10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Hidden Markov Models

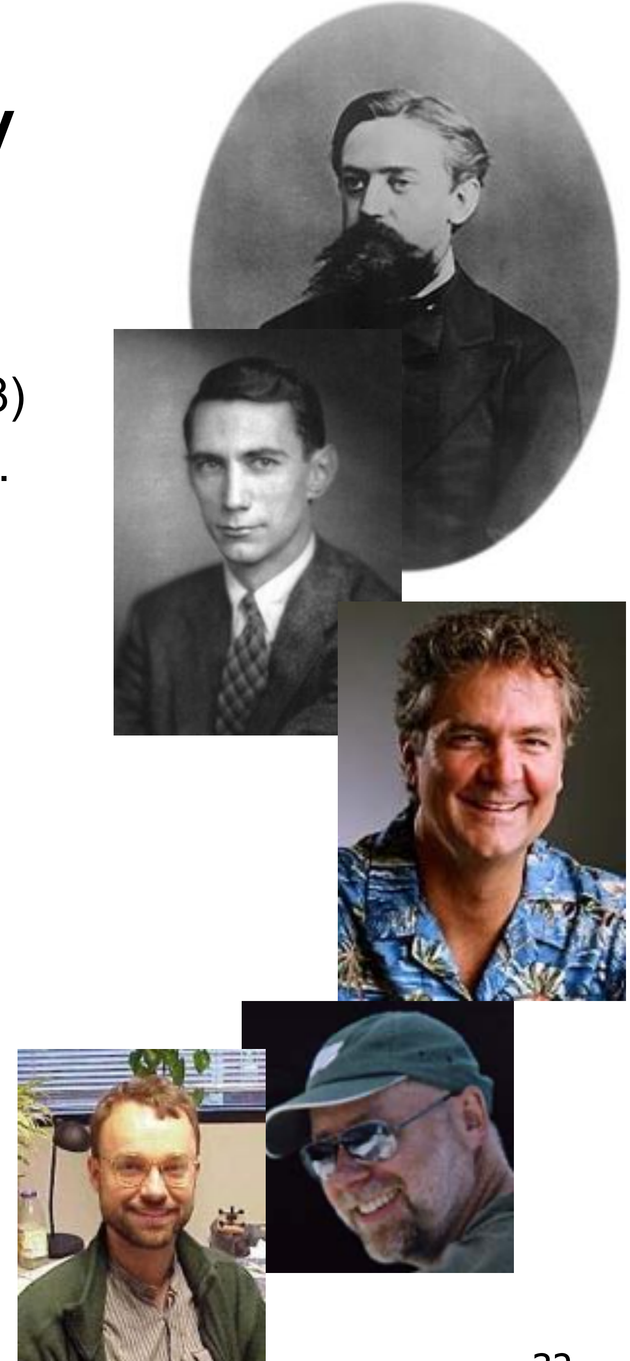
Matt Gormley
Lecture 20
Mar. 30, 2020

Reminders

- **Practice Problems for Exam 2**
  - Out: Fri, Mar 20
- **Midterm Exam 2**
  - Thu, Apr 2 – evening exam, details announced on Piazza
- **Homework 7: HMMs**
  - Out: Thu, Apr 02
  - Due: Fri, Apr 10 at 11:59pm
- **Today's In-Class Poll**
  - http://poll.mlcourse.org

HMMs: History

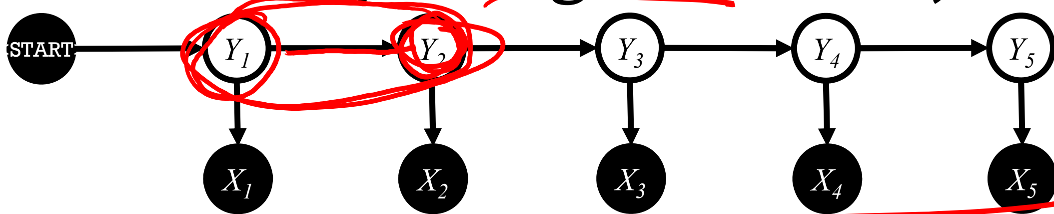
- Markov chains: Andrey Markov (1906)
  - Random walks and Brownian motion
- Used in Shannon's work on information theory (1948)
- Baum-Welsh learning algorithm: late 60's, early 70's.
  - Used mainly for speech in 60s-70s.
- Late 80's and 90's: David Haussler (major player in learning theory in 80's) 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
- …



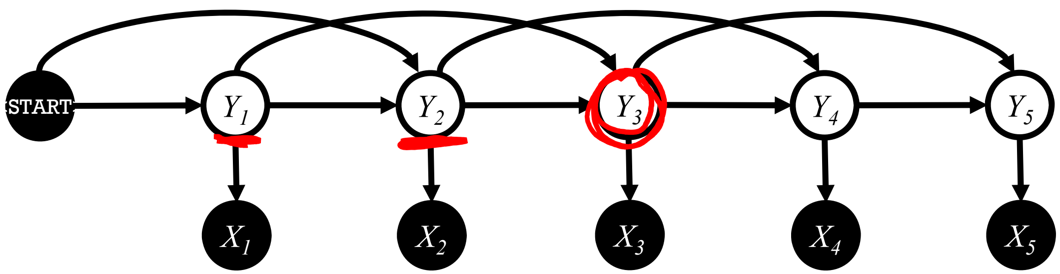
Slide from William Cohen

Higher-order HMMs

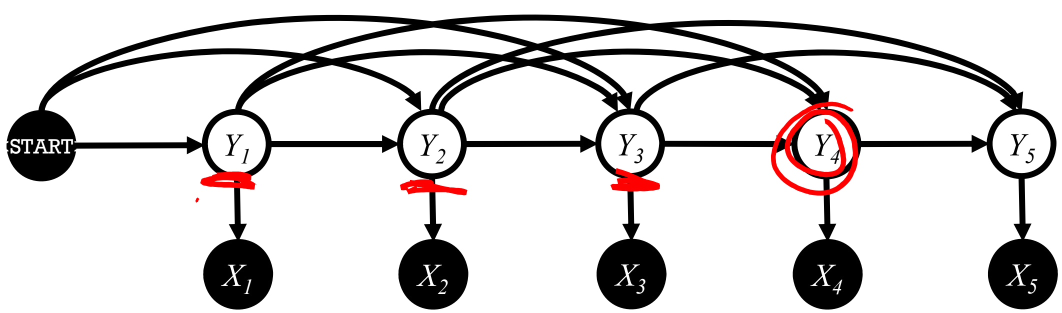
- 1st-order HMM (i.e. bigram HMM)



