Classic HMM tutorial – see class website:

*L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proc. of the IEEE, Vol.77, No.2, pp.257--286, 1989.

HMMs (cont.)

Machine Learning – 10701/15781

Carlos Guestrin

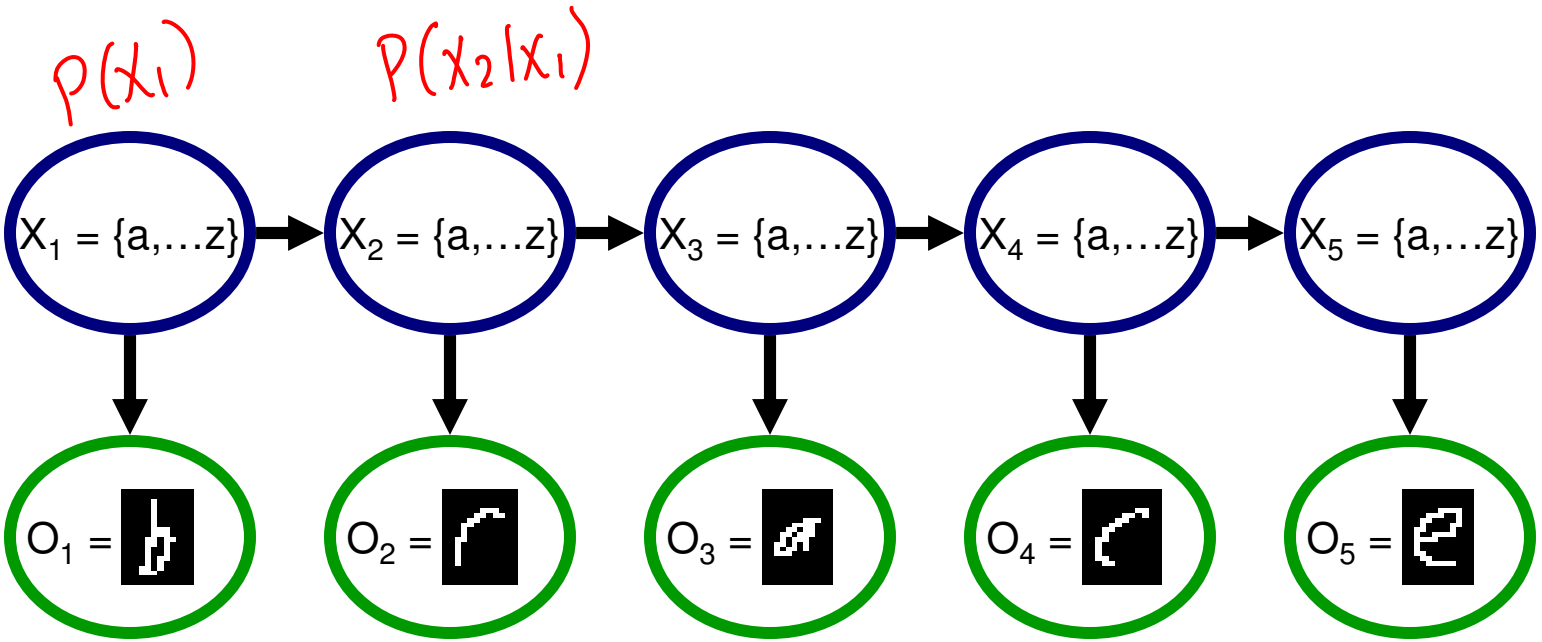
Carnegie Mellon University

March 29th, 2006

Announcements

- This weeks recitation:
  - Go through several BNs topics, representation, inference, learning, in the context of an example $\rightarrow$ very useful for homework

