HMMs

Machine Learning – 10701/15781

Carlos Guestrin

Carnegie Mellon University

March 28th, 2007

Adventures of our BN hero

- Compact representation for probability distributions
- Fast inference
- Fast learning

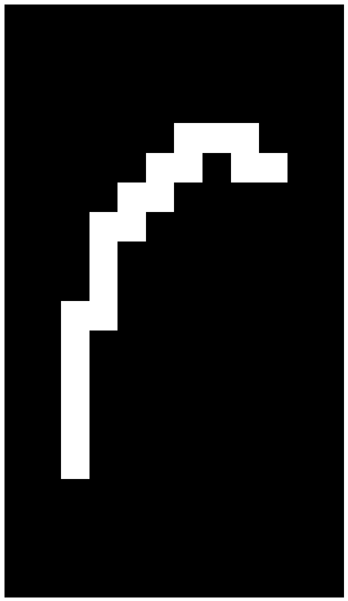
- But… Who are the most popular kids?

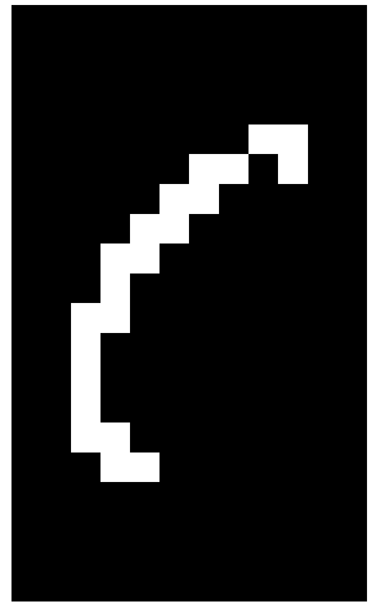
1. Naïve Bayes

2 and 3.
Hidden Markov models (HMMs)
Kalman Filters

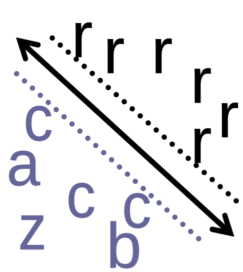
→ a HMM with Gaussians
→ Kalman Filter with discrete dist.

