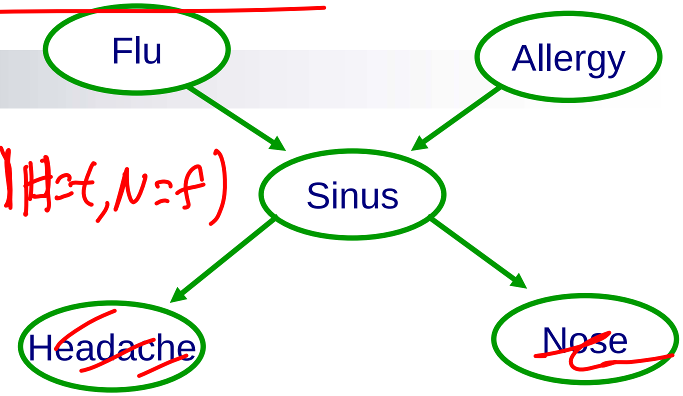
Bayesian Networks – Inference (cont.)

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

March 26th, 2007

General probabilistic inference



- Query: $\boxed{P(X \mid e)}$

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

Defn. cond. probs.

- Using ~~Bayes rule~~:

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

constant doesn't depend on X

- Normalization:

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

normalize

Compute to give answer

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

F	
t	.3
f	.2

not normalize

normalize

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

t	.6
f	.4

Marginalization

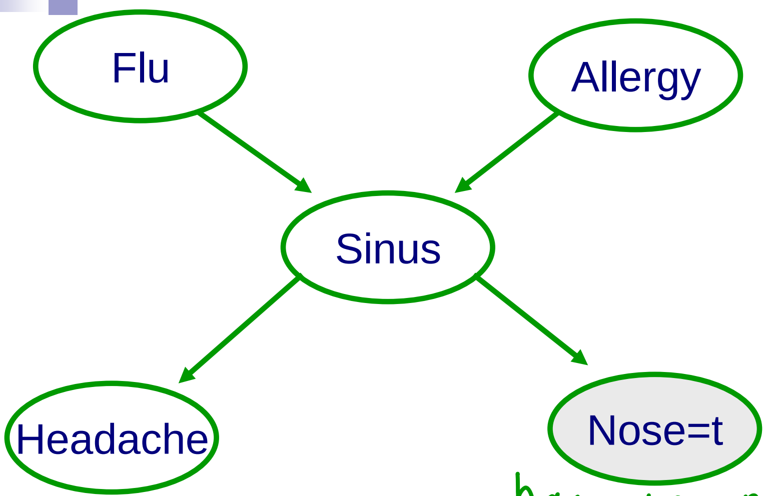
$$P(F,S,N) = P(F)\cdot P(S|F)\cdot P(N|S)$$

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

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

marginalize out S

Probabilistic inference example



$P(F, N=t)$ ← want

know

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

per value of F:
7 sums
8×4=32 multiplies

grand total
14 sums
64 multiply

how many terms adding? 2^3

$$P(F, N=t) = \sum_a \sum_s \sum_h P(F, A=a, S=s, H=h, N=t)$$

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

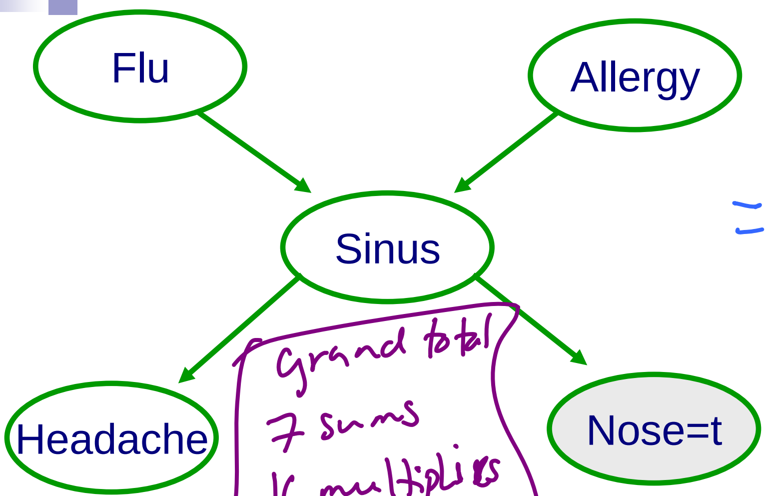
Inference seems exponential in number of variables!

in general

Actually, inference in graphical models is NP-hard ☹

Fast probabilistic inference example – Variable elimination

eliminate (marginalize) vars one at a time



grand total
7 sums
16 multiplies

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

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

$\sum_h P(h|s) = 1$ special case

1 sum
0 multiplies

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

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

For each assignment of F & A
↳ 1 sum
2 multiplies

total
4 sums
8 multiplies

$$= \sum_a P(F) . P(a) . g_1(F,a) = P(F, N=t)$$

for each assignment of F: 1 sum, 4 multiplies

total
2 sums
8 multiplies

(Potential for) Exponential reduction in computation!

Understanding variable elimination – Exploiting distributivity

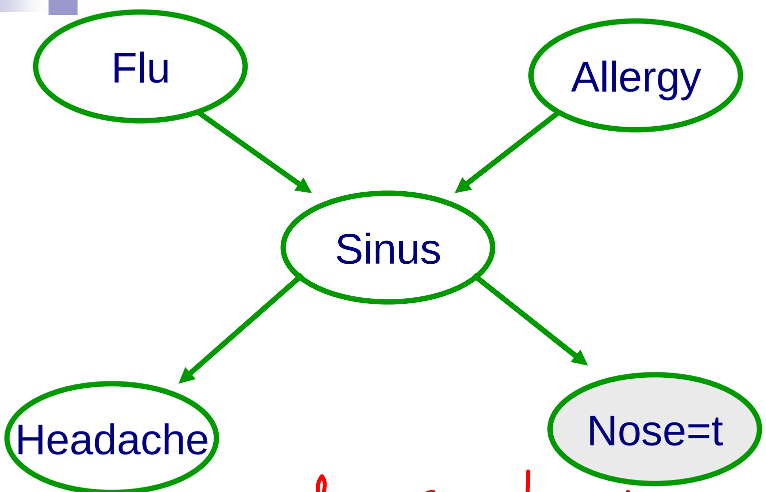
$a(b+c) = ab + ac$

$$\text{Flu} \rightarrow \text{Sinus} \rightarrow \text{Nose=t}$$

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

$$= P(F) \underbrace{\left(P(S=t \mid F) \cdot P(N=t \mid S=t) + P(S=f \mid F) \cdot P(N=t \mid S=f) \right)}_{g_1(F)}$$

