

HMMs

Machine Learning – 10701/15781

Carlos Guestrin

Carnegie Mellon University

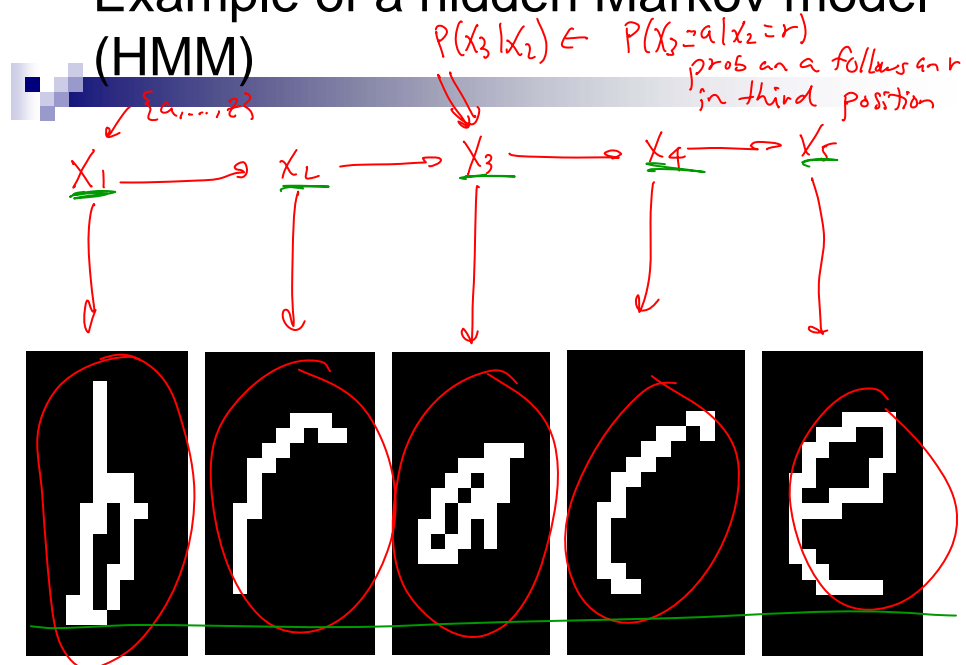
November 7th, 2007

©2005-2007 Carlos Guestrin

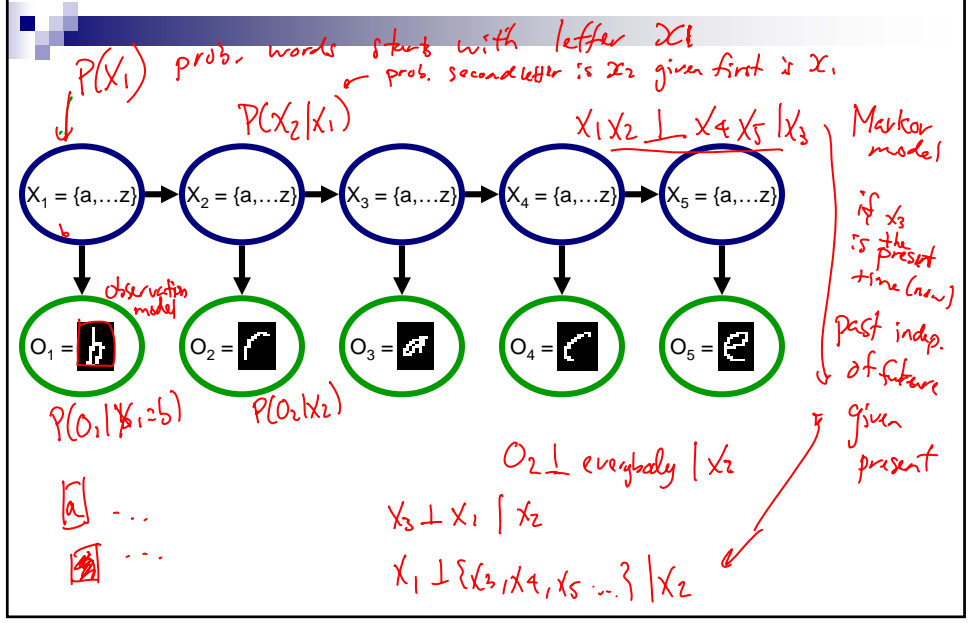
1

Example of a hidden Markov model

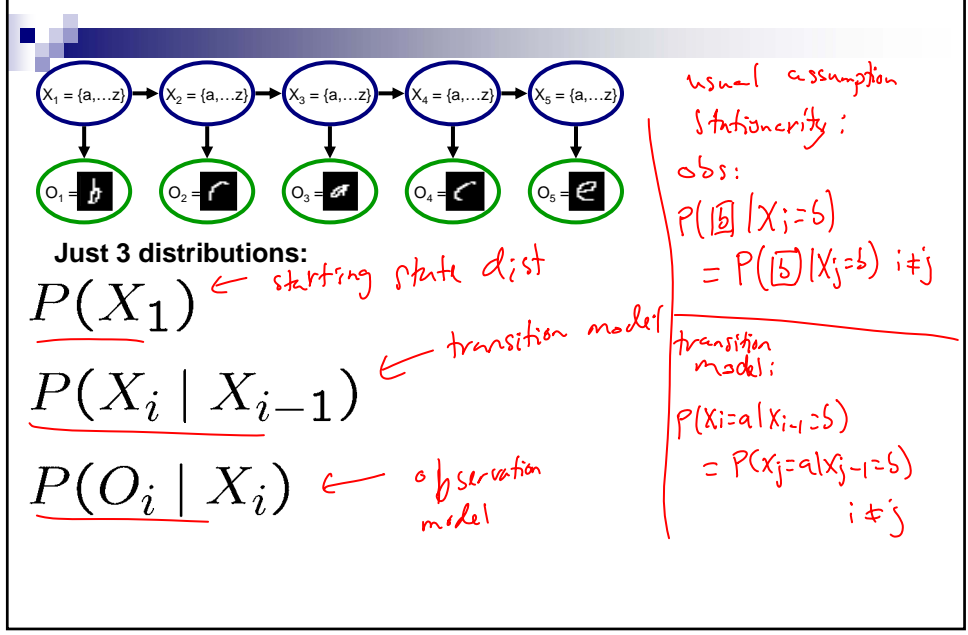
(HMM)



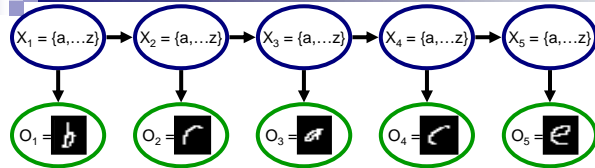
Understanding the HMM Semantics



HMMs semantics: Details



HMMs semantics: Joint distribution



$$P(X_1)$$

$$P(X_i | X_{i-1})$$

$$P(O_i | X_i)$$

$$P(X_1, \dots, X_n, o_1, \dots, o_n) = P(X_1) \cdot P(o_1 | X_1) \cdot P(X_2 | X_1) \cdot P(o_2 | X_2) \dots$$

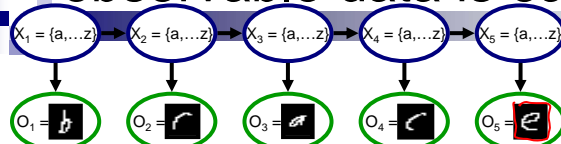
$$= P(X_1) \cdot P(o_1 | X_1) \prod_{t=2}^n P(X_t | X_{t-1}) P(o_t | X_t)$$

Given $O = \{b, r, a, c, e\}$

$$P(X_1, \dots, X_n | o_1, \dots, o_n) = P(X_{1:n} | o_{1:n})$$

$$\propto P(X_1) P(o_1 | X_1) \prod_{i=2}^n P(X_i | X_{i-1}) P(o_i | X_i)$$

Learning HMMs from fully observable data is easy



$$P(O_i = b | X_i = b) = \prod_{j \in \text{pixels}} P(O_{ij} = \dots | X_i = b)$$