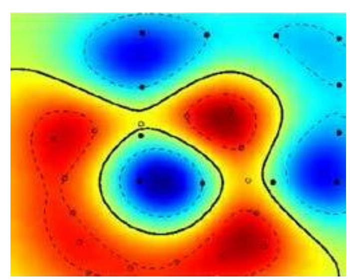
Handwriting recognition



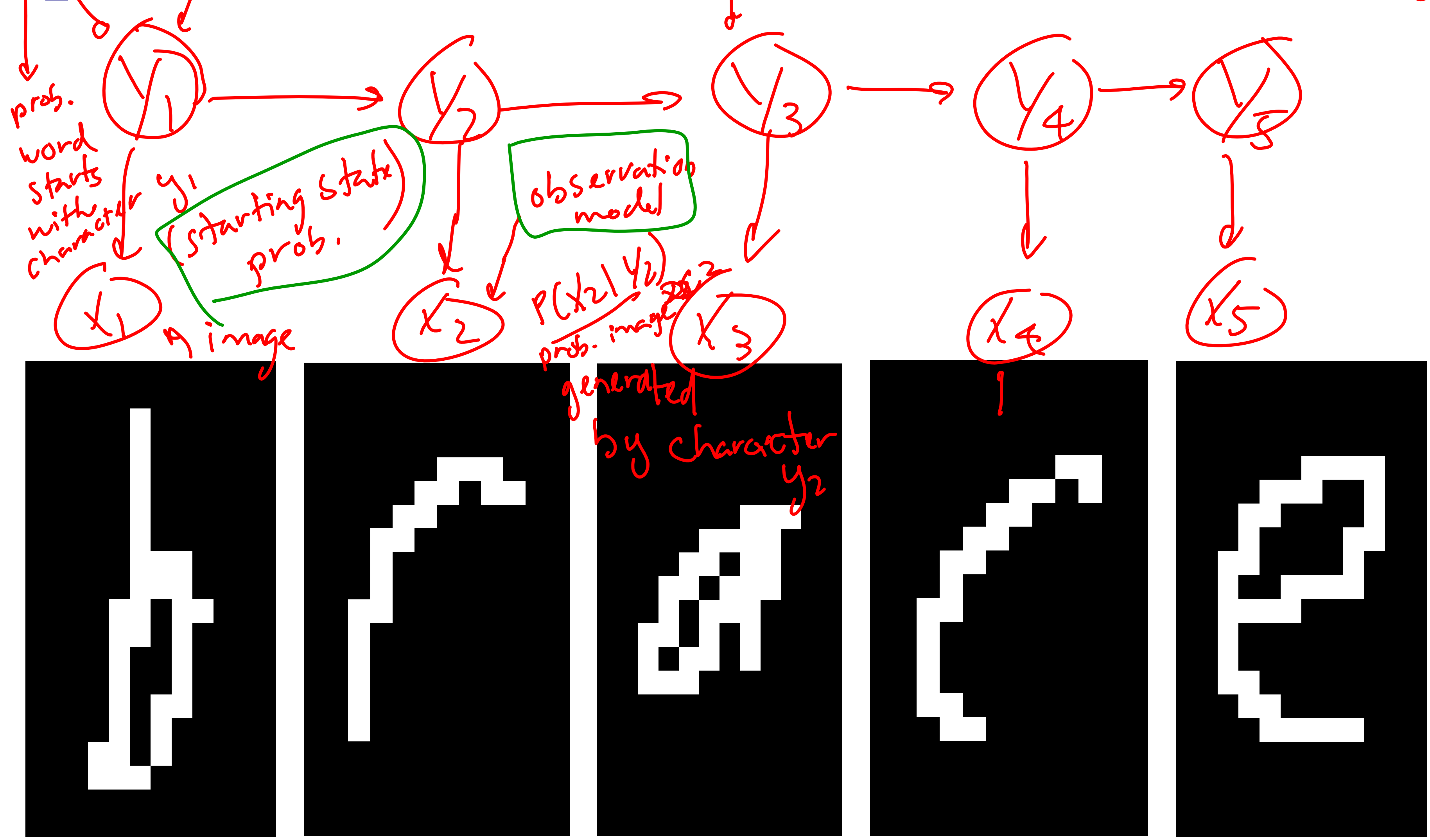


Character recognition, e.g., kernel SVMs

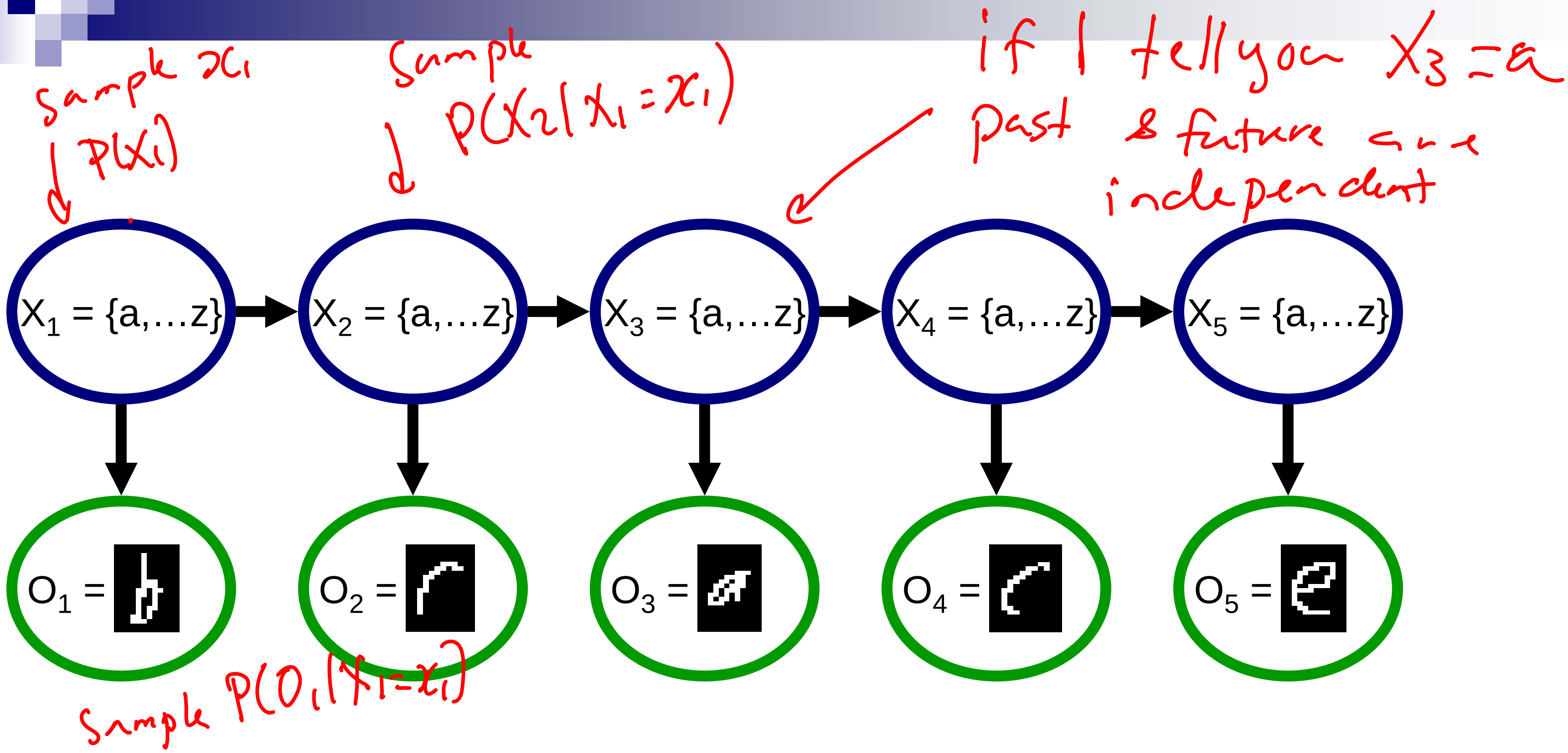




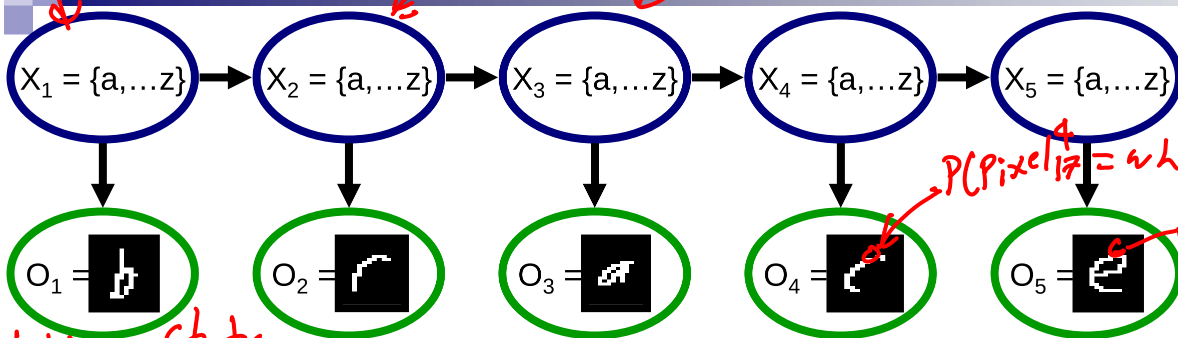
Example of a hidden Markov model (HMM)



Understanding the HMM Semantics



HMMs semantics: Details



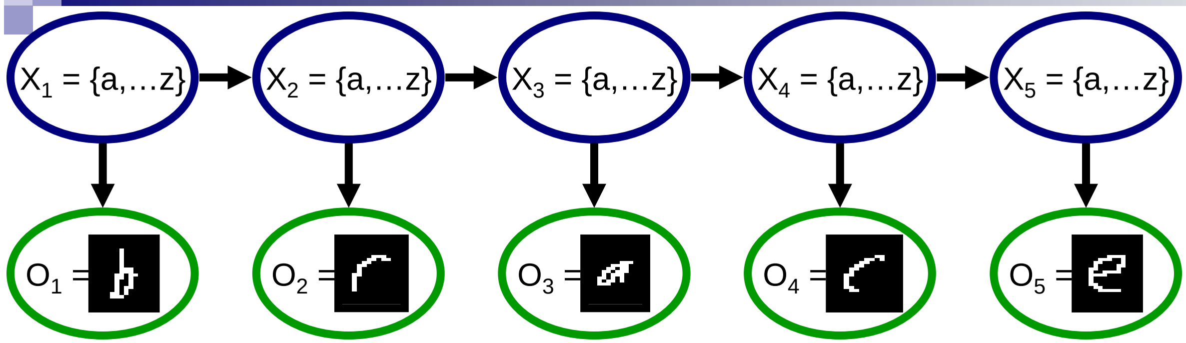
Just 3 distributions:

$$P(X_1)$$

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

$$P(O_i \mid X_i)$$

HMMs semantics: Joint distribution



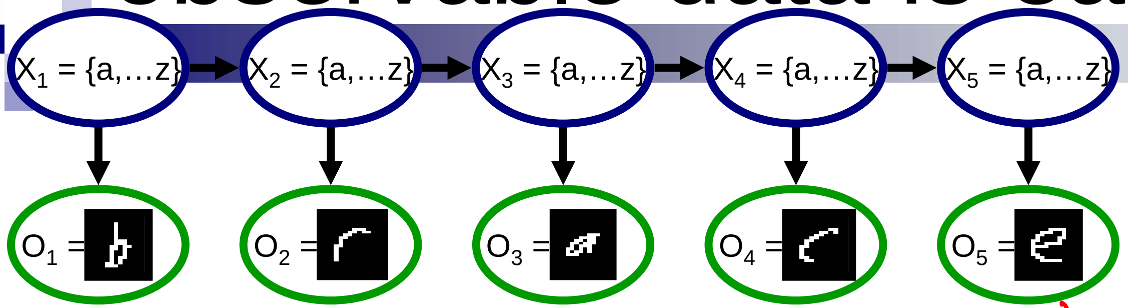
$$P(X_1)$$
$$P(X_i \mid X_{i-1})$$
$$P(O_i \mid X_i)$$

$$P(X_1, X_2, \dots X_5, O_1, \dots, O_5) = P(X_1) \cdot \prod_{i=2}^{5} P(X_i \mid X_{i-1}) \cdot \prod_{i=1}^{5} P(O_i \mid X_i)$$

$$= P(X_1) \cdot P(X_2 \mid X_1) \dots P(X_5 \mid X_4) \cdot P(O_1 \mid X_1) \dots P(O_5 \mid X_5)$$

$$P(X_1, \dots, X_n \mid o_1, \dots, o_n) = P(X_{1:n} \mid o_{1:n})$$
$$\propto P(X_1) P(o_1 \mid X_1) \prod_{i=2}^{n} P(X_i \mid X_{i-1}) P(o_i \mid X_i)$$

Learning HMMs from fully observable data is easy



Learn 3 distributions: (MLE sense)

