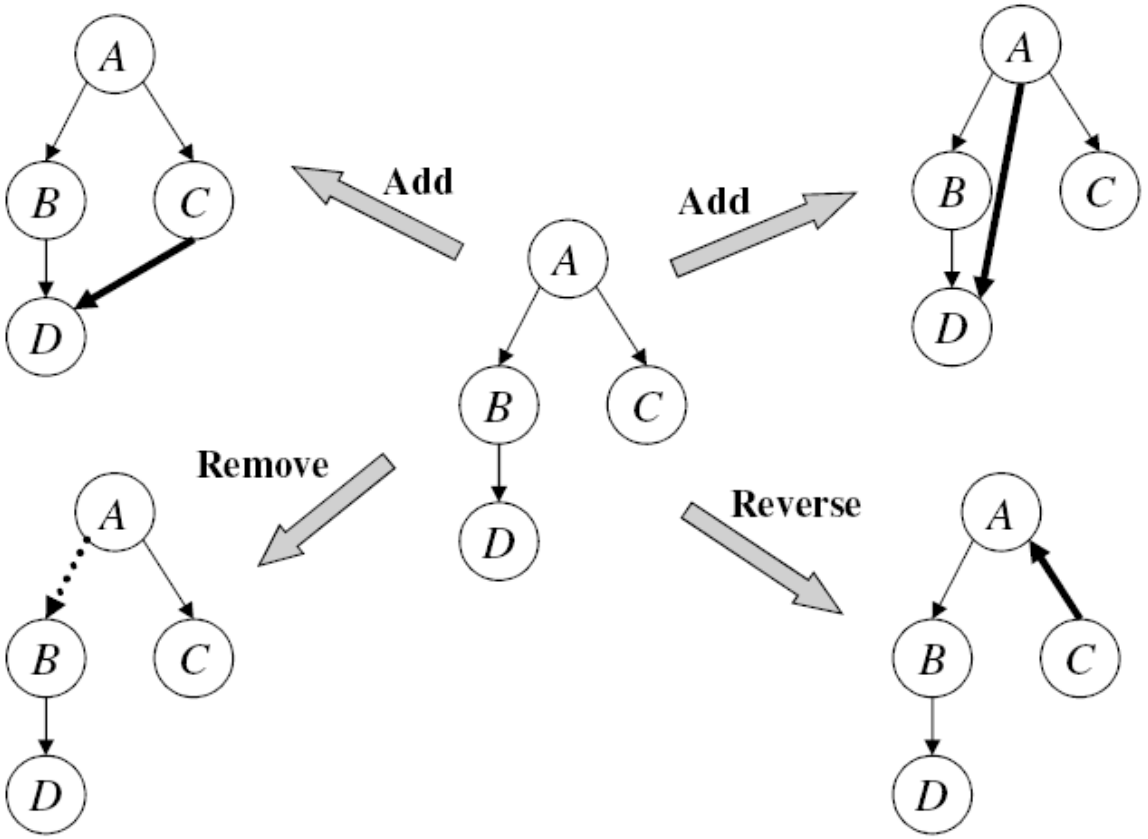
This score can be decomposed as follows:

$$\text{Score}(G|\mathcal{D}) = \sum_{X\mathbf{U}} \text{Score}(X, \mathbf{U}|\mathcal{D}),$$

where

$$\text{Score}(X, \mathbf{U}|\mathcal{D}) \stackrel{def}{=} -N \cdot \text{ENT}_{\mathcal{D}}(X|\mathbf{U}) - \psi(N) \cdot ||X\mathbf{U}||$$

Local Search



Adding or removing an edge will change only one family, while reversing an edge will change only two families.

Hence, the score can always be updated locally as a result of the local network change induced by adding, removing or reversing an edge.

Local Search

The local modifications to the structure are then constrained to: adding an edge, removing an edge, or reversing an edge, while ensuring that the structure remains a DAG.

These local changes to the network structure will also change the score, possibly increasing or decreasing it.

The goal, however, is to commit to the change that will increase the score the most.

If none of the local changes can increase the score, the algorithm will terminate and return the current structure.

Local Search

Local search is not guaranteed to return an optimal network structure, i.e., one that has the largest score.

The only guarantee provided by the algorithm is that the structure it returns will be locally optimal in that no local change can improve its score.

This sub-optimal behavior of local search can usually be improved by techniques such as random restarts.

According to this technique, one would repeat the local search multiple times, each time starting with a different initial network, and then return the network with the best score across all repetitions.

