



10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Hidden Markov Models

Graphical Model Readings:

Murphy 10 – 10.2.1
Bishop 8.1, 8.2.2
HTF --
Mitchell 6.11

HMM Readings:

Murphy 10.2.2 – 10.2.3
Bishop 13.1 – 13.2
HTF --
Mitchell –

Matt Gormley
Lecture 24
April 17, 2017

Reminders

- **Peer Tutoring**
- **Homework 7: Deep Learning**
 - **Release: Wed, Apr. 05**
 - **Part I due Wed, Apr. 12 at 11:59pm**
 - **Part II due Mon, Apr. 17 at 11:59pm**
- **Homework 8: Graphical Models**
 - **Release: Mon, Apr. 17**
 - **Due: Mon, Apr. 24 at 11:59pm**

Start Early

Bayes Nets Outline

- **Motivation**
 - Structured Prediction
- **Background**
 - Conditional Independence
 - Chain Rule of Probability
- **Directed Graphical Models**
 - Writing Joint Distributions
 - Definition: Bayesian Network
 - Qualitative Specification
 - Quantitative Specification
 - Familiar Models as Bayes Nets
- **Conditional Independence in Bayes Nets**
 - Three case studies
 - D-separation
 - Markov blanket
- **Learning**
 - Fully Observed Bayes Net
 - (Partially Observed Bayes Net)
- **Inference**
 - Background: Marginal Probability
 - Sampling directly from the joint distribution
 - Gibbs Sampling



Last Lecture(s)



This Lecture

UNSUPERVISED LEARNING FOR BAYES NETS

Learning Partially Observed BNs

Recall EM:

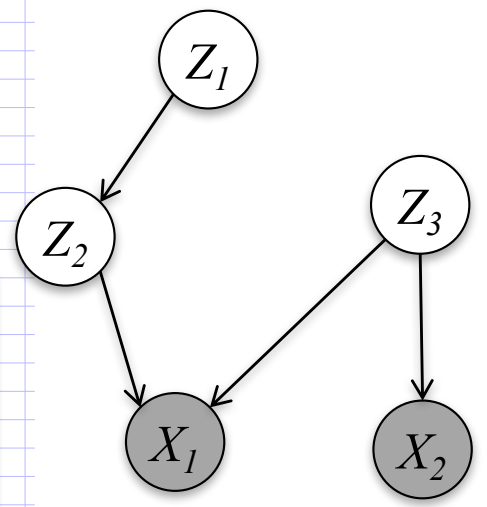
Expectation-Maximization

- ① EM is a framework for deriving local optimization algorithms
- ② Assumes a model of the form:

$$p(\vec{x}|\vec{\theta}) = \sum_{\vec{z}} p(\vec{x}, \vec{z}|\vec{\theta})$$

← parameters
← latent

- ③ Locally optimizes the marginal likelihood
- $$l(\theta) = \sum_{i=1}^N \log p(\vec{x}^{(i)}|\vec{\theta})$$



How to compute this for an arbitrary Bayes Net?

EM Algorithm

- ① Randomly initialize θ
- ② Iterate until convergence:

E-step Compute "expected" $p(z^{(i)}|x^{(i)}, \theta)$ using current parameters θ

M-step Update $\theta \leftarrow \underset{\theta'}{\operatorname{argmax}} Q(\theta'|\theta)$

↑ current value
↑ new value

★ Each step increases $Q(\theta'|\theta)$ which in turn increases $l(\theta)$ marginal l.l.

Beyond the scope of today's lecture

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?

$$t,h,a,c \sim P(T, H, A, C)$$

3. How do we compute marginal probabilities?

$$P(A) = \dots$$

4. How do we draw samples from a conditional distribution?

$$t,h,a \sim P(T, H, A \mid C = c)$$

5. How do we compute conditional marginal probabilities?

$$P(H \mid C = c) = \dots$$



Can we
use
samples
?

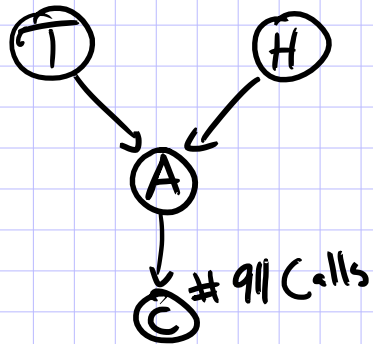
Inference for Bayes Nets

Whiteboard

- Background: Marginal Probability
- Sampling from a joint distribution
- Gibbs Sampling

Sampling from a Joint Distribution

Ex: Tornado



$$T \sim \text{Bernoulli}(\eta)$$

$$\eta = 1/2$$

$$H \sim \text{Bernoulli}(\eta)$$

$$\eta = 1/3$$

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

$$\alpha = \begin{matrix} H=0 & H=1 \\ T=0 & 0 & 1/2 \\ T=1 & 1/2 & 1 \end{matrix}$$

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

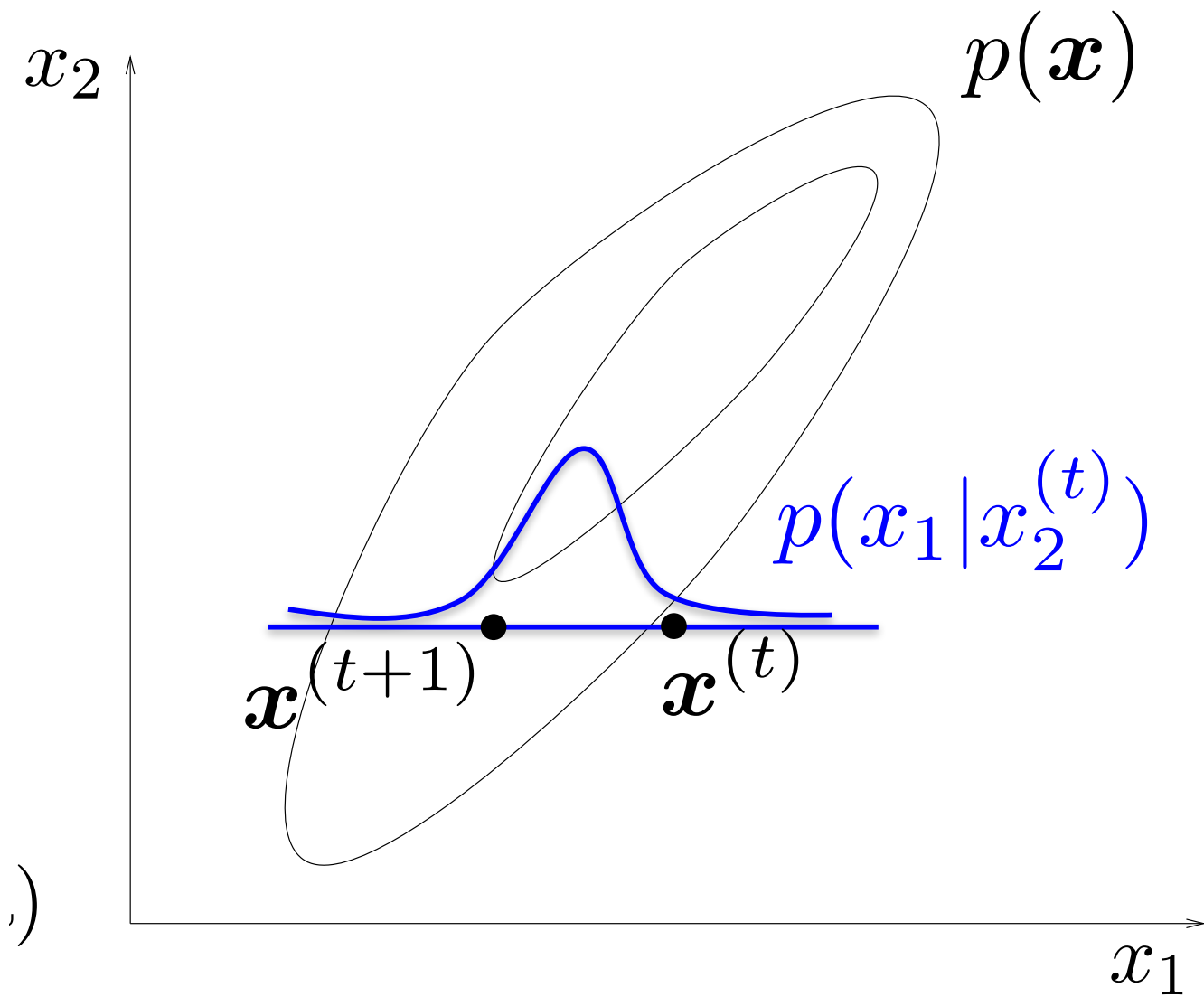
Integer

We can use these samples to estimate many different probabilities!

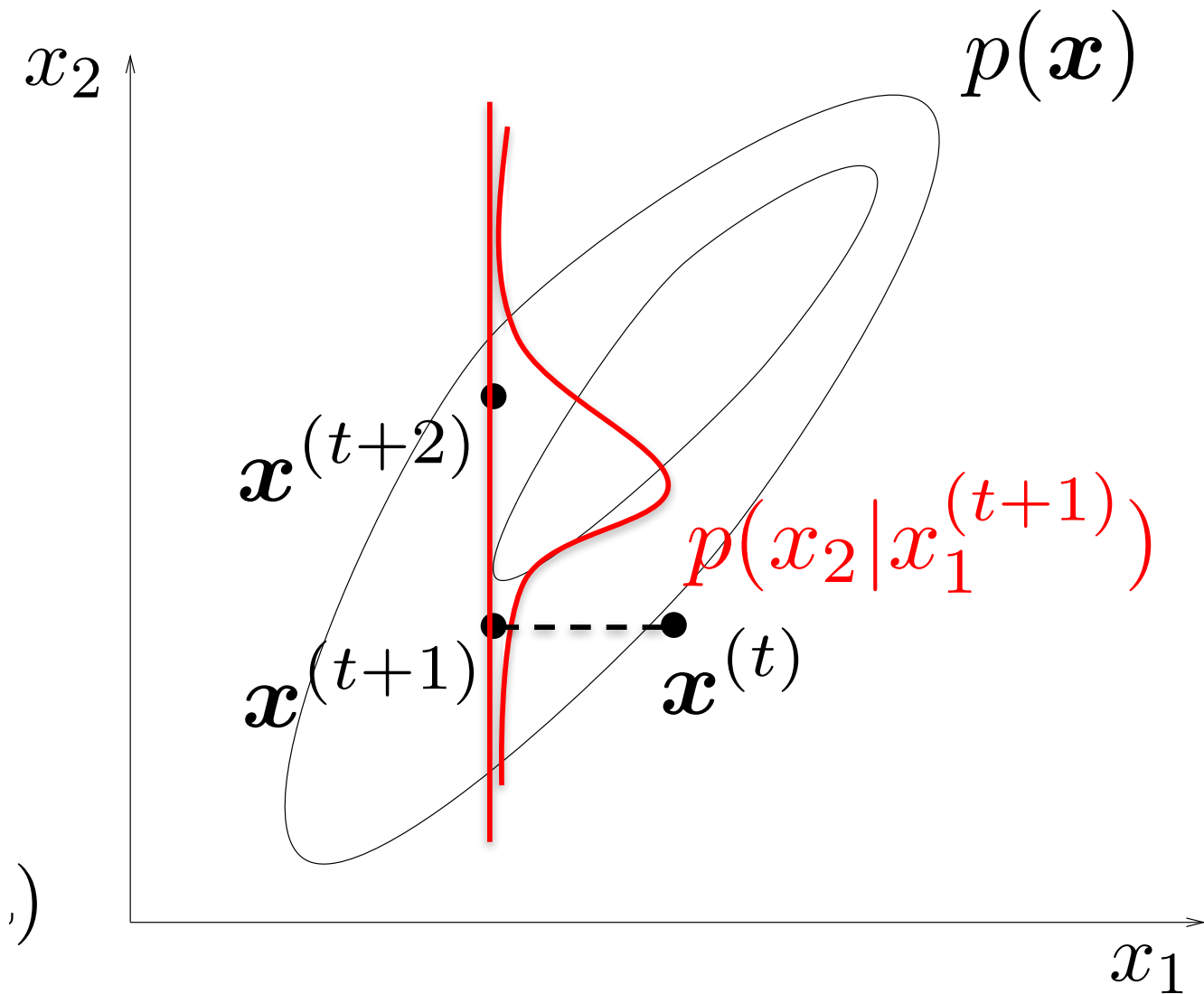


T	H	A	C

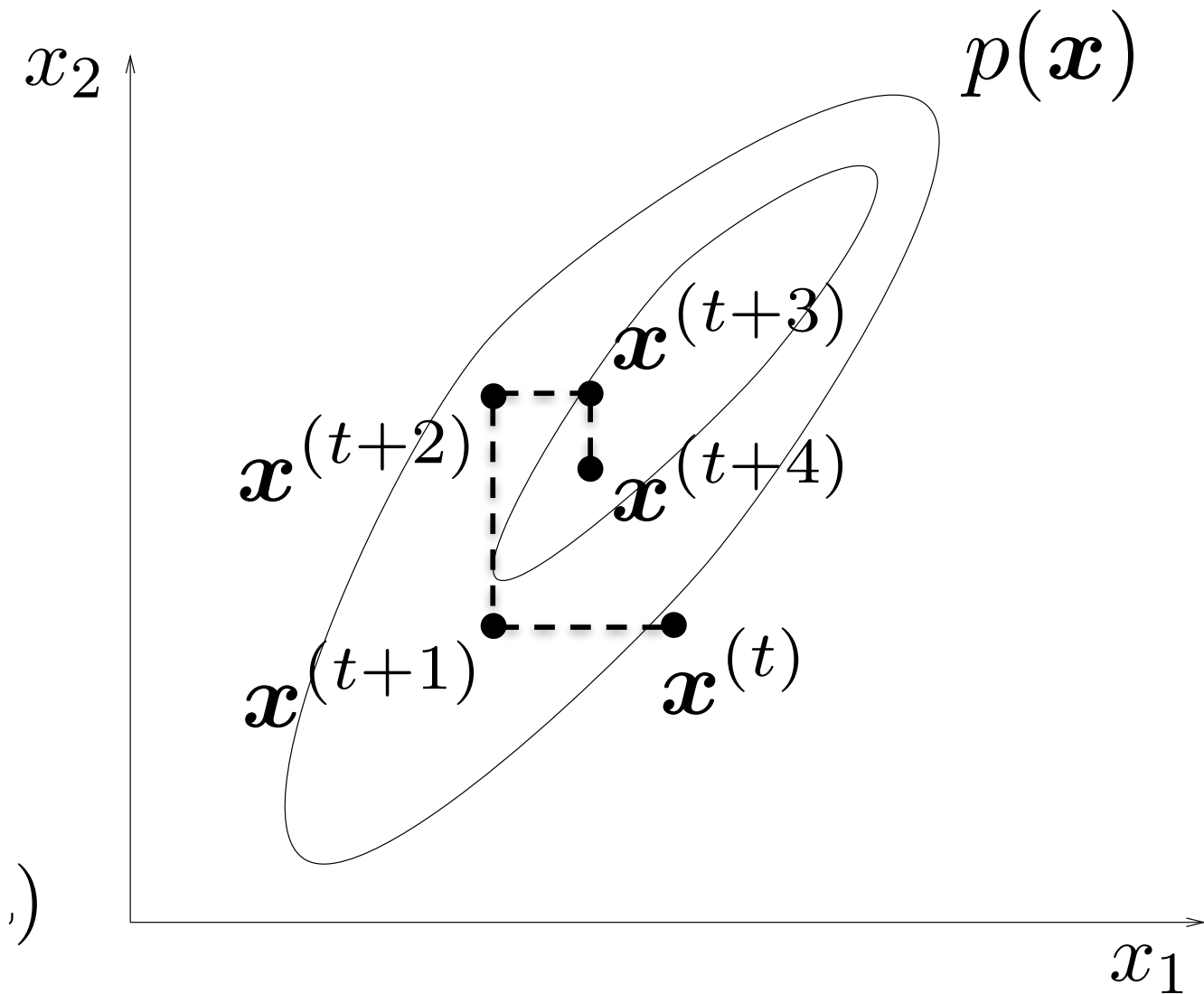
Gibbs Sampling



Gibbs Sampling



Gibbs Sampling



Gibbs Sampling

Question:

How do we draw samples from a conditional distribution?

$$y_1, y_2, \dots, y_J \sim p(y_1, y_2, \dots, y_J \mid x_1, x_2, \dots, x_J)$$

(Approximate) Solution:

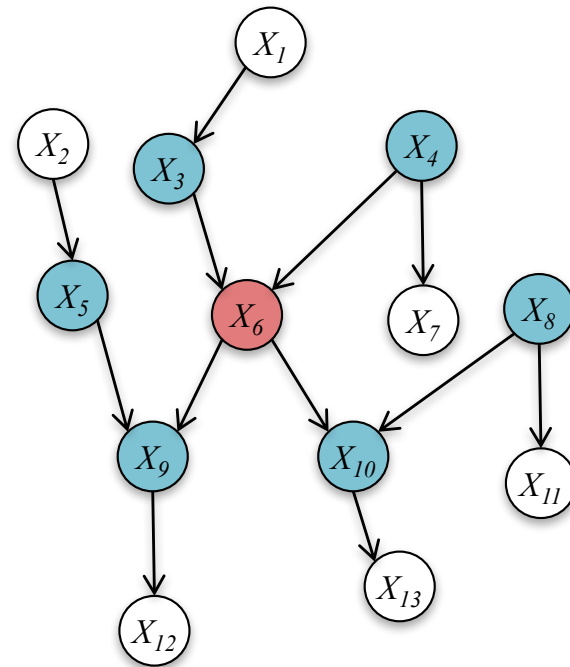
- Initialize $y_1^{(0)}, y_2^{(0)}, \dots, y_J^{(0)}$ to arbitrary values
- For $t = 1, 2, \dots$:
 - $y_1^{(t+1)} \sim p(y_1 \mid y_2^{(t)}, \dots, y_J^{(t)}, x_1, x_2, \dots, x_J)$
 - $y_2^{(t+1)} \sim p(y_2 \mid y_1^{(t+1)}, y_3^{(t)}, \dots, y_J^{(t)}, x_1, x_2, \dots, x_J)$
 - $y_3^{(t+1)} \sim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

Properties:

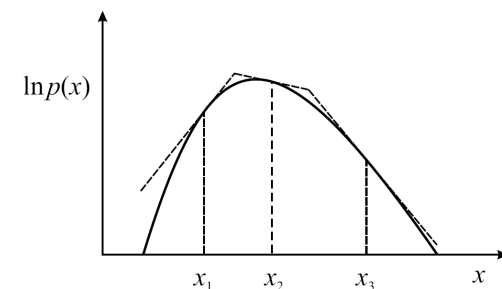
- This will eventually yield samples from $p(y_1, y_2, \dots, y_J \mid x_1, x_2, \dots, x_J)$
- But it might take a long time -- just like other Markov Chain Monte Carlo methods

Gibbs Sampling

Full conditionals
only need to
condition on the
Markov Blanket



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



HIDDEN MARKOV MODEL (HMM)

HMM Outline

- **Motivation**
 - Time Series Data
- **Hidden Markov Model (HMM)**
 - Example: Squirrel Hill Tunnel Closures [courtesy of Roni Rosenfeld]
 - Background: Markov Models
 - From Mixture Model to HMM
 - History of HMMs
 - Higher-order HMMs
- **Training HMMs**
 - (Supervised) Likelihood for HMM
 - Maximum Likelihood Estimation (MLE) for HMM
 - EM for HMM (aka. Baum-Welch algorithm)
- **Forward-Backward Algorithm**
 - Three Inference Problems for HMM
 - Great Ideas in ML: Message Passing
 - Example: Forward-Backward on 3-word Sentence
 - Derivation of Forward Algorithm
 - Forward-Backward Algorithm
 - Viterbi algorithm

Markov Models

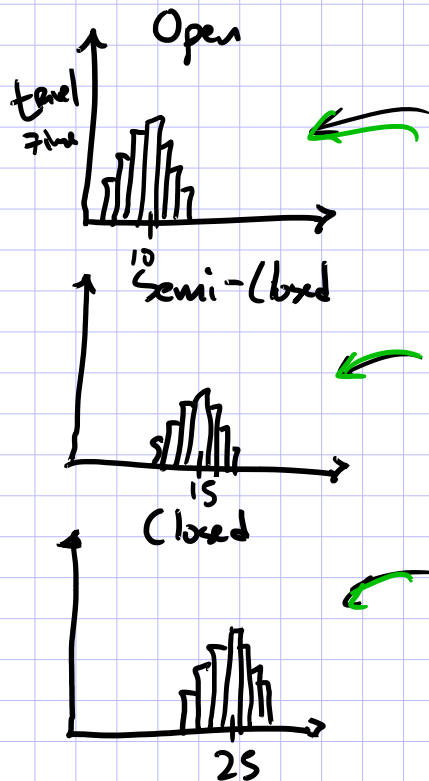
Whiteboard

- Example: Squirrel Hill Tunnel Closures
[courtesy of Roni Rosenfeld]
- First-order Markov assumption
- Conditional independence assumptions