Understanding variable elimination – Order can make a HUGE difference

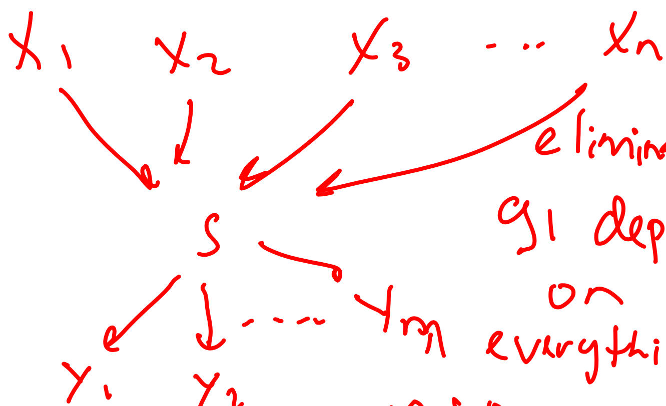


$$P(F, N=t) =$$

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

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

increased g_1 by 1,



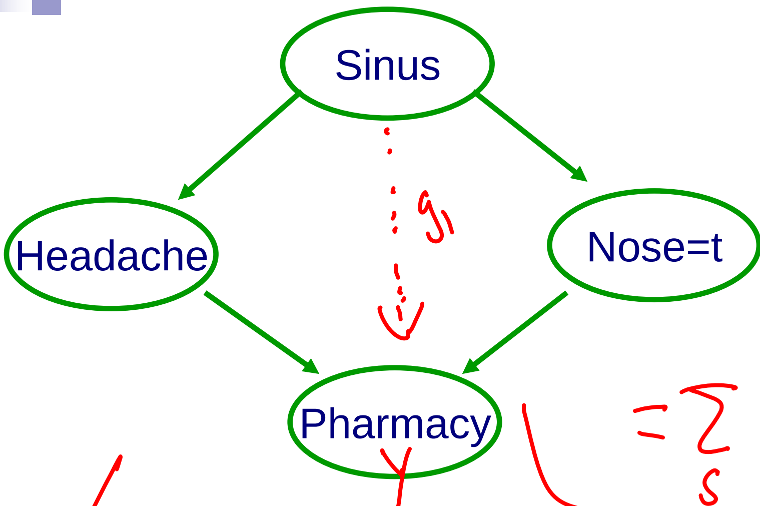
eliminates S

g_1 depends on everything

$m+n$ vars

exponentially larger.

Understanding variable elimination – Another example



$P(Y \mid N=t)$

$$P(Y, N=t) = \sum_s \sum_h P(s) \cdot P(h \mid s) \cdot P(N=t \mid s) \cdot P(Y \mid h, N=t)$$

$$= \sum_s P(S)\, P(N=t \mid S) \underbrace{\sum_h P(h \mid s) \cdot P(Y \mid h, N=t)}_{}$$

Y, S never appear together in a factor (in a CPT)

$g_1(S, Y)$ ← S & Y appear together

$g_1(S, Y, N=t)$

after eliminate H, distribution over S, N, Y

$P(S,Y,N=t) = P(S) \cdot P(N=t \mid S)\; g_1(S, Y, N=t)$

V.E. generates distributions over smaller sets of variables

Variable elimination algorithm

$\{A, F\}$ $e \leftarrow \{H=t, S=f, \ldots\}$

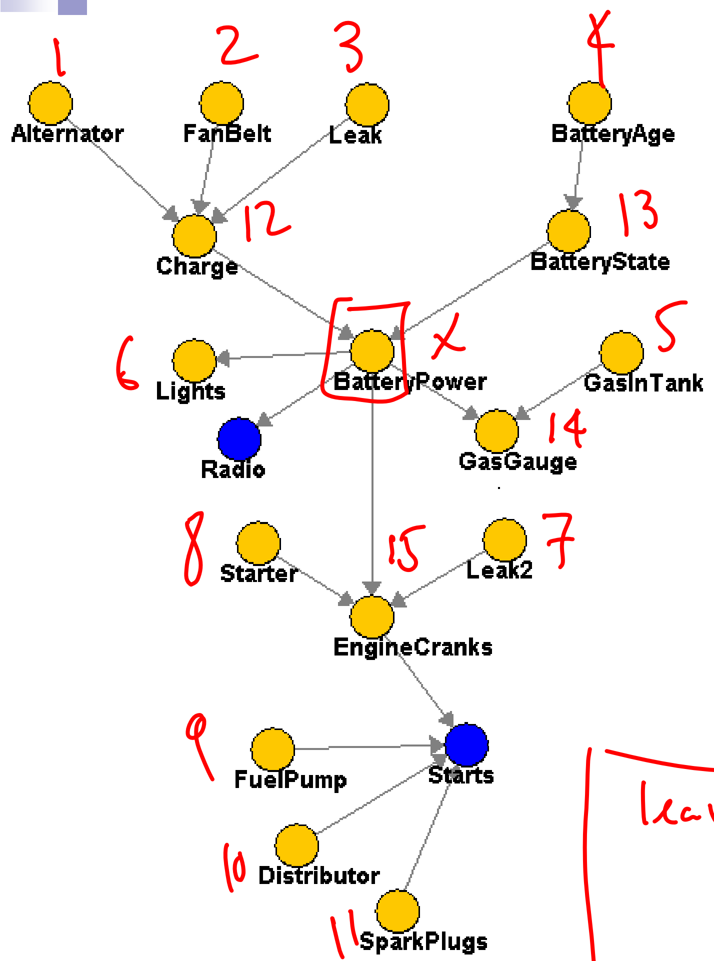
- Given a BN and a query $P(X|e) \propto P(X,e)$
- Instantiate evidence e ← H=t, S=f, ... **IMPORTANT!!!**
- Choose an ordering on variables, e.g., $X_1, \ldots, X_n$
- For i = 1 to n, If $X_i \notin \{X,e\}$
  - Collect factors $f_1,\ldots,f_k$ that include X_i
  - Generate a new factor by eliminating X_i from these factors

marginalize X_i ← multiply everything that depends on X_i

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

  - 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

complexity not using VE
about $2^{15} \cdot c$

with VE
about $15 \cdot c$

leaves can be eliminated in any order.