Understanding the HMM Semantics



$P(X_1)$

$P(X_2|X_1)$

$P(O_1|X_1)$

one option is Naive Bayes

$P(O_1|X_1) = \prod_{\text{pixels } i} P(O_i|X_1)$

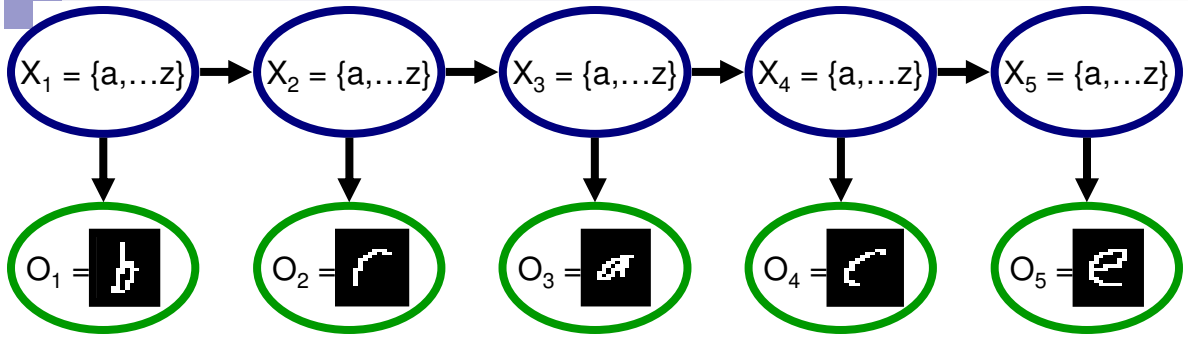
Markov assumption: the future is indep. of past given present

$X_{1:t} \perp X_{t+1:N} \mid X_t$

$O_t \perp$ everybody $\mid X_t$

basic!

HMMs semantics: Details



Just 3 distributions:

$$P(X_1)$$

$$P(X_i \mid X_{i-1})$$

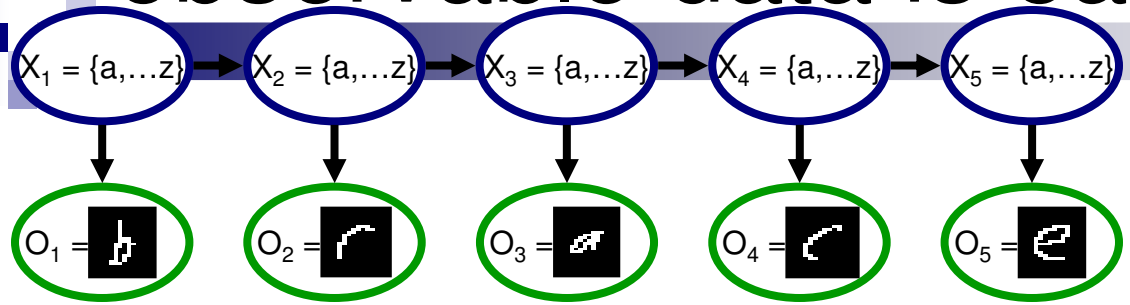
$$P(O_i \mid X_i)$$

observing a after b has same prob. no matter where in word

a distribution $P(X_3 \mid X_2) \sim P(X_2 \mid X_1) \sim P(X_{10} \mid X_9)$

prob. image given letter is same for all positions in word.

Learning HMMs from fully observable data is easy



Learn 3 distributions:

$$P(X_1 = a) = \frac{\text{Count}(\#\text{ first letter was } a)}{\#\text{= dataset size}}$$

