10-301/601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Hidden Markov Models (Part II)

Matt Gormley & Henry Chai
Lecture 19
Oct. 29, 2021

Reminders

- Homework 6: Learning Theory / Generative Models
  - Out: Thu, Oct. 21
  - Due: Thu, Oct. 28 at 11:59pm
  - Same collaboration policy as Homework 3
    - Opt-in to homework groups on Piazza
  - IMPORTANT: you may only use 2 grace days on Homework 6
    - Last posible moment to submit HW6: Sat, Oct. 30 at 11:59pm
- Midterm Exam 2
  - Tue, Nov. 2, 6:30pm – 8:30pm
- Practice for Exam 2
  - Practice problems released on course website
    - (Tentatively) Out: Thu, Oct. 21
  - Mock Exam 2
    - (Tentatively) Out: Thu, Oct. 28
    - Due Sun, Oct. 31 at 11:59pm

SUPERVISED LEARNING FOR HMMS

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)

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

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

   …

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

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 $\boldsymbol{\theta}^{MLE}$

MLE of Categorical Distribution

1. Suppose we have a **dataset** obtained by repeatedly rolling a M-sided (weighted) die N times. That is, we have data

$$\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$$

   where $x^{(i)} \in \{1, \ldots, M\}$ and $x^{(i)} \sim \text{Categorical}(\phi)$.

2. A random variable is **Categorical** written $X \sim \text{Categorical}(\phi)$ iff

$$P(X = x) = p(x; \phi) = \phi_x$$

   where $x \in \{1, \ldots, M\}$ and $\sum_{m=1}^{M} \phi_m = 1$. The **log-likelihood** of the data becomes:

$$\ell(\phi) = \sum_{i=1}^{N} \log \phi_{x^{(i)}} \text{ s.t. } \sum_{m=1}^{M} \phi_m = 1$$

3. Solving this *constrained* optimization problem yields the **maximum likelihood estimator** (MLE):

$$\phi_m^{MLE} = \frac{N_{x=m}}{N} = \frac{\sum_{i=1}^{N} \mathbb{I}(x^{(i)} = m)}{N}$$

Hidden Markov Model

HMM Parameters:

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

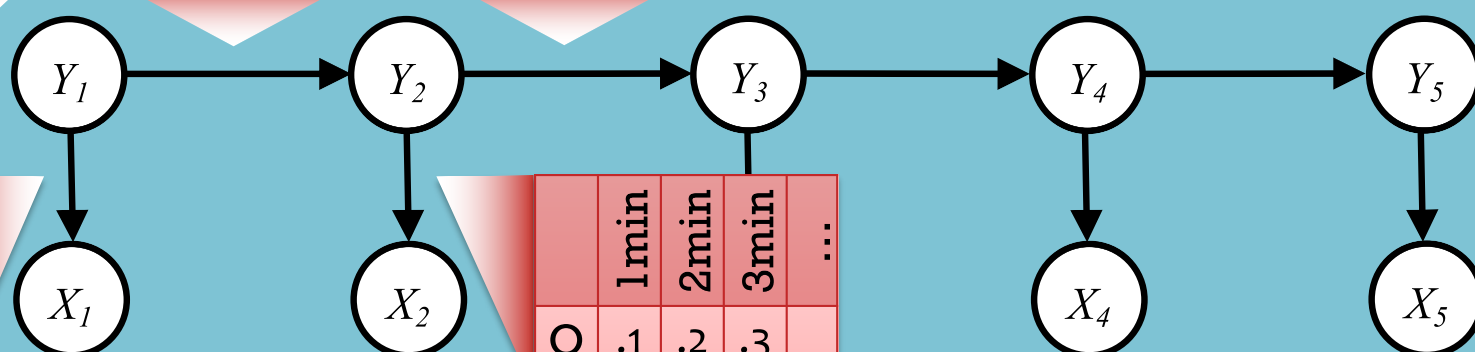
Transition matrix, $\mathbf{B}$, where $P(Y_t = k | Y_{t-1} = j) = B_{j,k}, \forall t, k$

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

O	.8
S	.1
C	.1

	O	S	C
O	.9	.08	.02
S	.2	.7	.1
C	.9	0	.1

	O	S	C
O	.9	.08	.02
S	.2	.7	.1
C	.9	0	.1



	1min	2min	3min	…
O	.1	.2	.3	
S	.01	.02	.03	
C	0	0	0	

	1min	2min	3min	…
O	.1	.2	.3	
S	.01	.02	.03	
C	0	0	0	

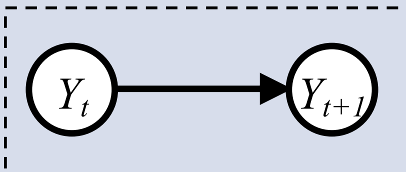
Training HMMs

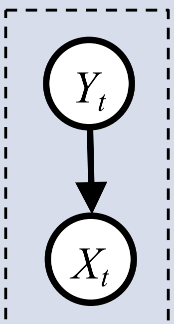
Whiteboard

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

Supervised Learning for HMMs

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





Data: $\mathcal{D} = \{(\vec{x}^{(i)}, \vec{y}^{(i)})\}_{i=1}^{N}$ $\quad \vec{x} = [x_1, \ldots, x_T]^T$, $\quad \vec{y} = [y_1, \ldots, y_T]^T$

Likelihood:

$$\ell(A,B,C) = \sum_{i=1}^{N} \log p(\vec{x}^{(i)}, y^{(i)} \mid A,B,C)$$

$$= \sum_{i=1}^{N} \Big[\underbrace{\log p(y_1^{(i)} \mid C)}_{\text{initial}} + \underbrace{\Big(\sum_{t=2}^{T} \log p(y_t^{(i)} \mid y_{t-1}^{(i)}, B)\Big)}_{\text{transition}} + \underbrace{\Big(\sum_{t=1}^{T} \log p(x_t^{(i)} \mid y_t^{(i)}, A)\Big)}_{\text{emission}} \Big]$$

MLE:

$$\hat{A}, \hat{B}, \hat{C} = \operatorname*{argmax}_{A,B,C} \ell(A,B,C)$$

$$\Rightarrow \hat{C} = \operatorname*{argmax}_{C} \sum_{i=1}^{N} \log p(y_1^{(i)} \mid C)$$

$$\hat{B} = \operatorname*{argmax}_{B} \sum_{i=1}^{N} \sum_{t=2}^{T} \log p(y_t^{(i)} \mid y_{t-1}^{(i)}, B)$$

$$\hat{A} = \operatorname*{argmax}_{A} \sum_{i=1}^{N} \sum_{t=1}^{T} \log p(x_t^{(i)} \mid y_t^{(i)}, A)$$

Can solve in closed form, which yields...

$$\hat{C}_k = \frac{\#(y_1^{(i)} = k)}{N} \quad \forall i,k$$

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

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

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 $P(Y_t = k | Y_{t-1} = j) = B_{j,k}, \forall t, k$

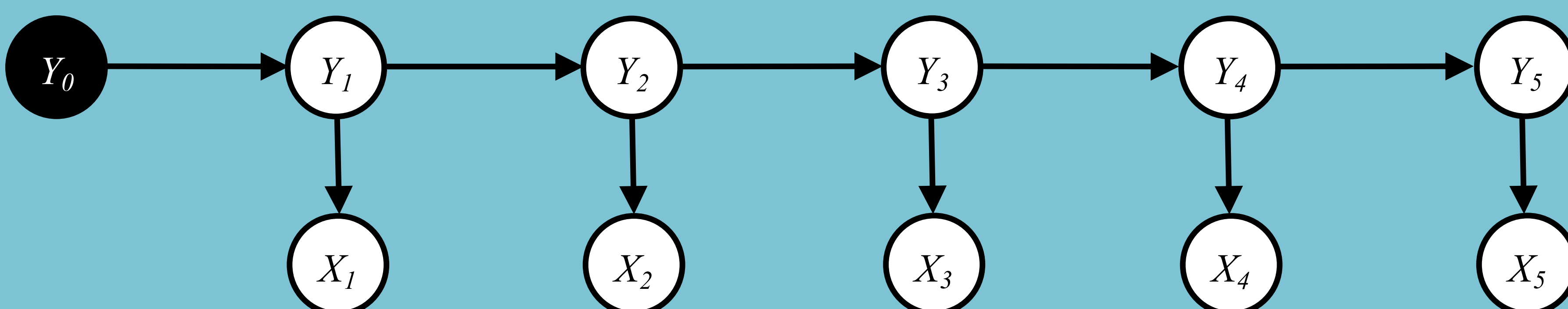
Assumption: $y_0 =$ START

Generative Story:

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

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

For notational convenience, we fold the *initial probabilities* **C** into the *transition matrix* **B** by our assumption.



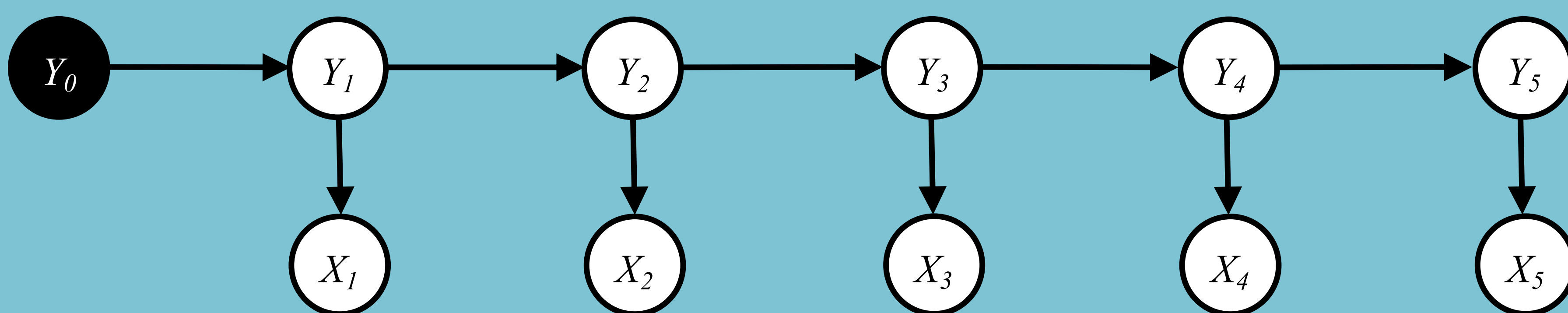
Hidden Markov Model

Joint Distribution:

$$y_0 = \text{START}$$

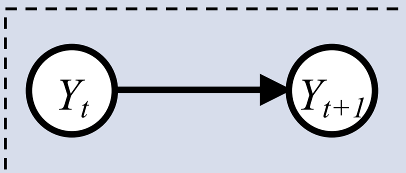
$$p(\mathbf{x}, \mathbf{y} | y_0) = \prod_{t=1}^{T} p(x_t | y_t) p(y_t | y_{t-1})$$

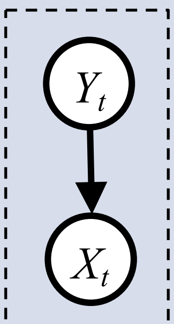
$$= \prod_{t=1}^{T} A_{y_t, x_t} B_{y_{t-1}, y_t}$$



Supervised Learning for HMMs

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





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

Likelihood:

$$\ell(A,B) = \sum_{i=1}^{N} \log p(\vec{x}^{(i)}, \vec{y}^{(i)})$$

$$= \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \log p(y_t^{(i)} \mid y_{t-1}^{(i)}, B) + \log p(x_t^{(i)} \mid y_t^{(i)}, A) \right]$$

MLE:

$$\hat{A}, \hat{B} = \operatorname{argmax} \ \ell(A,B)$$

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

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

↑ can solve in closed form to get...

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

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

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 = $\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ 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
- …

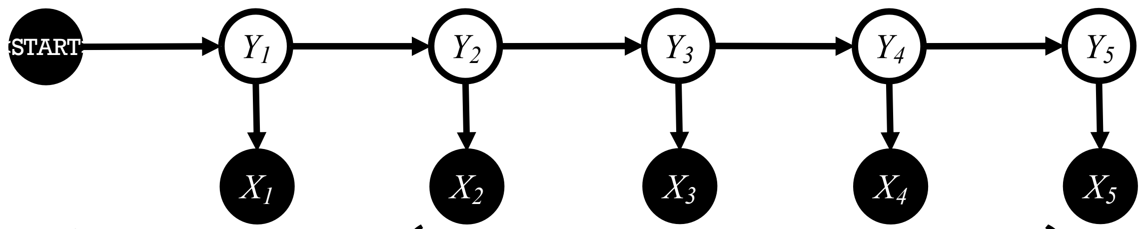




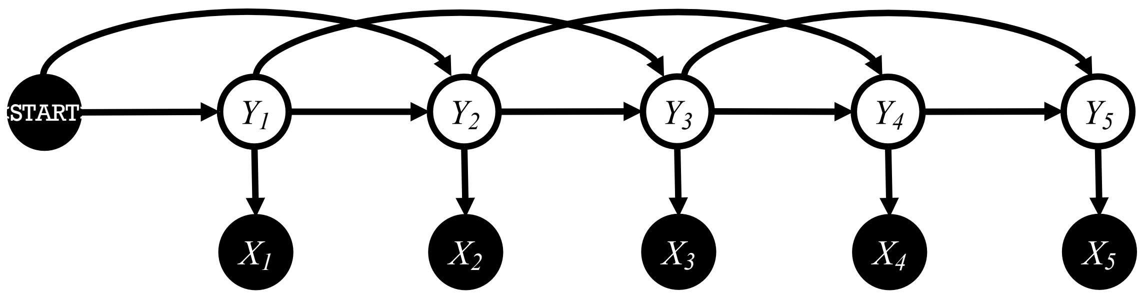
Slide from William Cohen

Higher-order HMMs

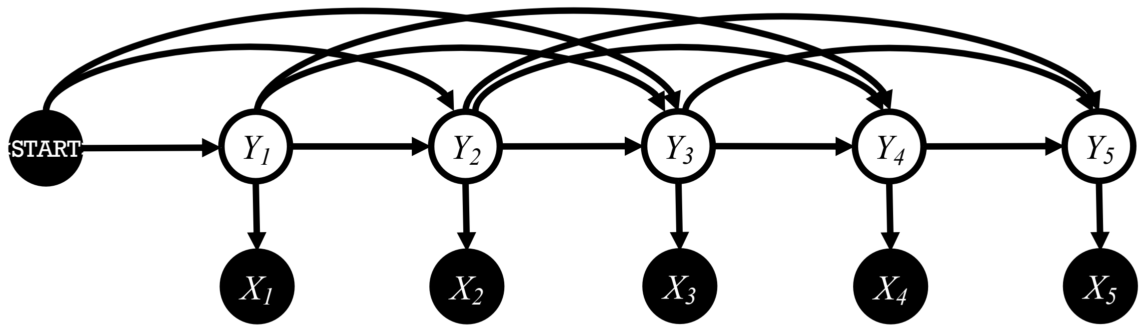
- 1st-order HMM (i.e. bigram HMM)



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

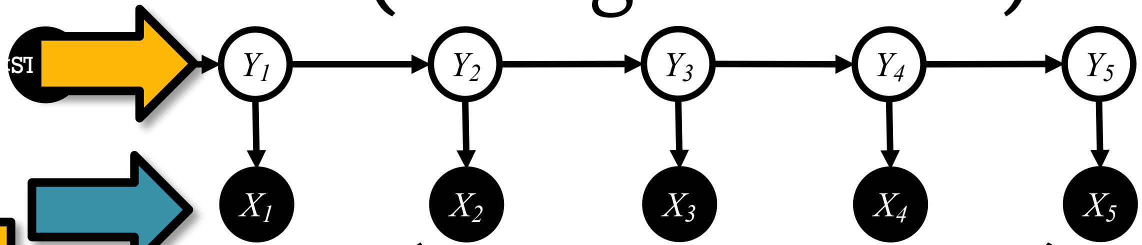


- 3rd-order HMM

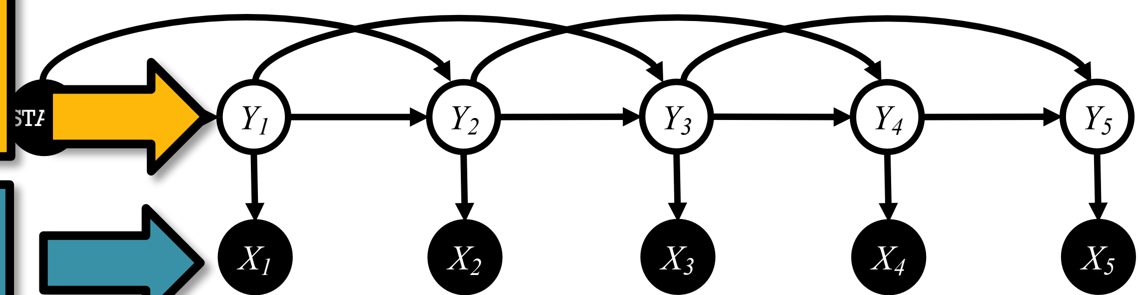


Higher-order HMMs

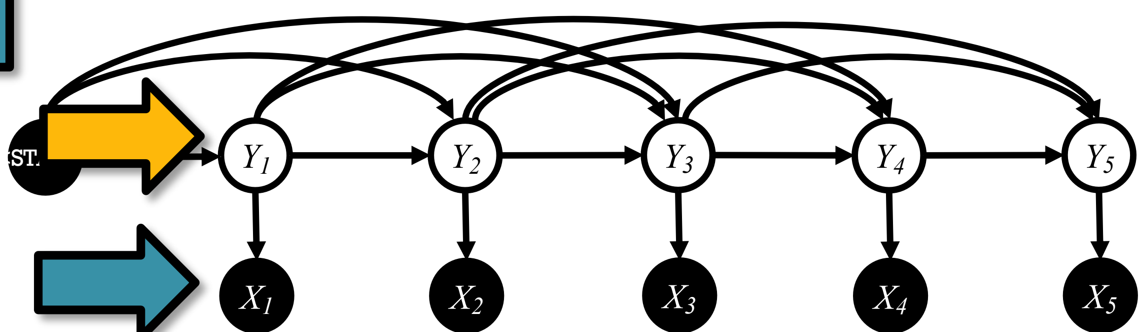
- 1^{st}-order HMM (i.e. bigram HMM)



