Learning Bayes Networks

6.034

Based on Russell & Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed., 2003
and D. Heckerman. A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models,* M. Jordan, ed.. MIT Press, Cambridge, MA, 1999.

Statistical Learning Task

- Given a set of observations (evidence),
  - find {any/good/best} hypothesis that describes the domain
  - and can predict the data
    - and, we hope, data not yet seen
- ML section of course introduced various learning methods
  - nearest neighbors, decision (classification) trees, naive Bayes classifiers, perceptrons, ...
  - Here we introduce methods that learn (non-naive) Bayes networks, which can exhibit more systematic structure

Characteristics of Learning BN Models

- Benefits
  - Handle incomplete data
  - Can model causal chains of relationships
  - Combine domain knowledge and data
  - Can avoid overfitting
- Two main uses:
  - Find (best) hypothesis that accounts for a body of data
  - Find a probability distribution over hypotheses that permits us to predict/interpret future data

An Example

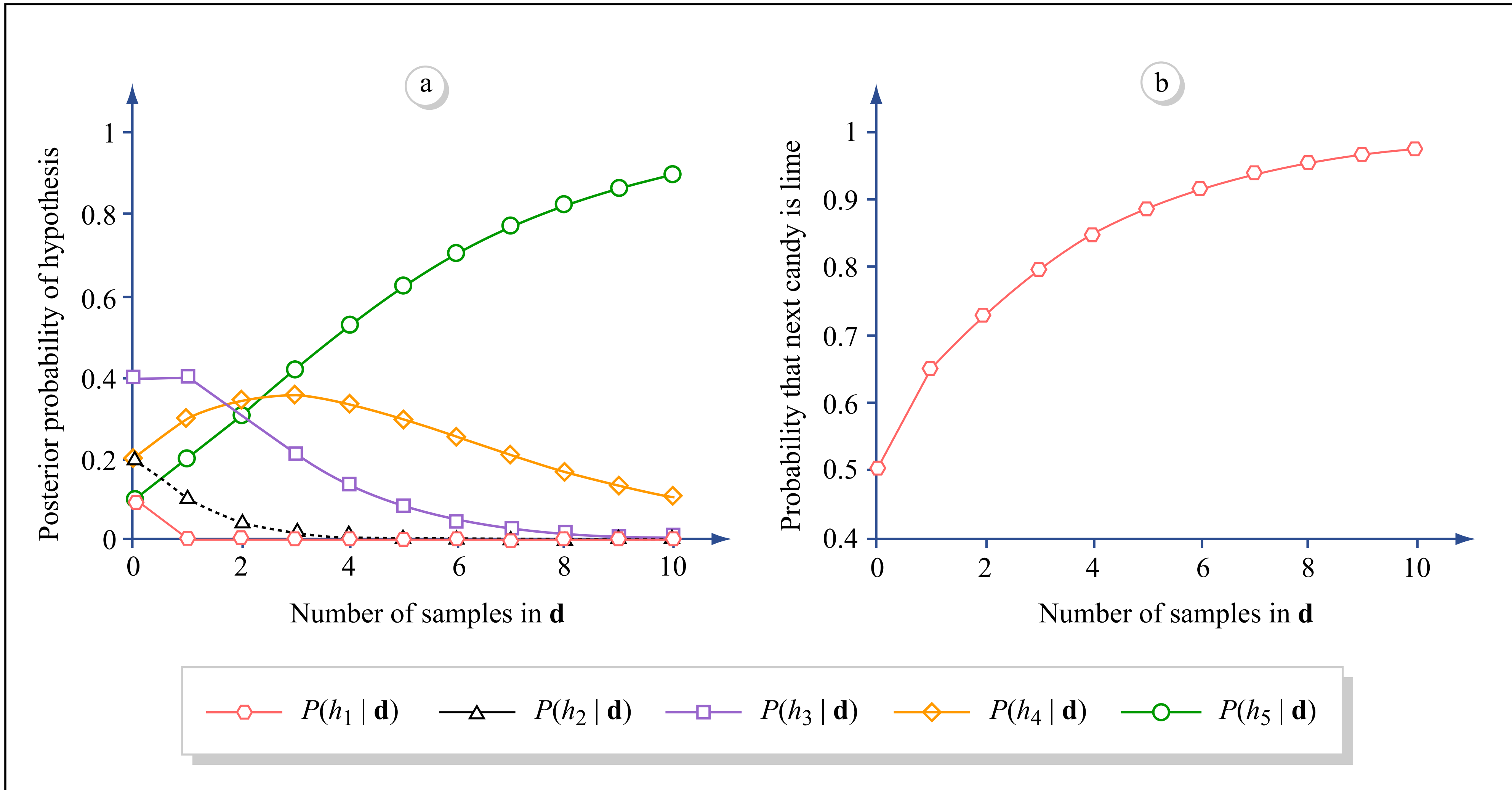
- Surprise Candy Corp. makes two flavors of candy: *cherry* and *lime*
- Both flavors come in the same opaque wrapper
- Candy is sold in large bags, which have one of the following distributions of flavors, but are visually indistinguishable:
  - h_1: 100% cherry
  - h_2: 75% cherry, 25% lime
  - h_3: 50% cherry, 50% lime
  - h_4: 25% cherry, 75% lime
  - h_5: 100% lime
- Relative prevalence of these types of bags is (.1, .2, .4, .2, .1)
- As we eat our way through a bag of candy, predict the flavor of the next piece; actually a probability distribution.

