Learning Bayesian Networks (part 2)

Mark Craven and David Page

Computer Sciences 760

Spring 2019

Goals for the lecture

you should understand the following concepts

- the Chow-Liu algorithm for structure search
- structure learning as search
- Kullback-Leibler divergence
- Bayes nets for classification
- naïve Bayes
- tree-augmented naïve Bayes (TAN)

Learning structure + parameters

- number of structures is superexponential in the number of variables
- finding optimal structure is NP-complete problem
- two common options:
  - search very restricted space of possible structures (e.g. networks with tree DAGs)
  - use heuristic search (e.g., greedy hill-climbing)

The Chow-Liu algorithm

- learns a BN with a tree structure that maximizes the likelihood of the training data
- algorithm
  1. compute weight $I(X_i, X_j)$ of each possible edge (X_i, X_j)
  2. find maximum weight spanning tree (MST)
  3. assign edge directions in MST

The Chow-Liu algorithm

1. use mutual information to calculate edge weights

$$I(X,Y) = \sum_{x \in values(X)} \sum_{y \in values(Y)} P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}$$

Parenthetic Asides

- *Kullback-Leibler* (KL) *divergence* provides a distance measure between two distributions, P and Q

$$D_{KL}(P(X)\,\|\,Q(X)) = \sum_x P(x)\log\frac{P(x)}{Q(x)}$$

- mutual information can be thought of as the KL divergence between the distributions

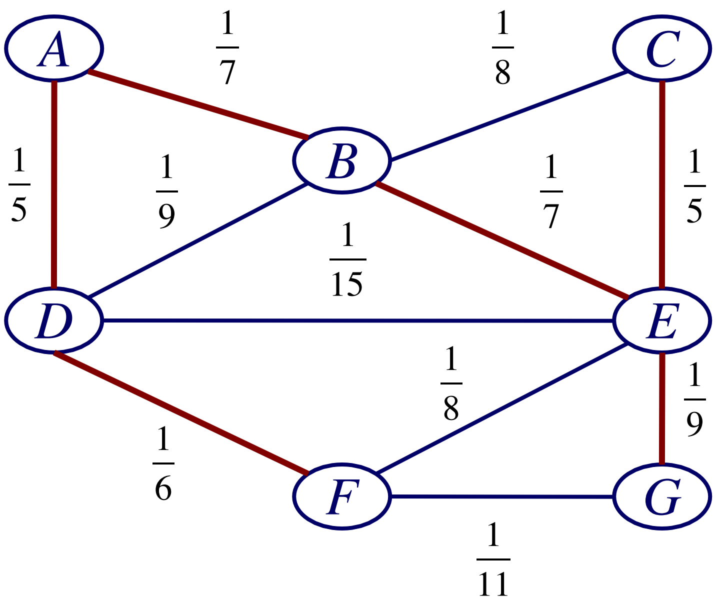
$$P(X,Y)$$

$$P(X)P(Y)$$ (assumes X and Y are independent)

- Mutual Information of X and Y is also same as Information Gain of X for predicting Y (or Y for X)

The Chow-Liu algorithm

2. find maximum weight spanning tree: a maximal-weight tree that connects all vertices in a graph



Prim's algorithm for finding an MST

given: graph with vertices V and edges E

$V_{new} \leftarrow \{\ v\ \}$ where v is an arbitrary vertex from V
$E_{new} \leftarrow \{\ \}$
repeat until $V_{new} = V$
{
    choose an edge (u, v) in E with max weight where u is in V_{new} and v is not
    add v to V_{new} and (u, v) to E_{new}
}
return V_{new} and E_{new} which represent an MST