- rder HMM (i.e. trigram HMM)



- rder HMM



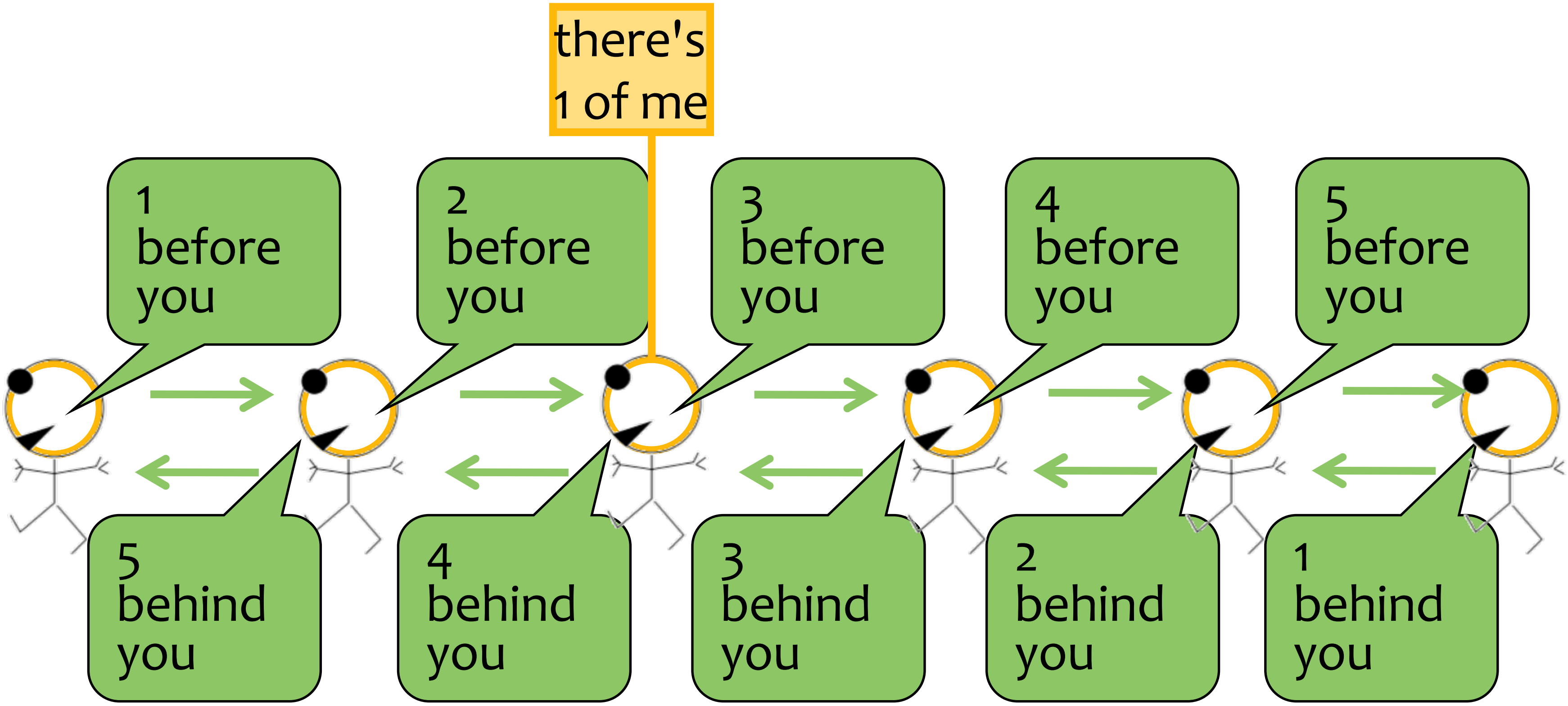
Hidden States, **y**

Observa-tions, **x**

BACKGROUND: MESSAGE PASSING

Great Ideas in ML: Message Passing

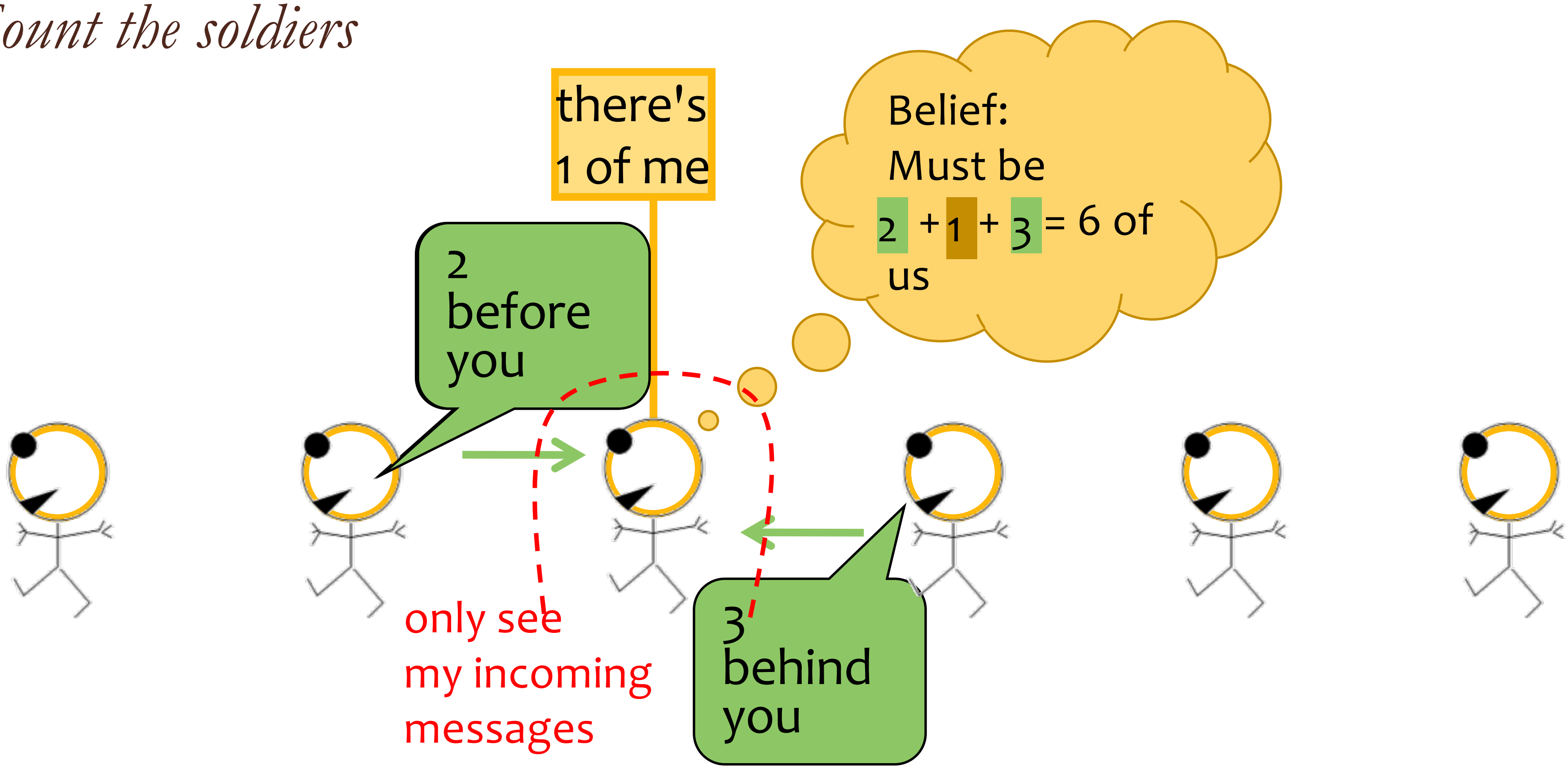
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

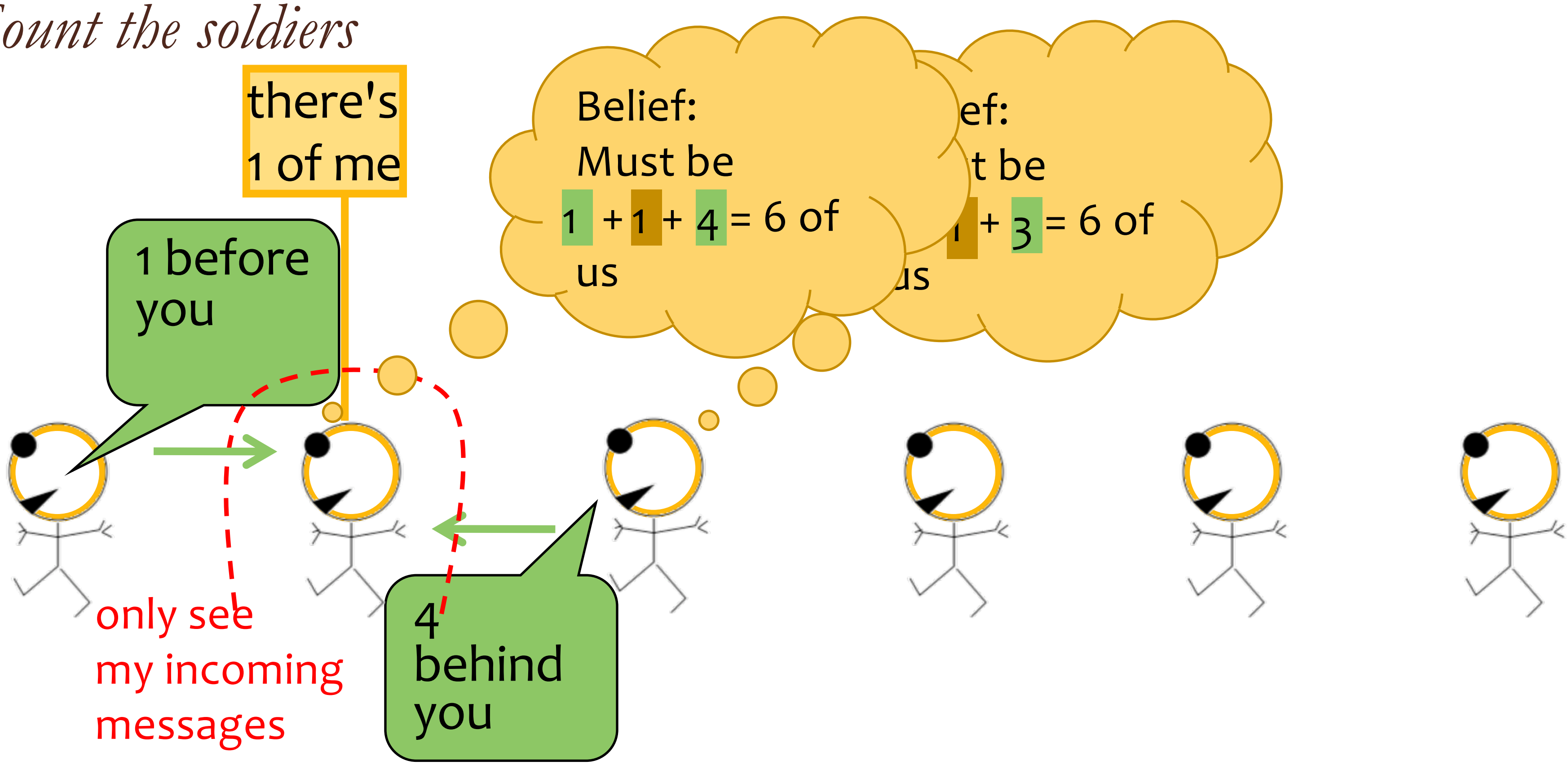
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

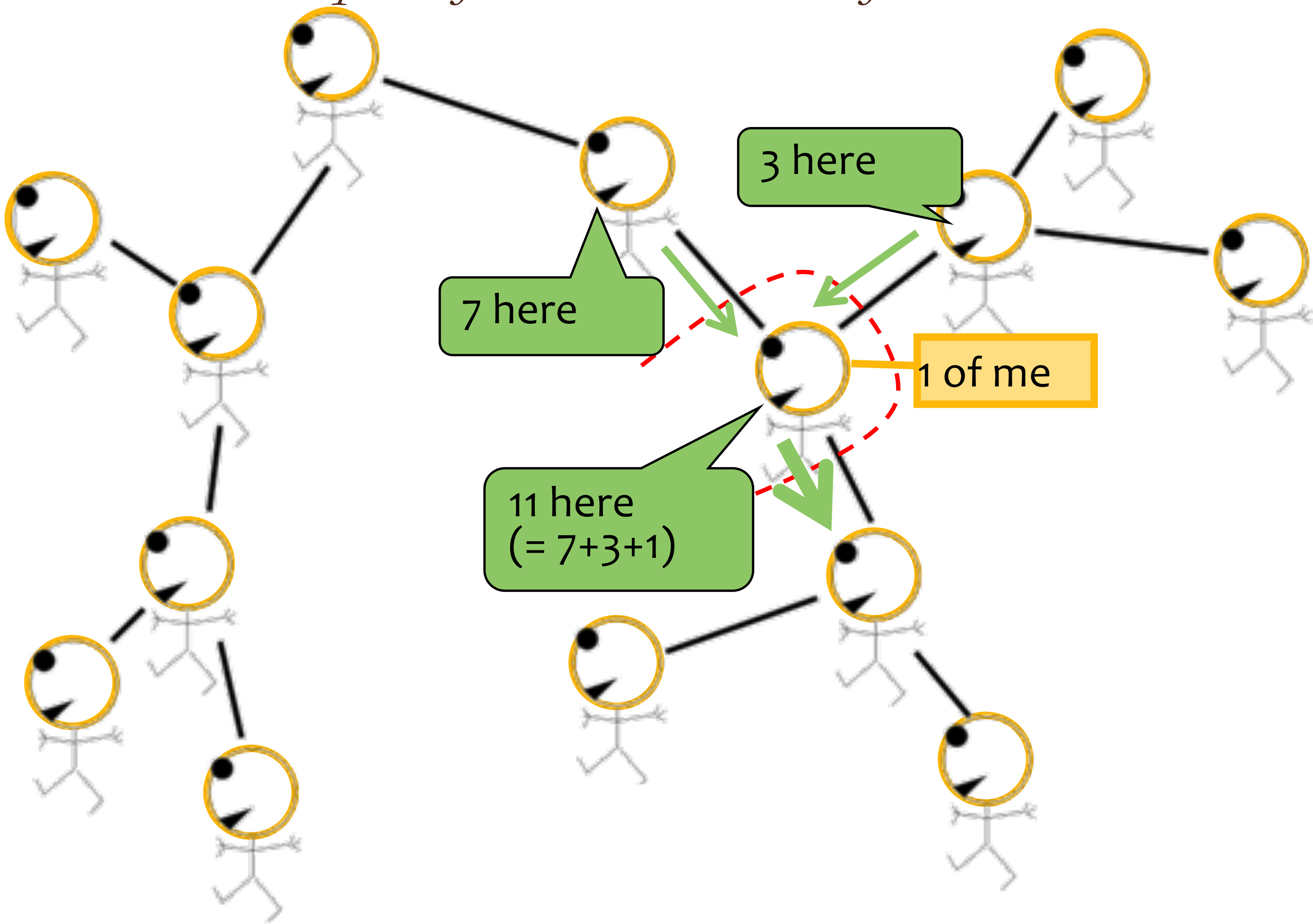
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

