02-750 Automation of Biological Research: Active Learning of Bayesian Networks

Ben Lengerich

Carnegie Mellon University

blengeri@cs.cmu.edu

November 22, 2016

Today

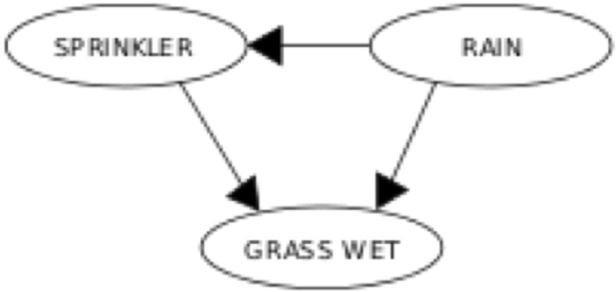
Today

Introduction to PGMs

- A **Probabilistic Graphical Model** (PGM) is a factorized encoding of a multivariate probability distribution $P(X_1, ..., X_n)$
- The encoding consists of a graph $G = (V, E)$, and a set of functions over the vertices and edges of G.

$$P(X_1, ..., X_n) = \prod_{f \in F} f(v|E(v))$$

Introduction to PGMs



- Our goal is to reduce the number of parameters needed to specify the joint probability distribution:

$$P(X_1, ..., X_n) = \prod_{f \in F} f(v|E(v))$$

Introduction to PGMs

There are many types of PGMs, but they can be broken down according to the **graph type**:

- Undirected: Markov Random Fields, Factor Graphs, etc
- Directed: Boltzmann Machines, Hidden Markov Models, Bayesian Networks, etc.

Recall ZLG algorithm used a Gaussian Random Field, which is a kind of Markov Random Field.

Directed vs Undirected PGMs

While **undirected** PGMs have a simpler definition of independence, **directed** PGMs have a few advantages:

- An edge from vertex A to vertex B can indicate that A "causes" B. While this isn't strictly true for learned networks (correlation does not imply casuality), it can help us to construct the graph structure.
- Directed models can encode deterministic relationships.

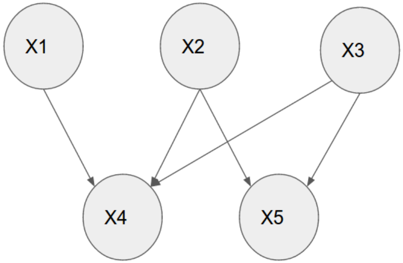
Introduction to PGMs

Today, we will look at a particular type of directed PGM - the Bayesian network.

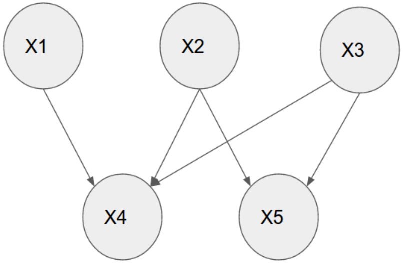
Today

Introduction to Bayesian Networks

- A Bayesian Network is a directed PGM that encodes the joint distribution $P(X_1, ..., X_n)$ by:
  - A **directed** acyclic graph (DAG)
  - Conditional probability distributions, $P(X_i|Pa(X_i))$, where $Pa(X_i)$ is the set of the parents of X_i in the DAG.



Introduction to Bayesian Networks

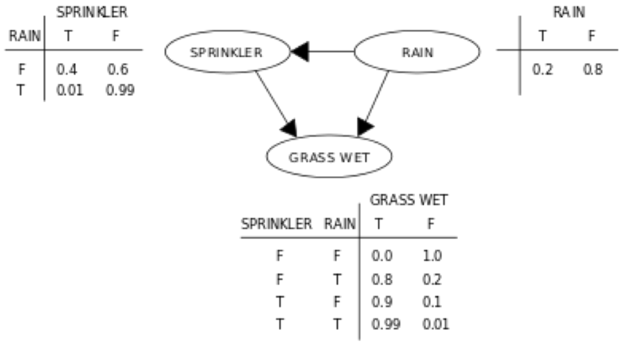


- The joint model is defined by:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i | Pa(X_i))$$

Introduction to Bayesian Networks

- For discrete random variables, the conditional distributions are implemented as conditional probability tables.