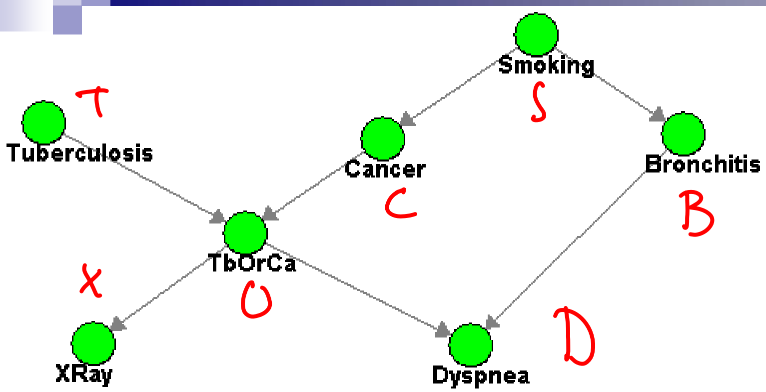
Linear in number of variables!!! (versus exponential)

Complexity of variable elimination – Graphs with loops

$(x_1 \vee \neg x_2 \vee x_7) \wedge (x_3 \vee \neg x_7 \vee x_{12}) \ldots$

$x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_{12} \quad x_n$

$C_1 \quad C_2 \quad --- \quad C_m$



$$\sum_B P(T)\, P(S) \,.\, P(B|S) \,.\, P(C|S) \,.\, P(O|C,T) \,.\, P(D|O,B) \,.\, P(X|O)$$

k, k, k values

$$g_1(S,O,D) = \sum_B P(B|S) \,.\, P(D|O\,B)$$

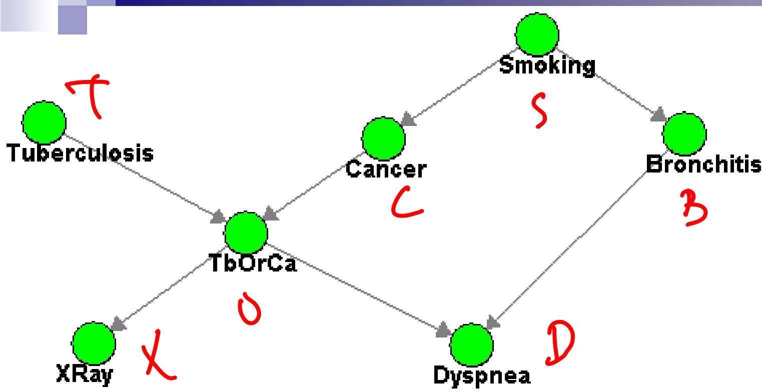
→ table of k^3 values

V.E. works on any graph! But, complexity depends on graph structure

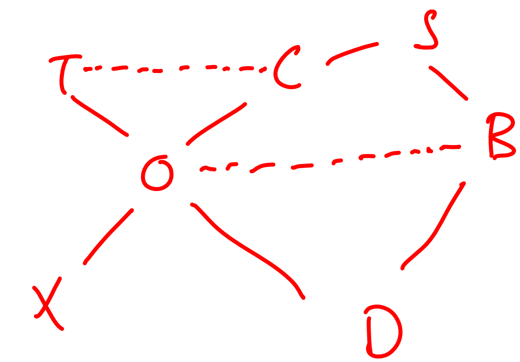
inference is #P-complete, even if nodes have small # of parents ";=)"
V.E. will generate large factors in some graphs

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



tree width of this graph

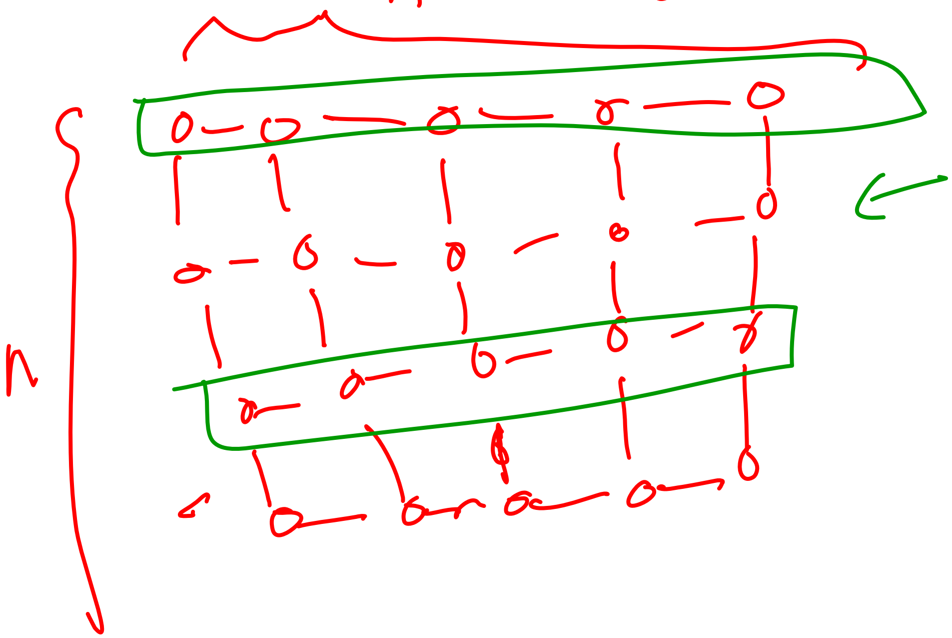
"max minimum node cut between pairs of vars"