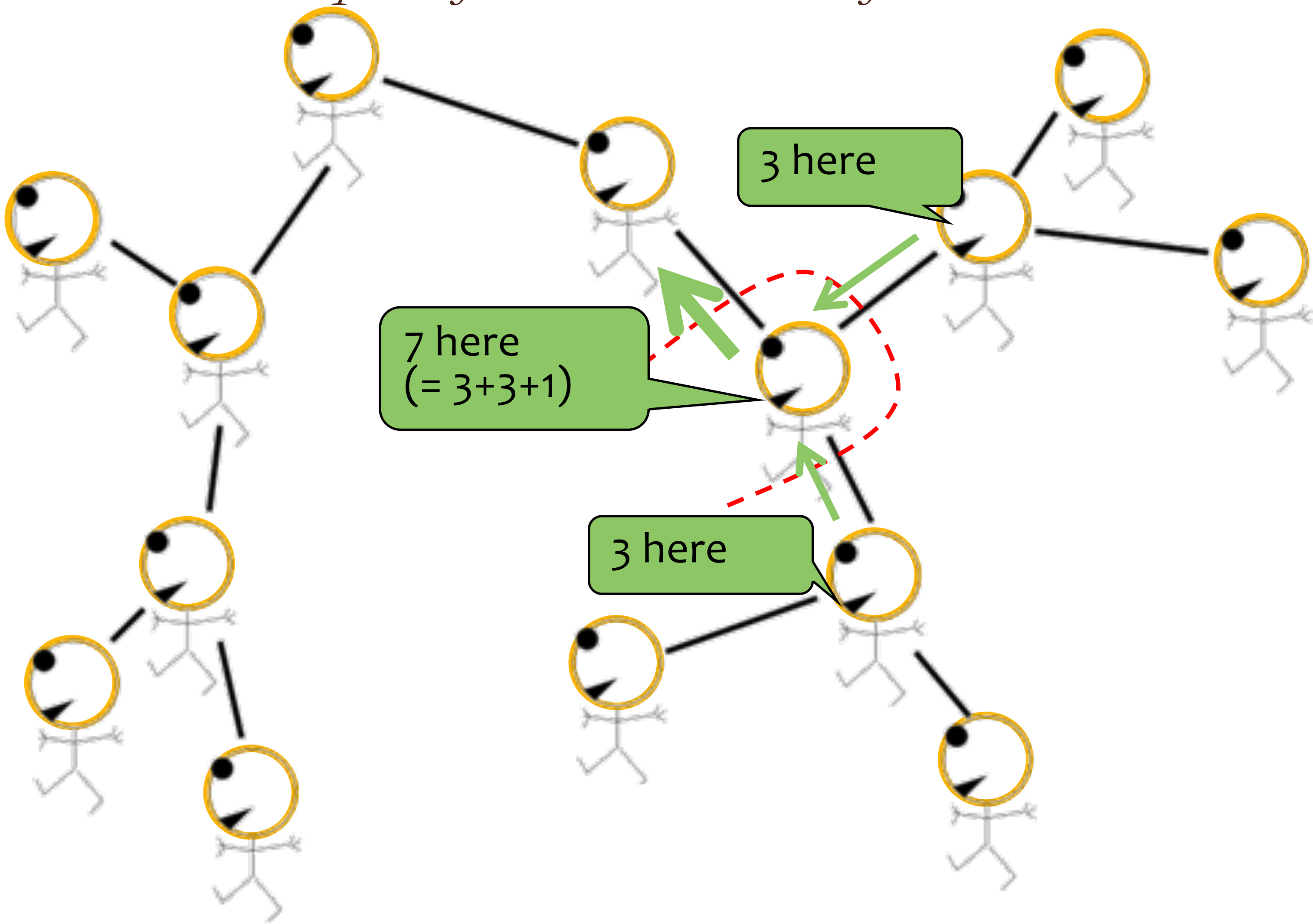
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

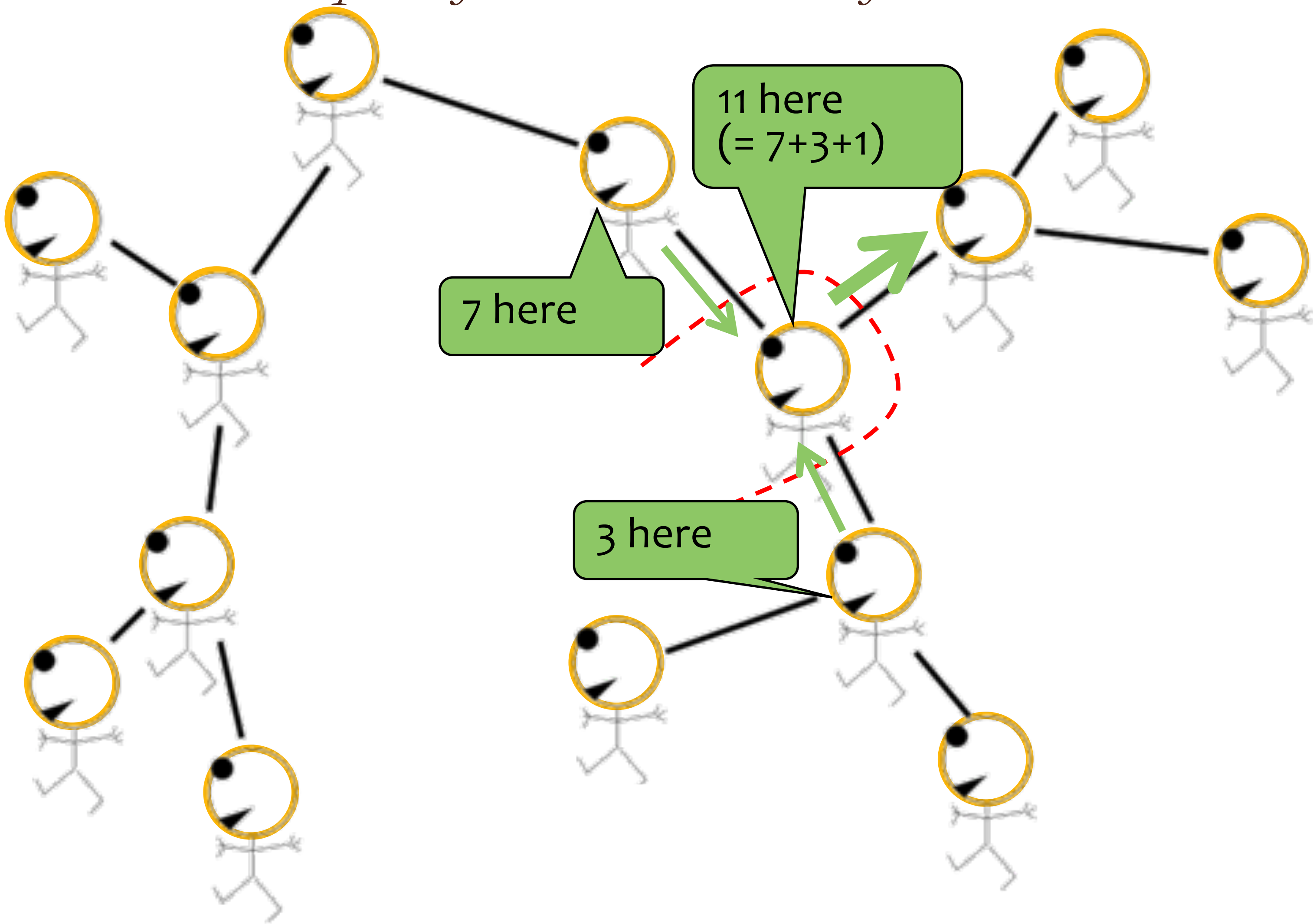
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

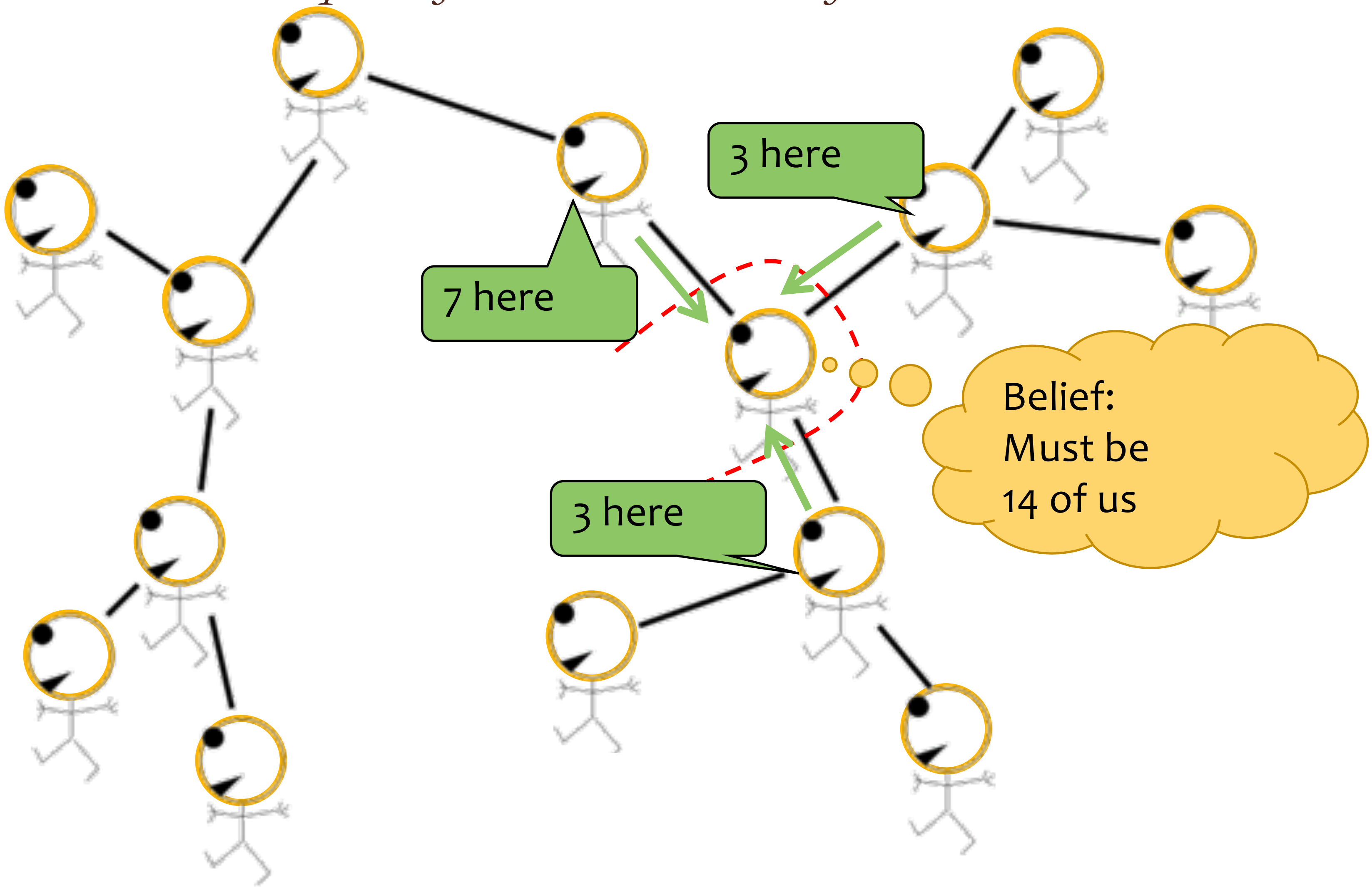
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

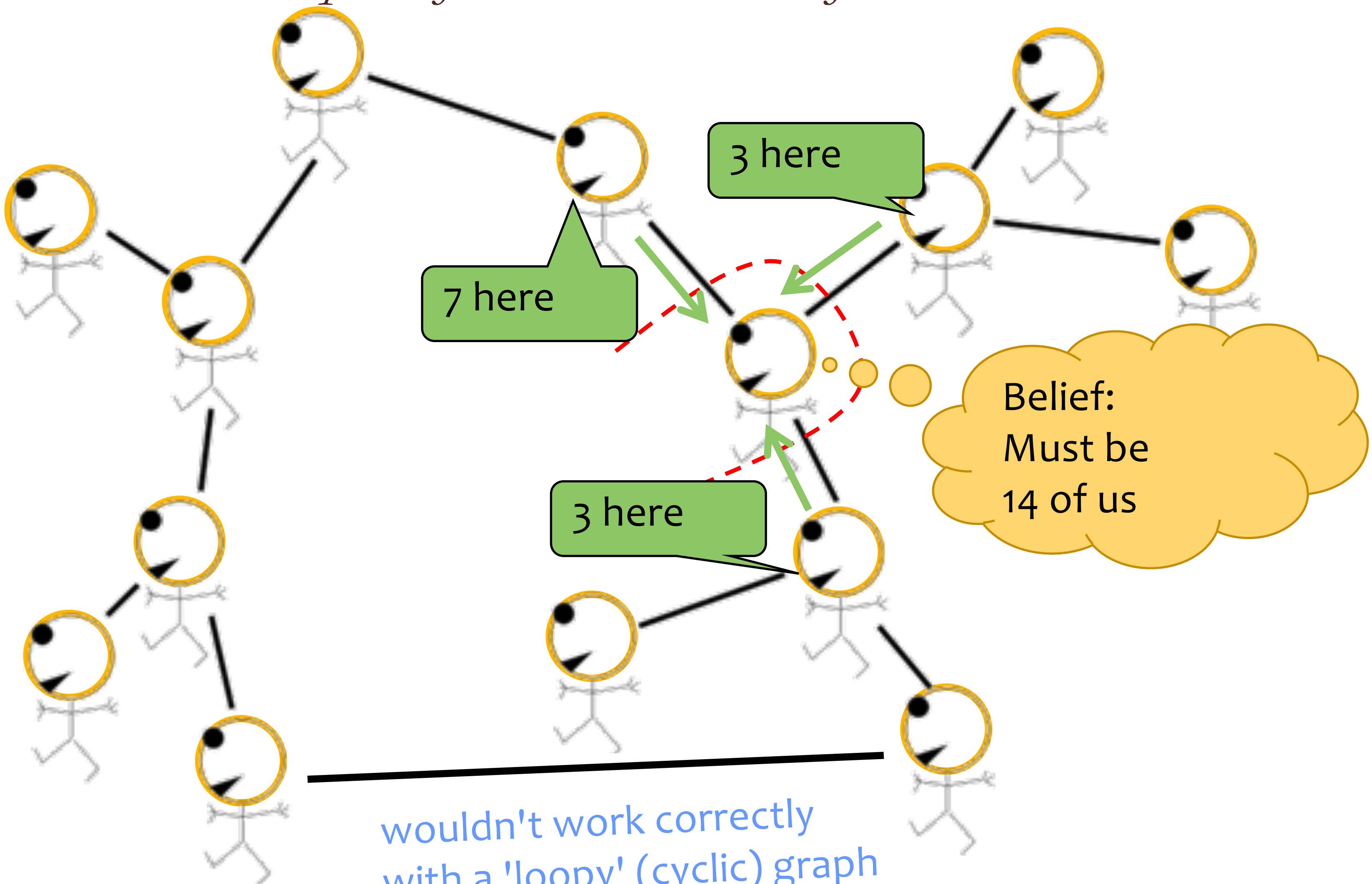
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

INFERENCE FOR HMMS

Inference

Question:

True or False: The **joint probability of the observations and the hidden states** in an HMM is given by:

$$P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = C_{y_1} \left[\prod_{t=1}^{T} A_{y_t, x_t} \right] \left[\prod_{t=1}^{T-1} B_{y_t, y_{t+1}} \right]$$

Recall:

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 $P(Y_1 = k) = C_k, \forall k$

Inference

Question:

True or False: The **probability of the observations** in an HMM is given by:

$$P(\mathbf{X} = \mathbf{x}) = \prod_{t=1}^{T} A_{x_t, x_{t-1}}$$

Recall:

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 $P(Y_1 = k) = C_k, \forall k$