Learn 3 distributions:

$$P(X_1 = b) = \frac{\text{Count}(X_1 = b)}{\text{# of words}}$$

$$P(O_i = b | X_i = b) = \frac{\text{Count}(X_i = b, O_i = b)}{\text{Count}(X_i = b)}$$

$$P(X_i = b | X_{i-1} = a) = \frac{\text{Count}(X_i = b, X_{i-1} = a)}{\text{Count}(X_{i-1} = a)}$$

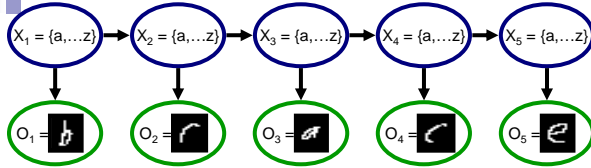
$$\propto \frac{P(X_i = b, X_{i-1} = a)}{P(X_{i-1} = a)}$$

only see b once... need make some assumption about $P(O_i | X_i)$

how often a b follows a n a

how often I see an a, but not in last position in word

Possible inference tasks in an HMM



forwards-backwards

Marginal probability of a hidden variable:

$$P(X_3 = b \mid O_1 = b, O_2 = c, O_3 = a, O_4 = c, O_5 = e)$$

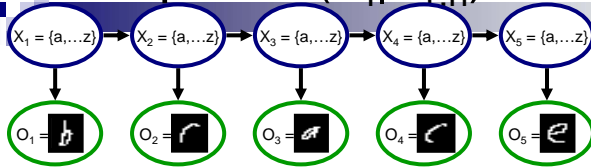
$$P(X_3 \mid O_{1..5})$$

Viterbi decoding – most likely trajectory for hidden vars:

$$\underset{x_1, x_2, x_3, x_4, x_5}{\text{argmax}} P(x_1, \dots, x_5 \mid O_1 = b, O_2 = c, O_3 = a, O_4 = c, O_5 = e)$$

maximum likelihood for entire word

Using variable elimination to compute $P(X_i \mid o_{1..n})$



Compute:

$$P(X_i \mid o_{1..n})$$

Variable elimination order?

1, 2, 4, 5

$$P(X_3 \mid O_{1..5})$$

complexity $O(n)$

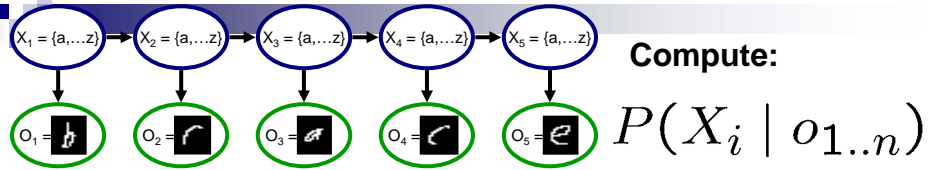
Example:

$$P(x_3, o_{1..5}) = \sum_{x_1, x_2, x_4, x_5} P(x_1) P(o_1 | x_1) P(x_2 | x_1) P(o_2 | x_2) \dots$$

$$= \sum_{x_2, x_4, x_5} P(o_2 | x_2) P(x_3 | x_2) P(o_3 | x_3) \dots \sum_{x_1} P(x_1) \cdot P(o_1 | x_1) \cdot P(x_2 | x_1)$$

$P(x_2, o_1) = g_2(x_2)$

What if I want to compute $P(X_i | o_{1:n})$ for each i ?



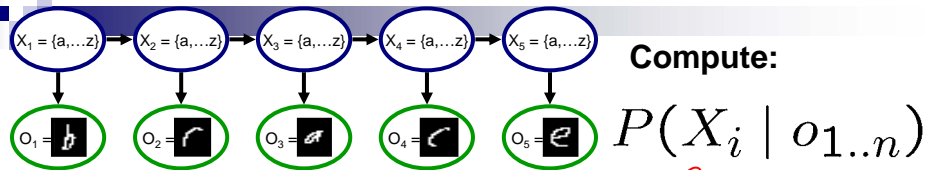
Variable elimination for ~~each~~ ^{one} i ? $O(n)$

Variable elimination for each i , what's the complexity?