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

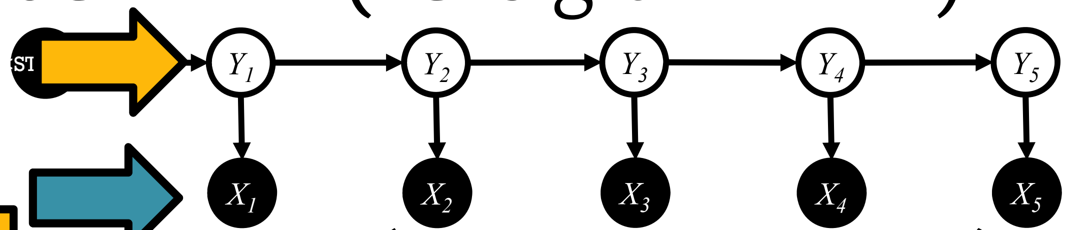


- 3rd-order HMM



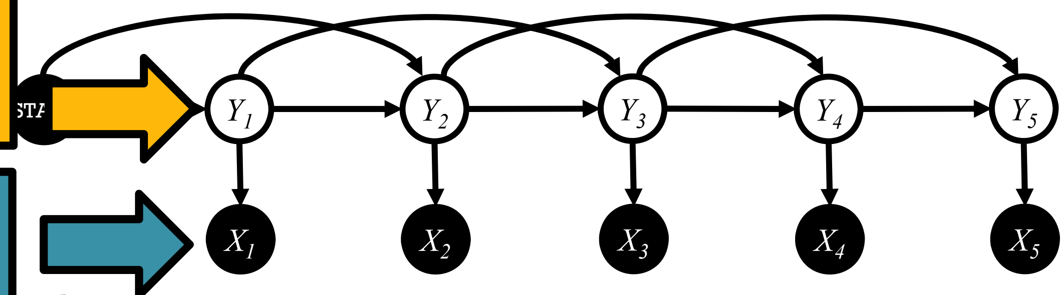
Higher-order HMMs

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



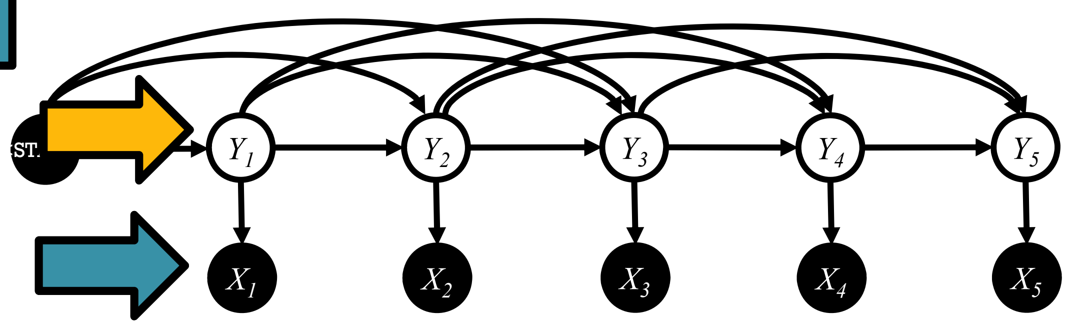
Hidden States, **y**

rder HMM (i.e. trigram HMM)



Observa-tions, **x**

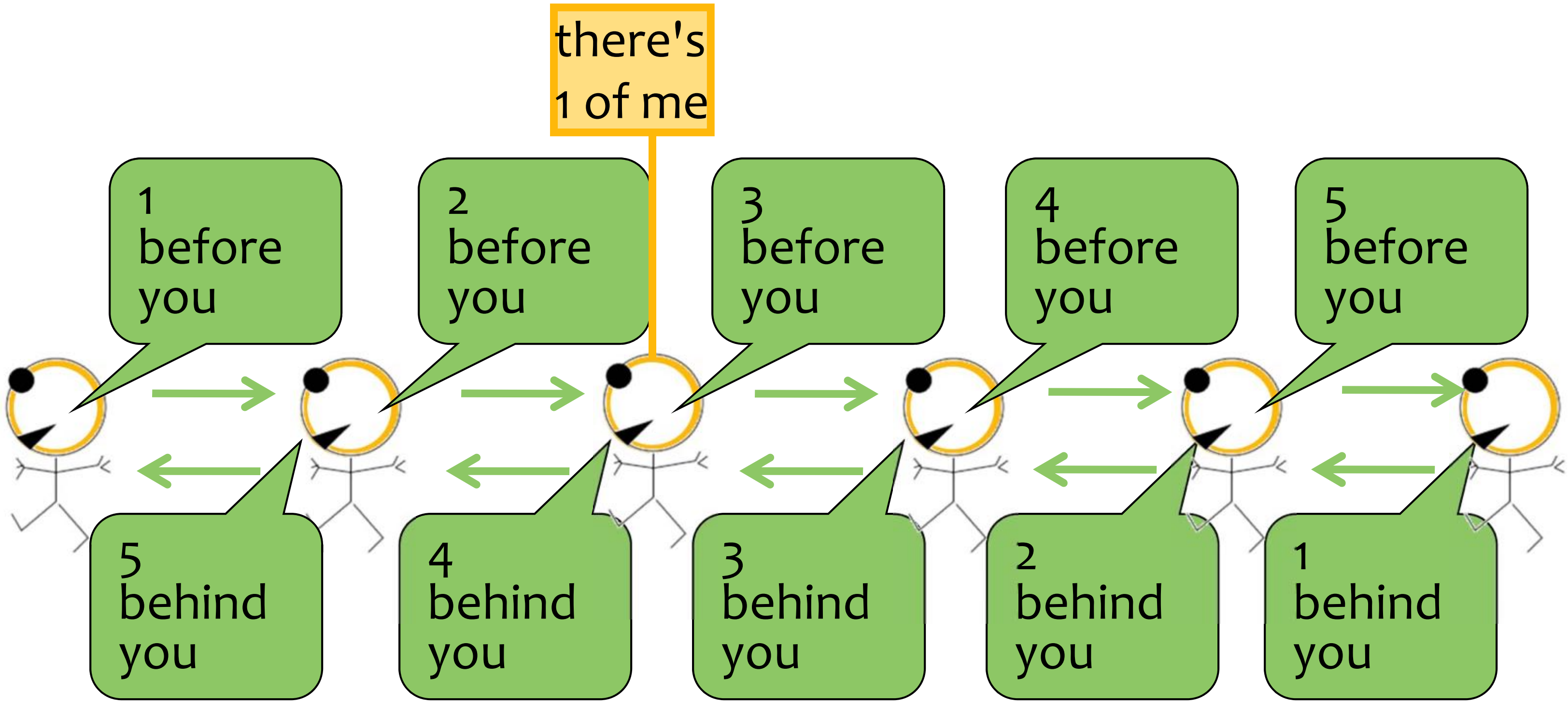
rder HMM



BACKGROUND: MESSAGE PASSING

Great Ideas in ML: Message Passing

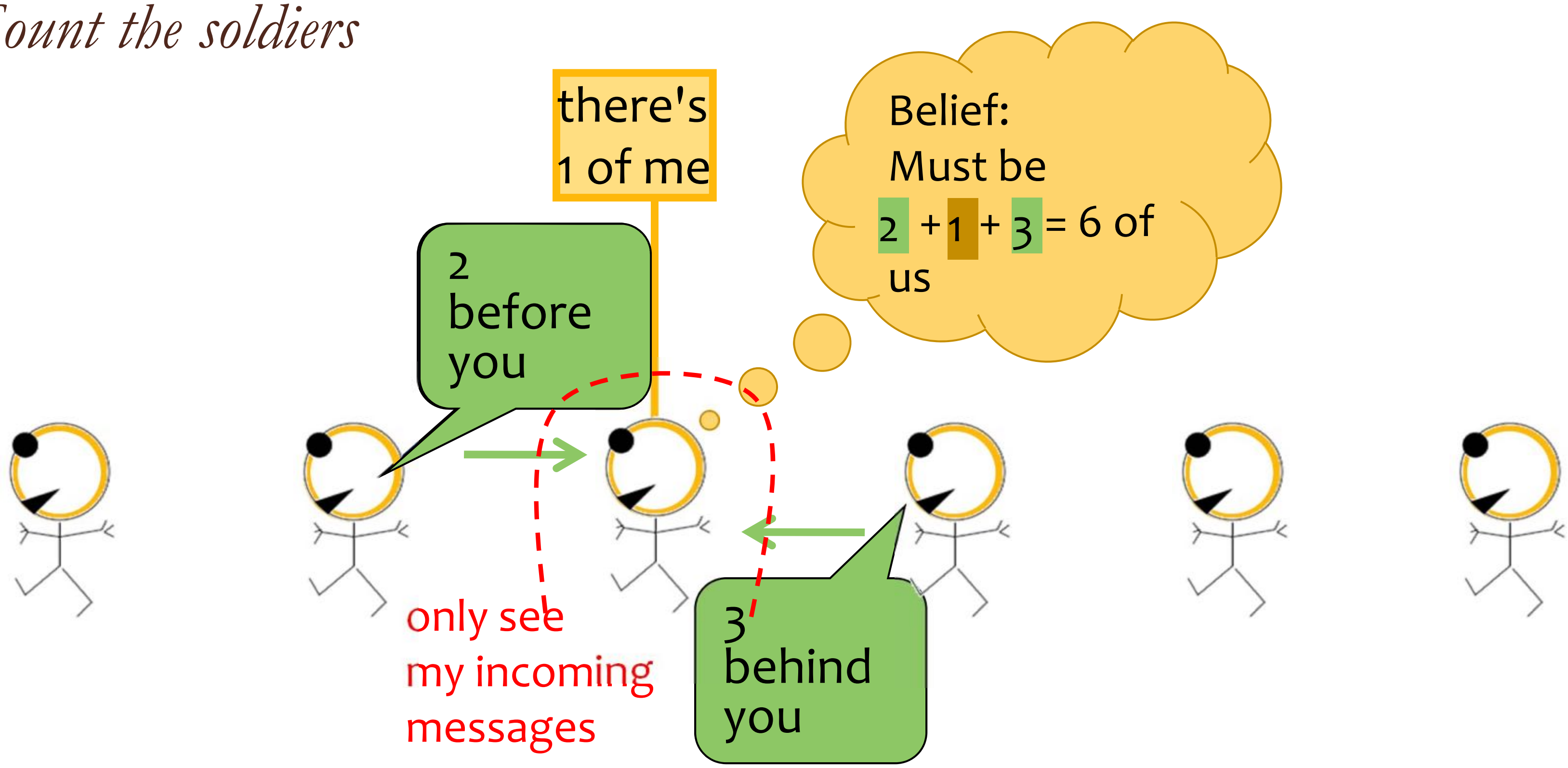
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

