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

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

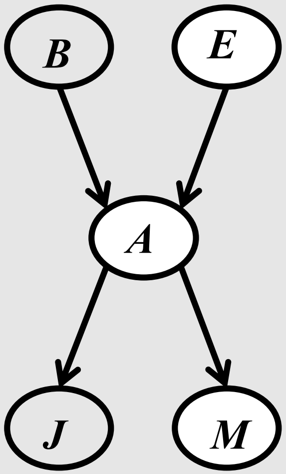
- *Expectation* (E) step: using current model, compute expectation over missing values
- *Maximization* (M) step: update model parameters with those that maximize probability of the data (MLE or MAP)

Example: EM for parameter learning

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

P(B)
0.1

P(E)
0.2



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

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

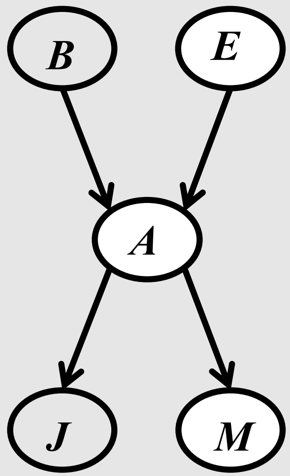
Example: E-step

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

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

P(B)
0.1

P(E)
0.2



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**

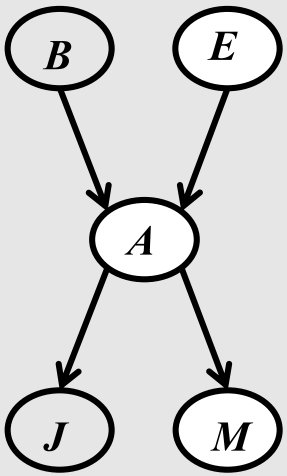
B	*E*	*A*	*J*	*M*
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f:0.8	t	f
t	f	t:0.98 f: 0.02	t	t
f	f	t: 0.2 f: 0.8	f	t
f	t	t: 0.3 f: 0.7	t	f
f	f	t:0.2 f: 0.8	f	t
t	t	t: 0.997 f: 0.003	t	t
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f: 0.8	t	f
f	f	t: 0.2 f: 0.8	f	t

Example: E-step

P(B)
0.1

P(E)
0.2

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

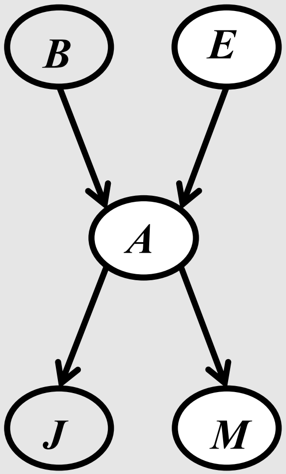
$$P(a \mid b, e) = \frac{E\#(a \wedge b \wedge e)}{E\#(b \wedge e)}$$

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

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

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

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



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 + 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

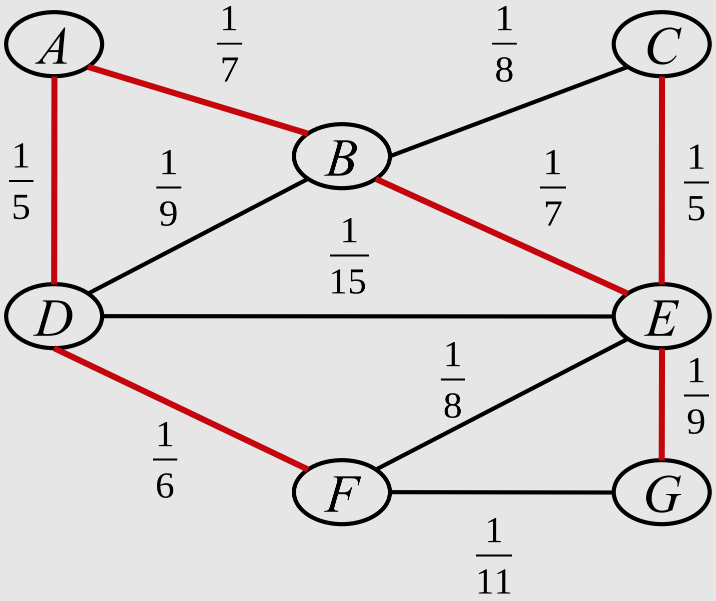
The Chow-Liu algorithm

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.

