Machine Learning Department
School of Computer Science
Carnegie Mellon University

HMMs

+

Bayesian Networks

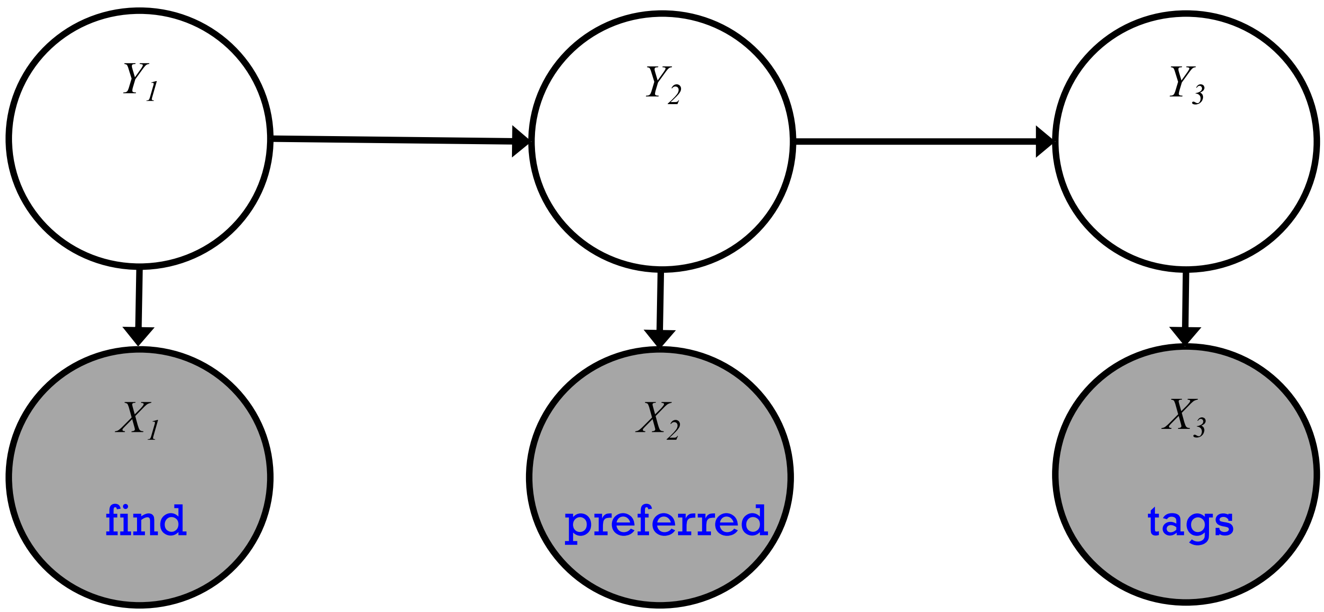
Matt Gormley
Lecture 20
Nov. 10, 2022

Reminders

- **Practice Problems: Exam 2**
  - Out: Fri, Nov. 4
- **Exam 2**
  - Thu, Nov. 10, 6:30pm – 8:30pm
- **Homework 7: Hidden Markov Models**
  - Out: Fri, Nov. 11
  - Due: Mon, Nov. 21 at 11:59pm

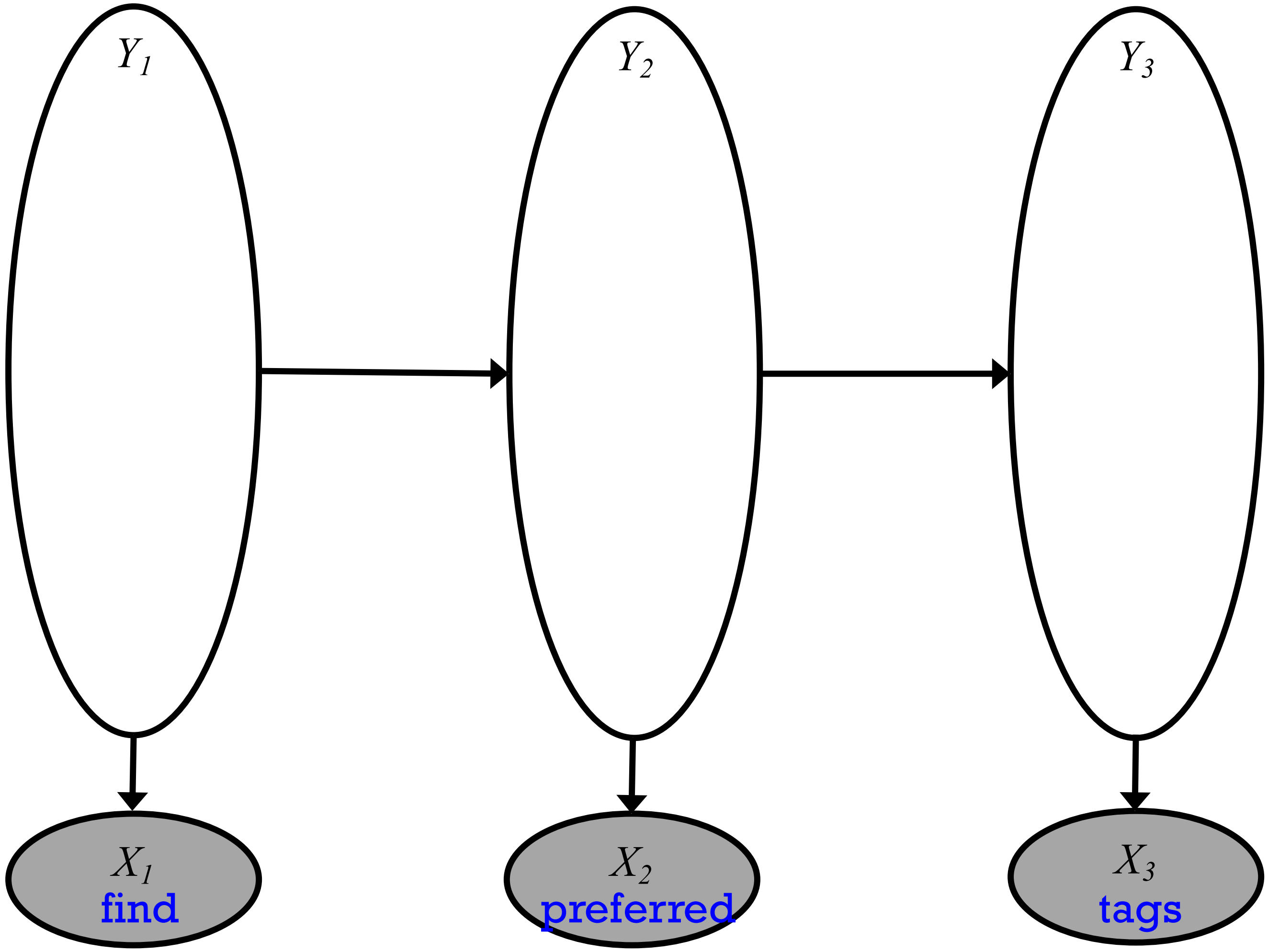
EXAMPLE: FORWARD-BACKWARD ON THREE WORDS

Forward-Backward Algorithm

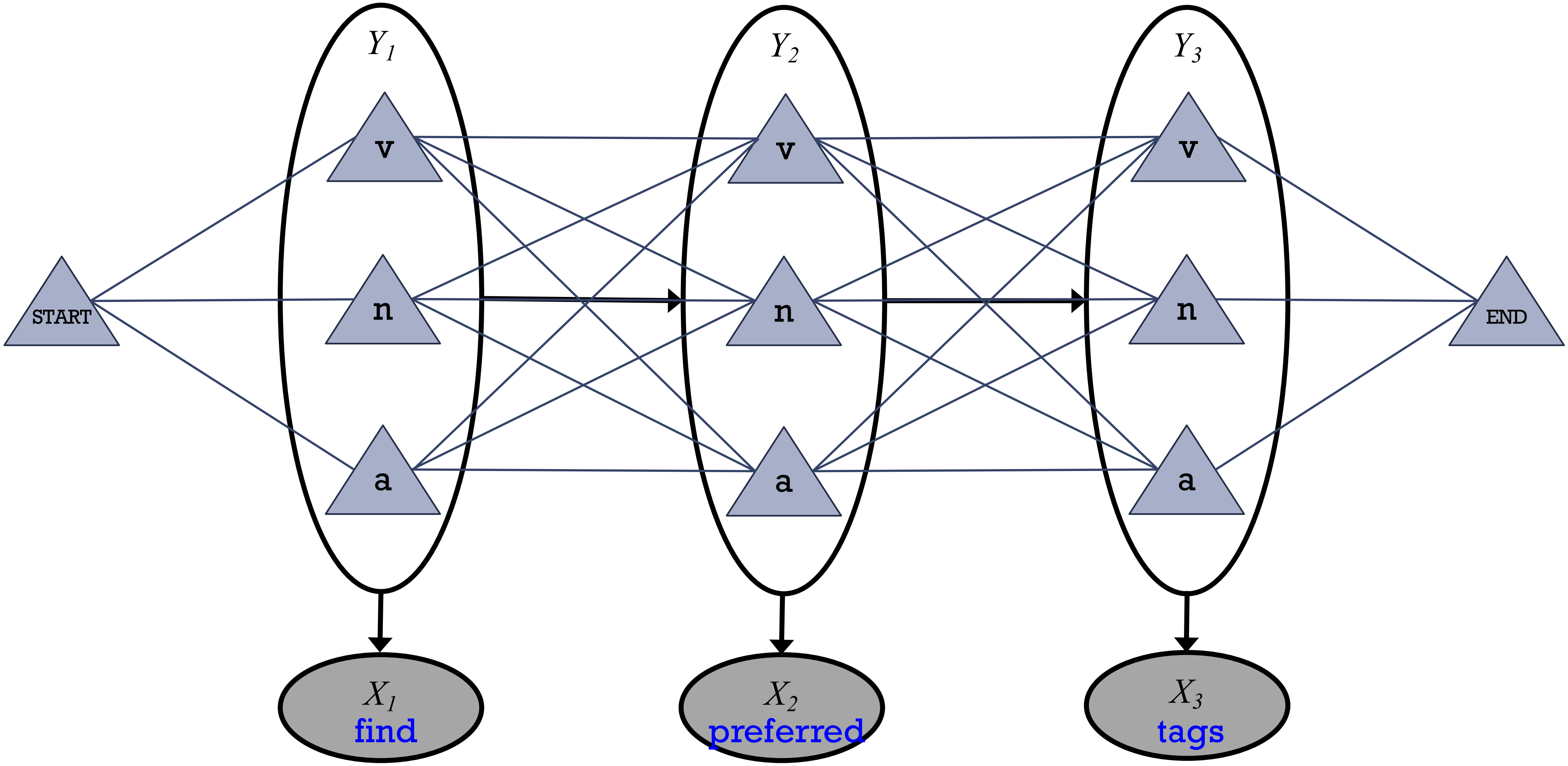


Could be verb or noun *Could be adjective or verb* *Could be noun or verb*

Forward-Backward Algorithm

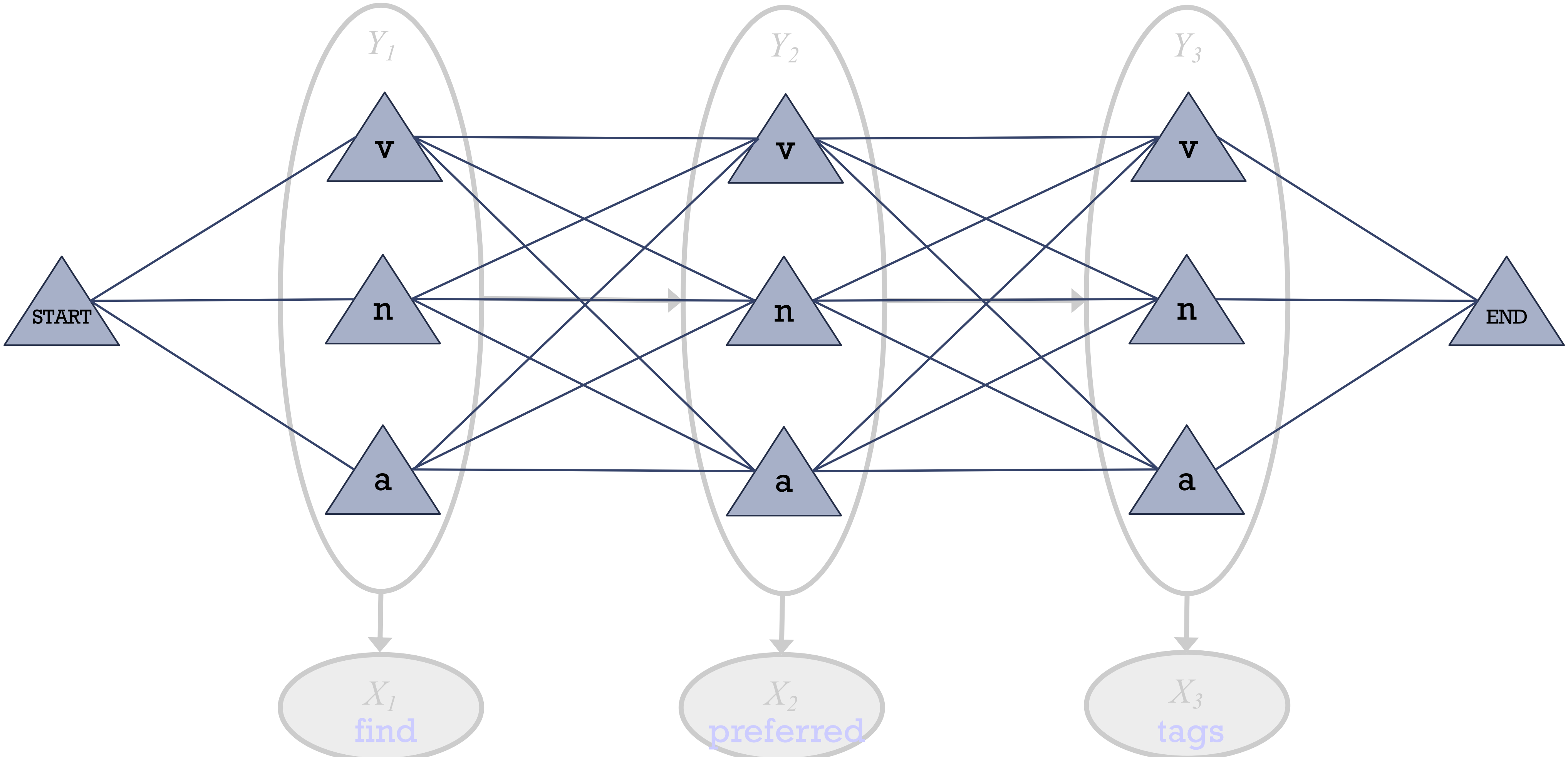


Forward-Backward Algorithm



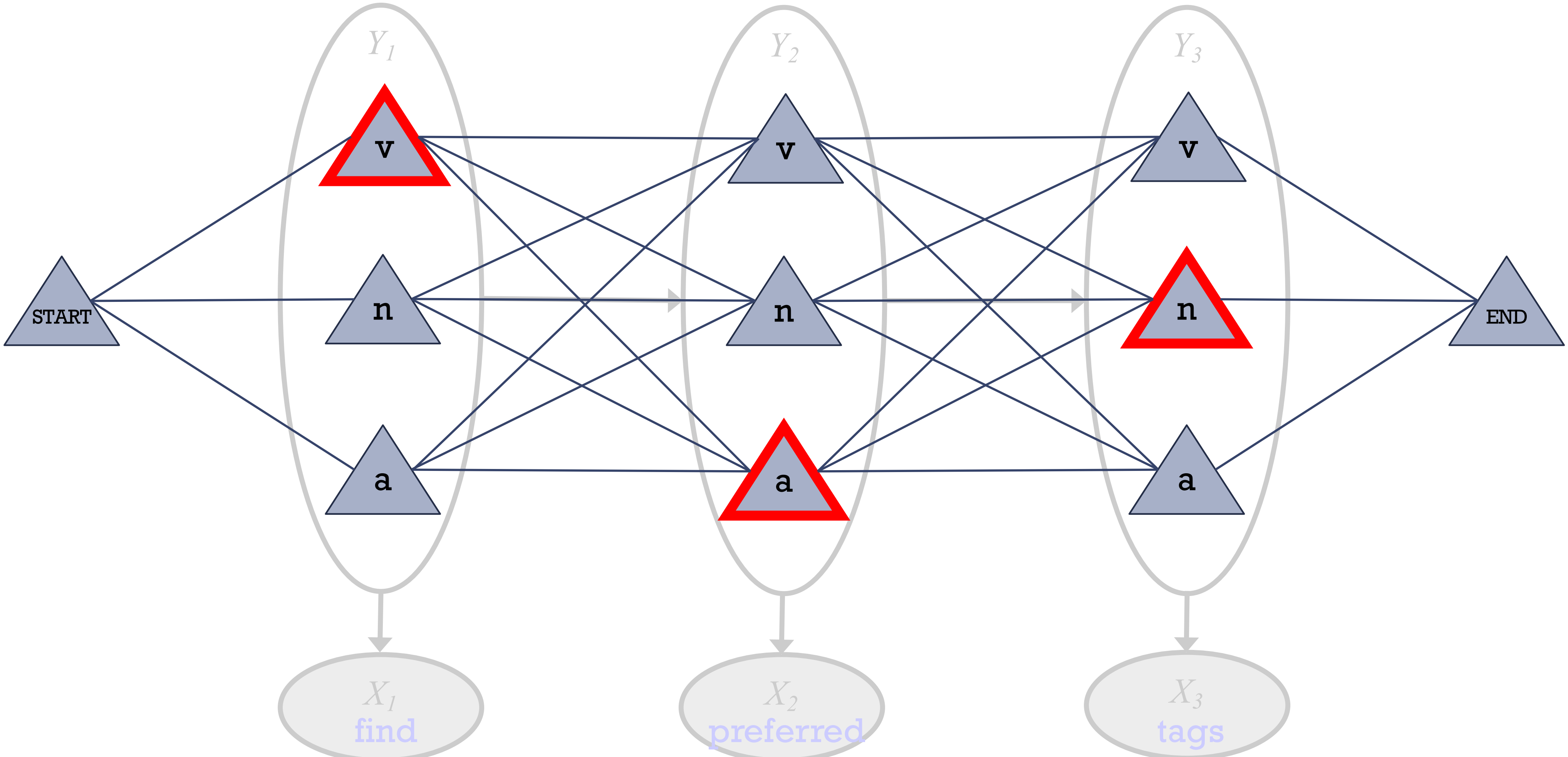
- Let's show the possible *values* for each variable

Forward-Backward Algorithm



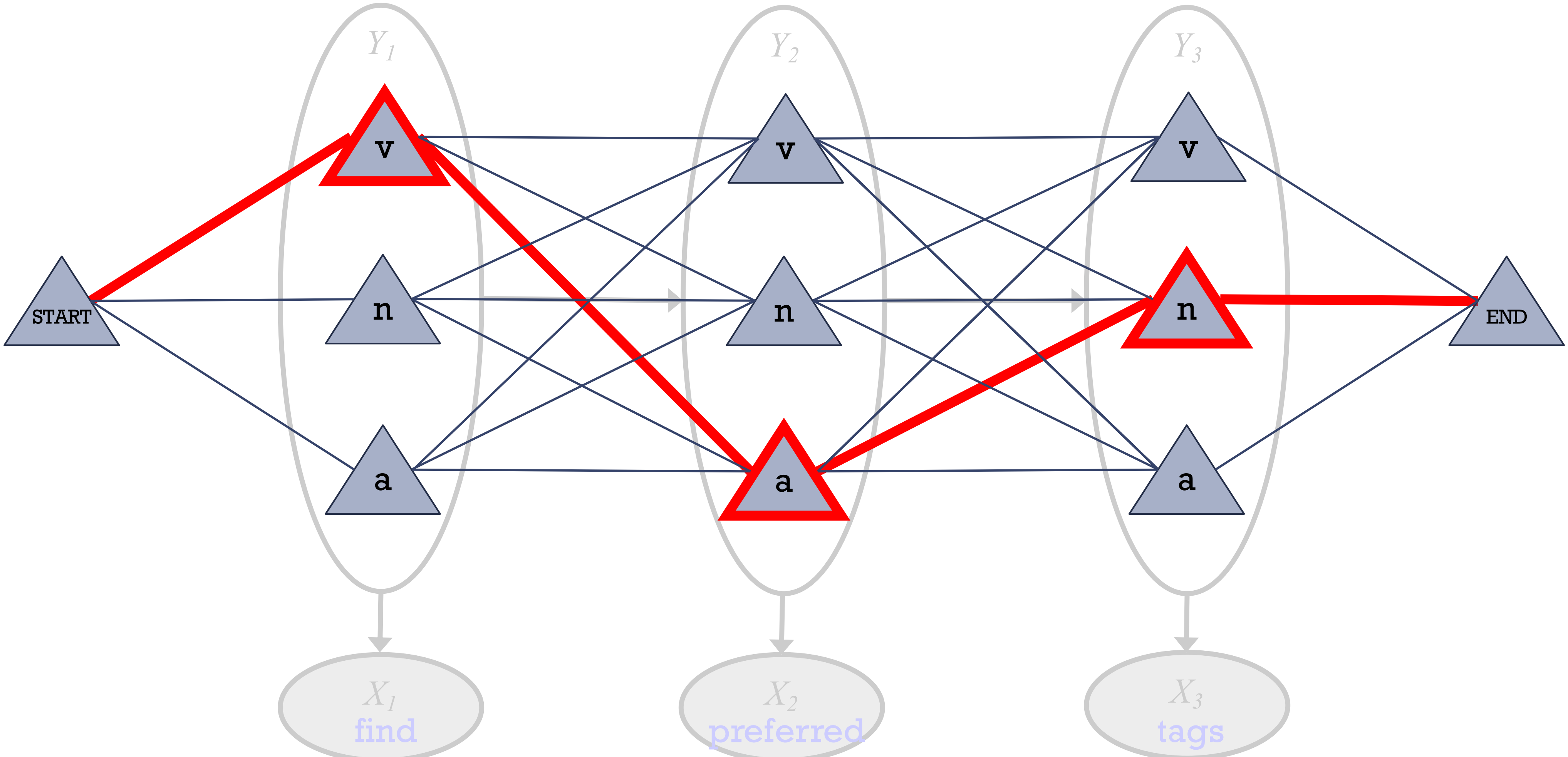
- Let's show the possible *values* for each variable