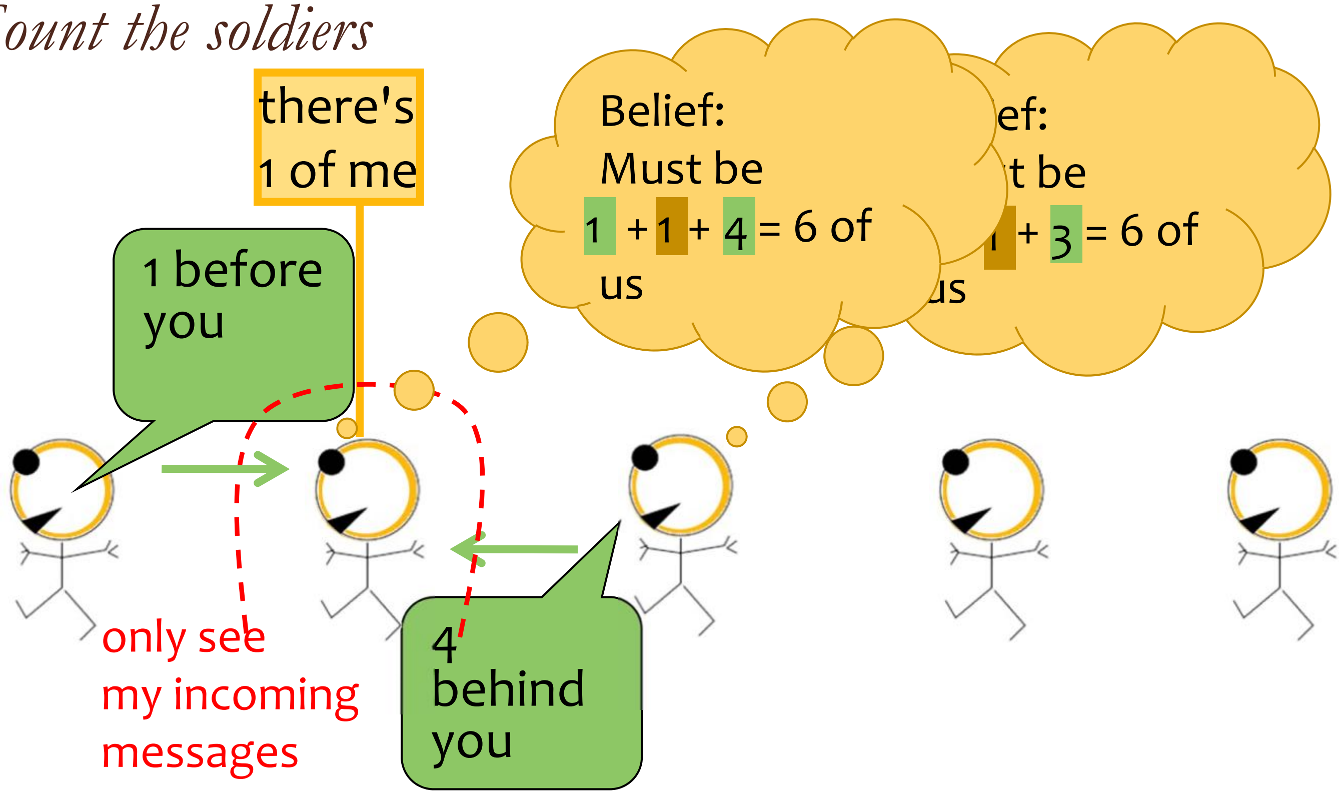
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

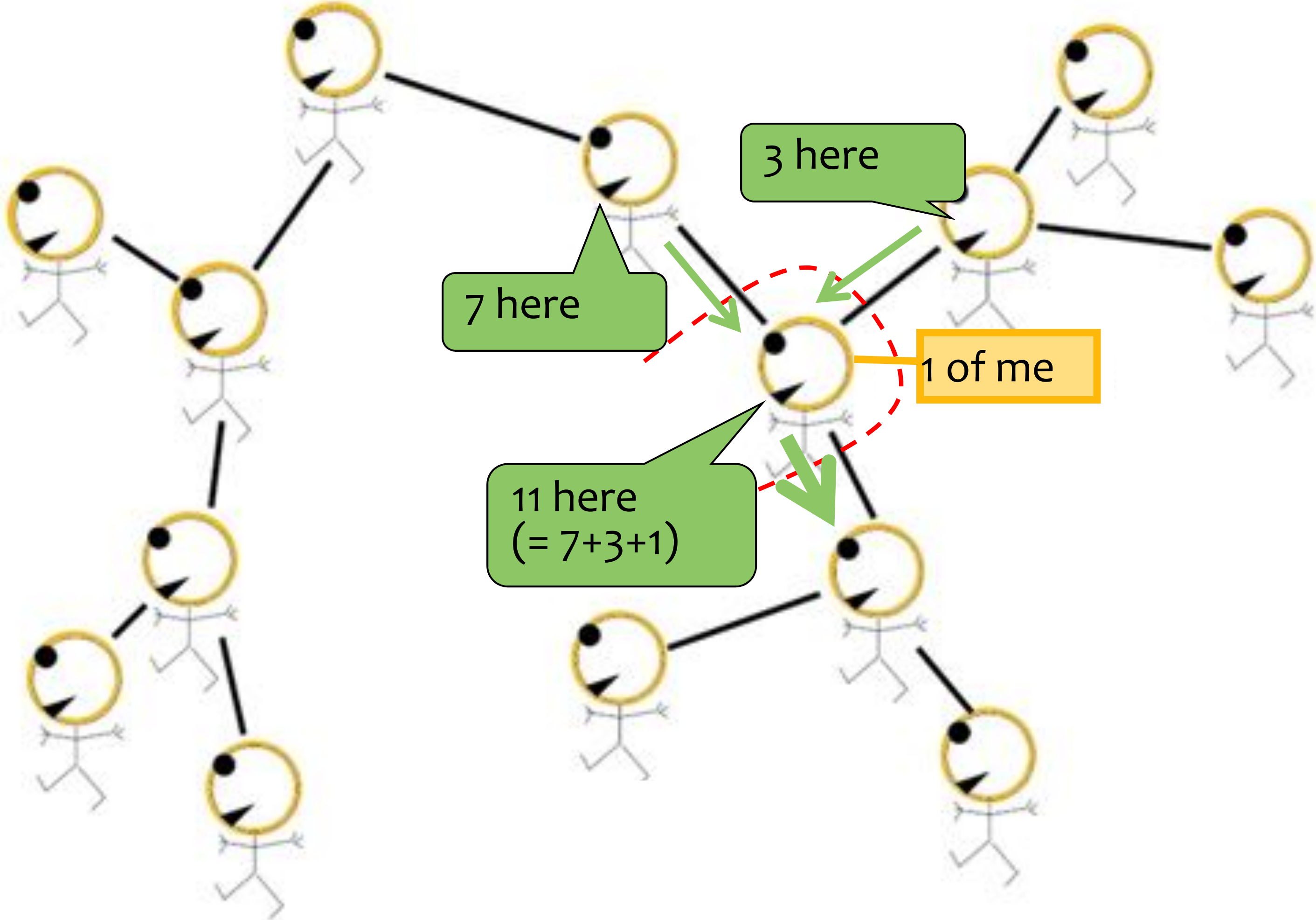
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