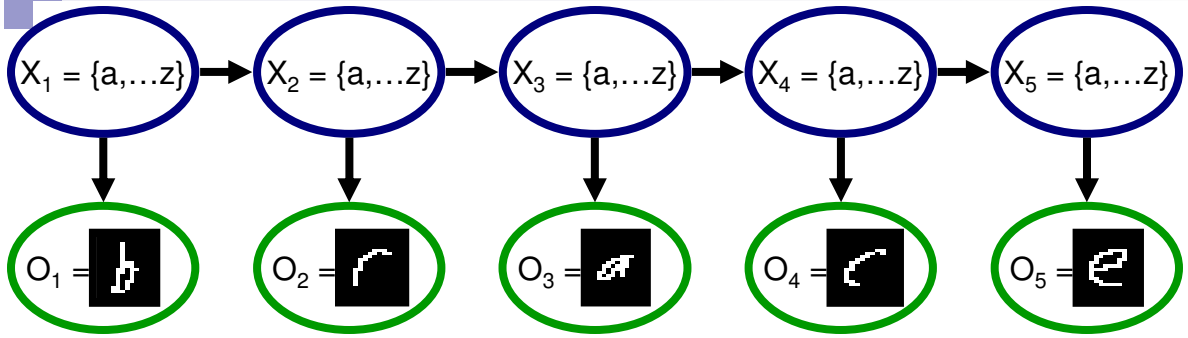
select training data where letter was a

$$P(O_i = \text{pixel 17 is white} \mid X_i = a) = \frac{\text{Count}(\text{pixel 17 was white}, X_i = a)}{\text{Count}(X_i = a)}$$

Count in any position in word

$$P(X_i = a \mid X_{i-1} = b) = \frac{\text{Count}(a \text{ appears after } b)}{\text{Count}(\#\text{ of } b\text{'s that are not at the end of the word})}$$

Possible inference tasks in an HMM



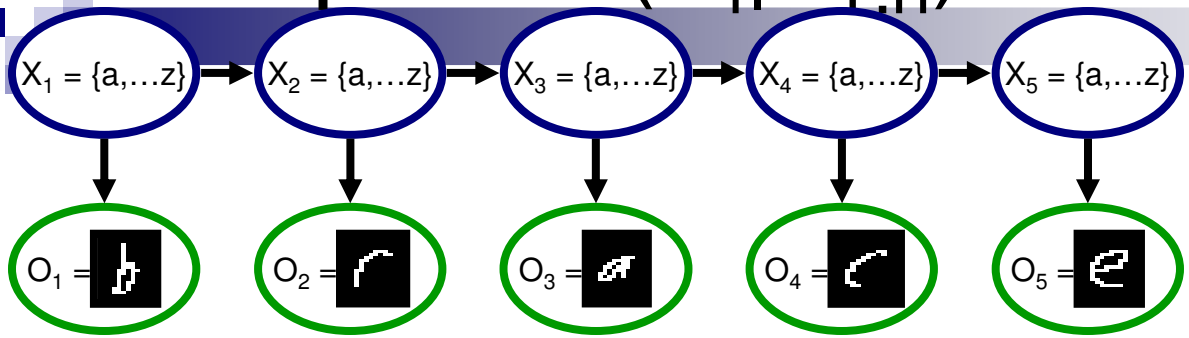
Marginal probability of a hidden variable:

$$p(X_i \mid O_1 = \boxed{b}, O_2 = \boxed{r}, O_3 = \boxed{a}, \ldots)$$

Viterbi decoding – most likely trajectory for hidden vars:

$$\underset{x_1 x_2 x_3 x_4 x_5}{\operatorname{argmax}} \; p(x_1, x_2, x_3, x_4, x_5 \mid O_{1:5})$$

Using variable elimination to compute $P(X_i|o_{1:n})$



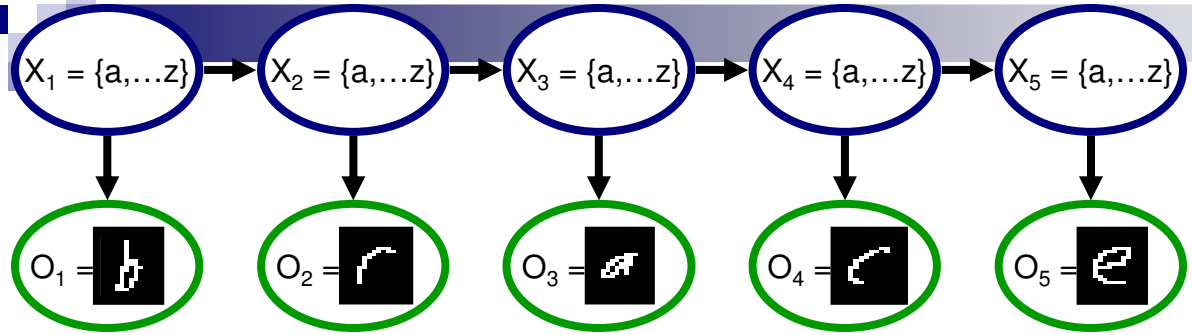
Compute:

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

Variable elimination order?

