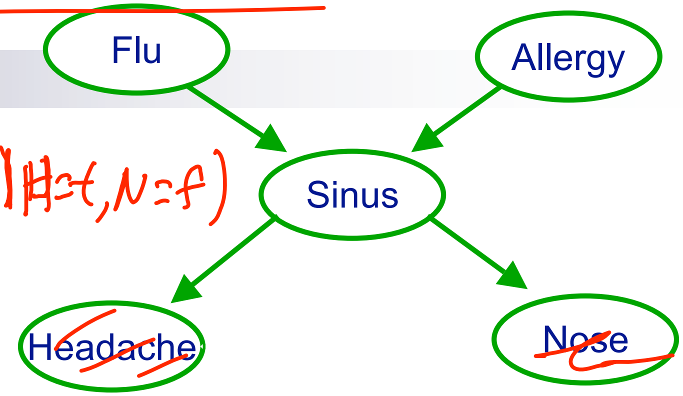
Bayesian Networks – Inference (cont.)

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

March 26th, 2007

General probabilistic inference



- Query: $P(X \mid e)$

$P(F=t \mid H=t, N=f)$

Defn. cond. probs.

- Using ~~Bayes rule~~:

$$P(X \mid e) = \frac{P(X,e)}{P(e)}$$

constant doesn't depend on X

$P(F, H=t, N=f)$

F	
t	.3
f	.2

not normalized

normalize

- Normalization:

$$P(X \mid e) \propto \boxed{P(X,e)}$$

normalize

Compute to give answer

$P(F \mid H=t, N=f)$

t	.6
f	.4

Marginalization

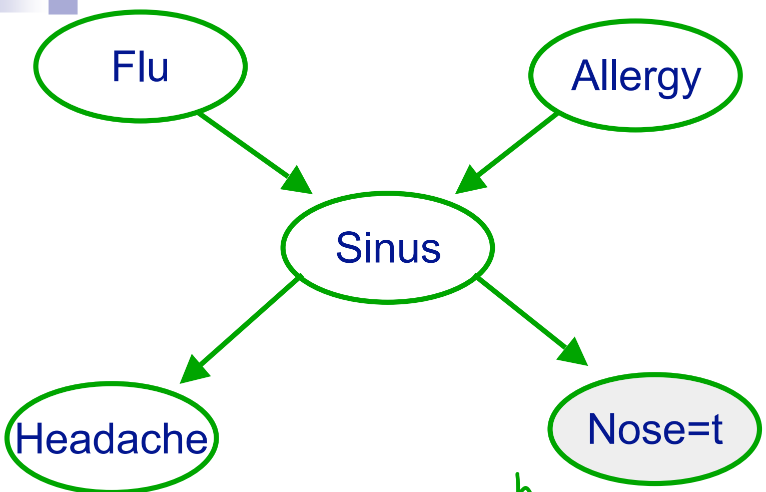
$$P(F, S, N) = P(F) \cdot P(S|F) \cdot P(N|S)$$

$$P(F=t, N=t) = P(F=t, S=t, N=t) + P(F=t, S=f, N=t)$$

$$= P(F=t) \cdot P(S=t|F=t) \cdot P(N=t|S=t) + P(F=t) \cdot P(S=f|F=t) \, P(N=t|S=f$$

marginalize out S

Probabilistic inference example



$P(F, N=t)$ ← want

know

$P(F, A, S, H, N=t) = P(F) \cdot P(A) \cdot P(S|FA) \cdot P(H|S)\ P(N=t|S)$

per value of F
7 sums
$8 \times 4 = 32$ multiplies

grand total
14 sums
64 multiply

how many terms adding? 2^3

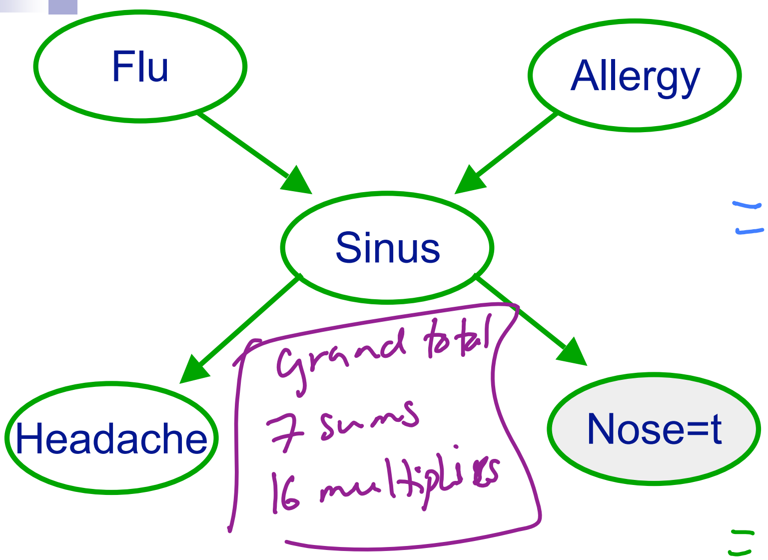
$$P(F, N=t) = \underbrace{\sum_a \sum_s \sum_h}_{\text{how many terms adding? } 2^3} P(F, A=a, S=s, H=h, N=t)$$

$$= \sum_a \sum_s \sum_h P(F) \cdot P(a) \cdot P(s|F,a) \cdot P(h|s) \cdot P(N=t|s)$$

Inference seems exponential in number of variables!
Actually, inference in graphical models is NP-hard in general **☹**