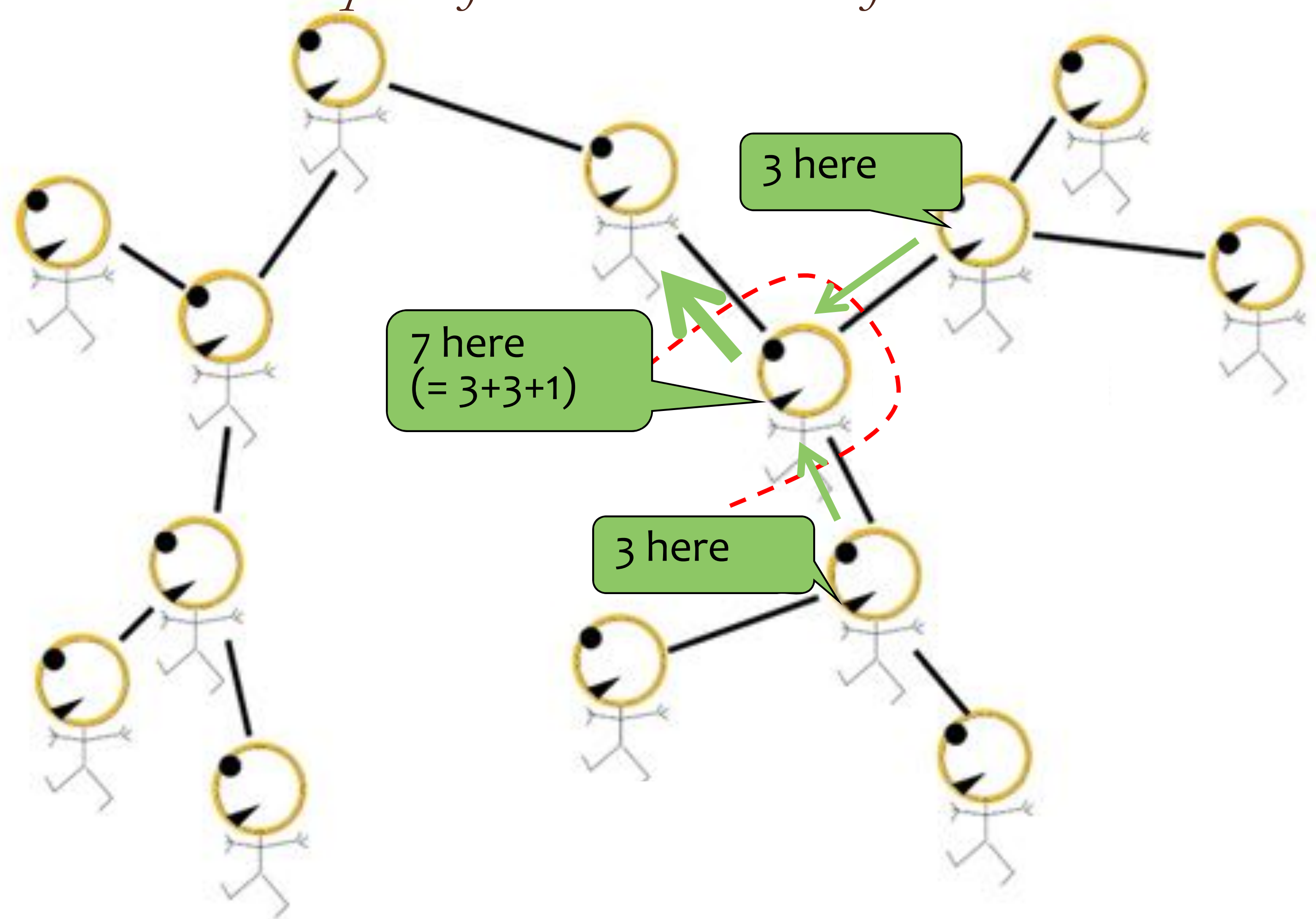
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

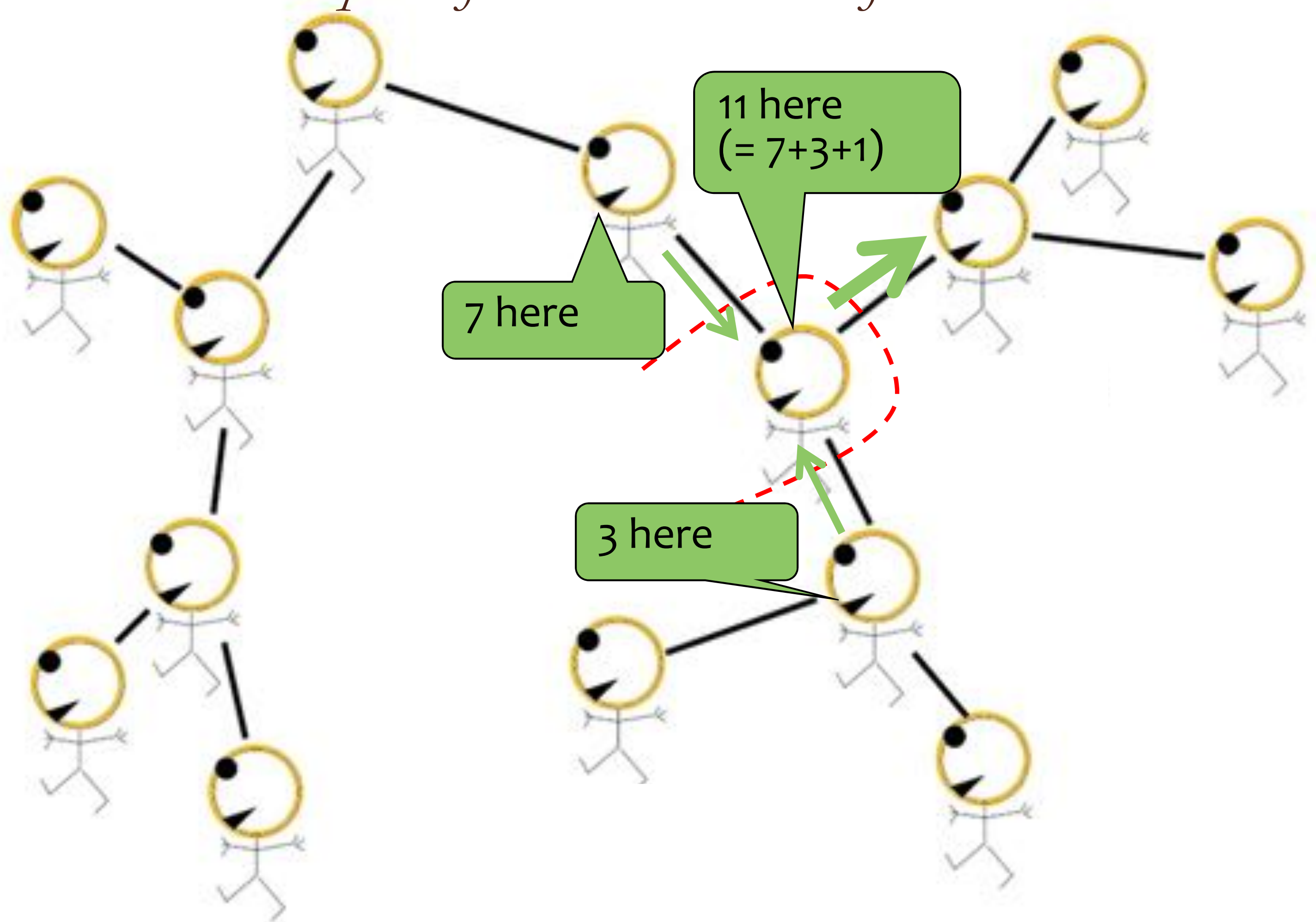
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