Bayesian Learning

- Calculate the probability of each hypothesis given the data
  $P(h_i|\boldsymbol{d}) = \alpha P(\boldsymbol{d}|h_i)P(h_i)$
- To predict the probability distribution over an unknown quantity, *X*,
  $\boldsymbol{P}(X|\boldsymbol{d}) = \sum_i \boldsymbol{P}(X|\boldsymbol{d}, h_i)\boldsymbol{P}(h_i|\boldsymbol{d}) = \sum_i \boldsymbol{P}(X|h_i)P(h_i|\boldsymbol{d})$
- If the observations ***d*** are independent, then
  $P(\boldsymbol{d}|h_i) = \prod_j P(d_j|h_i)$
- E.g., suppose the first 10 candies we taste are all lime
  $P(\boldsymbol{d}|h_3) = 0.5^{10} \approx 0.001$

Learning Hypotheses and Predicting from Them

- (a) probabilities of h_i after *k* *lime* candies; (b) prob. of next *lime*



Images by MIT OpenCourseWare.

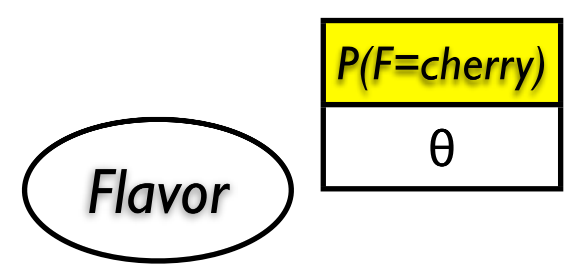
- MAP prediction: predict just from most probable hypothesis
  - After 3 limes, h_5 is most probable, hence we predict *lime*
  - Even though, by (b), it's only 80% probable

Observations

- Bayesian approach asks for prior probabilities on *hypotheses*!
  - Natural way to encode bias against complex hypotheses: make their prior probability very low
- Choosing h_{MAP} to maximize $P(h_i|\boldsymbol{d}) = \alpha P(\boldsymbol{d}|h_i)P(h_i)$
  - is equivalent to minimizing $-\log P(\boldsymbol{d}|h_i) - \log P(h_i)$
  - but from our earlier discussion of entropy as a measure of information, these two terms are
    - # of bits needed to describe the data given hypothesis
    - # bits needed to specify the hypothesis
  - Thus, MAP learning chooses the hypothesis that maximizes *compression* of the data; *Minimum Description Length* principle
- Assuming uniform priors on hypotheses makes MAP yield h_{ML}, the *maximum likelihood hypothesis*, which maximizes $P(h_i|\boldsymbol{d}) = \alpha P(\boldsymbol{d}|h_i)$

ML Learning (Simplest)