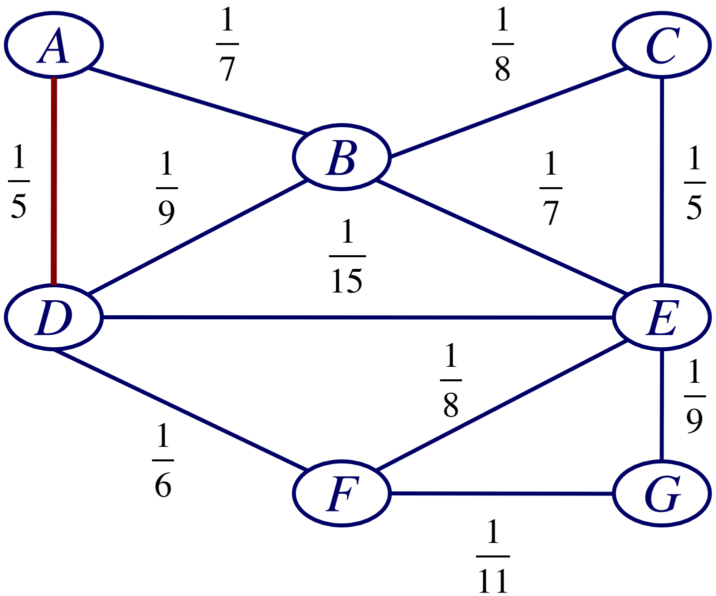
Kruskal’s algorithm for finding an MST

given: graph with vertices V and edges E

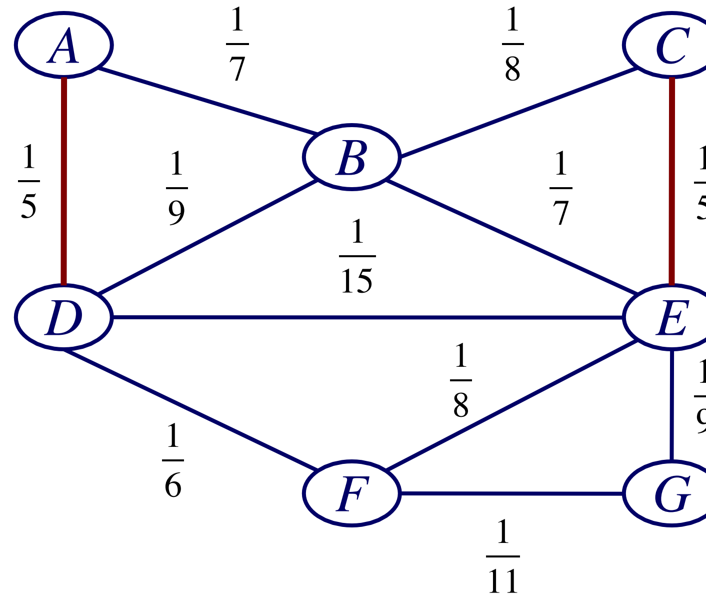
$E_{new} \leftarrow \{\ \}$
for each (u, v) in E ordered by weight (from high to low)
{
    remove (u, v) from E
    if adding (u, v) to E_{new} does not create a cycle
        add (u, v) to E_{new}
}
return V and E_{new} which represent an MST

Finding MST in Chow-Liu

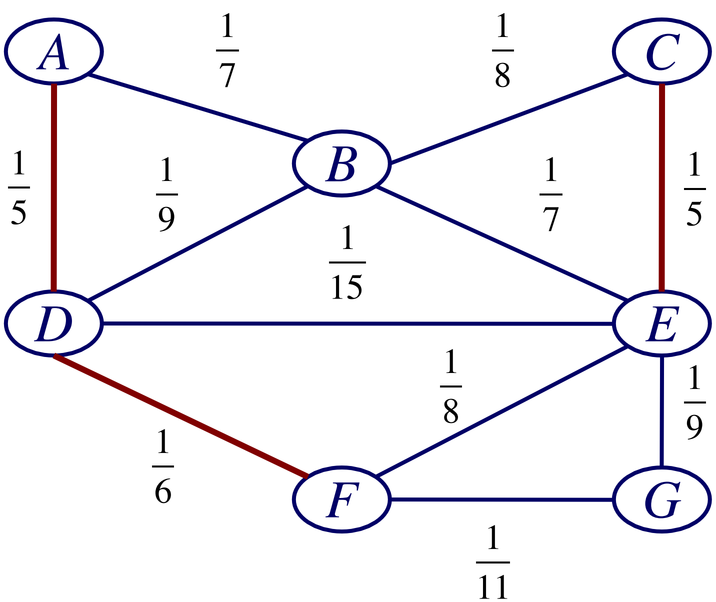
i.



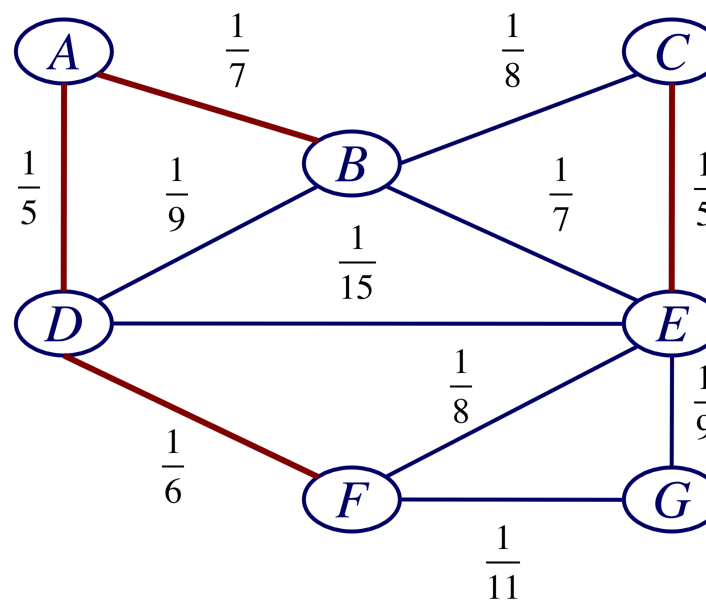
ii.



iii.

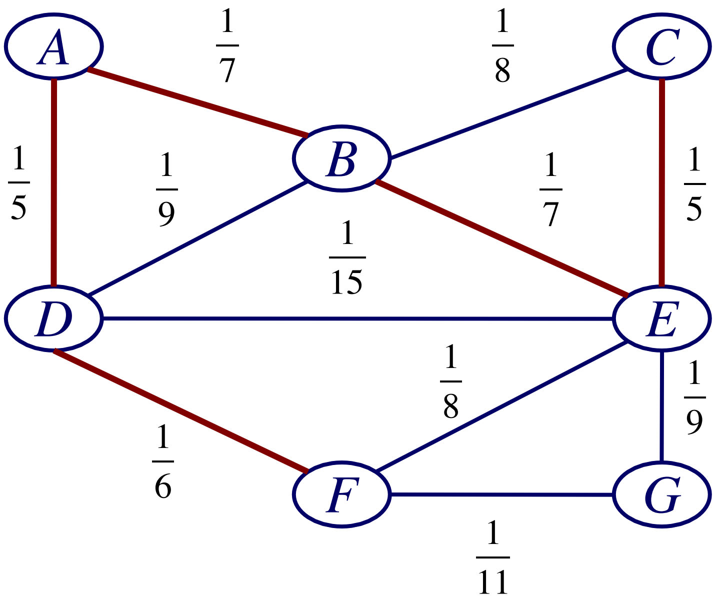


iv.