Example: Squirrel Hill Tunnel Closures

$x_t \triangleq$ travel time on day t

$y_t \triangleq$ state of the tunnel on day t



MLEs for States

$$\hat{P}_{\text{OPEN}} = \frac{\#(y_t = \text{OPEN})}{365} \quad \leftarrow S_1$$

$$\hat{P}_{\text{SEMI}} = \frac{\#(y_t = \text{semi})}{365} \quad \leftarrow S_2$$

$$\hat{P}_{\text{CLOSED}} = \dots \quad \leftarrow S_3$$

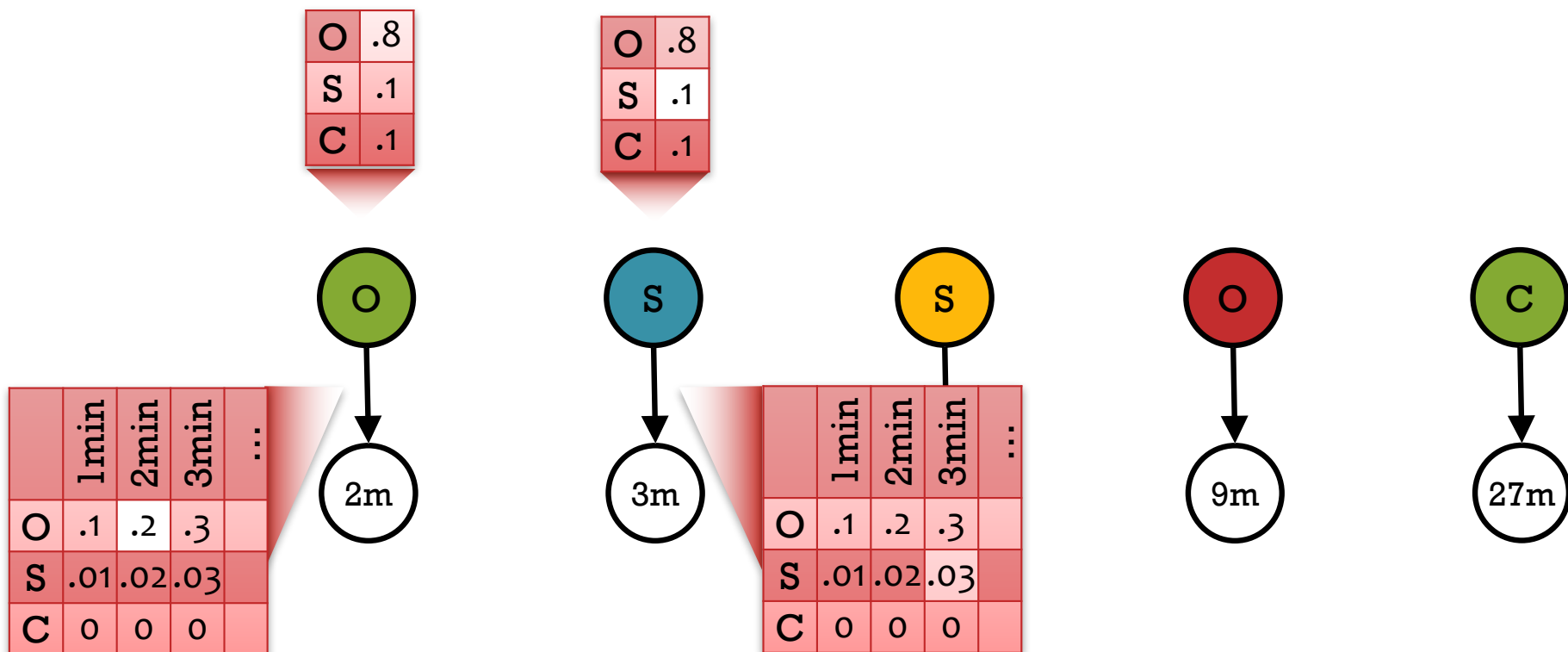
Prob. under Mixture Model

$$p(x_t) = \sum_{j=1}^3 \underline{p(y_t = s_j)} \underline{p(x_t | y_t = s_j)}$$

Mixture Model for Time Series Data

We could treat each (tunnel state, travel time) pair as independent. This corresponds to a Naïve Bayes model with a single feature (the word).

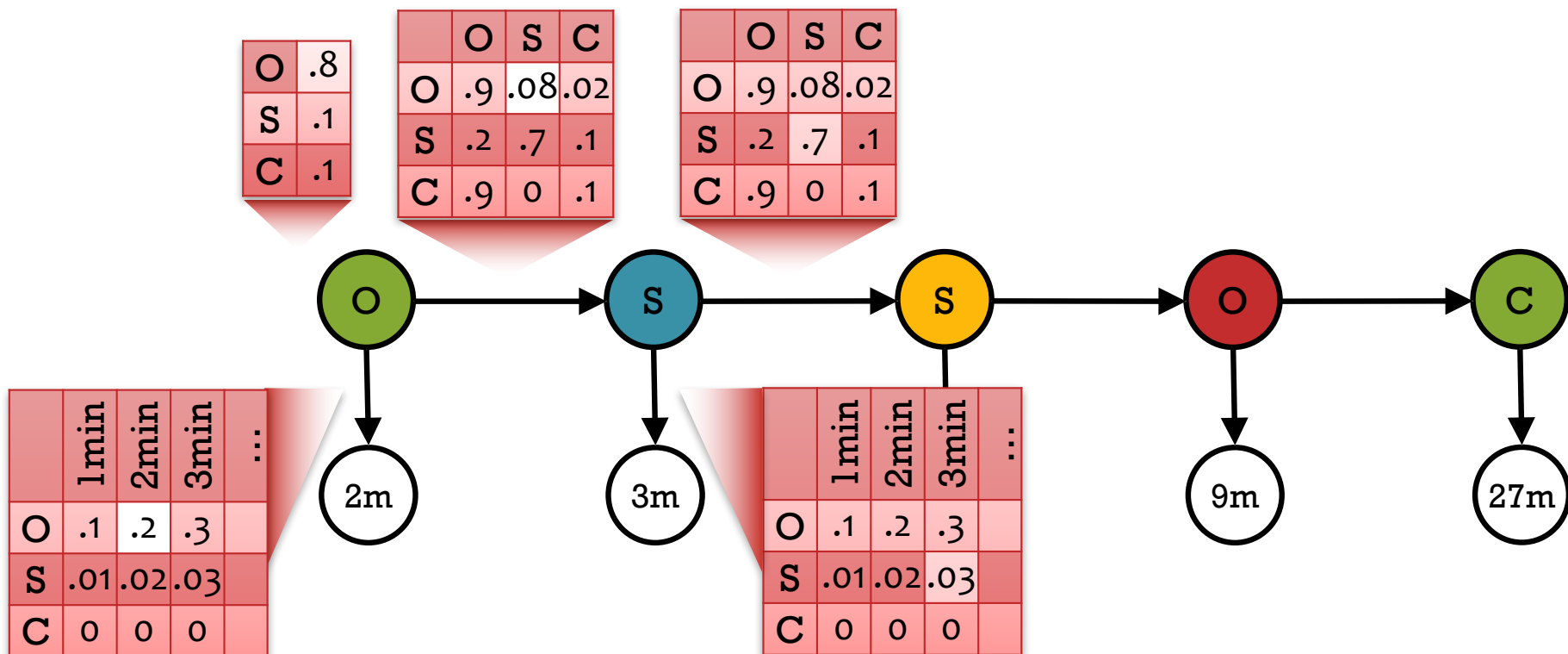
$$p(O, S, S, O, C, 2m, 3m, 18m, 9m, 27m) = (.8 * .2 * .1 * .03 * \dots)$$



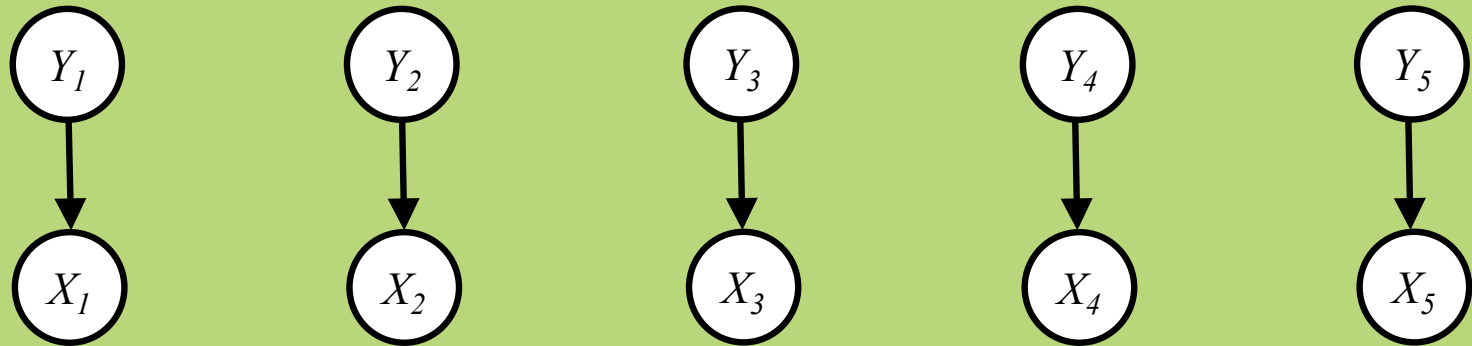
Hidden Markov Model

A Hidden Markov Model (HMM) provides a joint distribution over the the tunnel states / travel times with an assumption of dependence between adjacent tunnel states.

$$p(O, S, S, O, C, 2m, 3m, 18m, 9m, 27m) = (.8 * .08 * .2 * .7 * .03 * \dots)$$

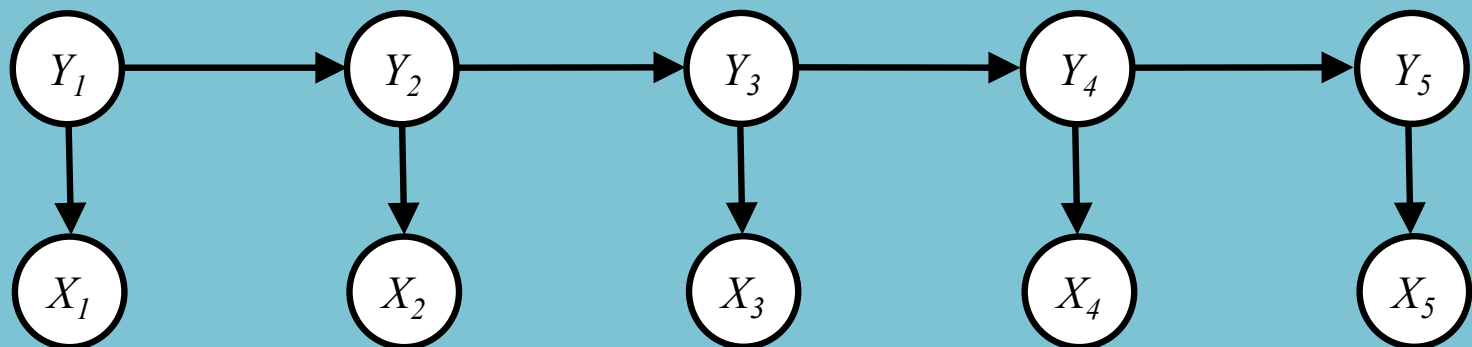


From Mixture Model to HMM



“Naïve Bayes”:

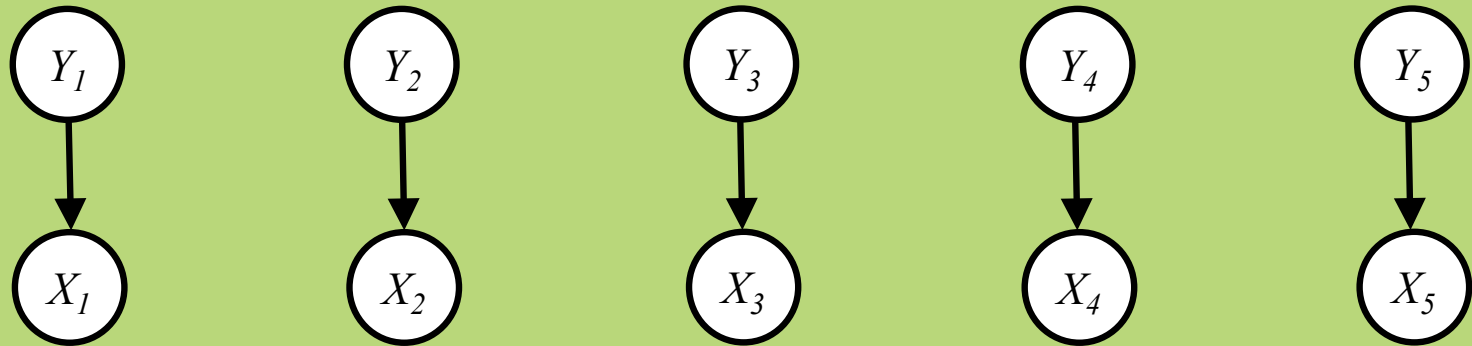
$$P(\mathbf{X}, \mathbf{Y}) = \prod_{t=1}^T P(X_t|Y_t)p(Y_t)$$



HMM:

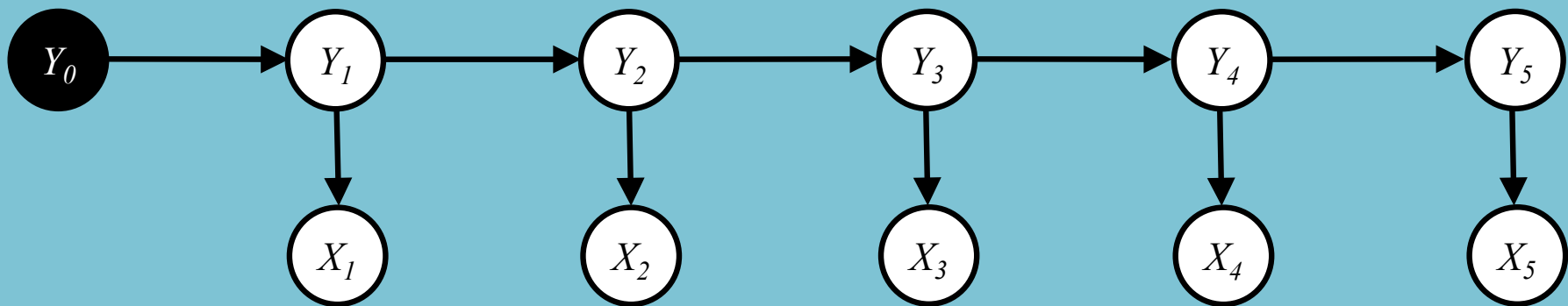
$$P(\mathbf{X}, \mathbf{Y}) = P(Y_1) \left(\prod_{t=1}^T P(X_t|Y_t) \right) \left(\prod_{t=2}^T p(Y_t|Y_{t-1}) \right)$$

From Mixture Model to HMM



“Naïve Bayes”:

$$P(\mathbf{X}, \mathbf{Y}) = \prod_{t=1}^T P(X_t|Y_t)p(Y_t)$$



HMM:

$$P(\mathbf{X}, \mathbf{Y}|Y_0) = \prod_{t=1}^T P(X_t|Y_t)p(Y_t|Y_{t-1})$$

SUPERVISED LEARNING FOR HMMS

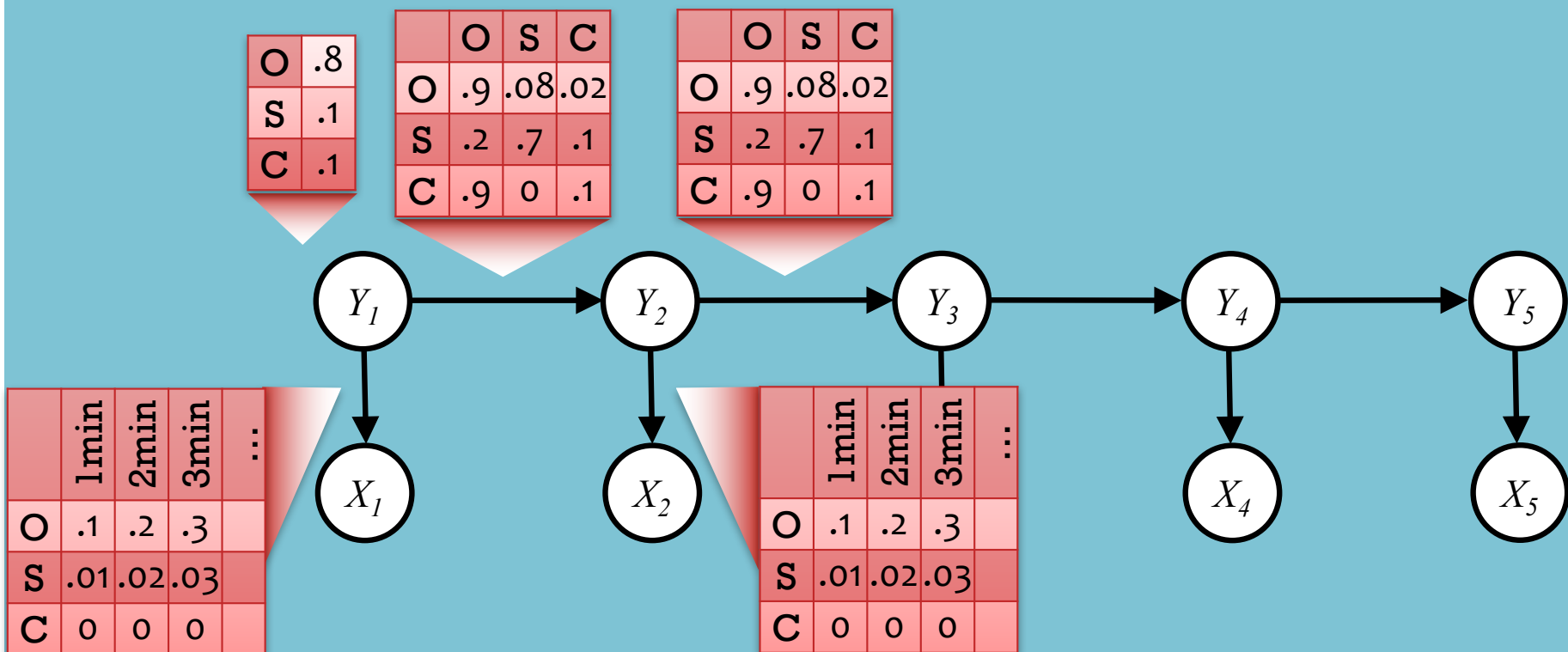
Hidden Markov Model

HMM Parameters:

Emission matrix, \mathbf{A} , where $P(X_t = k | Y_t = j) = A_{j,k}, \forall t, k$

Transition matrix, \mathbf{B} , where $P(Y_t = k | Y_{t-1} = j) = B_{j,k}, \forall t, k$

Initial probs, \mathbf{C} , where $P(Y_1 = k) = C_k, \forall k$



Hidden Markov Model

HMM Parameters:

Emission matrix, \mathbf{A} , where $P(X_t = k | Y_t = j) = A_{j,k}, \forall t, k$

Transition matrix, \mathbf{B} , where $P(Y_t = k | Y_{t-1} = j) = B_{j,k}, \forall t, k$

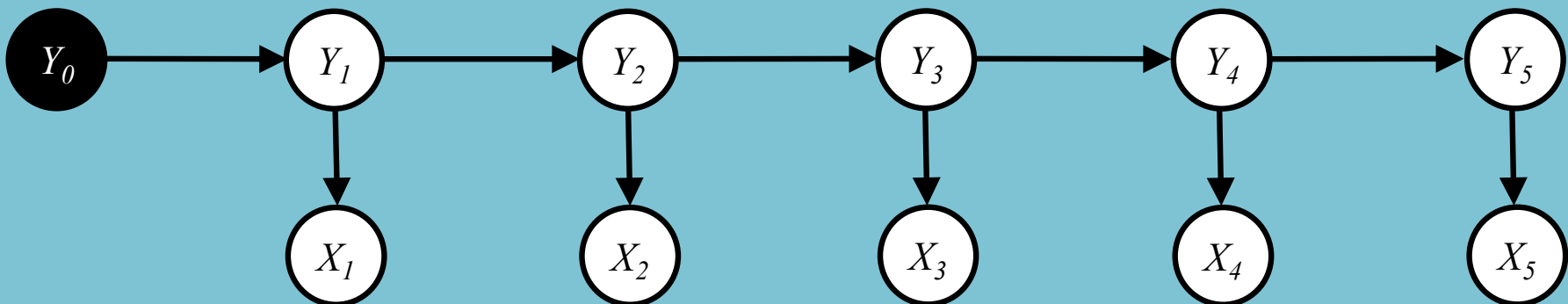
Assumption: $y_0 = \text{START}$

Generative Story:

$$Y_t \sim \text{Multinomial}(\mathbf{B}_{Y_{t-1}}) \quad \forall t$$

$$X_t \sim \text{Multinomial}(\mathbf{A}_{Y_t}) \quad \forall t$$

For notational convenience, we fold the initial probabilities \mathbf{C} into the transition matrix \mathbf{B} by our assumption.