Forward-Backward Algorithm



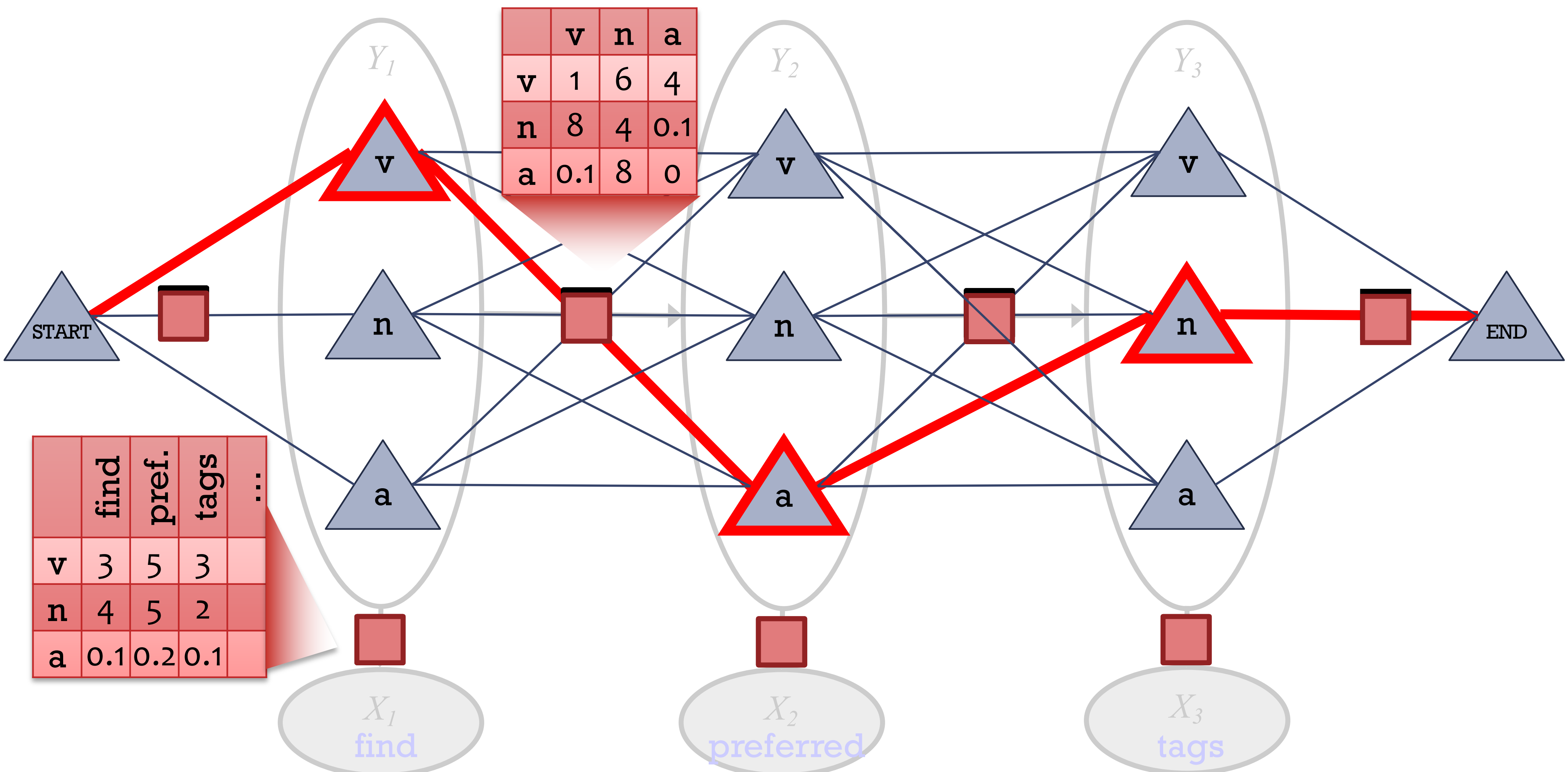
- Let's show the possible *values* for each variable
- One possible assignment

Forward-Backward Algorithm



- Let's show the possible *values* for each variable
- One possible assignment
- And what the 7 transition / emission factors think of it …

Forward-Backward Algorithm

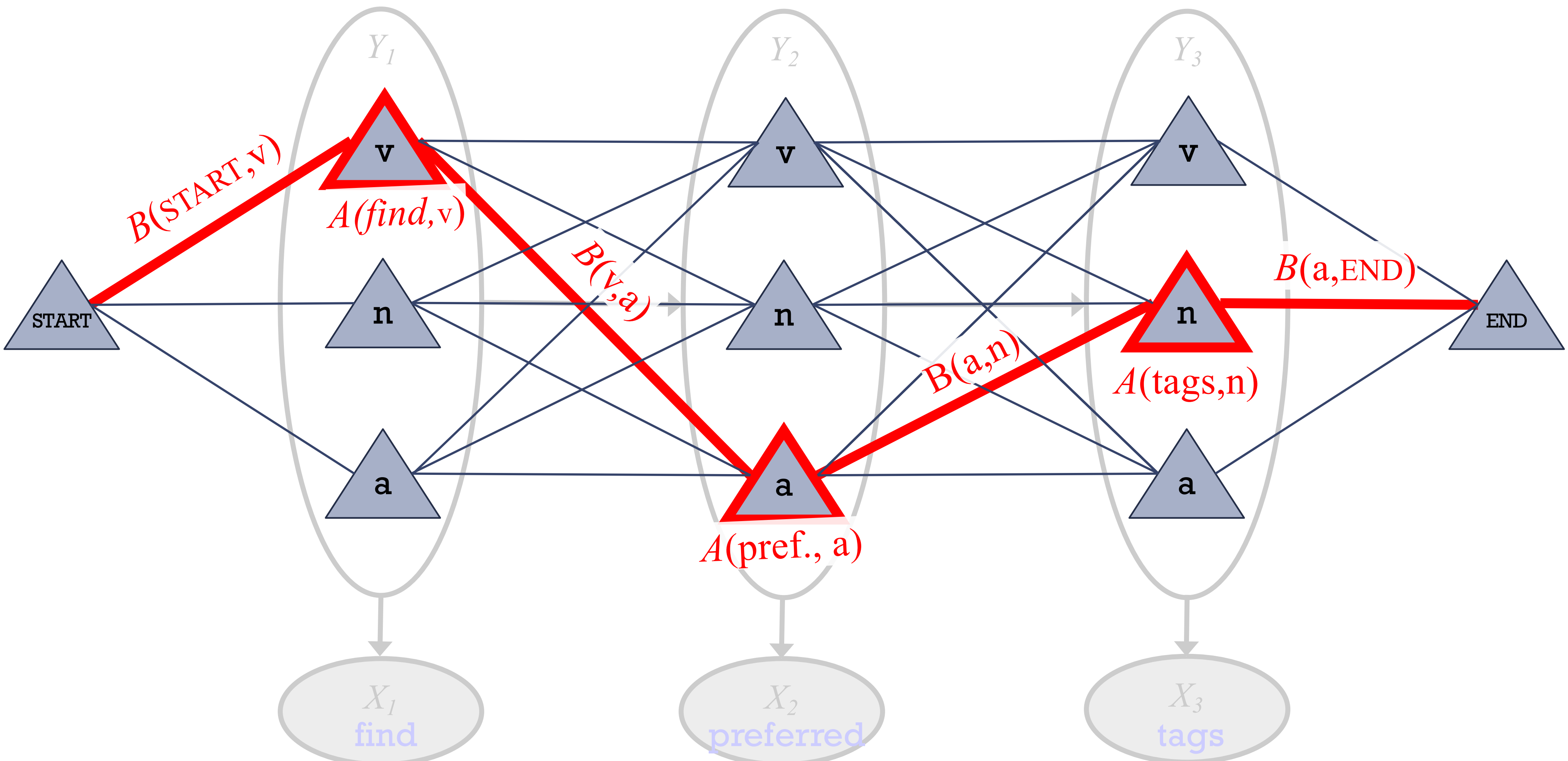


	v	n	a
v	1	6	4
n	8	4	0.1
a	0.1	8	0

	find	pref.	tags	…
v	3	5	3	
n	4	5	2	
a	0.1	0.2	0.1	

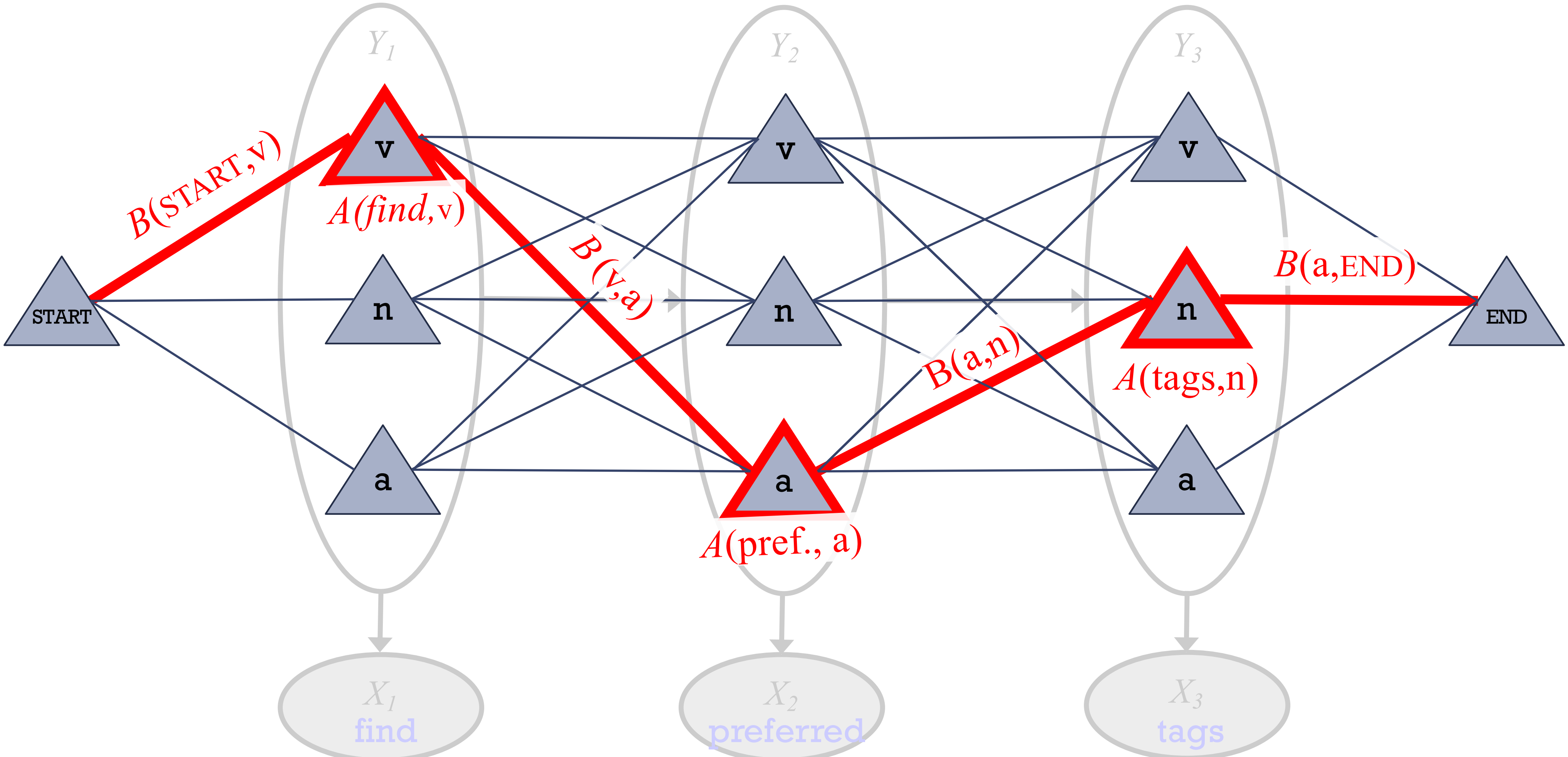
- Let’s show the possible *values* for each variable
- One possible assignment
- And what the 7 transition / emission factors think of it …

Viterbi Algorithm: Most Probable Assignment



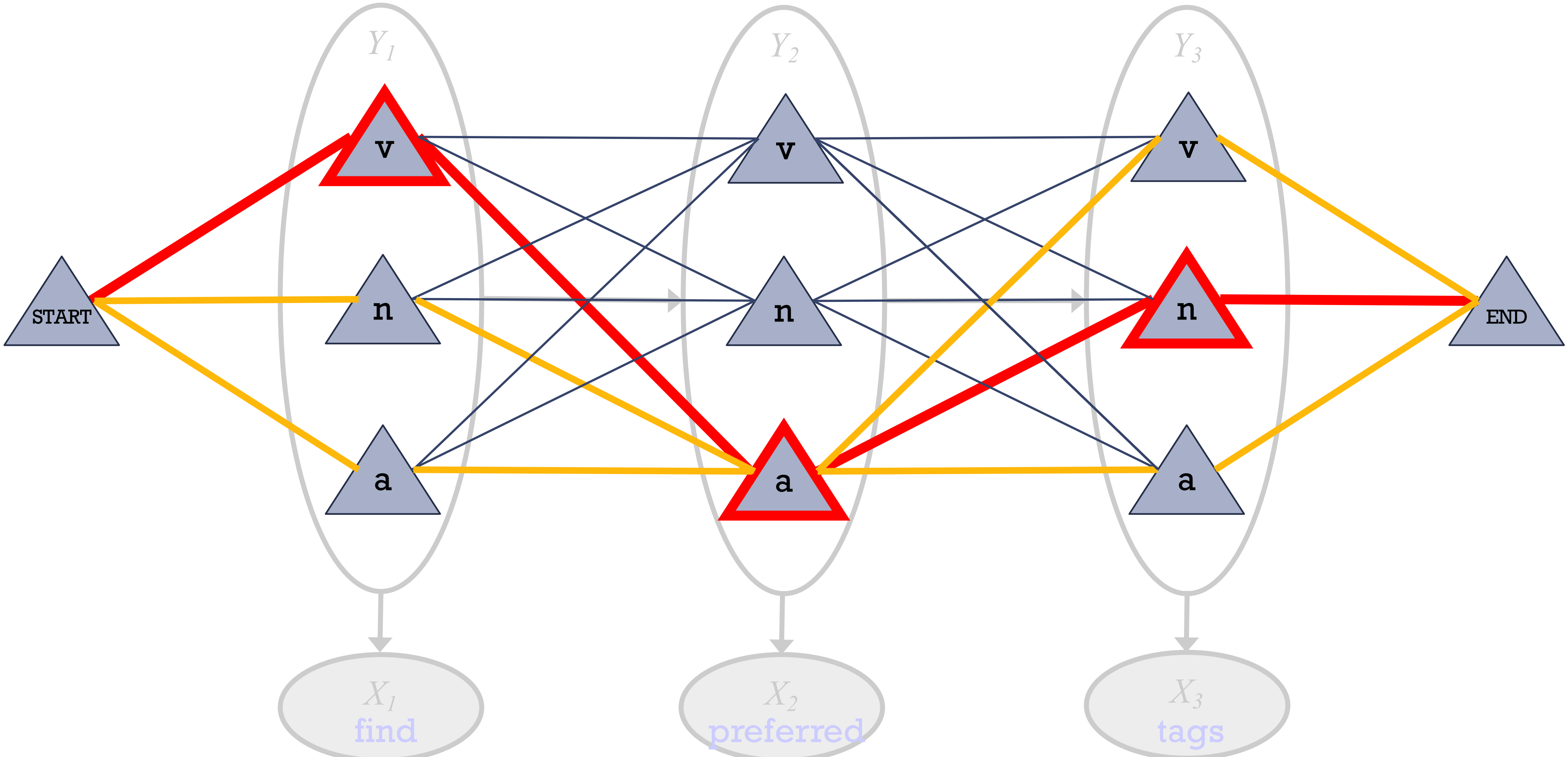
- So p(**v a n**) = (1/Z) * product of 7 numbers
- Numbers associated with edges and nodes of path
- Most probable assignment = **path with highest product**

Viterbi Algorithm: Most Probable Assignment



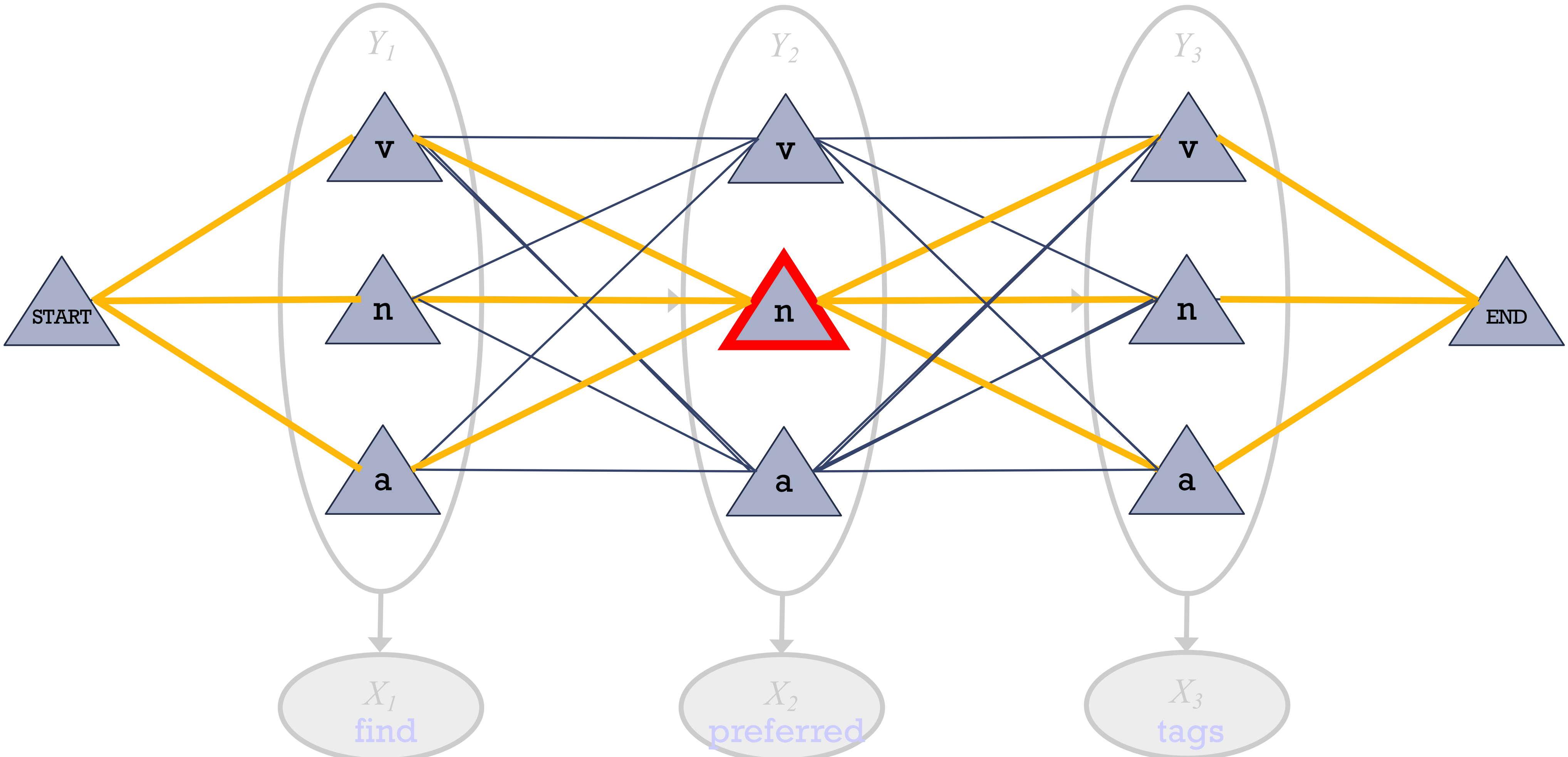
- So p(**v a n**) = (1/Z) * product weight of one path