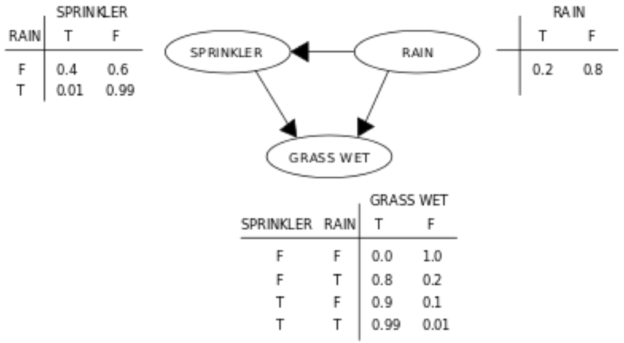
	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

RAIN	
T	F
0.2	0.8

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

Inference in Bayesian Networks

- Like all PGMs, Bayesian Networks let us answer probabilistic questions via inference.
- Example: "What is the probability that it is raining, given the grass is wet?"



	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

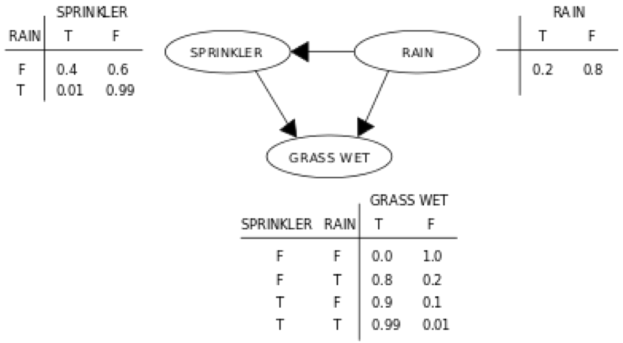
RAIN	
T	F
0.2	0.8

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

Inference in Bayesian Networks

"What is the probability that it is raining, given the grass is wet?"

$$P(R = T|G = T) = \frac{P(R = T, G = T)}{P(G = T)}$$
$$= \frac{\sum_S P(R = T, S, G = T)}{\sum_{S,R} P(R, S, G = T)}$$



	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

RAIN	
T	F
0.2	0.8

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

Learning Bayesian Networks

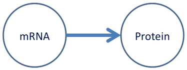
Observational vs Experimental Data

- Observational data:
  - Data obtained by monitoring the unaltered system
  - Reveals correlations between variables
- Experimental Data:
  - Data obtained by monitoring an altered system (e.g., gene knockout)

Learning Bayesian Networks

Many algorithms exist for estimating the parameters of the conditional probability distributions if the topology of the DAG is given.

- A given structure may encode prior knowledge about known causal relationships between variables.
  - e.g. mRNA A "causes" the translation of protein B



Structure Learning

If the topology is not known, we must attempt to learn it (and the parameters) from the data. This is known as the structure learning problem.

Bayesian Network structure learning algorithms have methods for three components:

1. Searching over DAG topologies
2. Estimating parameters for each DAG
3. Scoring each model

Searching over DAG Topologies

Searching over DAG topologies is very challenging because the number of DAGs on a set of n nodes is super-exponential in n:

$$G(n) = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} G(n-k)$$

In practice, Bayesian Network structure learning algorithms stochastically sample DAGs.

Scoring each Model

We score each model by the posterior probability. Given data D, the posterior probability is:

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

where $P(D|G) = \int P(D|G,\Theta)P(\Theta|G)d\Theta$.

Different approaches to defining the priors and likelihoods lead to different scoring metrics.

Identifiability

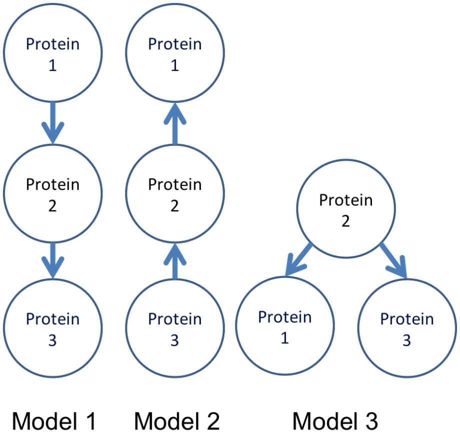
Different graph topologies can encode the same set of conditional independencies. So it is possible for two different DAGs to encode the same distribution, make them non-identifiable.