Example:

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



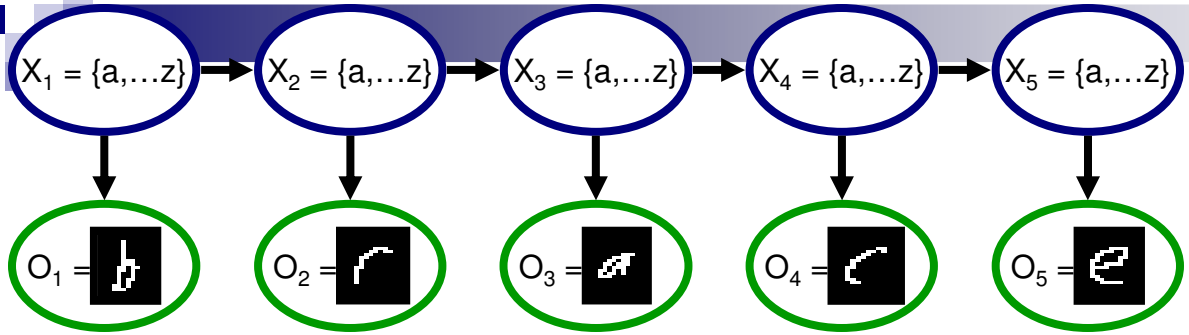
Compute:

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

Variable elimination for each i?

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

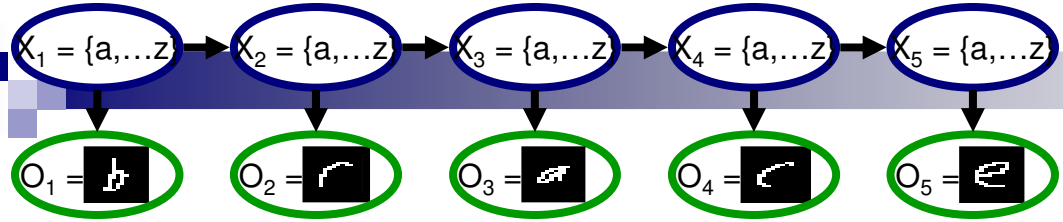
Reusing computation



Compute:

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

The forwards-backwards algorithm



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

- Initialization: $\alpha_1(X_1) = P(X_1)P(o_1 \mid X_1)$
- For i = 2 to n
  - Generate a forwards factor by eliminating X_{i-1}

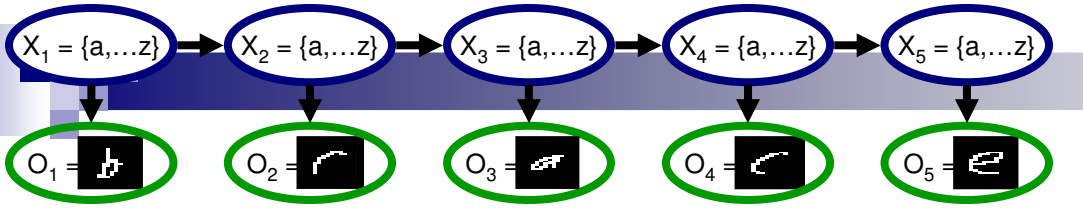
$$\alpha_i(X_i) = \sum_{x_{i-1}} P(o_i \mid X_i)P(X_i \mid X_{i-1} = x_{i-1})\alpha_{i-1}(x_{i-1})$$

- Initialization: $\beta_n(X_n) = 1$
- For i = n-1 to 1
  - Generate a backwards factor by eliminating X_{i+1}

$$\beta_i(X_i) = \sum_{x_{i+1}} P(o_{i+1} \mid x_{i+1})P(x_{i+1} \mid X_i)\beta_{i+1}(x_{i+1})$$