Forward-Backward Algorithm: Finds Marginals



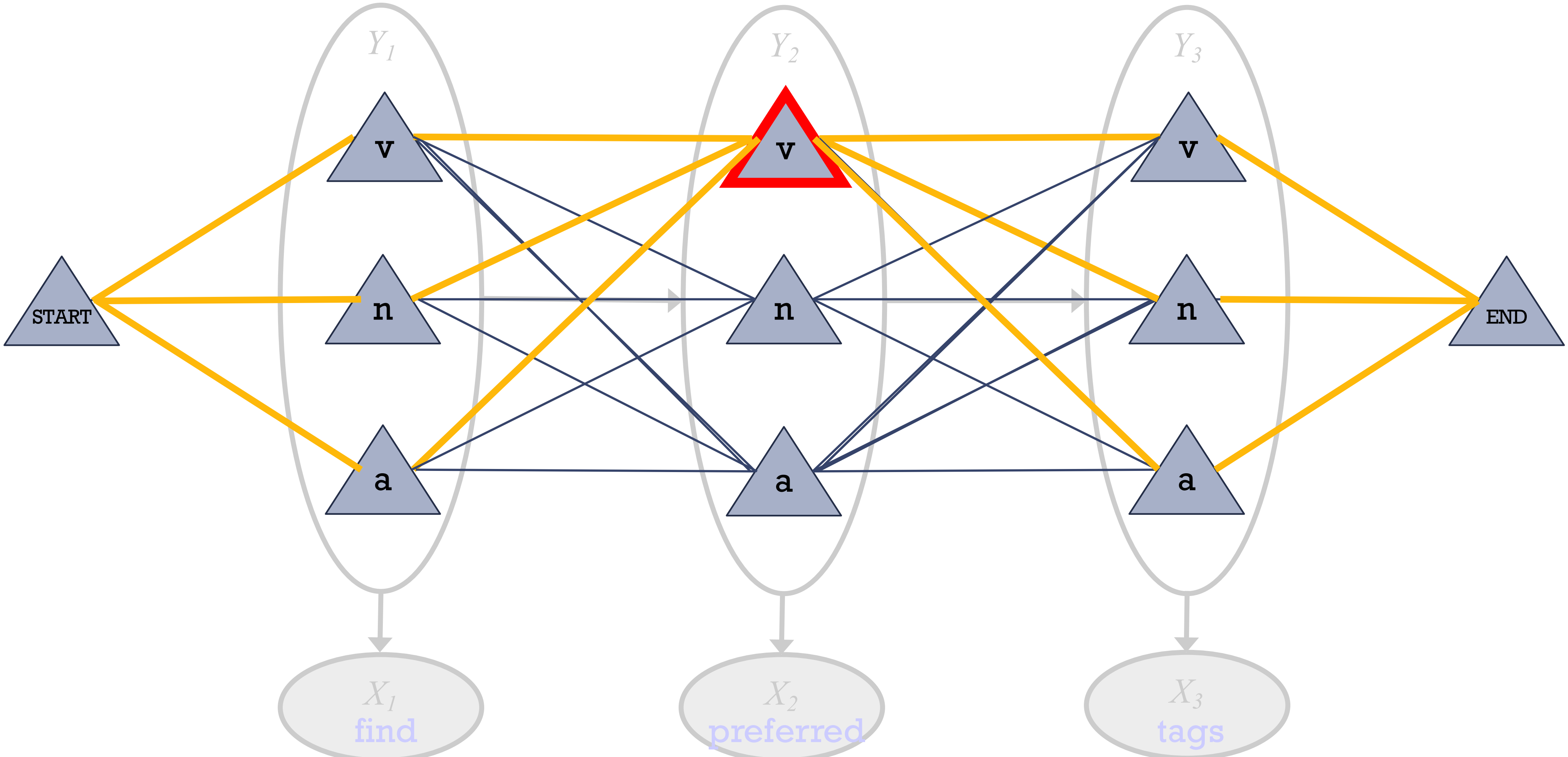
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability p(Y_2 = a)
  = (1/Z) * total weight of *all* paths through a

Forward-Backward Algorithm: Finds Marginals



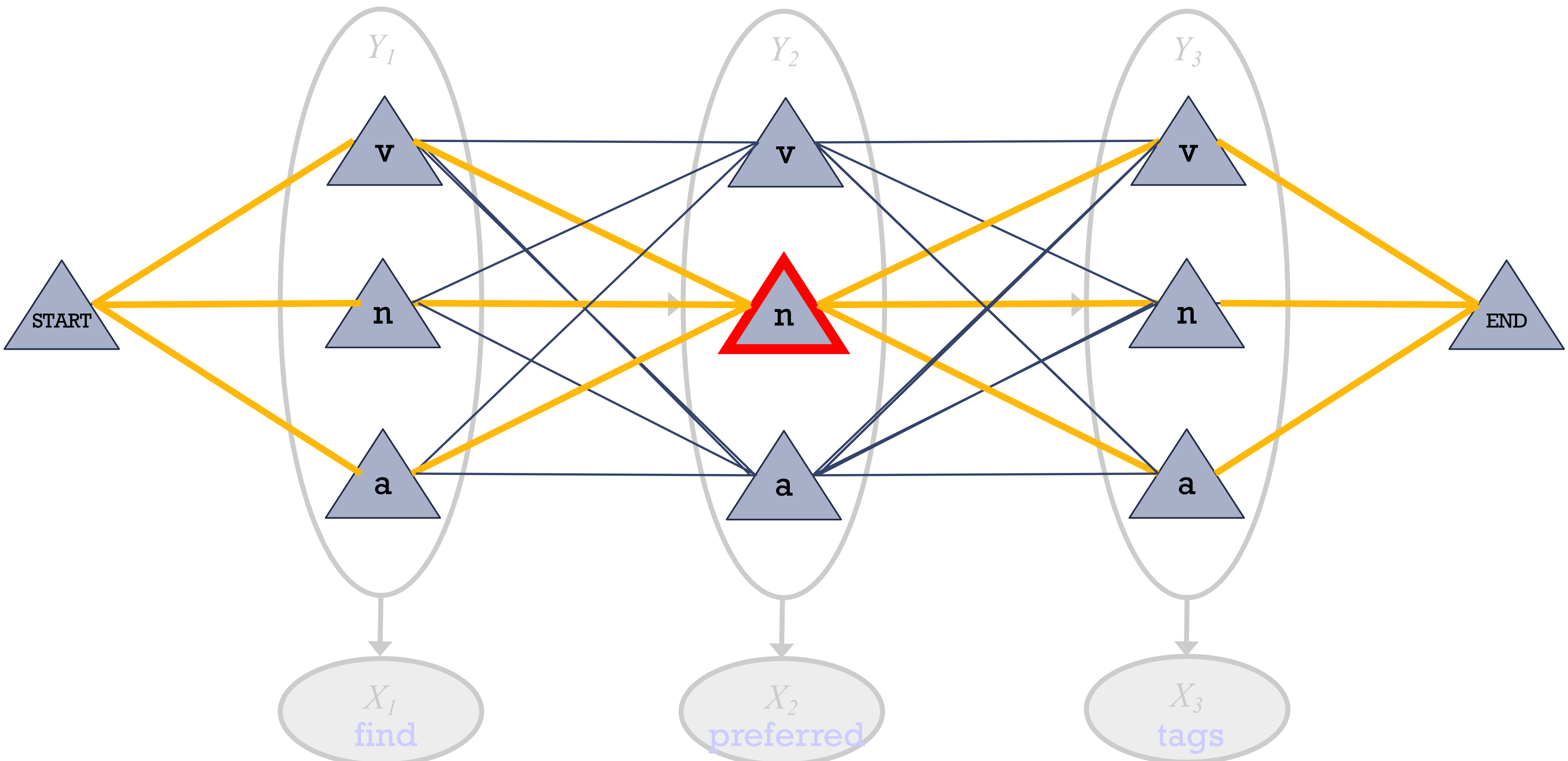
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(Y_2 = n)$
  = (1/Z) * total weight of *all* paths through **n**

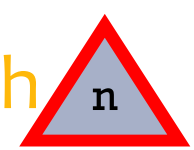
Forward-Backward Algorithm: Finds Marginals



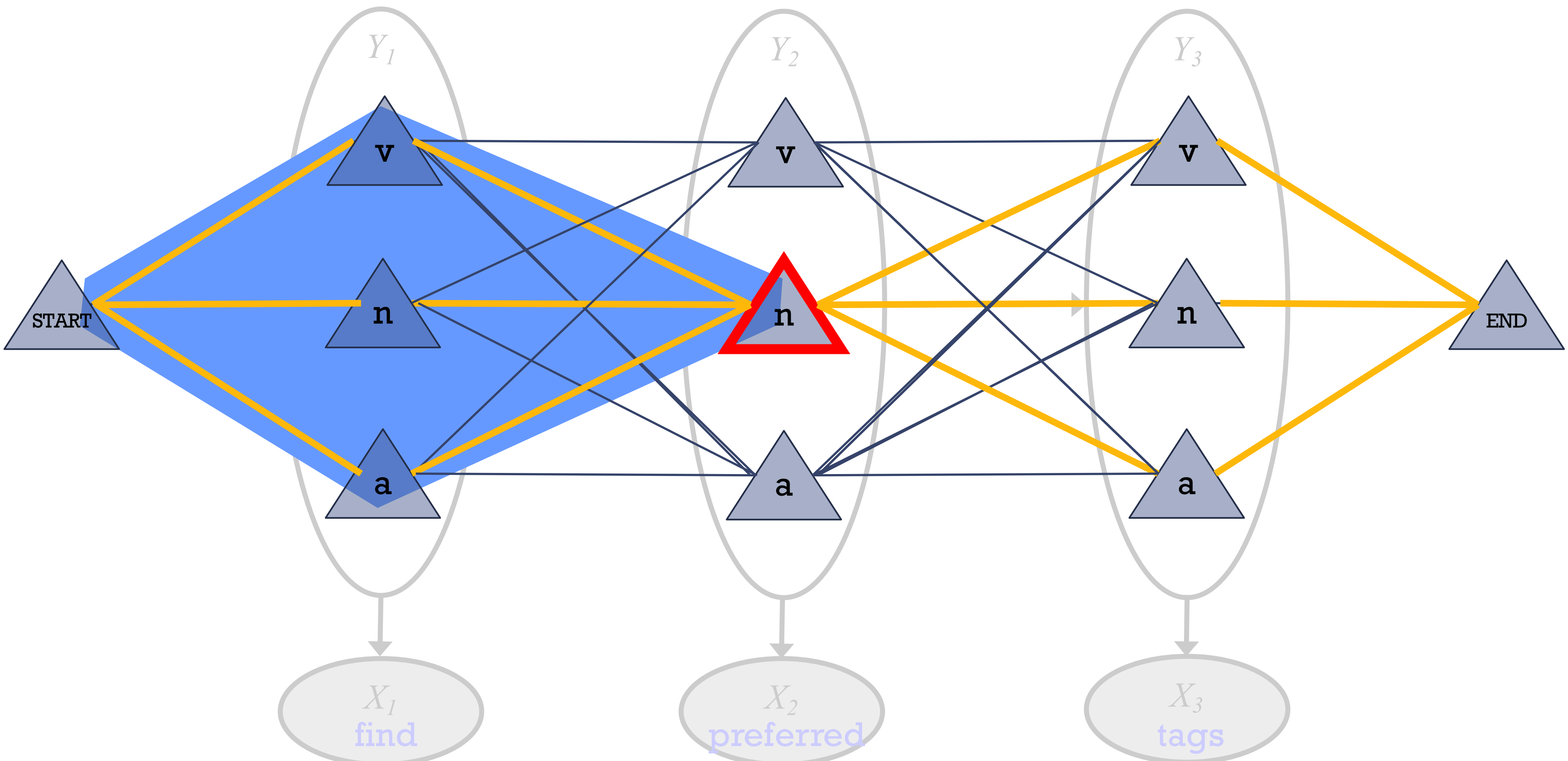
- So $p(\mathbf{v\ a\ n}) = (1/Z)$ * product weight of one path
- Marginal probability $p(Y_2 = \text{v})$
  $= (1/Z)$ * total weight of *all* paths through v

Forward-Backward Algorithm: Finds Marginals



- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(Y_2 = n)$
  = (1/Z) * total weight of *all* paths through 

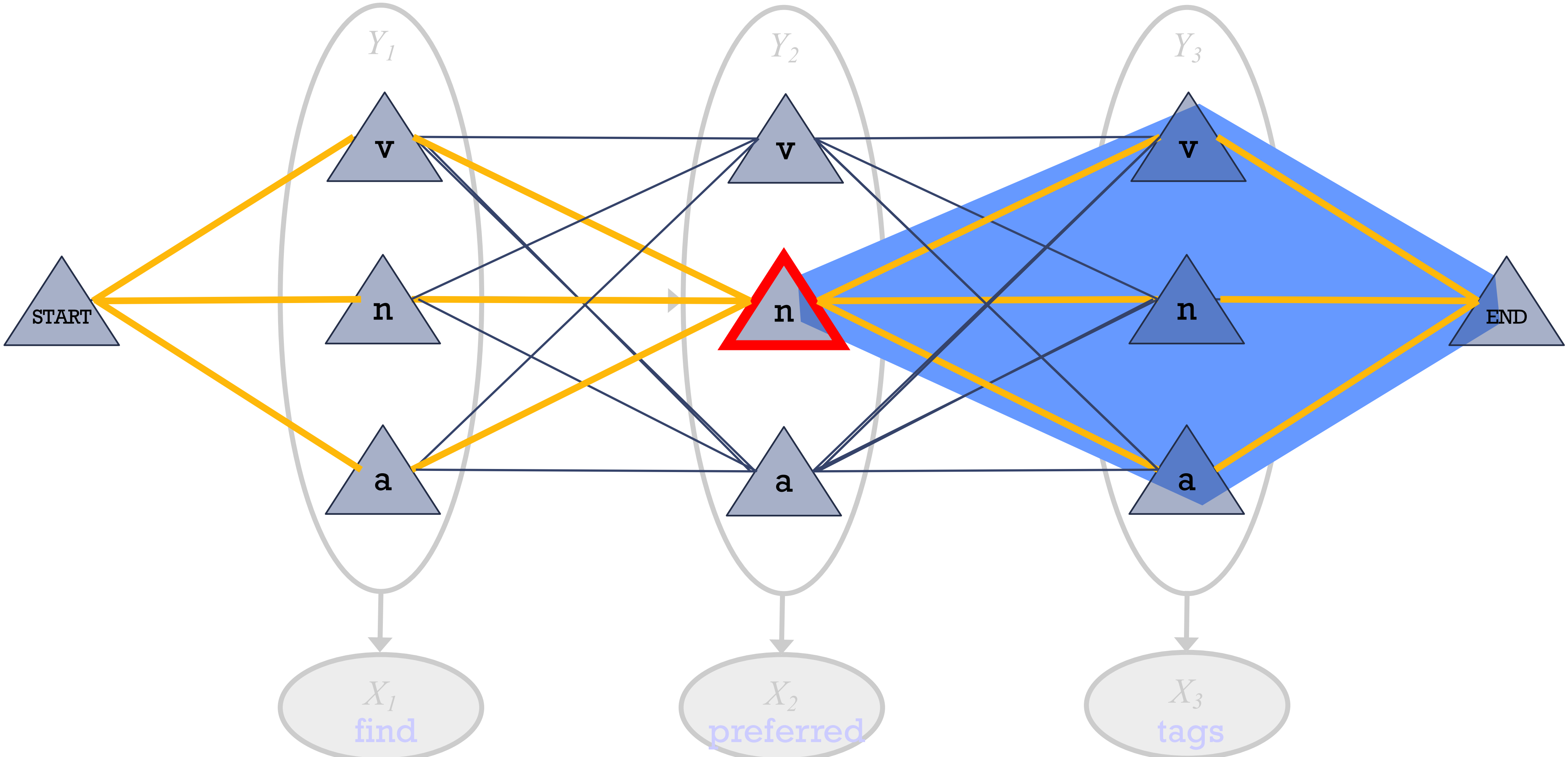
Forward-Backward Algorithm: Finds Marginals



$\alpha_2(\mathbf{n})$ = total weight of these path *prefixes*

(found by dynamic programming: matrix-vector products)

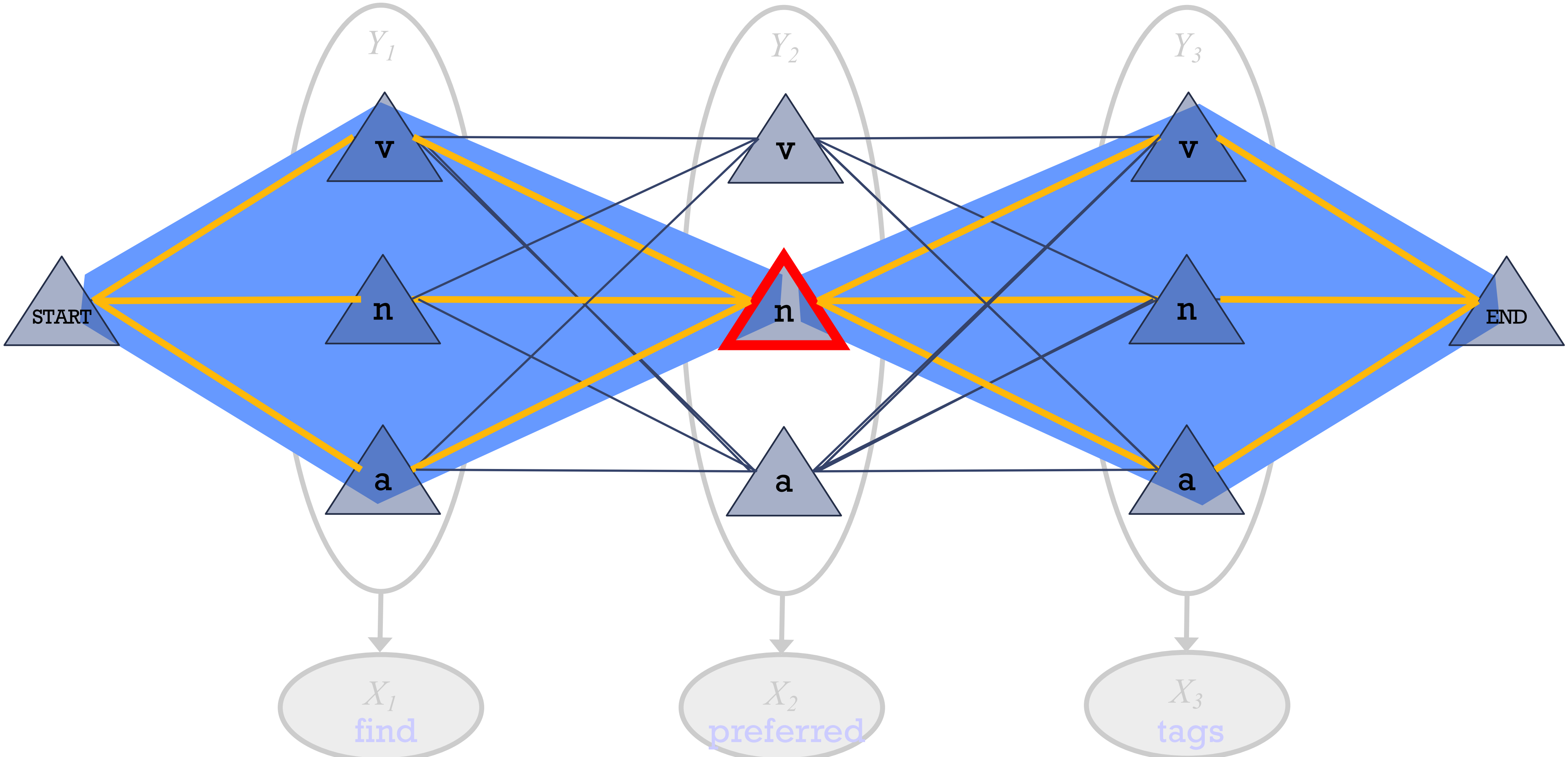
Forward-Backward Algorithm: Finds Marginals



$\beta_2(\mathbf{n})$ = total weight of these path *suffixes*

(found by dynamic programming: matrix-vector products)

Forward-Backward Algorithm: Finds Marginals



$\alpha_2(\mathbf{n})$ = total weight of these path *prefixes* (a + b + c)

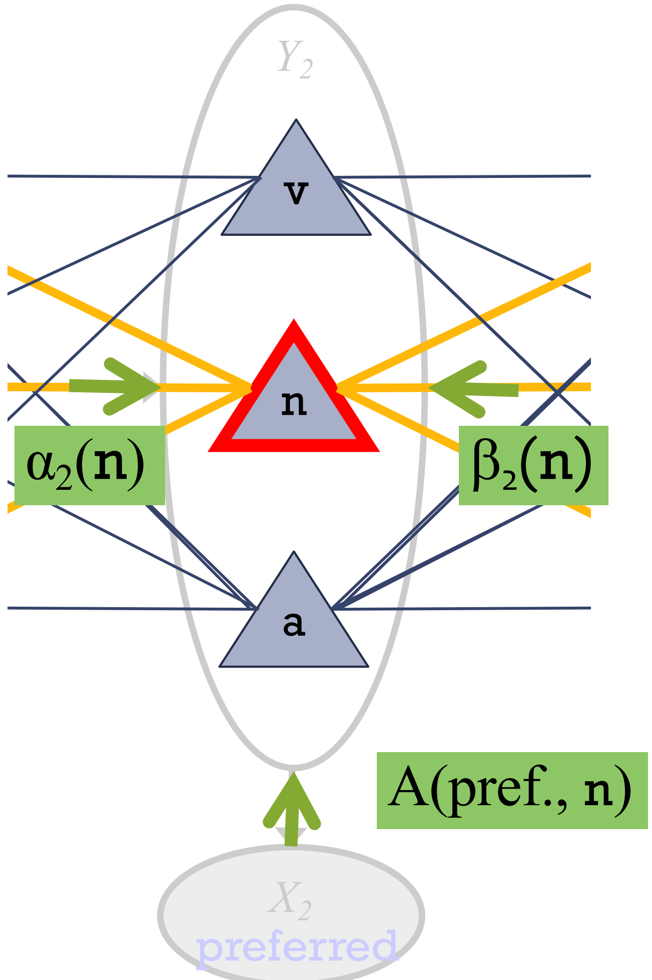
$\beta_2(\mathbf{n})$ = total weight of these path *suffixes* (x + y + z)

Product gives ax+ay+az+bx+by+bz+cx+cy+cz = total weight of paths

Forward-Backward Algorithm: Finds Marginals

Oops! The weight of a path through a state also includes a weight at that state.
So $\alpha(\mathbf{n})\cdot\beta(\mathbf{n})$ isn't enough.