- Non-identifiability may not be a problem if we simply need a model that makes accurate predictions, but it is a huge problem if we want to infer causal relationships between variables.

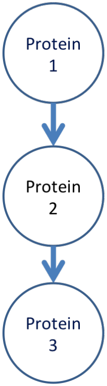
In general, we cannot infer causal relationships.

Conditional Independencies

- Models 1 and 2 are called **indirect effects** models.
- Model 3 is a **common cause** model.



Example 1



Protein 1

0	1
0.5	0.5

Protein 2

Protein 1	0	1
0	0.95	0.05
1	0.05	0.95

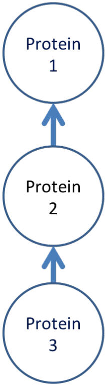
Protein 3

Protein 2	0	1
0	0.95	0.05
1	0.05	0.95

Protein 1	Protein 2	Protein 3	$P(P_1, P_2, P_3)$
0	0	0	0.4512
0	0	1	0.0238
0	1	0	0.0013
0	1	1	0.0238
1	0	0	0.0238
1	0	1	0.0013
1	1	0	0.0238
1	1	1	0.4512

- $P_1 \not\perp P_3$ $\quad P(P_1, P_3) \neq P(P_1)P(P_3)$
- BUT $P_1 \perp P_3 | P_2$ $\quad P(P_1, P_3 | P_2) = P(P_1|P_2)P(P_3|P_2)$

Example 2



Protein 1

Protein 2	0	1
0	0.95	0.05
1	0.05	0.95

Protein 2

Protein 3	0	1
0	0.95	0.05
1	0.05	0.95

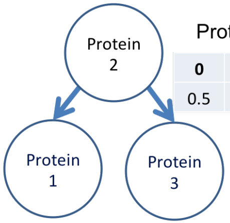
Protein 3

0	1
0.5	0.5

Protein 1	Protein 2	Protein 3	$P(P_1, P_2, P_3)$
0	0	0	0.4512
0	0	1	0.0238
0	1	0	0.0013
0	1	1	0.0238
1	0	0	0.0238
1	0	1	0.0013
1	1	0	0.0238
1	1	1	0.4512

- $P_1 \not\perp P_3$ $\quad P(P_1, P_3) \neq P(P_1)P(P_3)$
- BUT $P_1 \perp P_3 | P_2$ $\quad P(P_1, P_3 | P_2) = P(P_1 | P_2) P(P_3 | P_2)$

Example 3



Protein 2

0	1
0.5	0.5

Protein 1

Protein 2	0	1
0	0.95	0.05
1	0.05	0.95

Protein 3

Protein 2	0	1
0	0.95	0.05
1	0.05	0.95

Protein 1	Protein 2	Protein 3	$P(P_1, P_2, P_3)$
0	0	0	0.4512
0	0	1	0.0238
0	1	0	0.0013
0	1	1	0.0238
1	0	0	0.0238
1	0	1	0.0013
1	1	0	0.0238
1	1	1	0.4512

- $P_1 \not\perp P_3$ $\quad P(P_1, P_3) \neq P(P_1)P(P_3)$
- BUT $P_1 \perp P_3 | P_2$ $\quad P(P_1, P_3 | P_2) = P(P_1|P_2)P(P_3|P_2)$

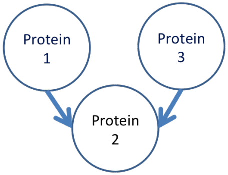
Example 4

Protein 1

0	1
0.5	0.5

Protein 3

0	1
0.5	0.5



Protein 2

P_1	P_3	0	1
0	0	0.95	0.05
0	1	0.5	0.5
1	0	0.5	0.5
1	1	0.05	0.95

Protein 1	Protein 2	Protein 3	$P(P_1, P_2, P_3)$
0	0	0	0.2375
0	0	1	0.0125
0	1	0	0.125
0	1	1	0.125
1	0	0	0.125
1	0	1	0.125
1	1	0	0.0125
1	1	1	0.2375