- $\forall$ i, probability is: $\boxed{P(X_i \mid o_{1..n}) = \alpha_i(X_i)\beta_i(X_i)}$

Most likely explanation

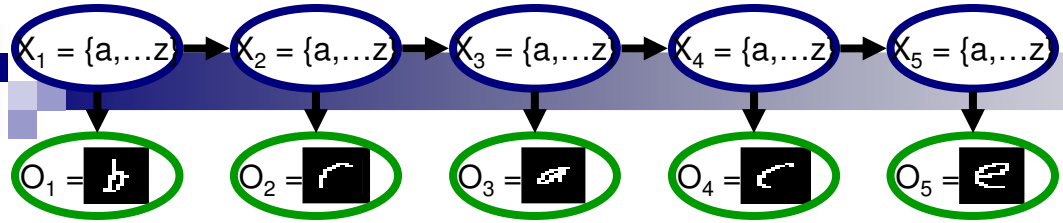


Compute:

Variable elimination order?

Example:

The Viterbi algorithm



- Initialization: $\alpha_1(X_1) = P(X_1)P(o_1 \mid X_1)$
- For i = 2 to n
  - Generate a forwards factor by eliminating X_{i-1}

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

- Computing best explanation: $x_n^* = \operatorname{argmax}_{x_n} \alpha_n(x_n)$
- For i = n-1 to 1
  - Use argmax to get explanation:

$$x_i^* = \operatorname{argmax}_{x_i} P(x_{i+1}^* \mid x_i)\alpha_i(x_i)$$

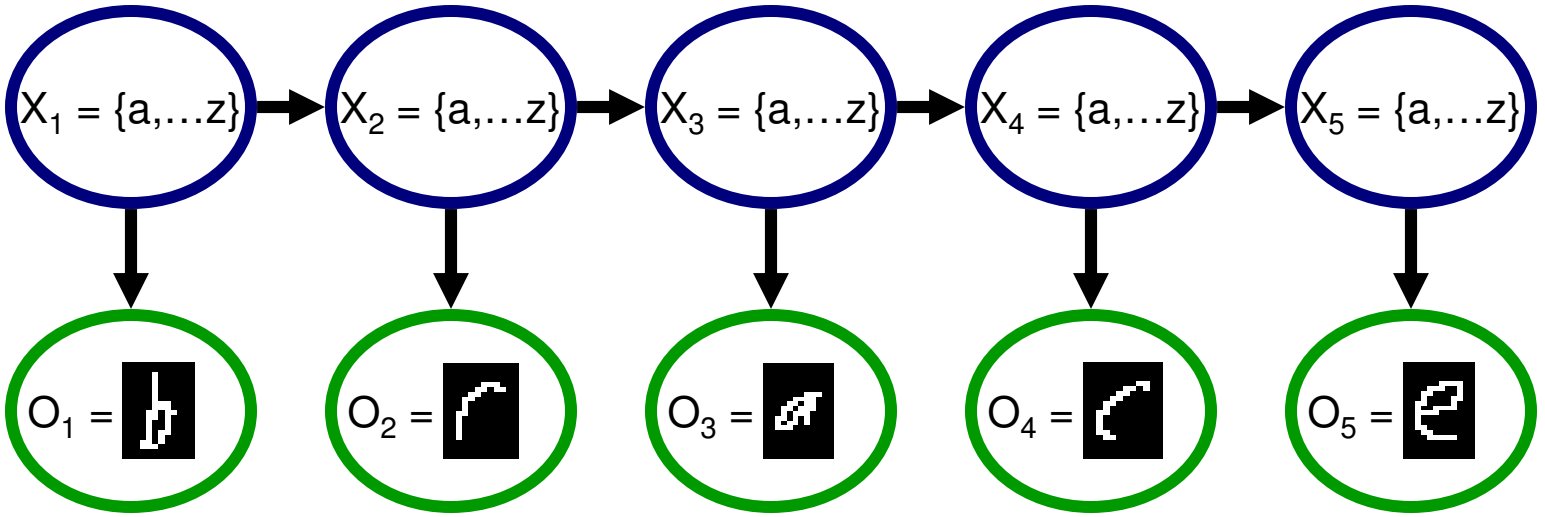
What you'll implement 1: multiplication

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

What you'll implement 2: max & argmax

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

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

Koller & Friedman Chapters (handed out):