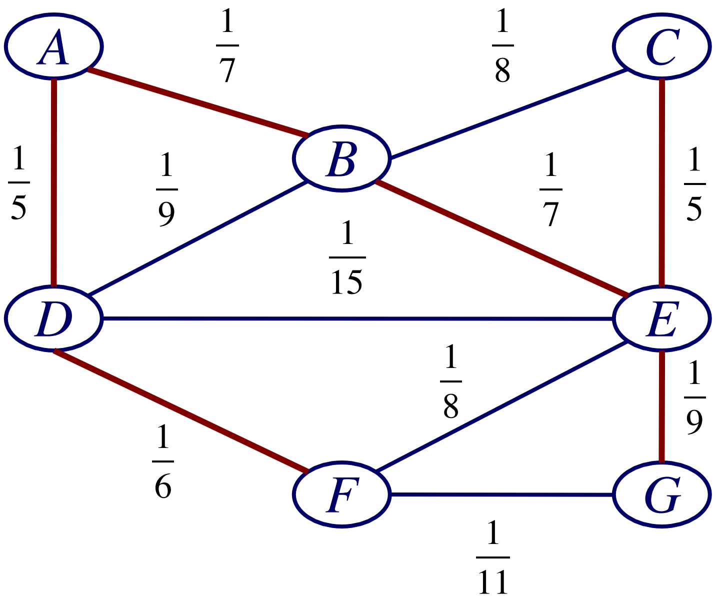


Finding MST in Chow-Liu

v.

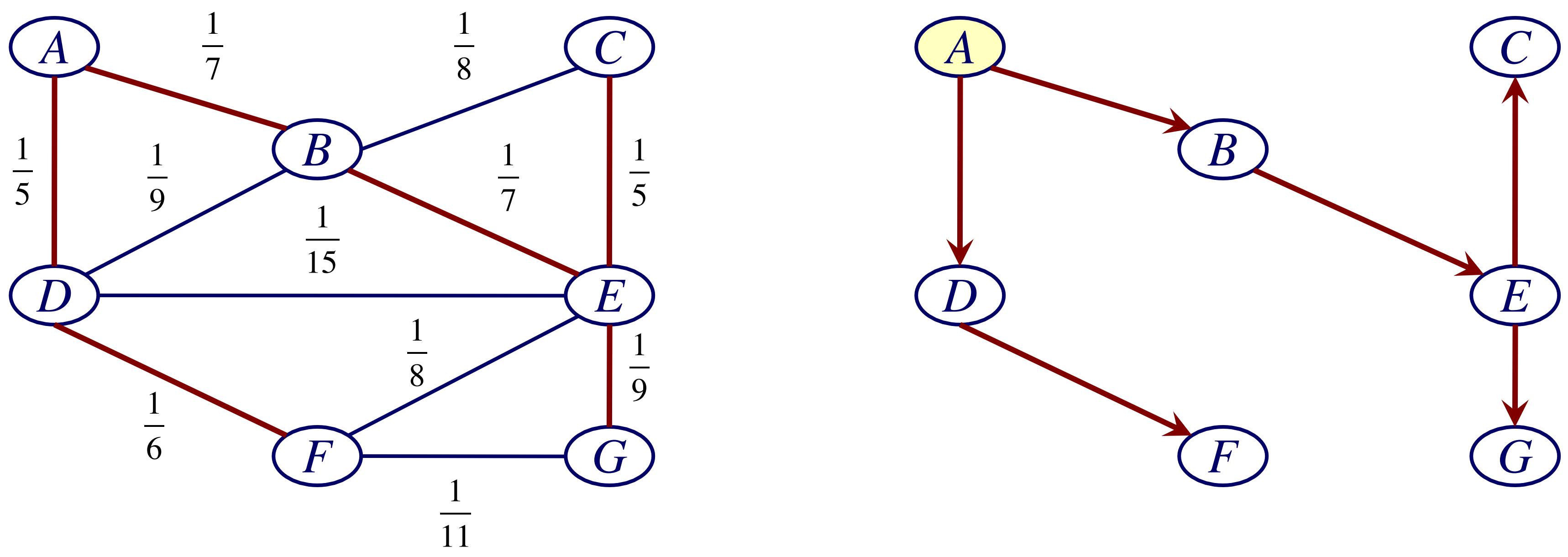


vi.



Returning directed graph in Chow-Liu

3. pick a node for the root, and assign edge directions



The Chow-Liu algorithm

- How do we know that Chow-Liu will find a tree that maximizes the data likelihood?
- Two key questions:
  - Why can we represent data likelihood as sum of $I(X;Y)$ over edges?
  - Why can we pick any direction for edges in the tree?