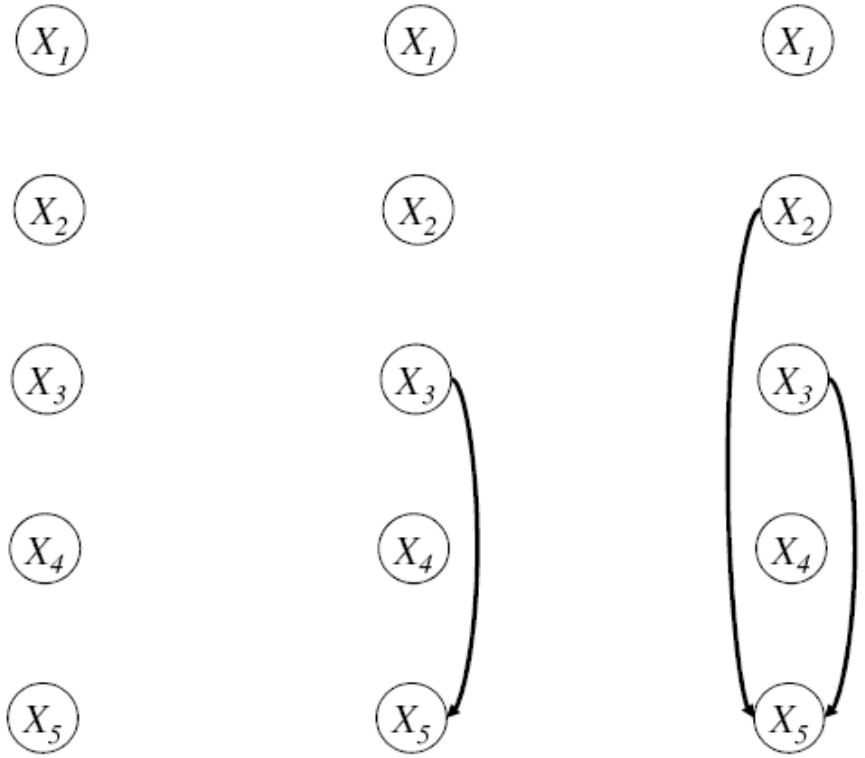
Constraining the Search Space

A common technique for reducing the search space size is to assume a total ordering on network variables and then search only among network structures that are consistent with the chosen order.

If we use the variable order $X_1, \ldots, X_n$, the search process can by viewed as trying to find, for each variable X_i, a set of parents $\mathbf{U}_i \subseteq X_1, \ldots, X_{i-1}$

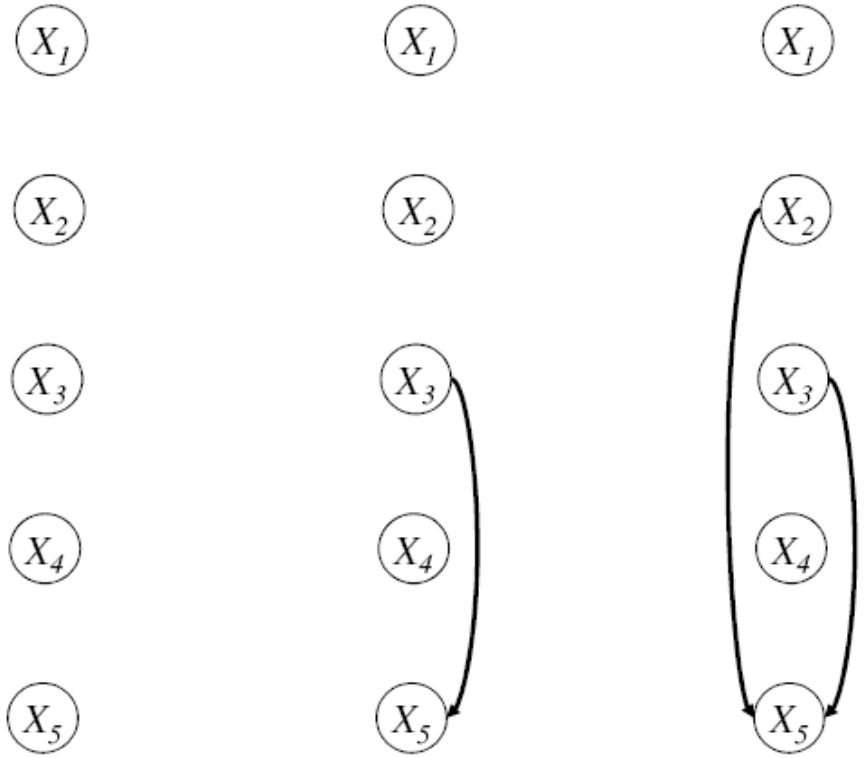
Not only does this technique reduce the size of search space, but it also allows one to decompose the search problem into n independent problems, each concerned with finding a set of parents for some network variable.