Chapter 11 (short)

Chapter 12: 12.1, 12.2, 12.3 (covered in the beginning of semester)

12.4 (Learning parameters for BNs)

Chapter 13: 13.1, 13.3.1, 13.4.1, 13.4.3 (basic structure learning)

Learning BN tutorial (class website):

ftp://ftp.research.microsoft.com/pub/tr/tr-95-06.pdf

TAN paper (class website):

http://www.cs.huji.ac.il/~nir/Abstracts/FrGG1.html

Bayesian Networks – (Structure) Learning

Machine Learning – 10701/15781

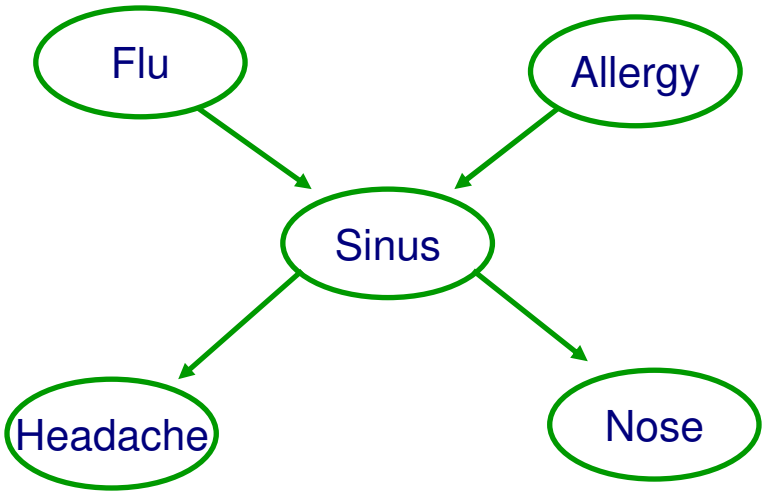
Carlos Guestrin

Carnegie Mellon University

March 29th, 2006

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

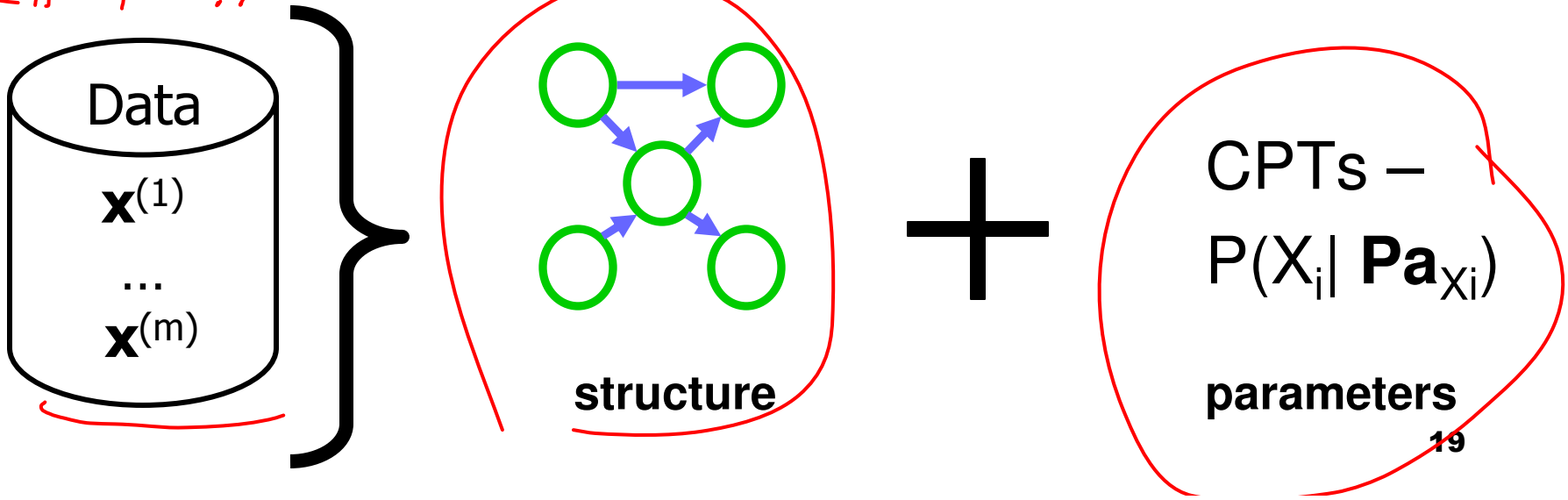


Learning Bayes nets

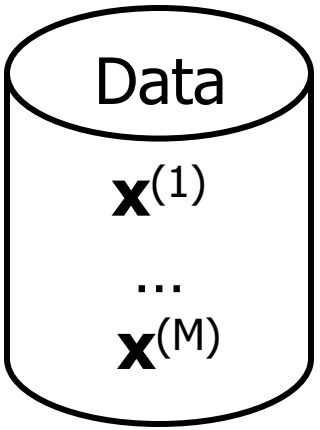
	Known structure	Unknown structure
Fully observable data	easy!! :)	NP-hard (not always) We'll see next week
Missing data	hard, in 2 weeks	really hard, next semester