Inference for HMMs

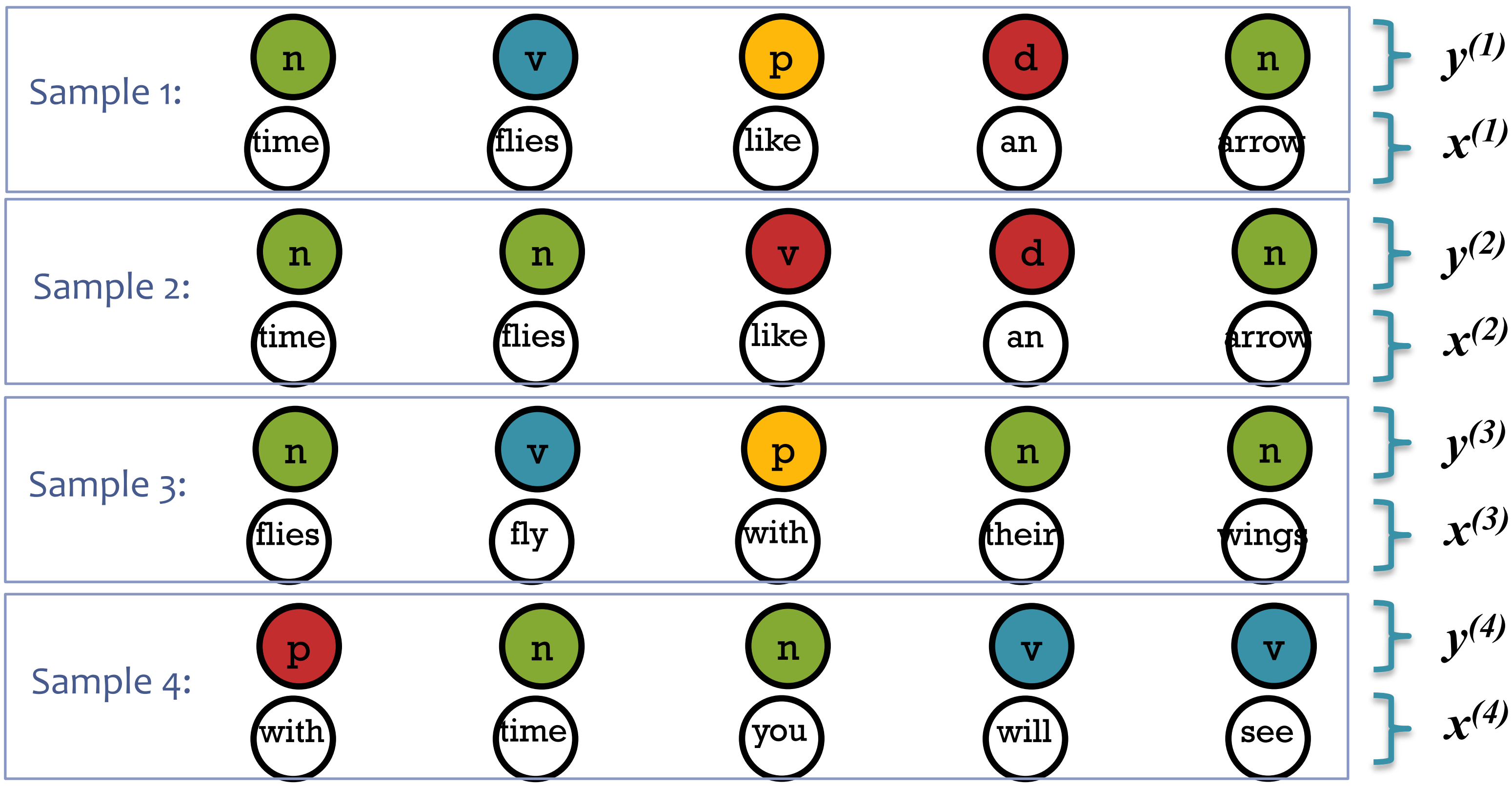
Whiteboard

- 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

THE SEARCH SPACE FOR FORWARD-BACKWARD

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

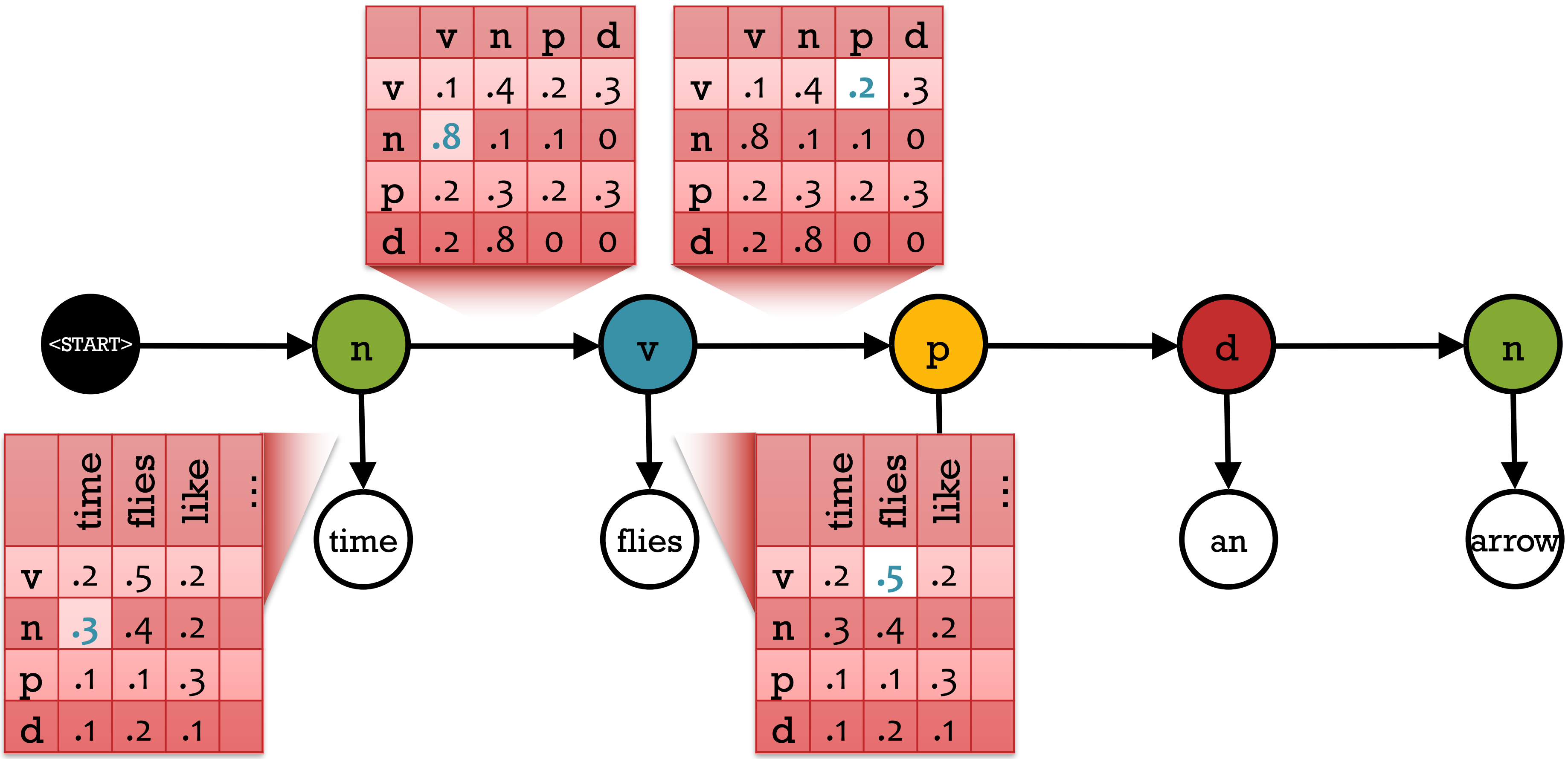
Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$



Example: HMM for POS Tagging

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

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



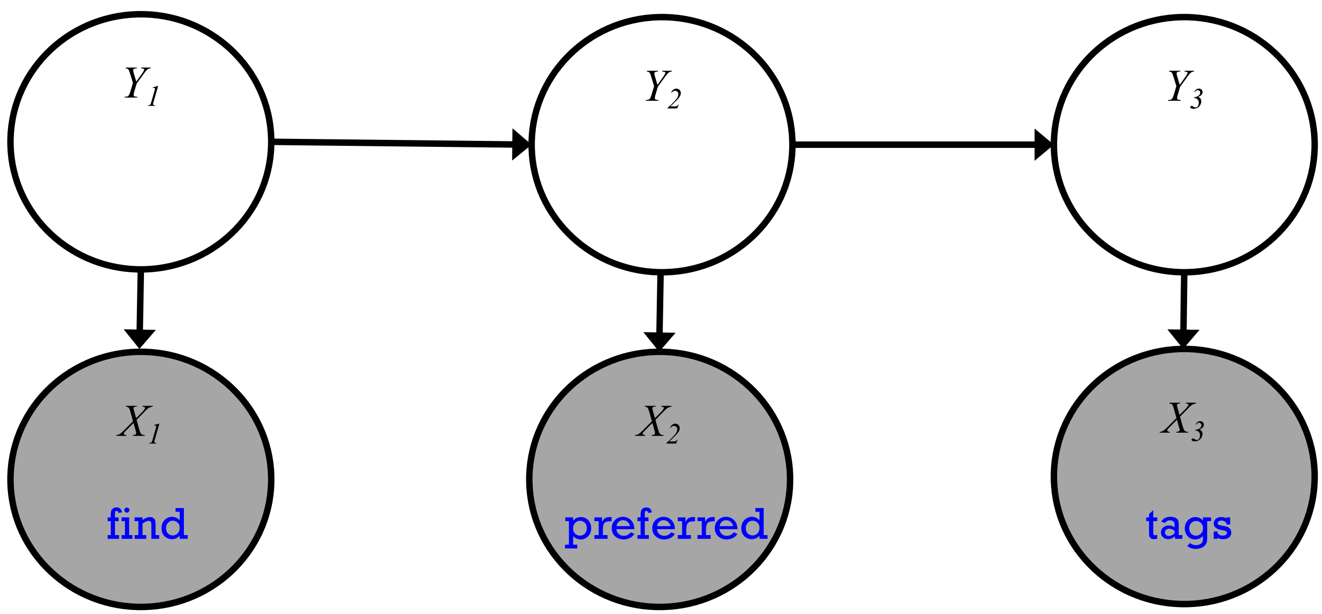
	v	n	p	d
v	.1	.4	.2	.3
n	.8	.1	.1	0
p	.2	.3	.2	.3
d	.2	.8	0	0

