Bayesian Networks Part 2

CS 760@UW-Madison

Goals for the lecture

you should understand the following concepts

- missing data in machine learning
  - hidden variables
  - missing at random
  - missing systematically
- the EM approach to imputing missing values in Bayes net parameter learning

- the Chow-Liu algorithm for structure search

- Kullback-Leibler divergence
- the Sparse Candidate algorithm


EM Algorithm

Missing data

- Commonly in machine learning tasks, some feature values are missing
- some variables may not be observable (i.e. *hidden*) even for training instances
- values for some variables may be *missing at random*: what caused the data to be missing does not depend on the missing data itself
  - e.g. someone accidentally skips a question on an questionnaire
  - e.g. a sensor fails to record a value due to a power blip
- values for some variables may be *missing systematically*: the probability of value being missing depends on the value
  - e.g. a medical test result is missing because a doctor was fairly sure of a diagnosis given earlier test results
  - e.g. the graded exams that go missing on the way home from school are those with poor scores

Missing data

- hidden variables; values *missing at random*
  - these are the cases we'll focus on
  - one solution: try impute the values

- values *missing systematically*
  - may be sensible to represent "*missing*" as an explicit feature value

Imputing missing data with EM

Given:

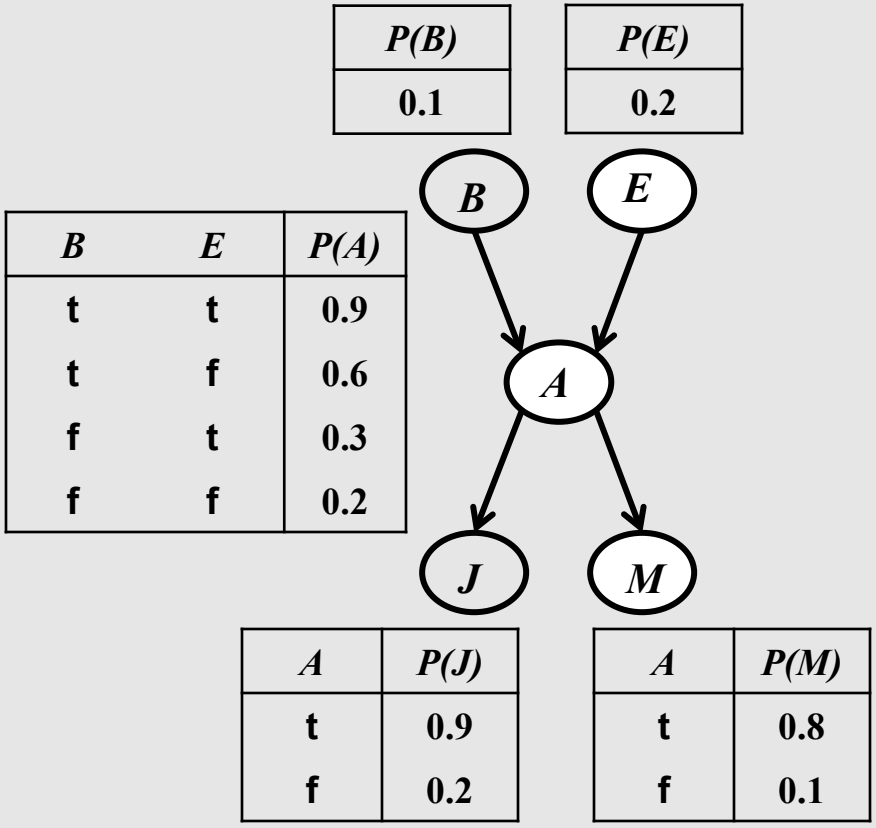
- data set with some missing values
- model structure, initial model parameters

Repeat until convergence

- *Expectation* (E) step: using current model, compute expectation over missing values
- *Maximization* (M) step: update model parameters with those that maximize probability of the data (MLE or MAP)