non adjacent sets vars

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

moralized graph



must cut n nodes

⇒ treewidth $\geq n$

actually $= n$

Compact representation $\not\Rightarrow$ Easy inference ☹

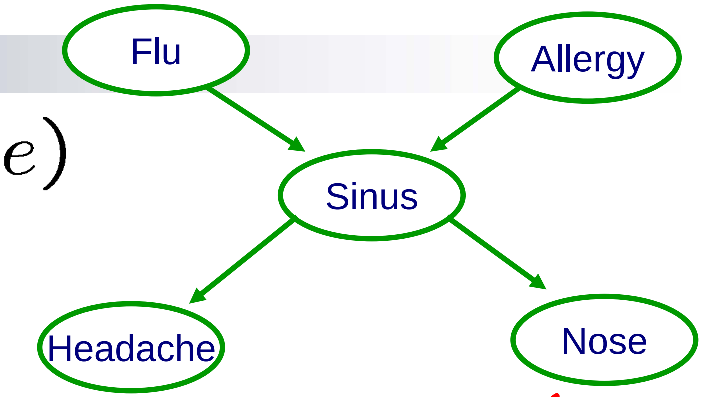
Choosing an elimination order

- Choosing best order is NP-complete
  - Reduction from MAX-Clique

see readings

- 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)



$N=t$

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

- Using Bayes rule:

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

constant

- 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

Forward pass

$$\max_F \max_S P(f)\,P(s|f)\cdot P(N=t|s)$$

$$\max_f P(f) \underbrace{\max_s P(s|f)\cdot P(N=t|s)}_{g_1(f)}$$

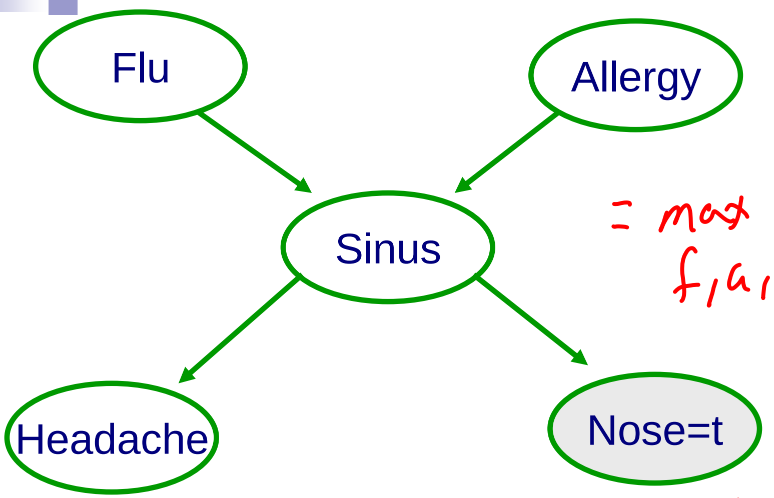
$$\max_f P(f)\cdot g_1(f) = g_2$$ ← constant ← prob. of best assignment

Backward pass

$$f^* = \arg\max_f P(f)\cdot g_1(f)$$

$$s^* = \arg\max_s P(s|f^*)\cdot P(N=t|s)$$

Example of variable elimination for MLE – Forward pass



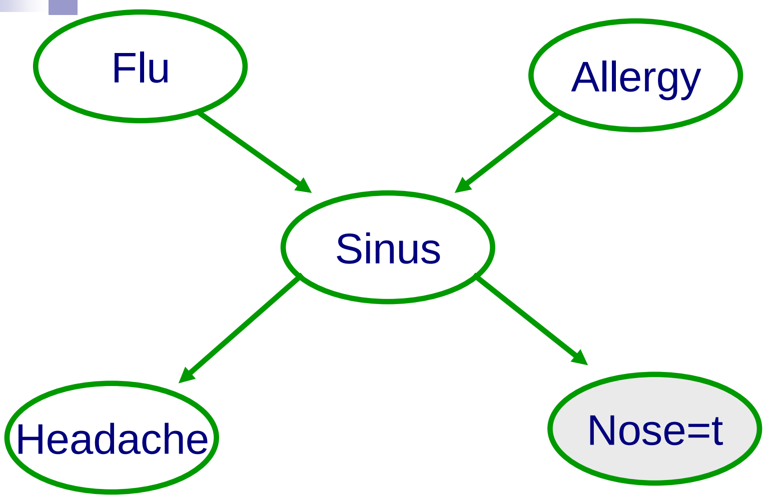
$$\max_{f,a,s,h} P(f) \cdot P(a) \cdot P(s|f,a) \cdot P(h|s) \cdot P(n=t|s)$$

$$= \max_{f,a,s} P(f) \cdot P(a)\, P(s|f,a)\, P(n=t|s) \underbrace{\max_{h} P(h|s)}_{g_1(s)}$$

$$= \max_{f,a} P(f) \cdot P(a) \underbrace{\max_{s} P(s|f,a) \cdot g_1(s)\, P(n=t|s)}_{g_2(f,a)}$$

↳ then eliminate f, a

Example of variable elimination for MLE – Backward pass



$$(f^*, a^*) = \arg\max_{f,a} P(f) \cdot P(a)\, g_2(f,a)$$

$$s^* = \arg\max_{s} P(s \mid f^*, a^*) \cdot P(N{=}t \mid s) \cdot g_1(s)$$

$$h^* = \arg\max_{h} P(h \mid s^*)$$

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

max marginal ← multiplication

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

  - Variable X_i has been eliminated!

remember / cache g's

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:

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

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

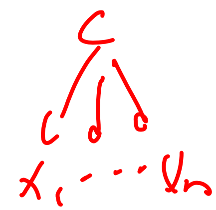
Carnegie Mellon University

March 26th, 2007

Adventures of our BN hero

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

1. Naïve Bayes



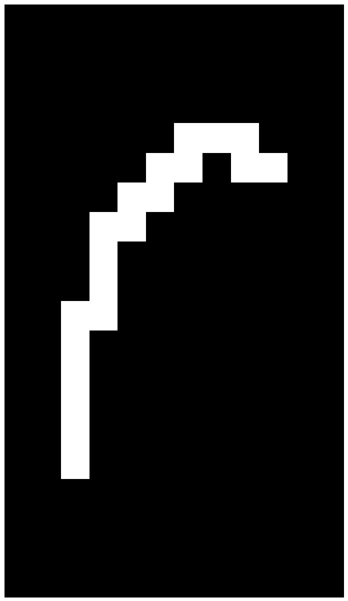
- But… Who are the most popular kids?