↳ naively $\leftarrow O(n^2)$

↖ but if you are smart, you can do it in $O(n)$

Reusing computation



$$P(X_5, o_{1:5}) = \sum_{x_1, \dots, x_4} P(x_1) P(o_1 | x_1) P(x_2 | x_1) P(o_2 | x_2) \dots$$

$$= \sum_{x_2, x_3, x_4} P(o_2 | x_2) \dots \underbrace{\sum_{x_1} P(x_1) P(o_1 | x_1) P(x_2 | x_1)}_{g_1(x_2)}$$

$$P(X_4, o_{1:5}) = \sum_{x_2, x_3, x_5} P(o_2 | x_2) \dots \underbrace{\sum_{x_1} P(x_1) P(o_1 | x_1) P(x_2 | x_1)}_{g_1(x_2)}$$

The forwards-backwards algorithm

$P(X_i | o_{1..n})$

- Initialization: $\alpha_1(X_1) = P(X_1)P(o_1 | X_1)$ *forwards*
- For $i = 2$ to n
 - Generate a forwards factor by eliminating X_{i-1}
$$\alpha_i(X_i) = \sum_{x_{i-1}} P(o_i | X_i) P(X_i | X_{i-1} = x_{i-1}) \alpha_{i-1}(x_{i-1})$$
- Initialization: $\beta_n(X_n) = 1$
- For $i = n-1$ to 1
 - Generate a backwards factor by eliminating X_{i+1}
$$\beta_i(X_i) = \sum_{x_{i+1}} P(o_{i+1} | x_{i+1}) P(x_{i+1} | X_i) \beta_{i+1}(x_{i+1})$$
- δ_i , probability is: $P(X_i | o_{1..n}) \propto \alpha_i(X_i) \beta_i(X_i)$

What you'll implement 1: multiplication

$$\alpha_i(X_i) = \sum_{x_{i-1}} P(o_i | X_i) P(X_i | X_{i-1} = x_{i-1}) \alpha_{i-1}(x_{i-1})$$

$f(X_i = a | X_{i-1} = b) = P(o_i | X_i = a) \cdot P(X_i = a | X_{i-1} = b) \cdot \alpha_{i-1}(X_{i-1} = b)$

table for $f(x_i, x_{i-1})$

x_{i-1}	x_i	a	b	c	...	z
26	a					
	b					
	⋮					
	z					

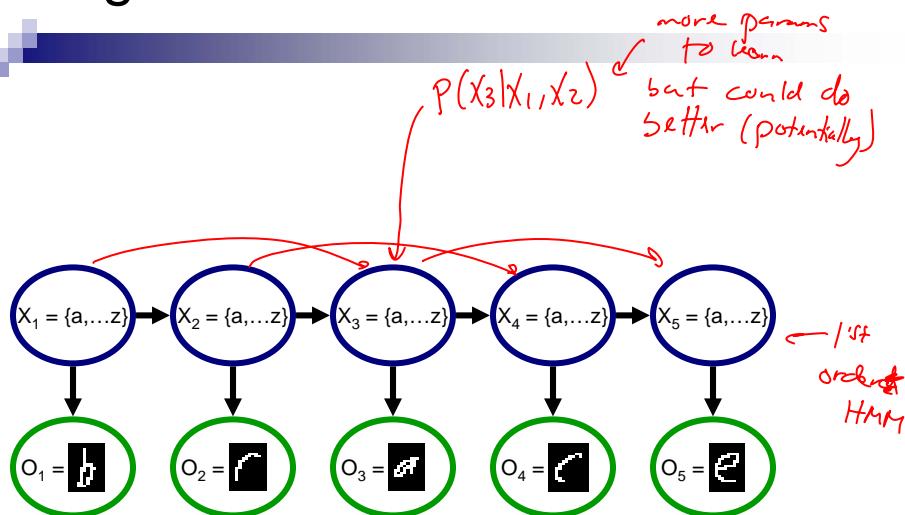
What you'll implement 2: marginalization

$$\alpha_i(X_i) = \sum_{x_{i-1}} P(o_i | X_i) P(X_i | X_{i-1} = x_{i-1}) \alpha_{i-1}(x_{i-1})$$

$$f(x_i, x_{i-1})$$

$$\alpha_i(x_i = a) = \sum_{x_{i-1}} f(x_i = a, x_{i-1} = x_{i-1})$$

Higher-order HMMs



**Add dependencies further back in time !
better representation, harder to learn**

What you need to know

■ Hidden Markov models (HMMs)