Constraining the Search Space



Greedy search for a parent set for variable X_5

Constraining the Search Space



Greedy search for a parent set for variable X_5

Greedy Search

Suppose the goal is to find a set of parents for X_5 from the set of variables $X_1, \ldots, X_4$. The K3 algorithm will start by setting $\mathbf{U}_5$ to the empty set, and then find a variable X_i (if any), $i = 1, \ldots, 4$, that will maximize

$$\text{Score}(X_5, X_i|\mathcal{D}) \geq \text{Score}(X_5|\mathcal{D})$$

Suppose that X_3 happens to be such a variable. The algorithm will then set $\mathbf{U}_5 = \{X_3\}$ and search for another variable X_i in X_1, X_2, X_4 that will maximize

$$\text{Score}(X_5, X_3X_i|\mathcal{D}) \geq \text{Score}(X_5, X_3|\mathcal{D})$$

Suppose again that X_2 happens to be such a variable, leading to the new set of parents $\mathbf{U}_5 = \{X_2, X_3\}$

It may happen that adding X_1 to this set will not increase the score, and neither will adding X_4

In this case, K3 will terminate, returning $\mathbf{U}_5 = \{X_2, X_3\}$ as the parent set for X_5

Greedy Search

K3 is a greedy algorithm that is not guaranteed to identify the optimal set of parents $\mathbf{U}_i$, i.e., the one that maximizes $\text{Score}(X_i, \mathbf{U}_i | \mathcal{D})$

Therefore, it is not uncommon to use the structure obtained by this algorithm as a starting point for other algorithms, such as the local search algorithm discussed earlier, or the optimal search algorithm we shall discuss next.