	v	n	p	d
v	.1	.4	.2	.3
n	.8	.1	.1	0
p	.2	.3	.2	.3
d	.2	.8	0	0

	time	flies	like	…
v	.2	.5	.2	
n	.3	.4	.2	
p	.1	.1	.3	
d	.1	.2	.1	

	time	flies	like	…
v	.2	.5	.2	
n	.3	.4	.2	
p	.1	.1	.3	
d	.1	.2	.1	

Example: HMM for POS Tagging



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

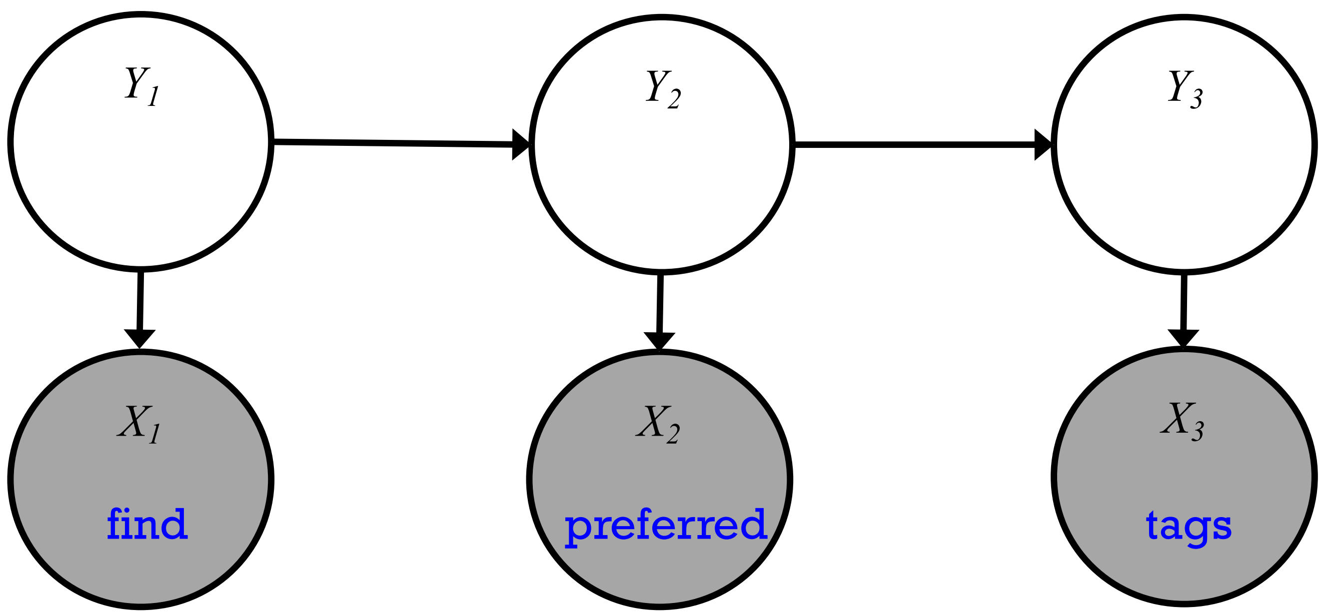
Inference for HMMs

Whiteboard

– Brute Force Evaluation
– Forward-backward search space

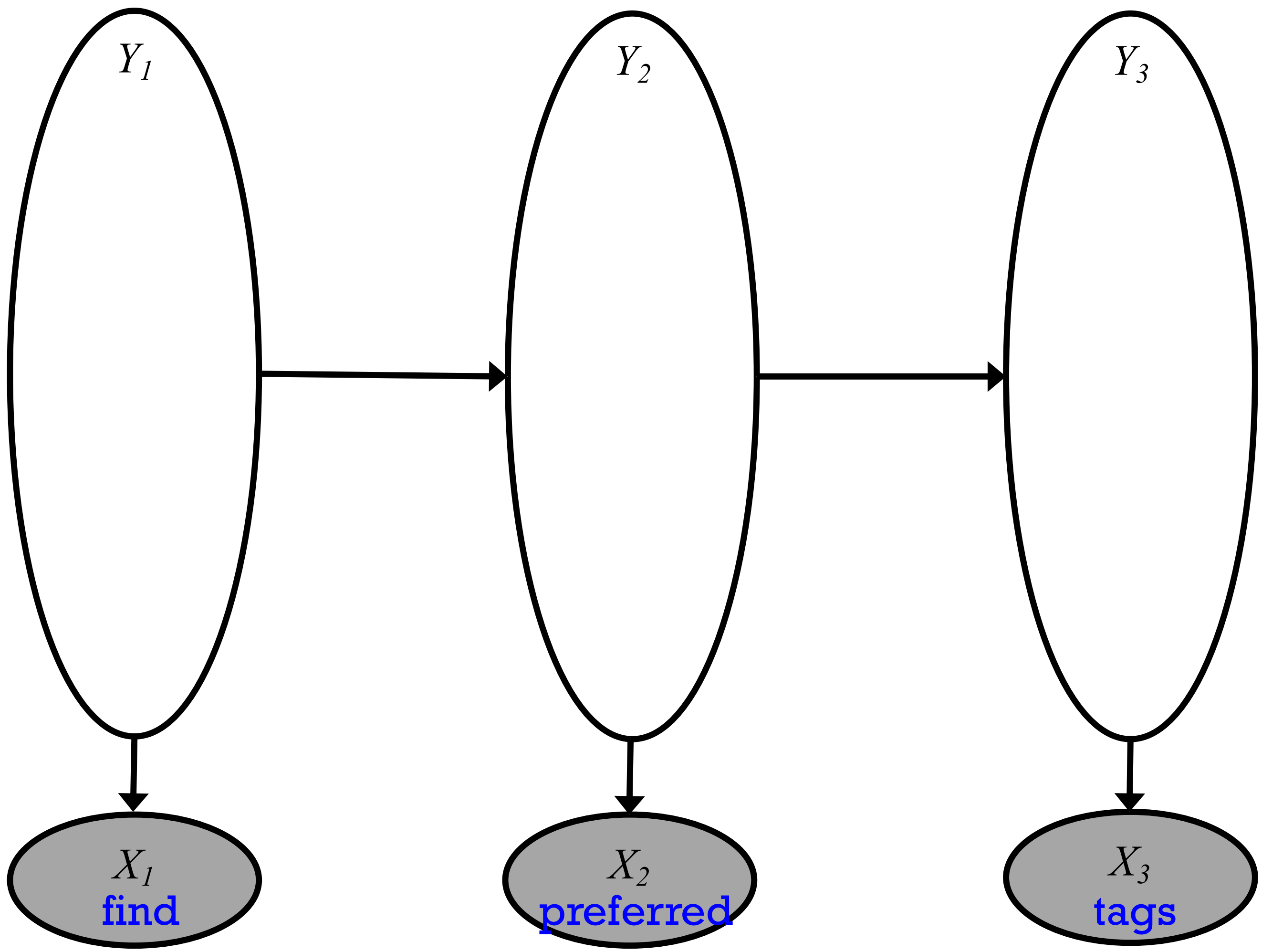
THE FORWARD-BACKWARD ALGORITHM

Forward-Backward Algorithm

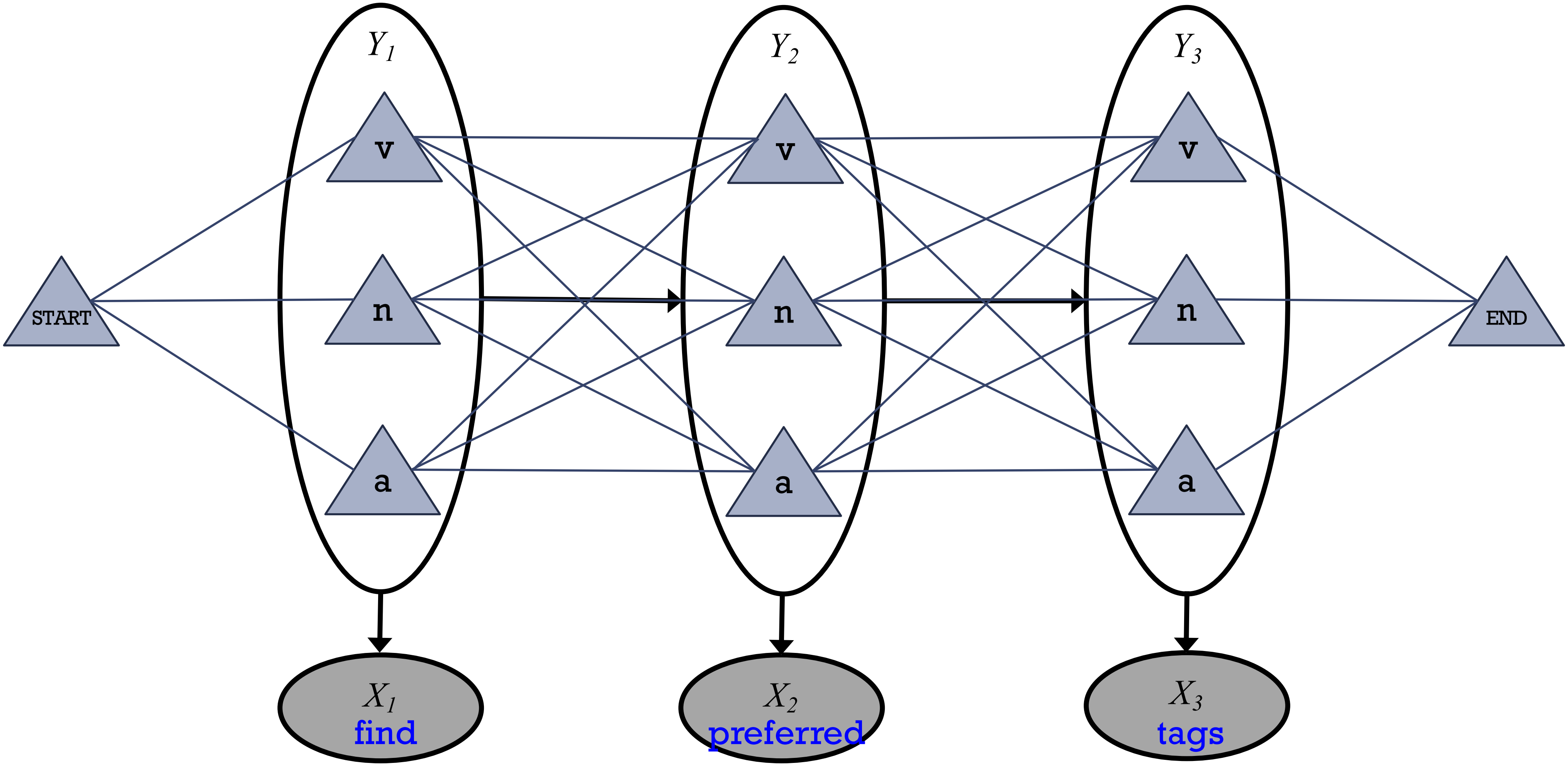


Could be verb or noun *Could be adjective or verb* *Could be noun or verb*

Forward-Backward Algorithm

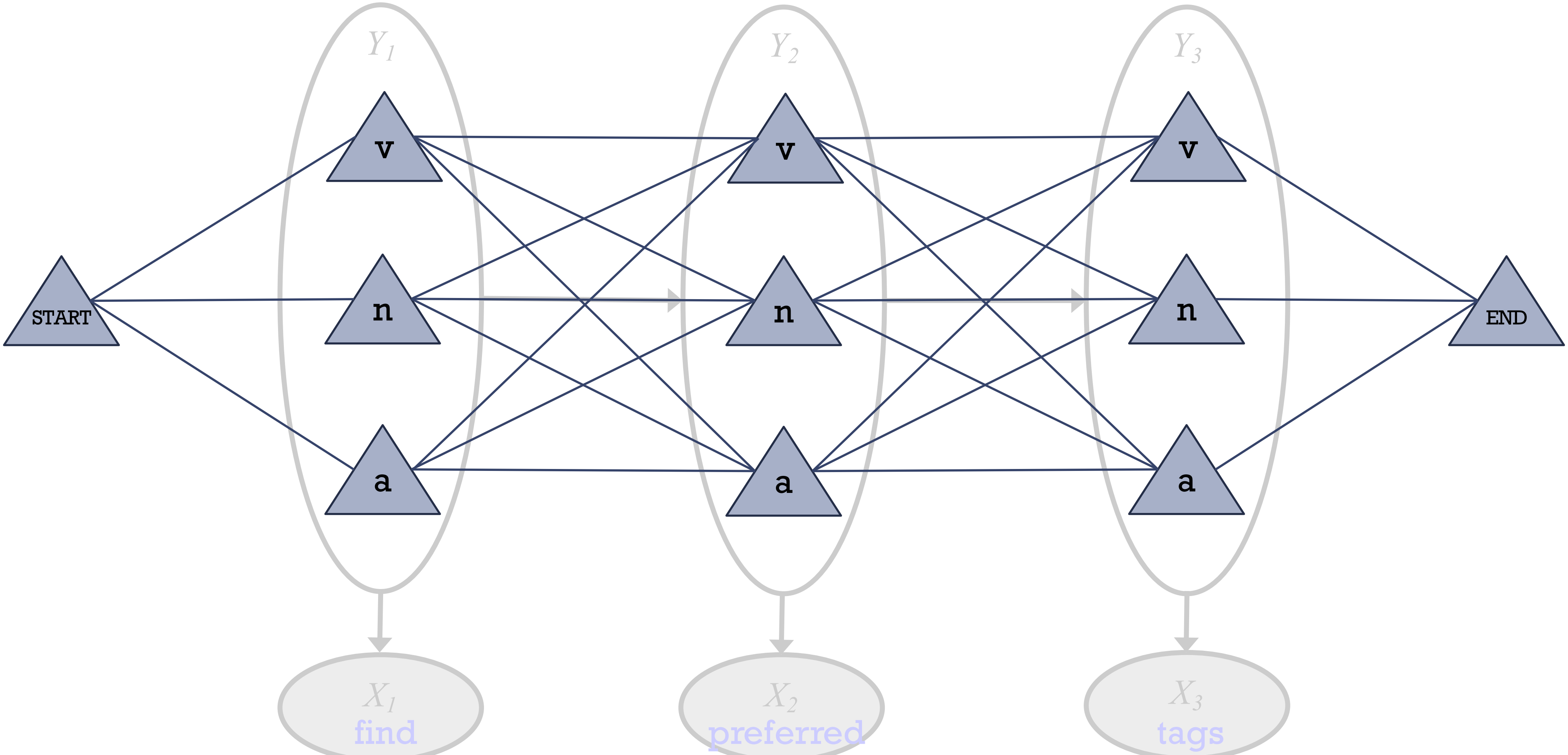


Forward-Backward Algorithm



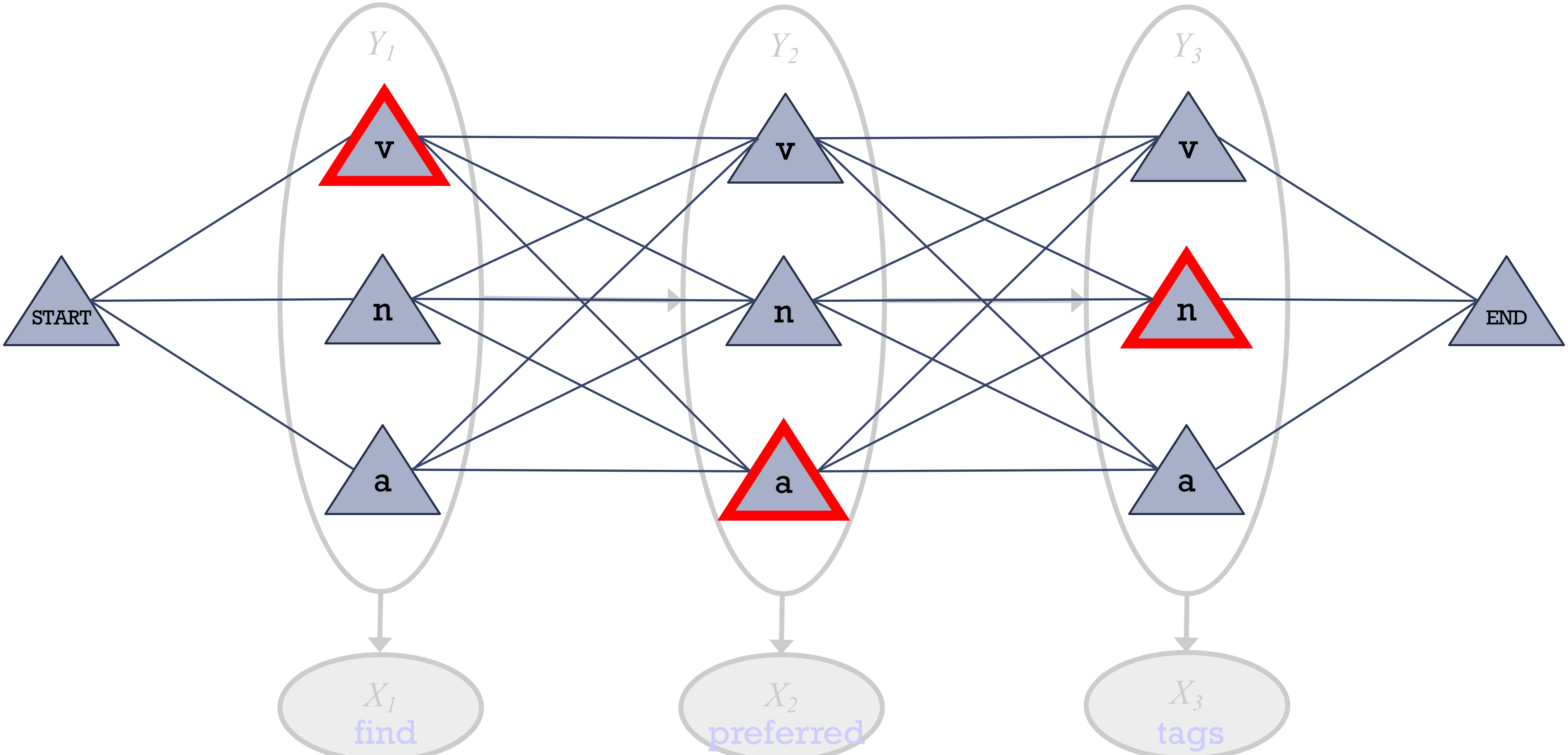
- Let's show the possible *values* for each variable

Forward-Backward Algorithm



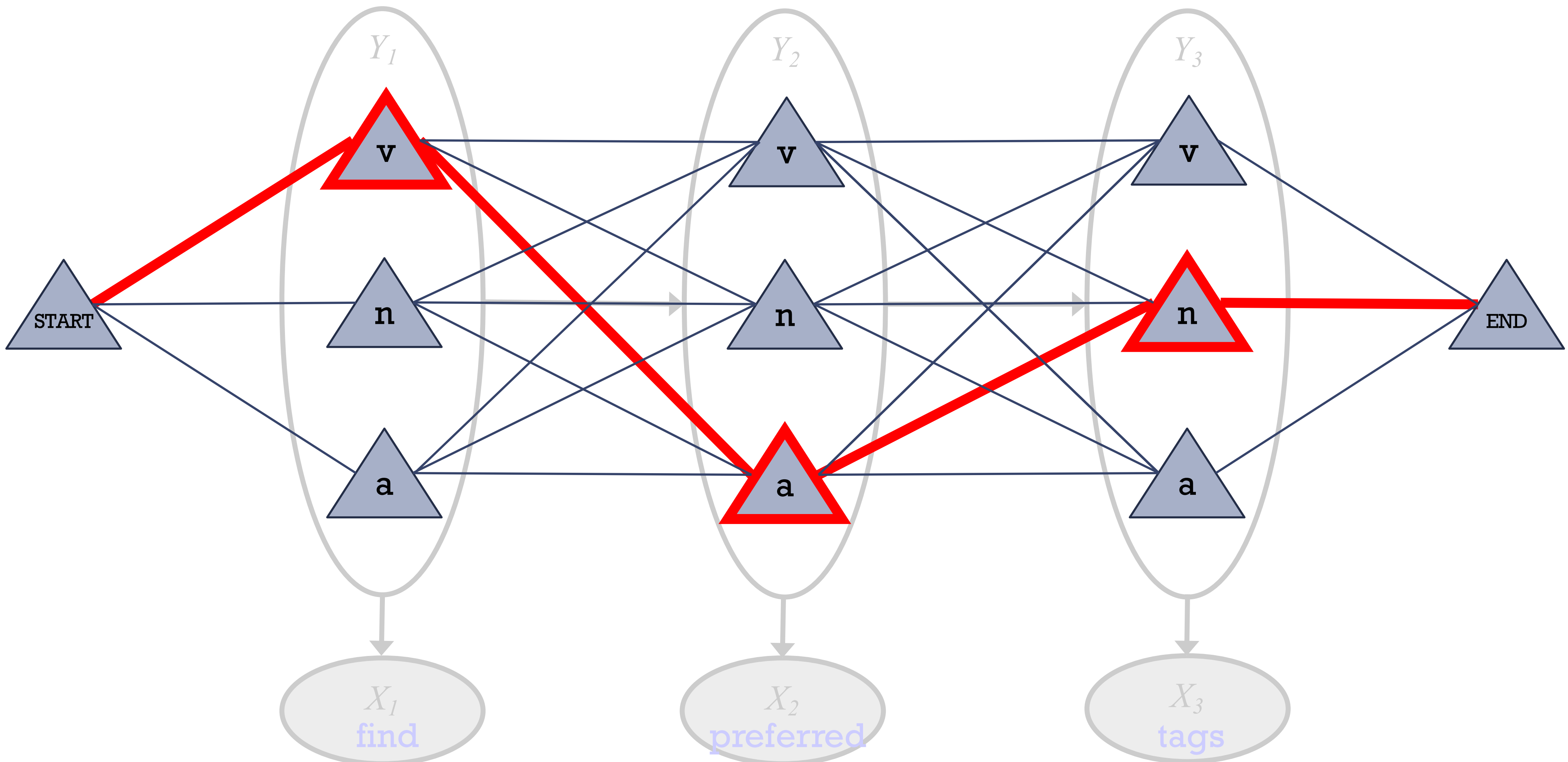
- Let's show the possible *values* for each variable