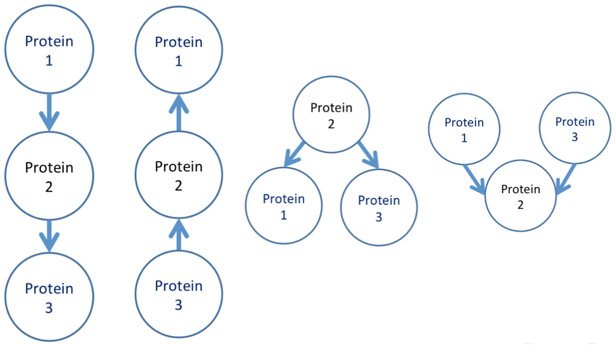
- $P_1 \perp P_3$ $\quad P(P_1, P_3) = P(P_1)P(P_3)$
- BUT $P_1 \not\perp P_3 | P_2$ $\quad P(P_1, P_3|P_2) \neq P(P_1|P_2)P(P_3|P_2)$

V-structure

- Model 4 is an example of a v structure model
- This is a triplet (x, y, z) with converging arcs $x \rightarrow y \leftarrow z$ with x and z not connected.
- V-structures are critical for distinguishing equivalence classes of Bayesian networks.

Equivalence Classes

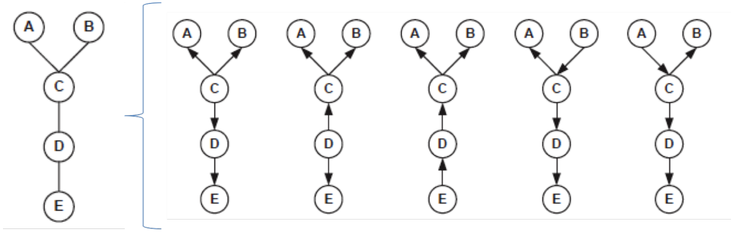
- (Verma and Pearl 1990) Models are equivalent if they have the same:
  - undirected topology
  - V-structure
- Examples:
  - Models 1, 2 and 3 are equivalent
  - Model 4 is in a different equivalence class



$$\begin{tikzcd} \text{Protein 1} \arrow[d] \\ \text{Protein 2} \arrow[d] \\ \text{Protein 3} \end{tikzcd} \quad \begin{tikzcd} \text{Protein 1} \\ \text{Protein 2} \arrow[u] \\ \text{Protein 3} \arrow[u] \end{tikzcd} \quad \begin{tikzcd} & \text{Protein 2} \arrow[dl] \arrow[dr] & \\ \text{Protein 1} & & \text{Protein 3} \end{tikzcd} \quad \begin{tikzcd} \text{Protein 1} \arrow[dr] & & \text{Protein 3} \arrow[dl] \\ & \text{Protein 2} & \end{tikzcd}$$

Equivalence Classes

- Can represent an equivalence class by a partially directed acyclic graph (PDAG).
- In a PDAG, only edges with the same direction all networks of the class are directed in the PDAG.



PDAG   Networks corresponding to the PDAG

Equivalence Classes

- If we only have observational data, the best we can do is to distinguish between equivalence classes (Verma and Pearl 1990)
- That is, if the data is a collection of observations of the joint configurations (P1, P2, and P3), then we will not be able to distinguish networks within the same equivalence class

Equivalence Classes

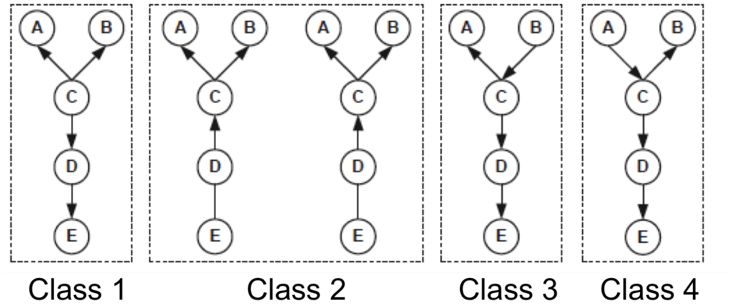
- But, if we have experimental data, we can distinguish between some equivalent models
- Ex: Model 1: $X \rightarrow Y \rightarrow Z$ vs Model 2: $X \leftarrow Y \leftarrow Z$
  - If we manipulate Y then X or Z will change, depending on whether model 1 or 2 is correct
- **Key Idea**: The goal of active learning is to decide which experiments will be the most useful for distinguishing models.