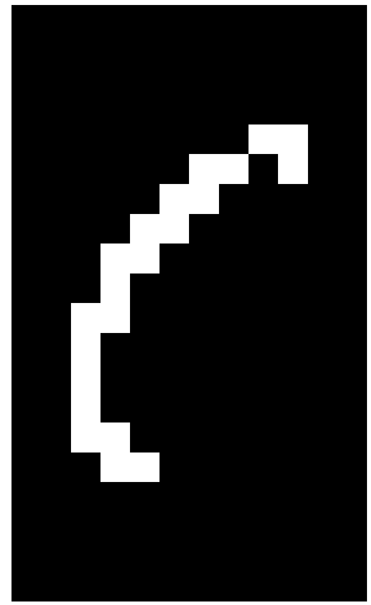
2 and 3.
Hidden Markov models (HMMs)
Kalman Filters

a HMM with Gaussians

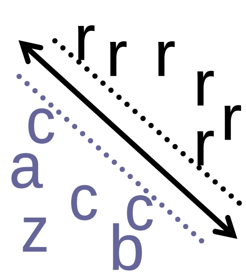
Kalman Filter with discrete dist.

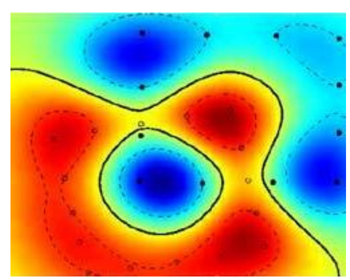
Handwriting recognition





Character recognition, e.g., kernel SVMs





Example of a hidden Markov model (HMM)

$P(Y_1)$

transition model

$P(Y_3 \mid Y_2)$

intuitively: prob of letter y_3 will follow letter y_2

true character

$$Y_1 \rightarrow Y_2 \rightarrow Y_3 \rightarrow Y_4 \rightarrow Y_5$$

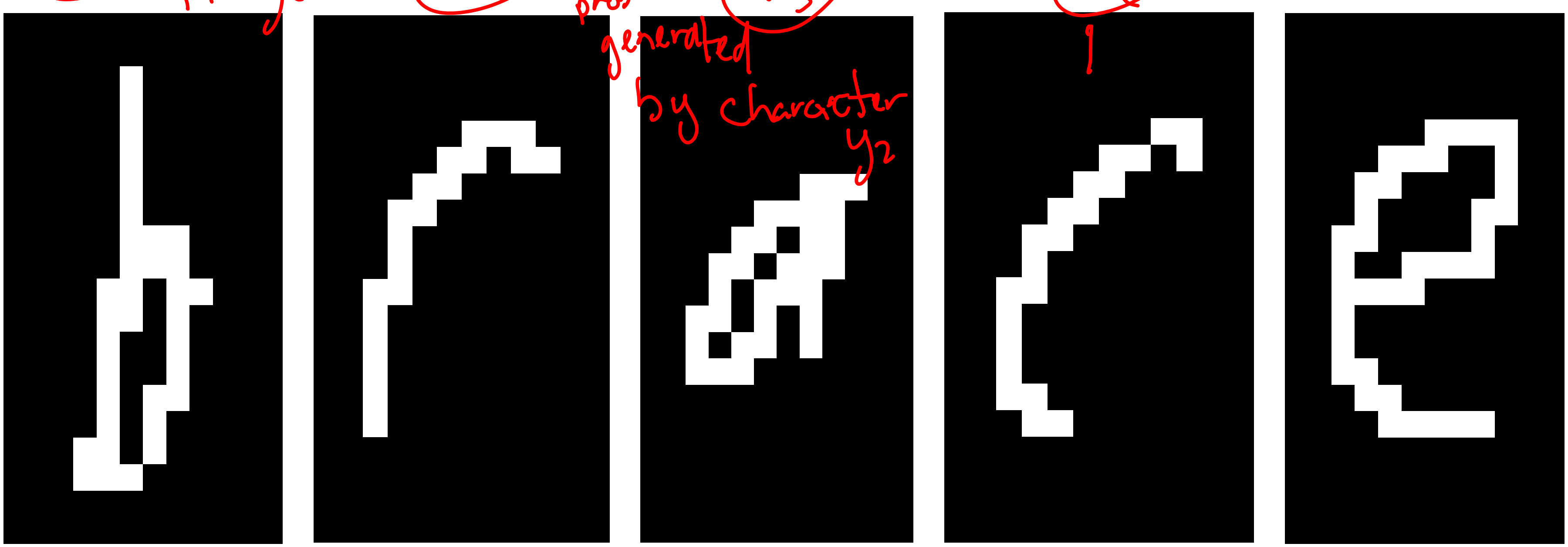
$$Y_1 \rightarrow X_1,\quad Y_2 \rightarrow X_2,\quad Y_3 \rightarrow X_3,\quad Y_4 \rightarrow X_4,\quad Y_5 \rightarrow X_5$$

prob. word starts with character y_1 (starting state prob.)