Fast probabilistic inference example – Variable elimination

eliminate (marginalizing) vars one at a time



$$P(F, N=t) = \sum_a \sum_s \sum_h P(F) \cdot P(a) \cdot P(s|F,a) \cdot P(h|s) \cdot P(N=t|s)$$

$$= \sum_a \sum_s P(F) \cdot P(a) \cdot P(s|F,a) \cdot P(N=t|s) \cdot \cancel{\sum_h P(h|s)}\ 1$$

special case

1 sum
0 multiplies

$$= \sum_a \sum_s P(F) \cdot P(a)\, P(s|F,a)\, P(N=t|s) \cdot 1$$

$$= \sum_a P(F)\, P(a) \underbrace{\sum_s P(s|F,a) \cdot P(N=t|s)}_{g_1(F,a)}$$

For each assignment of F & A: 1 sum, 2 multiplies

total: 4 sums, 8 multiplies

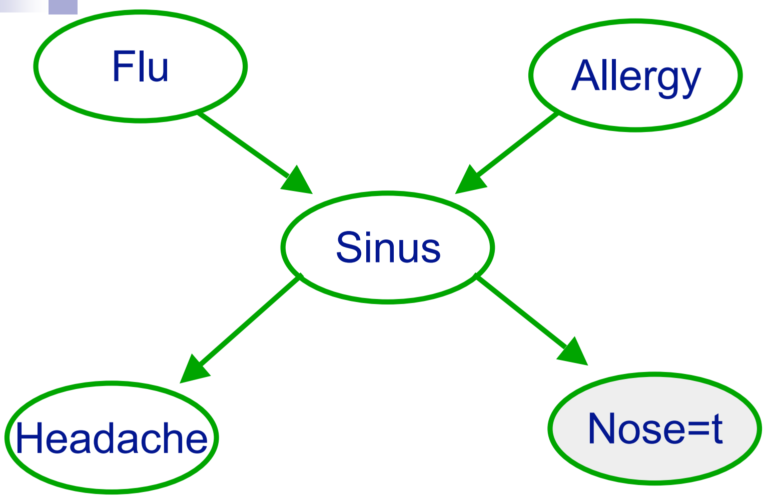
$$= \underbrace{\sum_a P(F) \cdot P(a) \cdot g_1(F,a)}_{\text{for each assignment of F: 1 sum, 4 multiplies}} = P(F, N=t)$$

total: 2 sums, 8 multiplies

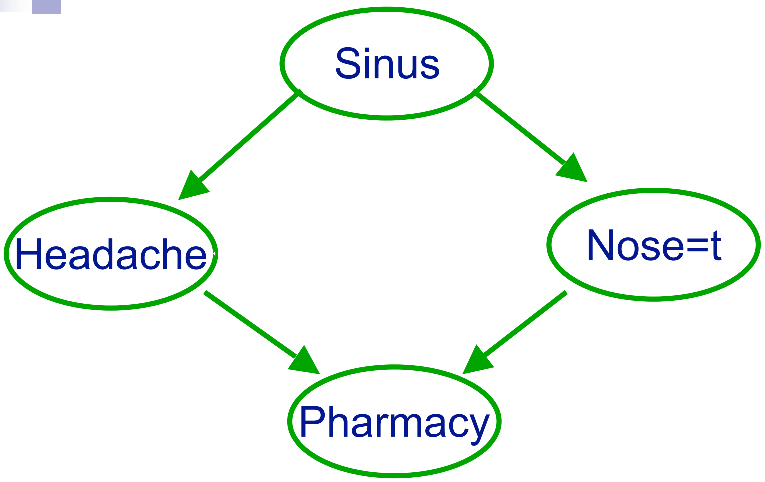
(Potential for) Exponential reduction in computation!

Understanding variable elimination – Exploiting distributivity

Understanding variable elimination – Order can make a HUGE difference



Understanding variable elimination – Another example



Variable elimination algorithm

- Given a BN and a query $P(X|e) \propto P(X,e)$
- Instantiate evidence e

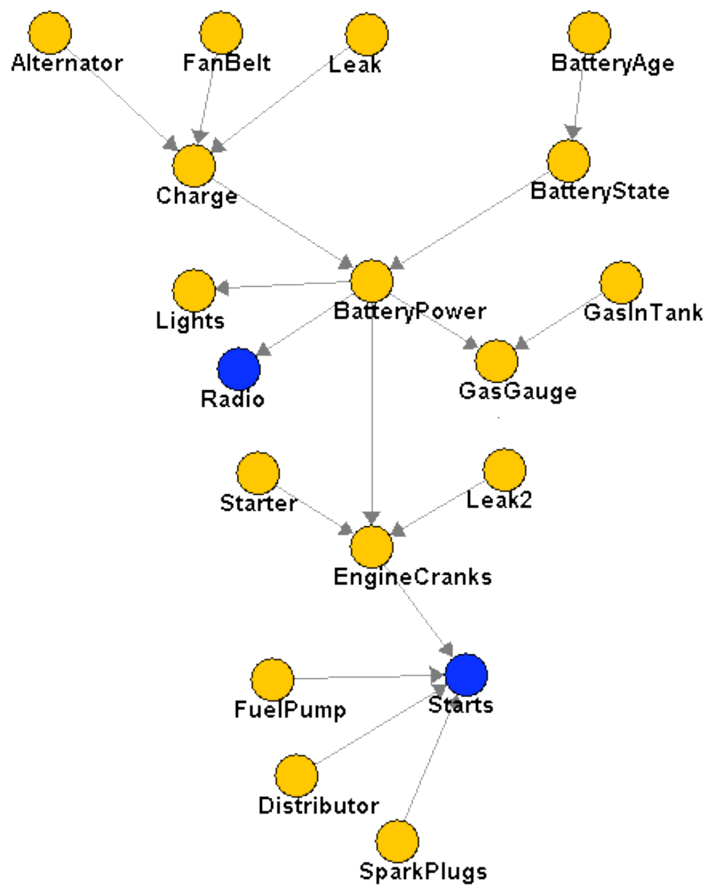
IMPORTANT!!!