Forward-Backward Algorithm



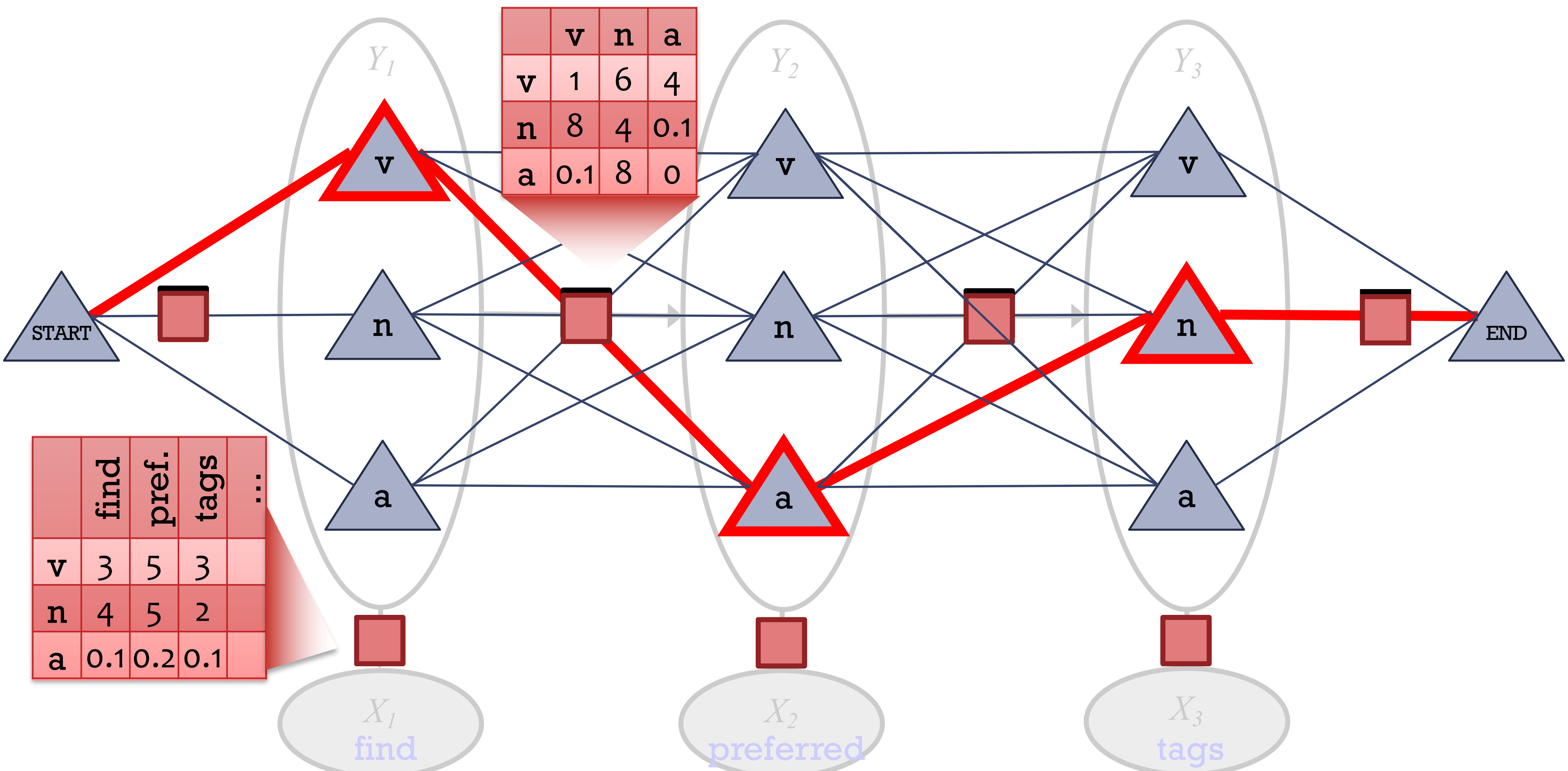
- Let's show the possible *values* for each variable
- One possible assignment

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

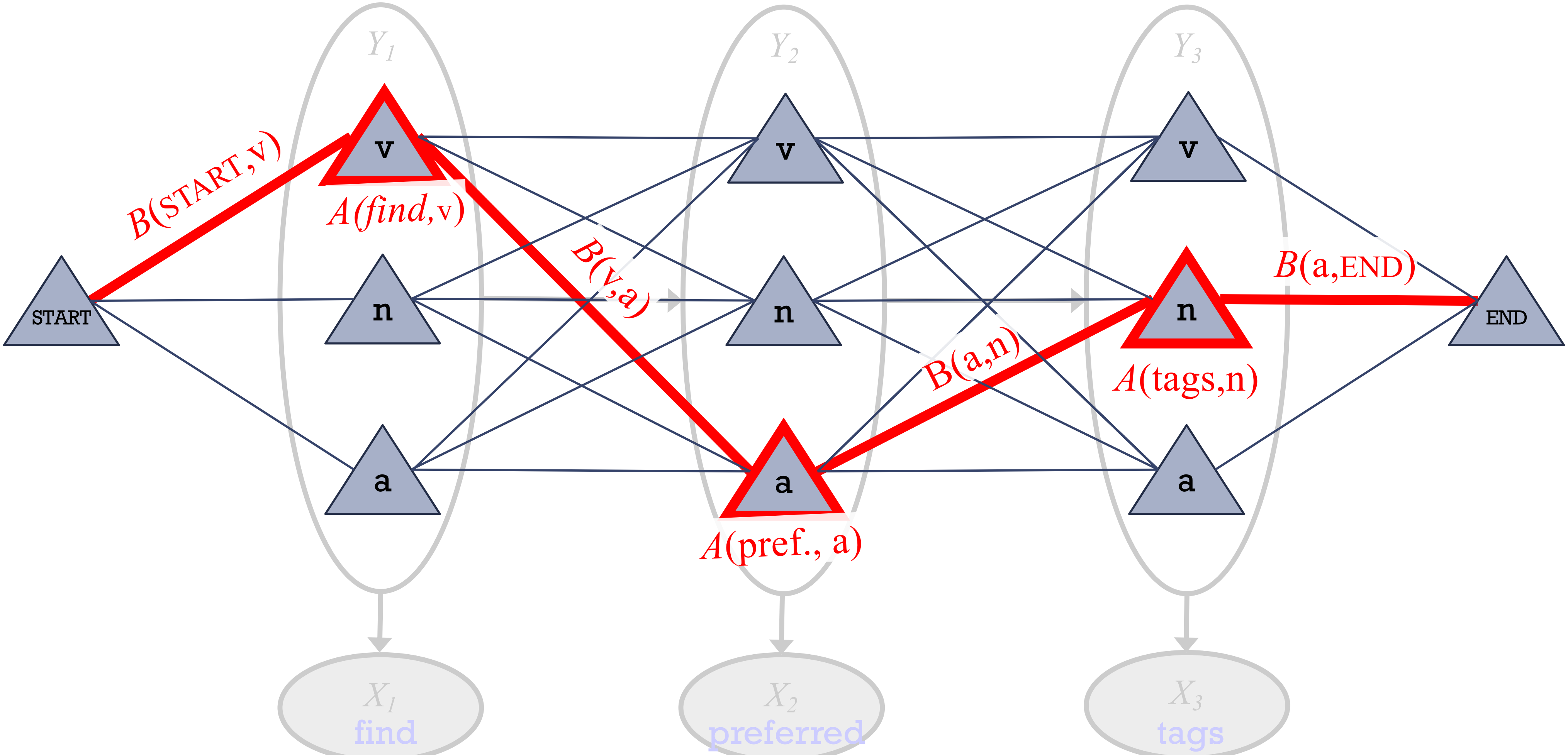


	v	n	a
v	1	6	4
n	8	4	0.1
a	0.1	8	0

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

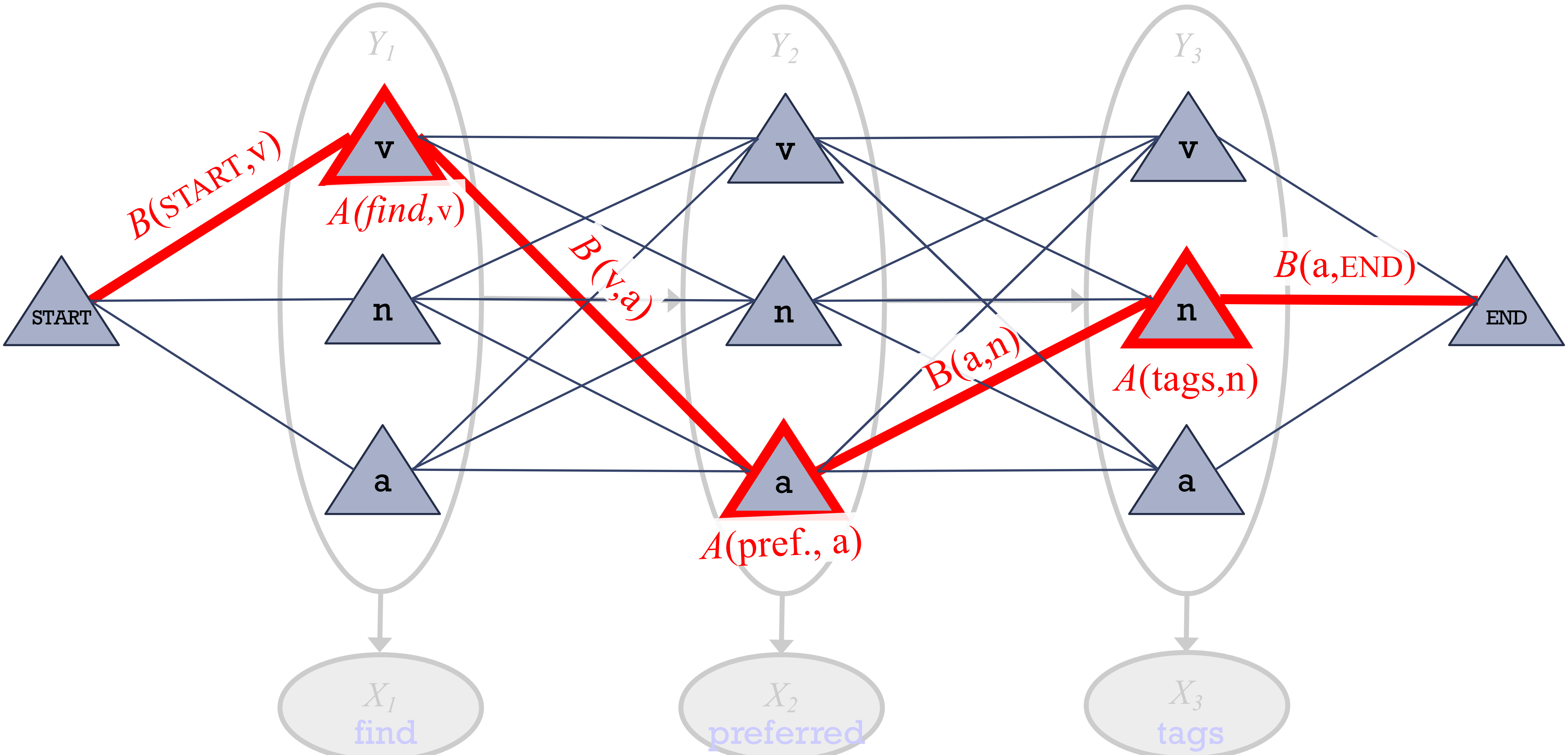
- Let's show the possible *values* for each variable
- One possible assignment
- And what the 7 transition / emission factors think of it …

Viterbi Algorithm: Most Probable Assignment



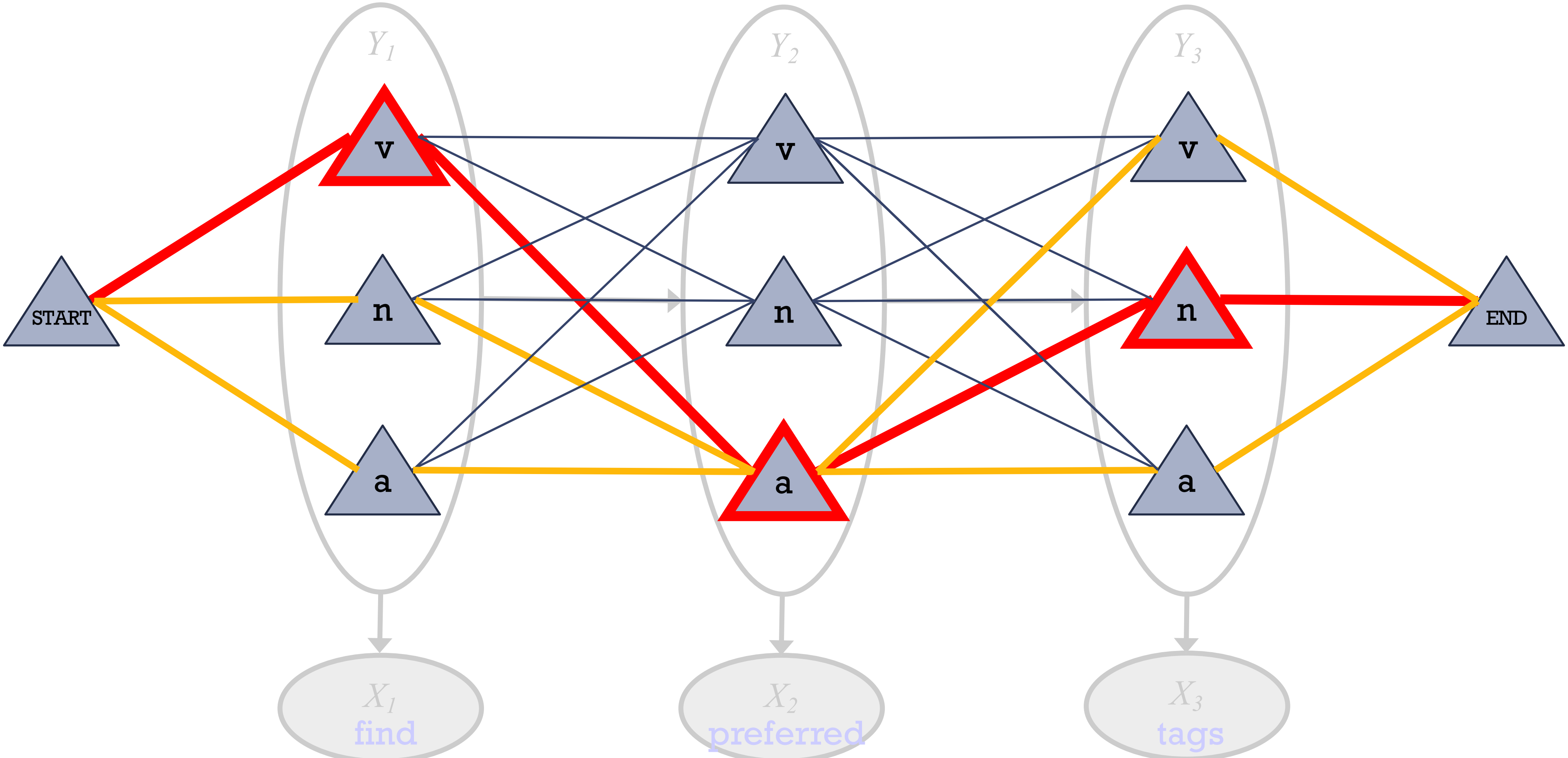
- So p(**v a n**) = (1/Z) * product of 7 numbers
- Numbers associated with edges and nodes of path
- Most probable assignment = **path with highest product**

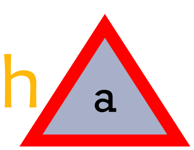
Viterbi Algorithm: Most Probable Assignment



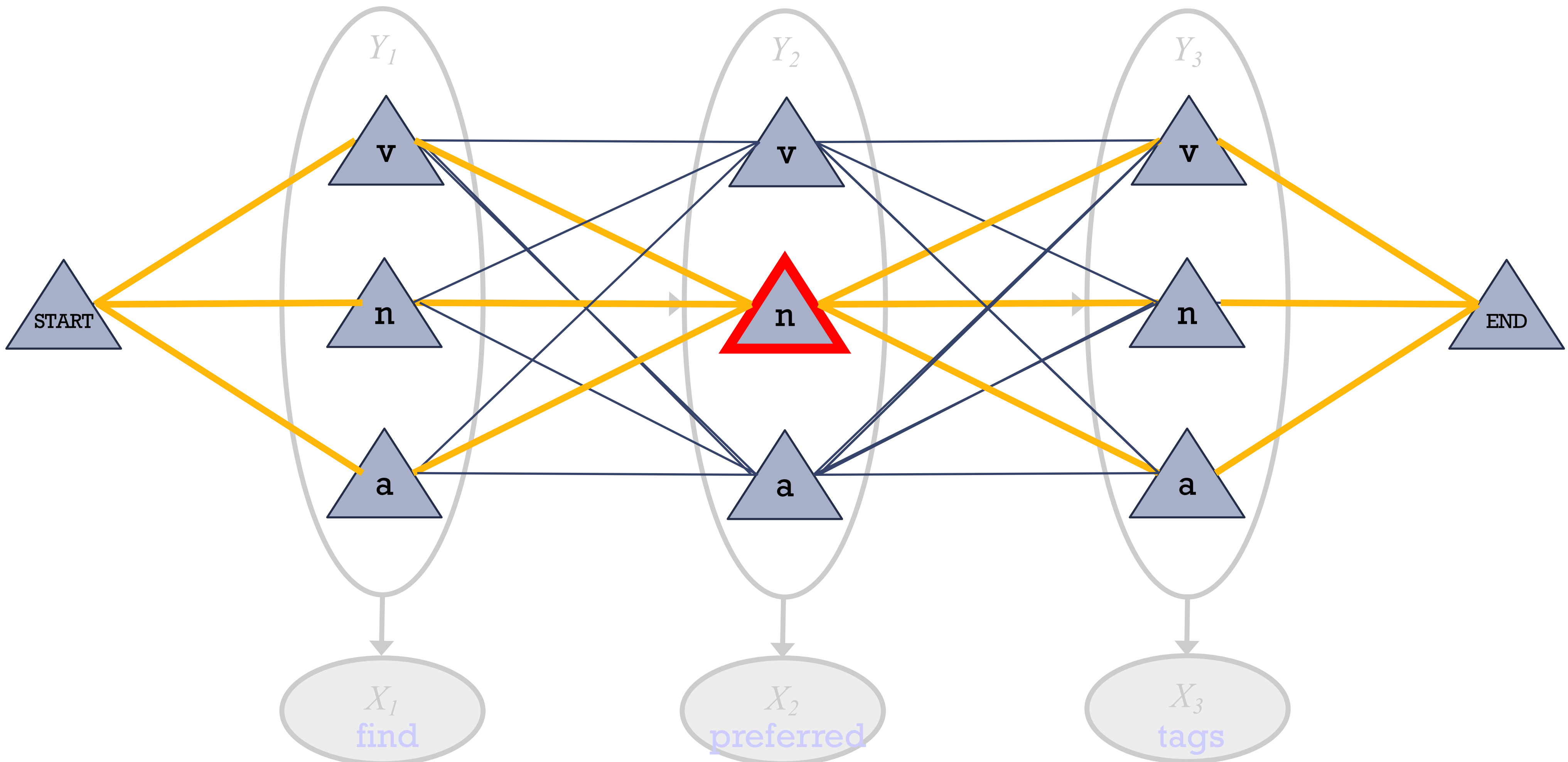
- So p(**v a n**) = (1/Z) * product weight of one path