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

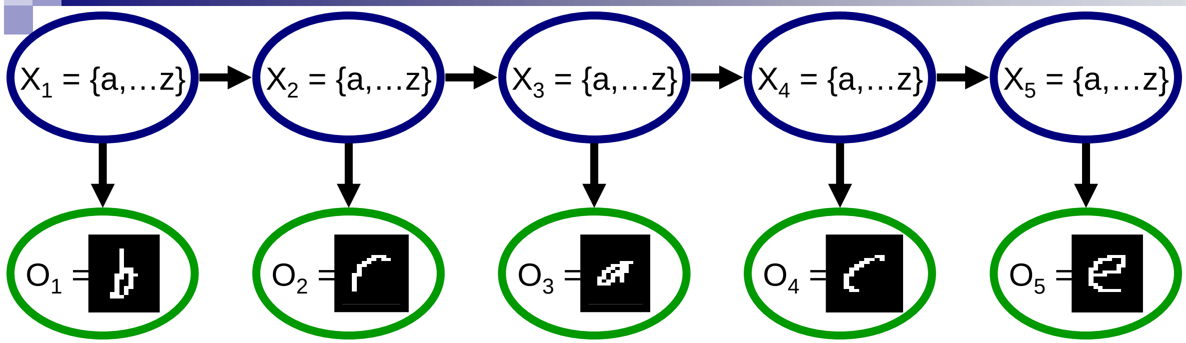
← starting state prob
count only first letter

$$P(O_i = 17 \mid X_i = a) = \frac{\text{Count}((\text{Character } X_i = a \text{ \& its Image } O_i = 17))}{\text{Count}(\text{Character } X_i = a)}$$

$$P(X_i = a \mid X_{i-1} = b) = \frac{\text{Count}((\text{Character } X_{i-1} = b, \text{ Next Character } X_i = a))}{\text{Count}(\text{Character } x_{i-1} = b)}$$

count how often character = b, but not last character in word.

Possible inference tasks in an HMM



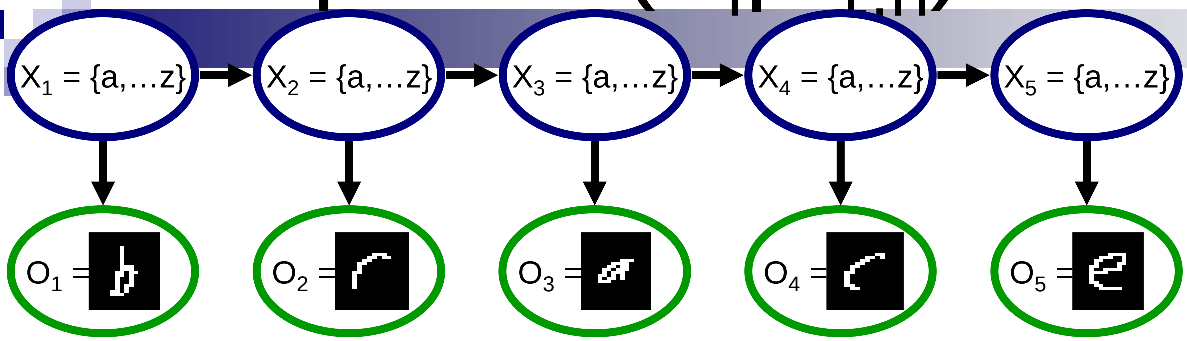
Marginal probability of a hidden variable:

$$P(X_i \mid O_1 = o_1, \ldots, O_n = o_n)$$

Viterbi decoding – most likely trajectory for hidden vars:

$$\max_{x_1 \ldots x_n} P(X_1 = x_1, \ldots, X_n = x_n \mid O_1 = o_1, \ldots, O_n = o_n)$$

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



Compute:

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

Variable elimination order?

$X_1 \ldots X_{i-1}, \quad X_n \ldots X_{i+1}$

Example:

$P(X_i \mid o_1 \ldots o_n) \propto P(X_i, o_1 \ldots o_n)$

$= \sum_{x_1 \ldots x_{i-1}} \sum_{x_{i+1} \ldots x_n} P(X_1) \cdot P(o_1|x_1) \ldots P(x_{i+1}|x_i) P(o_i|x_i) \ldots P(x_n|x_{n-1}) \cdot P(o_n|x_n)$

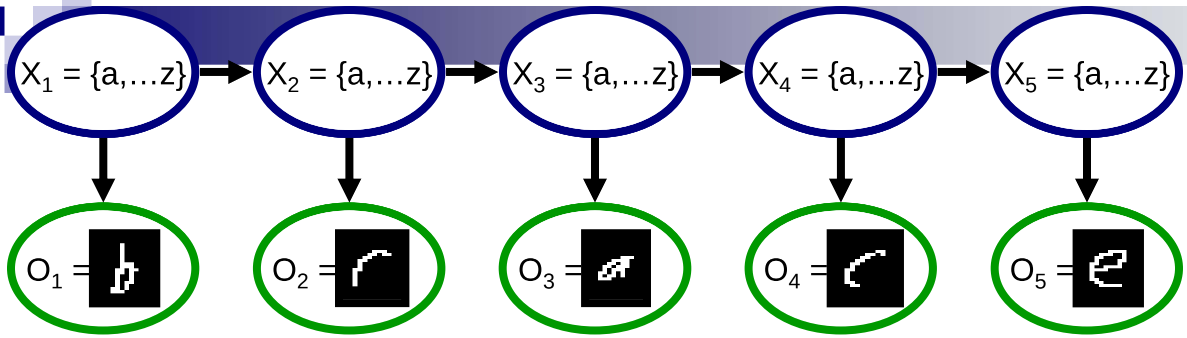
o_1 is fixed

eliminate X_1

$= \sum_{x_2 \ldots x_{i-1}} \sum_{x_{i+1} \ldots x_n} P(o_2|x_2) \ldots P(o_n|x_n) \cdot P(x_n|x_{n-1}) \sum_{x_1} P(x_1) \cdot P(o_1|x_1) P(x_2|x_1)$

$\alpha_2(x_2)$

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?