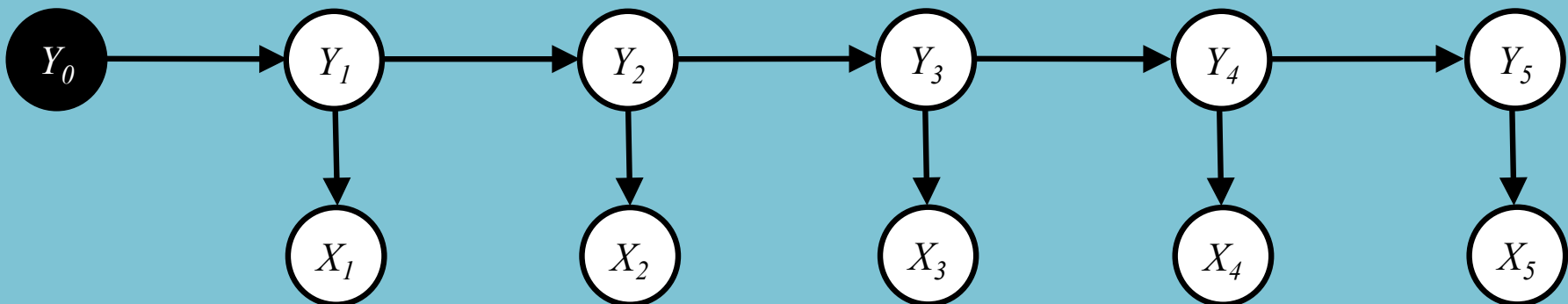


Hidden Markov Model

Joint Distribution:

$y_0 = \text{START}$

$$p(\mathbf{x}, \mathbf{y} | y_0) = \prod_{t=1}^T p(x_t | y_t) p(y_t | y_{t-1})$$
$$= \prod_{t=1}^T A_{y_t, x_t} B_{y_{t-1}, y_t}$$



Training HMMs

Whiteboard

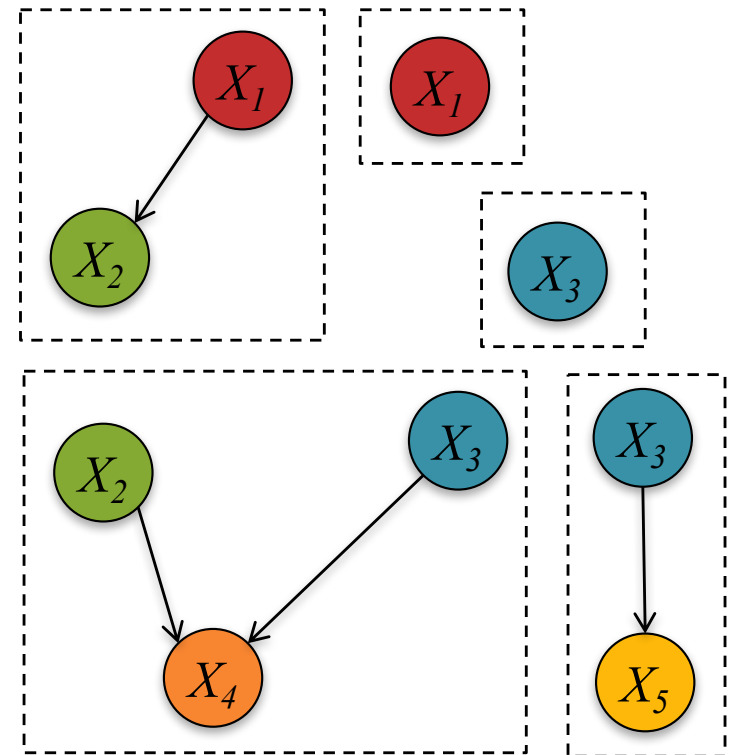
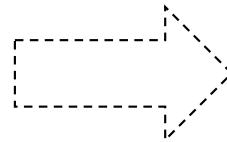
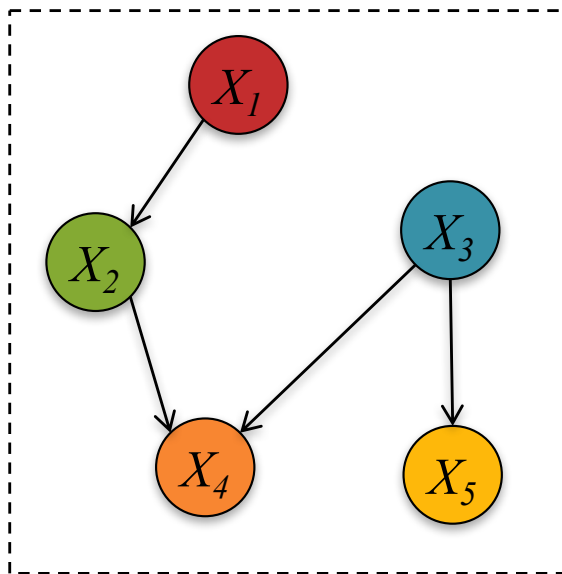
- (Supervised) Likelihood for an HMM
- Maximum Likelihood Estimation (MLE) for HMM

Recall...

Learning Fully Observed BNs

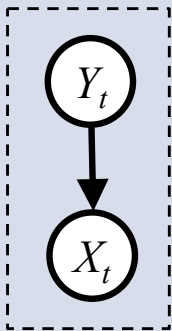
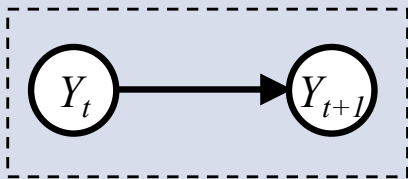
Learning this fully observed Bayesian Network is **equivalent** to learning five (small / simple) independent networks from the same data

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



Supervised Learning for HMMs

Learning an HMM decomposes into solving two (independent) Mixture Models



$$D = \{(\vec{x}^{(i)}, \vec{y}^{(i)})\}_{i=1}^N$$

$$\begin{aligned} \text{Likelihood: } \ell(A, B) &= \sum_{i=1}^N \log p(\vec{x}^{(i)}, \vec{y}^{(i)}) \\ &= \sum_{i=1}^N \left[\sum_{t=1}^T \log p(y_t^{(i)} | y_{t-1}^{(i)}, B) + \log p(x_t^{(i)} | y_t^{(i)}, A) \right] \end{aligned}$$

$$\text{MLE: } \hat{A}, \hat{B} = \text{argmax}_{A, B} \ell(A, B)$$

$$\hat{A} = \text{argmax}_A \sum_{i=1}^N \left[\sum_{t=1}^T \log p(x_t^{(i)} | y_t^{(i)}, A) \right]$$

$$\hat{B} = \text{argmax}_B \sum_{i=1}^N \left[\sum_{t=1}^T \log p(y_t^{(i)} | y_{t-1}^{(i)}, B) \right]$$

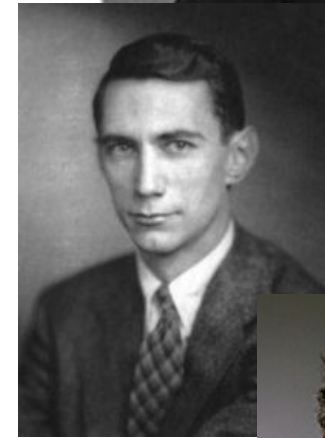
↑ can solve in closed form to get...

$$\hat{B}_{jk} = \frac{\#(y_t^{(i)} = k \text{ and } y_{t-1}^{(i)} = j)}{\#(y_{t-1}^{(i)} = j)}$$

$$\hat{A}_{jk} = \frac{\#(x_t^{(i)} = k \text{ and } y_t^{(i)} = j)}{\#(y_t^{(i)} = j)}$$

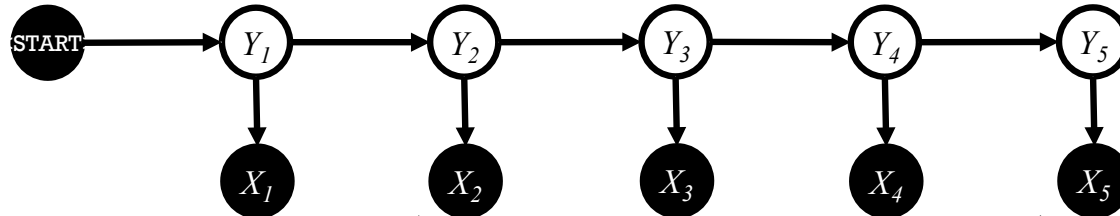
HMMs: History

- 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
- ...

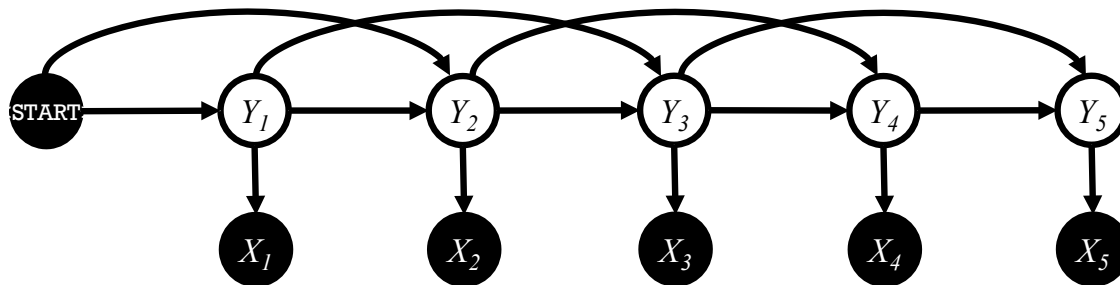


Higher-order HMMs

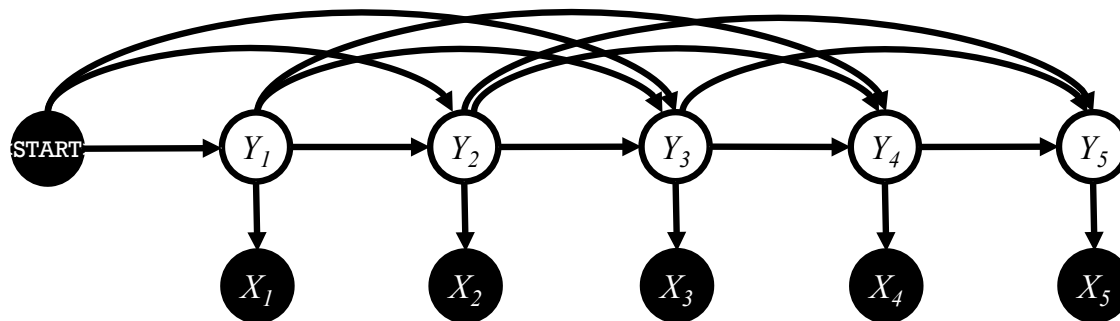
- 1st-order HMM (i.e. bigram HMM)