The extra weight is the opinion of the emission probability at this variable.

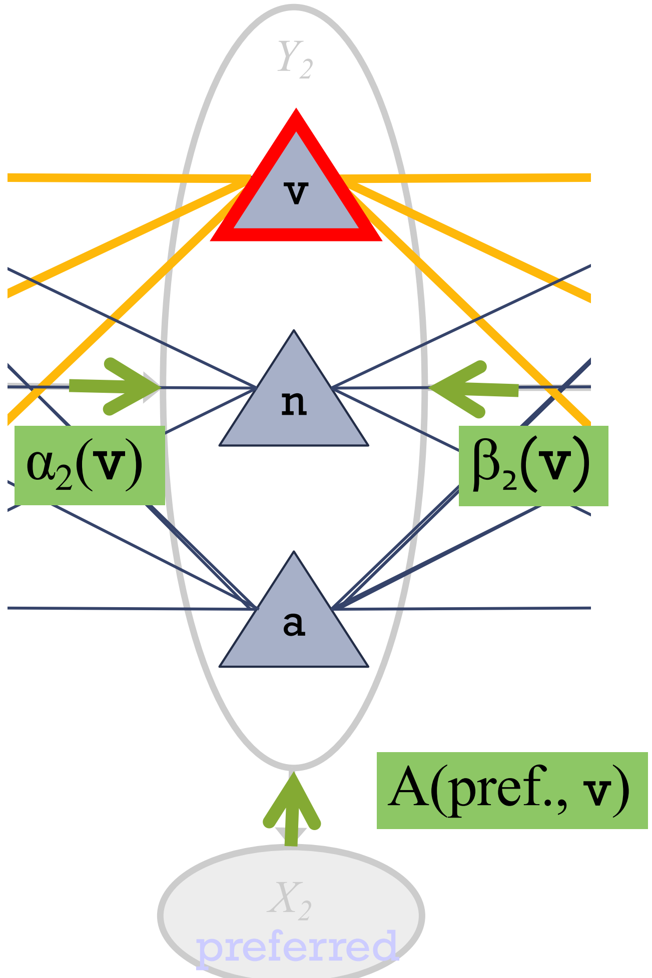


"belief that $Y_2 = \mathbf{n}$"

total weight of *all* paths through n

$= \alpha_2(\mathbf{n})\ \mathrm{A(pref., \mathbf{n})}\ \beta_2(\mathbf{n})$

Forward-Backward Algorithm: Finds Marginals



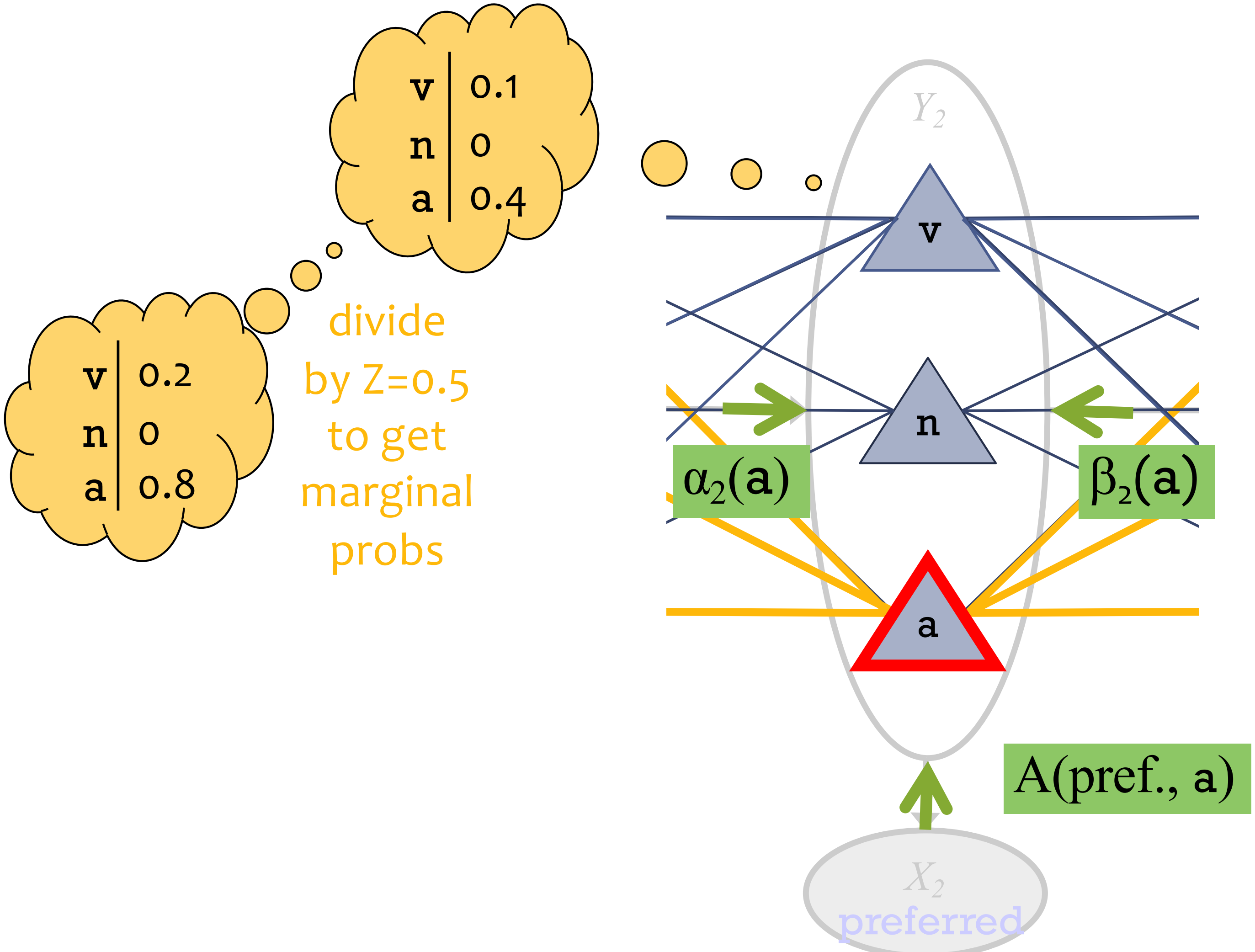
"belief that Y_2 = **v**"

"belief that Y_2 = **n**"

total weight of *all* paths through **v**

= $\alpha_2(\mathbf{v})$ A(pref., $\mathbf{v}$) $\beta_2(\mathbf{v})$

Forward-Backward Algorithm: Finds Marginals

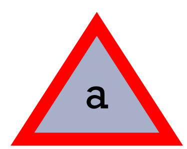


"belief that Y_2 = **v**"

"belief that Y_2 = **n**"

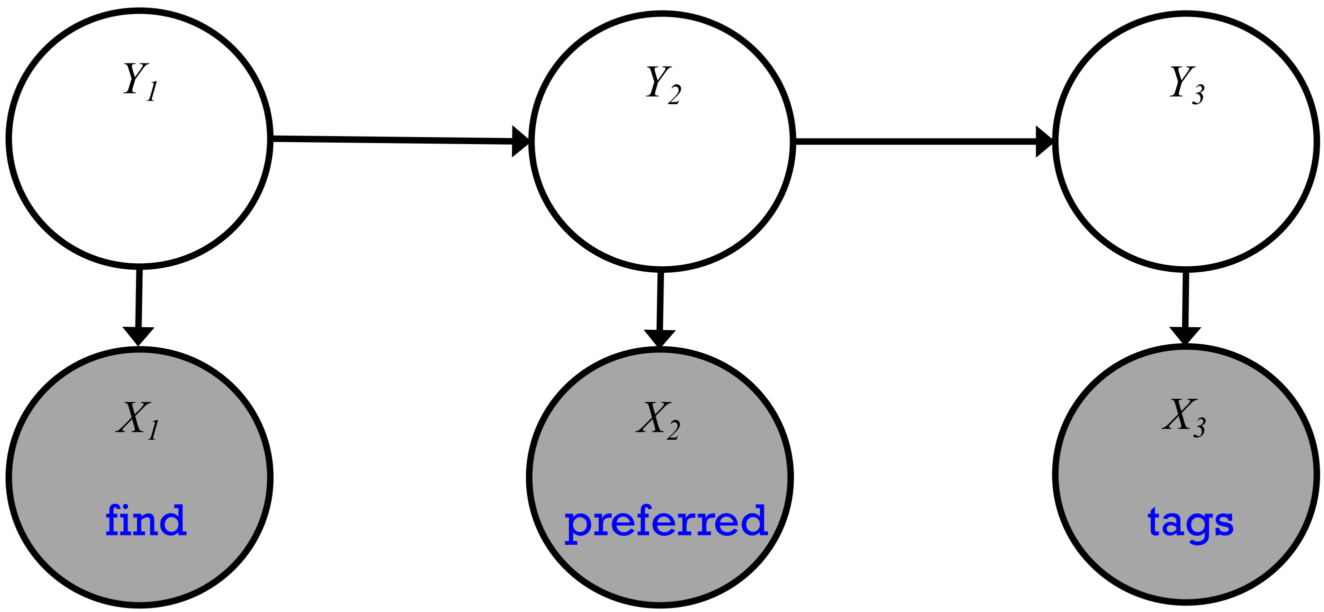
"belief that Y_2 = **a**"

sum = Z
(total weight
of *all* paths)

total weight of *all* paths through 

= $\alpha_2(\mathtt{a})$ A(pref., a) $\beta_2(\mathtt{a})$

Forward-Backward Algorithm



Could be verb or noun *Could be adjective or verb* *Could be noun or verb*

Forward-Backward Algorithm

Definitions

$\alpha_t(k) \triangleq p(x_1, \ldots, x_t, y_t = k)$

$\beta_t(k) \triangleq p(x_{t+1}, \ldots, x_T \mid y_t = k)$

Assume

$y_0 = \text{START}$

$y_{T+1} = \text{END}$

1. Initialize

$$\alpha_0(\text{START}) = 1 \qquad \alpha_0(k) = 0, \ \forall k \neq \text{START}$$

$$\beta_{T+1}(\text{END}) = 1 \qquad \beta_{T+1}(k) = 0, \ \forall k \neq \text{END}$$

2. Forward Algorithm

for $t = 1, \ldots, T+1$:

for $k = 1, \ldots, K$:

$$\alpha_t(k) = \sum_{j=1}^{K} p(x_t \mid y_t = k)\alpha_{t-1}(j)p(y_t = k \mid y_{t-1} = j)$$

3. Backward Algorithm

for $t = T, \ldots, 0$:

for $k = 1, \ldots, K$:

$$\beta_t(k) = \sum_{j=1}^{K} p(x_{t+1} \mid y_{t+1} = j)\beta_{t+1}(j)p(y_{t+1} = j \mid y_t = k)$$

4. Evaluation $p(\mathbf{x}) = \alpha_{T+1}(\text{END})$

5. Marginals $p(y_t = k \mid \mathbf{x}) = \frac{\alpha_t(k)\beta_t(k)}{p(\mathbf{x})}$

Forward-Backward Algorithm

Definitions

$\alpha_t(k) \triangleq p(x_1, \ldots, x_t, y_t = k)$

$\beta_t(k) \triangleq p(x_{t+1}, \ldots, x_T \mid y_t = k)$

Assume

$y_0 = \text{START}$

$y_{T+1} = \text{END}$

1. Initialize

$$\alpha_0(\text{START}) = 1 \qquad \alpha_0(k) = 0, \ \forall k \neq \text{START}$$

$$\beta_{T+1}(\text{END}) = 1 \qquad \beta_{T+1}(k) = 0, \ \forall k \neq \text{END}$$

2. Forward Algorithm

for $t = 1, \ldots, T+1$:

for $k = 1, \ldots, K$:

$$\alpha_t(k) = \sum_{j=1}^{K} p(x_t \mid y_t = k)\alpha_{t-1}(j)p(y_t = k \mid y_{t-1} = j)$$

O(K²T)

O(K)

kward Algorithm

for $t = T, \ldots, 0$:

or $k = 1, \ldots, K$:

$$\beta_t(k) = \sum_{j=1}^{K} p(x_{t+1} \mid y_{t+1} = j)\beta_{t+1}(j)p(y_{t+1} = j \mid y_t = k)$$

Brute force algorithm would be O(K^T)

4. Evaluation $p(\mathbf{x}) = \alpha_{T+1}(\text{END})$

5. Marginals $p(y_t = k \mid \mathbf{x}) = \frac{\alpha_t(k)\beta_t(k)}{p(\mathbf{x})}$

THE VITERBI ALGORITHM

Inference for HMMs

Whiteboard

– Viterbi algorithm
(edge weights version)

Viterbi Algorithm

Definitions

$$\omega_t(k) \triangleq \max_{y_1,\ldots,y_{t-1}} p(x_1,\ldots,x_t,y_1,\ldots,y_{t-1},y_t=k)$$

$$b_t(k) \triangleq \operatorname*{argmax}_{y_1,\ldots,y_{t-1}} p(x_1,\ldots,x_t,y_1,\ldots,y_{t-1},y_t=k)$$

Assume

$y_0 = \text{START}$

$y_{T+1} = \text{END}$

1. Initialize

$$\omega_0(\text{START}) = 1 \qquad\qquad \omega_0(k) = 0,\ \forall k \neq \text{START}$$

2. Viterbi Algorithm

for $t = 1,\ldots,T+1$:

for $k = 1,\ldots,K$:

$$\omega_t(k) = \max_{j\in\{1,\ldots,K\}} p(x_t \mid y_t = k)\,\omega_{t-1}(j)\,p(y_t = k \mid y_{t-1} = j)$$

$$b_t(k) = \operatorname*{argmax}_{j\in\{1,\ldots,K\}} p(x_t \mid y_t = k)\,\omega_{t-1}(j)\,p(y_t = k \mid y_{t-1} = j)$$

3. Compute Most Probable Assignment

$$\hat{y}_T = b_{T+1}(\text{END})$$

for $t = T,\ldots,1$:

$$\hat{y}_t = b_{t+1}(\hat{y}_{t+1})$$

Viterbi Algorithm

Definitions

$$\omega_t(k) \triangleq \max_{y_1,\ldots,y_{t-1}} p(x_1,\ldots,x_t,y_1,\ldots,y_{t-1},y_t=k)$$

$$b_t(k) \triangleq \operatorname*{argmax}_{y_1,\ldots,y_{t-1}} p(x_1,\ldots,x_t,y_1,\ldots,y_{t-1},y_t=k)$$

Assume

$y_0 =$ START

$y_{T+1} =$ END

1. Initialize

$$\omega_0(\text{START}) = 1 \qquad \omega_0(k) = 0,\ \forall k \neq \text{START}$$

2. Viterbi Algorithm

for $t = 1,\ldots,T+1$:

for $k = 1,\ldots,K$:

$$\omega_t(k) = \max_{j\in\{1,\ldots,K\}} p(x_t \mid y_t=k)\omega_{t-1}(j)p(y_t=k \mid y_{t-1}=j)$$

$$b_t(k) = \operatorname*{argmax}_{j\in\{1,\ldots,K\}} p(x_t \mid y_t=k)\omega_{t-1}(j)p(y_t=k \mid y_{t-1}=j)$$

$O(K^2T)$

3. Compute Most Probable Assignment

$\hat{y}_T = b_{T+1}(\text{END})$

for $t = T,\ldots,1$:

$\hat{y}_t = b_{t+1}(\hat{y}_{t+1})$

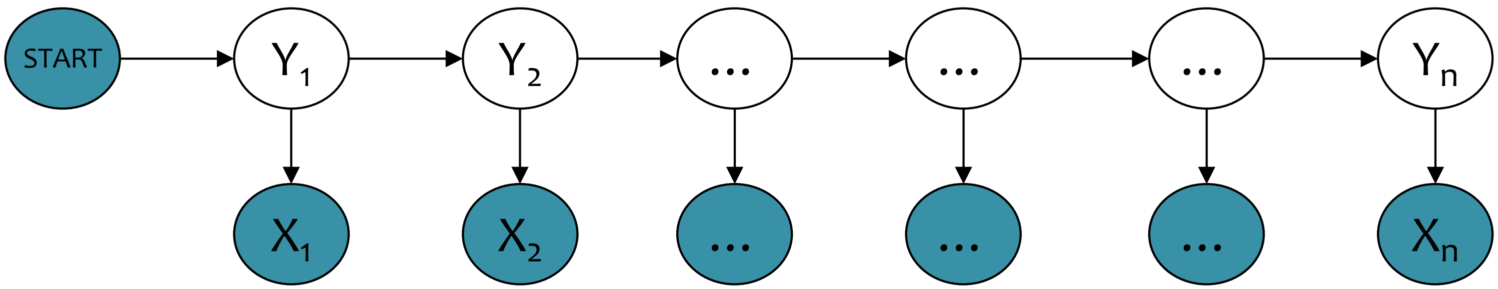
Brute force algorithm would be $O(K^T)$

Inference in HMMs

What is the **computational complexity** of inference for HMMs?

- The **naïve** (brute force) computations for *Evaluation*, *Decoding*, and *Marginals* take **exponential time**, $O(K^T)$
- The **forward-backward** algorithm and **Viterbi** algorithm run in **polynomial time**, $O(T*K^2)$
  - Thanks to dynamic programming!

Shortcomings of Hidden Markov Models



- HMM models capture dependences between each state and only its corresponding observation
  - NLP example: In a sentence segmentation task, each segmental state may depend not just on a single word (and the adjacent segmental stages), but also on the (non-local) features of the whole line such as line length, indentation, amount of white space, etc.
- Mismatch between learning objective function and prediction objective function
  - HMM learns a joint distribution of states and observations P(**Y**, **X**), but in a prediction task, we need the conditional probability P(**Y**|**X**)

FORWARD-BACKWARD IN LOG SPACE

Forward-Backward Algorithm

De

α_t

β_t

As

y_0

y_T

Problem:
Implementing F-B as shown here could run into **underflow** (i.e. floating point precision issues).

Why?
Because the algorithm is still multiplying O(T) probabilities together. Each probability is in [0,1] and so their product can get very small.

One solution:
work in log-space!