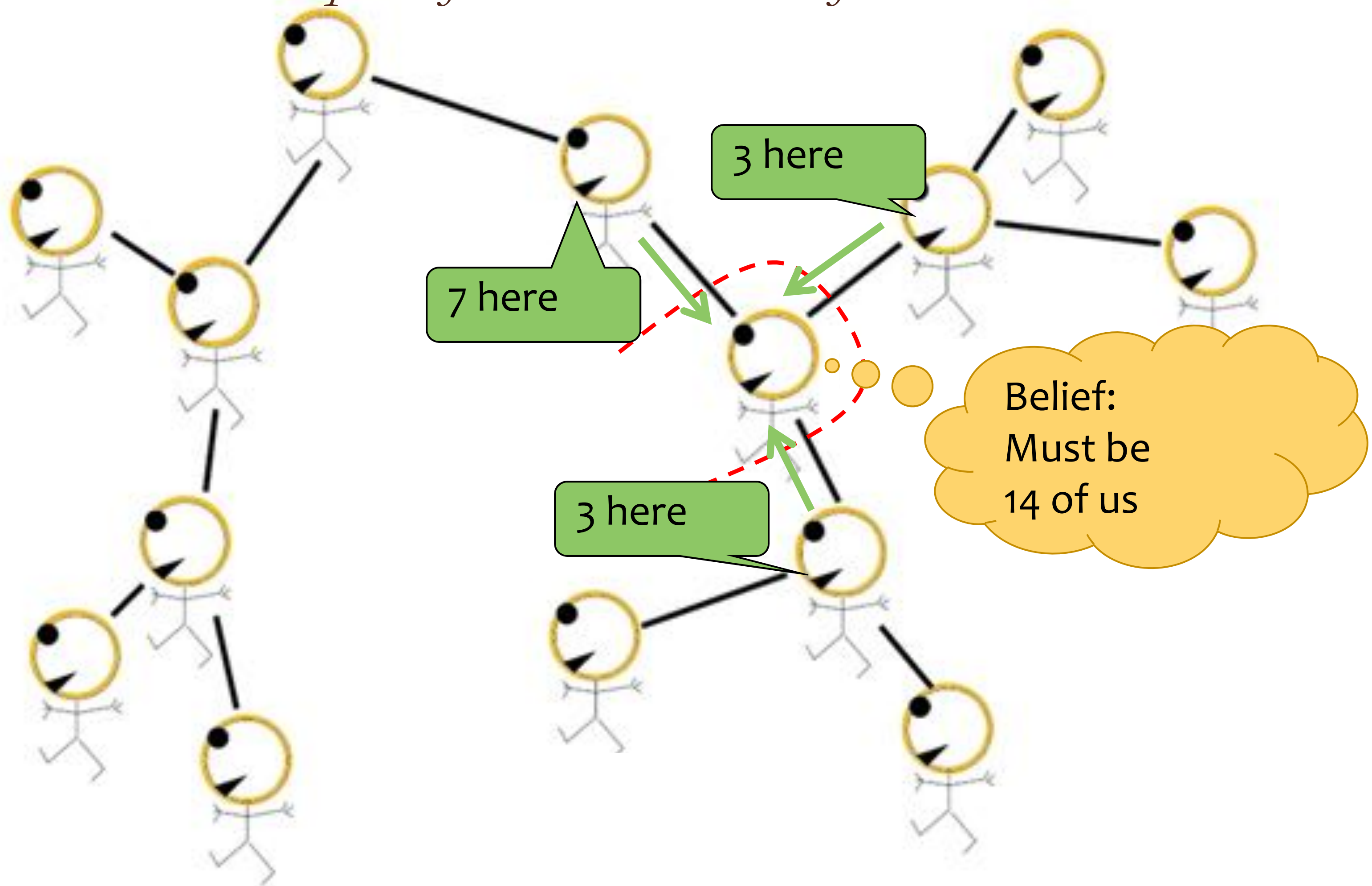
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

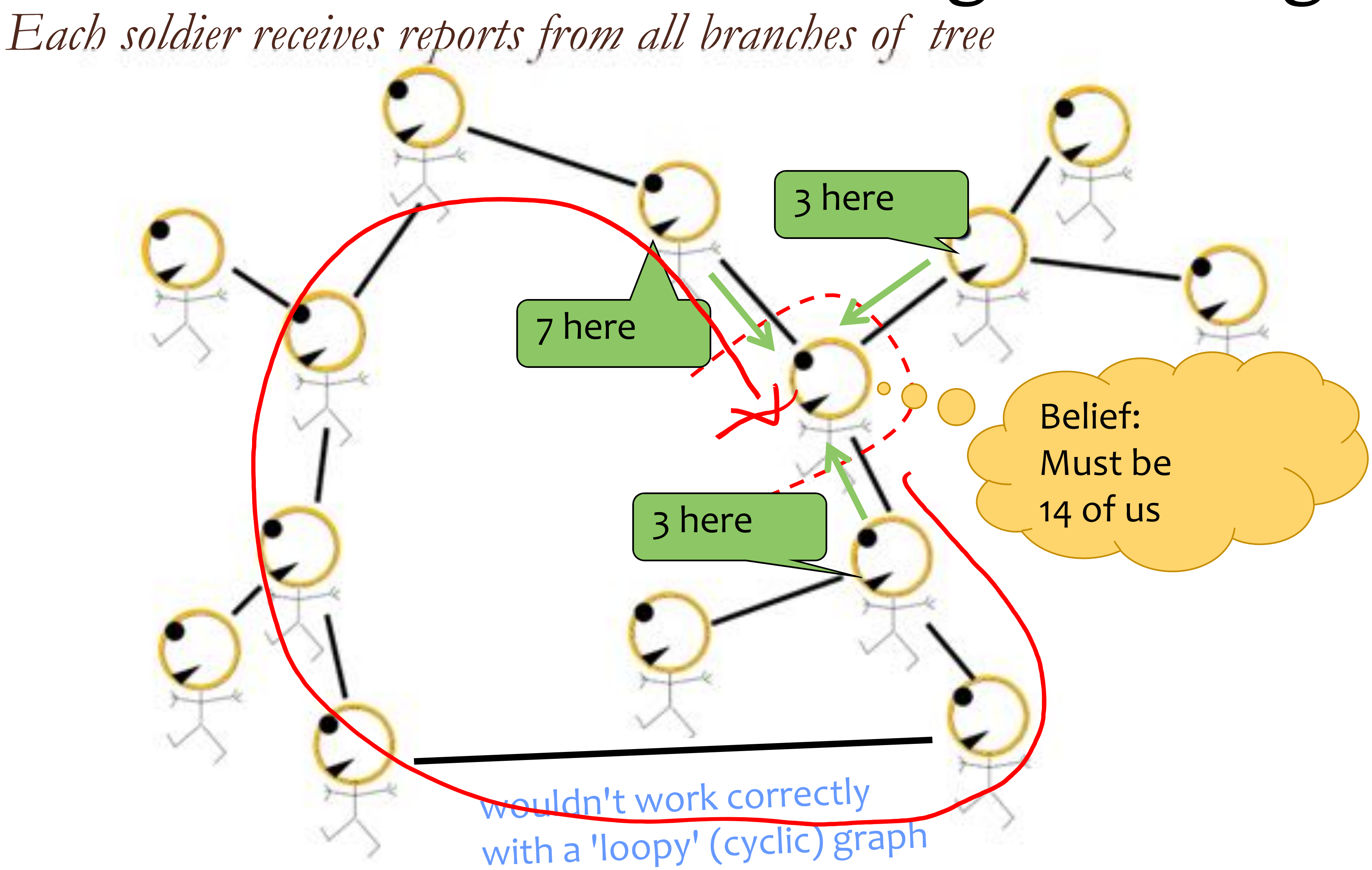
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing



adapted from MacKay (2003) textbook

THE FORWARD-BACKWARD ALGORITHM

Inference

Question:

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

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

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

Question:

True or False: Suppose each hidden state takes K values. The **marginal probability** of a hidden state $\mathbf{y}_t$ given the observations $\mathbf{x}$ is given by:

$$P(Y_t = y_t | \mathbf{X} = \mathbf{x}) = \sum_{j=1}^{K} B_{j,y_t}$$

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

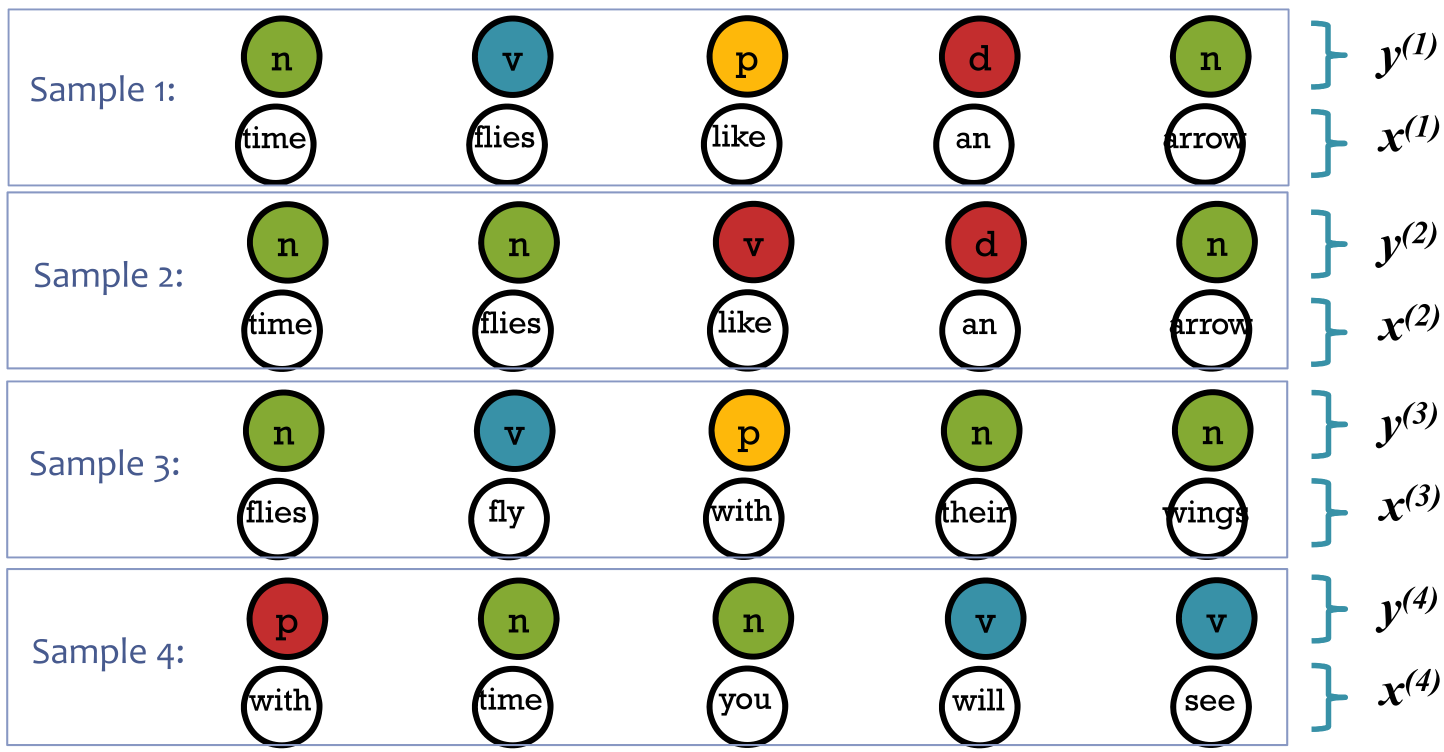
Inference for HMMs

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

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

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