Example: EM for parameter learning

suppose we're given the following initial BN and training set

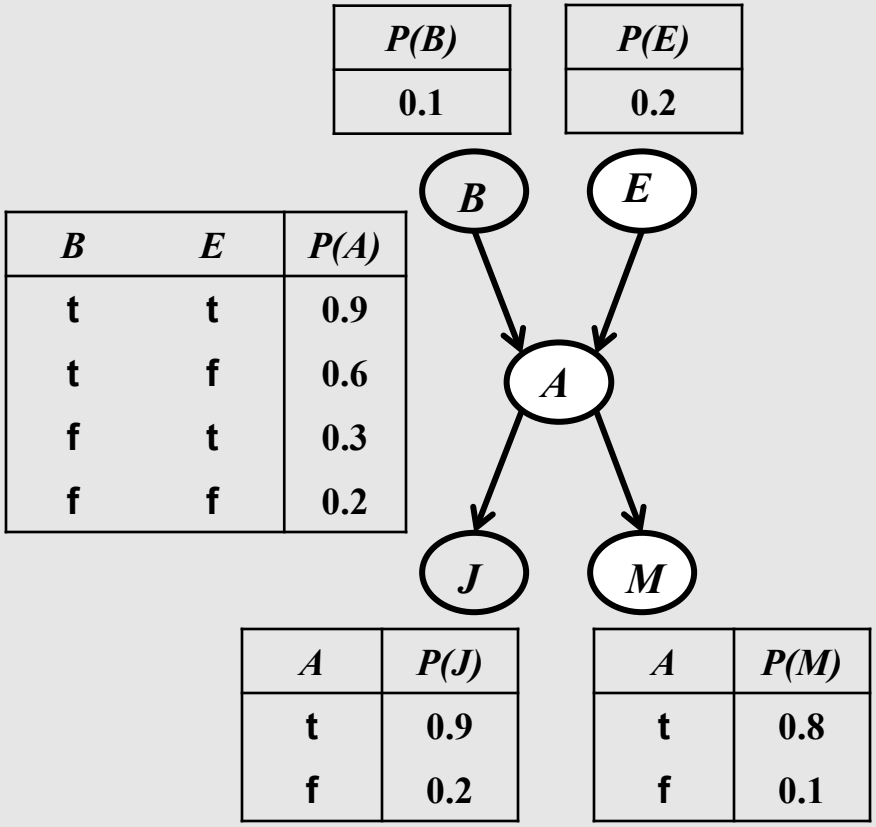


B	*E*	*A*	*J*	*M*
f	f	?	f	f
f	f	?	t	f
t	f	?	t	t
f	f	?	f	t
f	t	?	t	f
f	f	?	f	t
t	t	?	t	t
f	f	?	f	f
f	f	?	t	f
f	f	?	f	t

Example: E-step

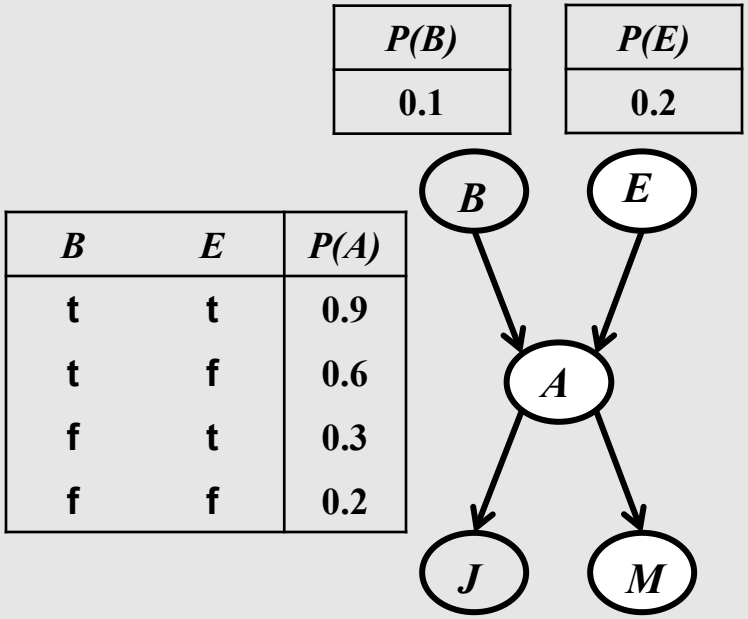
$$P(a \mid \neg b, \neg e, \neg j, \neg m)$$

$$P(\neg a \mid \neg b, \neg e, \neg j, \neg m)$$



B	E	A	J	M
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f:0.8	t	f
t	f	t:0.98 f: 0.02	t	t
f	f	t: 0.2 f: 0.8	f	t
f	t	t: 0.3 f: 0.7	t	f
f	f	t:0.2 f: 0.8	f	t
t	t	t: 0.997 f: 0.003	t	t
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f: 0.8	t	f
f	f	t: 0.2 f: 0.8	f	t

Example: E-step



B	E	P(A)
t	t	0.9
t	f	0.6
f	t	0.3
f	f	0.2

A	P(J)
t	0.9
f	0.2

A	P(M)
t	0.8
f	0.1

$$P(a \mid \neg b, \neg e, \neg j, \neg m)$$

$$= \frac{P(\neg b, \neg e, a, \neg j, \neg m)}{P(\neg b, \neg e, a, \neg j, \neg m) + P(\neg b, \neg e, \neg a, \neg j, \neg m)}$$

$$= \frac{0.9 \times 0.8 \times 0.2 \times 0.1 \times 0.2}{0.9 \times 0.8 \times 0.2 \times 0.1 \times 0.2 + 0.9 \times 0.8 \times 0.8 \times 0.8 \times 0.9}$$

$$= \frac{0.00288}{0.4176} = 0.0069$$

$$P(a \mid \neg b, \neg e, j, \neg m)$$

$$= \frac{P(\neg b, \neg e, a, j, \neg m)}{P(\neg b, \neg e, a, j, \neg m) + P(\neg b, \neg e, \neg a, j, \neg m)}$$

$$= \frac{0.9 \times 0.8 \times 0.2 \times 0.9 \times 0.2}{0.9 \times 0.8 \times 0.2 \times 0.9 \times 0.2 + 0.9 \times 0.8 \times 0.8 \times 0.2 \times 0.9}$$

$$= \frac{0.02592}{0.1296} = 0.2$$

Example: M-step

re-estimate probabilities using expected counts

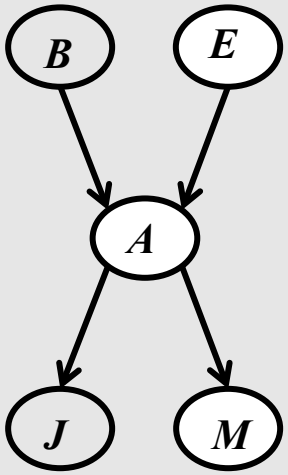
$$P(a \mid b,e) = \frac{E\#(a \wedge b \wedge e)}{E\#(b \wedge e)}$$

$$P(a \mid b,e) = \frac{0.997}{1}$$

$$P(a \mid b,\neg e) = \frac{0.98}{1}$$

$$P(a \mid \neg b,e) = \frac{0.3}{1}$$

$$P(a \mid \neg b,\neg e) = \frac{0.0069 + 0.2 + 0.2 + 0.2 + 0.0069 + 0.2 + 0.2}{7}$$