$P(X_1 | o_1 .. o_n)$ eliminate $x_2 \ldots x_n$

$P(X_2 | o_1 \ldots o_n)$ eliminate $x_1, x_3, \ldots x_n$

⋮

$P(X_n | o_1 \ldots o_n)$ eliminate $X_1 \ldots X_{n-1}$

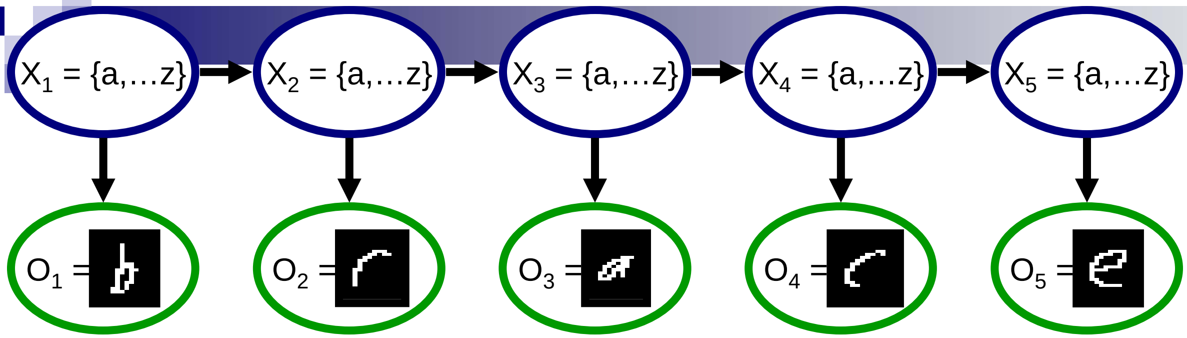
Variable elimination for each i, what's the complexity?

for one x_i, complexity linear in length n

for all x_i, complexity is n^2

by reusing factors, get all $P(X_i | o)$ at only twice the cost

Reusing computation



Compute:

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

eliminate x_1

$$P(x_3, o_1 .. o_n) = \sum_{x_2} \sum_{x_4} \sum_{x_5} P(o_2|x_2)\, P(x_3|x_2) \ldots P(o_5|x_5) \cdot P(x_5|x_4)$$

$$\underbrace{\sum_{x_1} P(x_1)\, P(o_1|x_1) \cdot P(x_2|x_1)}_{\alpha_2(x_2)}$$

eliminate x_5

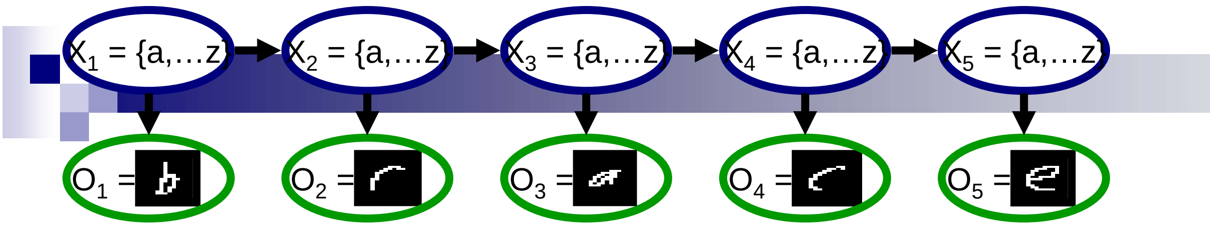
only difference

same

$$P(x_4, o_1 \ldots o_n) = \sum_{x_2} \sum_{x_3} \sum_{x_5} P(o_2|x_2)\, P(x_3|x_2) \ldots P(o_5|x_5)\, P(x_5|x_4)$$

$$\underbrace{\sum_{x_1} P(x_1)\, P(o_1|x_1)\, P(x_2|x_1)}_{\alpha_2(x_2)}$$

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 (n steps)
  - Generate a forwards factor by eliminating X_{i-1}

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

obs. transition

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

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

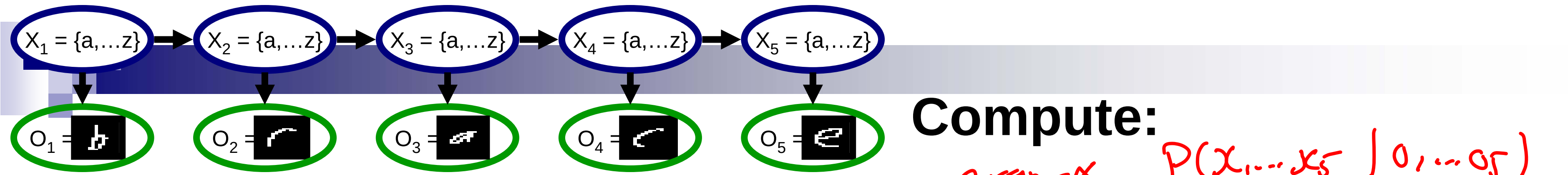
- ∀ i, probability is: $P(X_i \mid o_{1..n}) \propto \alpha_i(X_i)\beta_i(X_i)$

total is about 2n

Announcements

- HW4 out later today
- Recitation tomorrow

Most likely explanation



Compute:

$$\arg\max_{x_1,\ldots,x_5} P(x_1,\ldots,x_5 \mid o_1,\ldots,o_5)$$

Variable elimination order?