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



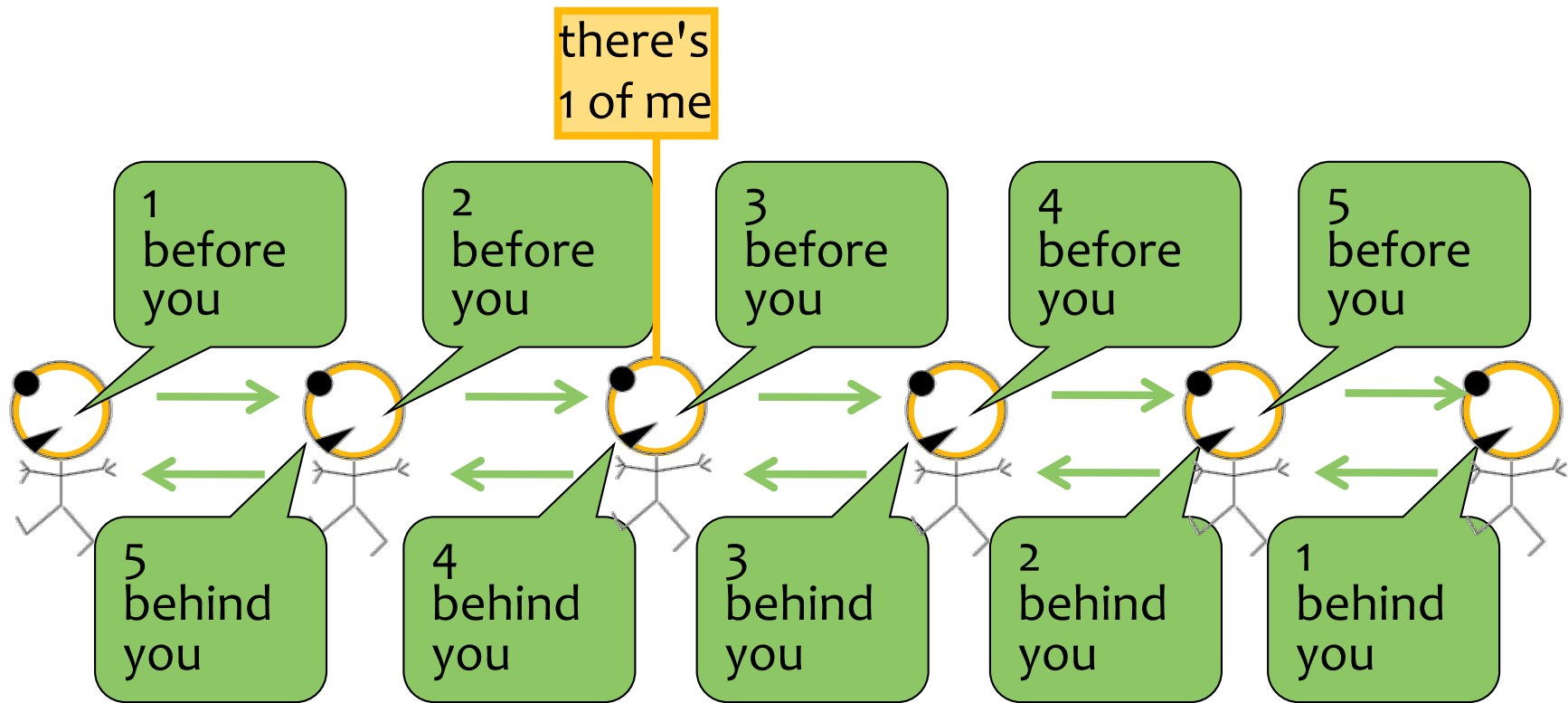
- 3rd-order HMM



BACKGROUND: MESSAGE PASSING

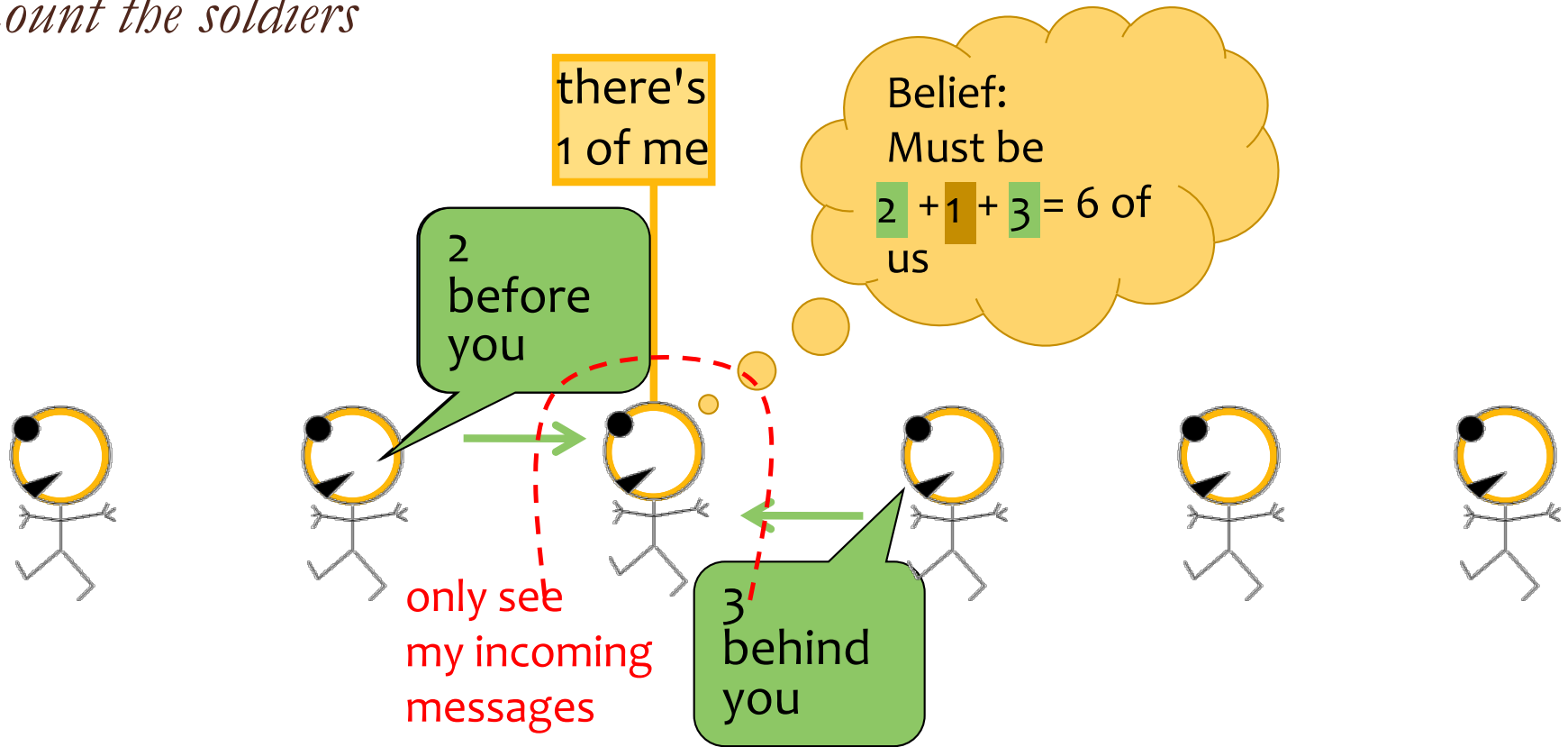
Great Ideas in ML: Message Passing

Count the soldiers



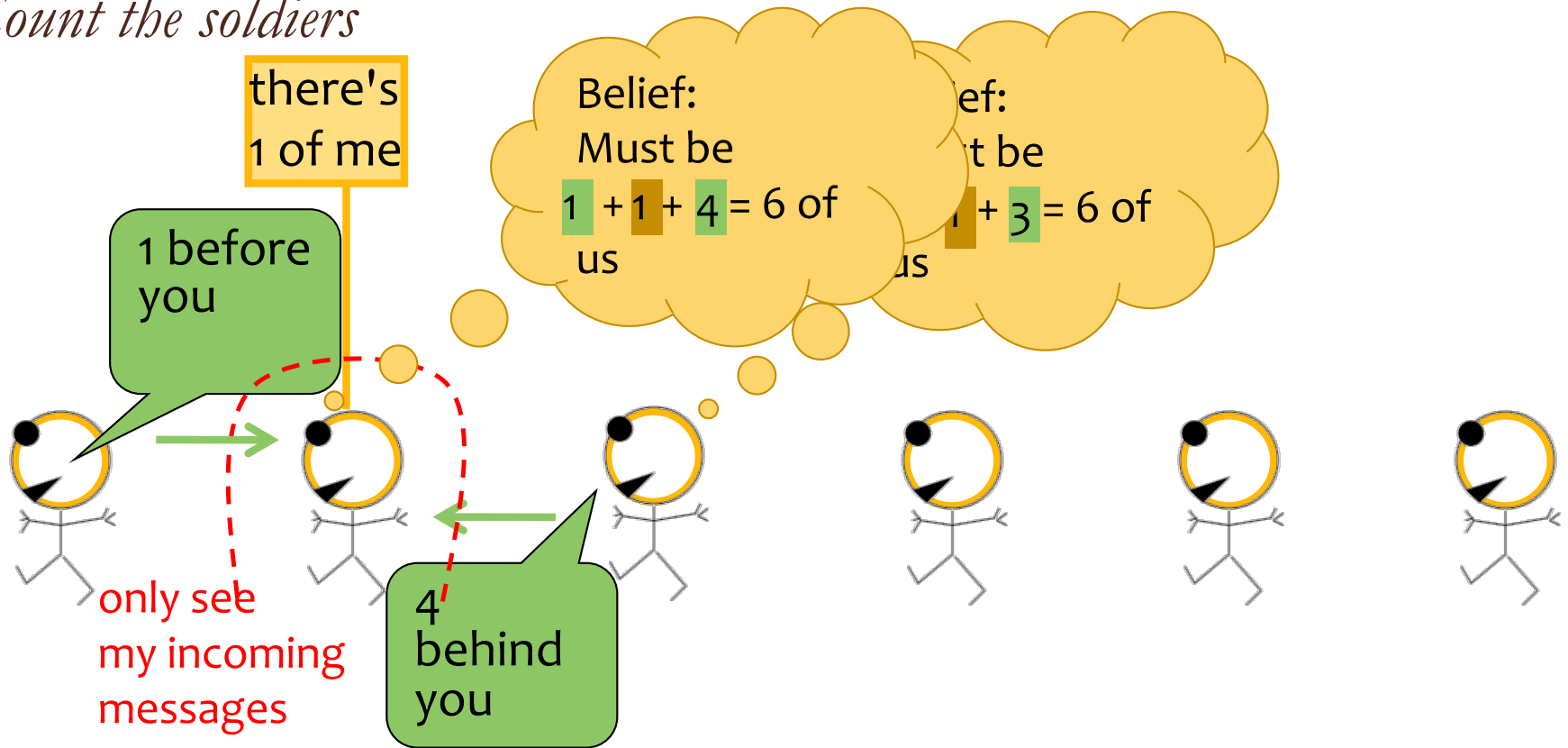
Great Ideas in ML: Message Passing

Count the soldiers



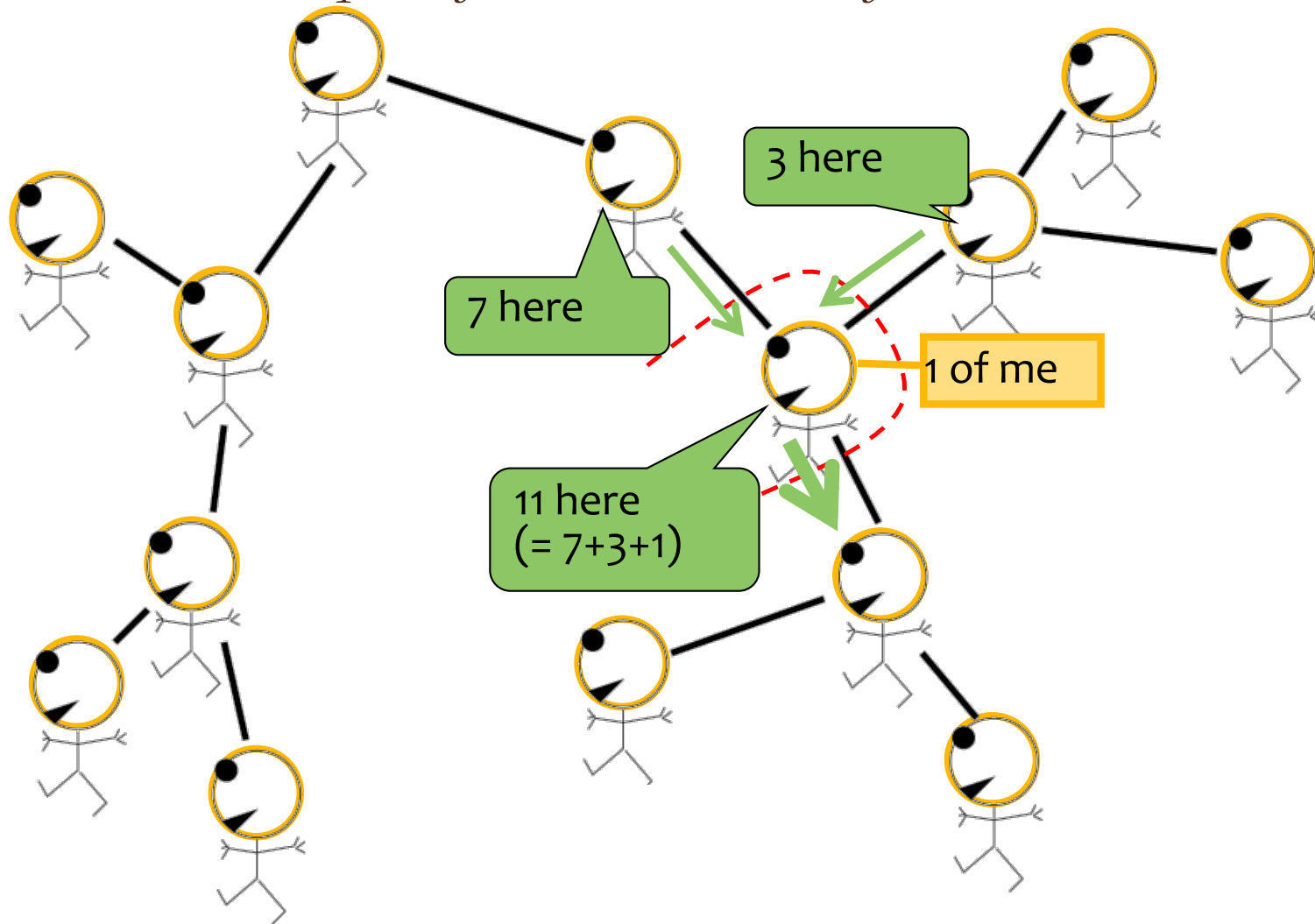
Great Ideas in ML: Message Passing

Count the soldiers



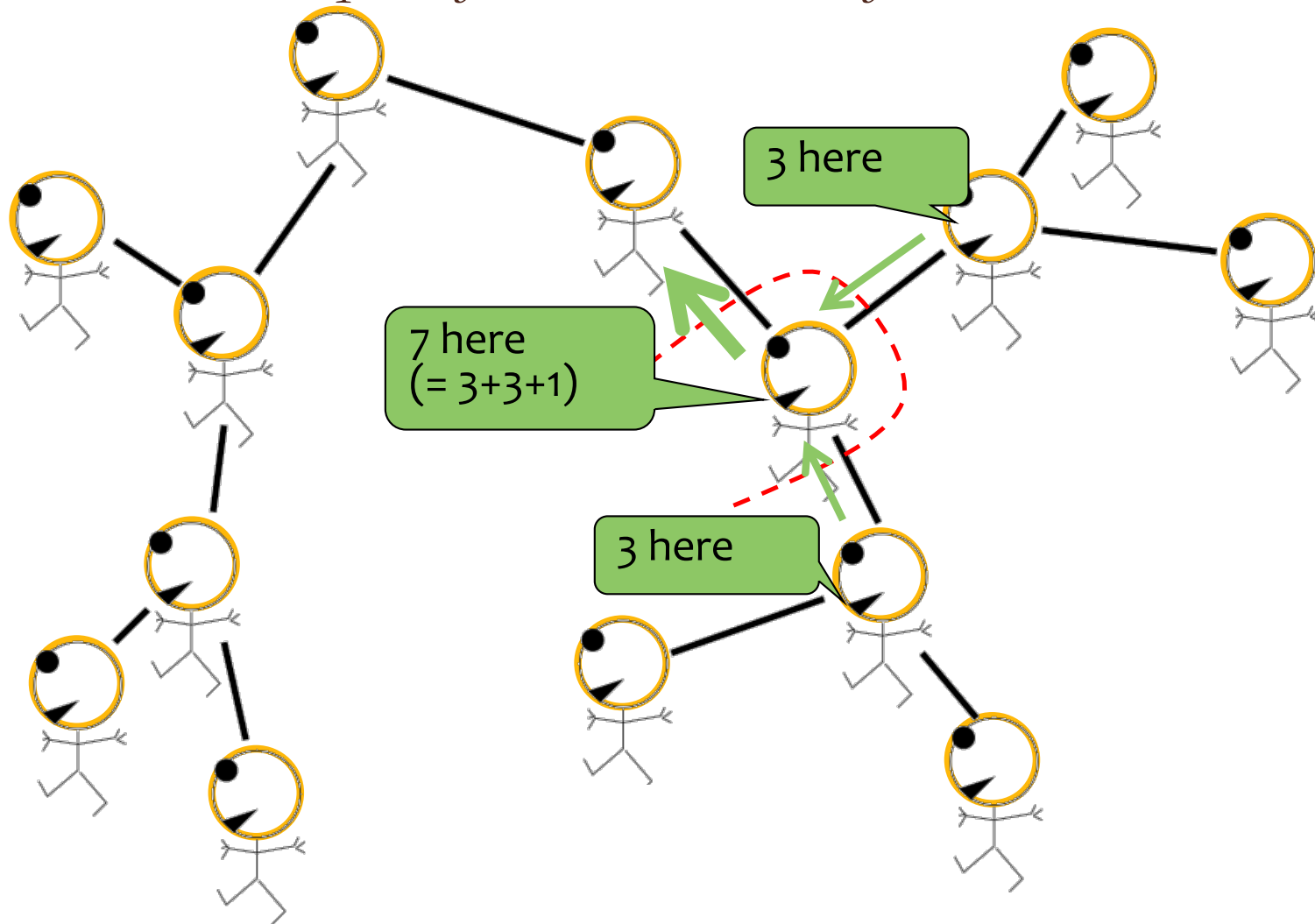
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



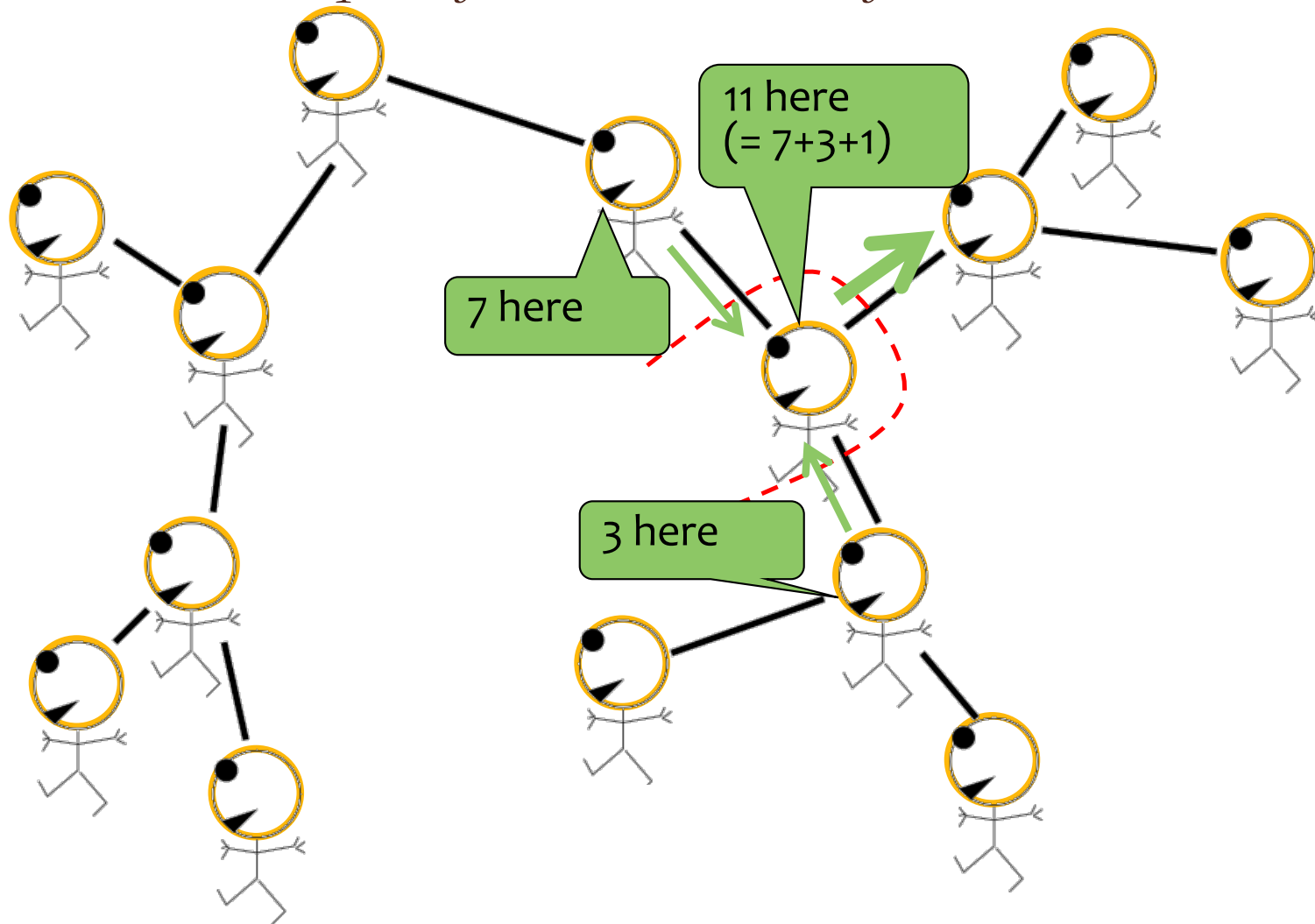
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



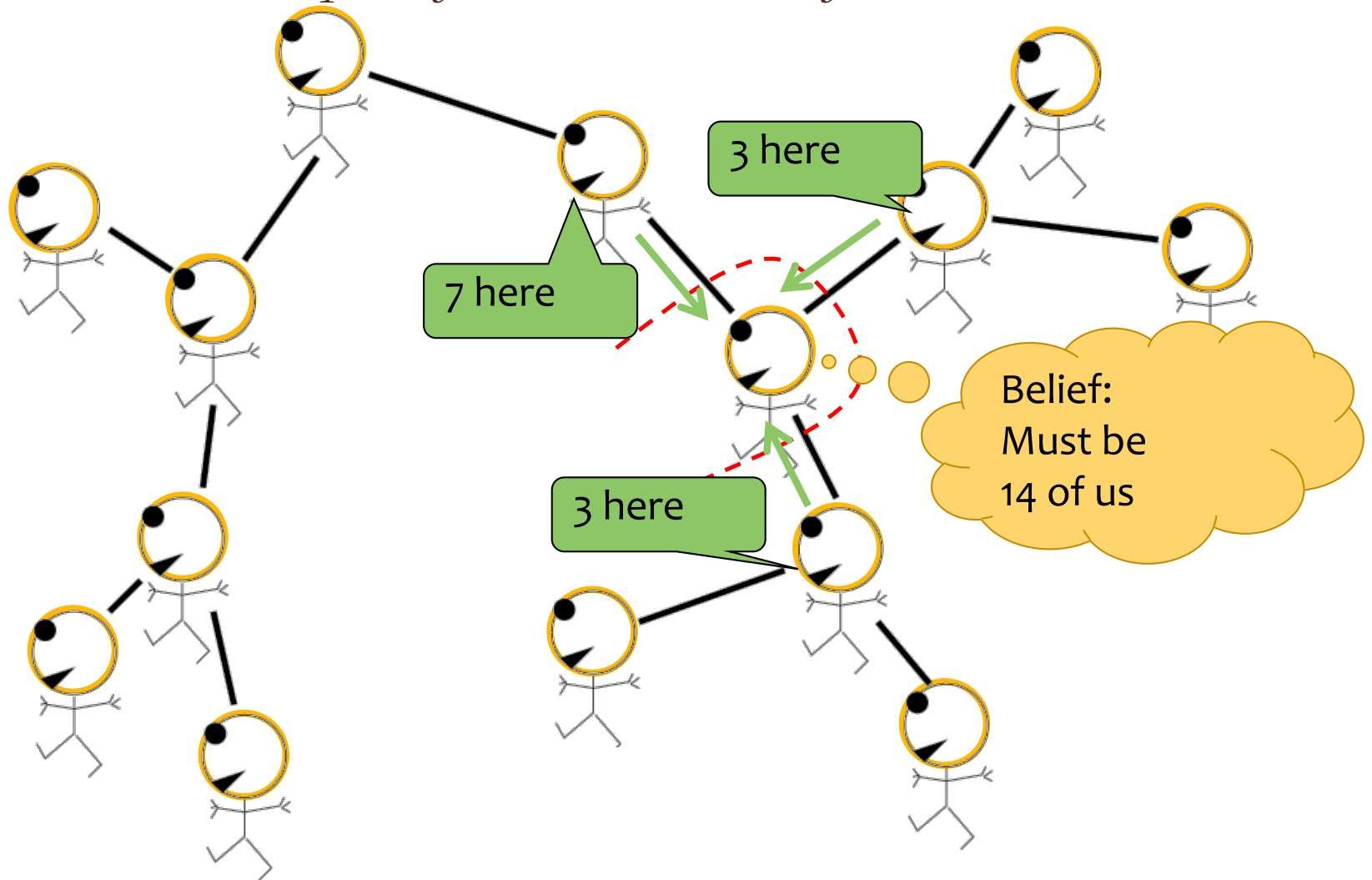
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



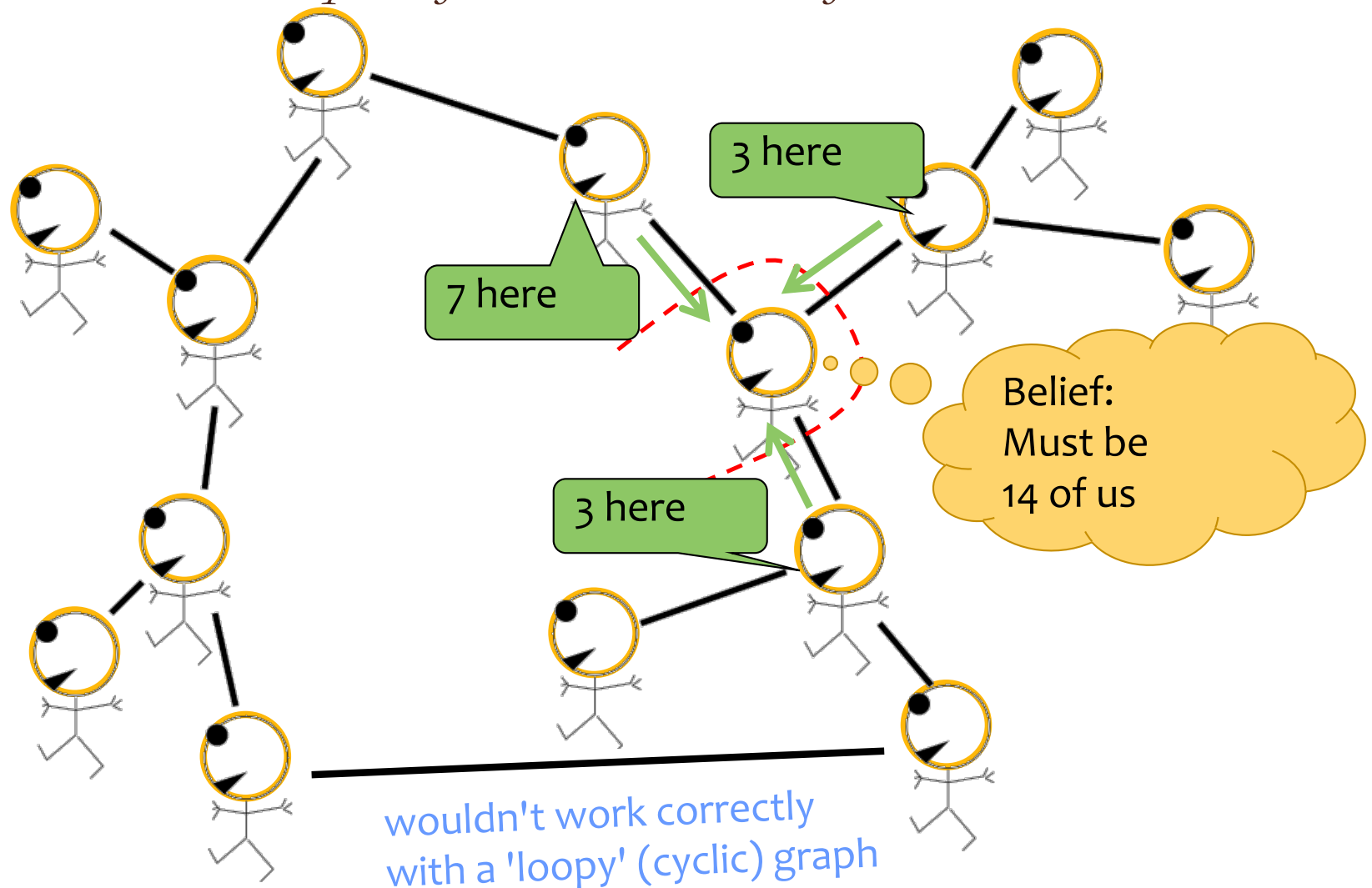
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