- Very useful, very powerful!
- Speech, OCR,...
- Parameter sharing, only learn 3 distributions
- Trick reduces inference from $O(n^2)$ to $O(n)$
- Special case of BN

Kalman Filter: HMM, with $P(x_i|x_{i-1})$ } *conditional*
 $P(\theta_i|x_i)$ } *Gaussian*



Bayesian Networks (Structure) Learning

Machine Learning – 10701/15781

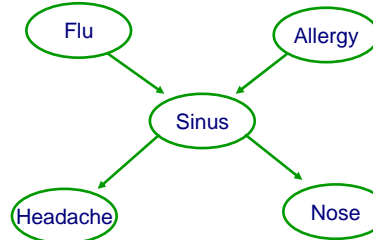
Carlos Guestrin

Carnegie Mellon University

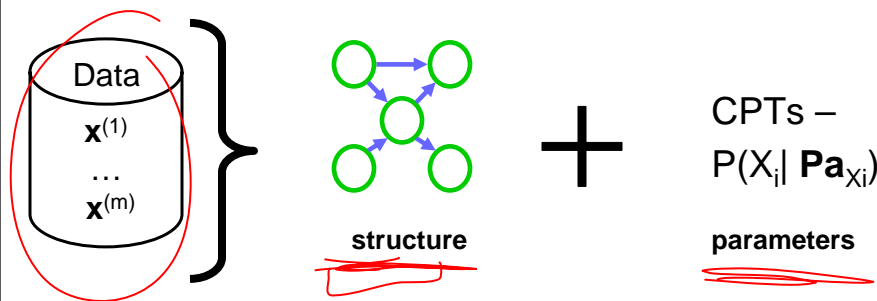
November 7th, 2007

Review

- Bayesian Networks
 - Compact representation for probability distributions
 - Exponential reduction in number of parameters
- Fast probabilistic inference using variable elimination
 - Compute $P(X|e)$
 - Time exponential in tree-width, not number of variables
- Today
 - Learn BN structure

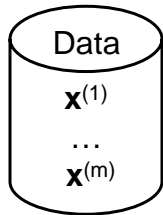


Learning Bayes nets



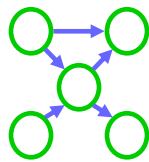
Learning the CPTs

$P(S|F,A)$



For each discrete variable X_i

$$P(X_i=x_i | P_{X_i}=u) \stackrel{MLE}{=} \frac{\text{Count}(X_i=x_i, P_{X_i}=u)}{\text{Count}(P_{X_i}=u)}$$



$$MLE: P(X_i = x_i | X_j = x_j) = \frac{\text{Count}(X_i = x_i, X_j = x_j)}{\text{Count}(X_j = x_j)}$$

Information-theoretic interpretation of maximum likelihood 1

Given structure, log likelihood of data:

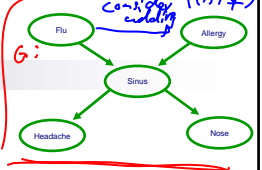
$$\log P(\mathcal{D} | \theta_G, \mathcal{G}) = \log \prod_{i=1}^n P(x^{(i)} | \theta_G, \mathcal{G}) = \sum_i \log P(x^{(i)} | \theta_G, \mathcal{G})$$

$$= \sum_i \log P(f^{(i)}) P(a^{(i)}) P(s^{(i)} | f^{(i)}, a^{(i)}) P(h^{(i)} | s^{(i)}) P(n^{(i)} | s^{(i)})$$

$$= \underbrace{\left[\sum_i \log P(f^{(i)}) \right]}_{P(F)} + \underbrace{\left[\sum_i \log P(a^{(i)}) \right]}_{P(A)} + \underbrace{\left[\sum_i \log P(s^{(i)} | f^{(i)}, a^{(i)}) \right]}_{P(S|F,A)} + \underbrace{\left[\sum_i \log P(h^{(i)} | s^{(i)}) \right]}_{P(H|S)} + \underbrace{\left[\sum_i \log P(n^{(i)} | s^{(i)}) \right]}_{P(N|S)}$$