Why Chow-Liu maximizes likelihood (for a tree)

data likelihood given directed edges

$$\log_2 P(D \mid G, \theta_G) = \sum_{d \in D} \sum_i \log_2 P(x_i^{(d)} \mid Parents(X_i))$$

$$= |D| \sum_i \Big(I(X_i, Parents(X_i)) - H(X_i) \Big)$$

we're interested in finding the graph G that maximizes this

$$\arg\max_G \log_2 P(D \mid G, \theta_G) = \arg\max_G \sum_i I(X_i, Parents(X_i))$$

if we assume a tree, each node has at most one parent

$$\arg\max_G \log_2 P(D \mid G, \theta_G) = \arg\max_G \sum_{(X_i, X_j) \in \text{edges}} I(X_i, X_j)$$

edge directions don't matter for likelihood, because MI is symmetric

$$I(X_i, X_j) = I(X_j, X_i)$$

Heuristic search for structure learning

- each state in the search space represents a DAG Bayes net structure
- to instantiate a search approach, we need to specify
  - scoring function
  - state transition operators
  - search algorithm

Scoring function decomposability

- when the appropriate priors are used, and all instances in D are complete, the scoring function can be decomposed as follows

$$\text{score}(G,\ D) = \sum_i \text{score}(X_i, Parents(X_i) : D)$$

- thus we can
  - score a network by summing terms over the nodes in the network
  - efficiently score changes in a *local* search procedure

Scoring functions for structure learning

- Can we find a good structure just by trying to maximize the likelihood of the data?

$$\arg\max_{G,\,\theta_G} \ \log P(D \mid G, \theta_G)$$

- If we have a strong restriction on the the structures allowed (e.g. a tree), then maybe.

- Otherwise, no! Adding an edge will never decrease likelihood. Overfitting likely.

Scoring functions for structure learning

- there are many different scoring functions for BN structure search
- one general approach

$$\underbrace{\arg\max_{G,\,\theta_G} \; \log P(D \mid G, \theta_G) - f(m)|\theta_G|}$$

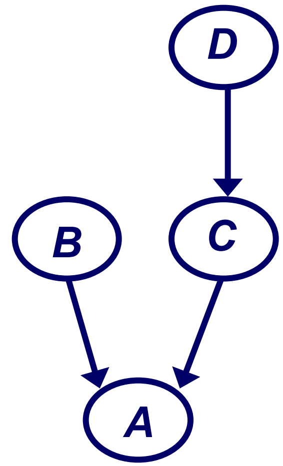
$$\log P(D \mid G, \theta_G) - \underbrace{f(m)|\theta_G|}_{\text{complexity penalty}}$$

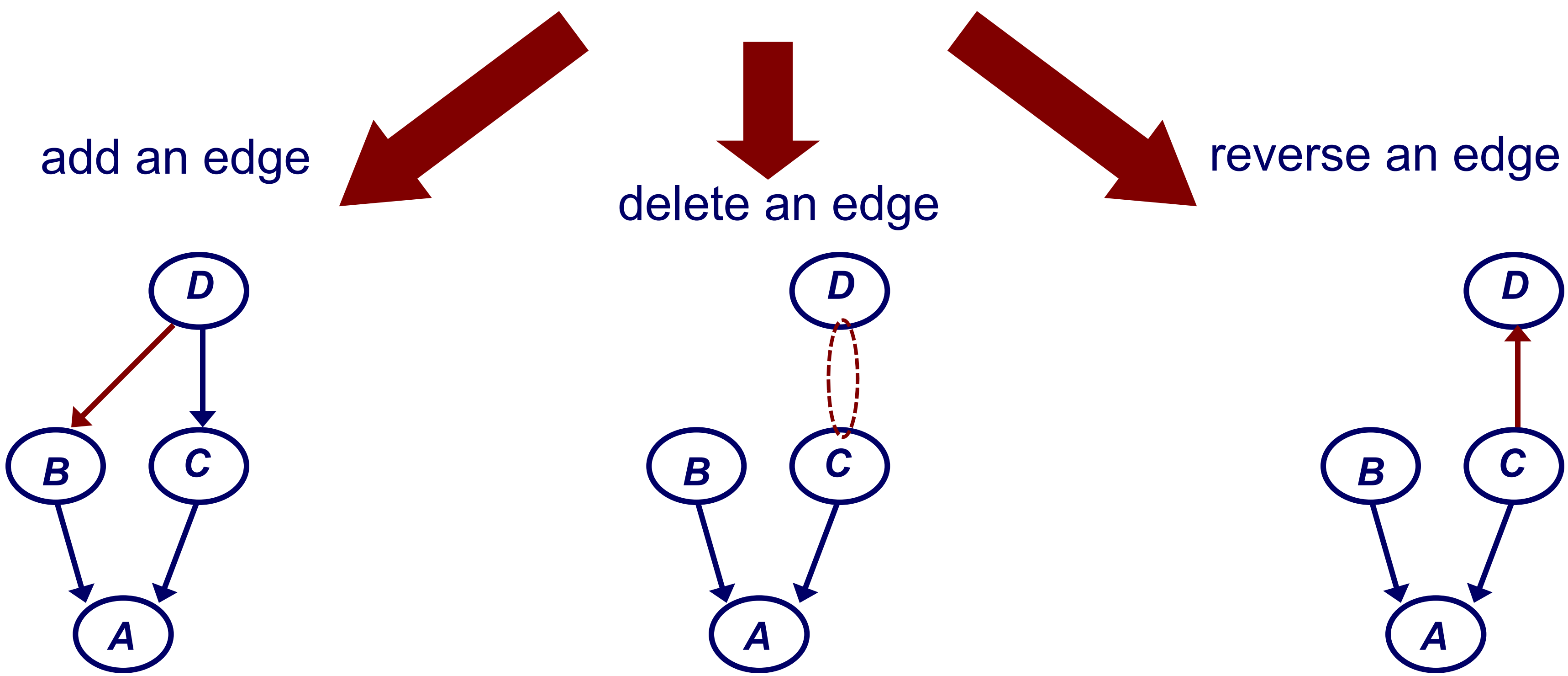
Akaike Information Criterion (AIC): $f(m) = 1$