- Surprise Candy Corp. is taken over by new management, who abandon their former bagging policies, but do continue to mix together θ cherry and (1-θ) lime candies in large bags
- Their policy is now represented by a *parameter* θ ∈ [0,1], and we have a continuous set of hypotheses, h_θ
- Assuming we taste *N* candies, of which *c* are cherry and *l*=*N*–*c* lime
  $P(\boldsymbol{d}|h_\theta) = \prod_{j=1}^{N} P(d_j|h_\theta) = \theta^c \cdot (1-\theta)^l$
- For convenience, we maximize the log likelihood
  $L(\boldsymbol{d}|h_\theta) = \log P(\boldsymbol{d}|h_\theta) = \sum_{j=1}^{N} \log P(d_j|h_\theta) = c \log \theta + l \log(1-\theta)$
- Setting the derivative = 0,
  $\frac{dL(\boldsymbol{d}|h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0 \quad \Rightarrow \quad \theta = \frac{c}{c+l} = \frac{c}{N}$
- Surprise!
- But need Laplace correction for small data sets



ML Parameter Learning

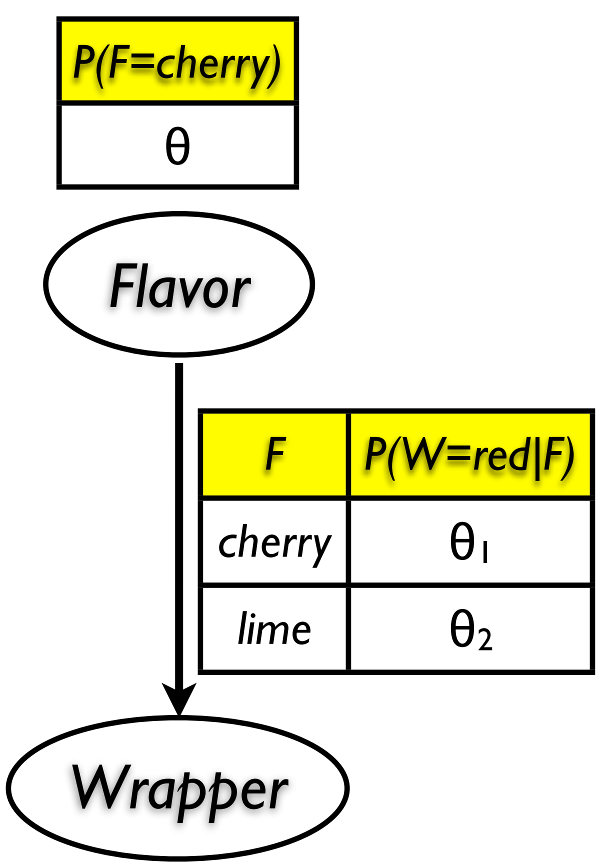
- Suppose the new SCC management decides to give a hint of the candy flavor by (probabilistically) choosing wrapper colors

$$
\begin{aligned}
&P(F=\text{cherry}, W=\text{green}|h_{\theta,\theta_1,\theta_2}) \\
&\quad = \quad P(F=\text{cherry}|h_{\theta,\theta_1,\theta_2})P(W=\text{green}|F=\text{cherry}, h_{\theta,\theta_1,\theta_2}) \\
&\quad = \quad \theta \cdot (1-\theta_1)
\end{aligned}
$$

- Now we unwrap *N* candies of which
c are cherries, with r_c in red wrappers and g_c in green,
and *l* are limes, with r_l in red wrappers and g_l in green

$$
\begin{aligned}
P(\boldsymbol{d}|h_{\theta,\theta_1,\theta_2}) &= \theta^c(1-\theta)^l \cdot \theta_1^{r_c}(1-\theta_1)^{g_c} \cdot \theta_2^{r_l}(1-\theta_2)^{g_l} \\
L &= [c\log\theta + l\log(1-\theta)] \\
&\quad + [r_c\log\theta_1 + g_c\log(1-\theta_1)] \\
&\quad + [r_l\log\theta_2 + g_l\log(1-\theta_2)]
\end{aligned}
$$

$$\theta = c/(c+l), \quad \theta_1 = r_c/(r_c+g_c), \quad \theta_2 = r_l/(r_l+g_l)$$