1. Initialize

$$\alpha_0(\text{START}) = 1 \qquad \alpha_0(k) = 0,\ \forall k \neq \text{START}$$

$$\beta_{T+1}(\text{END}) = 1 \qquad \beta_{T+1}(k) = 0,\ \forall k \neq \text{END}$$

Forward Algorithm

for $t = 1, \ldots, T+1$:

for $k = 1, \ldots, K$:

$$\alpha_t(k) = \sum_{j=1}^{K} \boxed{p(x_t \mid y_t = k)\alpha_{t-1}(j)p(y_t = k \mid y_{t-1} = j)}$$

Backward Algorithm

for $t = T, \ldots, 0$:

for $k = 1, \ldots, K$:

$$\beta_t(k) = \sum_{j=1}^{K} \boxed{p(x_{t+1} \mid y_{t+1} = j)\beta_{t+1}(j)p(y_{t+1} = j \mid y_t = k)}$$

4. Evaluation $p(\mathbf{x}) = \alpha_{T+1}(\text{END})$

5. Marginals $p(y_t = k \mid \mathbf{x}) = \frac{\alpha_t(k)\beta_t(k)}{p(\mathbf{x})}$

Log-space Arithmetic

Log-space Multiplication

- Suppose you wish to multiply two probabilities p_a and p_b together to get $p_c = p_a\ p_b$
- Yet, you want to represent all those numbers as the log of their value:
  - $o_a = \log(p_a)$
  - $o_b = \log(p_b)$
  - $o_c = \log(p_c)$
- To compute o_c from o_a and o_b we simply add them:

  $$\begin{aligned} o_c &= o_a + o_b \\ &= \log(p_a) + \log(p_b) \\ &= \log(p_a\ p_b) \\ &= \log(p_c) \end{aligned}$$

Log-space Addition

- Suppose you wish to add two probabilities p_a and p_b together to get $p_d = p_a + p_b$, yet all in log-space (e.g. $o_d = \log(p_d)$)
- To compute compute o_d from o_a and o_b we must be more careful:

  $$\begin{aligned} o_d &= \text{log-sum-exp}(o_a, o_b) \\ &= \log(\exp(o_a) + \exp(o_b)) \end{aligned}$$

- **Problem:** if we merely implement log-sum-exp as above, we'll probably run into underflow again b/c:
  - $p_a = \exp(o_a)$
  - $p_b = \exp(o_b)$

Log-space Arithmetic

A careful implementation:

$$
\begin{aligned}
&\textbf{def } \log\!-\!\mathrm{sum}\!-\!\exp(x_1, \ldots, x_N): \\
&\qquad c = \max(x_1, \ldots, x_N) \\
&\qquad y = c + \log \textstyle\sum_{n=1}^{N} \exp(x_n - c) \\
&\qquad \textbf{return } y
\end{aligned}
$$

Why does this work?

$$y = \log \sum_{n=1}^{N} \exp(x_n)$$

$$\Rightarrow \exp(y) = \sum_{n=1}^{N} \exp(x_n)$$

$$\Rightarrow \exp(y) = \frac{\exp(c)}{\exp(c)} \sum_{n=1}^{N} \exp(x_n)$$

$$\Rightarrow \exp(y) = \exp(c) \sum_{n=1}^{N} \exp(x_n - c)$$

$$\Rightarrow y = c + \log \sum_{n=1}^{N} \exp(x_n - c)$$

Log-space Addition

- Suppose you wish to add two probabilities p_a and p_b together to get $p_d = p_a + p_b$, yet all in log-space (e.g. $o_d = \log(p_d)$)
- To compute compute o_d from o_a and o_b we must be more careful:

  o_d = log-sum-exp(o_a, o_b)
  = log(exp(o_a) + exp(o_b))

- **Problem:** if we merely implement log-sum-exp as above, we'll probably run into underflow again b/c:
  - $p_a = \exp(o_a)$
  - $p_b = \exp(o_b)$

Forward Algorithm (in log-space)

We can run the forward algorithm in log-space using log-multiplication and log-addition. The backward algorithm is analogous.

Definitions

$\log \alpha_t(k) \triangleq \log p(x_1, \ldots, x_t, y_t = k)$

Assume

$y_0 =$ START

1. Initialize

$$\log \alpha_0(\text{START}) = 0 \qquad \log \alpha_0(k) = -\infty, \; \forall k \neq \text{START}$$

2. Forward Algorithm

for $t = 1, \ldots, T+1$:
  for $k = 1, \ldots, K$:
    for $j = 1, \ldots, K$:
      $o_j = \log p(x_t \mid y_t = k) + \log \alpha_{t-1}(j) + \log p(y_t = k \mid y_{t-1} = j)$
    $\log \alpha_t(k) = \text{log-sum-exp}(o_1, \ldots, o_K)$

3. Evaluation $\log p(\mathbf{x}) = \log \alpha_{T+1}(\text{END})$

MBR DECODING

Inference for HMMs

Four

– ~~Three~~ Inference Problems for an HMM

1. Evaluation: Compute the probability of a given sequence of observations
2. Viterbi Decoding: Find the most-likely sequence of hidden states, given a sequence of observations
3. Marginals: Compute the marginal distribution for a hidden state, given a sequence of observations
4. MBR Decoding: Find the lowest loss sequence of hidden states, given a sequence of observations (Viterbi decoding is a special case)

Minimum Bayes Risk Decoding

- Suppose we given a loss function $l(\boldsymbol{y}', \boldsymbol{y})$ and are asked for a single tagging
- How should we choose just one from our probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$?
- A minimum Bayes risk (MBR) decoder $h(\boldsymbol{x})$ returns the variable assignment with minimum **expected** loss under the model's distribution

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underset{\hat{\boldsymbol{y}}}{\operatorname{argmin}} \ \mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}[\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})]$$

$$= \underset{\hat{\boldsymbol{y}}}{\operatorname{argmin}} \ \sum_{\boldsymbol{y}} p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \ell(\hat{\boldsymbol{y}}, \boldsymbol{y})$$

Minimum Bayes Risk Decoding

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underset{\hat{\boldsymbol{y}}}{\text{argmin}} \; \mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}[\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})]$$

Consider some example loss functions:

The ***0-1* loss function** returns *0* only if the two assignments are identical and *1* otherwise:

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = 1 - \mathbb{I}(\hat{\boldsymbol{y}}, \boldsymbol{y})$$

The MBR decoder is:

$$\begin{aligned} h_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \underset{\hat{\boldsymbol{y}}}{\text{argmin}} \; \sum_{\boldsymbol{y}} p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})(1 - \mathbb{I}(\hat{\boldsymbol{y}}, \boldsymbol{y})) \\ &= \underset{\hat{\boldsymbol{y}}}{\text{argmax}} \; p_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}} \mid \boldsymbol{x}) \end{aligned}$$

which is exactly the Viterbi decoding problem!

Minimum Bayes Risk Decoding

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underset{\hat{\boldsymbol{y}}}{\operatorname{argmin}} \; \mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}[\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})]$$

Consider some example loss functions:

The **Hamming loss** corresponds to accuracy and returns the number of incorrect variable assignments:

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{i=1}^{V} (1 - \mathbb{I}(\hat{y}_i, y_i))$$

The MBR decoder is:

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\boldsymbol{x})_i = \underset{\hat{y}_i}{\operatorname{argmax}} \; p_{\boldsymbol{\theta}}(\hat{y}_i \mid \boldsymbol{x})$$

This decomposes across variables and requires the variable marginals.

TO HMMS AND BEYOND…

Unsupervised Learning for HMMs

- Unlike **discriminative** models p(y|x), **generative** models p(x,y) can maximize the likelihood of the data D = {$x^{(1)}$, $x^{(2)}$, …, $x^{(N)}$} where we don't observe any y's.
- This **unsupervised learning** setting can be achieved by finding parameters that maximize the **marginal likelihood**
- We optimize using the **Expectation-Maximization** algorithm

Since we don't observe $\mathbf{y}$, we define the marginal probability:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \quad (1)$$

The log-likelihood of the data is thus:

$$\ell(\boldsymbol{\theta}) = \log \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$$

$$= \sum_{i=1}^{N} \log \sum_{\mathbf{y} \in \mathcal{Y}} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{y}) \quad (3)$$

Beyond the scope of today's lecture!

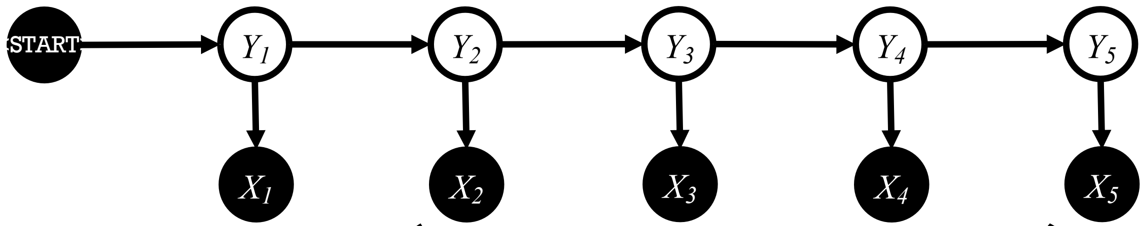
HMMs: History

- Markov chains: Andrey Markov (1906)
  - Random walks and Brownian motion
- Used in Shannon’s work on information theory (1948)
- Baum-Welsh learning algorithm: late 60’s, early 70’s.
  - Used mainly for speech in 60s-70s.
- Late 80’s and 90’s: David Haussler (major player in learning theory in 80’s) began to use HMMs for modeling biological sequences
- Mid-late 1990’s: Dayne Freitag/Andrew McCallum
  - Freitag thesis with Tom Mitchell on IE from Web using logic programs, grammar induction, etc.
  - McCallum: multinomial Naïve Bayes for text
  - With McCallum, IE using HMMs on CORA
- …

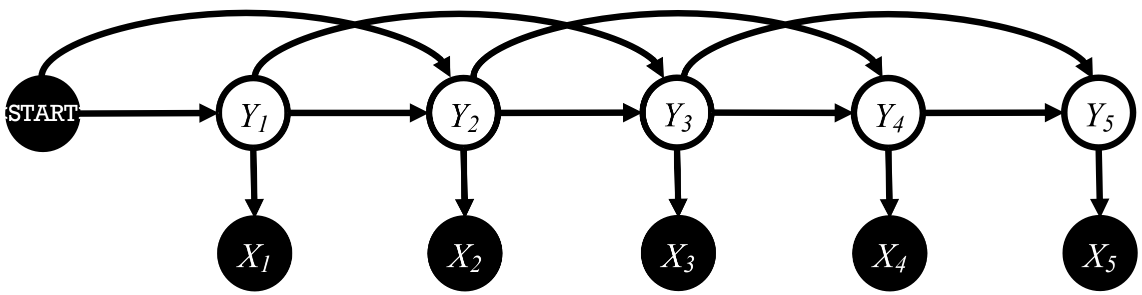
Slide from William Cohen

Higher-order HMMs

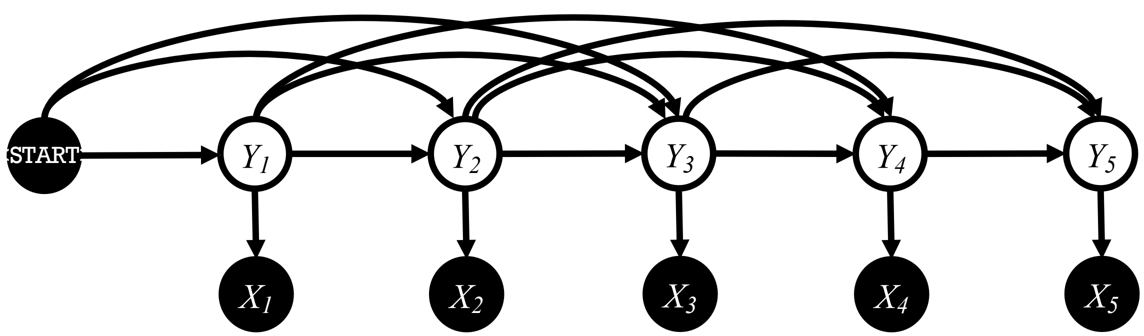
- 1st-order HMM (i.e. bigram HMM)



- 2nd-order HMM (i.e. trigram HMM)

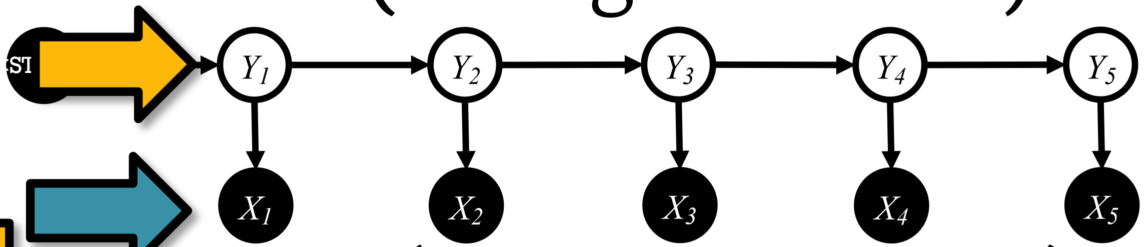


- 3rd-order HMM



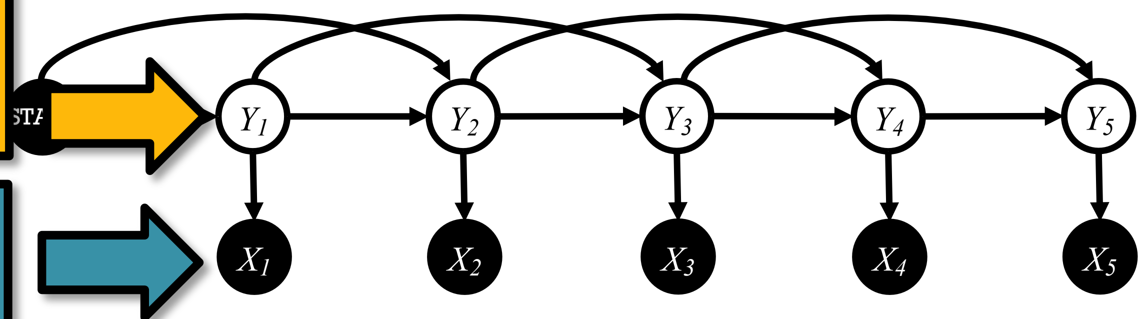
Higher-order HMMs

- 1^{st}-order HMM (i.e. bigram HMM)



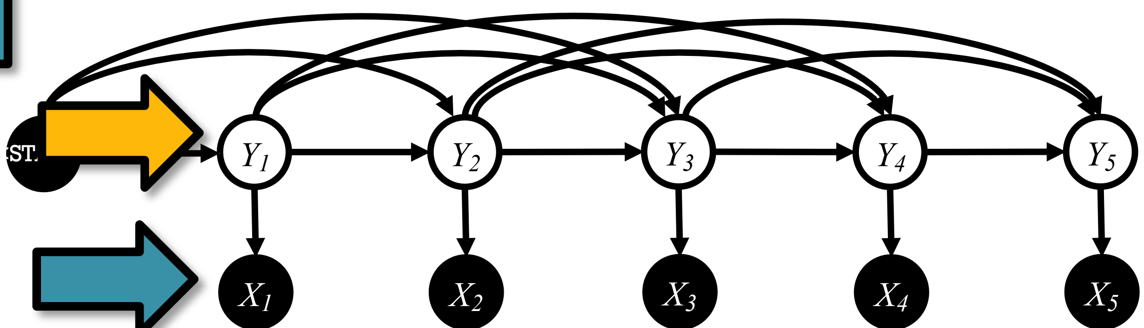
Hidden States, **y**

rder HMM (i.e. trigram HMM)



Observa-tions, **x**

rder HMM



Learning Objectives

Hidden Markov Models

You should be able to…

1. Show that structured prediction problems yield high-computation inference problems
2. Define the first order Markov assumption
3. Draw a Finite State Machine depicting a first order Markov assumption
4. Derive the MLE parameters of an HMM
5. Define the three key problems for an HMM: evaluation, decoding, and marginal computation
6. Derive a dynamic programming algorithm for computing the marginal probabilities of an HMM
7. Interpret the forward-backward algorithm as a message passing algorithm
8. Implement supervised learning for an HMM
9. Implement the forward-backward algorithm for an HMM
10. Implement the Viterbi algorithm for an HMM
11. Implement a minimum Bayes risk decoder with Hamming loss for an HMM

Bayesian Networks

DIRECTED GRAPHICAL MODELS

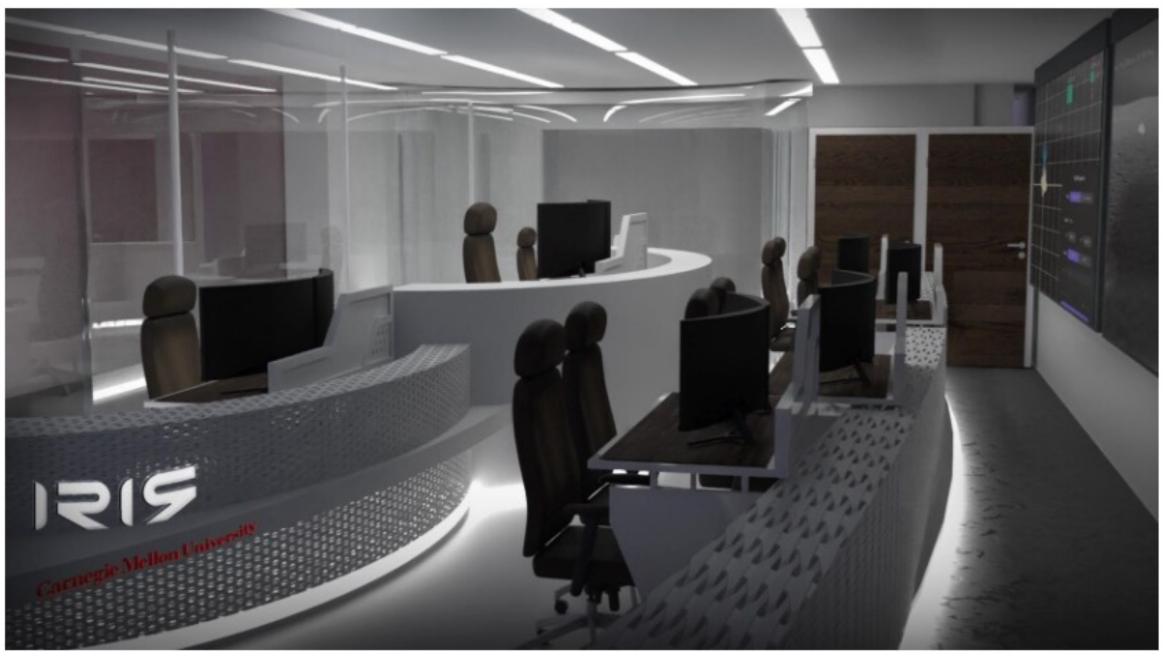
Example: CMU Mission Control

90.5 WESA
Pittsburgh's NPR News Station

WESA
Morning Edition

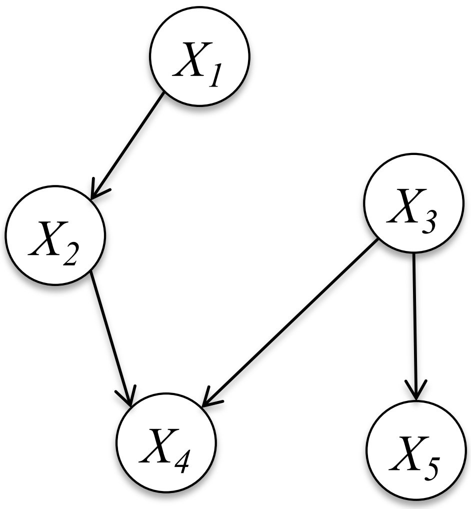
Pittsburgh's first mission control center to land at CMU ahead of 2022 lunar rover launch

90.5 WESA | By Kiley Koscinski
Published March 29, 2022 at 4:44 PM EDT



Courtesy Of Carnegie Mellon University

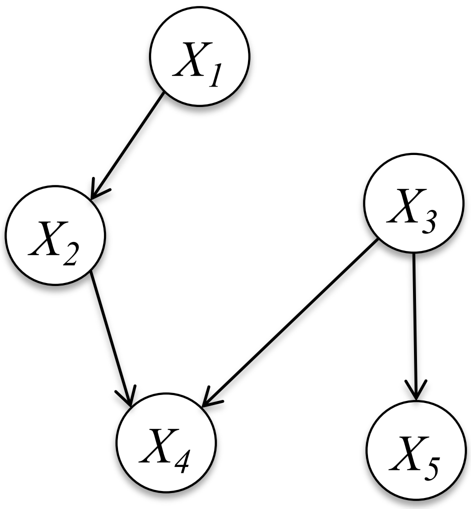
Bayesian Network



$$
\begin{aligned}
p(X_1, X_2, X_3, X_4, X_5) = \\
p(X_5|X_3)p(X_4|X_2, X_3) \\
p(X_3)p(X_2|X_1)p(X_1)
\end{aligned}
$$

Bayesian Network

Definition:



$$P(X_1, \ldots, X_T) = \prod_{t=1}^{T} P(X_t \mid \text{parents}(X_t))$$

- A Bayesian Network is a **directed graphical model**
- It consists of a graph **G** and the conditional probabilities **P**
- These two parts full specify the distribution:
  - Qualitative Specification: **G**
  - Quantitative Specification: **P**

Qualitative Specification

- Where does the qualitative specification come from?
  - Prior knowledge of causal relationships
  - Prior knowledge of modular relationships
  - Assessment from experts
  - Learning from data (i.e. structure learning)
  - We simply prefer a certain architecture (e.g. a layered graph)
  - …