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

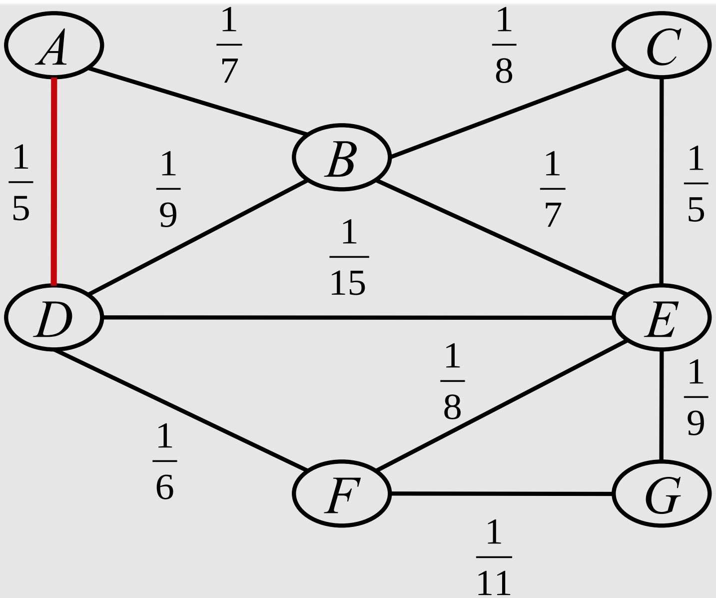
add (u, v) to E_{new}

}

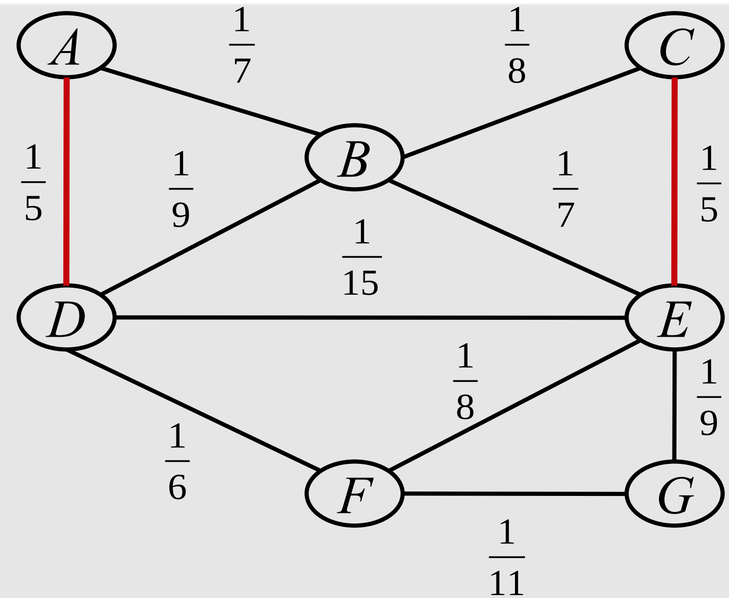
return V and E_{new} which represent an MST