TS Equivalence

- Two models are **Transition Sequence equivalent** if they have the same:
  - Undirected Topology
  - V-Structures
  - Set of parents for manipulated nodes
- Reduces to equivalence if no variables are manipulated

TS Equivalence

- Example: After manipulating C, we can distinguish the TS-equivalence classes:



Key Points

- Given just observational data, we only hope to distinguish between **equivalence classes**
- With experimental data, we can distinguish between **TS equivalence classes**
  - Models in the same TS equivalence class remain indistinguishable even after manipulation

Today

Problem Formulation

- We know that any particular experiment might let us refine a TS-equivalence class into subclasses, and then we can rank these subclasses.
- Different experiments induce different refinements. So our question is: Which experiment induces the best refinement?

Refinements

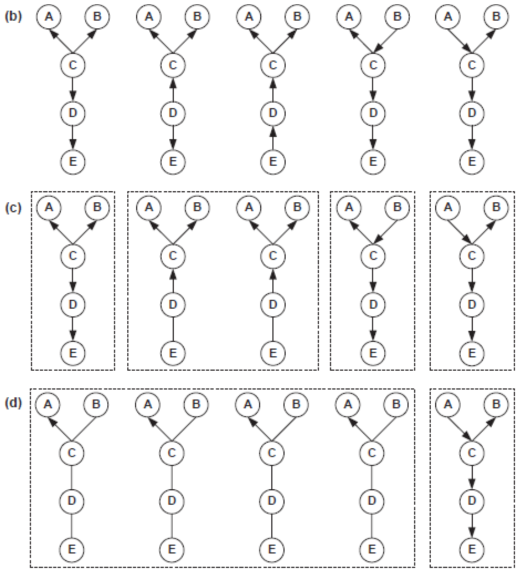
- A good refinement partitions a TS-equivalence class into **many smaller subclasses of roughly equal size**.
- A bad refinement partitions a TS-equivalence class into a **few subclasses of very different sizes**.
- So we judge the refinement by the **entropy** of the distribution of sizes of the induced subclasses.

Refinements

Manipulate C

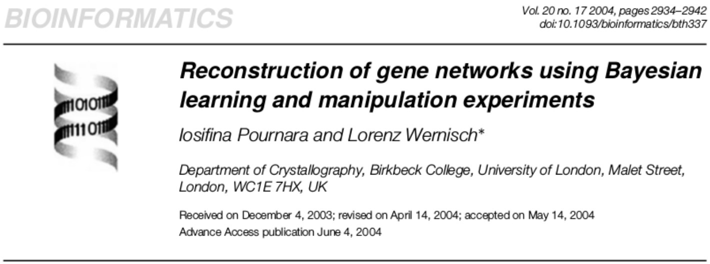
- This refinement is better

Maniuplate A



Today

Paper Overview



BIOINFORMATICS

Vol. 20 no. 17 2004, pages 2934–2942
doi:10.1093/bioinformatics/bth337

Reconstruction of gene networks using Bayesian learning and manipulation experiments

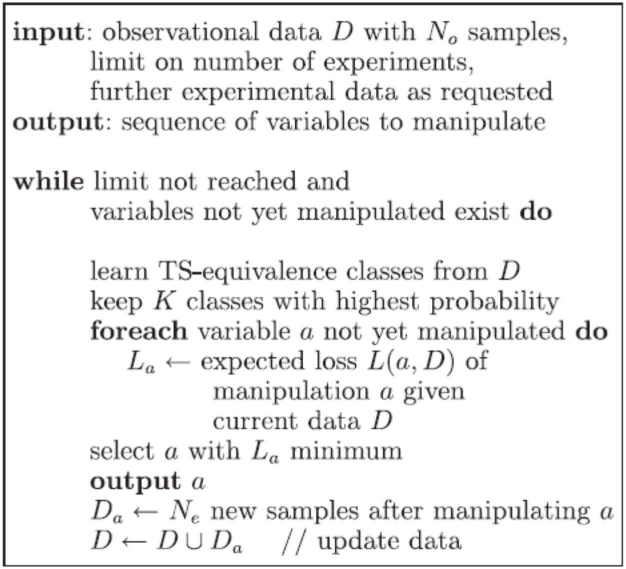
*Iosifina Pournara and Lorenz Wernisch**

Department of Crystallography, Birkbeck College, University of London, Malet Street, London, WC1E 7HX, UK

Received on December 4, 2003; revised on April 14, 2004; accepted on May 14, 2004
Advance Access publication June 4, 2004