THE FORWARD-BACKWARD ALGORITHM

Inference for HMMs

















































Whiteboard

– Three Inference Problems for an HMM

1. Evaluation: Compute the probability of a given sequence of observations
2. 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

Dataset for Supervised Part-of-Speech (POS) Tagging

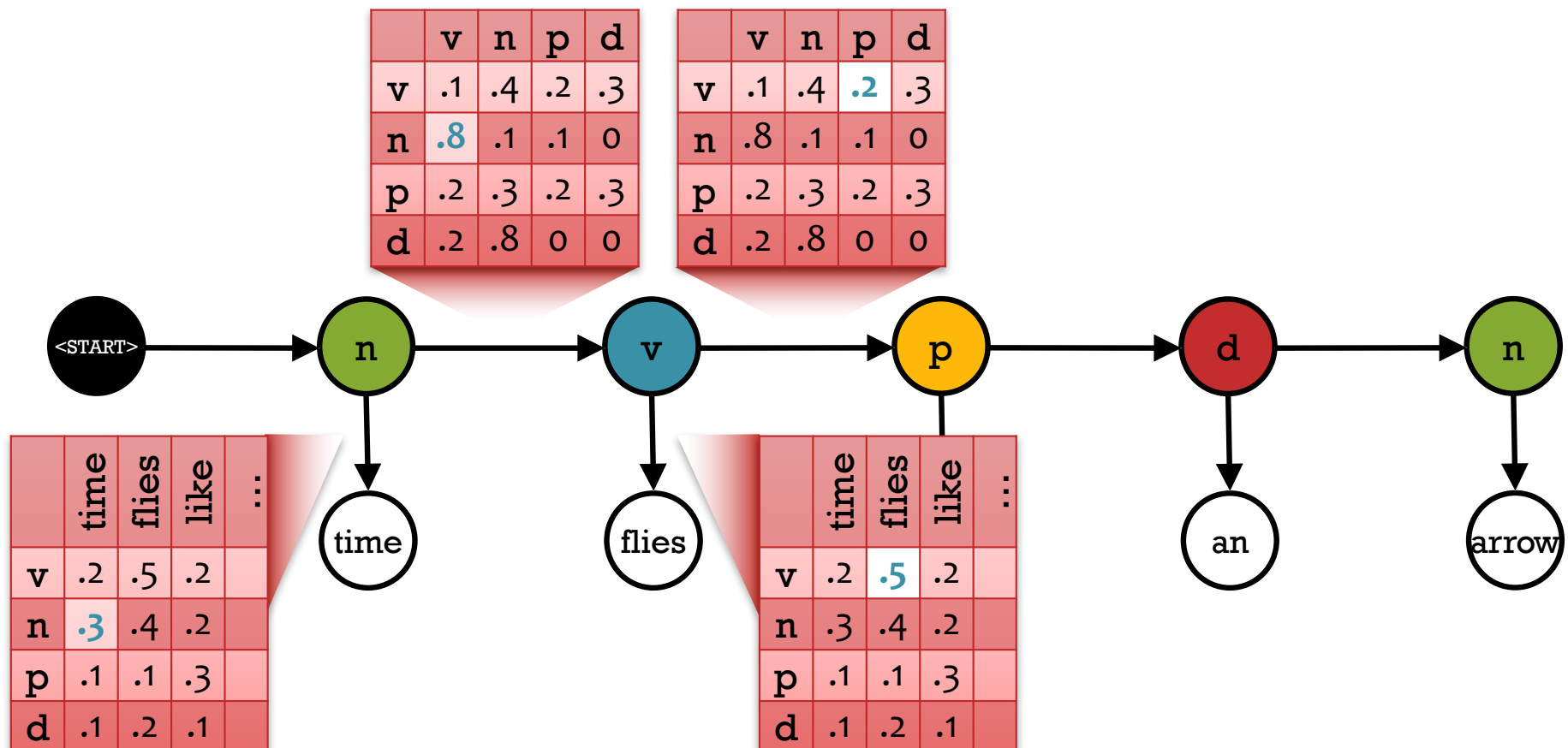
Data: $\mathcal{D} = \{x^{(n)}, y^{(n)}\}_{n=1}^N$

Sample 1:							$y^{(1)}$
							$x^{(1)}$
Sample 2:							$y^{(2)}$
							$x^{(2)}$
Sample 3:							$y^{(3)}$
							$x^{(3)}$
Sample 4:							$y^{(4)}$
							$x^{(4)}$

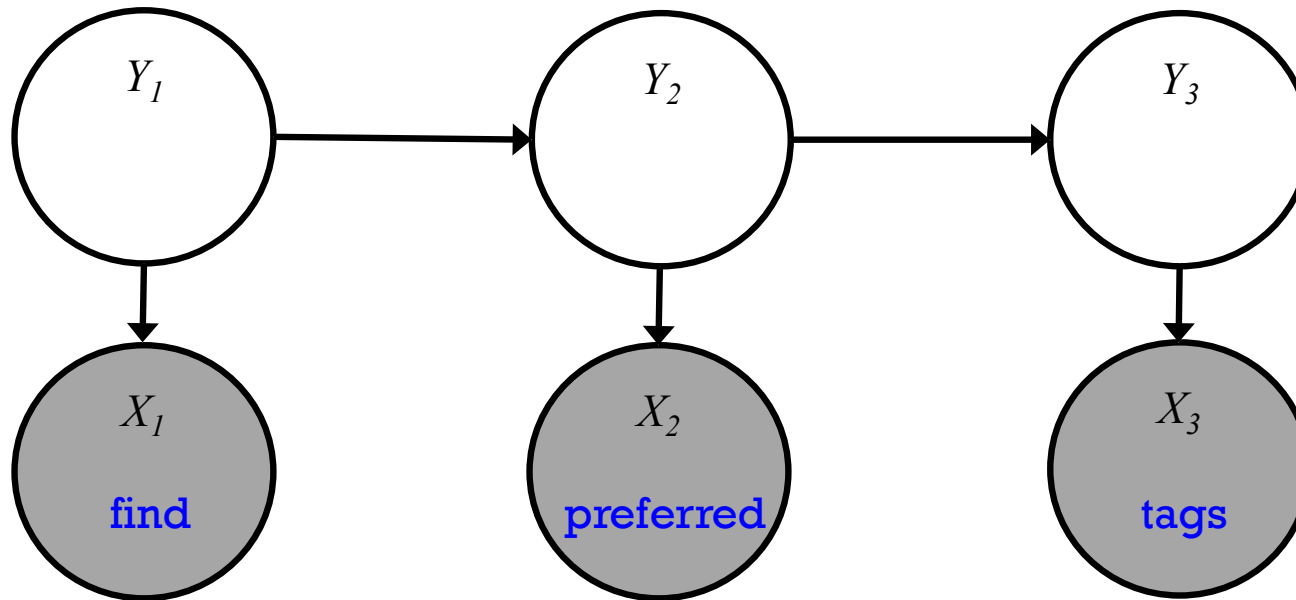
Hidden Markov Model

A Hidden Markov Model (HMM) provides a joint distribution over the the sentence/tags with an assumption of dependence between adjacent tags.

$$p(n, v, p, d, n, \text{time}, \text{flies}, \text{like}, \text{an}, \text{arrow}) = (.3 * .8 * .2 * .5 * \dots)$$



Forward-Backward Algorithm

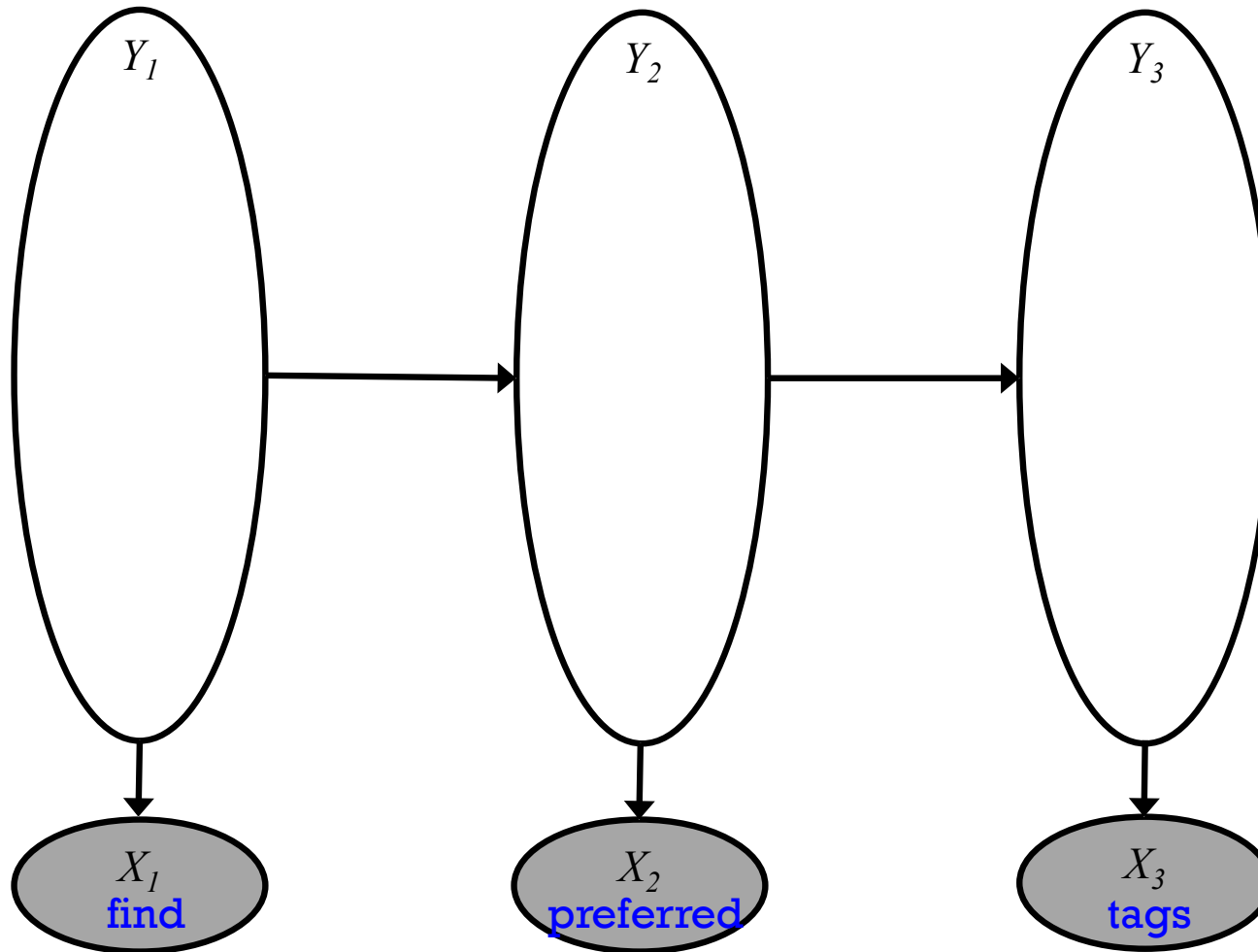


Could be verb or noun

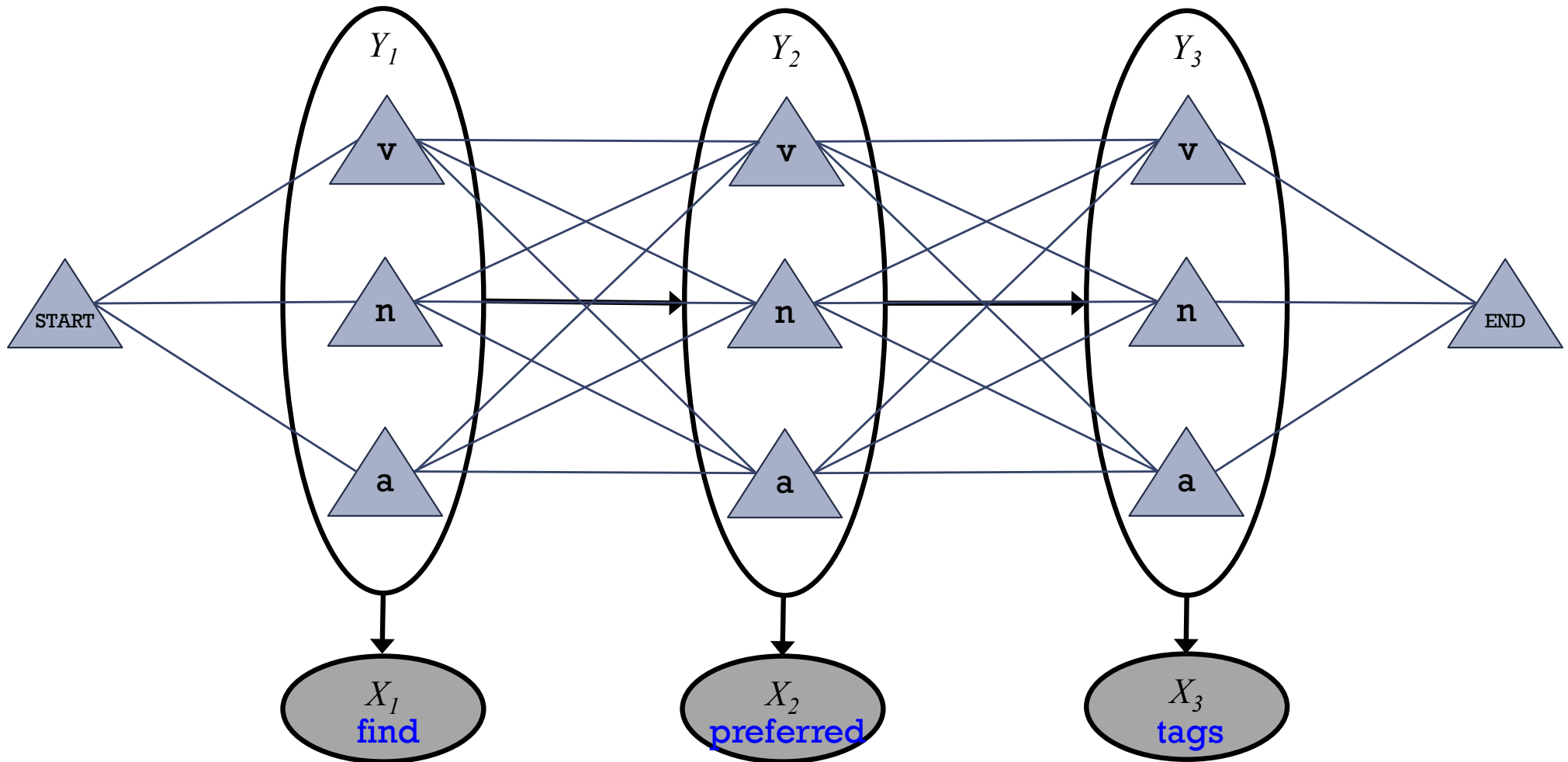
Could be adjective or verb

Could be noun or verb

Forward-Backward Algorithm

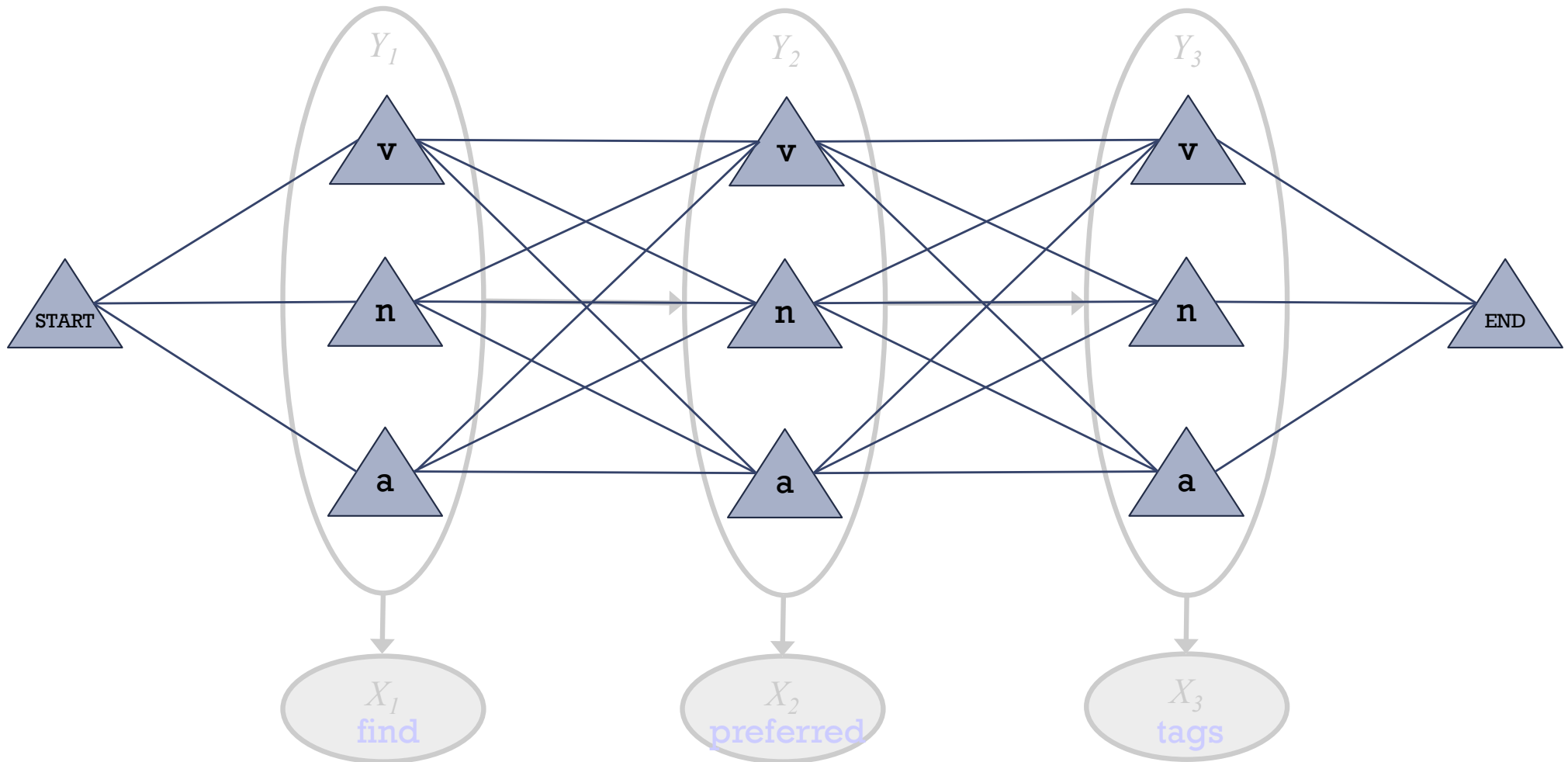


Forward-Backward Algorithm



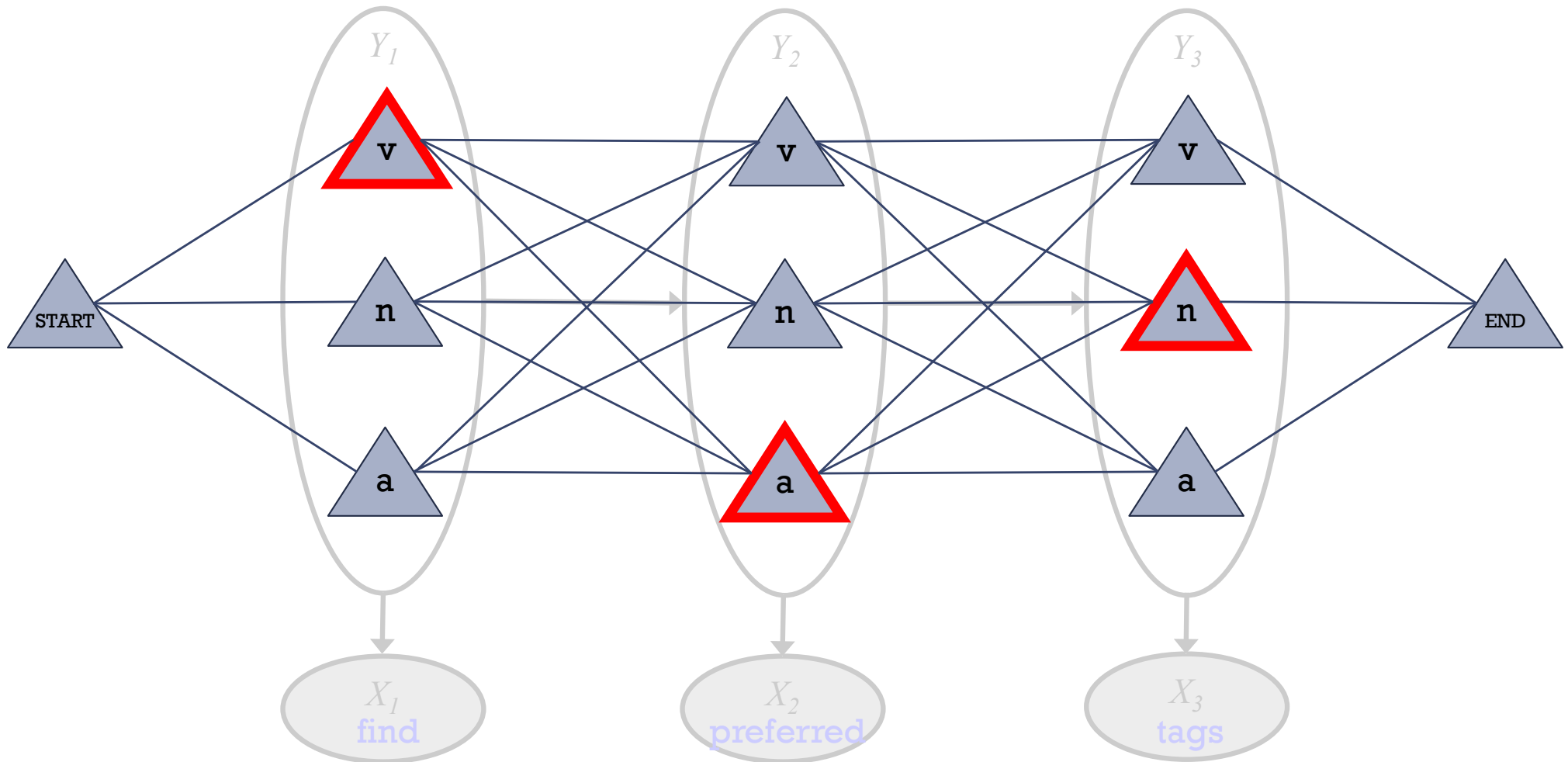
- Let's show the possible values for each variable