- Choose an ordering on variables, e.g., $X_1, \ldots, X_n$
- For i = 1 to n, If $X_i \notin \{X,e\}$
  - Collect factors $f_1,\ldots,f_k$ that include X_i
  - Generate a new factor by eliminating X_i from these factors

$$g = \sum_{X_i} \prod_{j=1}^{k} f_j$$

  - Variable X_i has been eliminated!
- Normalize $P(X,e)$ to obtain $P(X|e)$

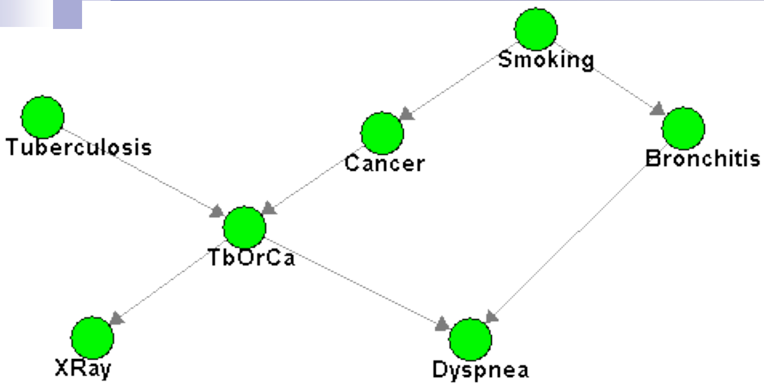
Complexity of variable elimination – (Poly)-tree graphs



Variable elimination order:
Start from "leaves" up –
find topological order, eliminate variables in reverse order

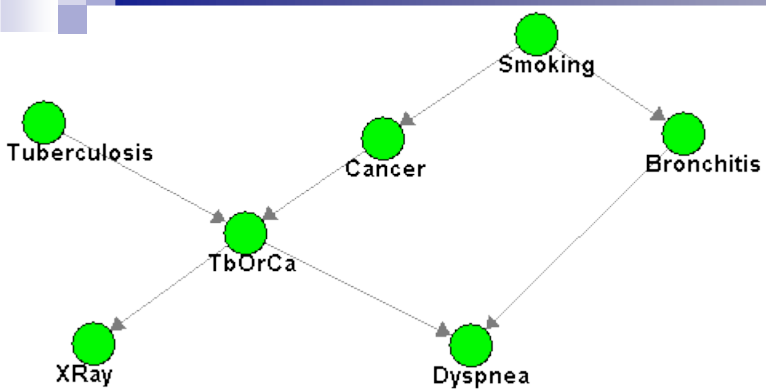
Linear in number of variables!!! (versus exponential)

Complexity of variable elimination – Graphs with loops



Exponential in number of variables in largest factor generated

Complexity of variable elimination –Tree-width



Moralize graph:
Connect parents into a clique and remove edge directions

Complexity of VE elimination:
("Only") exponential in tree-width
Tree-width is maximum node cut +1

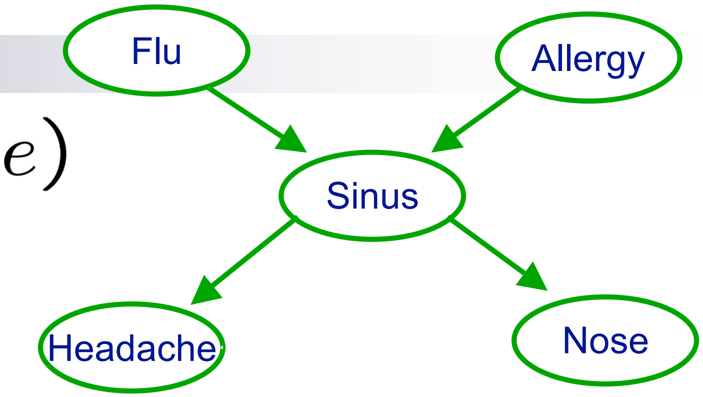
Example: Large tree-width with small number of parents

Compact representation $\not\Rightarrow$ Easy inference ☹

Choosing an elimination order

- Choosing best order is NP-complete
  - Reduction from MAX-Clique
- Many good heuristics (some with guarantees)
- Ultimately, can't beat NP-hardness of inference
  - Even optimal order can lead to exponential variable elimination computation
- In practice
  - Variable elimination often very effective
  - Many (many many) approximate inference approaches available when variable elimination too expensive

Most likely explanation (MLE)



- Query: $\underset{x_1,\ldots,x_n}{\text{argmax}}\, P(x_1,\ldots,x_n \mid e)$

- Using Bayes rule:

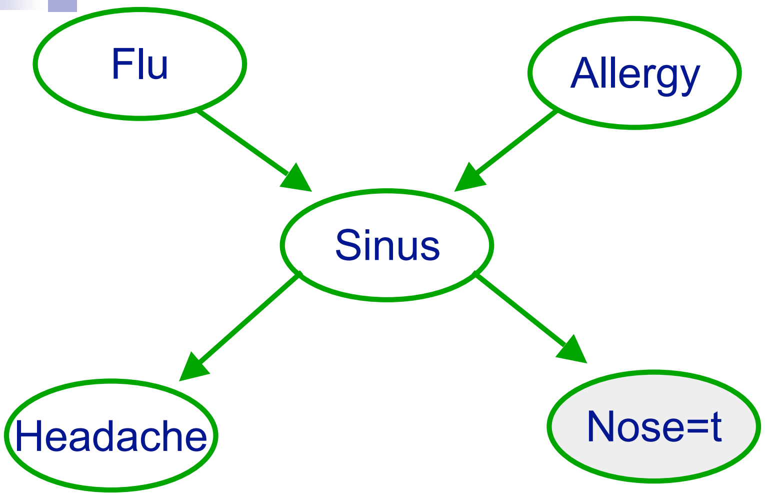
$$\underset{x_1,\ldots,x_n}{\text{argmax}}\, P(x_1,\ldots,x_n \mid e) = \underset{x_1,\ldots,x_n}{\text{argmax}}\, \frac{P(x_1,\ldots,x_n,e)}{P(e)}$$

- Normalization irrelevant:

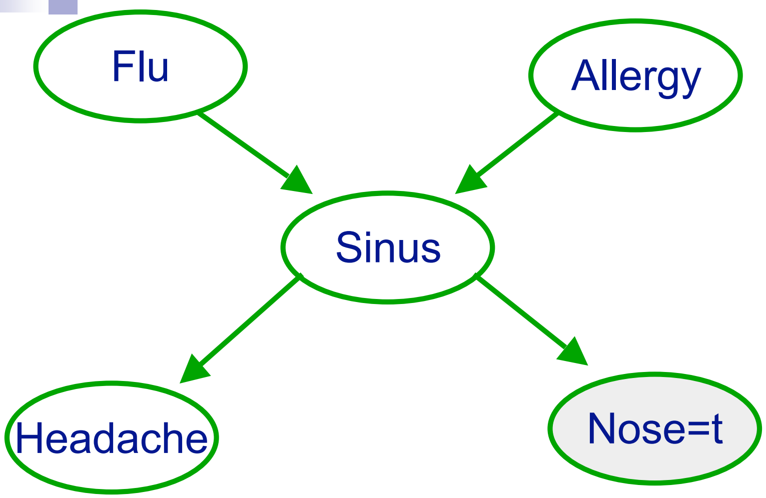
$$\underset{x_1,\ldots,x_n}{\text{argmax}}\, P(x_1,\ldots,x_n \mid e) = \underset{x_1,\ldots,x_n}{\text{argmax}}\, P(x_1,\ldots,x_n,e)$$

Max-marginalization

Example of variable elimination for MLE – Forward pass



Example of variable elimination for MLE – Backward pass



MLE Variable elimination algorithm – Forward pass

- Given a BN and a MLE query $\max_{x_1,\ldots,x_n} P(x_1,\ldots,x_n,e)$
- Instantiate evidence e
- Choose an ordering on variables, e.g., $X_1, \ldots, X_n$
- For i = 1 to n, If $X_i \notin \{e\}$
  - Collect factors $f_1,\ldots,f_k$ that include X_i
  - Generate a new factor by eliminating X_i from these factors

$$g = \max_{x_i} \prod_{j=1}^{k} f_j$$

  - Variable X_i has been eliminated!

MLE Variable elimination algorithm – Backward pass

- $\{x_1^*, \ldots, x_n^*\}$ will store maximizing assignment
- For i = n to 1, If $X_i \notin \{e\}$
  - Take factors $f_1, \ldots, f_k$ used when X_i was eliminated
  - Instantiate $f_1, \ldots, f_k$, with $\{x_{i+1}^*, \ldots, x_n^*\}$
    - Now each f_j depends only on X_i
  - Generate maximizing assignment for X_i:

$$x_i^* \in \operatorname*{argmax}_{x_i} \prod_{j=1}^{k} f_j$$

What you need to know

- Bayesian networks
  - A useful compact **representation** for large probability distributions
- Inference to compute
  - Probability of X given evidence e
  - Most likely explanation (MLE) given evidence e
  - Inference is NP-hard
- Variable elimination algorithm
  - Efficient algorithm ("only" exponential in tree-width, not number of variables)
  - Elimination order is important!
  - Approximate inference necessary when tree-width to large
    - not covered this semester
  - Only difference between probabilistic inference and MLE is "sum" versus "max"

HMMs

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

March 26th, 2007

Adventures of our BN hero

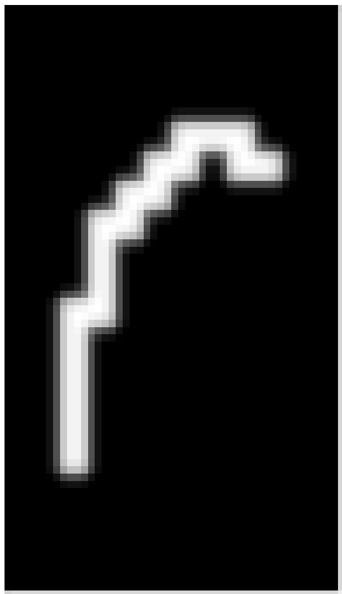
- Compact representation for probability distributions
- Fast inference
- Fast learning

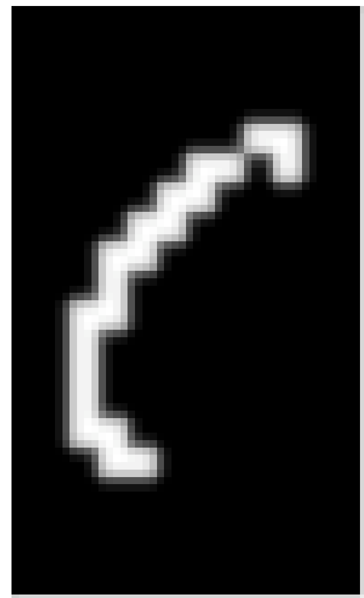
- But… Who are the most popular kids?