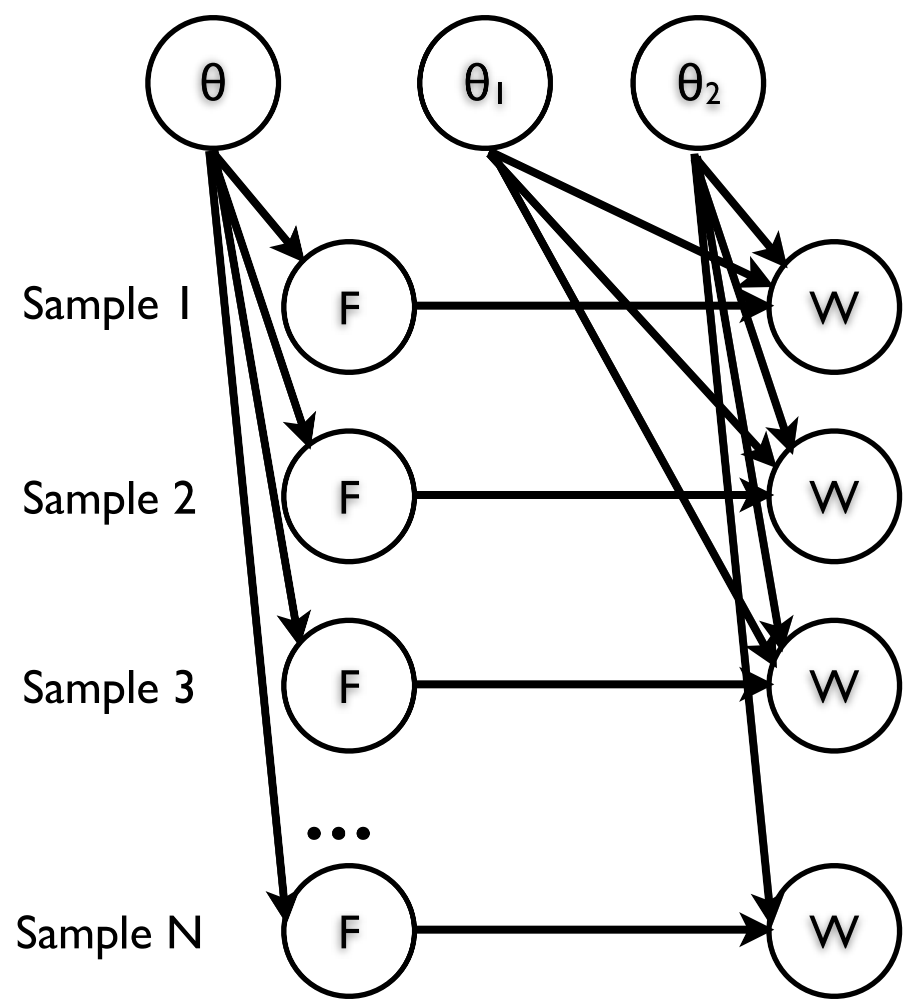


- With complete data, ML learning decomposes into *n* learning problems, one for each parameter

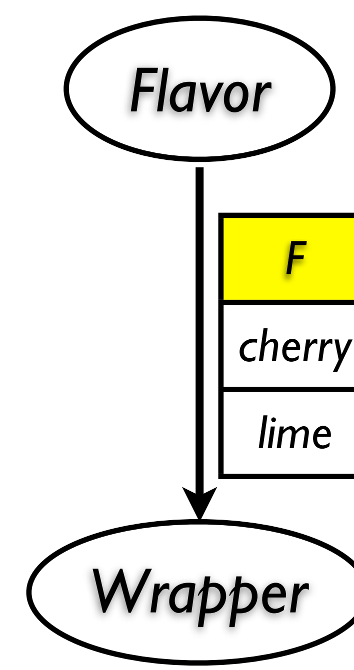
Use BN to learn Parameters

- If we extend BN to continuous variables (essentially, replace $\sum$ by $\int$)
- Then a BN showing the dependence of the observations on the parameters lets us compute (the distributions over) the parameters using just the “normal” rules of Bayesian inference.
- This is efficient if all observations are known
    - Need sampling methods if not