Finding MST in Chow-Liu

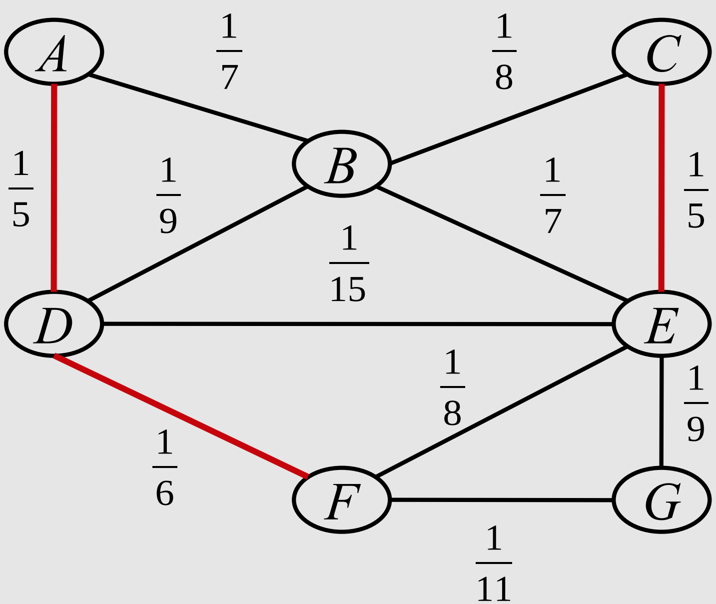
i.



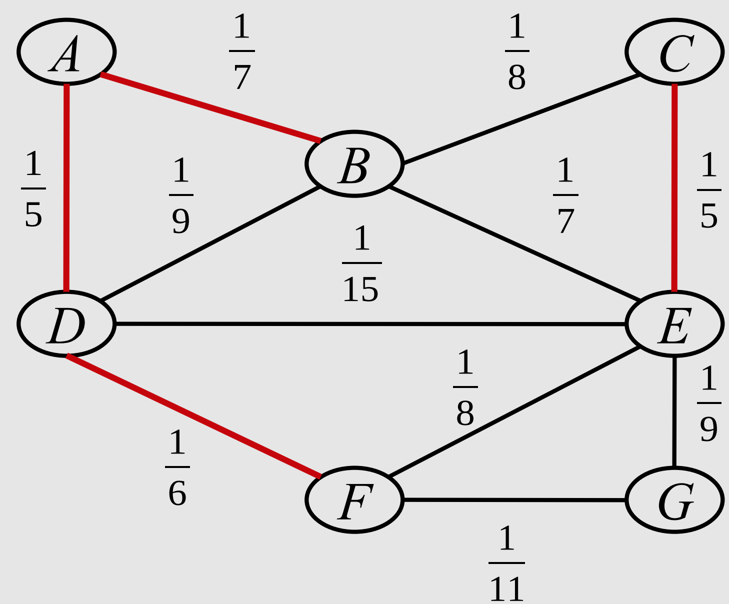
ii.



iii.

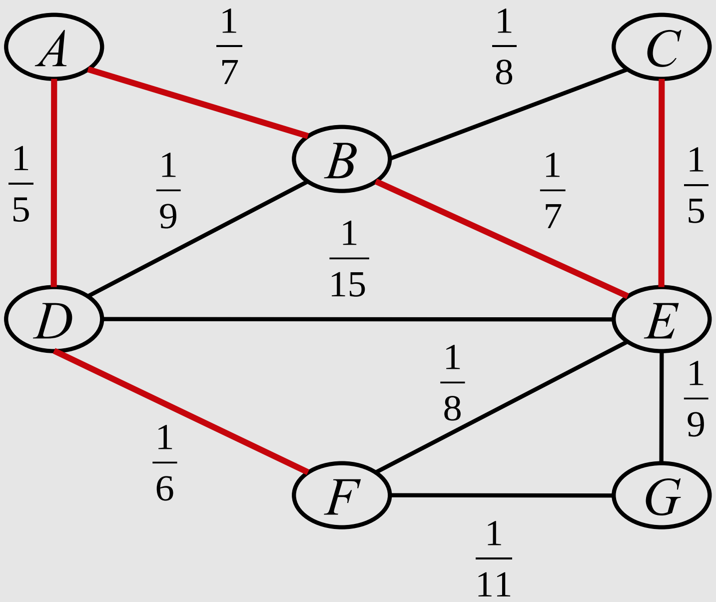


iv.