$X_1, X_2, \ldots, X_5$

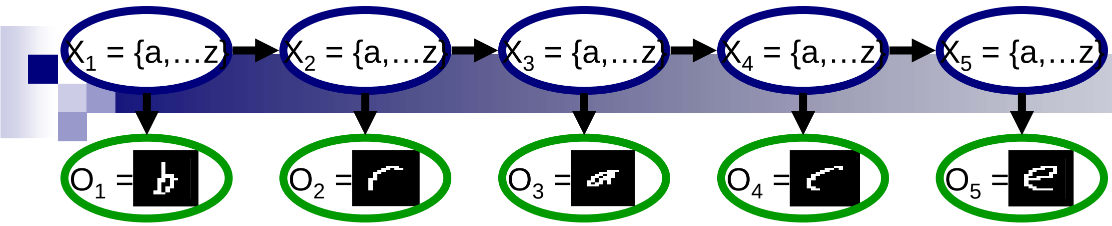
Example:

$$\max_{x_1,\ldots,x_5} \underbrace{P(x_1)P(o_1|x_1)P(x_2|x_1)}P(o_2|x_2)\ldots P(o_5|x_5)\,P(x_5|x_4)$$

$$= \max_{x_2,\ldots,x_5} P(o_2|x_2)\ldots P(o_5|x_5)\,P(x_5|x_4) \underbrace{\max_{x_1}\left[P(x_1)\,P(o_1|x_1)\,P(x_2|x_1)\right]}_{\lambda_2(x_2)}$$

The Viterbi algorithm

optimal argmax (not greedy)



- Initialization: $\alpha_1(X_1) = P(X_1)P(o_1 \mid X_1)$
- For i = 2 to n *max instead of sum*
  - 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: $x_n^* = \operatorname{argmax}_{x_n} \alpha_n(x_n)$ *who was α_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

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

multiply factors

$f_1(X_i)$ $f_2(X_i, X_{i-1})$ $f_3(X_{i-1})$

$g(X_{i-1} = a, X_i = b) = f_1 \cdot f_2 \cdot f_3$

$= f_1(X_i = b) \cdot f_2(X_i = a, X_{i-1} = a) \; f_3(X_{i-1} = a)$

What you'll implement 2: max & argmax

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

$g(X_{i-1}, X_i)$

a, b

$h(X_i) = \max_{x_{i-1}} g$

$= \max \big(g(X_i = b, X_{i-1} = a)$

$\vdots$

$g(X_i = b, X_{i-1} = z \big)$

Higher-order HMMs

Standard HMM : $\{X_1\, X_2\, O_1\, O_2\} \perp \{X_4 X_5\, O_4 O_5\} \mid X_3$

past $\perp$ future | present

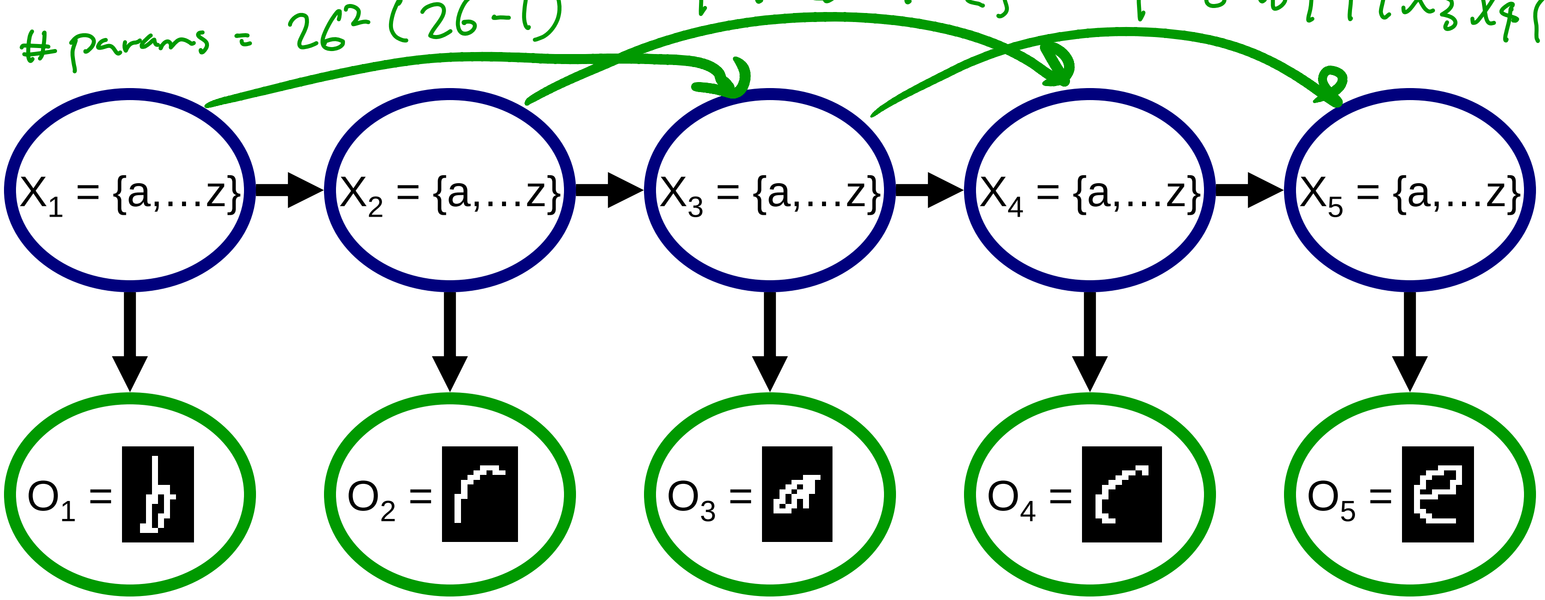
CPT $P(X_i \mid X_{i-1})$ = # params = $26(26-1)$