Bayesian Information Criterion (BIC): $f(m) = \frac{1}{2}\log(m)$

Structure search operators

given the current network
at some stage of the search,
we can…





Bayesian network search: *hill-climbing*

given: data set D, initial network B_0

$i = 0$

$\mathrm{B}_{best} \leftarrow B_0$

while stopping criteria not met

{

  for each possible operator application a

  {

    $B_{new} \leftarrow$ apply(a, B_i)

    if score(B_{new}) > score(B_{best})

      $\mathrm{B}_{best} \leftarrow B_{new}$

  }

  $++i$

  $B_i \leftarrow B_{best}$

}

return B_i

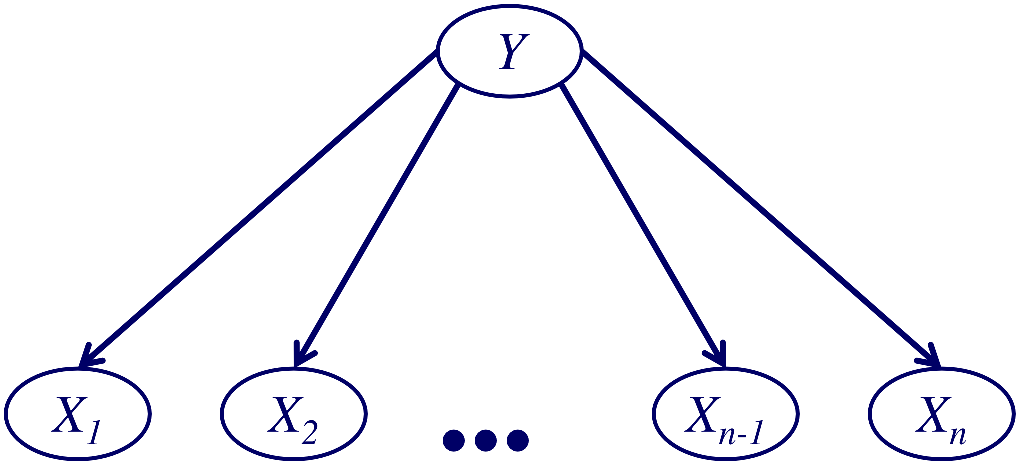
Bayes nets for classification

- the learning methods for BNs we've discussed so far can be thought of as being unsupervised
  - the learned models are not constructed to predict the value of a special class variable
  - instead, they can predict values for arbitrarily selected query variables

- now let's consider BN learning for a standard supervised task (learn a model to predict Y given $X_1 \ldots X_n$)

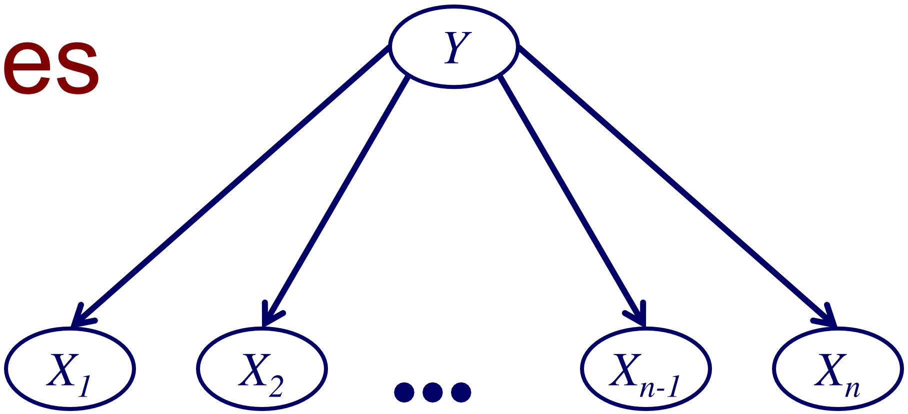
Naïve Bayes

- one very simple BN approach for supervised tasks is *naïve Bayes*
- in naïve Bayes, we assume that all features X_i are conditionally independent given the class Y



$$P(X_1, \ \ldots, \ X_n, \ Y) = P(Y) \prod_{i=1}^{n} P(X_i \mid Y)$$

Naïve Bayes



Learning

- estimate $P(Y = y)$ for each value of the class variable Y
- *estimate* $P(X_i = x \mid Y = y)$ *for each* X_i

Classification: use Bayes' Rule

$$P(Y = y \mid \boldsymbol{x}) = \frac{P(y)P(\boldsymbol{x} \mid y)}{\sum_{y' \in \text{values}(Y)} P(y')P(\boldsymbol{x} \mid y')} = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{\sum_{y' \in \text{values}(Y)} \left(P(y')\prod_{i=1}^{n} P(x_i \mid y') \right)}$$