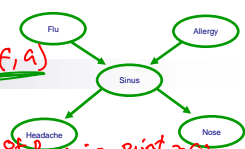
one per CPT $P(X_i | P_{X_i})$

only this term changes about $P(A|F)$



Information-theoretic interpretation of maximum likelihood 2

- Given structure, log likelihood of data:



$$\log P(\mathcal{D} | \theta_{\mathcal{G}}, \mathcal{G}) = \sum_{j=1}^m \sum_{i=1}^n \log P(X_i = x_i^{(j)} | \text{Pa}_{X_i} = x^{(j)} [\text{Pa}_{X_i}])$$

Handwritten notes: $P(S|F,A)$, $\text{value of } Pa_{X_i} \text{ in point } z^{(j)}$, $\text{print } j$, $\text{children parents nodes}$

$$= \sum_{i=1}^n \sum_{j=1}^m \log P(X_i = x_i^{(j)} | \text{Pa}_{X_i} = z^{(j)} [\text{Pa}_{X_i}])$$

Handwritten notes: for point j where $F=t, A=f, S=t$ (Count $(F=t, A=f, S=t)$)

$$= \sum_{i=1}^n \sum_{x_i} \sum_u \frac{\text{count}(X_i = x_i, \text{Pa}_{X_i} = u)}{m} \log P(X_i = x_i | \text{Pa}_{X_i} = u)$$

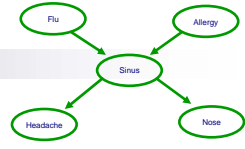
Handwritten notes: $\log P(S=t | F=t, A=f)$

$$\hat{P}(X_i = x_i, \text{Pa}_{X_i} = u)$$

MLE estimate

Information-theoretic interpretation of maximum likelihood 3

- Given structure, log likelihood of data:



$$\log \hat{P}(\mathcal{D} | \theta, \mathcal{G}) = m \sum_i \sum_{x_i, \text{Pa}_{x_i, \mathcal{G}}} \hat{P}(x_i, \text{Pa}_{x_i, \mathcal{G}}) \log \hat{P}(x_i | \text{Pa}_{x_i, \mathcal{G}})$$

Handwritten notes: hat because MLE, \hat{P} , $=u$, $=u$

$$= m \sum_i H(X_i | \text{Pa}_{X_i})$$

Handwritten notes: MLE, trial to pick parents s.t. $H(X_i | \text{Pa}_{X_i})$ small; little uncertainty about $X_i | \text{Pa}_{X_i}$

$$= m \sum_i (I(X_i, \text{Pa}_{X_i}) - H(X_i))$$

$$H(A|B) = - \sum_{a,b} P(a,b) \cdot \log P(a|b)$$

$$I(A, B) = - \sum_{a,b} P(a,b) \log \frac{P(a,b)}{P(a)P(b)}$$

$$= H(A) - H(A|B)$$

Decomposable score

- Log data likelihood

$$\log \hat{P}(\mathcal{D} | \theta, \mathcal{G}) = m \sum_i \hat{I}(x_i, \mathbf{Pa}_{x_i, \mathcal{G}}) - M \sum_i \hat{H}(X_i)$$

- Decomposable score:

- Decomposes over families in BN (node and its parents)
- Will lead to significant computational efficiency!!!
- $\text{Score}(G : D) = \sum_i \text{FamScore}(X_i | \mathbf{Pa}_{X_i} : D)$

How many trees are there?