second-order HMM : $\{X_1\, X_2\, O_1\, O_2\} \perp \{X_4 X_5\, O_4 O_5\} \mid X_3$ NO!!

$\{X_1\, X_2\, O_1\, O_2\} \perp \{O_5 X_5\} \mid \{X_3\, X_4\}$

CPT $P(X_i \mid X_{i-1}, X_{i-2})$

params = $26^2(26-1)$



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

Bayesian Networks – (Structure) Learning

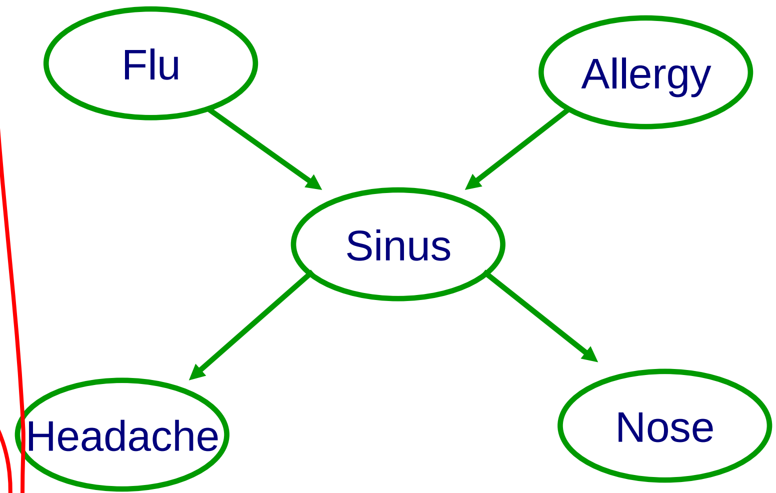
Machine Learning – 10701/15781

Carlos Guestrin
Carnegie Mellon University

March 28th, 2007

Review

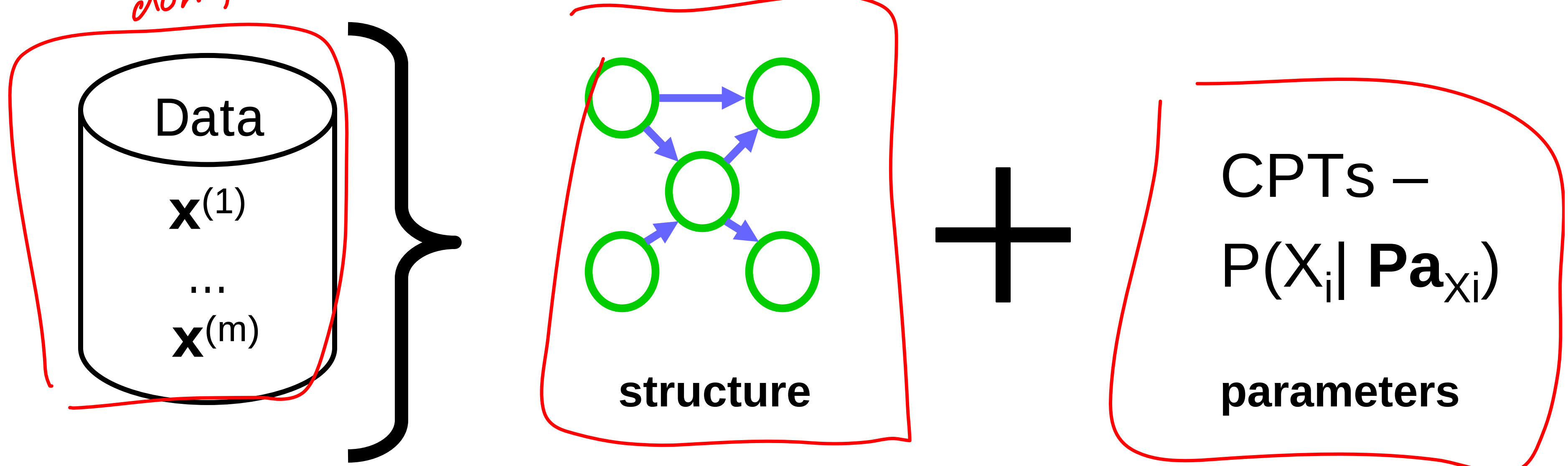
- Bayesian Networks
  - Compact representation for probability distributions
  - Exponential reduction in number of parameters
- Fast probabilistic inference using variable elimination
  - Compute P(X|e)
  - Time exponential in tree-width, not number of variables
- Today
  - Learn BN structure