Forward-Backward Algorithm



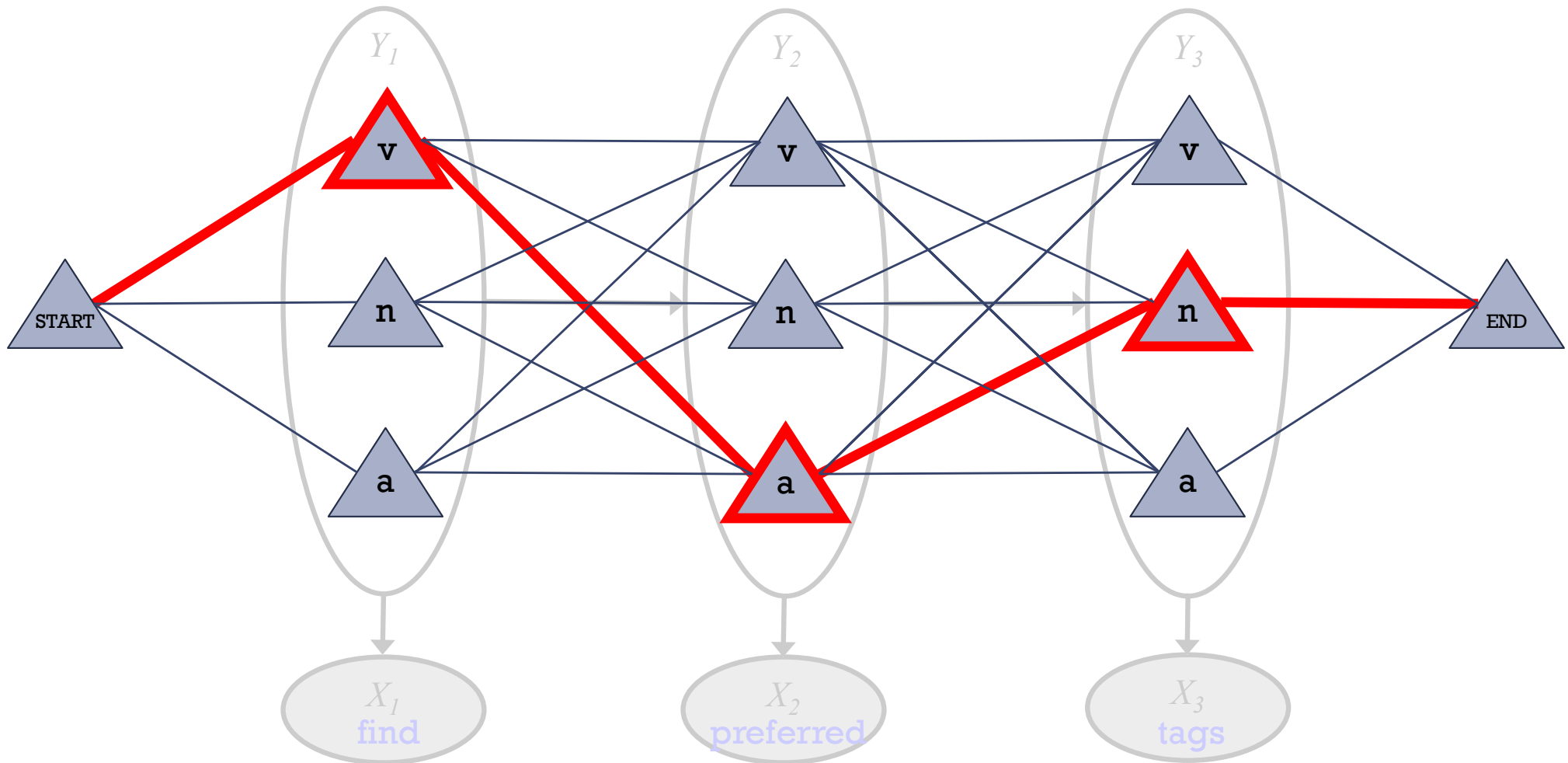
- Let's show the possible *values* for each variable

Forward-Backward Algorithm



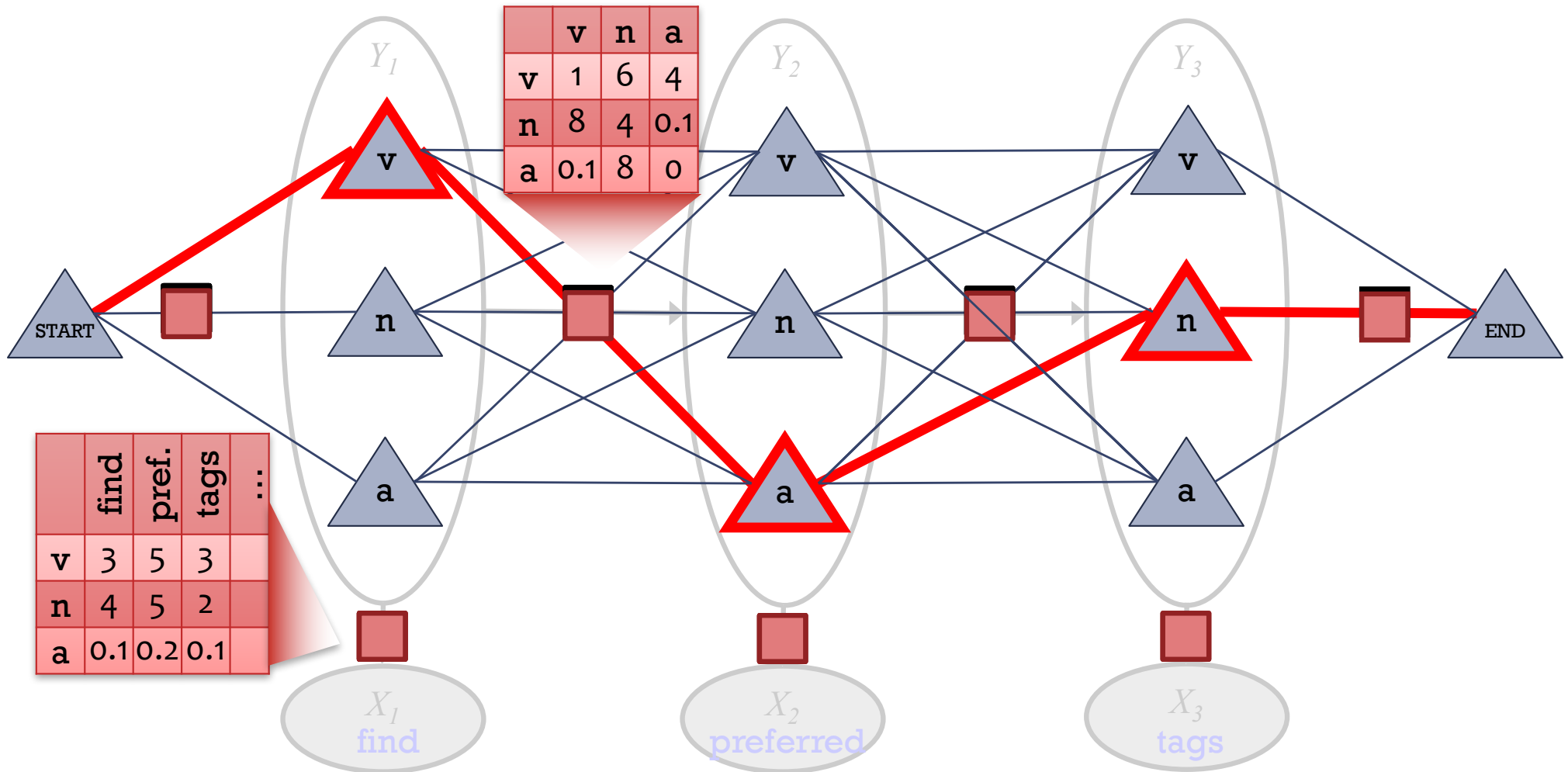
- Let's show the possible *values* for each variable
- One possible assignment

Forward-Backward Algorithm



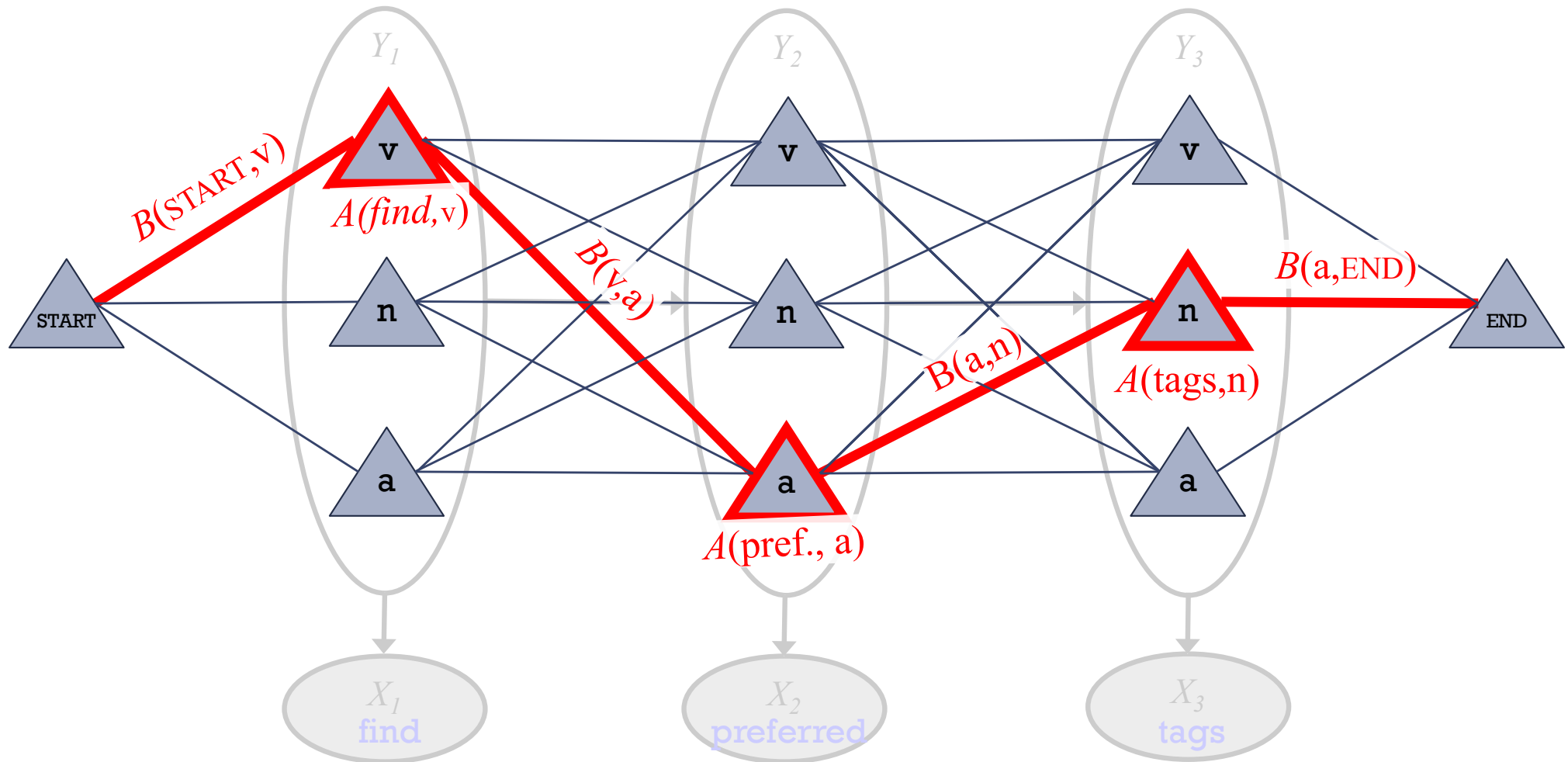
- Let's show the possible *values* for each variable
- One possible assignment
- And what the 7 transition / emission factors **think of it** ...

Forward-Backward Algorithm



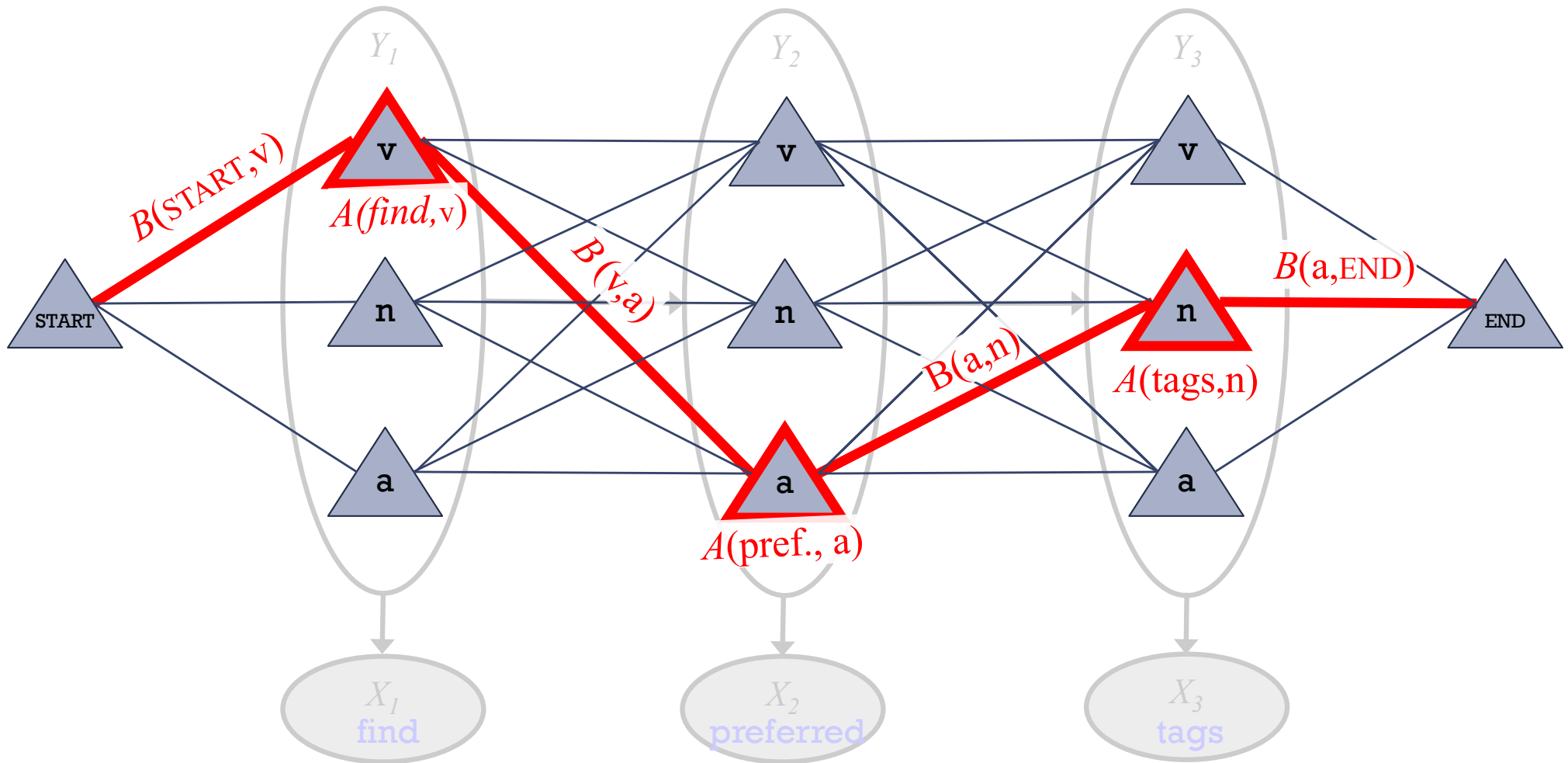
- Let's show the possible *values* for each variable
- One possible assignment
- And what the 7 transition / emission factors **think of it** ...

Viterbi Algorithm: Most Probable Assignment



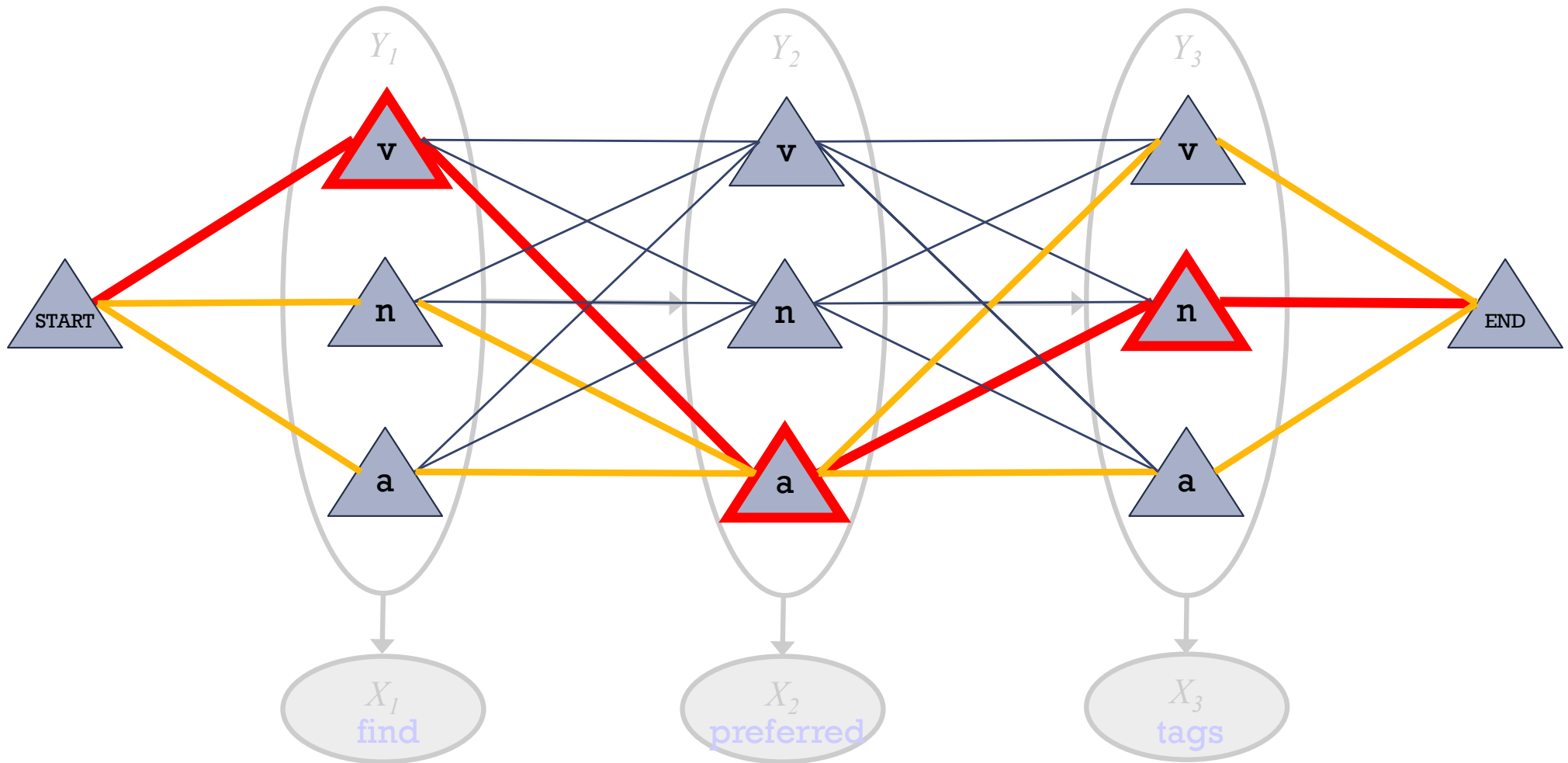
- So $p(\mathbf{v a n}) = (1/Z) * \text{product of 7 numbers}$
- Numbers associated with edges and nodes of path
- Most probable assignment = **path with highest product**⁶⁰