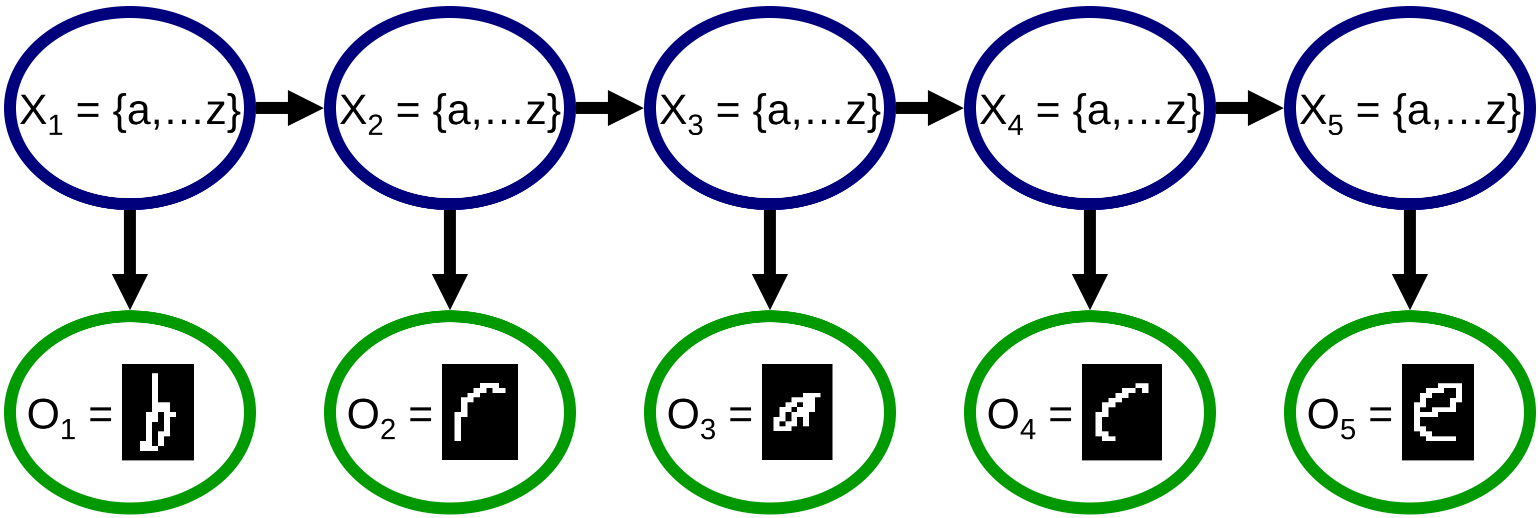
observation model

$P(X_2 \mid Y_2)$ prob. image X_2 generated by character y_2

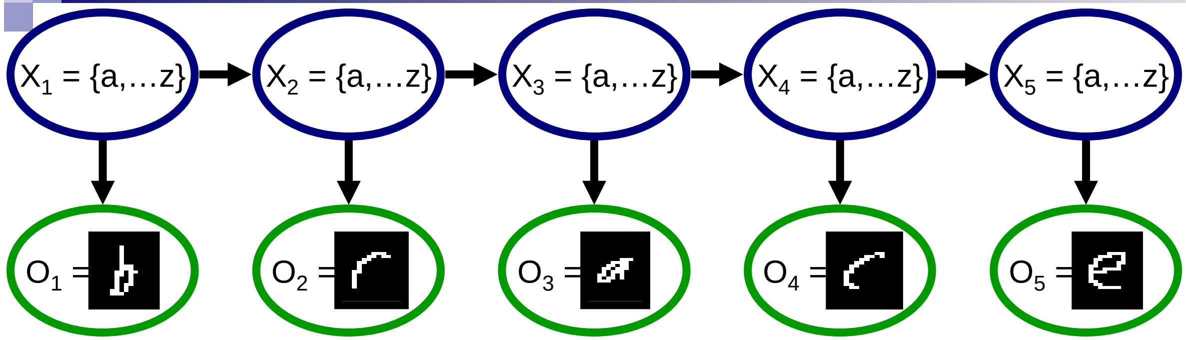
image



Understanding the HMM Semantics



HMMs semantics: Details



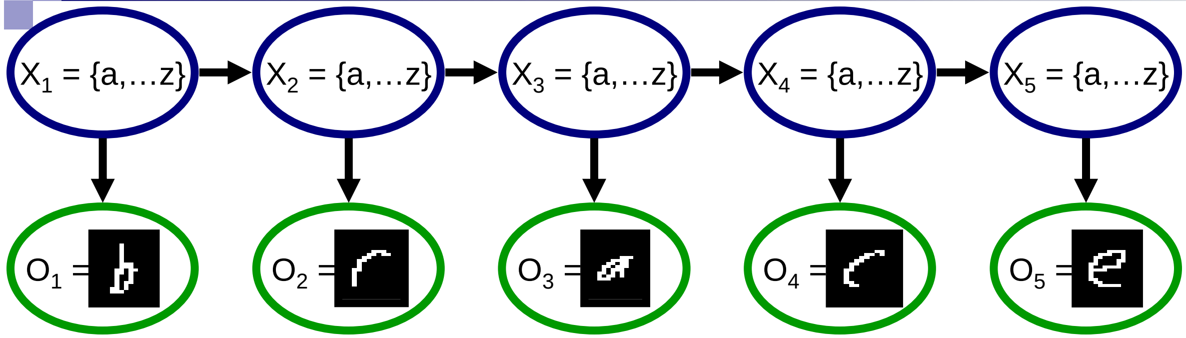
Just 3 distributions:

$$P(X_1)$$

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

$$P(O_i \mid X_i)$$

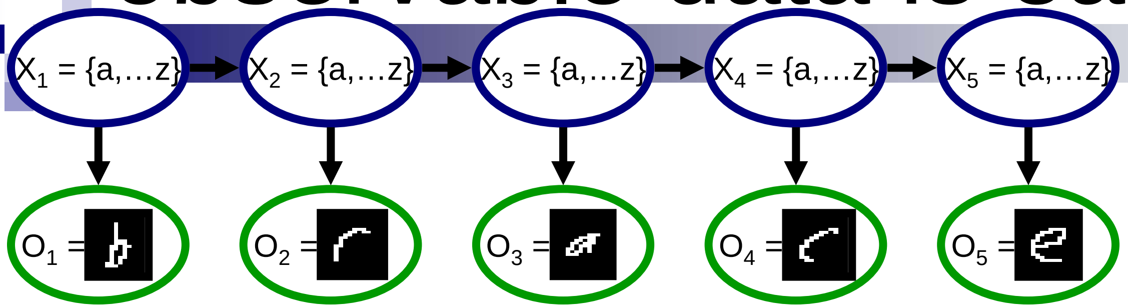
HMMs semantics: Joint distribution



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

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

Learning HMMs from fully observable data is easy



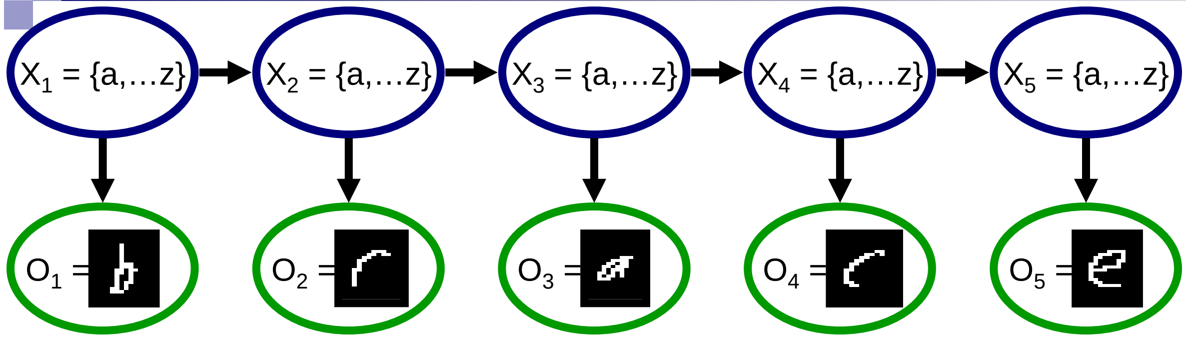
Learn 3 distributions:

$$P(X_1)$$

$$P(O_i \mid X_i)$$

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

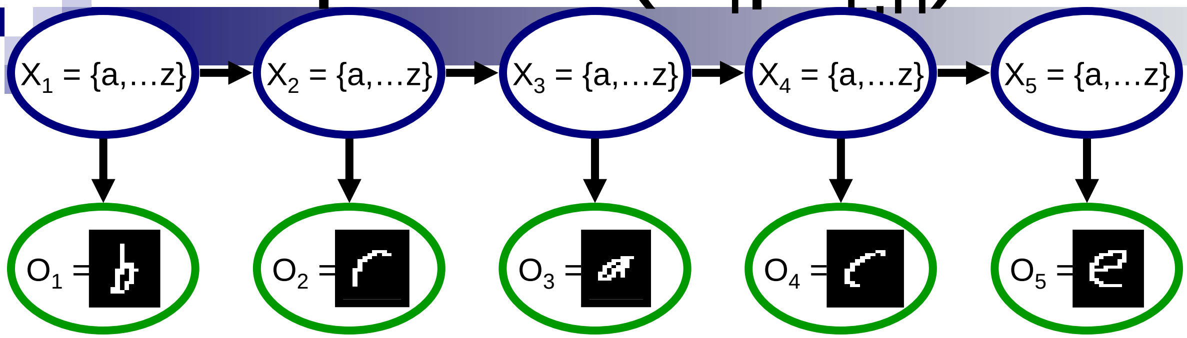
Possible inference tasks in an HMM



Marginal probability of a hidden variable:

Viterbi decoding – most likely trajectory for hidden vars:

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



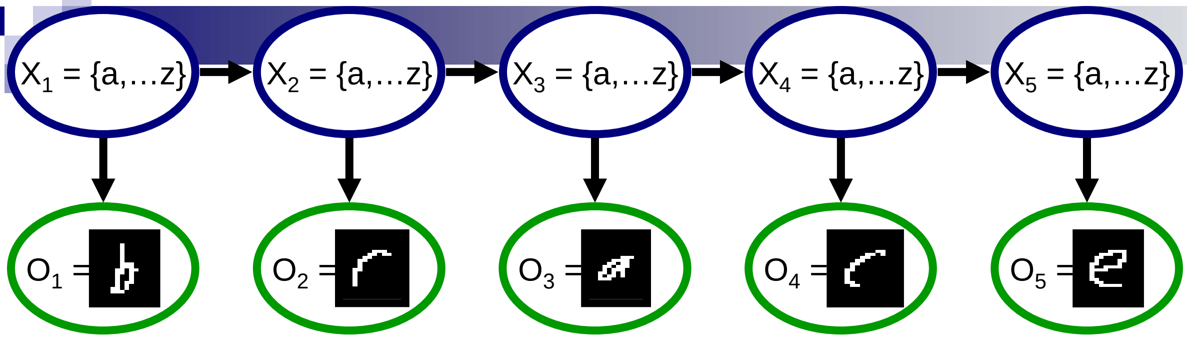
Compute:

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

Variable elimination order?

Example:

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



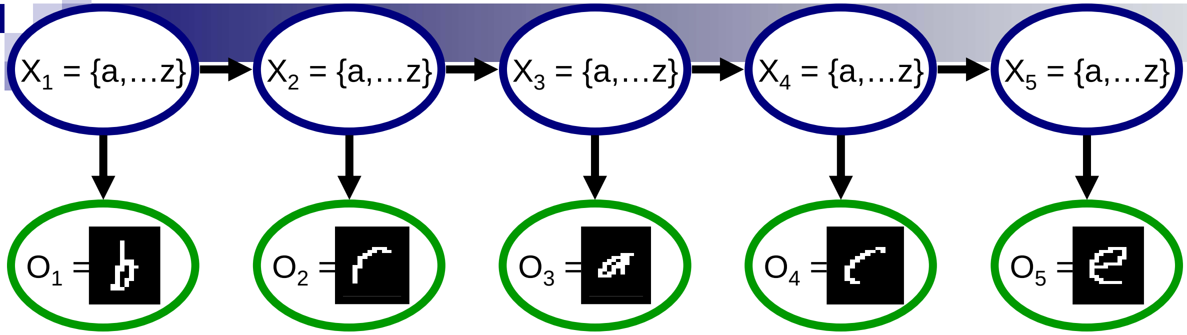
Compute:

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

Variable elimination for each i?