B	*E*	*P(A)*
t	t	0.997
t	f	0.98
f	t	0.3
f	f	0.145

re-estimate probabilities for $P(J \mid A)$ and $P(M \mid A)$ in same way

B	*E*	*A*	*J*	*M*
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f:0.8	t	f
t	f	t:0.98 f: 0.02	t	t
f	f	t: 0.2 f: 0.8	f	t
f	t	t: 0.3 f: 0.7	t	f
f	f	t:0.2 f: 0.8	f	t
t	t	t: 0.997 f: 0.003	t	t
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f: 0.8	t	f
f	f	t: 0.2 f: 0.8	f	t

Example: M-step

re-estimate probabilities using expected counts

$$P(j \mid a) = \frac{E\#(a \wedge j)}{E\#(a)}$$

$$P(j \mid a) = \frac{0.2 + 0.98 + 0.3 + 0.997 + 0.2}{0.0069 + 0.2 + 0.98 + 0.2 + 0.3 + 0.2 + 0.997 + 0.0069 + 0.2 + 0.2}$$

$$P(j \mid \neg a) = \frac{0.8 + 0.02 + 0.7 + 0.003 + 0.8}{0.9931 + 0.8 + 0.02 + 0.8 + 0.7 + 0.8 + 0.003 + 0.9931 + 0.8 + 0.8}$$

B	*E*	*A*	*J*	*M*
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f:0.8	t	f
t	f	t:0.98 f: 0.02	t	t
f	f	t: 0.2 f: 0.8	f	t
f	t	t: 0.3 f: 0.7	t	f
f	f	t:0.2 f: 0.8	f	t
t	t	t: 0.997 f: 0.003	t	t
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f: 0.8	t	f
f	f	t: 0.2 f: 0.8	f	t

Convergence of EM

- E and M steps are iterated until probabilities converge
- will converge to a maximum in the data likelihood (MLE or MAP)
- the maximum may be a local optimum, however
- the optimum found depends on starting conditions (initial estimated probability parameters)

Learning Structure: The Chow-Liu Algorithm

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. sparse candidate)

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

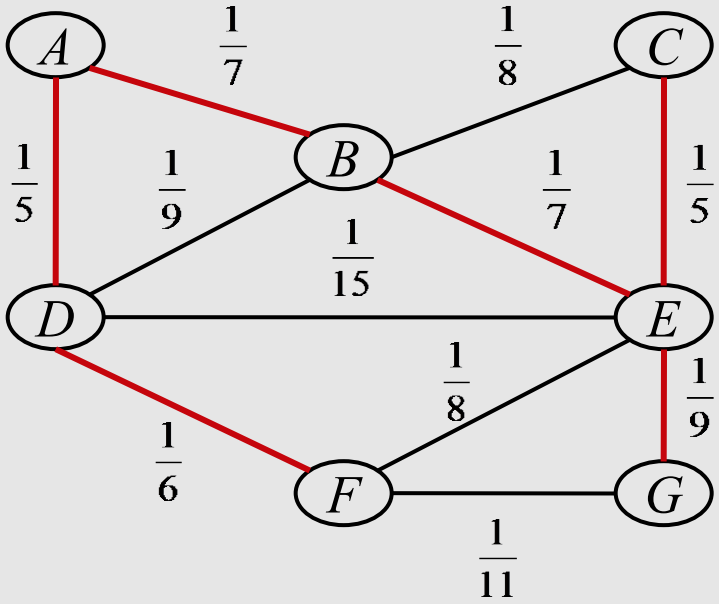
The Chow-Liu algorithm

1. use mutual information to calculate edge weights

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

The Chow-Liu algorithm

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



The Chow-Liu algo always have a complete graph, but here we use a non-complete graph as the example for clarity.