Viterbi Algorithm: Most Probable Assignment

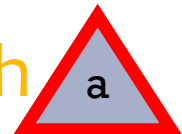


- So $p(\mathbf{v a n}) = (1/Z) * \text{product weight of one path}$

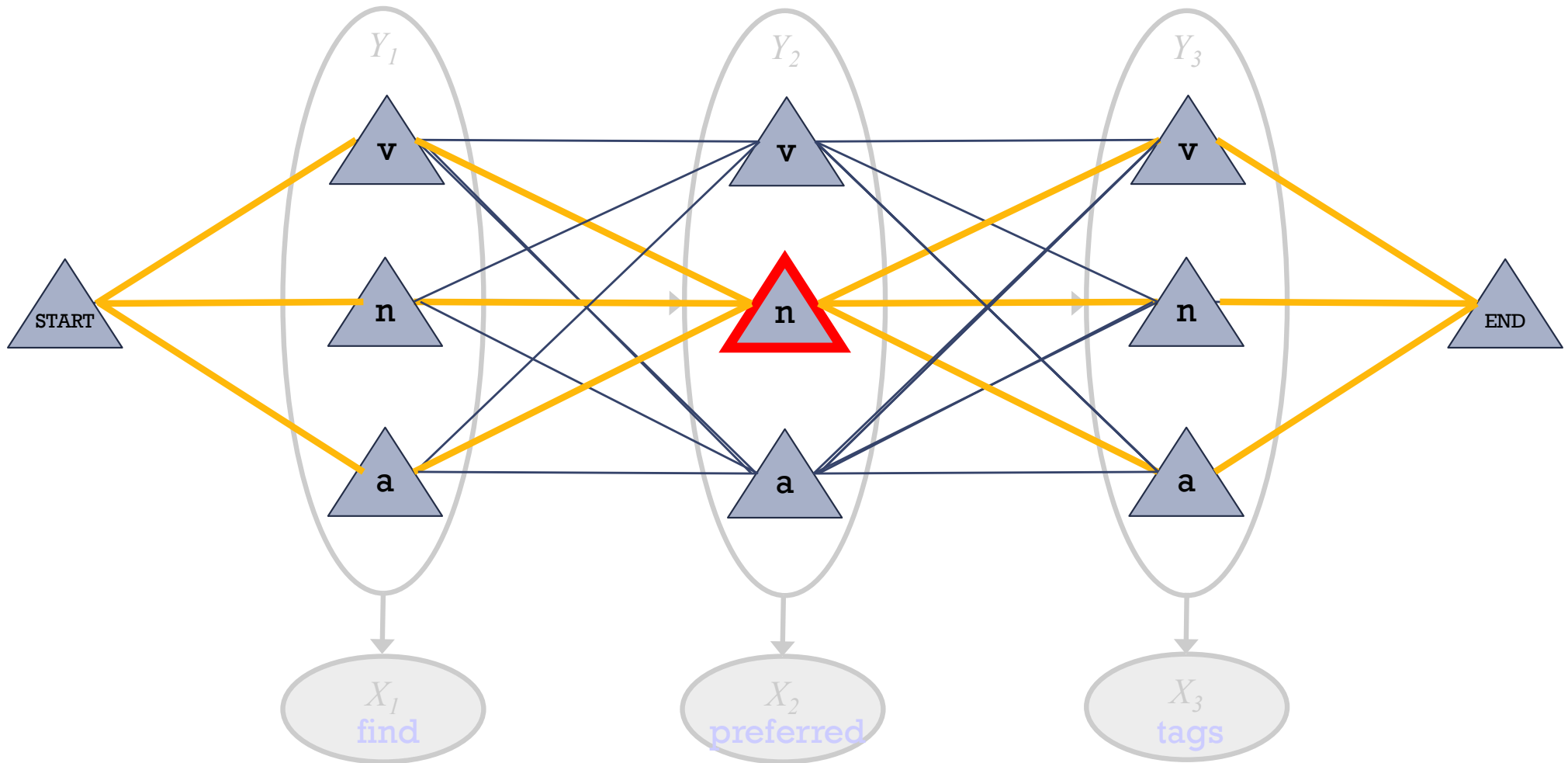
Forward-Backward Algorithm: Finds Marginals



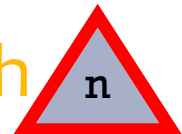
- So $p(\mathbf{v} \mathbf{a} \mathbf{n}) = (1/Z) * \text{product weight of one path}$
- Marginal probability $p(Y_2 = a)$
= $(1/Z) * \text{total weight of all paths through } \mathbf{a}$



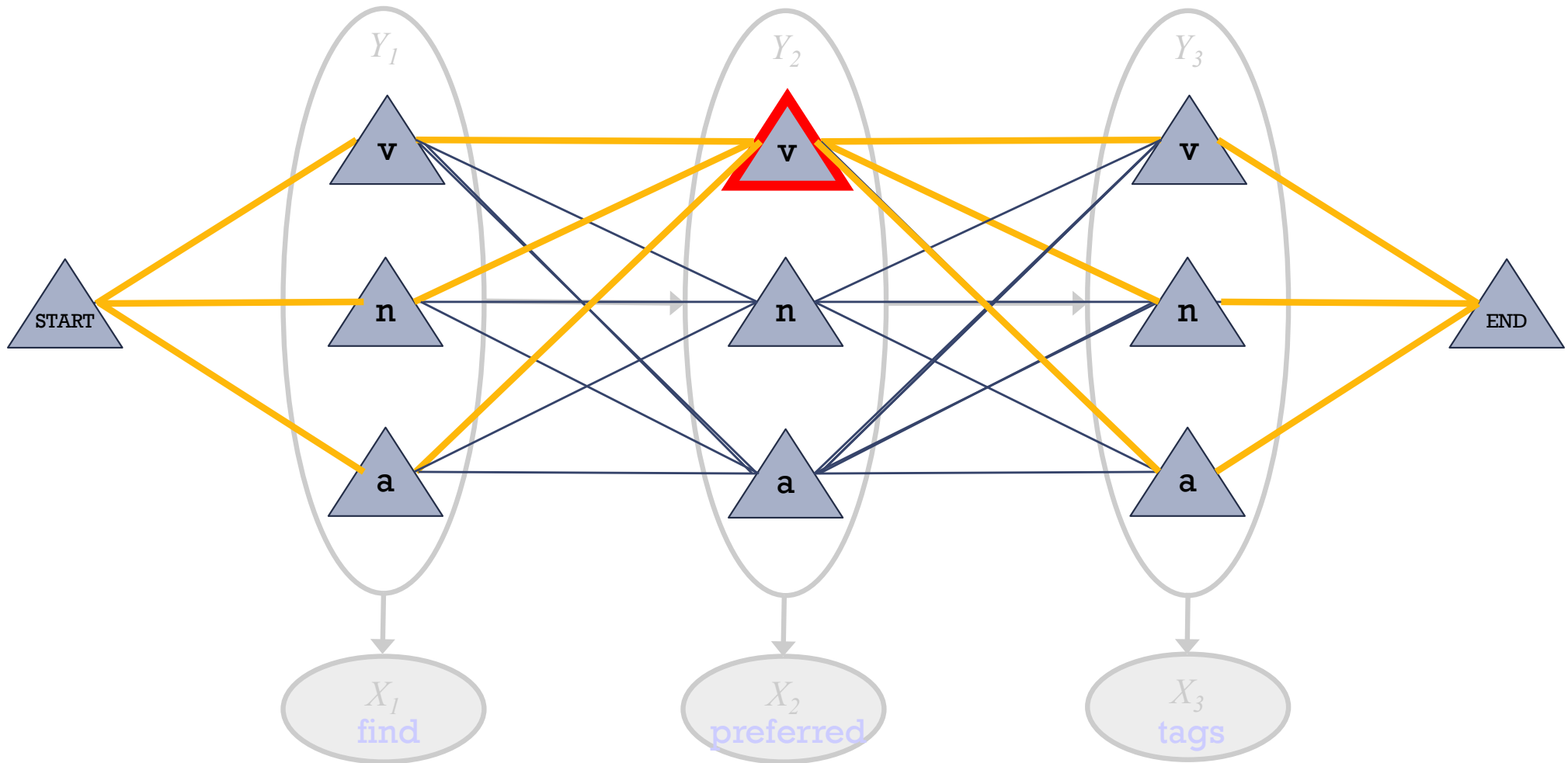
Forward-Backward Algorithm: Finds Marginals



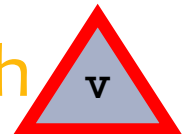
- So $p(\mathbf{v} \mathbf{a} \mathbf{n}) = (1/Z) * \text{product weight of one path}$
- Marginal probability $p(Y_2 = \mathbf{n}) = (1/Z) * \text{total weight of all paths through } \mathbf{n}$



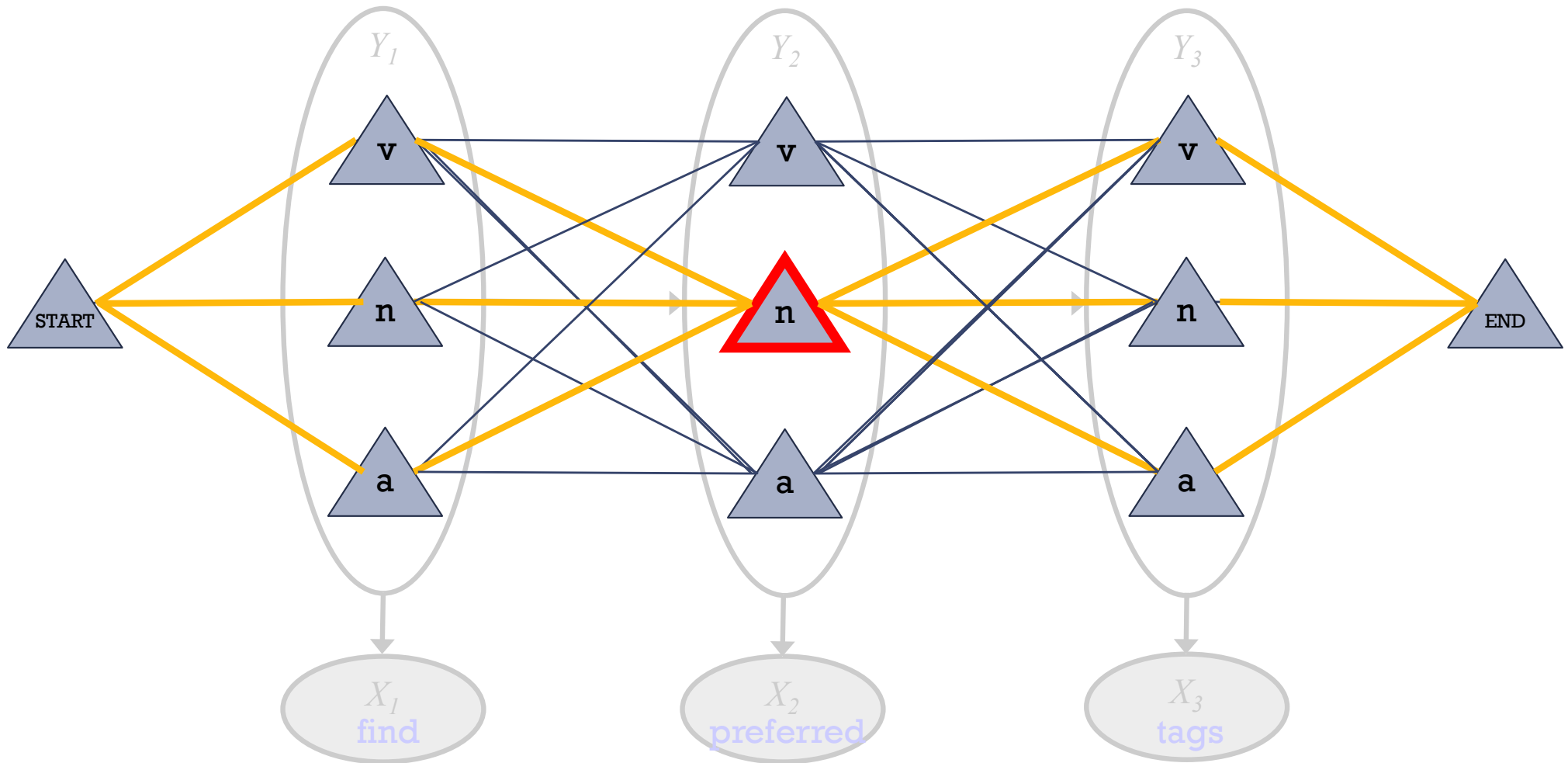
Forward-Backward Algorithm: Finds Marginals



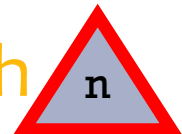
- So $p(\mathbf{v} \ \mathbf{a} \ \mathbf{n}) = (1/Z) * \text{product weight of one path}$
- Marginal probability $p(Y_2 = \mathbf{v}) = (1/Z) * \text{total weight of all paths through } \mathbf{v}$



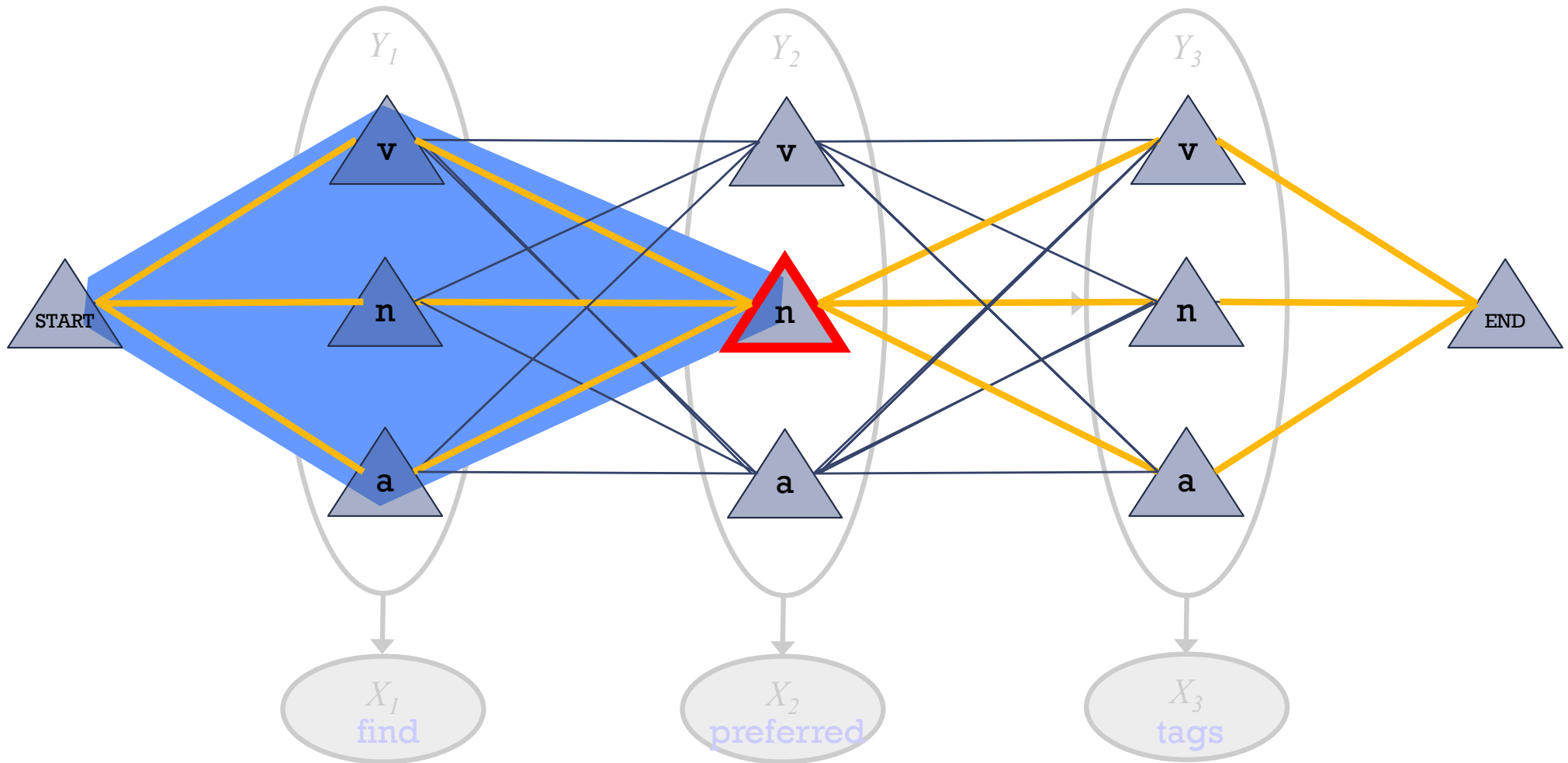
Forward-Backward Algorithm: Finds Marginals



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



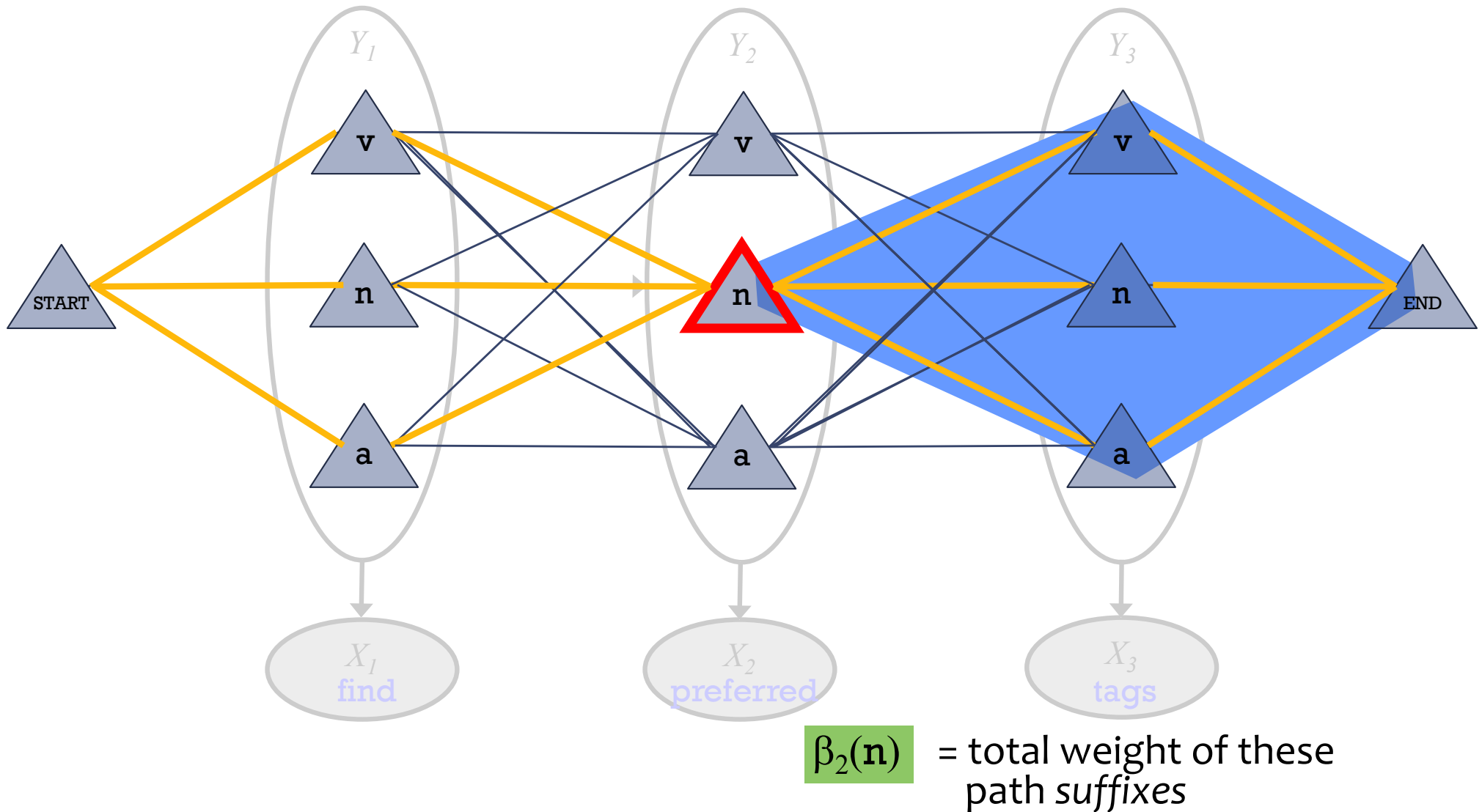
Forward-Backward Algorithm: Finds Marginals