Inference for HMMs

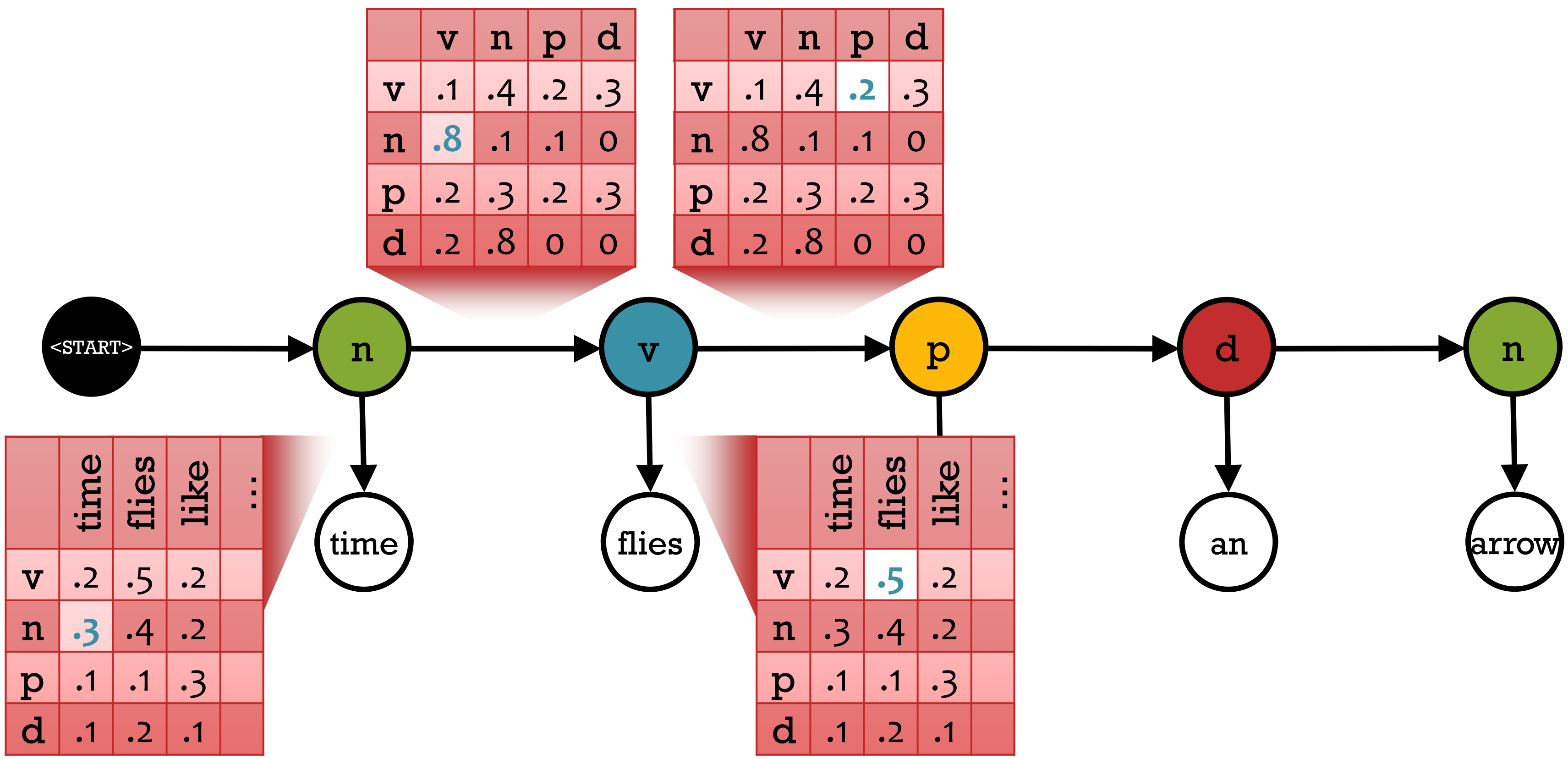
Whiteboard

- Forward-backward search space

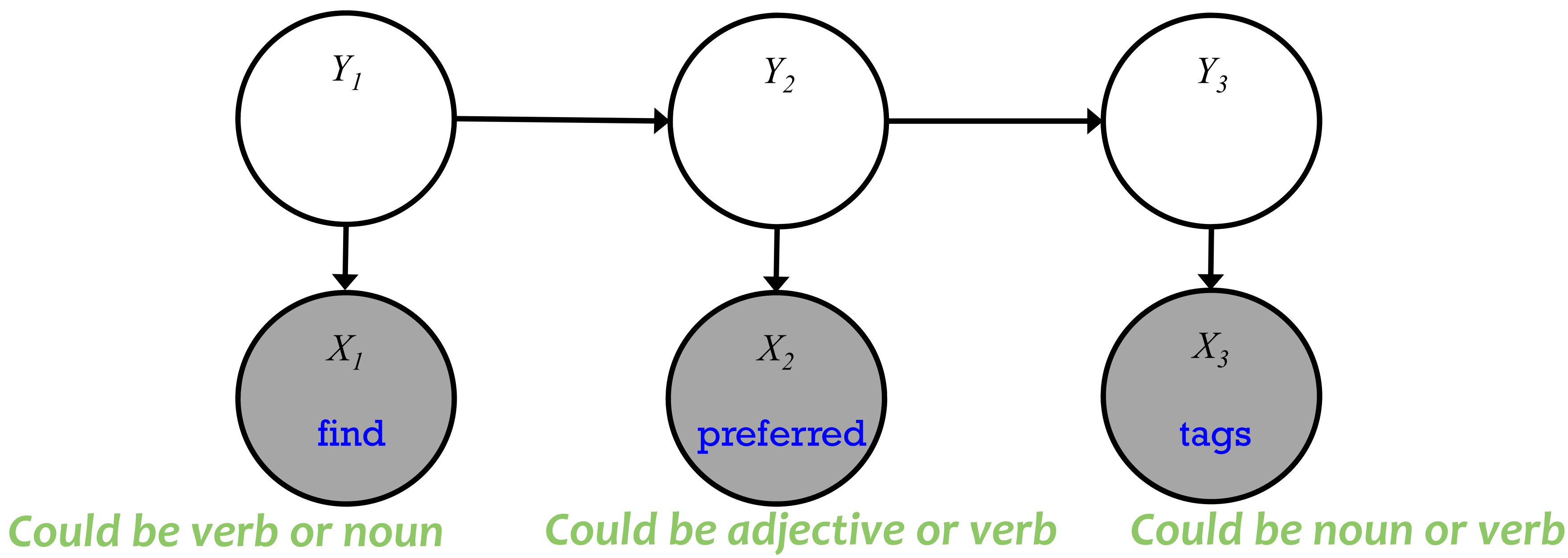
Hidden Markov Model

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

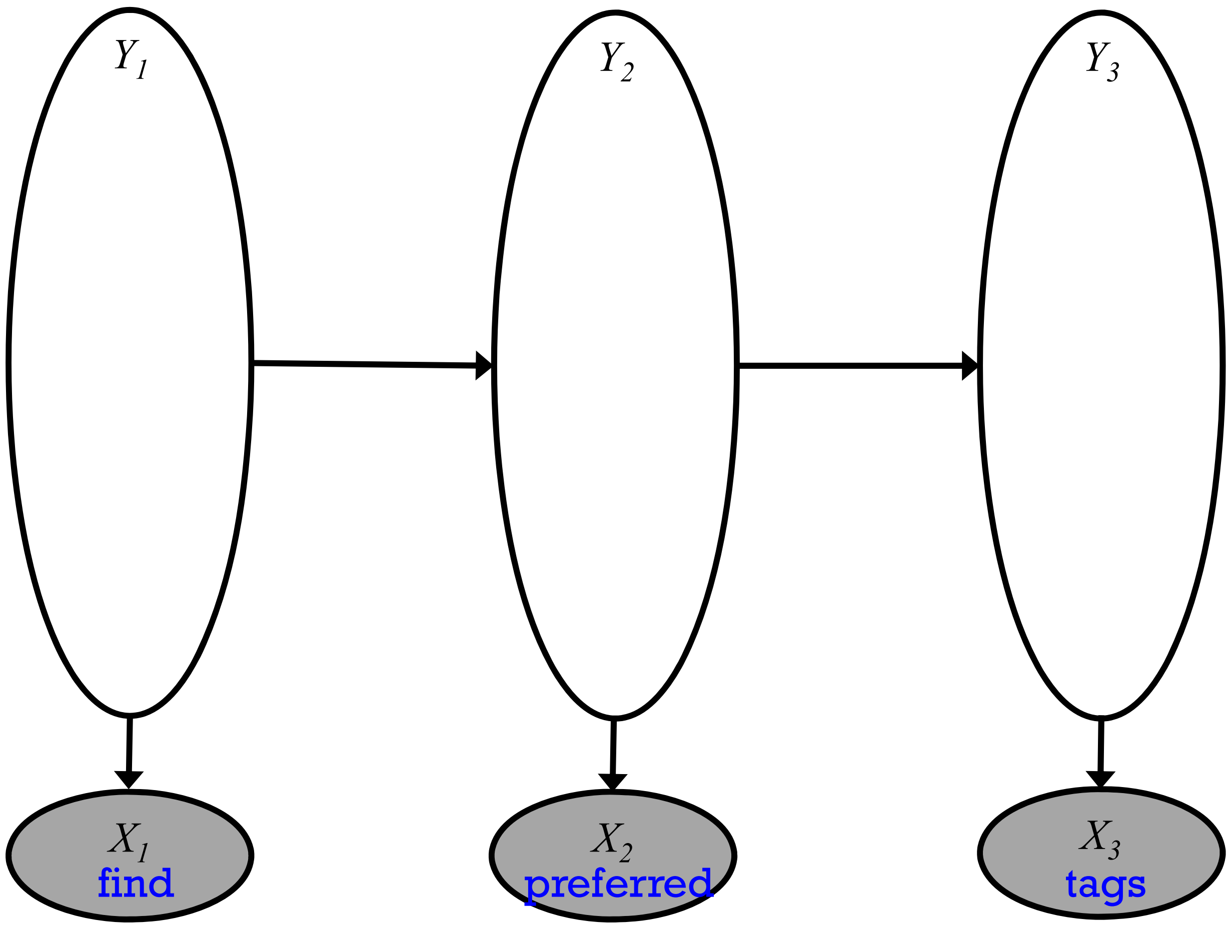
$$p(\text{n, v, p, d, n, time, flies, like, an, arrow}) = (.3 * .8 * .2 * .5 * \ldots)$$