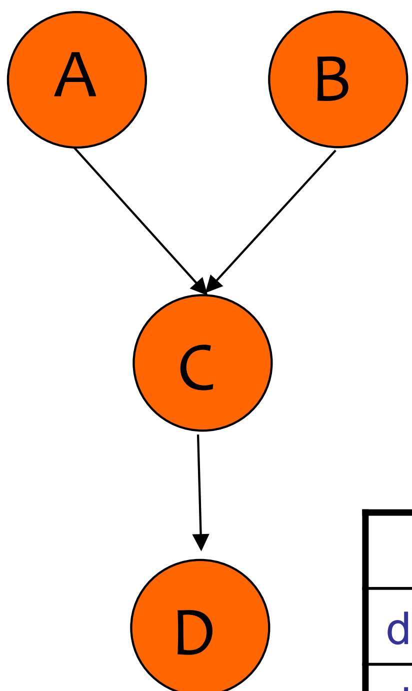
Quantitative Specification

Example: Conditional probability tables (CPTs) for discrete random variables

a^0	0.75
a^1	0.25

b^0	0.33
b^1	0.67

$$P(a,b,c.d) = P(a)P(b)P(c|a,b)P(d|c)$$



	a^0b^0	a^0b^1	a^1b^0	a^1b^1
c^0	0.45	1	0.9	0.7
c^1	0.55	0	0.1	0.3

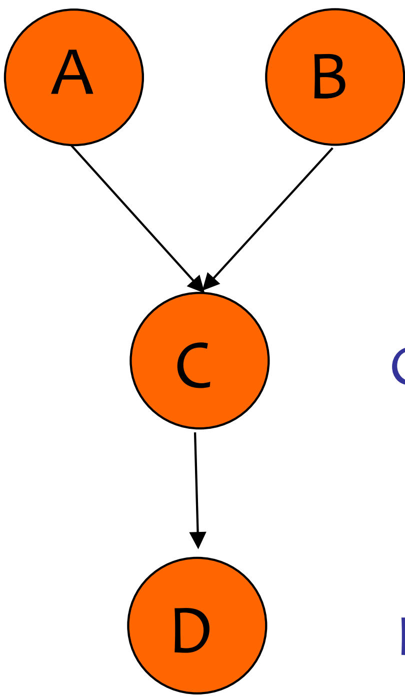
	c^0	c^1
d^0	0.3	0.5
d^1	07	0.5

Quantitative Specification

Example: Conditional probability density functions (CPDs) for continuous random variables

$A \sim N(\mu_a, \Sigma_a)$ $B \sim N(\mu_b, \Sigma_b)$

$$\boxed{\begin{gathered} P(a,b,c.d) = \\ P(a)P(b)P(c|a,b)P(d|c) \end{gathered}}$$



$C \sim N(A+B, \Sigma_c)$

$D \sim N(\mu_d+C, \Sigma_d)$

$P(D|C)$

D

C

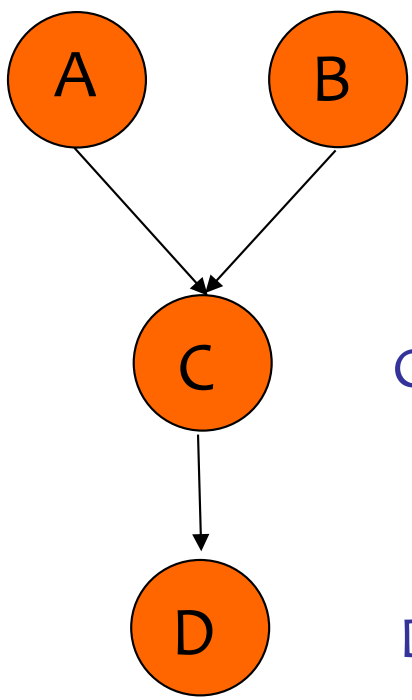
Quantitative Specification

Example: Combination of CPTs and CPDs for a mix of discrete and continuous variables

a^0	0.75
a^1	0.25

b^0	0.33
b^1	0.67

$$P(a,b,c.d) = P(a)P(b)P(c|a,b)P(d|c)$$



$C \sim N(A+B, \Sigma_c)$

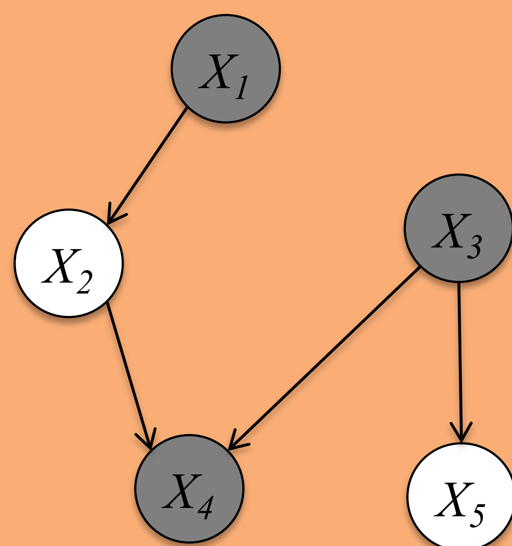
$D \sim N(\mu_d + C, \Sigma_d)$

Observed Variables

- In a graphical model, **shaded nodes** are "**observed**", i.e. their values are given

Example:

$$P(X_2, X_5 \mid X_1 = 0, X_3 = 1, X_4 = 1)$$



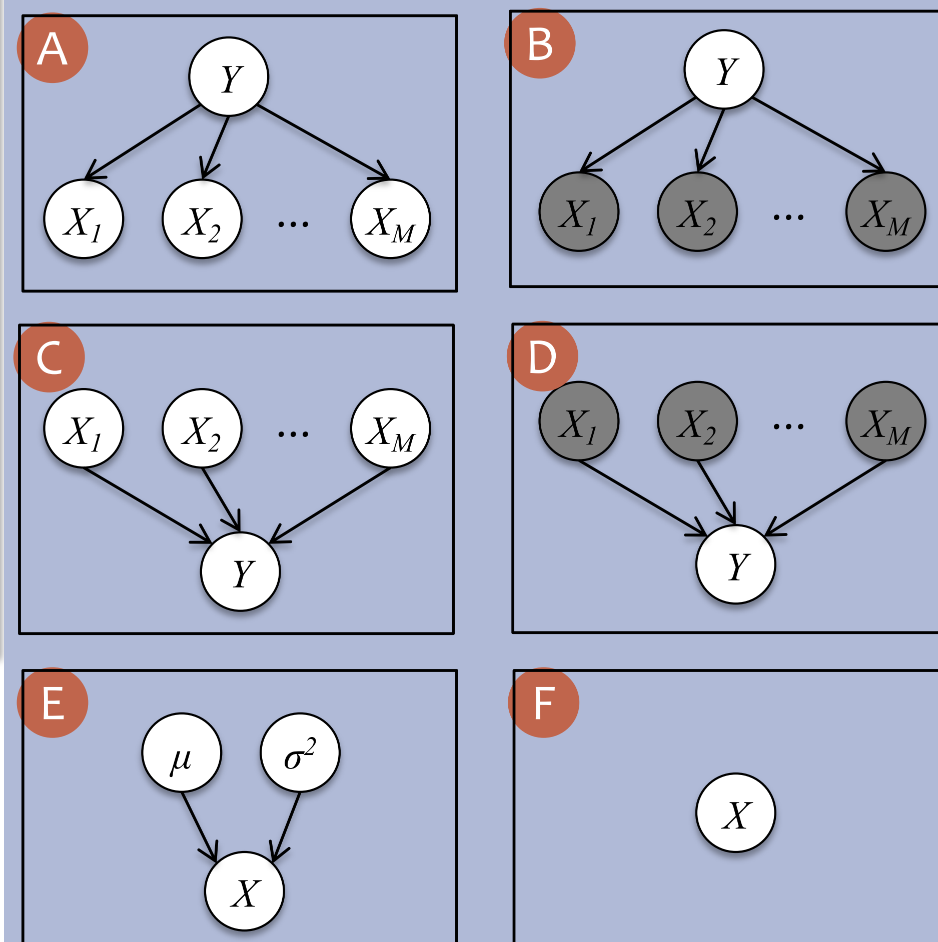
Familiar Models as Bayesian Networks

Question:

Match the model name to the corresponding Bayesian Network

1. Logistic Regression
2. Linear Regression
3. Bernoulli Naïve Bayes
4. Gaussian Naïve Bayes
5. 1D Gaussian

Answer:



GRAPHICAL MODELS: DETERMINING CONDITIONAL INDEPENDENCIES

What Independencies does a Bayes Net Model?

- In order for a Bayesian network to model a probability distribution, the following must be true:

  Each variable is conditionally independent of all its non-descendants in the graph given the value of all its parents.

- This follows from $P(X_1, \ldots, X_T) = \prod_{t=1}^{T} P(X_t \mid \text{parents}(X_t))$

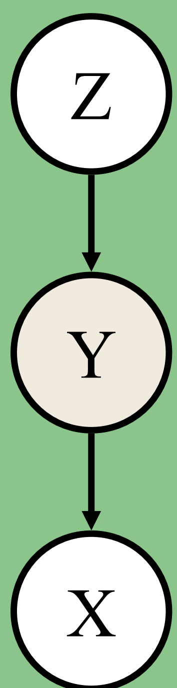
$$= \prod_{t=1}^{T} P(X_t \mid X_1, \ldots, X_{t-1})$$

- But what else does it imply?

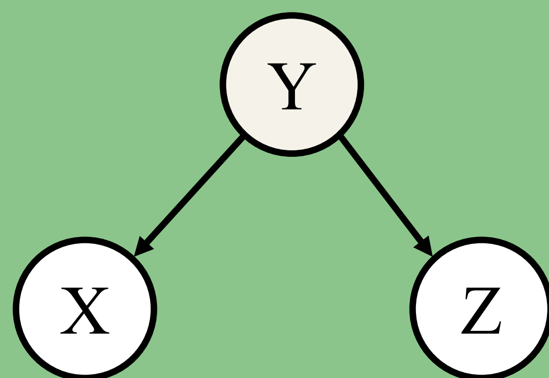
What Independencies does a Bayes Net Model?

Three cases of interest…

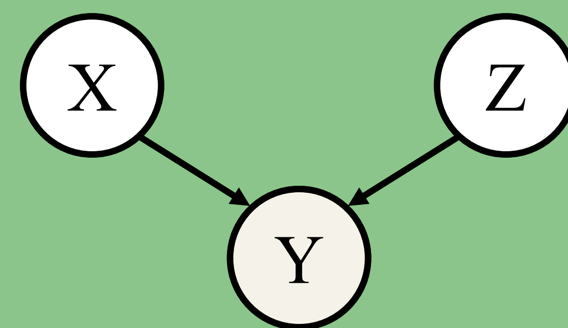
Cascade



Common Parent



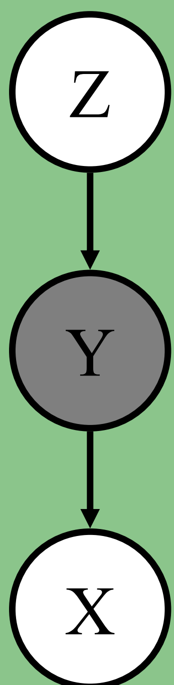
V-Structure



What Independencies does a Bayes Net Model?

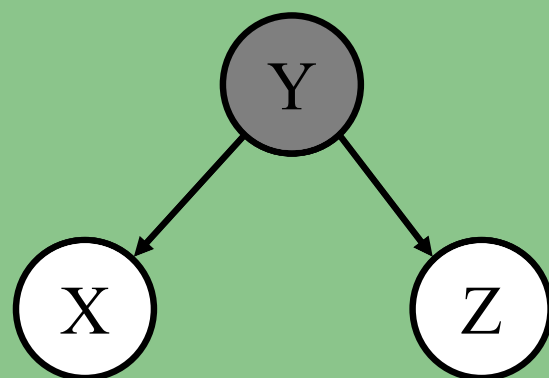
Three cases of interest…

Cascade



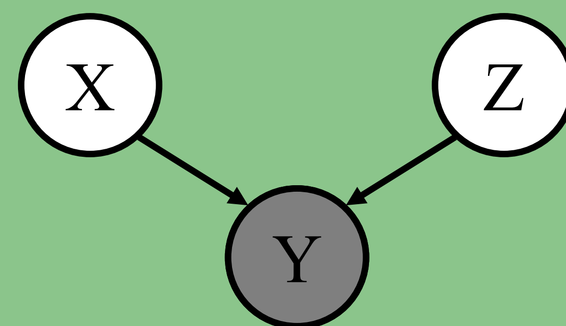
$$X \perp\!\!\!\perp Z \mid Y$$

Common Parent



$$X \perp\!\!\!\perp Z \mid Y$$

V-Structure



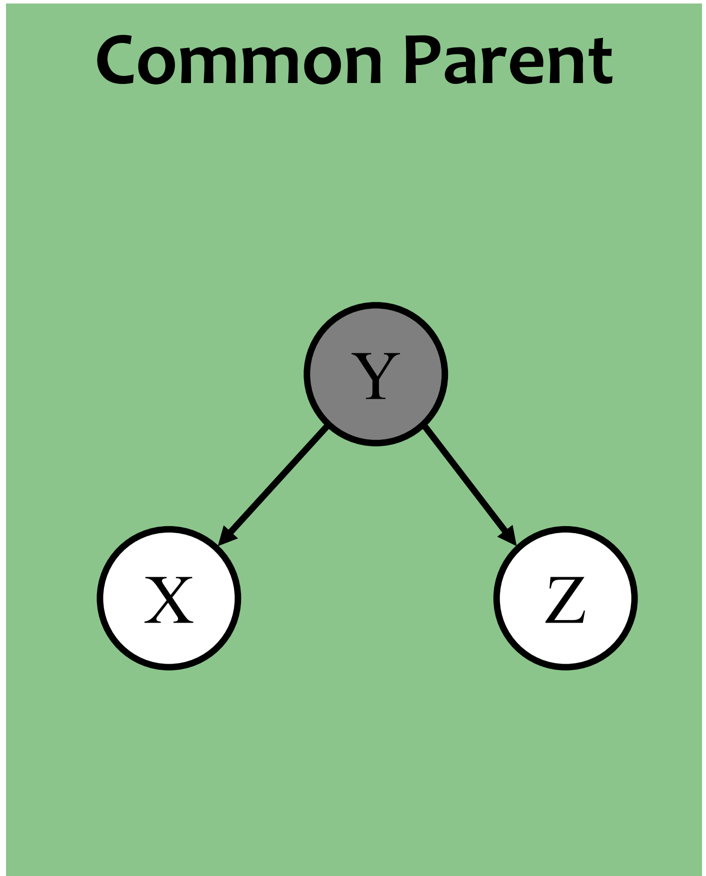
$$X \not\perp\!\!\!\perp Z \mid Y$$

Knowing Y
decouples X and Z

Knowing Y
couples X and Z

Whiteboard

Proof of conditional independence

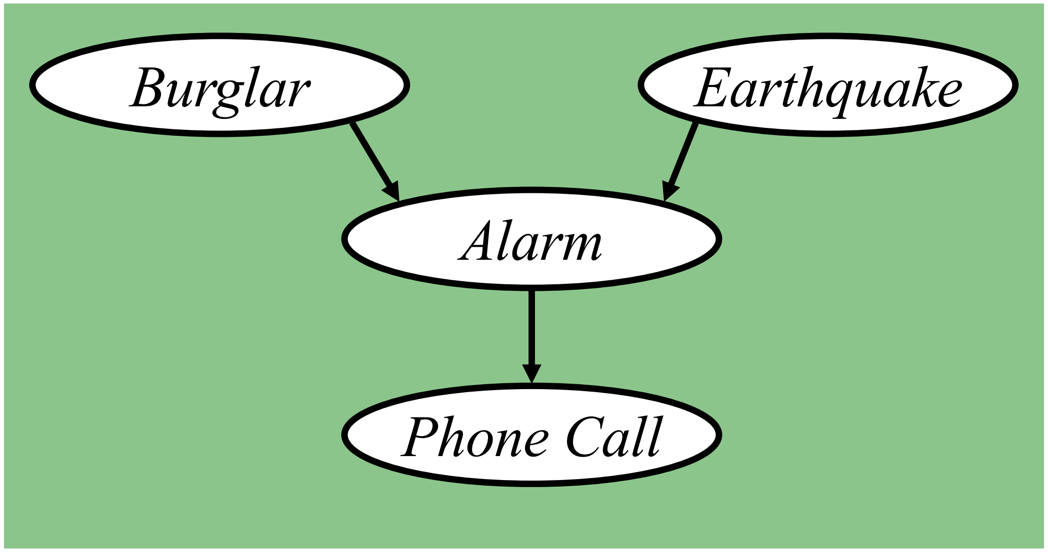


$$X \perp\!\!\!\perp Z \mid Y$$

(The other two cases can be shown just as easily.)

The "Burglar Alarm" example

- Your house has a twitchy burglar alarm that is also sometimes triggered by earthquakes.
- Earth arguably doesn't care whether your house is currently being burgled
- While you are on vacation, one of your neighbors calls and tells you your home's burglar alarm is ringing. Uh oh!



Quiz: True or False?

$$Burglar \perp\!\!\!\perp Earthquake \mid PhoneCall$$

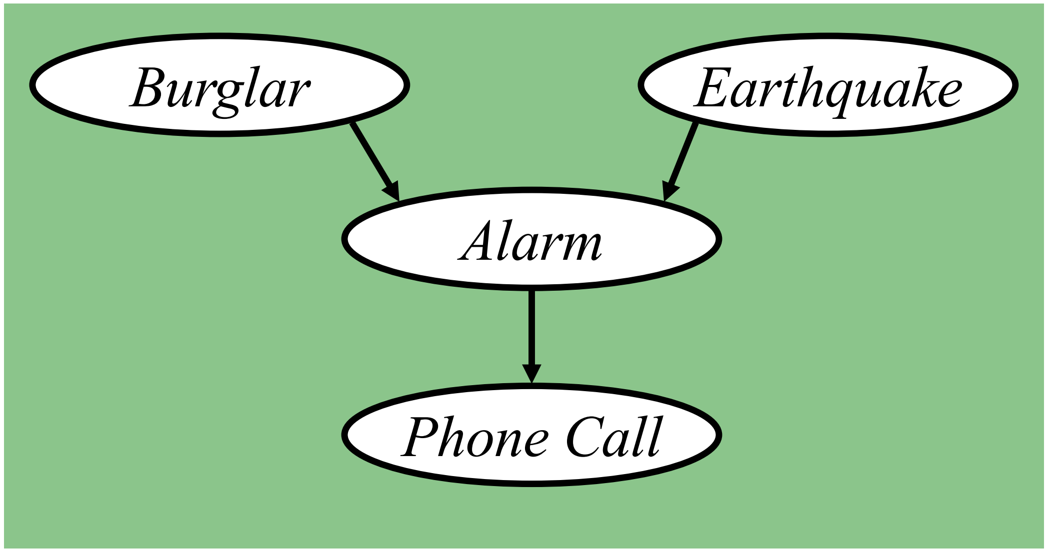
The "Burglar Alarm" example

- After you get this phone call, suppose you learn that there was a medium-sized earthquake in your neighborhood. Oh, whew! Probably not a burglar after all.
- Earthquake "explains away" the hypothetical burglar.
- But then it must **not** be the case that

$$Burglar \perp\!\!\!\perp Earthquake \mid PhoneCall$$

even though

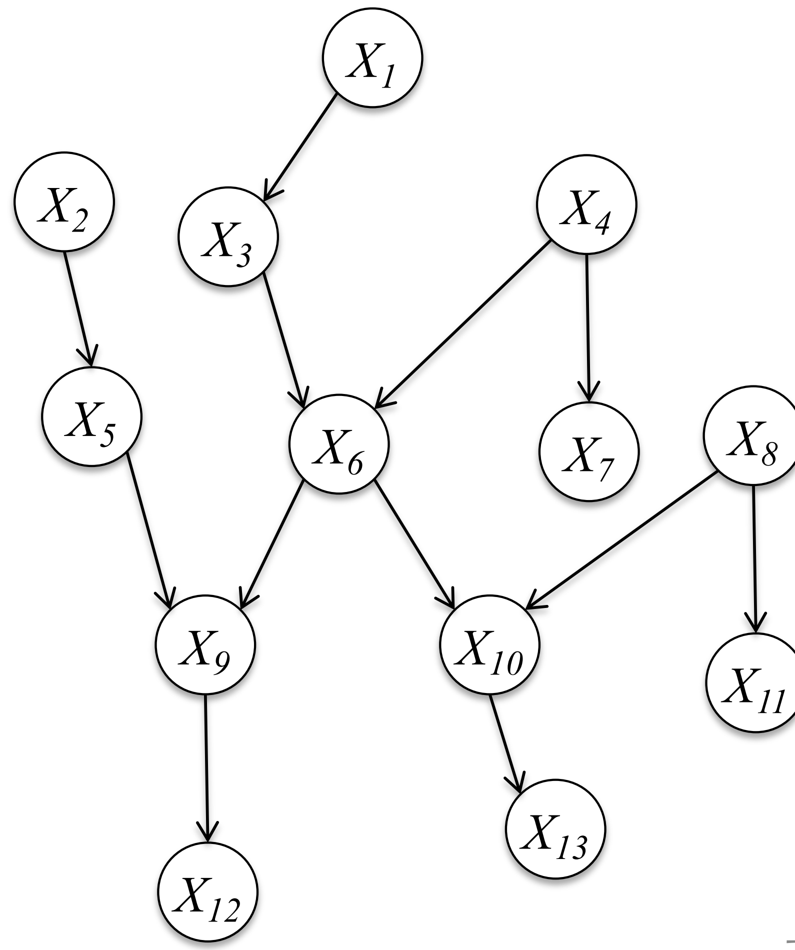
$$Burglar \perp\!\!\!\perp Earthquake$$



Markov Boundary

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov boundary** of a node is the set containing the node's parents, children, and co-parents.