Parameter Independence



P(F=cherry)
θ



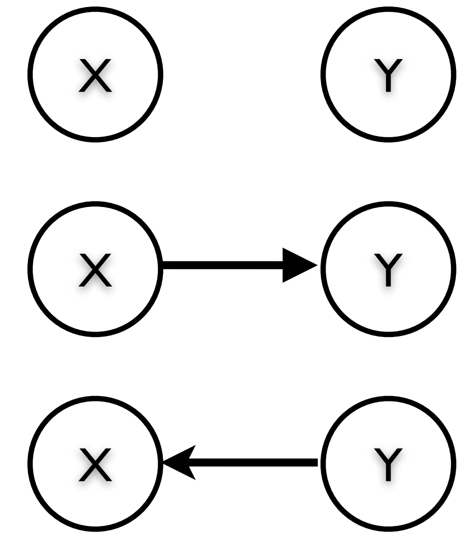
F	P(W=red|F)
cherry	θ_1
lime	θ_2

Learning Structure

- In general, we are trying to determine not only parameters for a known structure but in fact which structure is best
  - (or the probability of each structure, so we can average over them to make a prediction)

Structure Learning

- Recall that a Bayes Network is fully specified by
  - a DAG *G* that gives the (in)dependencies among variables
  - the collection of parameters θ that define the conditional probability tables for each of the $P(x_i|\mathrm{Par}(X_i))$
- Then $P(G|D) = \dfrac{P(D|G)P(G)}{P(D)} \propto P(D|G)P(G)$
- We define the *Bayesian score* as $\log P(D|G) + \log P(G)$
- But $P(D|G) = \int_{\Theta_G} P(D|\theta_G, G)P(\theta_G|G)P(G)d\theta_G$
  - First term: usual marginal likelihood calculation
  - Second term: parameter priors
  - Third term: "penalty" for complexity of graph
- Define a search problem over all possible graphs & parameters



Searching for Models

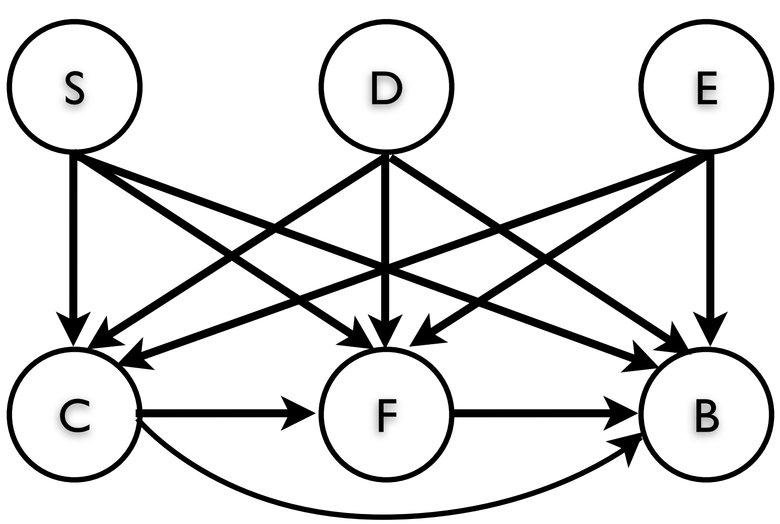
- How many possible DAGs are there for *n* variables?
  - $< 3^{n^2}$ = all possible directed graphs on *n* vars
  - Not all are DAGs
- To get a closer estimate, imagine that we order the variables so that the parents of each var come before it in the ordering. Then
  - there are *n*! possible ordering, and
  - the *j*-th var can have any of the previous vars as a parent