Naïve Bayes vs. BNs learned with an unsupervised structure search

test-set error on 25 classification data sets from the UC-Irvine Repository

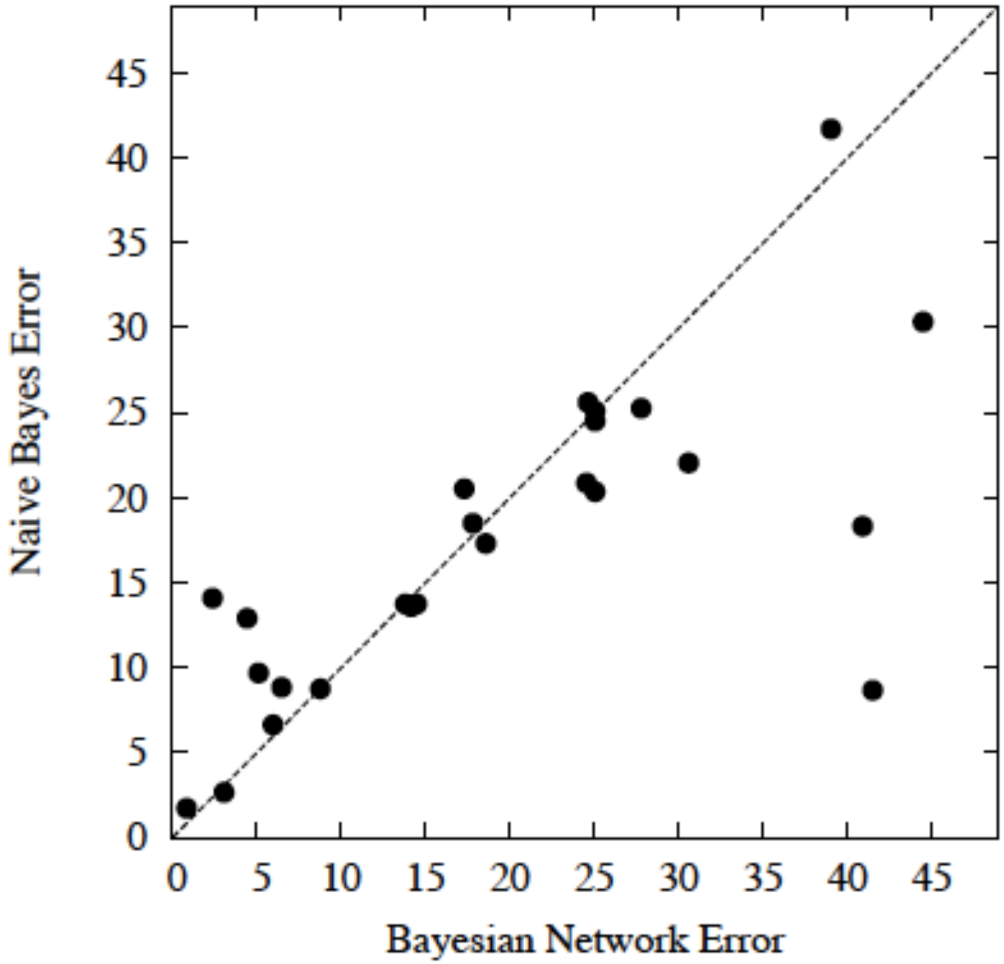


Figure from Friedman et al., *Machine Learning* 1997

The Tree Augmented Network (TAN) algorithm

[Friedman et al., *Machine Learning* 1997]

- learns a <u>tree structure</u> to augment the edges of a naïve Bayes network

- algorithm
  1. compute weight $I(X_i, X_j \mid Y)$ for each possible edge (X_i, X_j) between <u>features</u>
  2. find maximum weight spanning tree (MST) for graph over $X_1 \ldots X_n$
  3. assign edge directions in MST
  4. construct a TAN model by adding node for Y and an edge from Y to each X_i