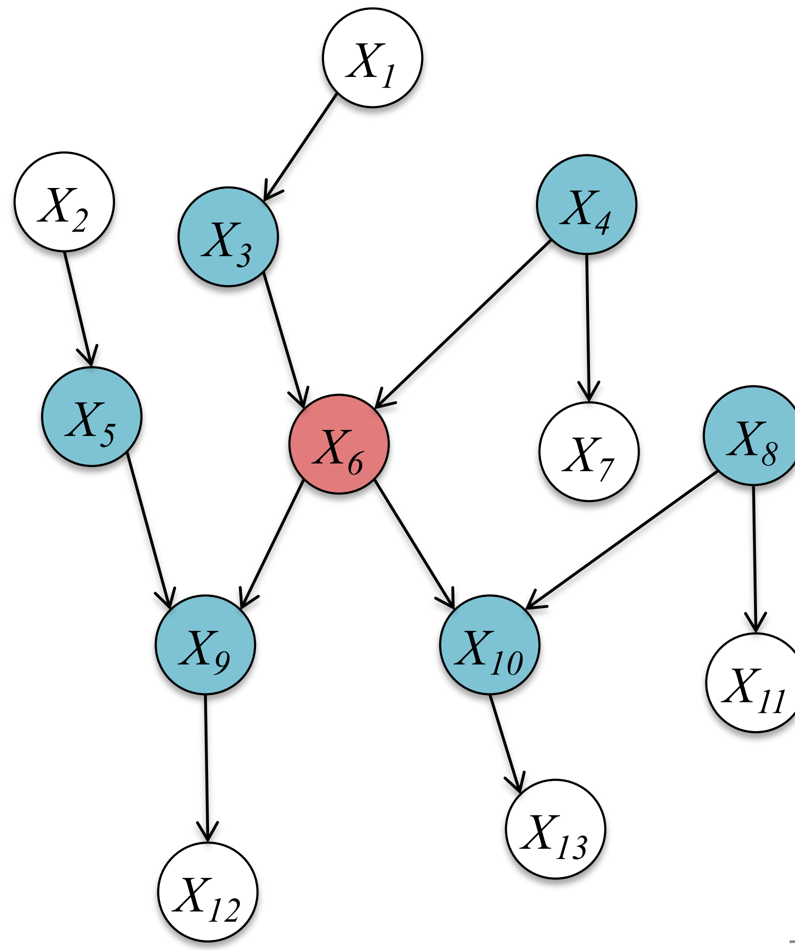


Markov Boundary

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov boundary** of a node is the set containing the node's parents, children, and co-parents.

Example: The Markov boundary of X_6 is $\{X_3, X_4, X_5, X_8, X_9, X_{10}\}$



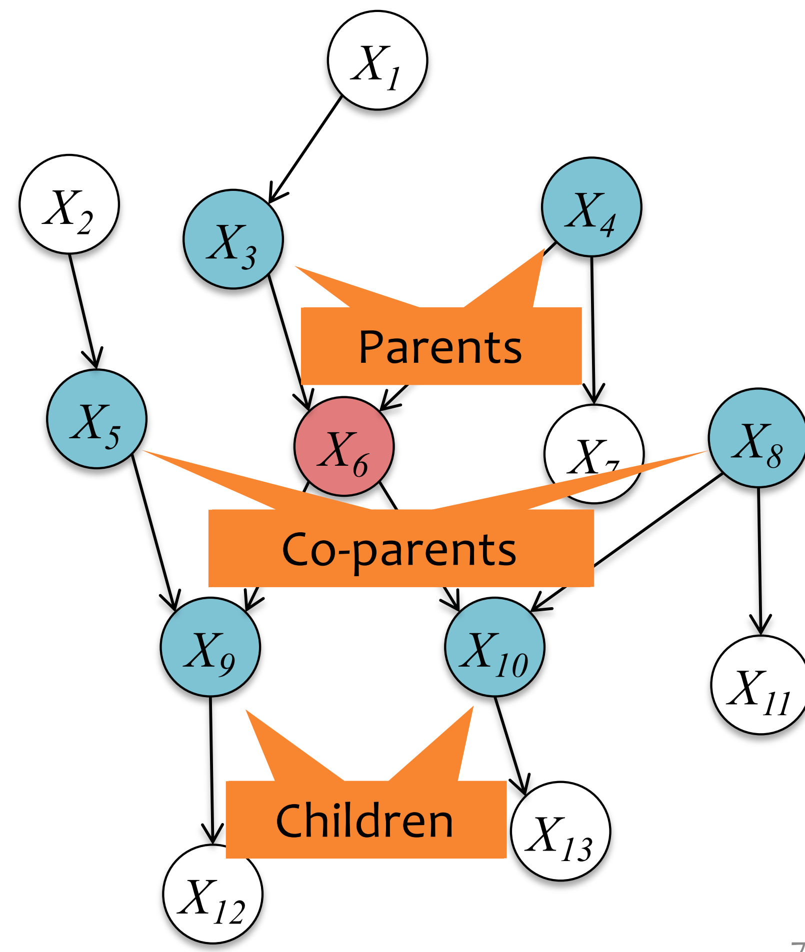
Markov Boundary

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov boundary** of a node is the set containing the node's parents, children, and co-parents.

Theorem: a node is **conditionally independent** of every other node in the graph given its **Markov boundary**

Example: The Markov boundary of X_6 is $\{X_3, X_4, X_5, X_8, X_9, X_{10}\}$

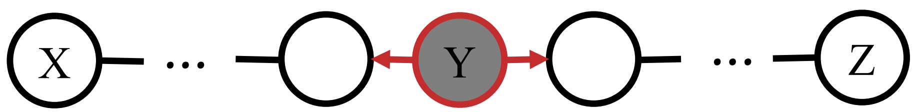


D-Separation

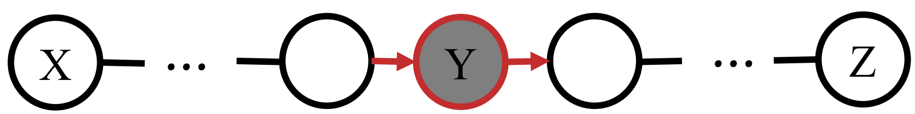
<u>Definition #1:</u>
Variables X and Z are **d-separated** given a **set** of evidence variables E (variables that are observed) iff every path from X to Z is “blocked”.

A path is “blocked” whenever:

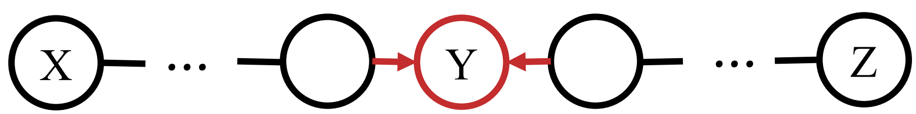
1. ∃Y on path s.t. Y ∈ E and Y is a “common parent”



2. ∃Y on path s.t. Y ∈ E and Y is in a “cascade”



3. ∃Y on path s.t. {Y, descendants(Y)} ∉ E and Y is in a “v-structure”



If variables X and Z are **d-separated** given a **set** of variables E
Then X and Z are **conditionally independent** given the **set** E

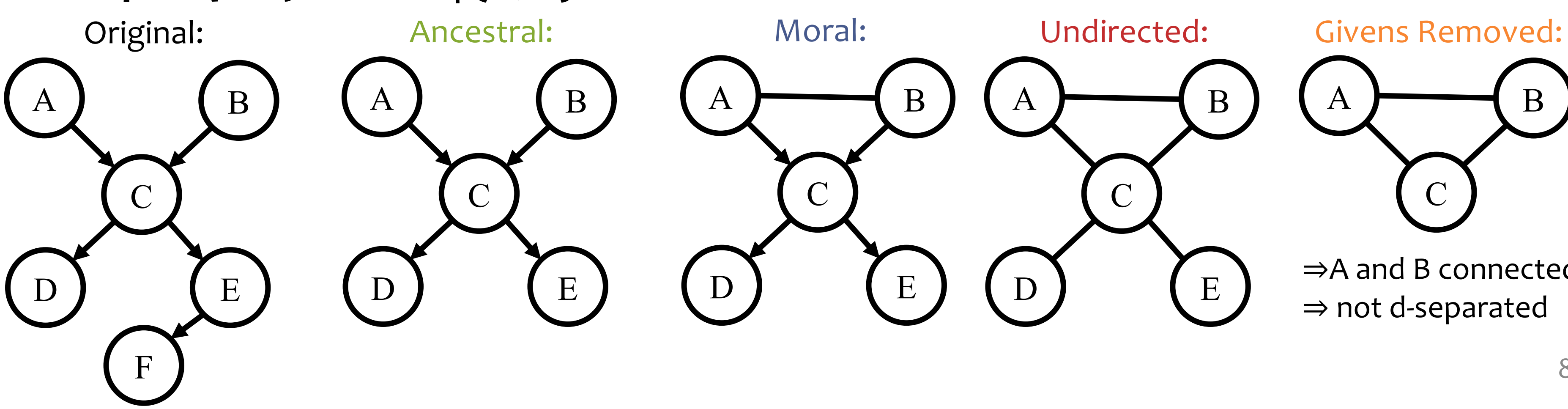
D-Separation

If variables X and Z are **d-separated** given a **set** of variables E
Then X and Z are **conditionally independent** given the **set** E

Definition #2:
Variables X and Z are **d-separated** given a **set** of evidence variables E iff there does **not** exist a path between X and Z in the **undirected ancestral moral** graph **with E removed**.

1. **Ancestral graph**: keep only X, Z, E and their ancestors
2. **Moral graph**: add undirected edge between all pairs of each node's parents
3. **Undirected graph**: convert all directed edges to undirected
4. Givens Removed: delete any nodes in E

Example Query: $A \perp\!\!\!\perp B \mid \{D, E\}$



⇒A and B connected
⇒ not d-separated

SUPERVISED LEARNING FOR BAYES NETS

Recipe for Closed-form MLE

1. Assume data was generated i.i.d. from some model (i.e. write the generative story)

   $x^{(i)} \sim p(x|\boldsymbol{\theta})$

2. Write log-likelihood

   $\ell(\boldsymbol{\theta}) = \log p(x^{(1)}|\boldsymbol{\theta}) + \ldots + \log p(x^{(N)}|\boldsymbol{\theta})$

3. Compute partial derivatives (i.e. gradient)

   $\partial\ell(\boldsymbol{\theta})/\partial\theta_1 = \ldots$

   $\partial\ell(\boldsymbol{\theta})/\partial\theta_2 = \ldots$

   $\ldots$

   $\partial\ell(\boldsymbol{\theta})/\partial\theta_M = \ldots$

4. Set derivatives to zero and solve for $\boldsymbol{\theta}$

   $\partial\ell(\boldsymbol{\theta})/\partial\theta_m = 0$ for all $m \in \{1, \ldots, M\}$

   $\boldsymbol{\theta}^{MLE}$ = solution to system of M equations and M variables

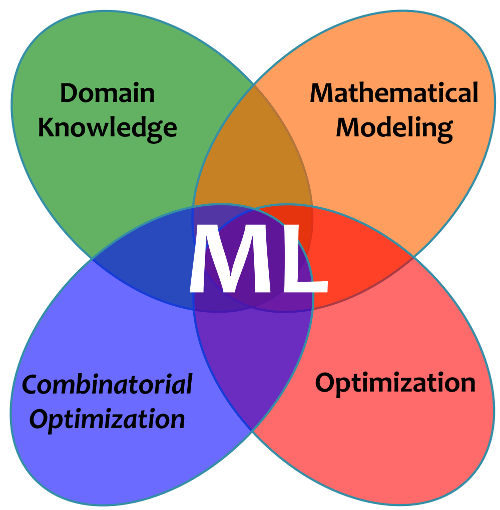
5. Compute the second derivative and check that $\ell(\boldsymbol{\theta})$ is concave down at $\boldsymbol{\theta}^{MLE}$

Machine Learning

The **data** inspires the structures we want to predict

Our **model** defines a score for each structure

It also tells us what to optimize



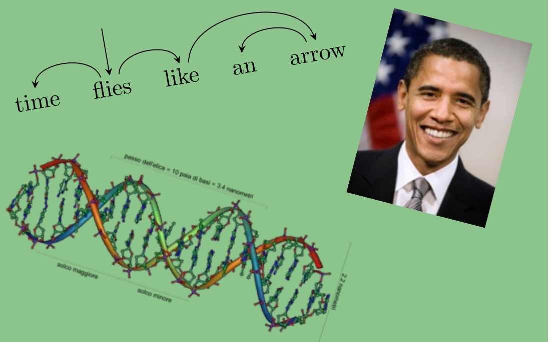
Inference finds {best structure, marginals, partition function} for a new observation

(**Inference** is usually called as a subroutine in learning)

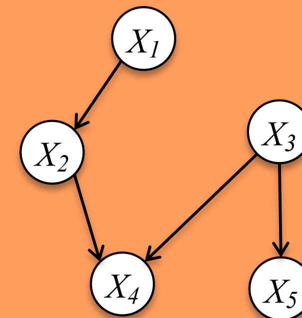
Learning tunes the parameters of the model

Machine Learning

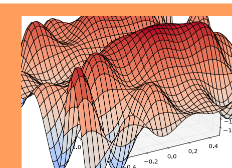
Data



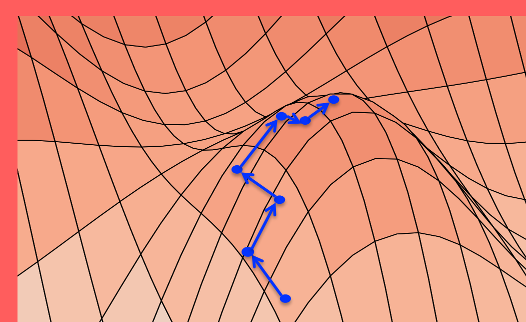
Model



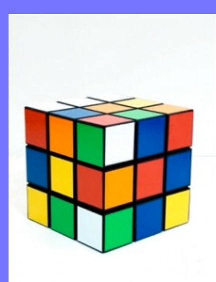
Objective



Learning

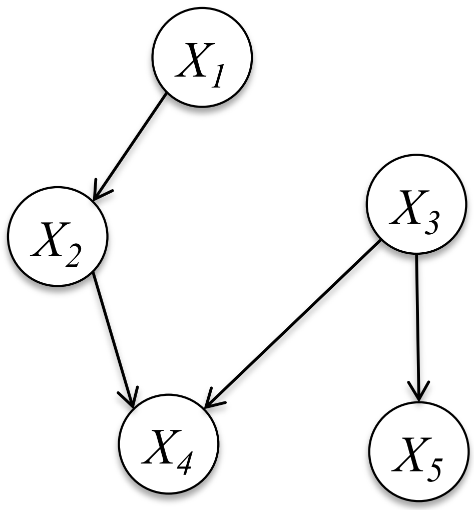


Inference



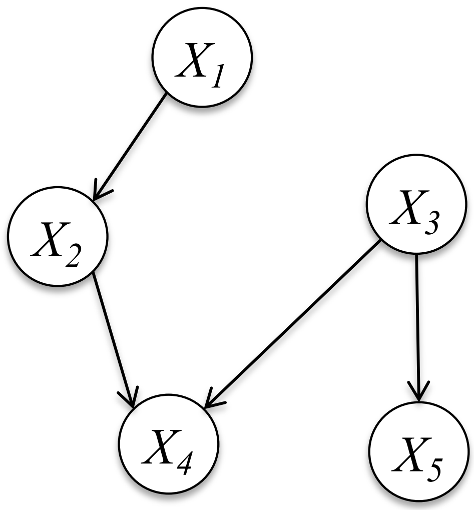
(**Inference** is usually called as a subroutine in learning)

Learning Fully Observed BNs



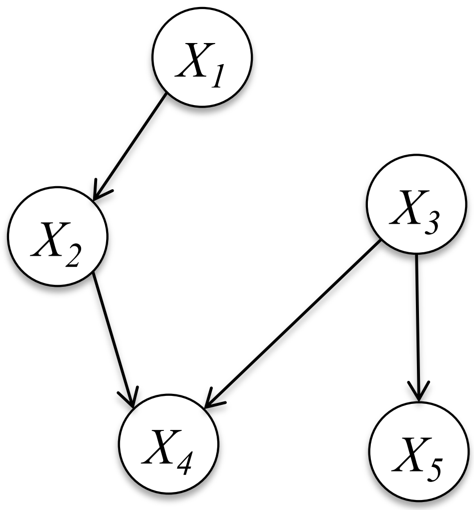
$$
\begin{aligned}
p(X_1, X_2, X_3, X_4, X_5) = \\
p(X_5|X_3)p(X_4|X_2, X_3) \\
p(X_3)p(X_2|X_1)p(X_1)
\end{aligned}
$$

Learning Fully Observed BNs



$$p(X_1, X_2, X_3, X_4, X_5) =$$
$$p(X_5|X_3)p(X_4|X_2, X_3)$$
$$p(X_3)p(X_2|X_1)p(X_1)$$

Learning Fully Observed BNs



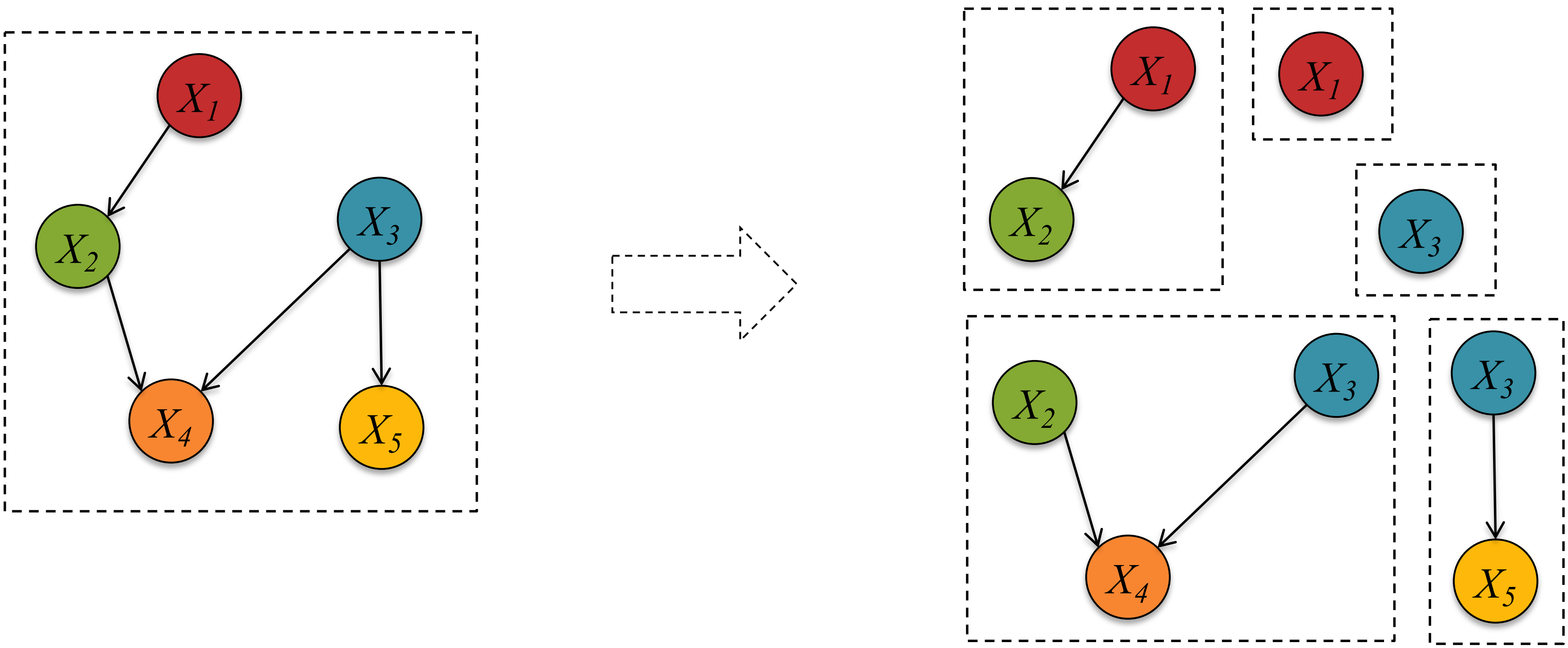
$$p(X_1, X_2, X_3, X_4, X_5) =$$
$$p(X_5|X_3)p(X_4|X_2, X_3)$$
$$p(X_3)p(X_2|X_1)p(X_1)$$

How do we learn these conditional and marginal distributions for a Bayes Net?