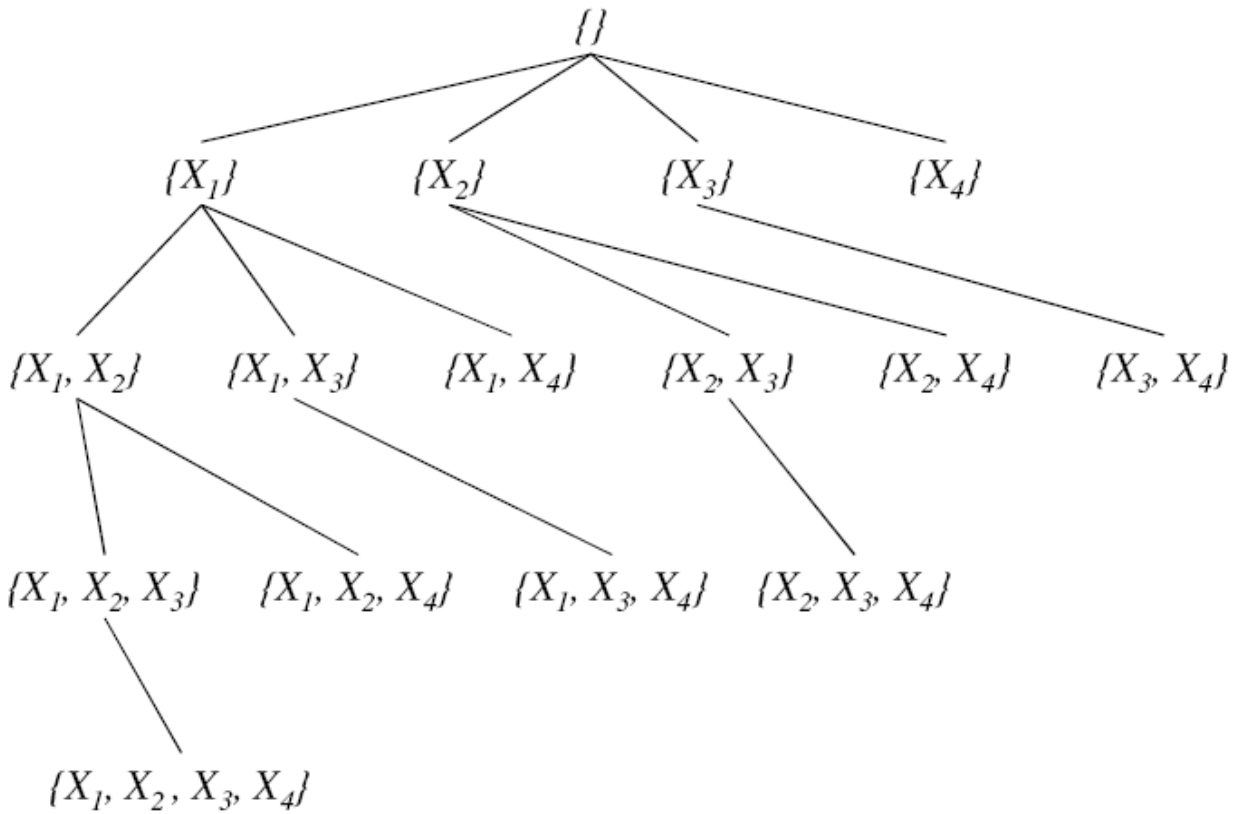
Optimal Search

We will next discuss an optimal search algorithm for network structures, which is based on branch-and-bound depth-first search.

Similar to K3, the algorithm will assume a total order of network variables, $X_1, \ldots, X_n$ and search only among network structures that are consistent with this order.

As mentioned earlier, this allows one to decompose the search process into n independent search problems.

Optimal Search



(a) order X_1, X_2, X_3, X_4

Tree nodes are in one-to-one correspondence with parent sets for X_5. A search tree for variable X_i will have a total of 2^{i-1} nodes, corresponding to the number of subsets one can choose from variables $X_1, \ldots, X_{i-1}$

Optimal Search

One can search the tree using depth-first search, while maintaining the score s of the best parent set visited thus far.

The complexity of this algorithm can be improved on average if one can compute for each search node $\mathbf{U}_i$ an upper bound on $\mathrm{Score}(X_i, \mathbf{U}_i^\star | \mathcal{D})$, where $\mathbf{U}_i \subseteq \mathbf{U}_i^\star$

If the computed upper bound at node $\mathbf{U}_i$ is not better than the best score s obtained thus far, then one can prune $\mathbf{U}_i$ and all nodes below it in the search tree, since none of these parent sets can be better than the one found thus far.

This pruning allows one to escape the exponential complexity in some cases.

The extent of pruning depends on the quality of upper bound used.

Optimal Search

One can search the tree using depth-first search, while maintaining the score s of the best parent set visited thus far.

The complexity of this algorithm can be improved on average if one can compute for each search node $\mathbf{U}_i$ an upper bound on $\mathrm{Score}(X_i, \mathbf{U}_i^\star | \mathcal{D})$, where $\mathbf{U}_i \subseteq \mathbf{U}_i^\star$

If the computed upper bound at node $\mathbf{U}_i$ is not better than the best score s obtained thus far, then one can prune $\mathbf{U}_i$ and all nodes below it in the search tree, since none of these parent sets can be better than the one found thus far.

This pruning allows one to escape the exponential complexity in some cases.

The extent of pruning depends on the quality of upper bound used.

Optimal Search

Upper bound for MDL score

Let $\mathbf{U}_i$ be a parent set, and let $\mathbf{U}_i^+$ be the largest parent set appearing below $\mathbf{U}_i$ in the search tree. If $\mathbf{U}_i^\star$ is a parent set in the tree rooted at $\mathbf{U}_i$, then

$$\mathrm{MDL}(X_i, \mathbf{U}_i^\star | \mathcal{D}) \leq -N \cdot \mathrm{ENT}_{\mathcal{D}}(X_i | \mathbf{U}_i^+) - \psi(N) \cdot ||X_i \mathbf{U}_i||$$

Consider tree in (a). At the search node $\mathbf{U}_5 = \{X_2\}$, we get $\mathbf{U}_5^+ = \{X_2, X_3, X_4\}$. Moreover, $\mathbf{U}_5^\star$ ranges over parent sets $\{X_2\}$, $\{X_2, X_3\}$, $\{X_2, X_4\}$ and $\{X_2, X_3, X_4\}$

Optimal Search

Our discussion on the search for network structures has been restricted to complete data sets.

The main reason for this is computational.

The likelihood of a network structure does not admit a closed form when the data set is incomplete and does not decompose into components.

Algorithms for learning structures with incomplete data will typically involve two searches: an outer search in the space of network structures, and an inner search in the space of network parameters.