- The authors consider the problem of using active learning to estimate the topology and parameters of a Bayesian Network of gene regulation
- Want a generative model (as opposed to discriminative)

Algorithm



```
input: observational data $D$ with $N_o$ samples,
       limit on number of experiments,
       further experimental data as requested
output: sequence of variables to manipulate

while limit not reached and
       variables not yet manipulated exist do

       learn TS-equivalence classes from $D$
       keep $K$ classes with highest probability
       foreach variable $a$ not yet manipulated do
           $L_a \leftarrow$ expected loss $L(a, D)$ of
                 manipulation $a$ given
                 current data $D$
       select $a$ with $L_a$ minimum
       output $a$
       $D_a \leftarrow N_e$ new samples after manipulating $a$
       $D \leftarrow D \cup D_a$     // update data
```

Loss Function

The loss function for manipulating variable a is given by

$$L(a, D) = E_g(log|E_{j(g)}^{i(g)}|) = \sum_g p_{i(g)} log|E_{j(g)}^{i(g)}|$$

$$= \sum_{i=1}^{K} \sum_{j=1}^{n_i} p_i |E_j^i| log|E_j^i|$$

Loss Function

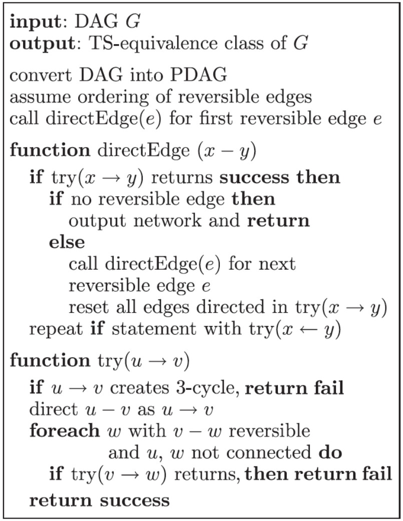
The loss function for manipulating variable *a* is given by

$$L(a, D) = E_g(log|E_{j(g)}^{i(g)}|) = \sum_g p_{i(g)} log|E_{j(g)}^{i(g)}|$$

$$= \sum_{i=1}^{K} \sum_{j=1}^{n_i} p_i |E_j^i| log|E_j^i|$$

The expected loss associated with manipulating variable *a* is a function of the **probability of a given equivalence class**, p_i based on the data and the scoring function, as well as **the size of the induced subclasses**, E_j^i.

Enumeration Algorithm



```
input: DAG $G$
output: TS-equivalence class of $G$

convert DAG into PDAG
assume ordering of reversible edges
call directEdge($e$) for first reversible edge $e$

function directEdge ($x - y$)
    if try($x \rightarrow y$) returns success then
        if no reversible edge then
            output network and return
        else
            call directEdge($e$) for next
            reversible edge $e$
            reset all edges directed in try($x \rightarrow y$)
    repeat if statement with try($x \leftarrow y$)

function try($u \rightarrow v$)
    if $u \rightarrow v$ creates 3-cycle, return fail
    direct $u - v$ as $u \rightarrow v$
    foreach $w$ with $v - w$ reversible
                and $u$, $w$ not connected do
        if try($v \rightarrow w$) returns, then return fail
    return success
```

Fig. 7. Enumeration algorithm for the equivalence class of a DAG

Enumeration Algorithm

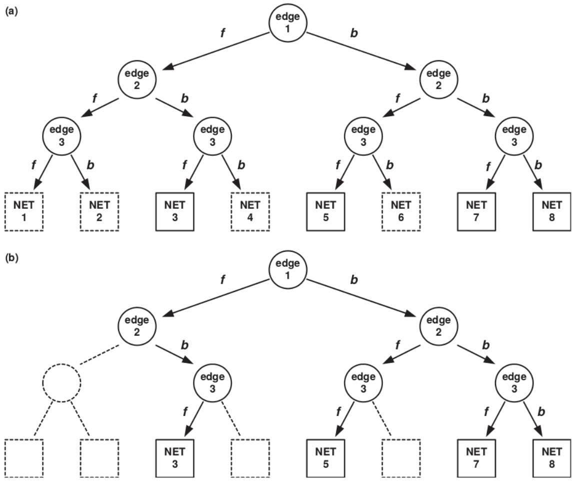


Fig. 8. A tree representation of the naive algorithm (**a**) and the algorithm of Figure 7 (**b**) for enumerating all equivalent networks, applied to the network in Figure 1. Internal nodes (circles) represent edges, leaves (squares) full orientations. Each edge can have one of two directions: f (nodes in alphabetical order) and b (reverse order). Valid orienting moves are indicated by full arrows, invalid moves by dashed lines.