Nonetheless – Efficient optimal algorithm finds best tree

Scoring a tree 1: equivalent trees

$$\log \hat{P}(\mathcal{D} | \theta, \mathcal{G}) = M \sum_i \hat{I}(x_i, \text{Pa}_{x_i, \mathcal{G}}) - M \sum_i \hat{H}(X_i)$$

Scoring a tree 2: similar trees

$$\log \hat{P}(\mathcal{D} | \theta, \mathcal{G}) = M \sum_i \hat{I}(x_i, \text{Pa}_{x_i, \mathcal{G}}) - M \sum_i \hat{H}(X_i)$$

Chow-Liu tree learning algorithm 1

- For each pair of variables X_i, X_j
 - Compute empirical distribution:

$$\bar{P}(x_i, x_j) = \frac{\text{Count}(x_i, x_j)}{m}$$

- Compute mutual information:

$$\bar{I}(X_i, X_j) = \sum_{x_i, x_j} \bar{P}(x_i, x_j) \log \frac{\bar{P}(x_i, x_j)}{\bar{P}(x_i)\bar{P}(x_j)}$$

- Nodes X_1, \dots, X_n
- Edge (i,j) gets weight

$$\hat{I}(X_i, X_j)$$

Chow-Liu tree learning algorithm 2

$$\log \hat{P}(\mathcal{D} | \theta, \mathcal{G}) = M \sum_i \hat{I}(x_i, \text{Pa}_{x_i, \mathcal{G}}) - M \sum_i \hat{H}(X_i)$$

- Optimal tree BN
 - Compute maximum weight spanning tree
 - Directions in BN: pick any node as root, breadth-first-search defines directions

Can we extend Chow-Liu 1

- Tree augmented naïve Bayes (TAN) [Friedman et al. '97]
 - Naïve Bayes model overcounts, because correlation between features not considered
 - Same as Chow-Liu, but score edges with:

$$\hat{I}(X_i, X_j | C) = \sum_{c, x_i, x_j} P(c, x_i, x_j) \log \frac{P(x_i, x_j | c)}{P(x_i | c)P(x_j | c)}$$

Can we extend Chow-Liu 2

- (Approximately learning) models with tree-width up to k
 - [Checheta & Guestrin '07]
 - But, $O(n^{2k+6})$...

What you need to know about learning BN structures so far

- Decomposable scores
 - Maximum likelihood
 - Information theoretic interpretation
- Best tree (Chow-Liu)
- Best TAN
- Nearly best k-treewidth (in $O(N^{2k+6})$)

Scoring general graphical models – Model selection problem

What's the best structure?



$\langle x_1^{(1)}, \dots, x_n^{(1)} \rangle$
...
 $\langle x_1^{(m)}, \dots, x_n^{(m)} \rangle$



The more edges, the fewer independence assumptions, the higher the likelihood of the data, but will overfit...

Maximum likelihood overfits!

$$\log \hat{P}(\mathcal{D} | \theta, \mathcal{G}) = M \sum_i \hat{I}(x_i, \text{Pa}_{x_i, \mathcal{G}}) - M \sum_i \hat{H}(X_i)$$

- Information never hurts:

- Adding a parent always increases score!!!

Bayesian score avoids overfitting

- Given a structure, distribution over parameters
$$\log P(D | \mathcal{G}) = \log \int_{\theta_{\mathcal{G}}} P(D | \mathcal{G}, \theta_{\mathcal{G}}) P(\theta_{\mathcal{G}} | \mathcal{G}) d\theta_{\mathcal{G}}$$
- Difficult integral: use Bayes information criterion (BIC) approximation (equivalent as M! 1)
$$\log P(D | \mathcal{G}) \approx \log P(D | \mathcal{G}, \theta_{\mathcal{G}}) - \frac{\text{NumberParams}(\mathcal{G})}{2} \log M + \mathcal{O}(1)$$

- Note: regularize with MDL score
- Best BN under BIC still NP-hard

Structure learning for general graphs

- In a tree, a node only has one parent
- **Theorem:**
 - The problem of learning a BN structure with at most d parents is **NP-hard for any (fixed) $d \geq 2$**
- Most structure learning approaches use heuristics
 - Exploit score decomposition
 - (Quickly) Describe two heuristics that exploit decomposition in different ways

Learn BN structure using local search

Starting from
Chow-Liu tree

Local search,
possible moves:

- Add edge
- Delete edge
- Invert edge

Score using BIC

What you need to know about learning BNs

- Learning BNs

- Maximum likelihood or MAP learns parameters
- Decomposable score
- Best tree (Chow-Liu)
- Best TAN
- Other BNs, usually local search with BIC score