Conditional mutual information in the TAN algorithm

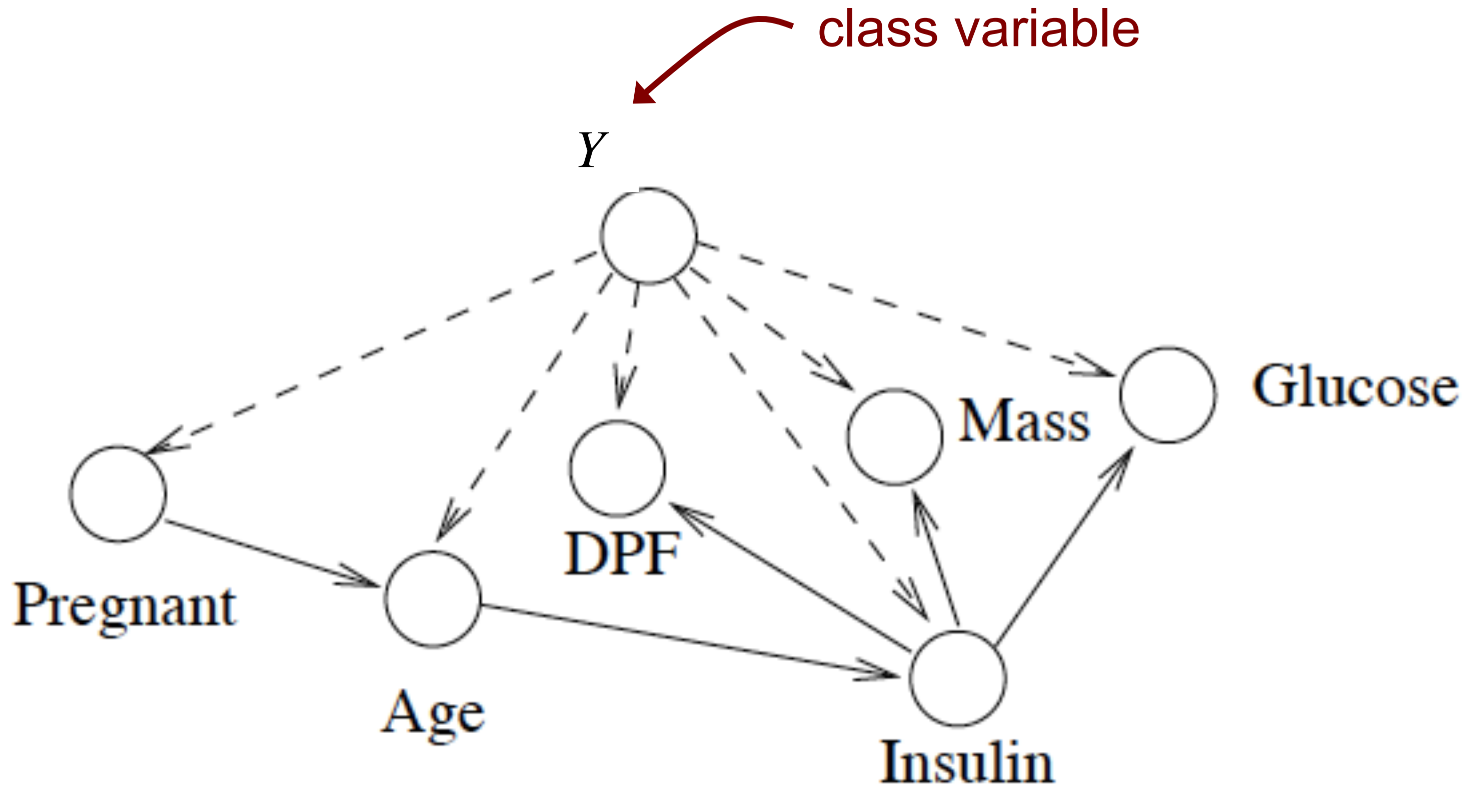
conditional mutual information is used to calculate edge weights

$$I(X_i, X_j \mid Y) =$$

$$\sum_{x_i \in \text{values}(X_i)} \sum_{x_j \in \text{values}(X_j)} \sum_{y \in \text{values}(Y)} P(x_i, x_j, y) \log_2 \frac{P(x_i, x_j \mid y)}{P(x_i \mid y) P(x_j \mid y)}$$

“how much information X_i provides about X_j when the value of Y is known”

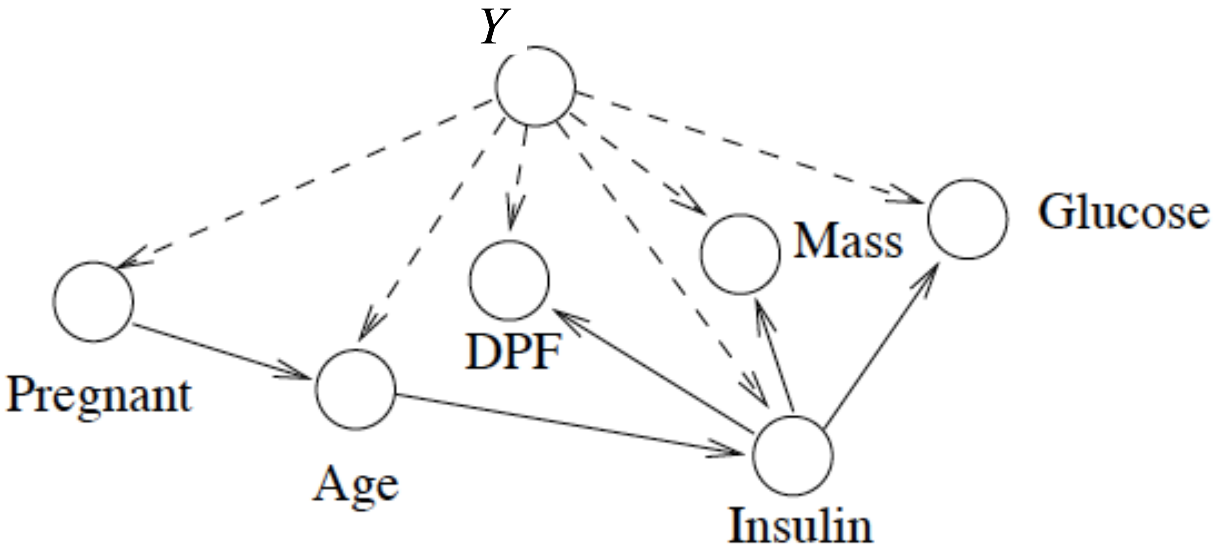
Example TAN network



naïve Bayes edges

edges determined by MST

Classification with a TAN network



As before use Bayes' Rule:

$$P(Y = y|\boldsymbol{x}) = \frac{P(y)P(\boldsymbol{x}|y)}{\sum_{y'} P(y')P(\boldsymbol{x}|y')}$$