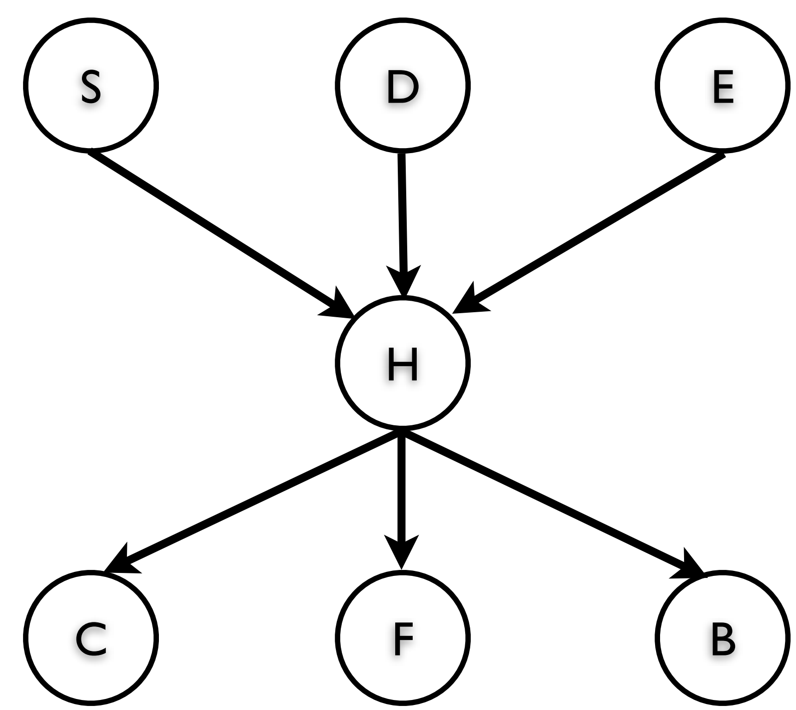
$$n! \prod_{i=1}^{n} 2^{i-1} = n! \cdot 2^{\sum_{i=1}^{n}(i-1)} = O(n! \cdot 2^{n^2})$$

- If we can choose a particular ordering, say based on prior knowledge, then we need consider "merely" $O(2^{n^2})$ models
- If we restrict |Par(X)| to no more than *k*, consider $\leq \sum_{i=1}^{n} \binom{n}{k}$ models; this is actually practical
- Search actions: add, delete, reverse an arc
- Hill-climb on P(D|G) or on P(G|D)
- All "usual" tricks in search: simulated annealing, random restart, ...

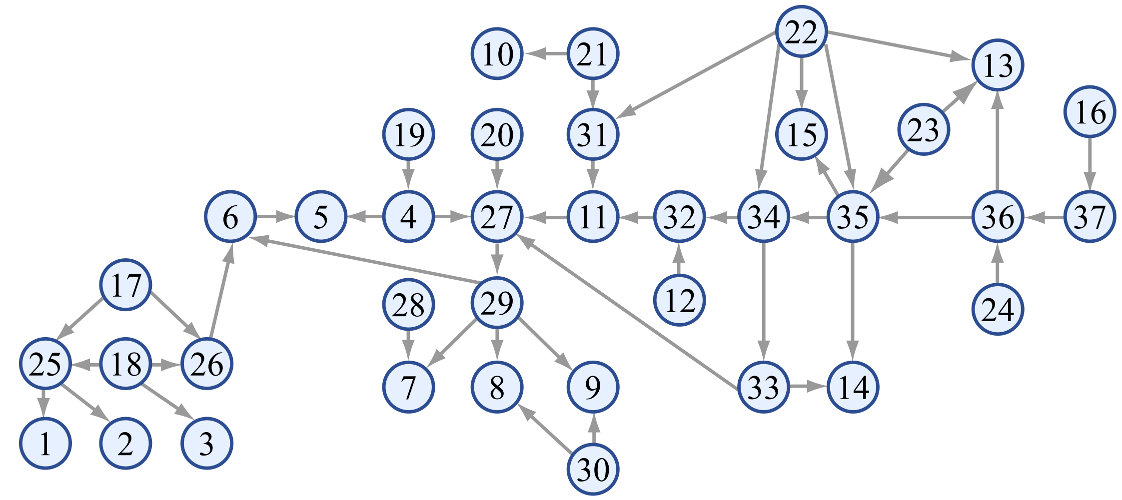
Caution about Hidden Variables

- Suppose you are given a dataset containing data on patients' smoking, diet, exercise, chest pain, fatigue, and shortness of breath
- You would probably learn a model like the one below left
- If you can hypothesize a "hidden" variable (not in the data set), e.g., *heart disease*, the learned network might be much simpler, such as the one below right
- But, there are potentially infinitely many such variables