A, B, C

<A=a, B=?, C=c>



Learning the CPTs

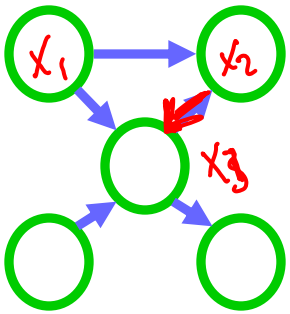


For each discrete variable X_i

$P(X_3 | X_1, X_2)$

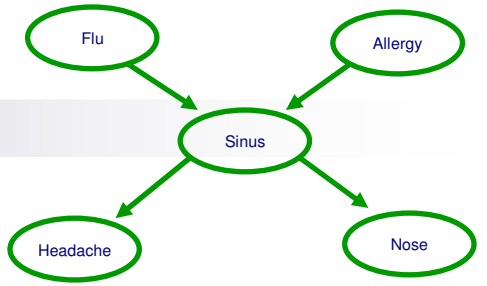
$P(X_3 = x_3 | X_1 = x_1, X_2 = x_2) = \frac{\text{Count}(X_3 = x_3, X_1 = x_1, X_2 = x_2)}{\text{Count}(X_1 = x_1, X_2 = x_2)}$

WHY??????????



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

Information-theoretic interpretation of maximum likelihood



- Given structure, log likelihood of data:

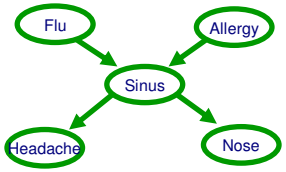
$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G})$$

Maximum likelihood (ML) for learning BN structure

Possible structures

Learn parameters using ML

Score structure



Data

$\langle x_1^{(1)},\ldots,x_n^{(1)}\rangle$

…

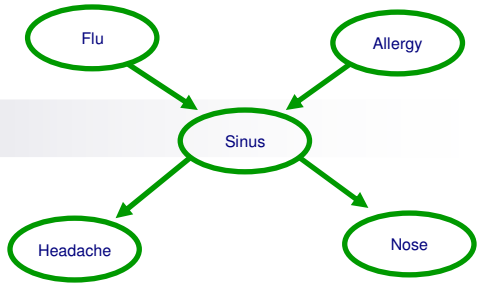
$\langle x_1^{(M)},\ldots,x_n^{(M)}\rangle$

How many trees are there?