1. Naïve Bayes

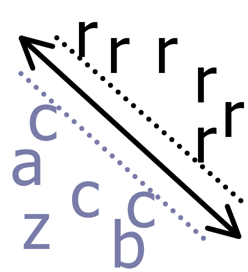
2 and 3.
Hidden Markov models (HMMs)
Kalman Filters

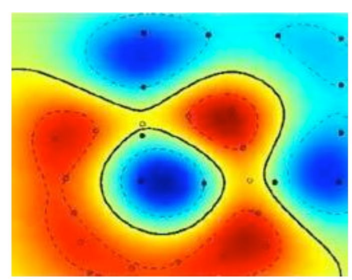
Handwriting recognition



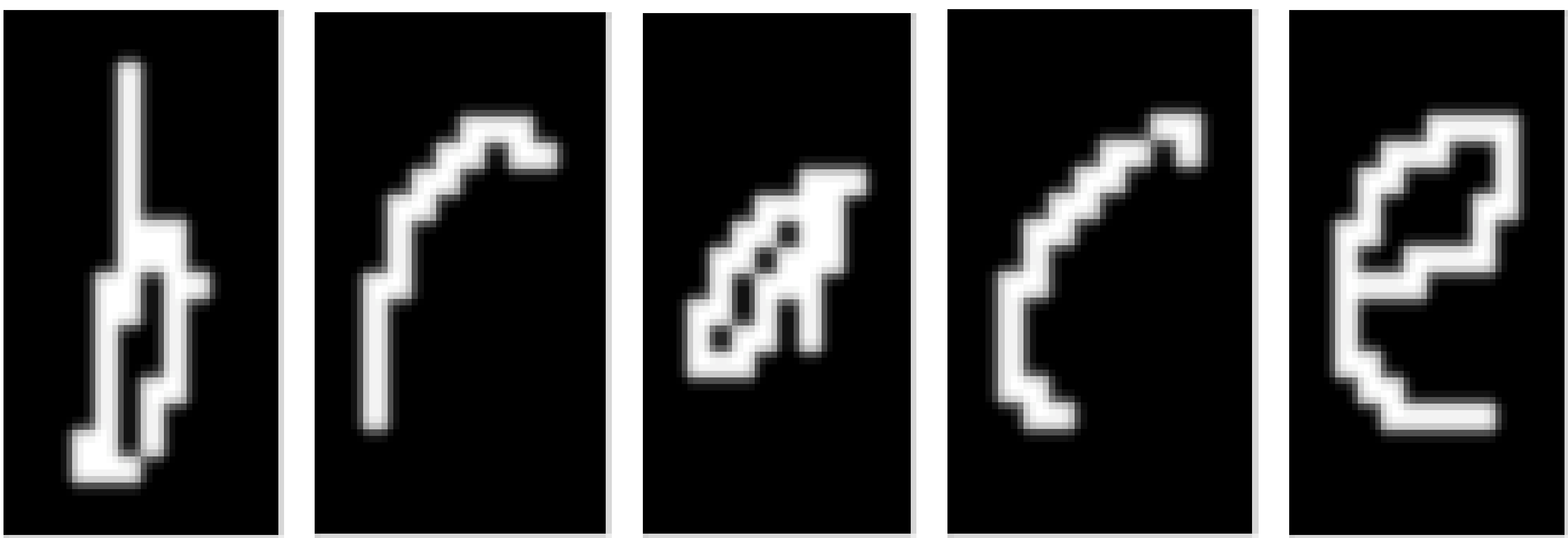


Character recognition, e.g., kernel SVMs