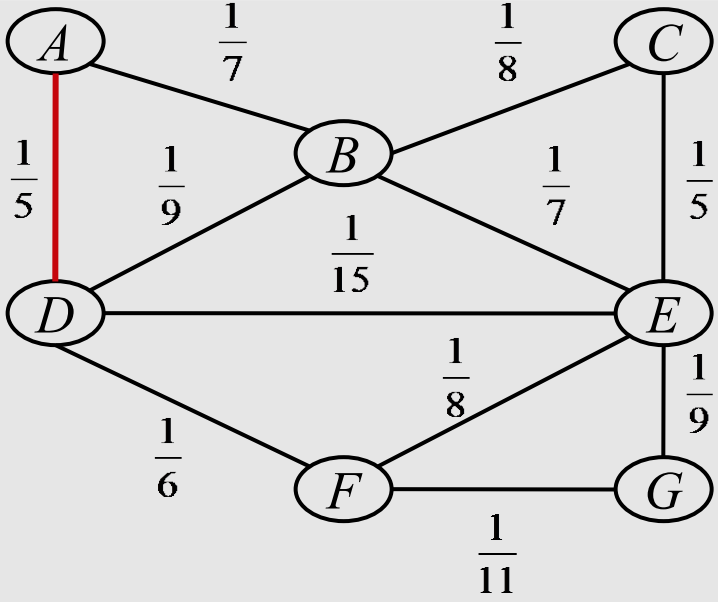
Kruskal's algorithm for finding an MST

given: graph with vertices V and edges E

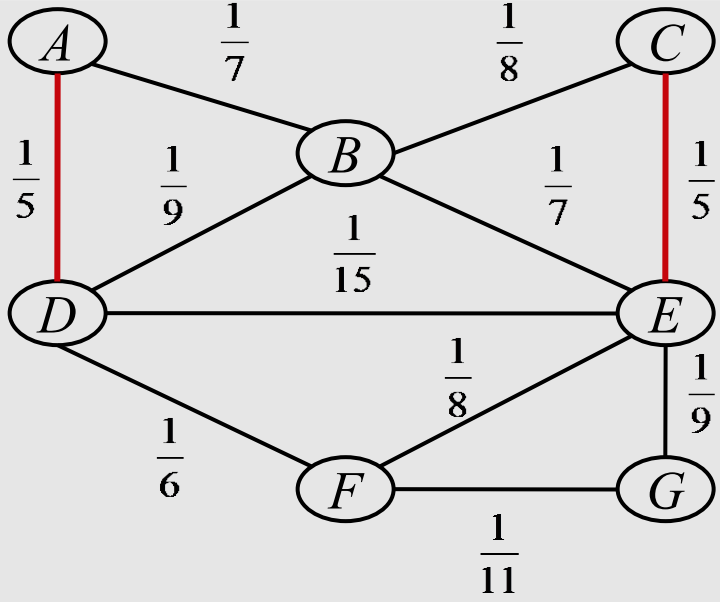
$E_{new} \leftarrow \{\ \}$
for each (u, v) in E ordered by weight (from high to low)
{
  remove (u, v) from E
  if adding (u, v) to E_{new} does not create a cycle
    add (u, v) to E_{new}
}
return V and E_{new} which represent an MST

Finding MST in Chow-Liu

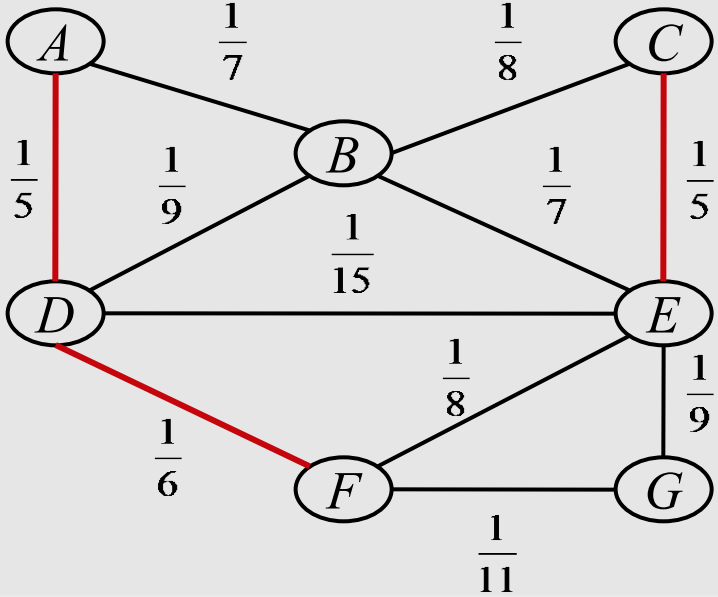
i.



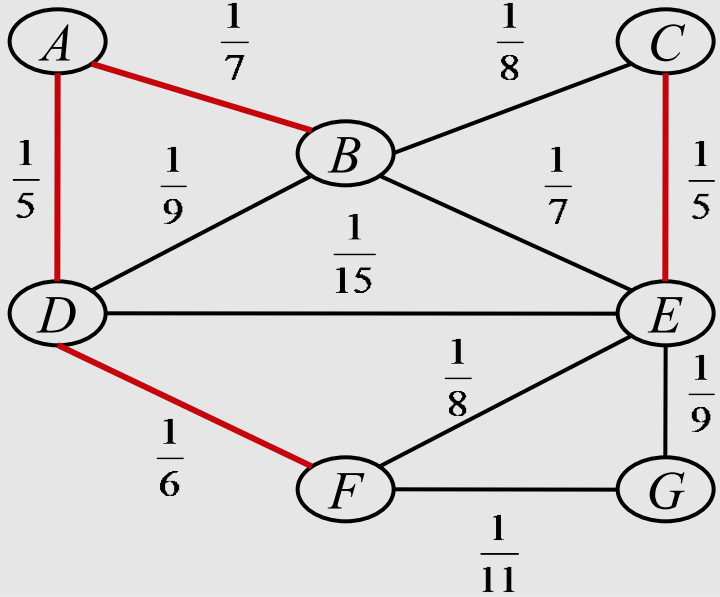
ii.



iii.

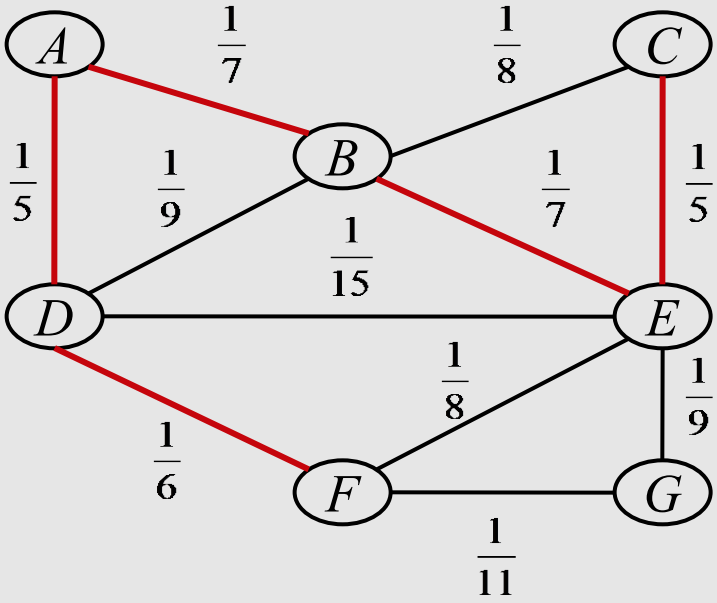


iv.