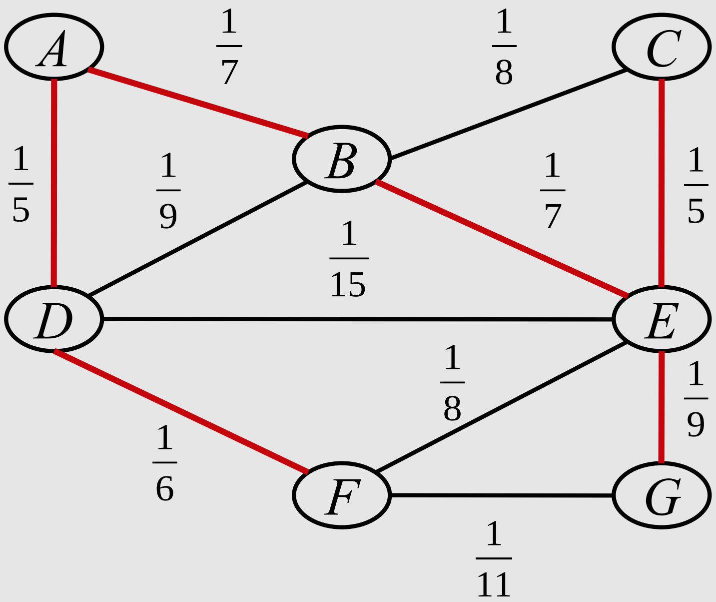


Finding MST in Chow-Liu

v.

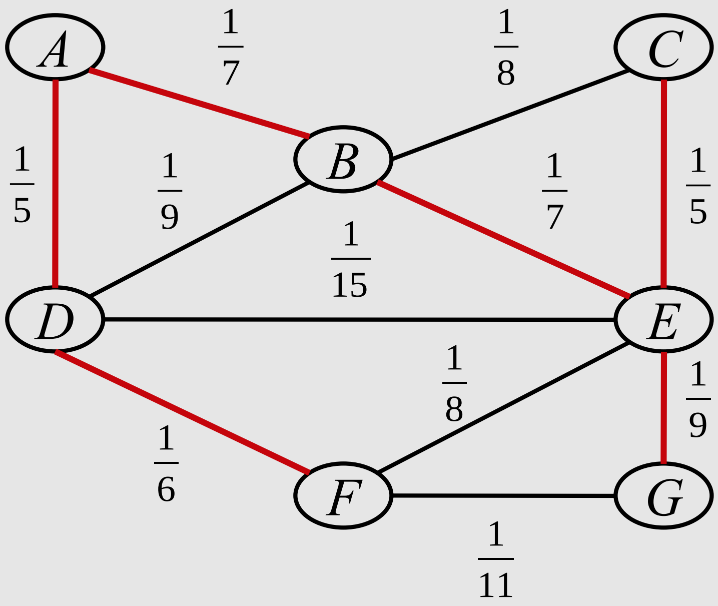


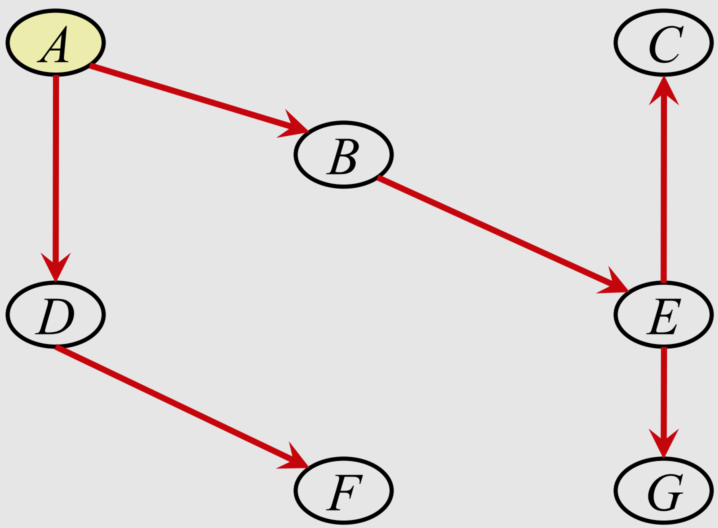
vi.