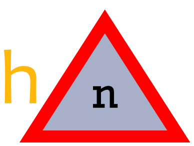
Forward-Backward Algorithm: Finds Marginals



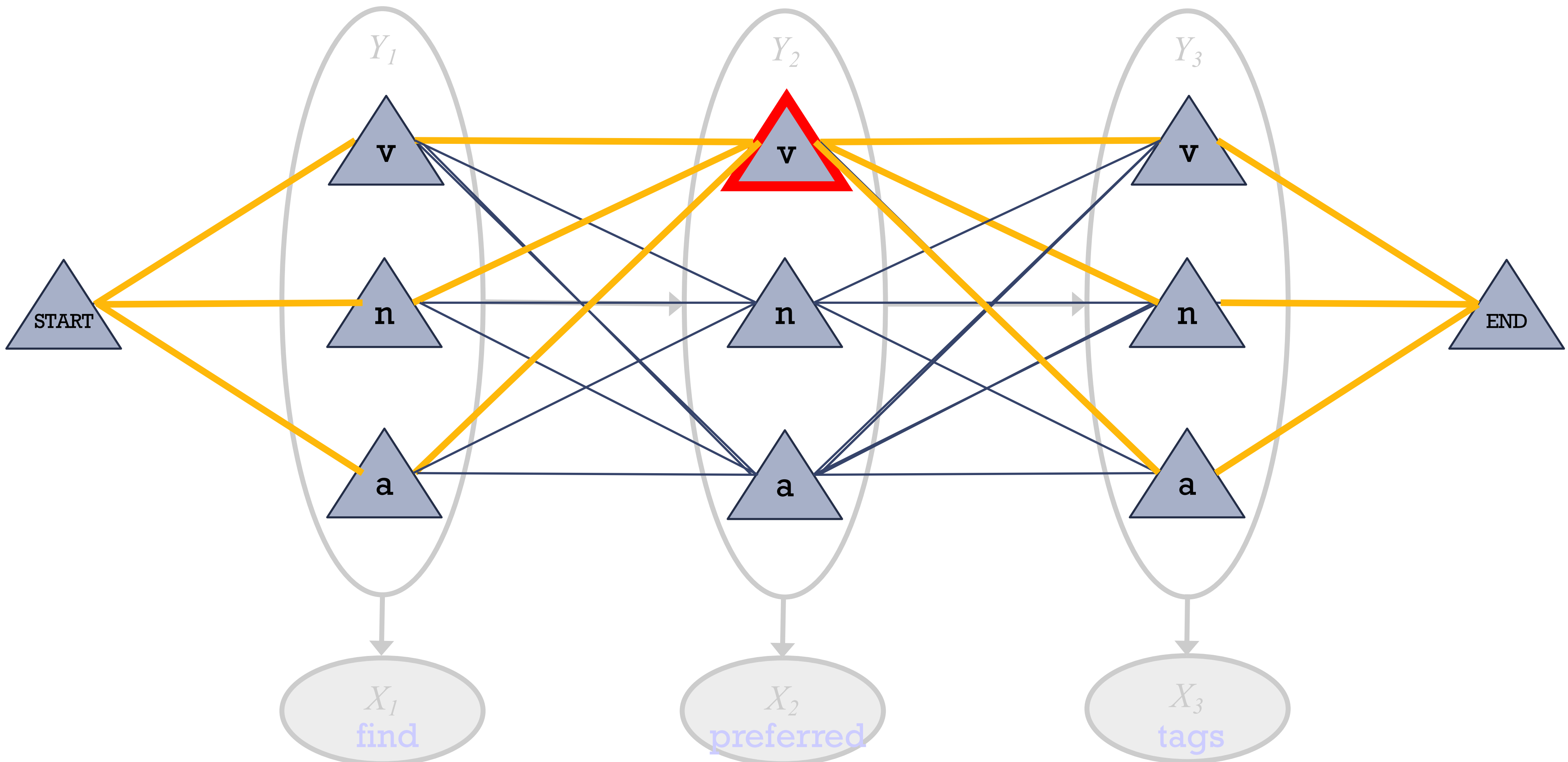
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(Y_2 = a)$
  = (1/Z) * total weight of *all* paths through 

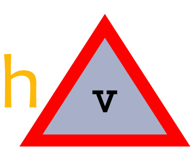
Forward-Backward Algorithm: Finds Marginals



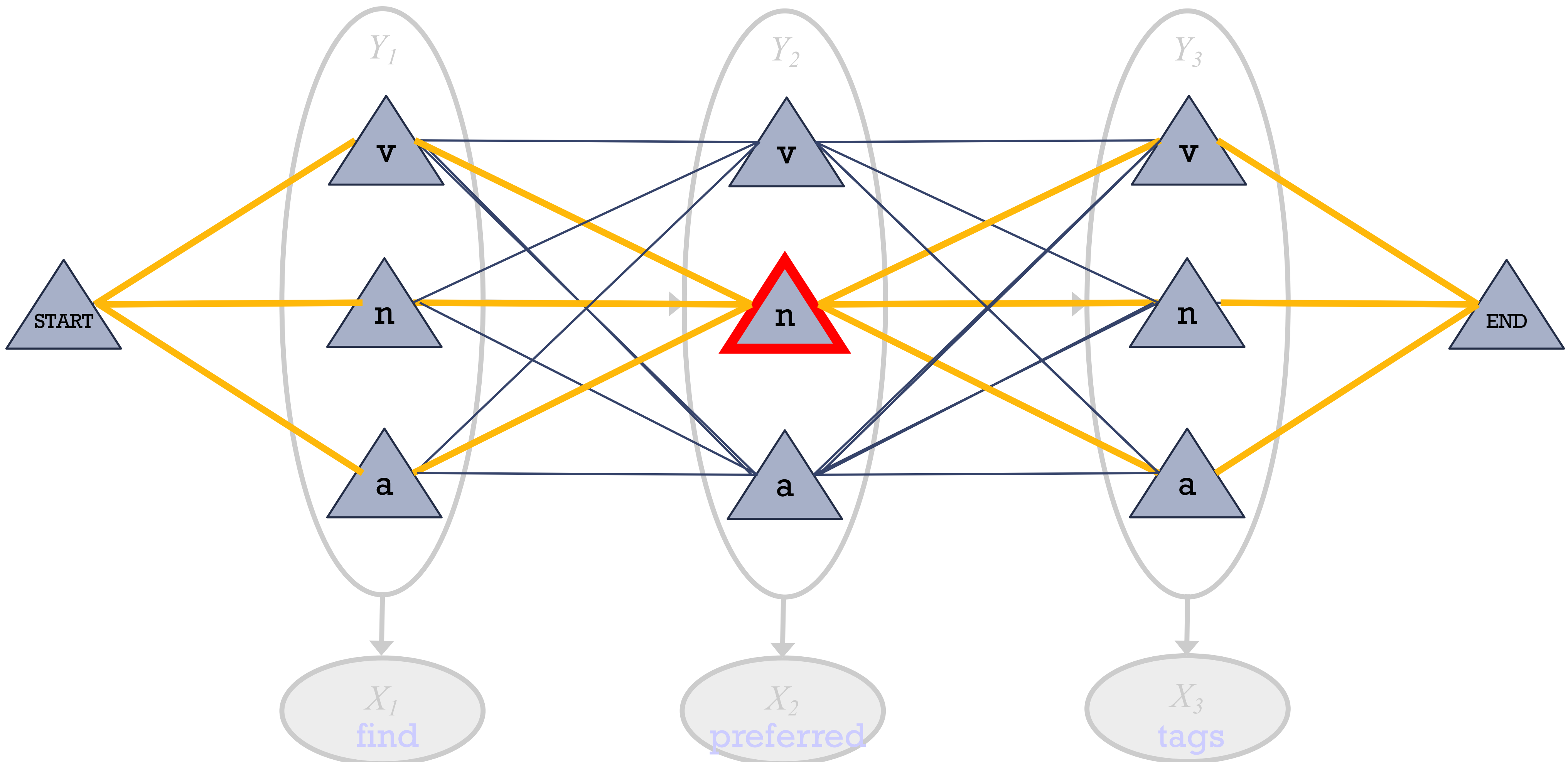
- So $p(\mathbf{v\ a\ n}) = (1/Z)$ * product weight of one path
- Marginal probability $p(Y_2 = n)$
  $= (1/Z)$ * total weight of *all* paths through 

Forward-Backward Algorithm: Finds Marginals



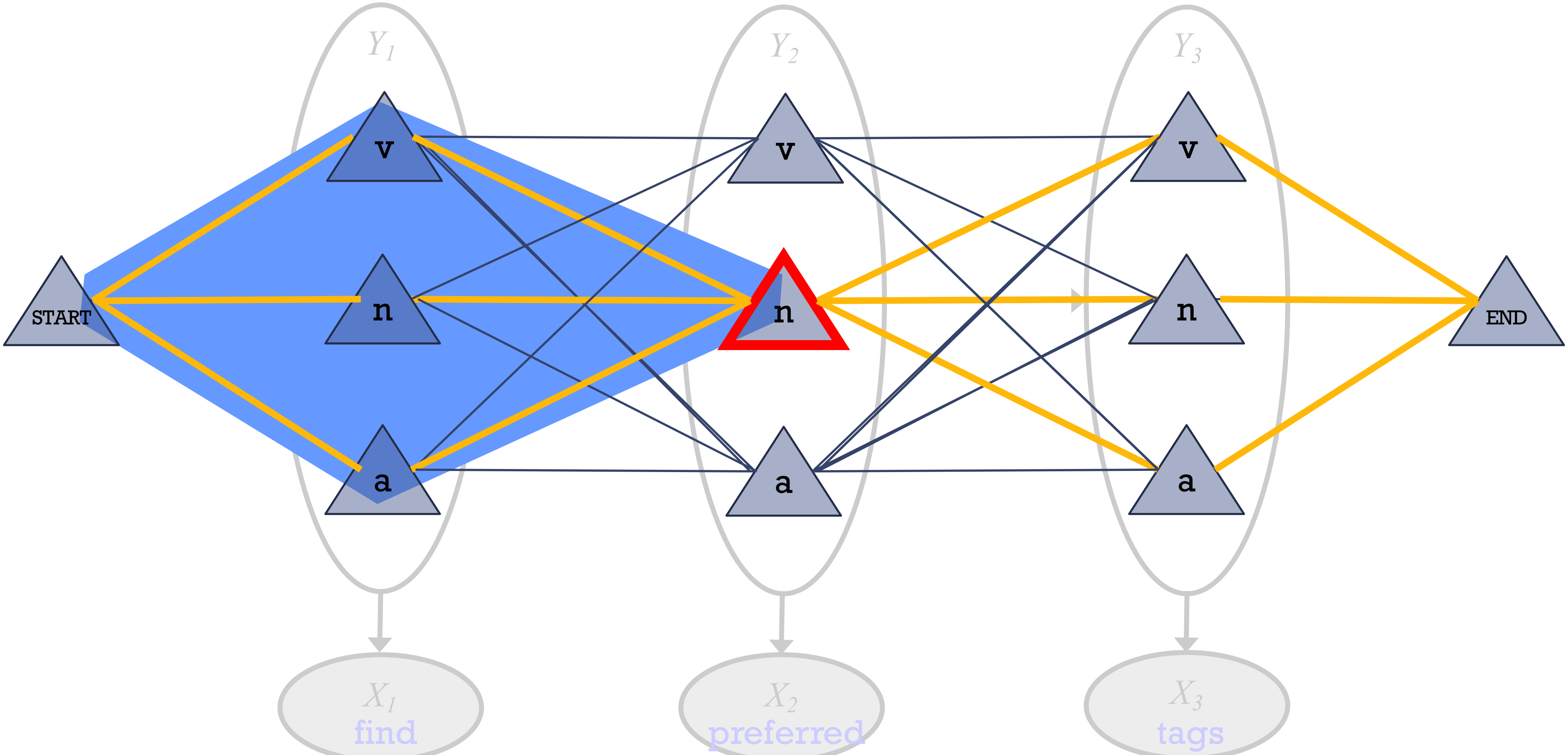
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(Y_2 = \mathrm{v})$
  = (1/Z) * total weight of *all* paths through 

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

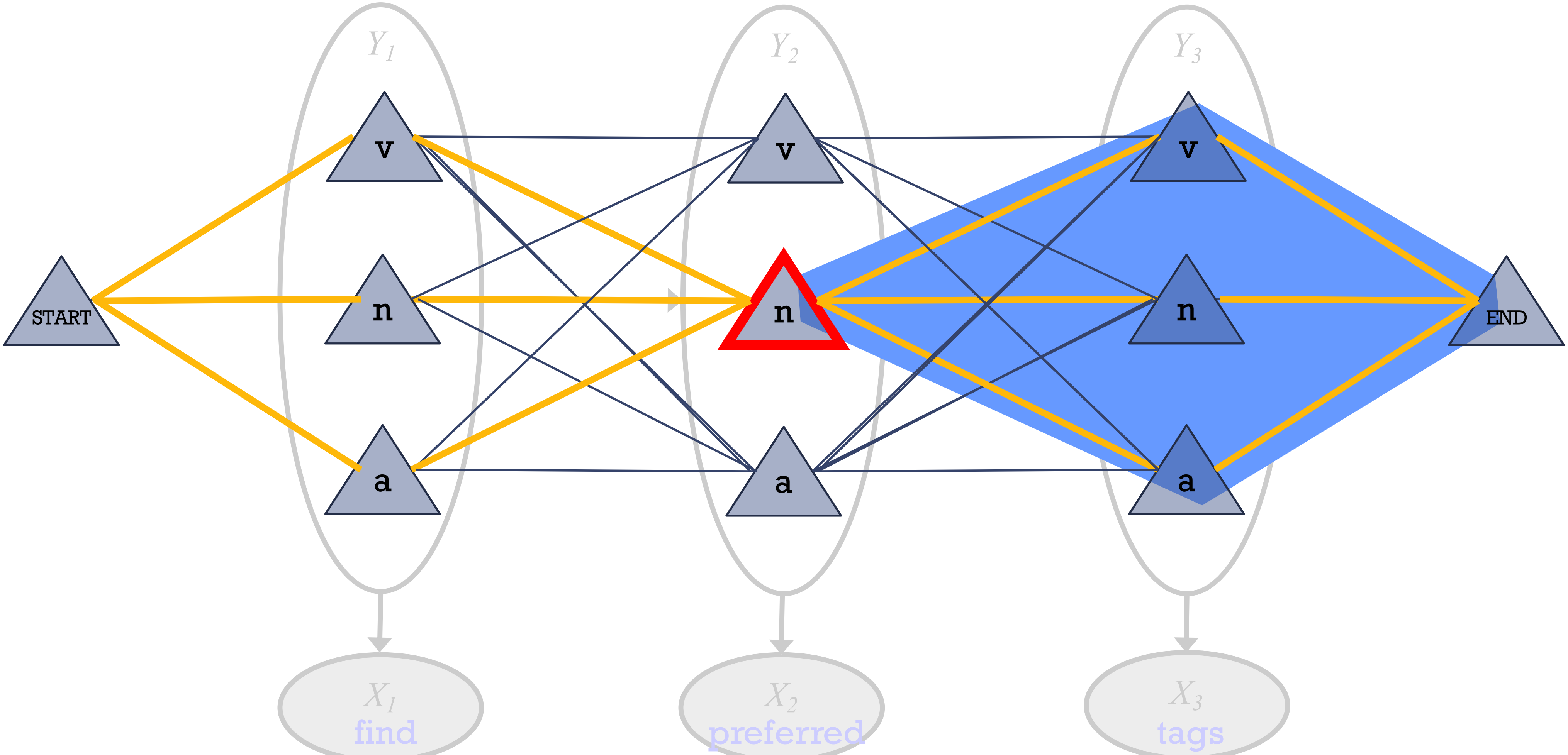
Forward-Backward Algorithm: Finds Marginals