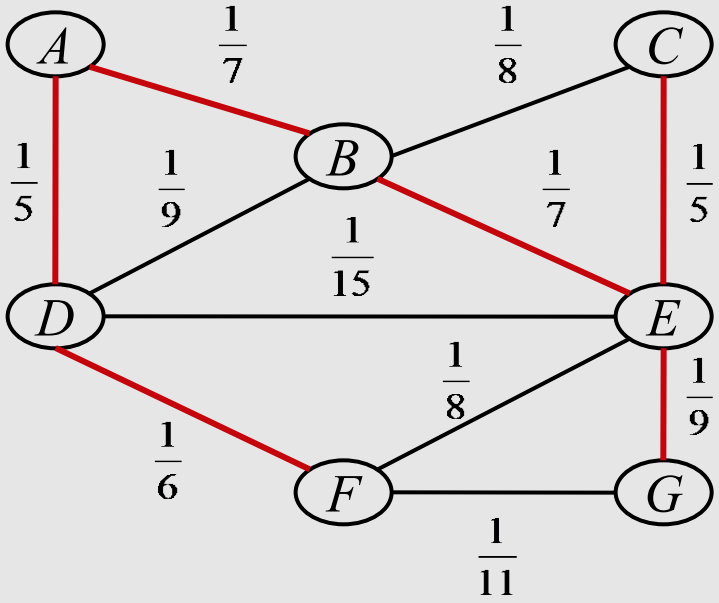


Finding MST in Chow-Liu

v.

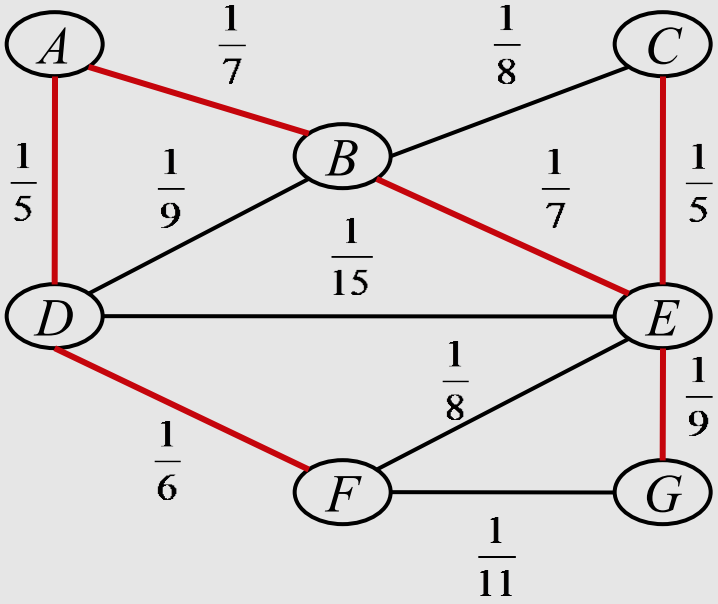


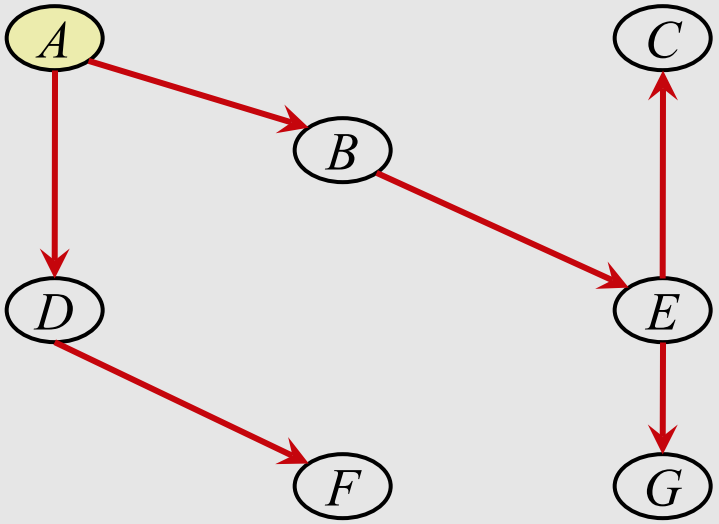
vi.



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

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

$$\mathrm{E}\left[\log_2 P(D \mid G, \theta_G)\right] = \mid D \mid \sum_i (I(X_i, Parents(X_i)) - H(X_i))$$