In the example network, we calculate $P(\boldsymbol{x}|y)$ as:

$$P(\boldsymbol{x}|y) = P(pregnant\ |y)P(age|y, pregnant)P(insulin|y, age)P(dpf\,|y, insulin)$$
$$P(mass|y, insulin)P(glucose|y, insulin)$$

TAN vs. Chow-Liu

- TAN is mostly* focused on learning a Bayes net specifically for classification problems
- the MST includes only the feature variables (the class variable is used only for calculating edge weights)
- conditional mutual information is used instead of mutual information in determining edge weights in the undirected graph
- the directed graph determined from the MST is added to the $Y \rightarrow X_i$ edges that are in a naïve Bayes network

* although parameters are still set to maximize $P(y, \boldsymbol{x})$ instead of $P(y \mid \boldsymbol{x})$

TAN vs. Naïve Bayes

test-set error on 25 data sets from the UC-Irvine Repository

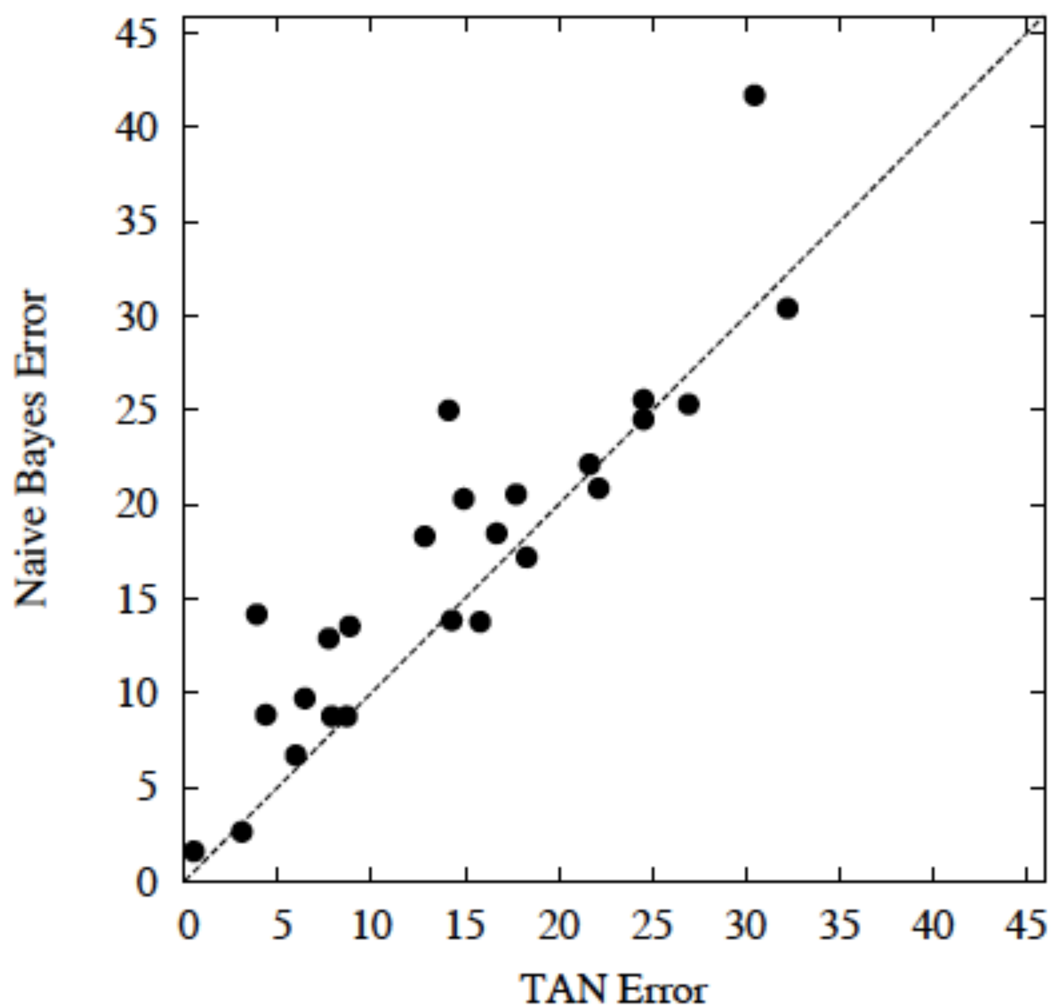


Figure from Friedman et al., *Machine Learning* 1997

Comments on Bayesian networks

- the BN representation has many advantages
  - easy to encode domain knowledge (direct dependencies, causality)
  - can represent uncertainty
  - principled methods for dealing with missing values
  - can answer arbitrary queries (in theory; in practice may be intractable)
- for supervised tasks, it may be advantageous to use a learning approach (e.g. TAN) that focuses on the dependencies that are most important

Comments on Bayesian networks (continued)

- although very simplistic, naïve Bayes often learns highly accurate models
- we focused on learning Bayes nets with only discrete variables; can also have numeric variables (although not as parents)
- BNs are one instance of a more general class of *probabilistic graphical models*