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

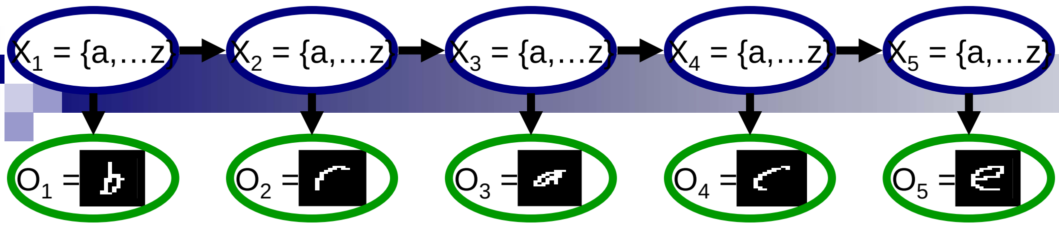
Reusing computation



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}

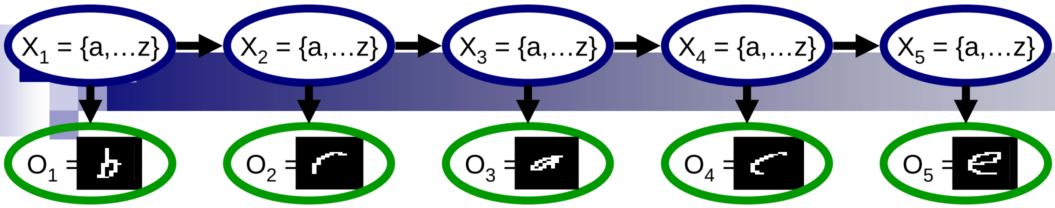
$$\alpha_i(X_i) = \sum_{x_{i-1}} P(o_i \mid X_i)P(X_i \mid X_{i-1} = x_{i-1})\alpha_{i-1}(x_{i-1})$$

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

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

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

Most likely explanation

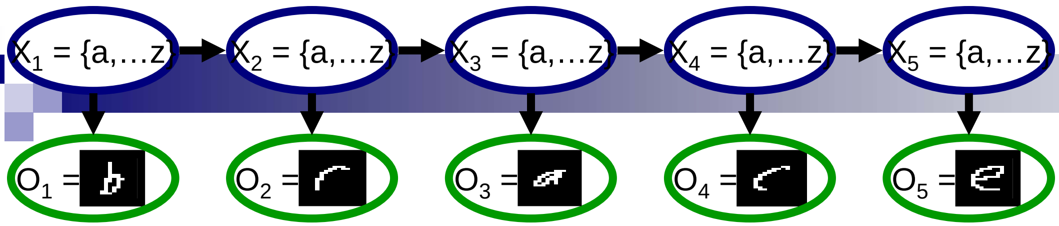


Compute:

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:

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

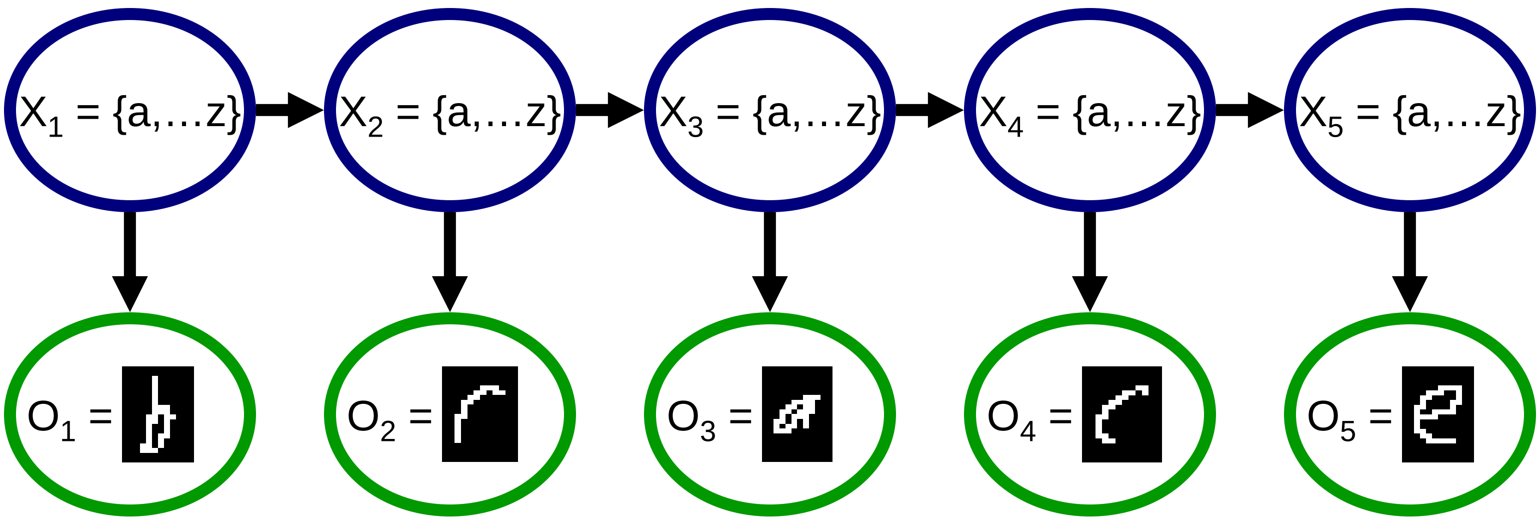
What you'll implement 1: multiplication

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

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