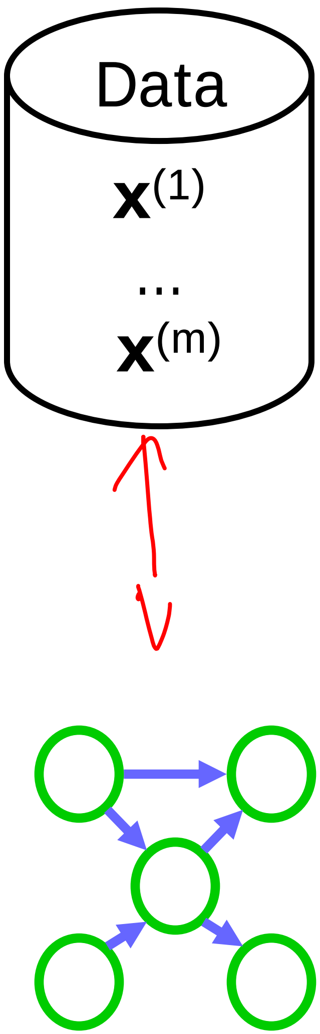
Learning Bayes nets

	Known structure	Unknown structure
Fully observable data ⟨A=t, H=f, S=t, F=t⟩	very easy!! ✓	learning "good" structure hard ... next week
Missing data	hard... talk about in two weeks	really really hard... we'll talk about it next semester

⟨A=?, H=f, S=t, F=?, N=t⟩ — ? don't know



Learning the CPTs



For each discrete variable X_i

want to learn $P(X_i \mid Pa_{X_i})$

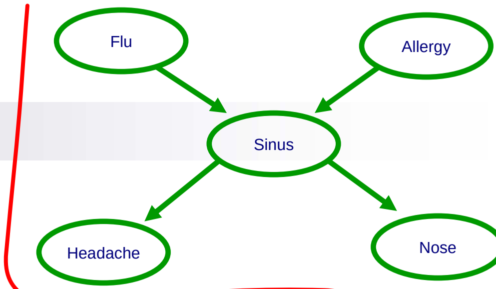
$$P(S=t \mid F=t, A=f) = \frac{\text{Count}(S=t, F=t, A=f)}{\text{Count}(F=t, A=f)}$$

Maximum likelihood estimates

set of parents

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

Information-theoretic interpretation of maximum likelihood



- Given structure, log likelihood of data:

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

data, params, structure

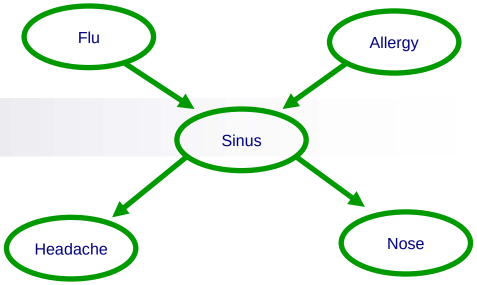
$|D| = m$

$$= \log \prod_{j=1}^{m} P(f^{(j)}, a^{(j)}, s^{(j)}, h^{(j)}, n^{(j)} \mid \theta_G, G)$$

what is G

$$= \log \prod_{j=1}^{m} P(f^{(j)} \mid \theta_F, G) \cdot P(a^{(j)} \mid \theta_A, G) \cdot P(s^{(j)} \mid f^{(j)}, a^{(j)}, \theta_{S|FA}, G) \cdot P(h^{(j)} \mid s^{(j)}, \theta_{H|S}, G) \, P(n^{(j)} \mid s^{(j)}, \theta_{N|S}, G)$$

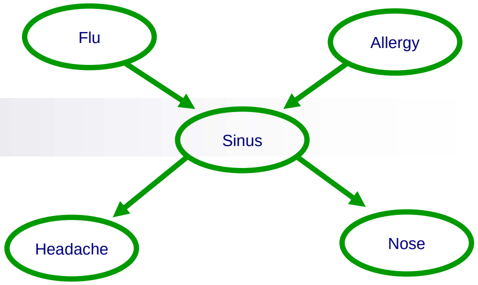
Information-theoretic interpretation of maximum likelihood



- Given structure, log likelihood of data:

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

Information-theoretic interpretation of maximum likelihood 2



- Given structure, log likelihood of data:

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

Decomposable score

- Log data likelihood

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

- Decomposable score:
  - Decomposes over families in BN (node and its parents)
  - Will lead to significant computational efficiency!!!
  - Score(*G : D*) = $\sum_i$ 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

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

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:

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

  - Compute mutual information:

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

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

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

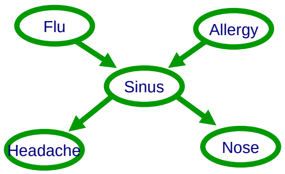
- (Approximately learning) models with tree-width up to *k*
  - [Narasimhan & Bilmes '04]
  - But, $O(n^{k+1})$…
    - and more subtleties

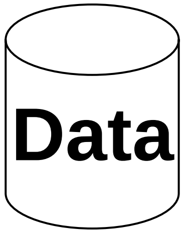
What you need to know about learning BN structures so far

- Decomposable scores
  - Maximum likelihood
  - Information theoretic interpretation
- Best tree (Chow-Liu)
- Best TAN
- Nearly best k-treewidth (in $O(N^{k+1})$)

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!

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

- 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