Enumeration Algorithm

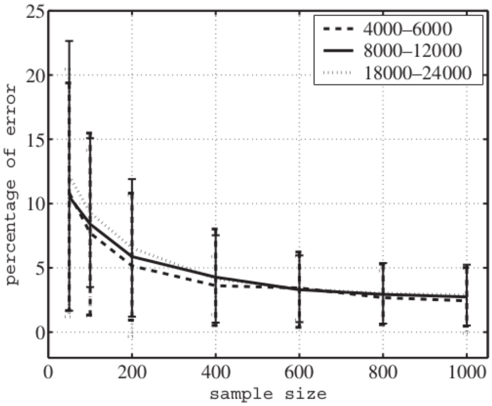


Fig. 6. The average percentage of error between estimated and real size of equivalence classes for 2000 networks with 30 nodes. The results have been split into three groups (4000–6000, 8000–12 000, 18 000–24 000) according to the real size of the equivalence class.

Experiments - Synthetic Data

Random Networks

- 100 target networks generated at random
- 100 observational points

Experiments - Synthetic Data

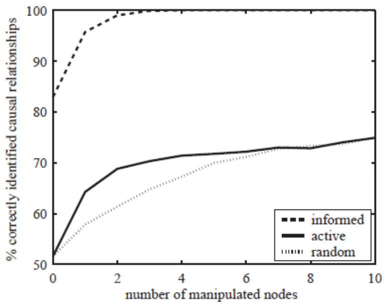


Figure : 10 Discrete Variables

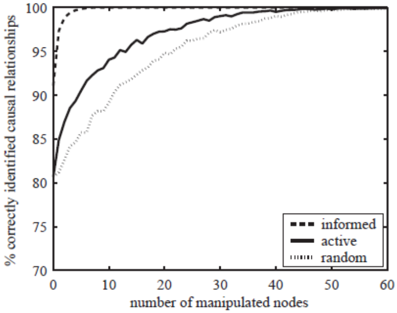


Figure : 60 Continuous Variables

Experiments - Real Data

Cancer Networks

- 5-node network
- 20 observational points
- 100 random runs with different CPTs
- Error is the L_1 edge error

Experiments - Real Data

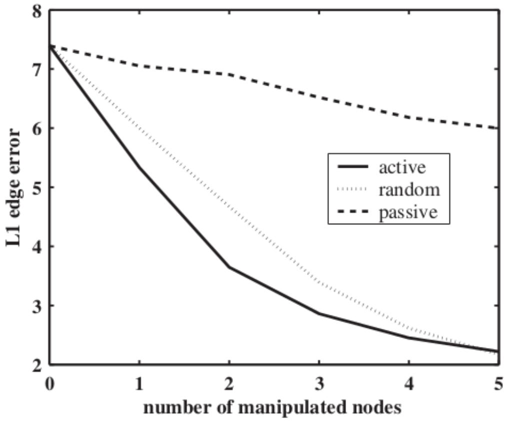


Fig. 4. Edge error E(P) of learning strategies for cancer network: active learning (solid line), learning by random manipulations (dotted line), and passive learning with observations only (dashed line). Lines are averages over 100 runs with randomly generated conditional probability tables.

Today

Summary

- Bayesian Networks are generative PGMs that are often used to model biological phenomena.
  - Their primary limitation is that they cannot model cyclic dependencies.
- The directed topology of a Bayesian Network can imply causal relationships.
- Unfortunately, it is not possible to learn causal models from observational and experimental data.
  - Though we can use prior knowledge to construct.
- Active learning algorithms can efficiently identify equivalence classes of Bayesian Networks after very few experiments.

For Next Time

- Read an be prepared to discuss the two papers listed on the website for Lecture 27:
- Choo et al Reconstructing Causal Biological Networks through Active Learning, PLoS One, 2016 11(3): e0150611. doi:10.1371/journal.pone.0150611
- King et al Functional genomic hypothesis generation and experimentation by a robot scientist , Nature 427, 247-252, 2004