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

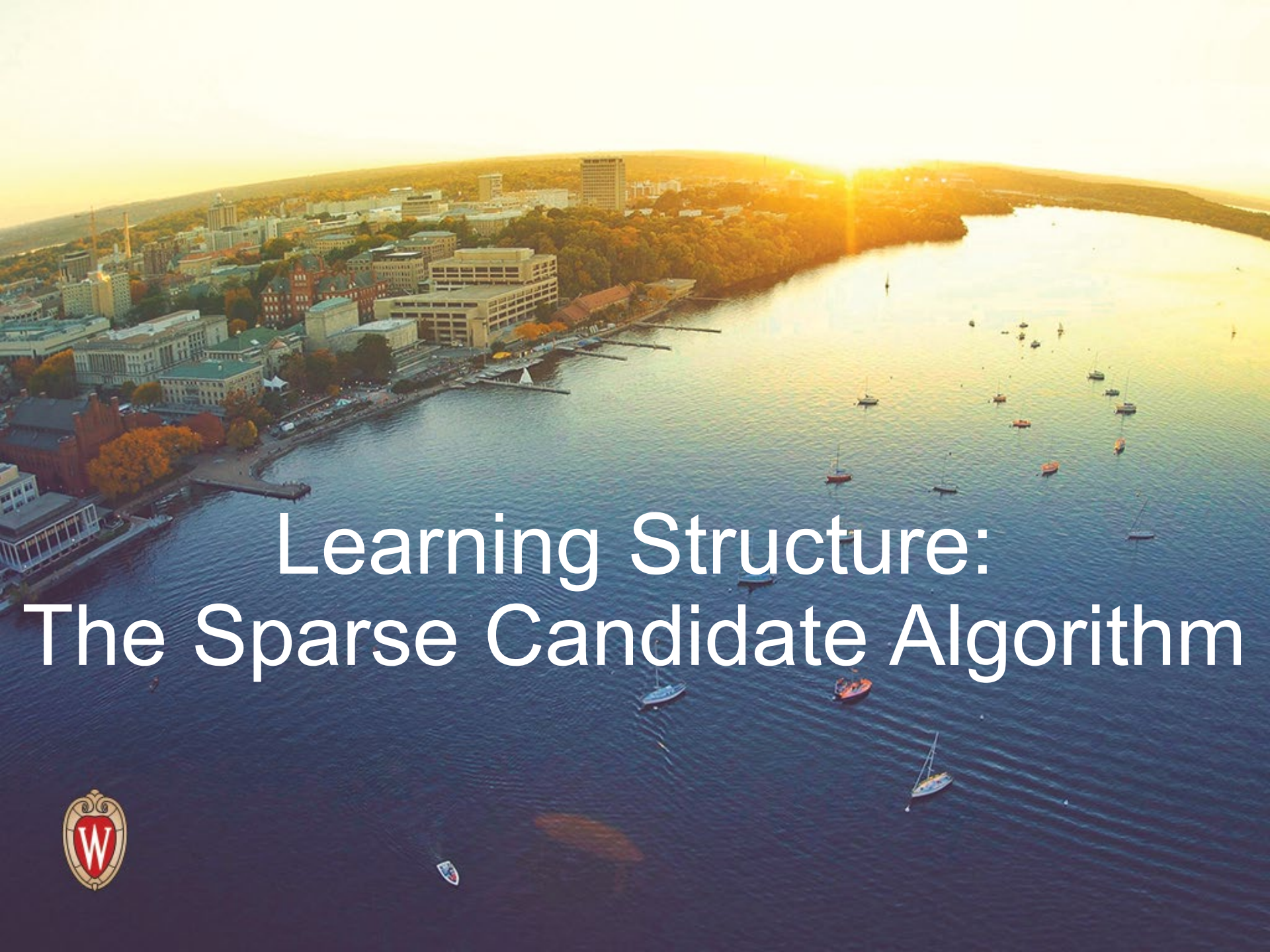
$$\arg\max_G \log_2 P(D \mid G, \theta_G) = \arg\max_G \sum_i I(X_i, Parents(X_i))$$

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

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



Learning Structure: The Sparse Candidate Algorithm

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

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

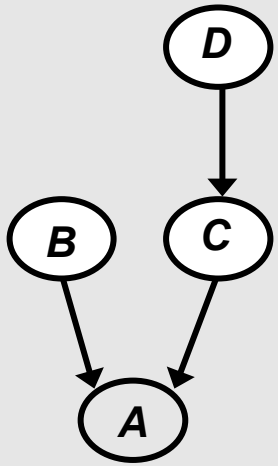
complexity penalty

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

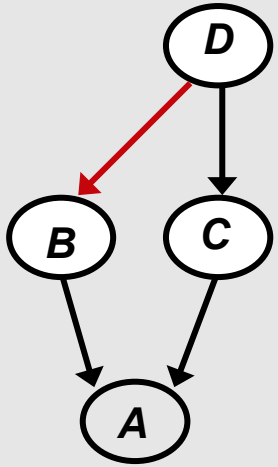
Bayesian Information Criterion (BIC): $f(m) = \frac{1}{2}\log(m)$

Structure search operators

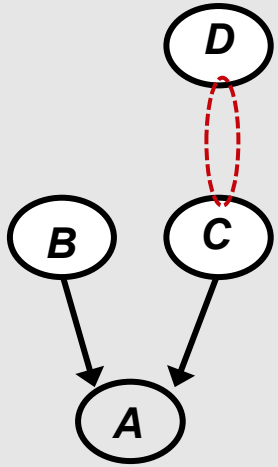
given the current network
at some stage of the search,
we can…



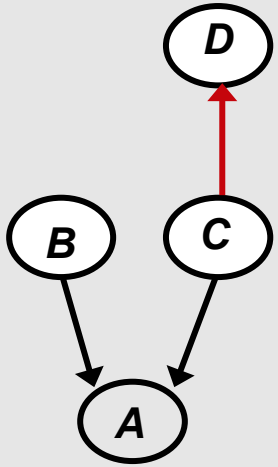
add an edge



delete an edge



reverse an edge



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

Bayesian network search: the *Sparse Candidate* algorithm [Friedman et al., *UAI* 1999]

given: data set D, initial network B_0, parameter k

$i = 0$

repeat

{

$++i$

// restrict step

select for each variable X_j a set C_j^i of candidate parents ($|C_j^i| \leq k$)