$$\binom{n}{k} \geq \left(\frac{n}{k}\right)^k$$

Nonetheless – Efficient optimal algorithm finds best tree

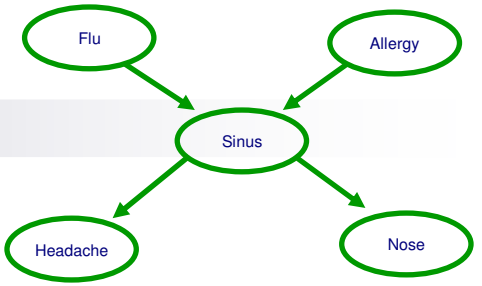
Information-theoretic interpretation of maximum likelihood 2



- Given structure, log likelihood of data:

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) = \sum_{i=1}^{n} \sum_{j=1}^{M} \log P(x_i^{(j)} \mid \mathbf{Pa}_{x_i,\mathcal{G}}^{(j)})$$

Information-theoretic interpretation of maximum likelihood 3



- Given structure, log likelihood of data:

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) = M \sum_{i=1}^{n} \sum_{x_i, \mathbf{Pa}_{x_i,\mathcal{G}}} \hat{P}(x_i, \mathbf{Pa}_{x_i,\mathcal{G}}) \log \hat{P}(x_i \mid \mathbf{Pa}_{x_i,\mathcal{G}})$$

Mutual information → Independence tests

- Statistically difficult task!
- Intuitive approach: **Mutual information**

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

- Mutual information and independence:
  - X_i and X_j independent if and only if $I(X_i,X_j)=0$

- Conditional mutual information:

Decomposable score

- Log data likelihood

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) = M \sum_i \hat{I}(X_i, \mathbf{Pa}_{X_i,\mathcal{G}}) - M \sum_i \hat{H}(X_i)$$

Scoring a tree 1: equivalent trees

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

Scoring a tree 2: similar trees

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

Chow-Liu tree learning algorithm 1

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

$$\hat{P}(x_i, x_j) = \frac{\text{Count}(x_i, x_j)}{M}$$

  - Compute mutual information:

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

- Define a graph
  - Nodes X_1,…,X_n
  - Edge (i,j) gets weight $\hat{I}(X_i, X_j)$

Chow-Liu tree learning algorithm 2

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

- 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:

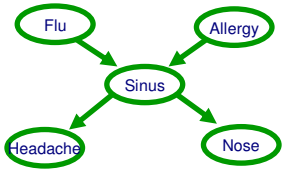
$$\hat{I}(X_i, X_j \mid C) = \sum_{c,x_i,x_j} \hat{P}(c, x_i, x_j) \log \frac{\hat{P}(x_i, x_j \mid c)}{\hat{P}(x_i \mid c)\hat{P}(x_j \mid c)}$$

Can we extend Chow-Liu 2

- (Approximately learning) models with tree-width up to *k*
  - [Narasimhan & Bilmes '04]
  - But, $O(n^{k+1})$…

Scoring general graphical models – Model selection problem

What's the best structure?



<x_1^{(1)},…,x_n^{(1)}>
…
<x_1^{(m)},…,x_n^{(m)}>

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

Maximum likelihood overfits!

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

- Information never hurts:

- Adding a parent always increases score!!!

Bayesian score avoids overfitting

- Given a structure, distribution over parameters

$$\log P(D \mid \mathcal{G}) = \log \int_{\theta_{\mathcal{G}}} P(D \mid \mathcal{G}, \theta_{\mathcal{G}}) P(\theta_{\mathcal{G}} | \mathcal{G}) d\theta_{\mathcal{G}}$$

- Difficult integral: use Bayes information criterion (BIC) approximation (equivalent as M$\rightarrow \infty$)

$$\log P(D \mid \mathcal{G}) \approx \log P(D \mid \mathcal{G}, \theta_{\mathcal{G}}) - \frac{\text{NumberParams}(\mathcal{G})}{2} \log M + \mathcal{O}(1)$$

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

How many graphs are there?

$$\sum_{k=1}^{n} \binom{n}{k} = 2^n - 1$$

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