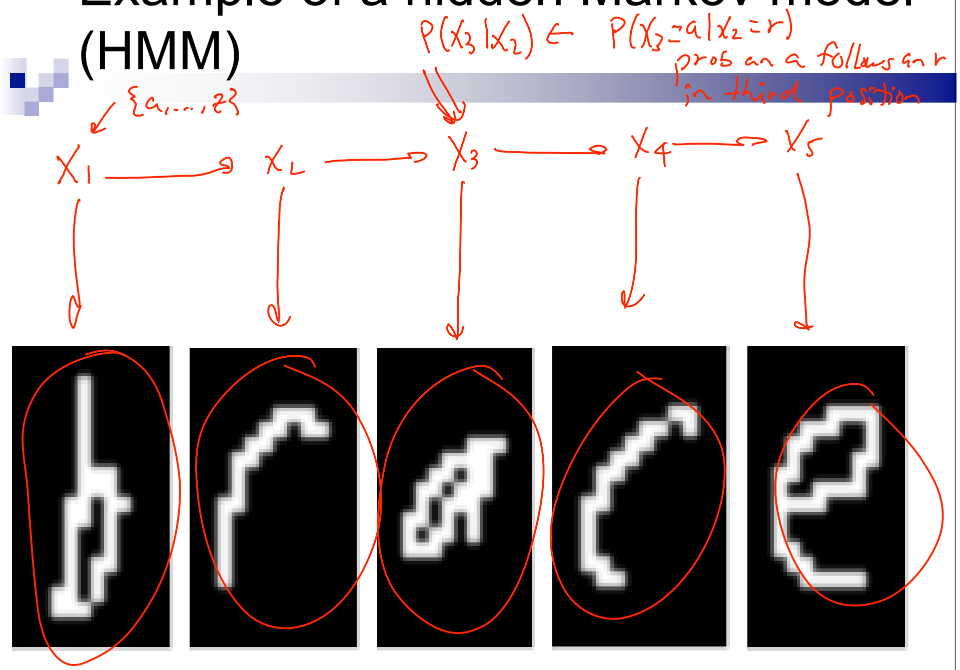
HMMs

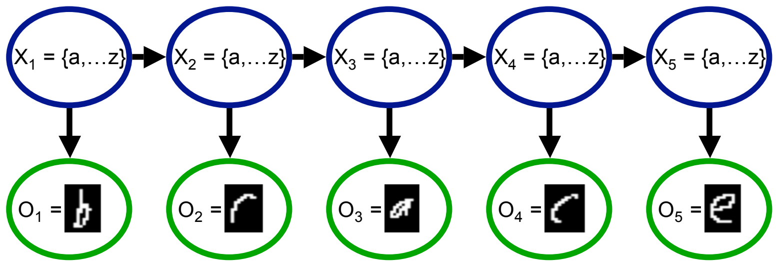
Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

November 7th, 2007

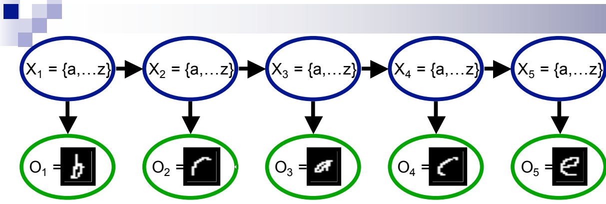
Example of a hidden Markov model (HMM)



Understanding the HMM Semantics



HMMs semantics: Details



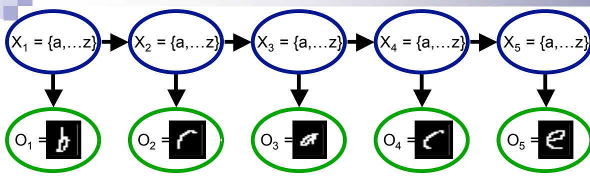
Just 3 distributions:

$$P(X_1)$$

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

$$P(O_i \mid X_i)$$

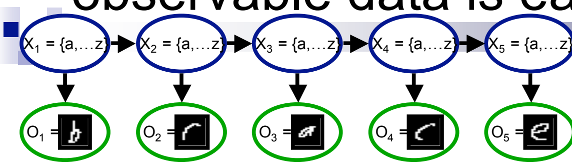
HMMs semantics: Joint distribution



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

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

Learning HMMs from fully observable data is easy



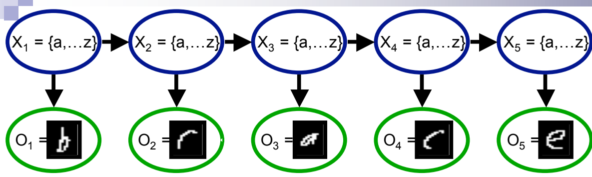
Learn 3 distributions:

$P(X_1)$

$P(O_i \mid X_i)$

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

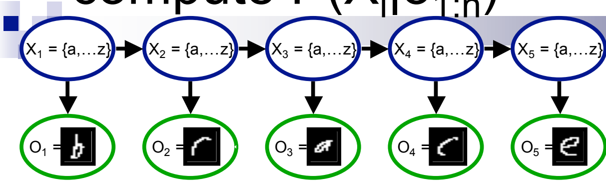
Possible inference tasks in an HMM



Marginal probability of a hidden variable:

Viterbi decoding – most likely trajectory for hidden vars:

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



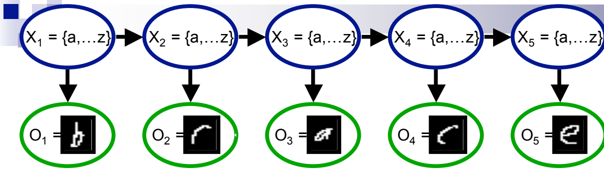
Compute:

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

Variable elimination order?

Example:

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



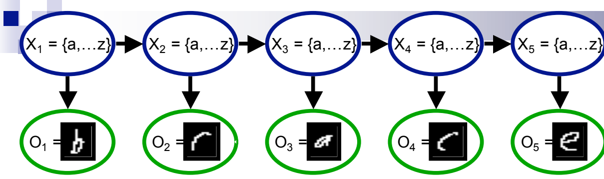
Compute:

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

Variable elimination for each i?

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

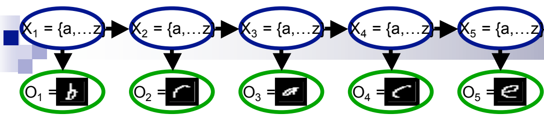
Reusing computation



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}

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

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

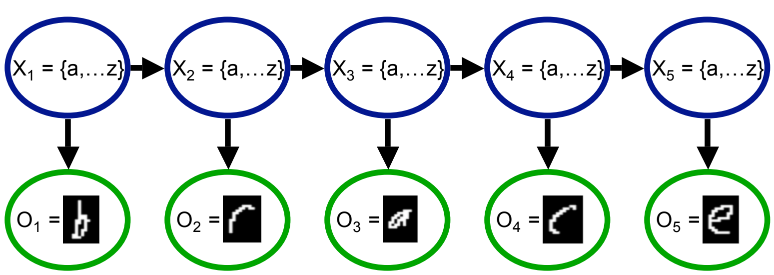
What you'll implement 1: multiplication

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

What you'll implement 2: marginalization

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

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

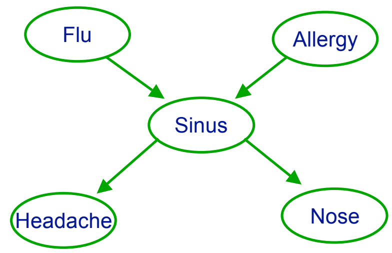
Bayesian Networks (Structure) Learning

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

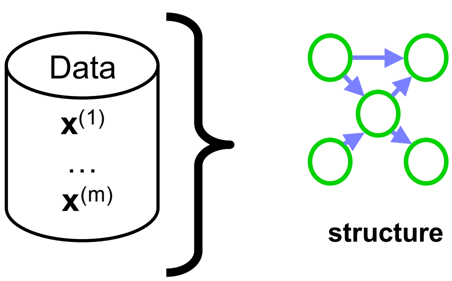
November 7^{th}, 2007

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

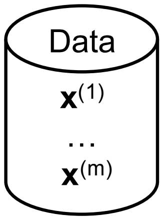


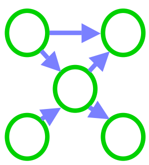
+

CPTs –
$P(X_i| \mathbf{Pa}_{X_i})$

parameters

Learning the CPTs

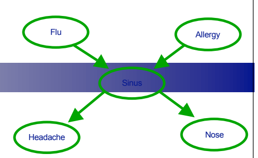




For each discrete variable X_i

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

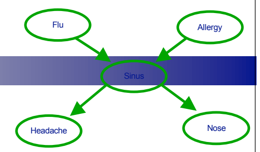
Information-theoretic interpretation of maximum likelihood 1



- Given structure, log likelihood of data:

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

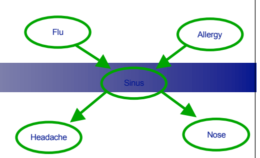
Information-theoretic interpretation of maximum likelihood 2



- 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 3



- Given structure, log likelihood of data:

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

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:

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

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

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

Chow-Liu tree learning algorithm 2

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

- 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

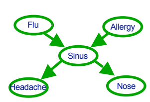
- (Approximately learning) models with tree-width up to *k*
  - [Chechetka & Guestrin '07]
  - But, $O(n^{2k+6})$…

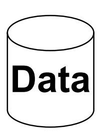
What you need to know about learning BN structures so far

- 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?





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

…

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

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

Maximum likelihood overfits!

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

- 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

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