// maximize step

find network B_i maximizing score among networks where $\forall X_j$, Parents(X_j) $\subseteq C_j^i$

} until convergence

return B_i

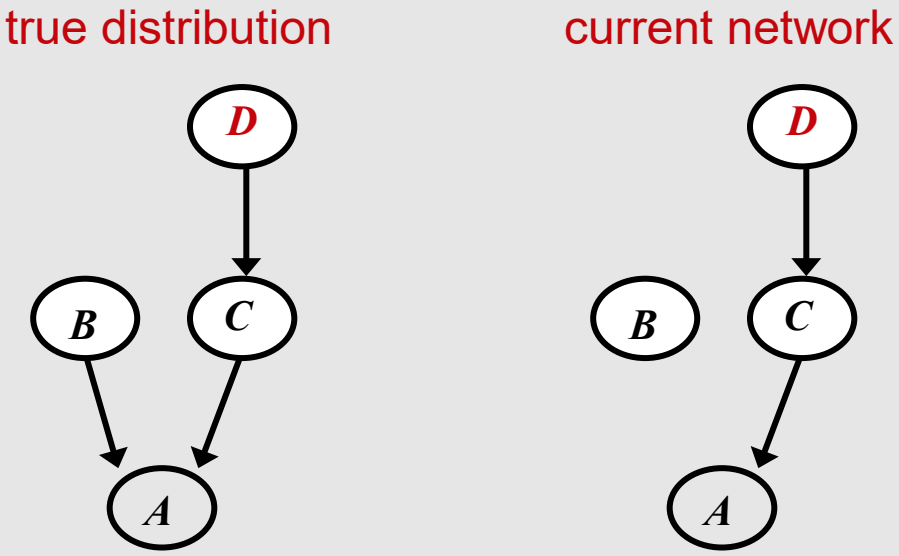
The *restrict* step in Sparse Candidate

- to identify candidate parents in the first iteration, can compute the *mutual information* between pairs of variables

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

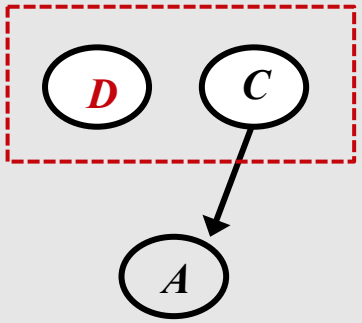
The *restrict* step in Sparse Candidate

- Suppose:



we're selecting two candidate parents for A, and $I(A, C) > I(A, D) > I(A, B)$

- with mutual information, the candidate parents for A would be C and D



- how could we get B as a candidate parent?

The *restrict* step in Sparse Candidate

- *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

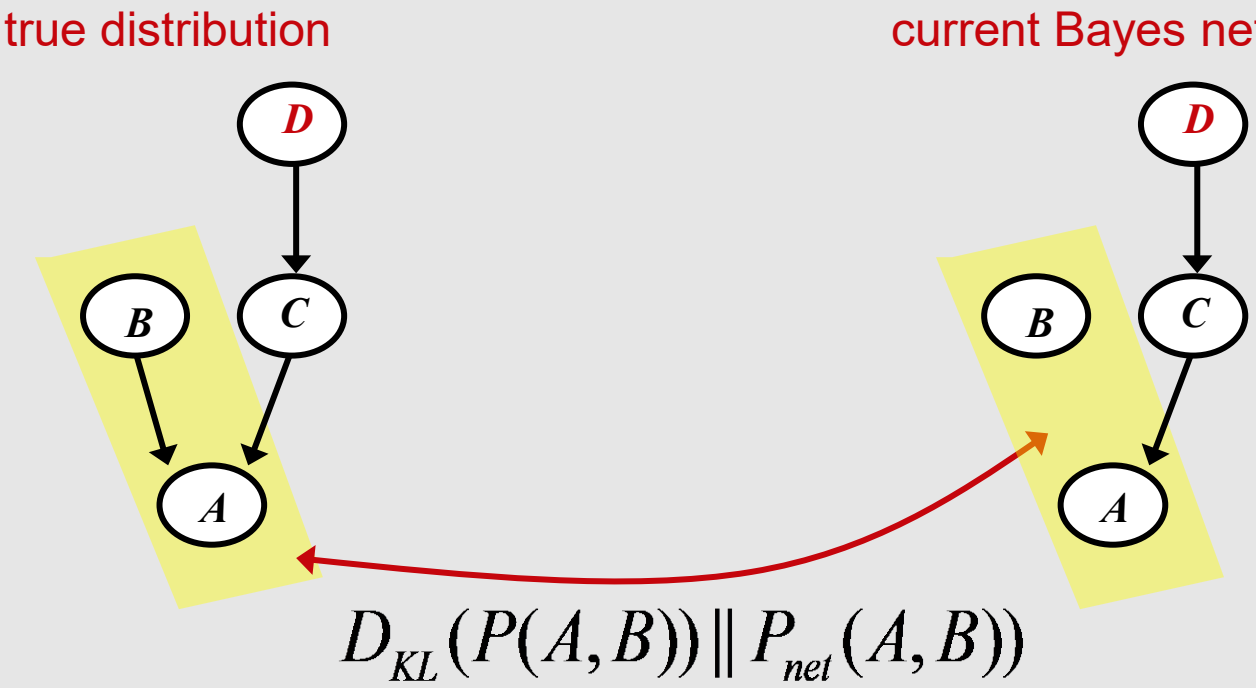
$$P(X,Y)$$

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

The *restrict* step in Sparse Candidate

- we can use KL to assess the discrepancy between the network's $P_{net}(X, Y)$ and the empirical $P(X, Y)$

$$M(X,Y) = D_{KL}(P(X,Y)) \| P_{net}(X,Y))$$



- can estimate $P_{net}(X, Y)$ by sampling from the network (i.e. using it to generate instances)