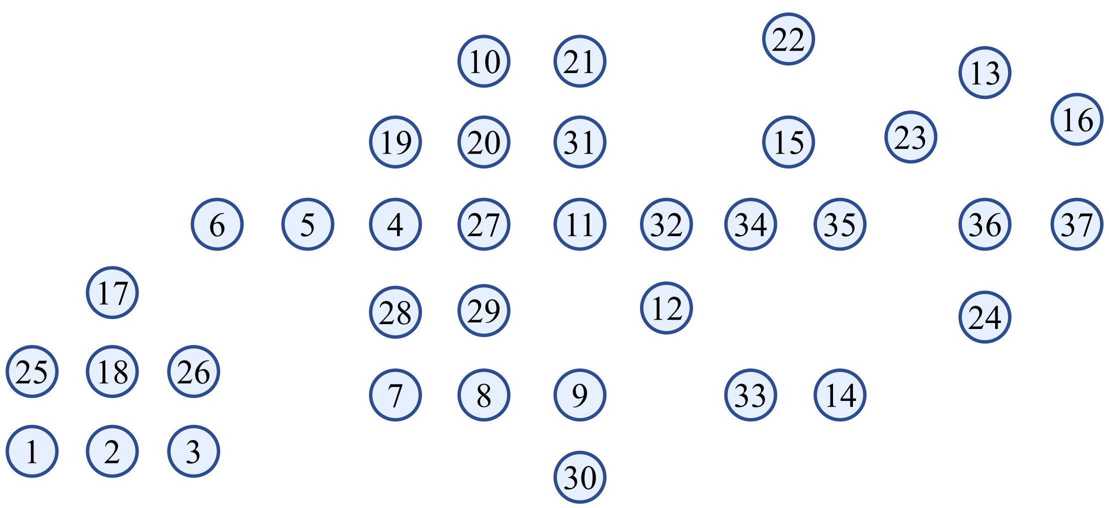




Re-Learning the ALARM Network from 10,000 Samples



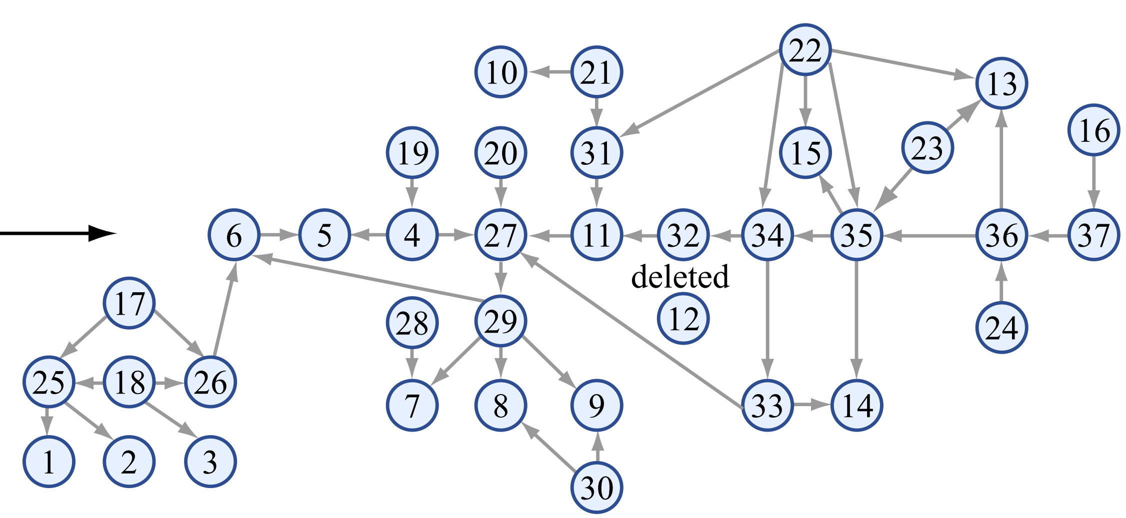
a) Original Network



b) Starting Network Complete independence

case #	x_1	x_2	x_3	. . .	x_{37}
1	3	3	2		4
2	2	2	2	. . .	3
3	1	3	3		3
4	3	2	3		1
	⋮			⋱	
10,000	2	2	2		3

c) Sampled Data



d) Learned Network

Images by MIT OpenCourseWare.

MIT OpenCourseWare
http://ocw.mit.edu

HST.950J / 6.872 Biomedical Computing
Fall 2010

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.