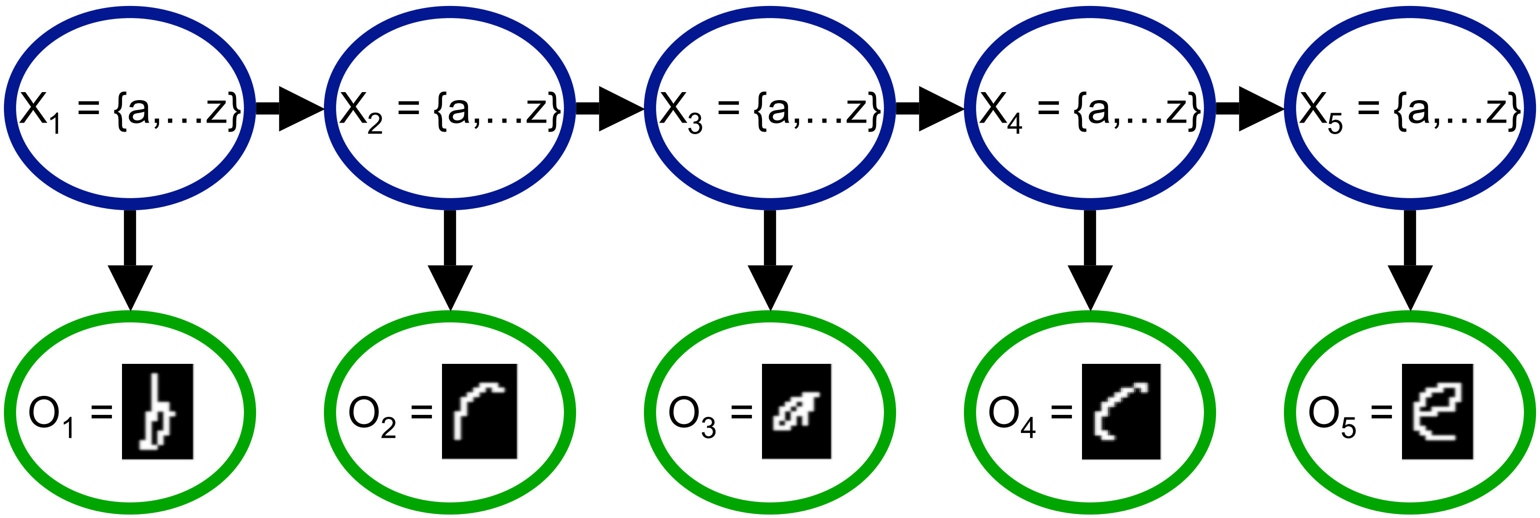




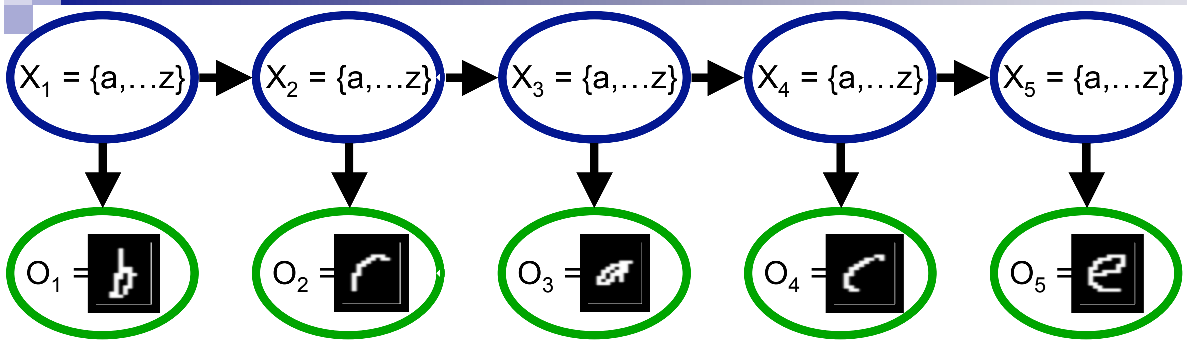
Example of a hidden Markov model (HMM)



Understanding the HMM Semantics



HMMs semantics: Details



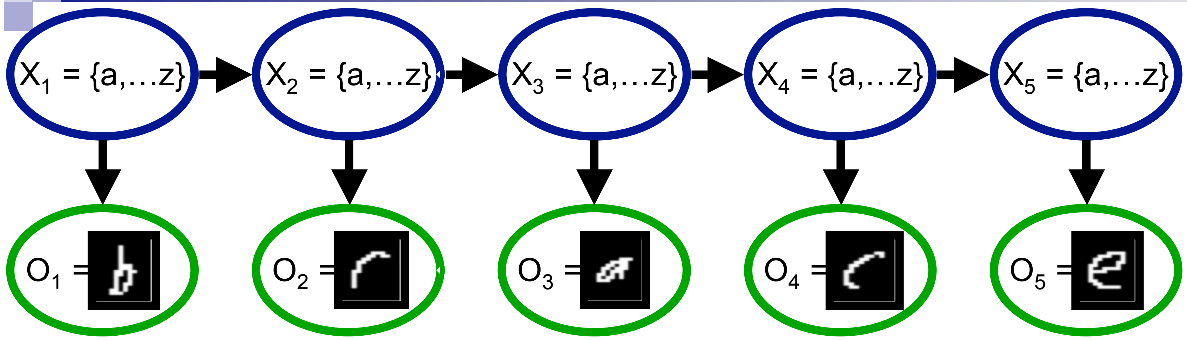
Just 3 distributions:

$$P(X_1)$$

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

$$P(O_i \mid X_i)$$

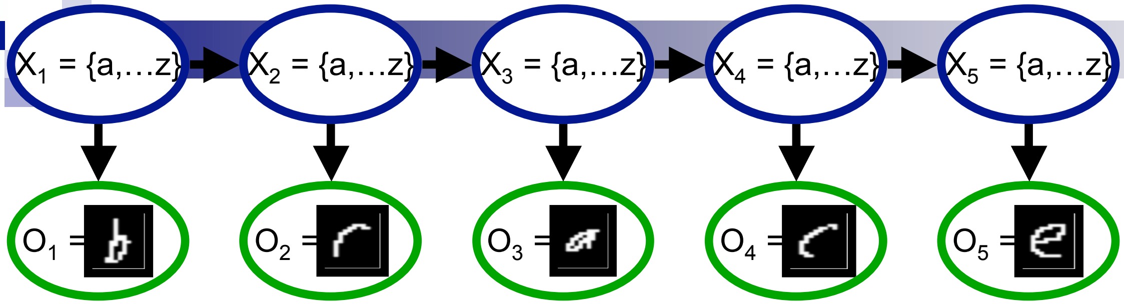
HMMs semantics: Joint distribution



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

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

Learning HMMs from fully observable data is easy



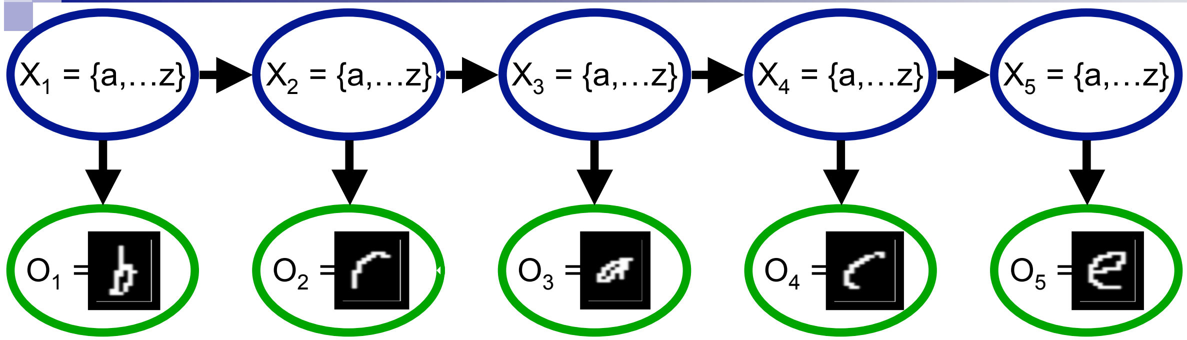
Learn 3 distributions:

$$P(X_1)$$

$$P(O_i \mid X_i)$$

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

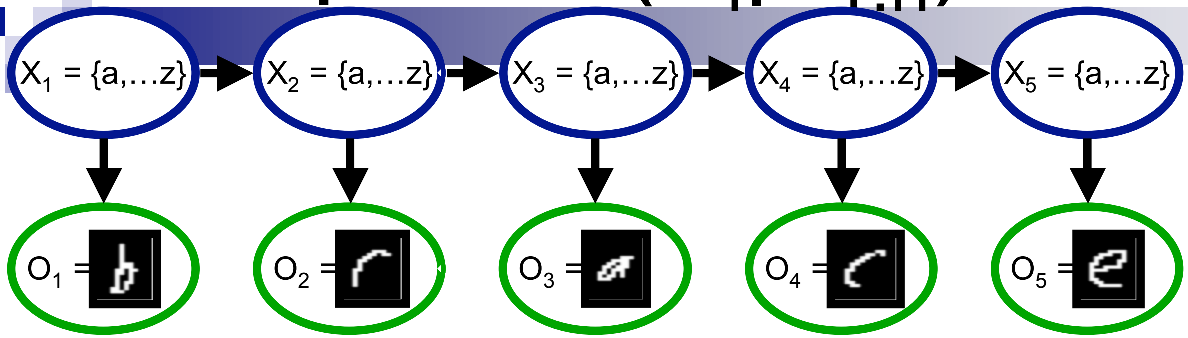
Possible inference tasks in an HMM



Marginal probability of a hidden variable:

Viterbi decoding – most likely trajectory for hidden vars:

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



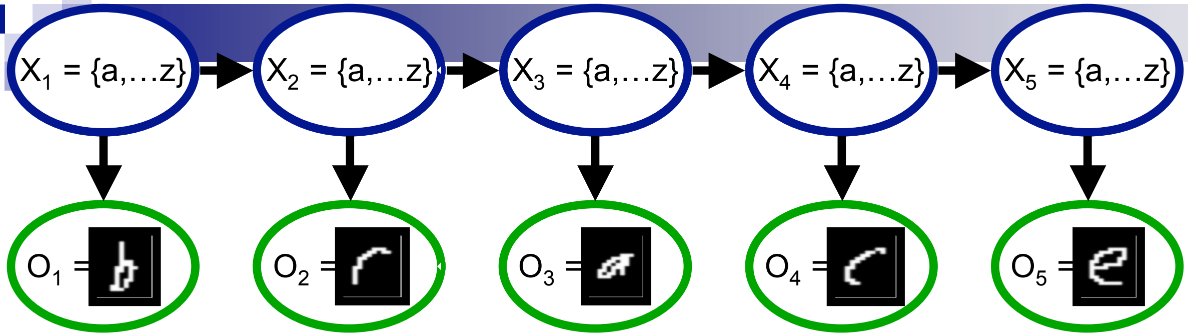
Compute:

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

Variable elimination order?

Example:

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



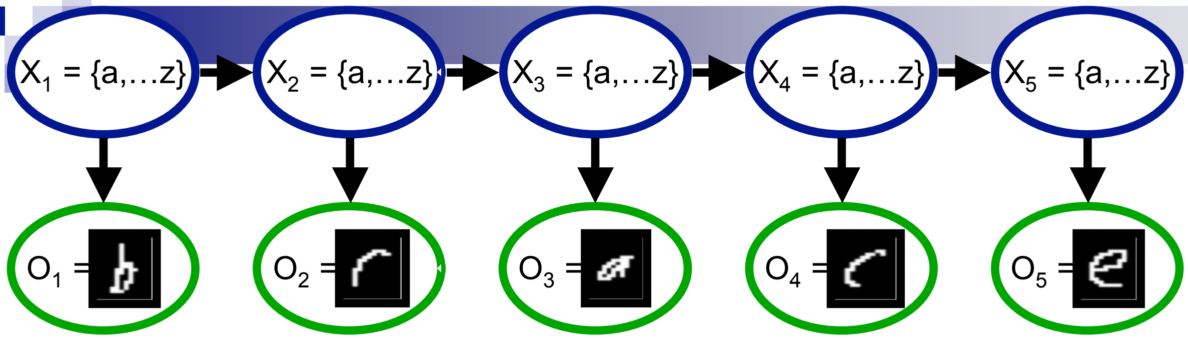
Compute:

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

Variable elimination for each i?