Learning Fully Observed BNs

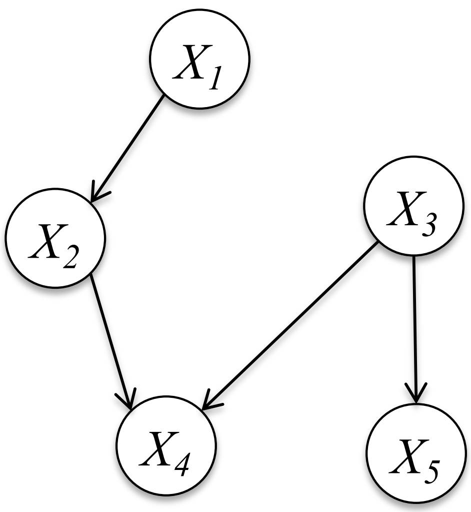
Learning this fully observed Bayesian Network is **equivalent** to learning five (small / simple) independent networks from the same data

$$p(X_1, X_2, X_3, X_4, X_5) = \\ p(X_5|X_3)p(X_4|X_2, X_3) \\ p(X_3)p(X_2|X_1)p(X_1)$$



Learning Fully Observed BNs

How do we **learn** these conditional and marginal distributions for a Bayes Net?



$$
\begin{aligned}
\boldsymbol{\theta}^* &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(X_1, X_2, X_3, X_4, X_5) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(X_5 | X_3, \theta_5) + \log p(X_4 | X_2, X_3, \theta_4) \\
&\qquad + \log p(X_3 | \theta_3) + \log p(X_2 | X_1, \theta_2) \\
&\qquad + \log p(X_1 | \theta_1)
\end{aligned}
$$

$$\theta_1^* = \underset{\theta_1}{\operatorname{argmax}} \log p(X_1 | \theta_1)$$

$$\theta_2^* = \underset{\theta_2}{\operatorname{argmax}} \log p(X_2 | X_1, \theta_2)$$

$$\theta_3^* = \underset{\theta_3}{\operatorname{argmax}} \log p(X_3 | \theta_3)$$

$$\theta_4^* = \underset{\theta_4}{\operatorname{argmax}} \log p(X_4 | X_2, X_3, \theta_4)$$

$$\theta_5^* = \underset{\theta_5}{\operatorname{argmax}} \log p(X_5 | X_3, \theta_5)$$

Example: Tornado Alarms

1. Imagine that you work at the 911 call center in Dallas
2. You receive six calls informing you that the Emergency Weather Sirens are going off
3. What do you conclude?

Example: Tornado Alarms

Hacking Attack Woke Up Dallas With Emergency Sirens, Officials Say

By ELI ROSENBERG and MAYA SALAM APRIL 8, 2017

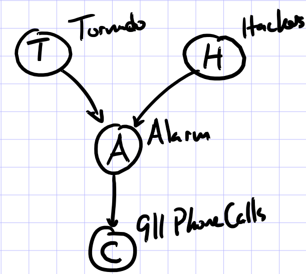
Warning sirens in Dallas, meant to alert the public to emergencies like severe weather, started sounding around 11:40 p.m. Friday, and were not shut off until 1:20 a.m. Rex C. Curry for The New York Times

1. Imagine that you work at the 911 call center in Dallas
2. You receive six calls informing you that the Emergency Weather Sirens are going off
3. What do you conclude?

Figure from https://www.nytimes.com/2017/04/08/us/dallas-emergency-sirens-hacking.html

Learning Fully Observed BNs

Ex: Tornado Alarms



$H \sim \text{Bernoulli}(\eta)$
$T \sim \text{Bernoulli}(\tau)$
$A \sim \text{Bernoulli}(\alpha_{H,T})$

parameters

$C \sim \text{Uniform}(\{1,\dots,6\}) + A * \text{Uniform}(\{1,\dots,6\})$

no parameters

integer

Dataset

i	T	H	A	C
1	0	0	0	2
2	0	0	0	6
3	0	0	0	4
⋮	1	0	0	3
	1	0	0	1
	1	0	1	10
	1	0	1	7
	0	1	0	2
	0	1	1	12
	0	1	0	5
⋮	1	1	1	10
12	1	0	0	2

MLEs in Closed Form

$$\ell(\eta,\tau,\alpha) = \log \prod_{i=1}^{12} p(t^{(i)},h^{(i)},a^{(i)},c^{(i)} \mid \eta,\tau,\alpha)$$

$$= \sum_{i=1}^{12} \log p(t^{(i)}\mid\tau) + \log p(h^{(i)}\mid\eta) + \log p(a^{(i)}\mid t^{(i)},h^{(i)},\alpha) + \log p(c^{(i)}\mid a^{(i)})$$

$$\hat\eta,\hat\tau,\hat\alpha = \operatorname{argmax} \ell(\eta,\tau,\alpha)$$

$$\hat\eta = \operatorname*{argmax}_{\eta} \sum_{i=1}^{12} \log p(h^{(i)}\mid\eta) = \#(T=1)/N$$

$$\hat\tau = \operatorname*{argmax}_{\tau} \sum_{i=1}^{12} \log p(t^{(i)}\mid\tau) = \#(H=1)/N$$

$$\hat\alpha = \operatorname*{argmax}_{\alpha} \sum_{i=1}^{12} \log p(a^{(i)}\mid t^{(i)},h^{(i)},\alpha)$$

$$\hat\alpha_{t,h} = \frac{\#(A=1,\ T=t,\ H=h)}{\#(T=t,\ H=h)}$$

What are the MLEs?

$\hat\eta = 1/3$

$\hat\tau = 1/2$

$\hat\alpha =$

	H=0	H=1
T=0	0	1/3
T=1	2/3	1

INFERENCE FOR BAYESIAN NETWORKS

A Few Problems for Bayes Nets

Suppose we already have the parameters of a Bayesian Network…

1. How do we compute the probability of a specific assignment to the variables?
   P(T=t, H=h, A=a, C=c)

2. How do we draw a sample from the joint distribution?
   t,h,a,c ~ P(T, H, A, C)

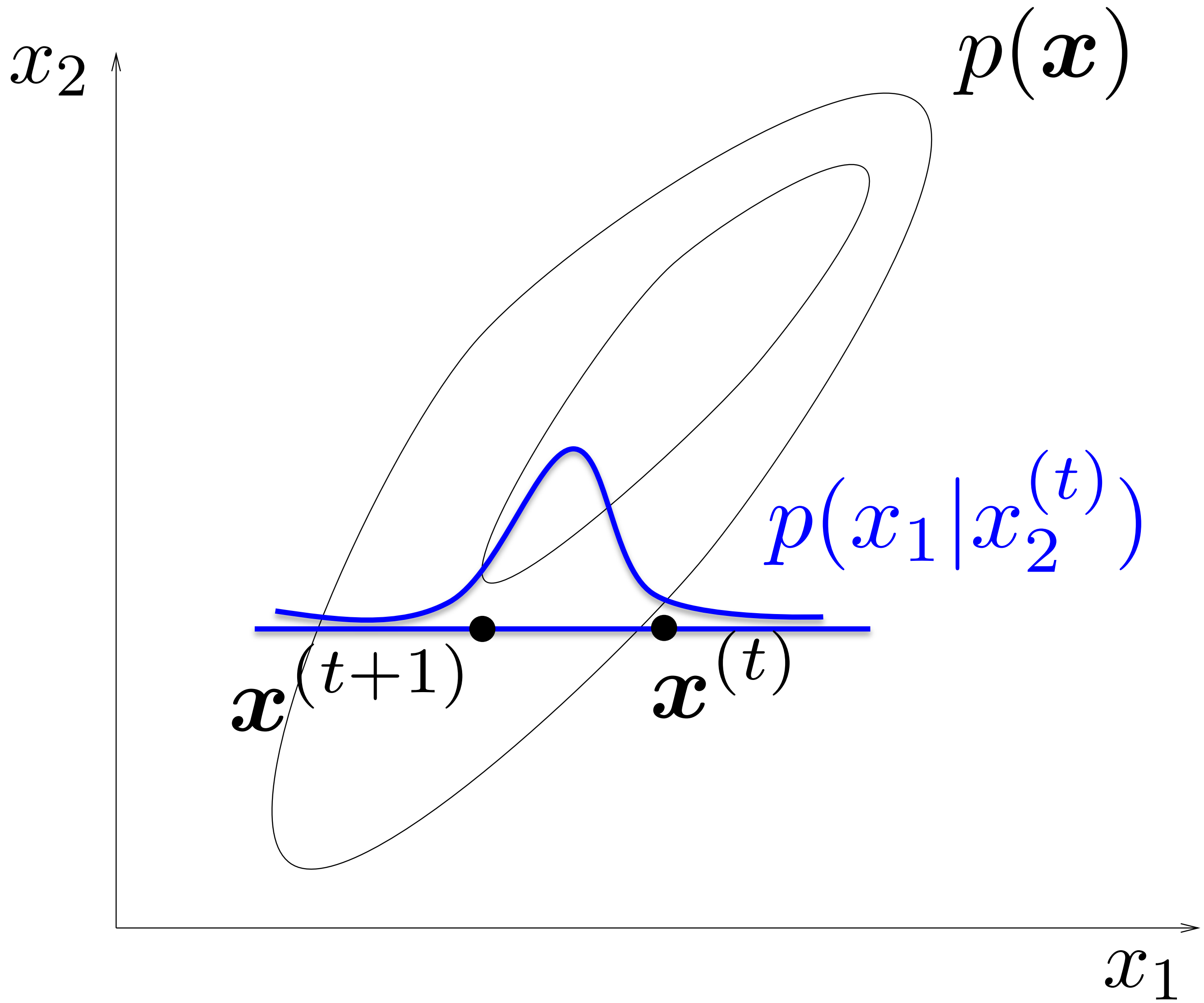
3. How do we compute marginal probabilities?
   P(A) = …

4. How do we draw samples from a conditional distribution?
   t,h,a ~ P(T, H, A | C = c)

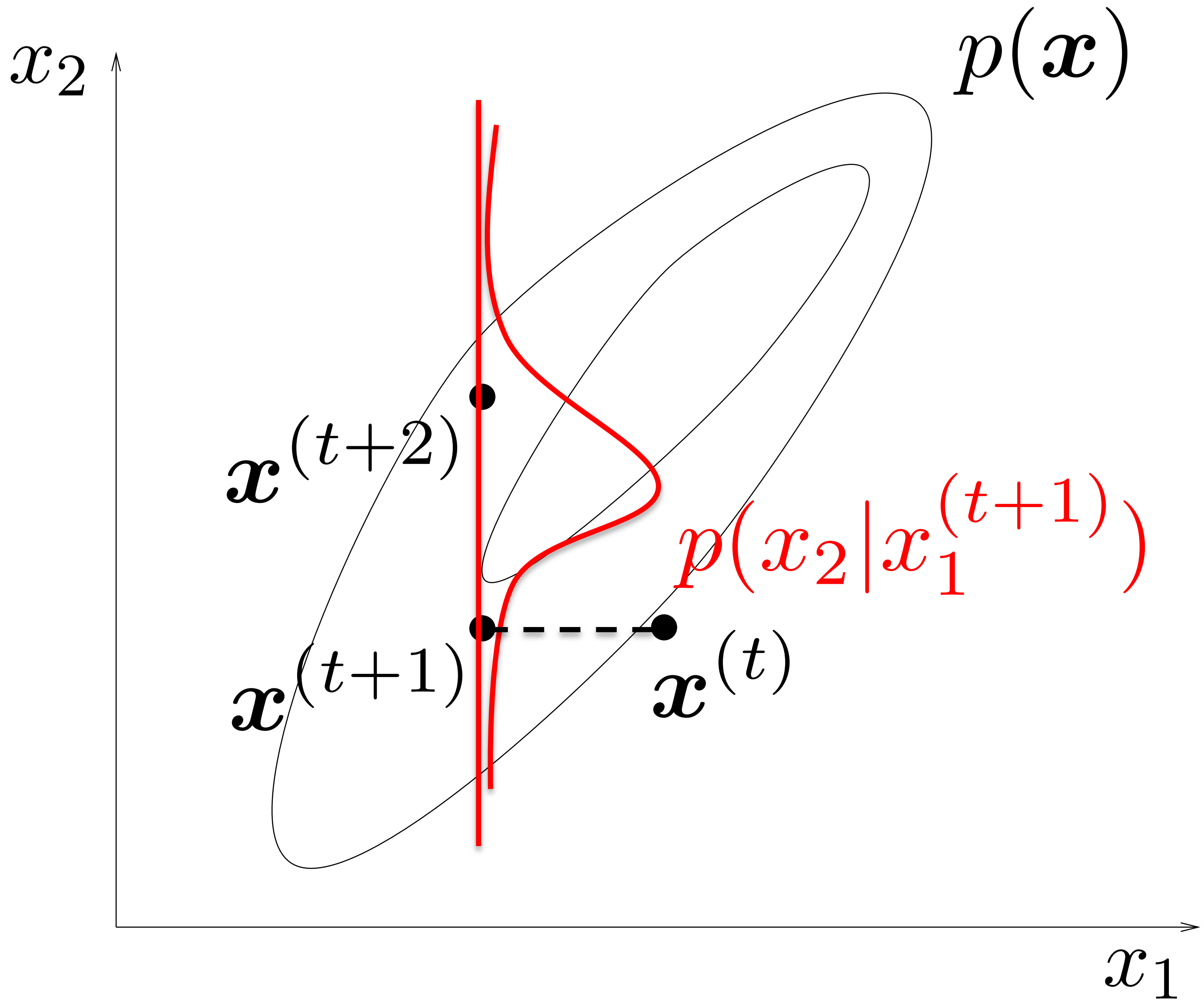
5. How do we compute conditional marginal probabilities?
   P(H | C = c) = …

Can we use samples ?

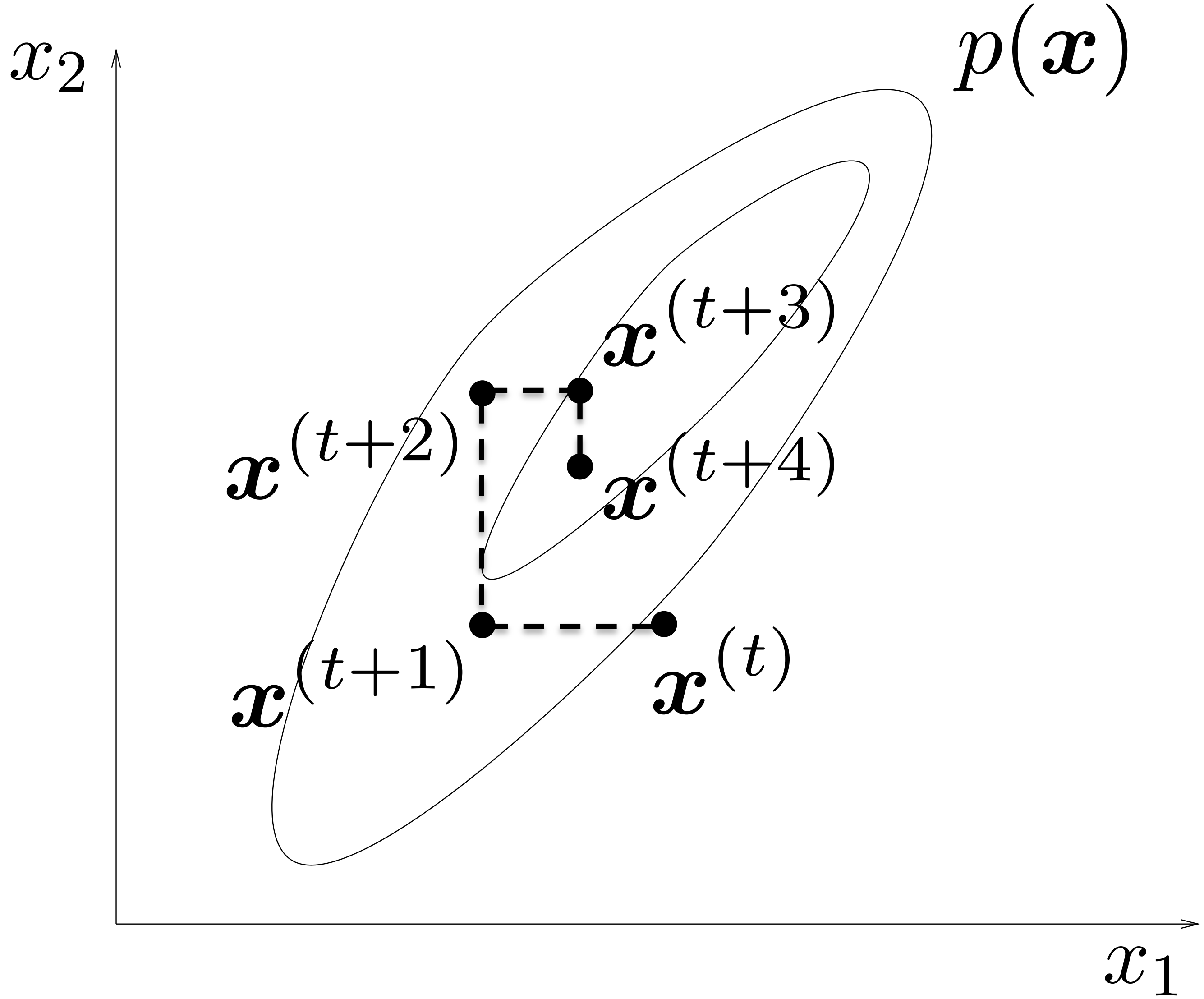
Gibbs Sampling



Gibbs Sampling



Gibbs Sampling



Gibbs Sampling

Question:
How do we draw samples from a conditional distribution?
$y_1, y_2, \ldots, y_J \sim p(y_1, y_2, \ldots, y_J \mid x_1, x_2, \ldots, x_J)$

(Approximate) Solution:

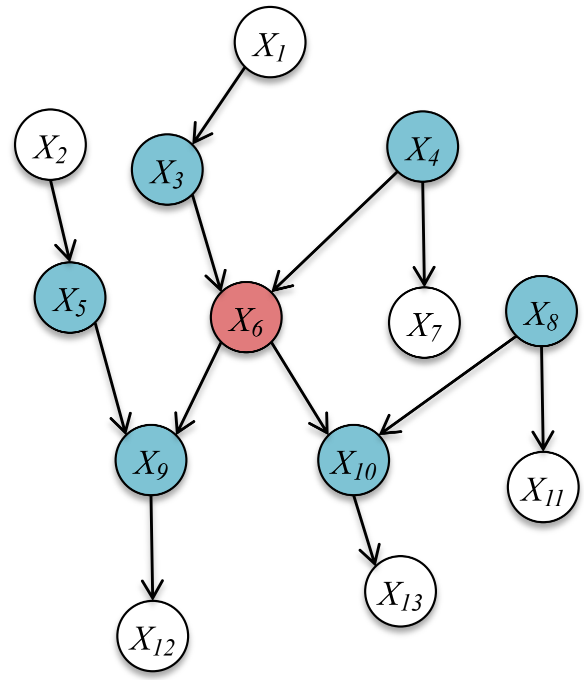
- Initialize $y_1^{(0)}, y_2^{(0)}, \ldots, y_J^{(0)}$ to arbitrary values
- For $t = 1, 2, \ldots$:
  - $y_1^{(t+1)} \sim p(y_1 \mid y_2^{(t)}, \ldots, y_J^{(t)}, x_1, x_2, \ldots, x_J)$
  - $y_2^{(t+1)} \sim p(y_2 \mid y_1^{(t+1)}, y_3^{(t)}, \ldots, y_J^{(t)}, x_1, x_2, \ldots, x_J)$
  - $y_3^{(t+1)} \sim p(y_3 \mid y_1^{(t+1)}, y_2^{(t+1)}, y_4^{(t)}, \ldots, y_J^{(t)}, x_1, x_2, \ldots, x_J)$
  - …
  - $y_J^{(t+1)} \sim p(y_J \mid y_1^{(t+1)}, y_2^{(t+1)}, \ldots, y_{J-1}^{(t+1)}, x_1, x_2, \ldots, x_J)$

Properties:

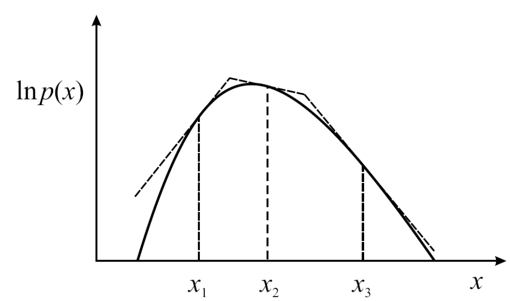
- This will eventually yield samples from
  $p(y_1, y_2, \ldots, y_J \mid x_1, x_2, \ldots, x_J)$
- But it might take a long time -- just like other Markov Chain Monte Carlo methods

Gibbs Sampling

Full conditionals only need to condition on the **Markov boundary**



- Must be "easy" to sample from conditionals
- Many conditionals are log-concave and are amenable to adaptive rejection sampling



Learning Objectives

Bayesian Networks

You should be able to…

1. Identify the conditional independence assumptions given by a generative story or a specification of a joint distribution
2. Draw a Bayesian network given a set of conditional independence assumptions
3. Define the joint distribution specified by a Bayesian network
4. User domain knowledge to construct a (simple) Bayesian network for a real-world modeling problem
5. Depict familiar models as Bayesian networks
6. Use d-separation to prove the existence of conditional indenpendencies in a Bayesian network
7. Employ a Markov boundary to identify conditional independence assumptions of a graphical model
8. Develop a supervised learning algorithm for a Bayesian network
9. Use samples from a joint distribution to compute marginal probabilities
10. Sample from the joint distribution specified by a generative story
11. Implement a Gibbs sampler for a Bayesian network