The *restrict* step in Sparse Candidate

given: data set D, current network B_i, parameter k

for each variable X_j

{

calculate M(X_j , X_l) for all $X_j \neq X_l$ such that $X_l \notin$ Parents(X_j)

choose highest ranking $X_1 \ldots X_{k-s}$ where $s =$ | Parents(X_j) |

// include current parents in candidate set to ensure monotonic

// improvement in scoring function

$C_j^i =$Parents(X_j) ∪ $X_1 \ldots X_{k-s}$

}

return { C_j^i } for all X_j

The *maximize* step in Sparse Candidate

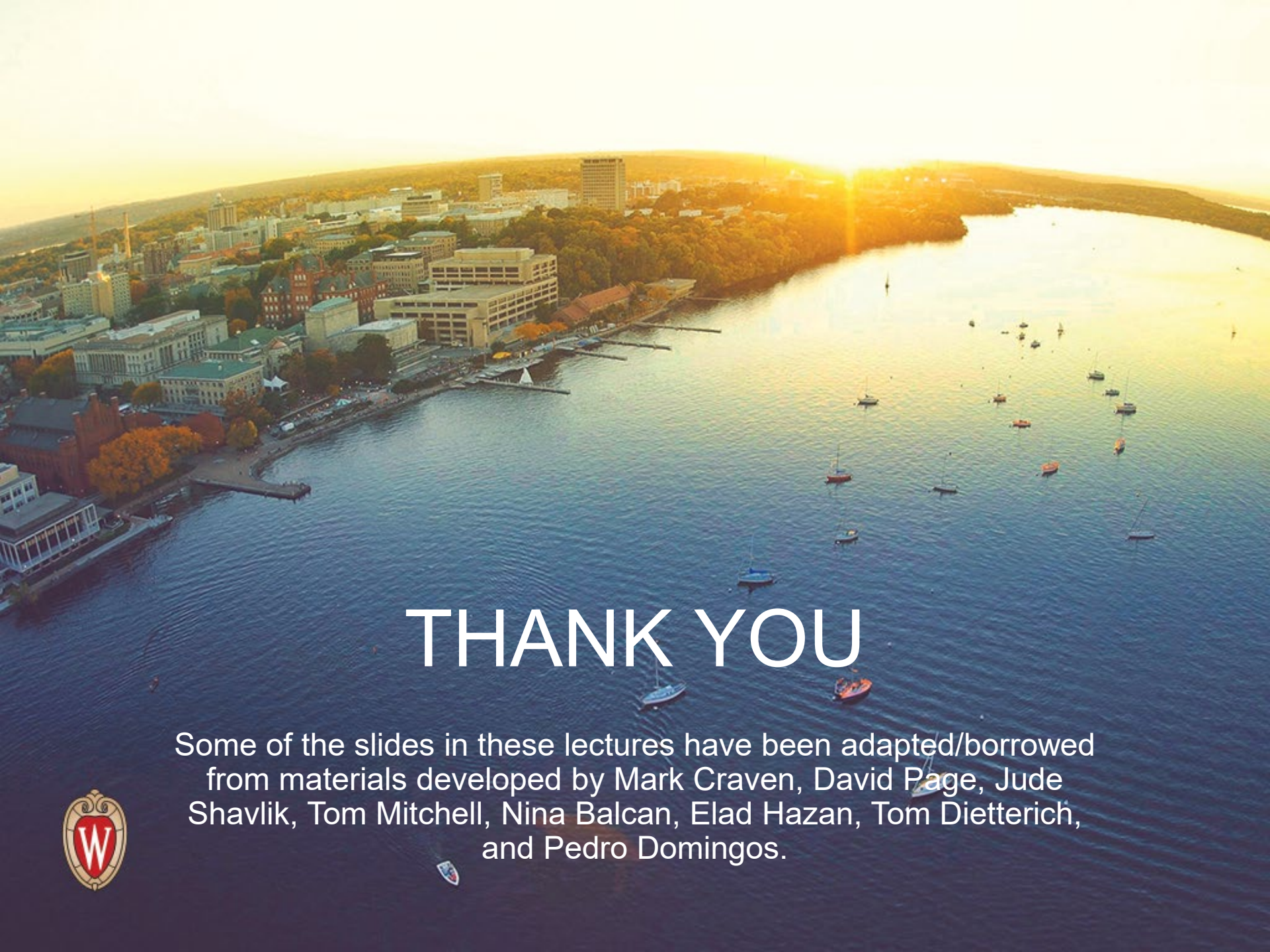
- hill-climbing search with *add-edge*, *delete-edge*, *reverse-edge* operators
- test to ensure that cycles aren't introduced into the graph

Efficiency of Sparse Candidate

n = number of variables

	possible parent sets for each node	changes scored on first iteration of search	changes scored on subsequent iterations
ordinary greedy search	$O(2^n)$	$O(n^2)$	$O(n)$
greedy search w/at most k parents	$O\left(\binom{n}{k}\right)$	$O(n^2)$	$O(n)$
Sparse Candidate	$O(2^k)$	$O(kn)$	$O(k)$

after we apply an operator, the scores will change only for edges from the parents of the node with the new impinging edge



THANK YOU

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, and Pedro Domingos.