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

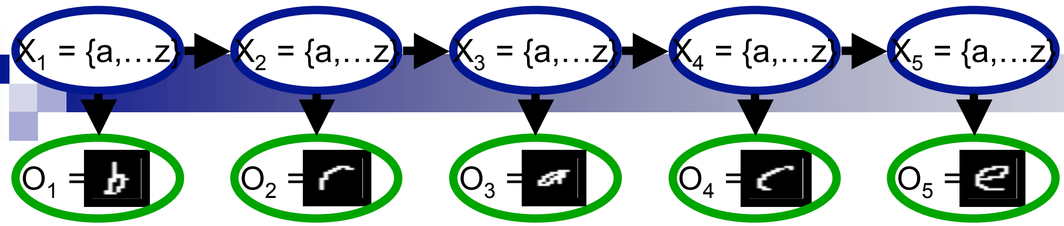
Reusing computation



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}

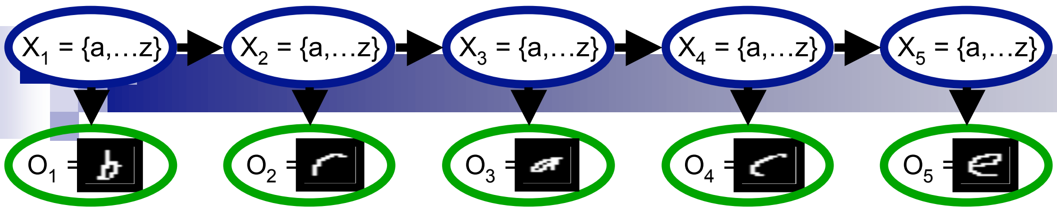
$$\alpha_i(X_i) = \sum_{x_{i-1}} P(o_i \mid X_i)P(X_i \mid X_{i-1} = x_{i-1})\alpha_{i-1}(x_{i-1})$$

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

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

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

Most likely explanation

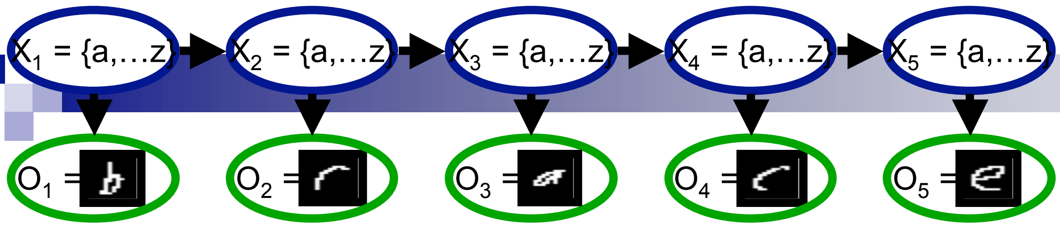


Compute:

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:

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

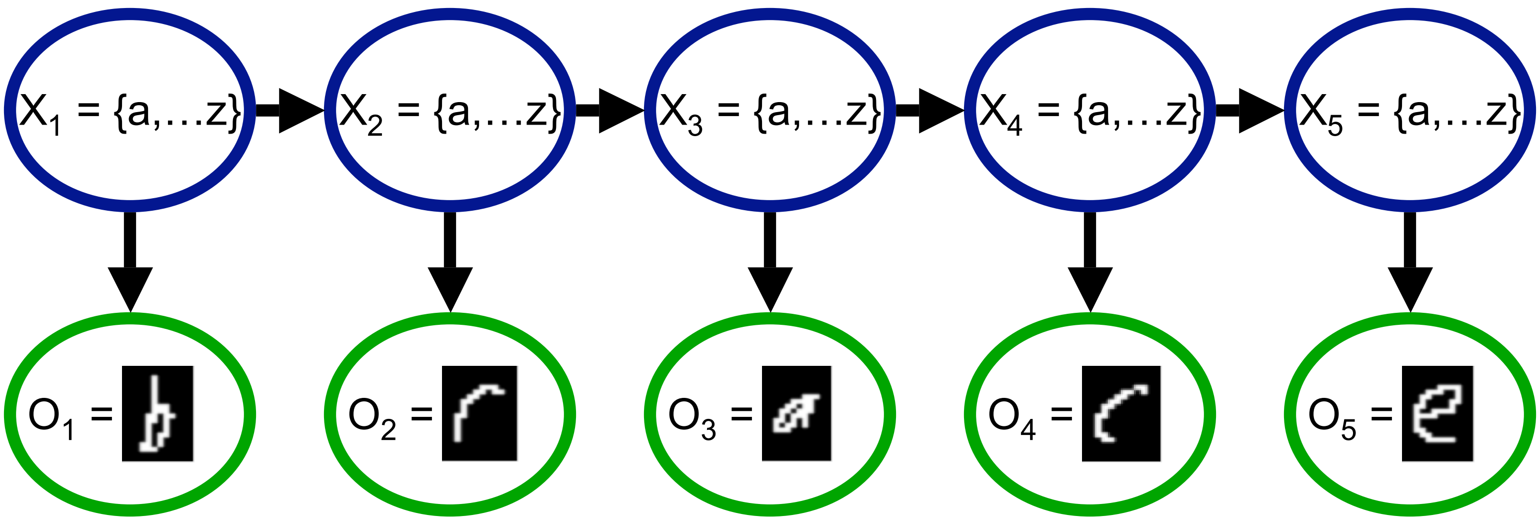
What you'll implement 1: multiplication

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

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