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

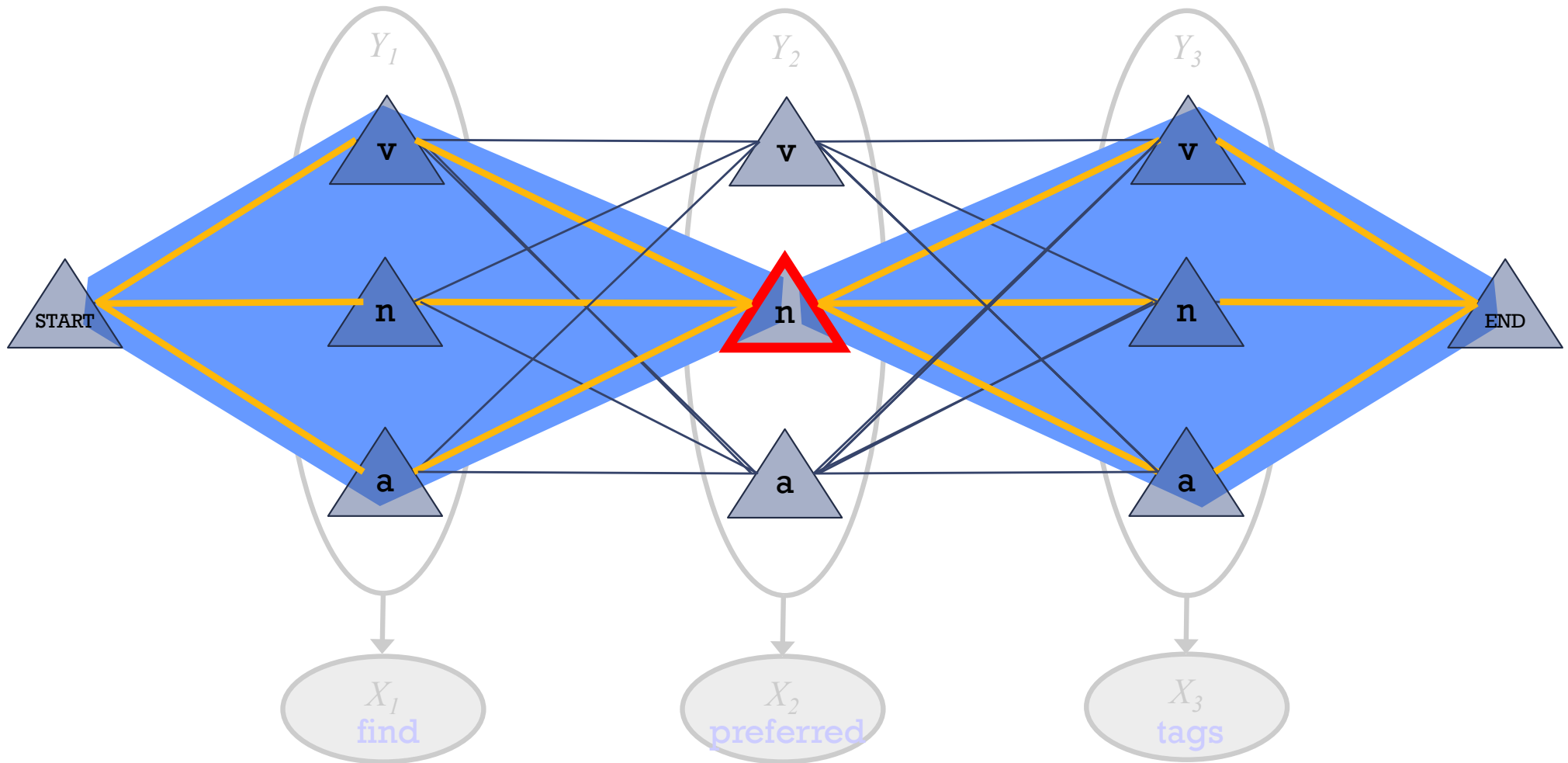
(found by dynamic programming: matrix-vector products)

Forward-Backward Algorithm: Finds Marginals



(found by dynamic programming: matrix-vector products)

Forward-Backward Algorithm: Finds Marginals



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

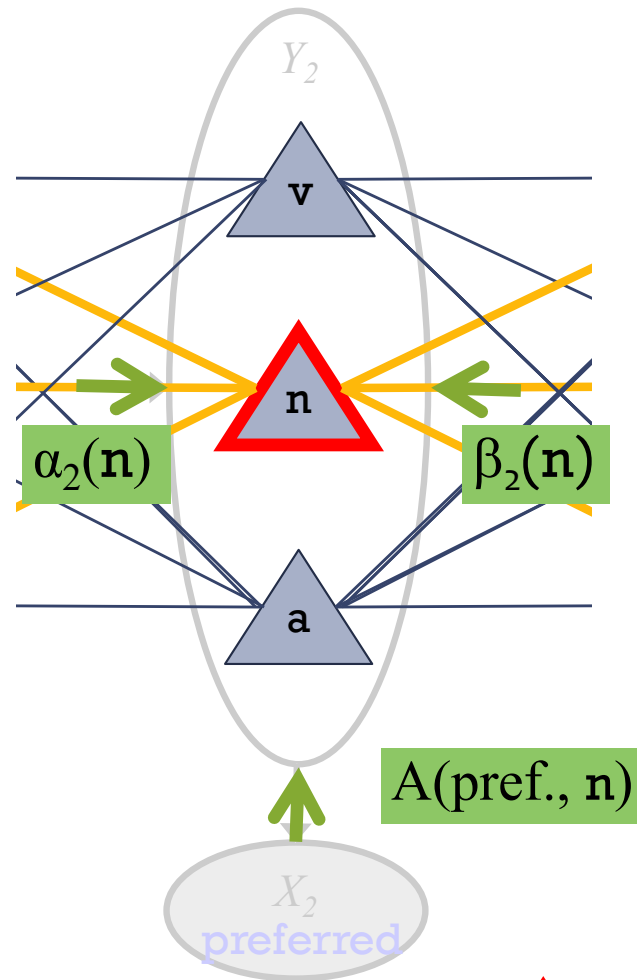
$\beta_2(\mathbf{n})$ = total weight of these path suffixes $(x + y + z)$

Product gives $ax+ay+az+bx+by+bz+cx+cy+cz$ = total weight of paths ⁶⁸

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.

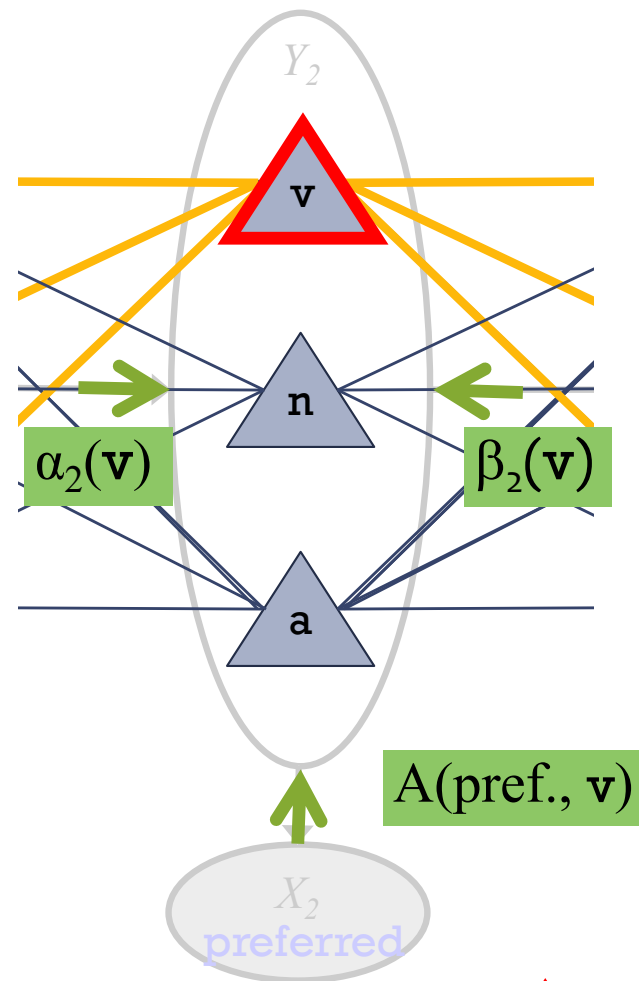


“belief that $Y_2 = \mathbf{n}$ ”

total weight of *all paths through* 

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

Forward-Backward Algorithm: Finds Marginals



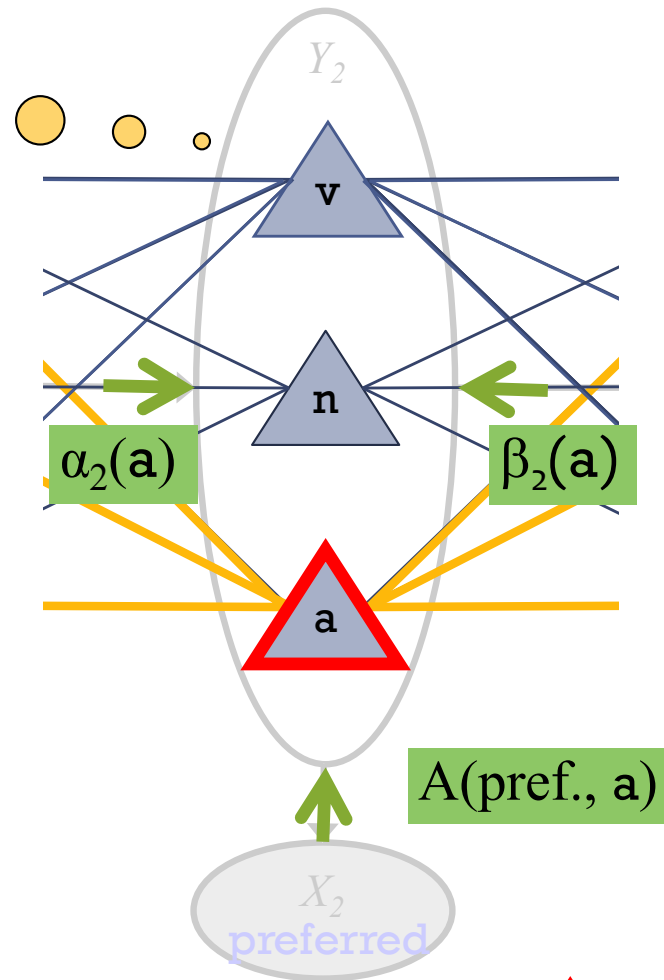
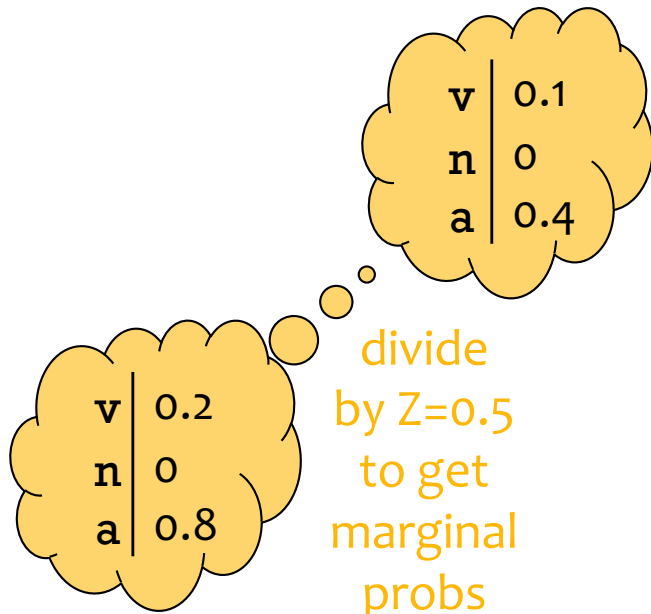
“belief that $Y_2 = \mathbf{v}$ ”

“belief that $Y_2 = \mathbf{n}$ ”

total weight of *all paths through* 

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

Forward-Backward Algorithm: Finds Marginals



“belief that $Y_2 = \mathbf{v}$ ”

“belief that $Y_2 = \mathbf{n}$ ”

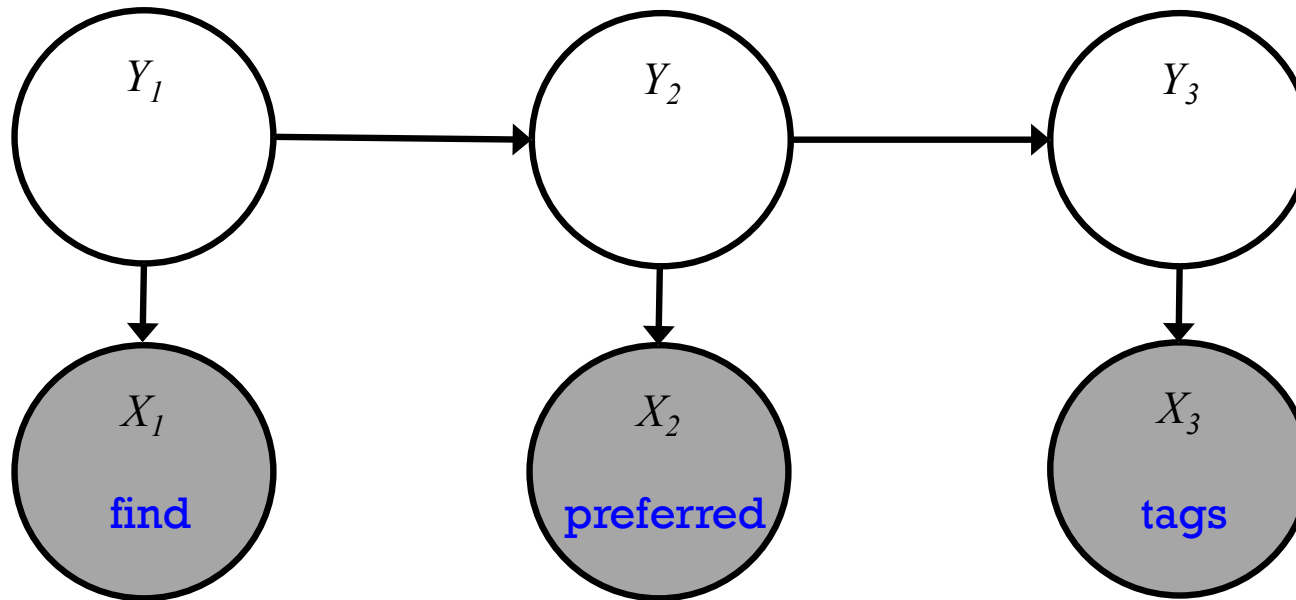
“belief that $Y_2 = \mathbf{a}$ ”

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

total weight of *all* paths through 

$$= \alpha_2(\mathbf{a}) \cdot A(\text{pref.}, \mathbf{a}) \cdot \beta_2(\mathbf{a})$$

Forward-Backward Algorithm



Could be verb or noun

Could be adjective or verb

Could be noun or verb

Inference for HMMs

Whiteboard

- Derivation of Forward algorithm
- Forward-backward algorithm
- Viterbi algorithm

Derivation of Forward Algorithm

Definition: $\alpha_t(k) \triangleq p(x_1, \dots, x_t, y_t = k)$

Derivation:

$$\begin{aligned}\alpha_T(\text{END}) &= p(x_1, \dots, x_T, y_T = \text{END}) \\ &= p(x_1, \dots, x_T | y_T) p(y_T) \quad \leftarrow \text{by def. of joint} \\ &= p(x_T | y_T) p(x_1, \dots, x_{T-1} | y_T) p(y_T) \quad \leftarrow \text{by cond. indep. of HMM} \\ &= p(x_T | y_T) p(x_1, \dots, x_{T-1}, y_T) \quad \leftarrow \text{by def. of joint} \\ &= p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \dots, x_{T-1}, y_{T-1}, y_T) \quad \leftarrow \text{by def. of marginal} \\ &= p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \dots, x_{T-1}, y_T | y_{T-1}) p(y_{T-1}) \quad \leftarrow \text{by def. of joint} \\ &= p(x_T | y_T) \sum_{y_{T-1}} \underbrace{p(x_1, \dots, x_{T-1} | y_{T-1})}_{\leftarrow \text{by cond. indep. of HMM}} p(y_T | y_{T-1}) p(y_{T-1}) \quad \leftarrow \text{by cond. indep. of HMM} \\ &= p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \dots, x_{T-1}, y_{T-1}) p(y_T | y_{T-1}) \quad \leftarrow \text{by def. of joint} \\ &= p(x_T | y_T) \sum_{y_{T-1}} \alpha_{T-1}(y_{T-1}) p(y_T | y_{T-1}) \quad \leftarrow \text{by def. of } \alpha_t(k)\end{aligned}$$

Herein using "y_T" as shorthand for "y_T = END"

Forward-Backward Algorithm

Define: $\alpha_t(k) \triangleq p(x_1, \dots, x_t, y_t = k)$
 $\beta_t(k) \triangleq p(x_{t+1}, \dots, x_T | y_t = k)$

Assume $y_0 = \text{START}$
 $y_{T+1} = \text{END}$

① Initialize $\alpha_0(\text{START}) = 1$ $\alpha_0(k) = 0 \quad \forall k \neq \text{START}$
 $\beta_{T+1}(\text{END}) = 1$ $\beta_{T+1}(k) = 0 \quad \forall k \neq \text{END}$

② For $t = 1, \dots, T$:

For $k = 1, \dots, K$:

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

the alphas include the emission probabilities
so we don't multiply them in separately

③ For $t = T, \dots, 1$:

For $k = 1, \dots, K$:

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

④ Compute $p(\vec{x}) = \alpha_{T+1}(\text{END})$

[Evaluation]

⑤ Compute $p(y_t = k | \vec{x}) = \frac{\alpha_t(k) \beta_t(k)}{p(\vec{x})}$

[Marginals]

Viterbi Algorithm

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

"backpointers" \rightarrow $b_t(k) \triangleq \arg \max_{y_1, \dots, y_{t-1}} p(x_1, \dots, x_t, y_1, \dots, y_{t-1}, y_t = k)$

Assume $y_0 = \text{START}$

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

② For $t = 1, \dots, T$:

For $k = 1, \dots, K$:

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

$$b_t(k) = \arg \max_{j \in \{1, \dots, K\}} p(x_t | y_t = k) \omega_{t-1}(j) p(y_t = k | y_{t-1} = j)$$

③ Compute Most Probable Assignment

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

For $t = T-1, \dots, 1$

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

[Decoding]

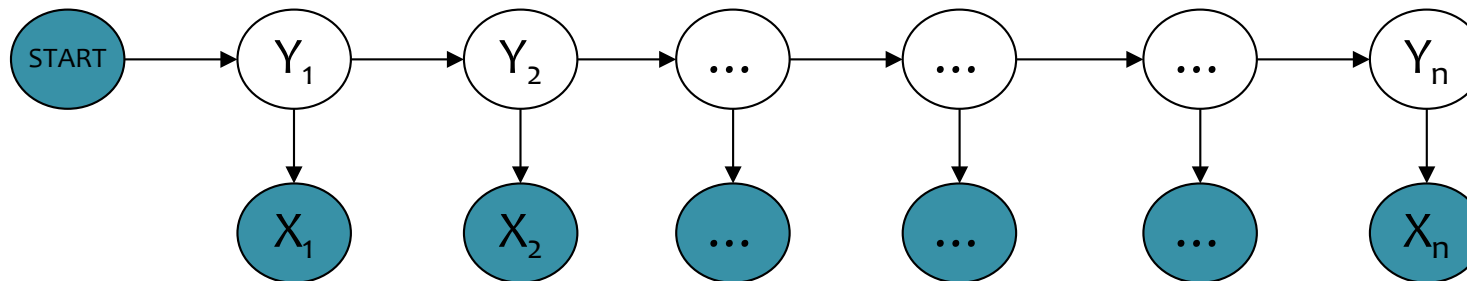
follow the
"backpointers"

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 $P(\mathbf{Y}, \mathbf{X})$, but in a prediction task, we need the conditional probability $P(\mathbf{Y}|\mathbf{X})$