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

(found by dynamic programming: matrix-vector products)

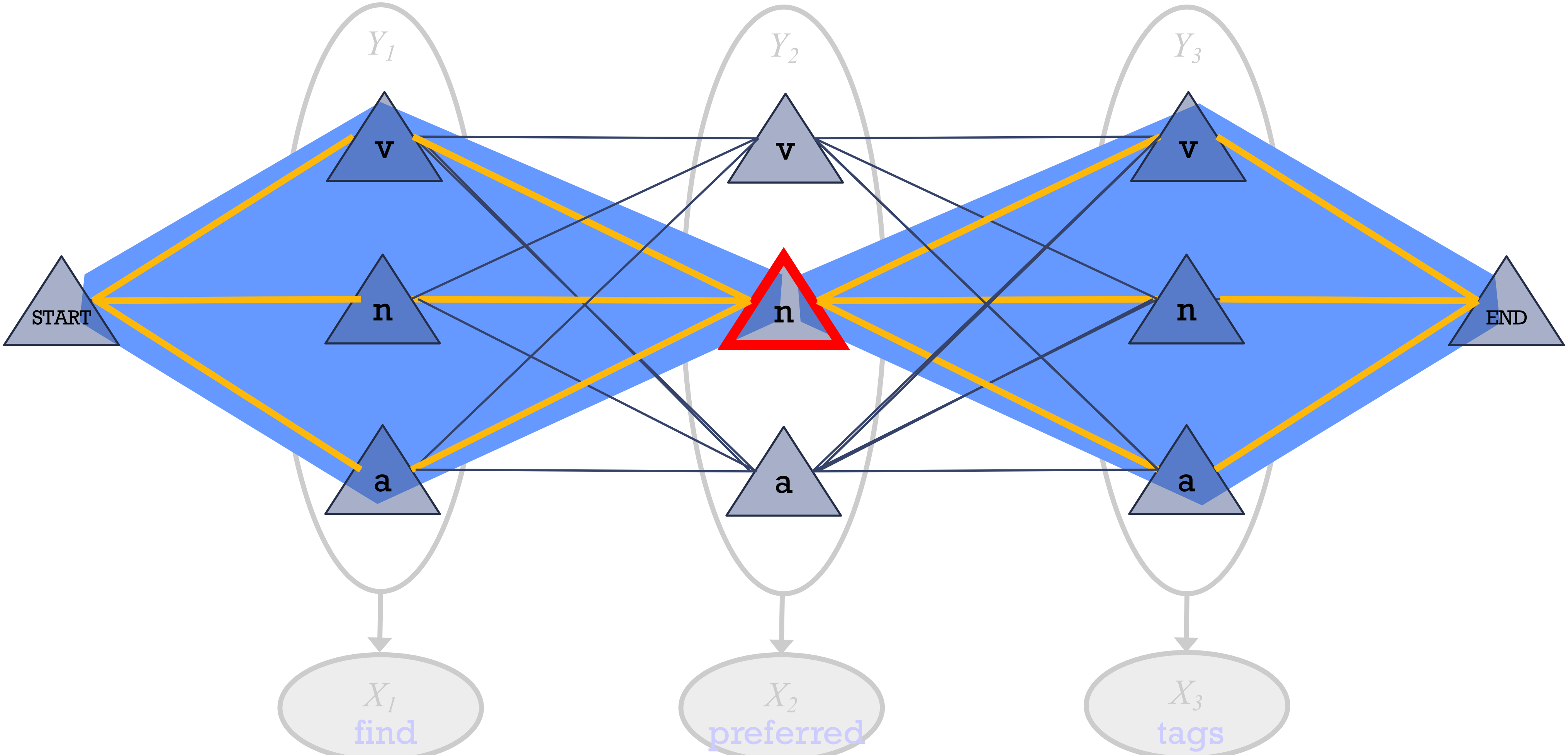
Forward-Backward Algorithm: Finds Marginals



$\beta_2(\mathbf{n})$ = 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)

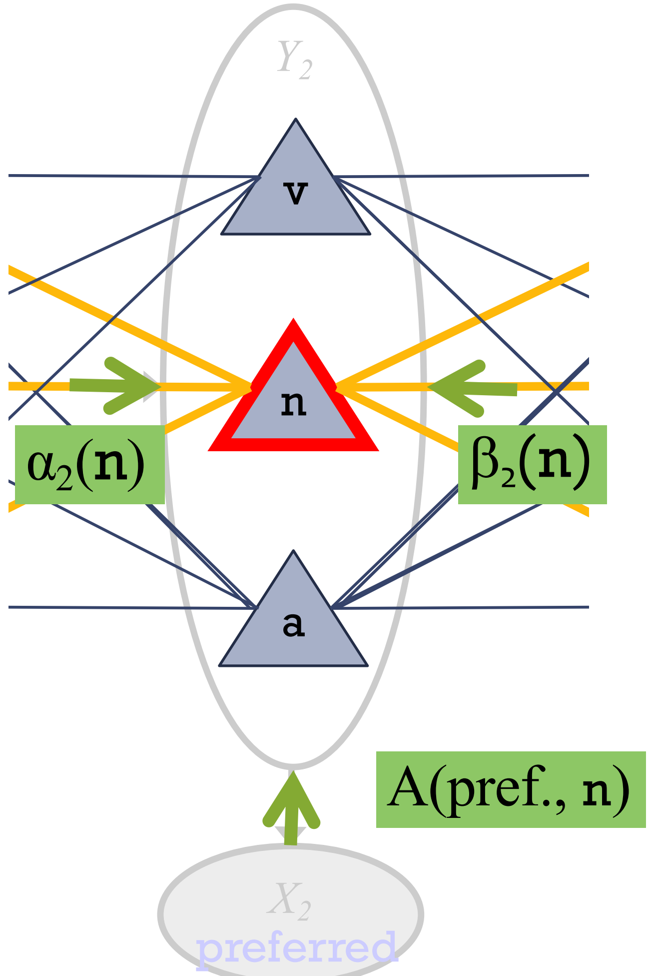
$\beta_2(\mathbf{n})$ = total weight of these path *suffixes* (x + y + z)

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

Forward-Backward Algorithm: Finds Marginals

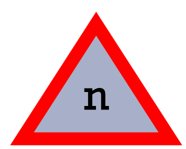
Oops! The weight of a path through a state also includes a weight at that state.
So $\alpha(\mathbf{n}) \cdot \beta(\mathbf{n})$ isn't enough.

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



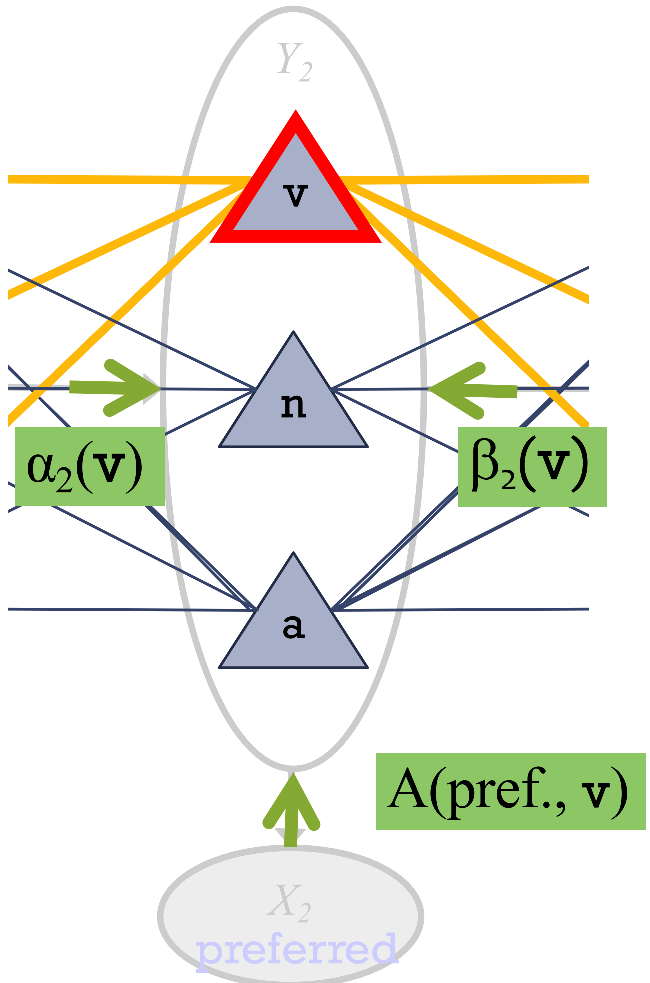
"belief that $Y_2 = \mathbf{n}$"

total weight of *all* paths through



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

Forward-Backward Algorithm: Finds Marginals



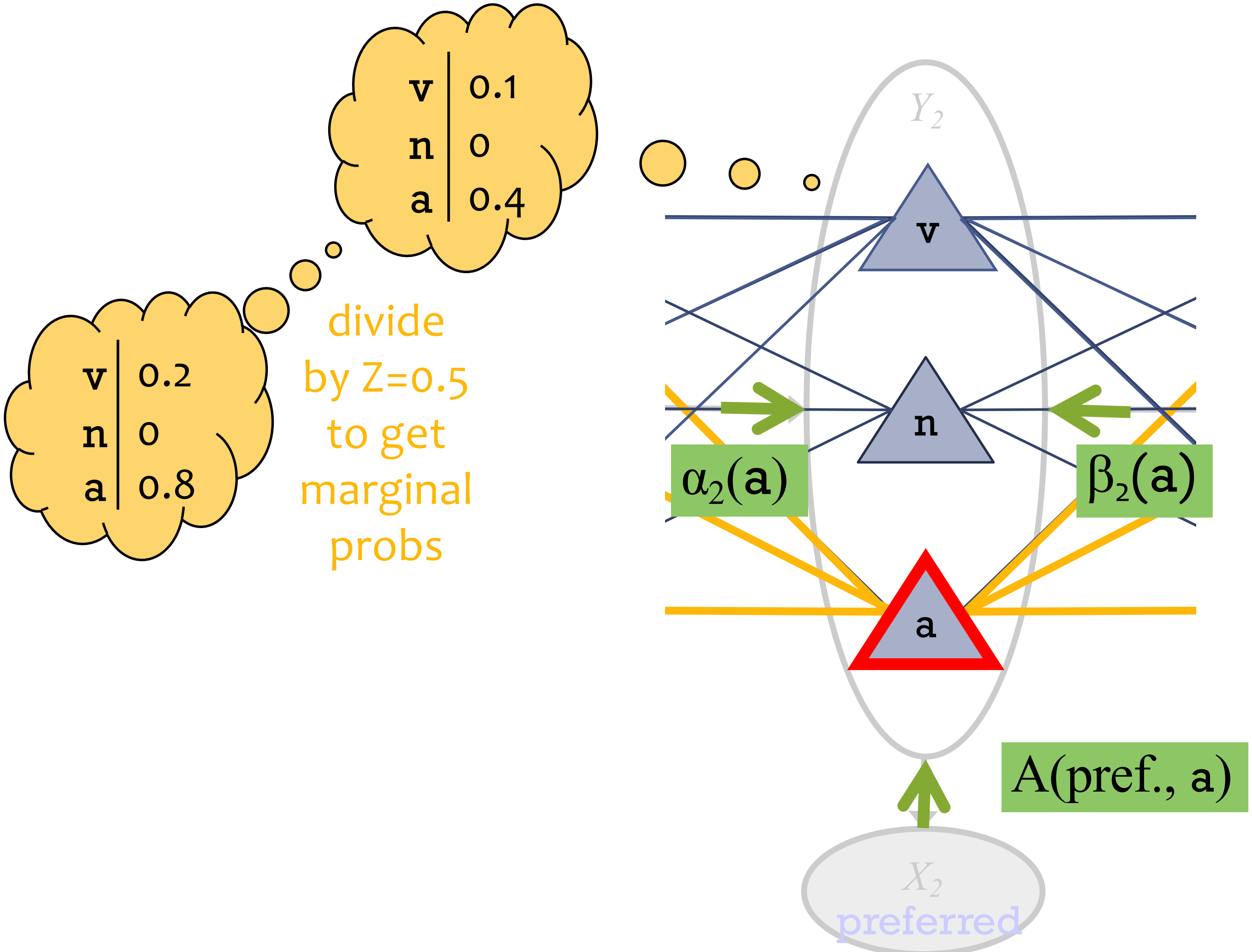
"belief that Y_2 = **v**"

"belief that Y_2 = **n**"

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

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

Forward-Backward Algorithm: Finds Marginals



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

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

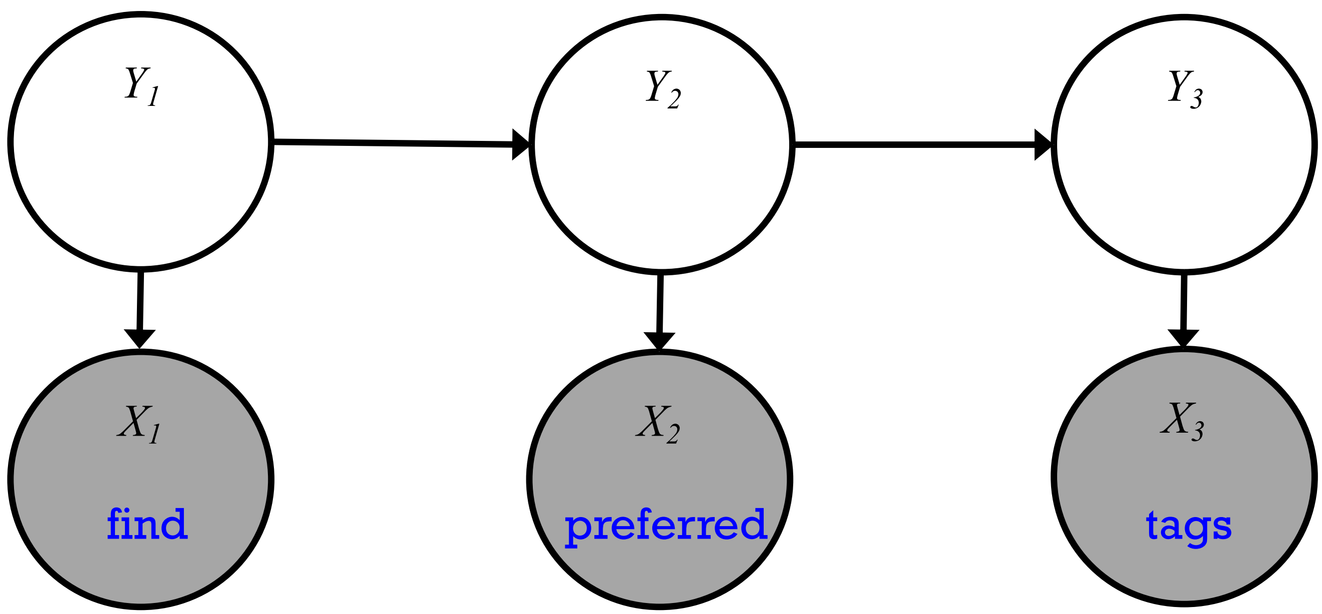
"belief that $Y_2 = \mathbf{a}$"

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

total weight of *all* paths through a

= $\alpha_2(\mathtt{a})$ 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

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

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

Assume $y_0 = START$

$y_{T+1} = END$

① Initialize $\alpha_0(START) = 1 \qquad \alpha_0(k) = 0 \quad \forall k \neq START$

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

② For $t = 1, \ldots, T$:

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

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

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

③ For $t = T, \ldots, T$:

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 Compute $p(\vec{x}) = \alpha_{T+1}(END)$ [Evaluation]

⑤ Compute $p(y_t = k \mid \vec{x}) = \dfrac{\alpha_t(k)\, \beta_t(k)}{p(\vec{x})}$ [Marginals]