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

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

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

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

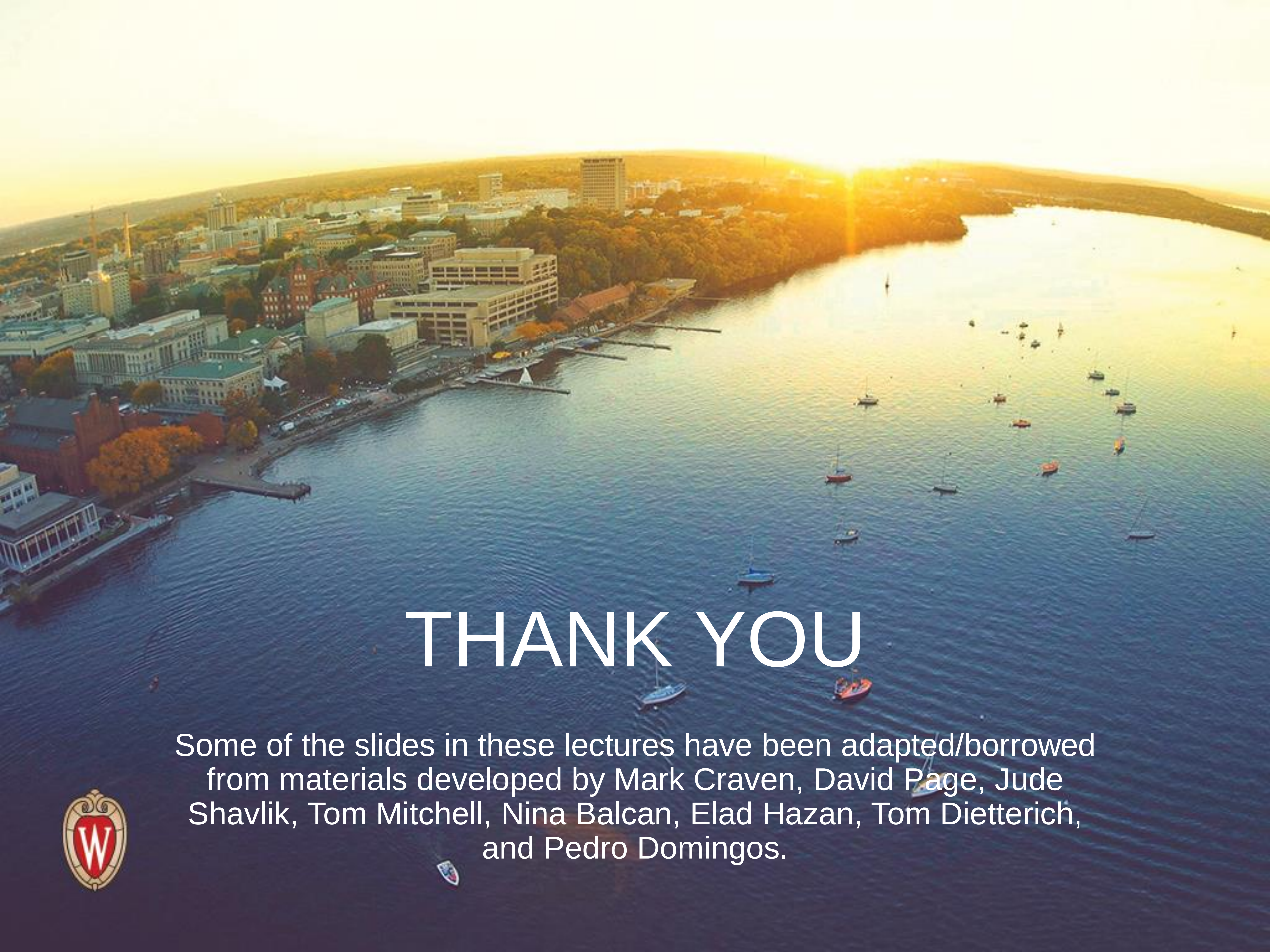
$$\arg\max_G \log_2 P(D \mid G, \theta_G) = \arg\max_G \sum_i I(X_i, Parents(X_i))$$

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

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

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

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



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.