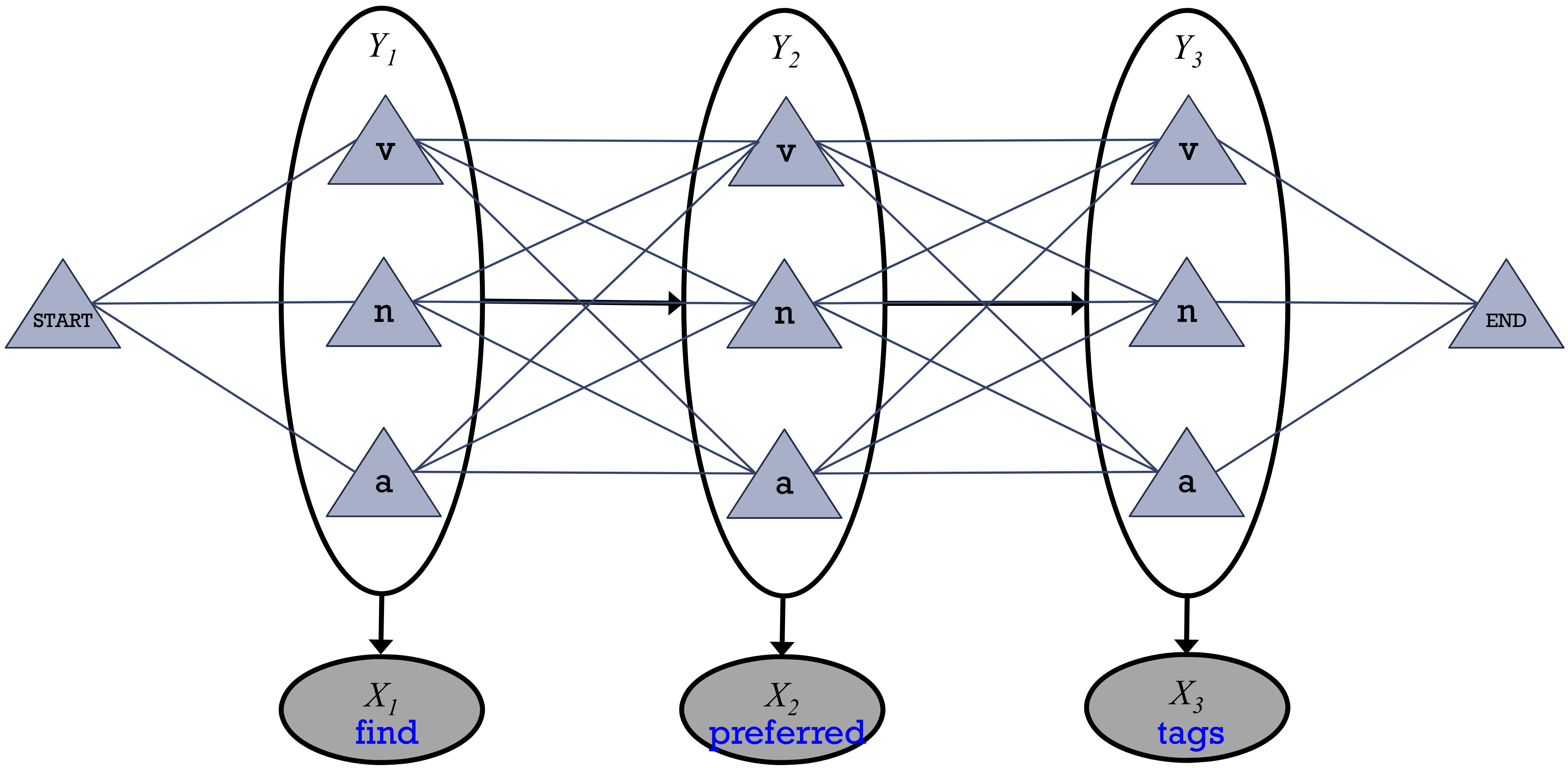
Forward-Backward Algorithm



Forward-Backward Algorithm

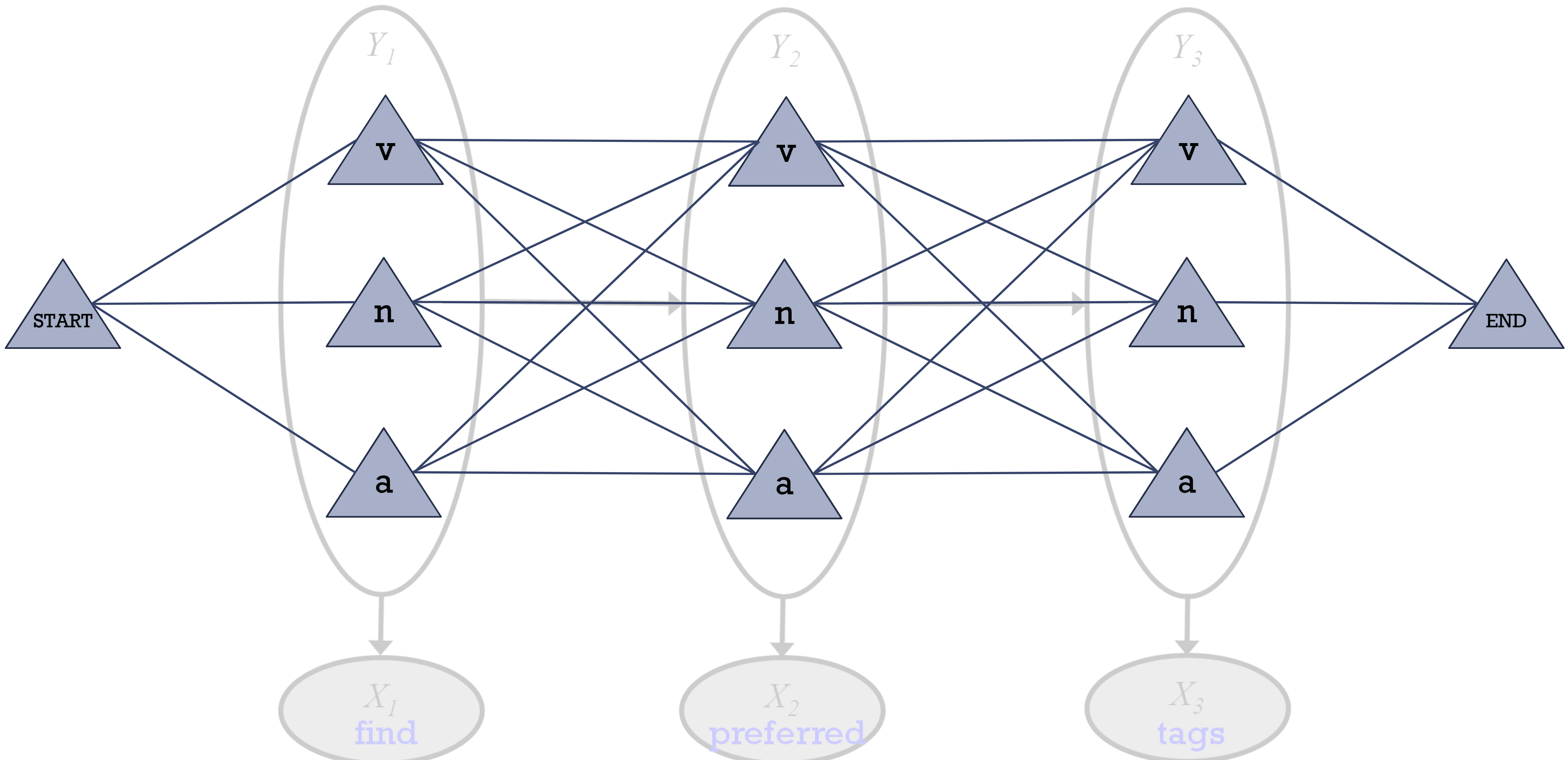


Forward-Backward Algorithm



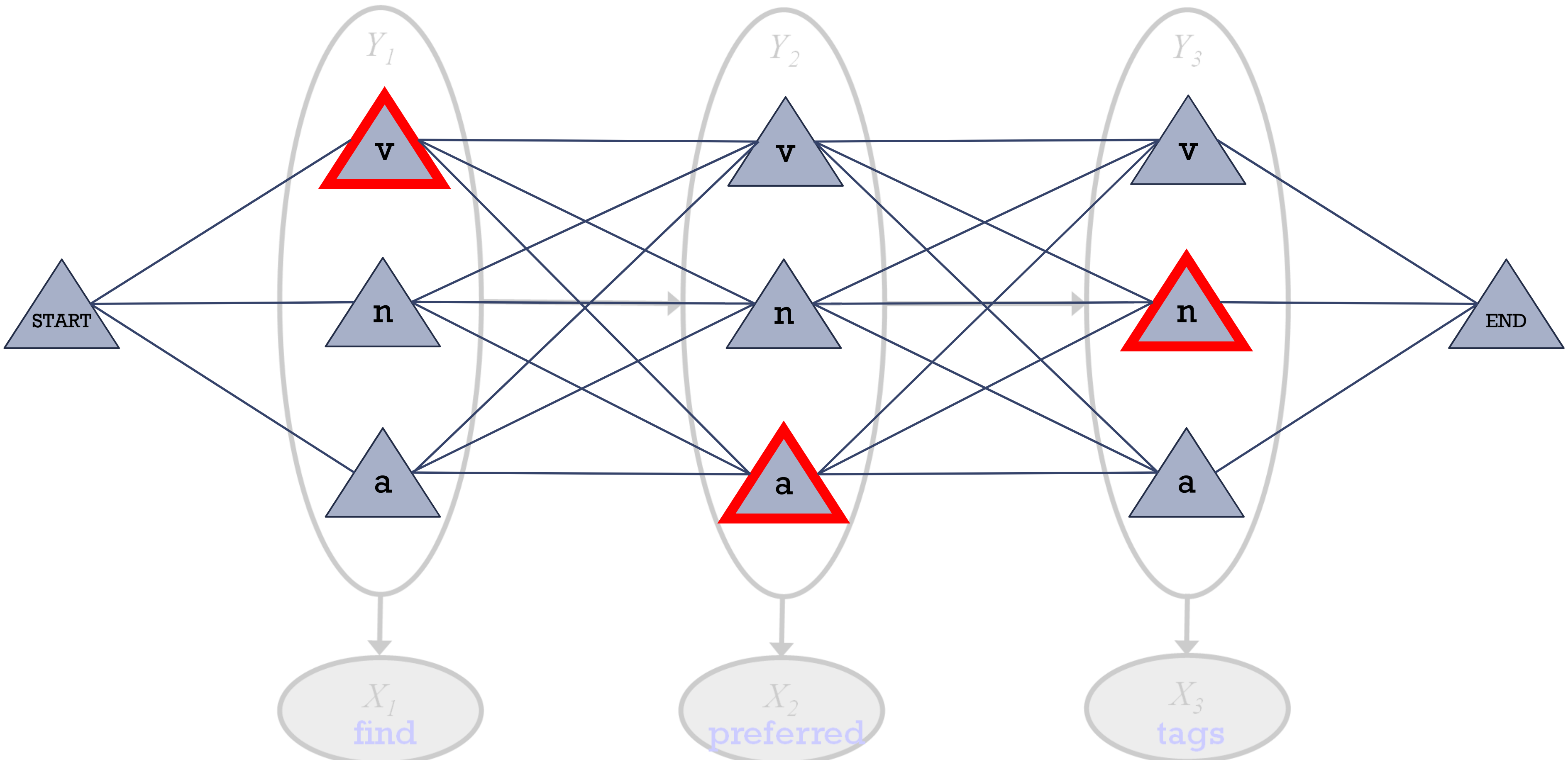
- Let's show the possible *values* for each variable

Forward-Backward Algorithm



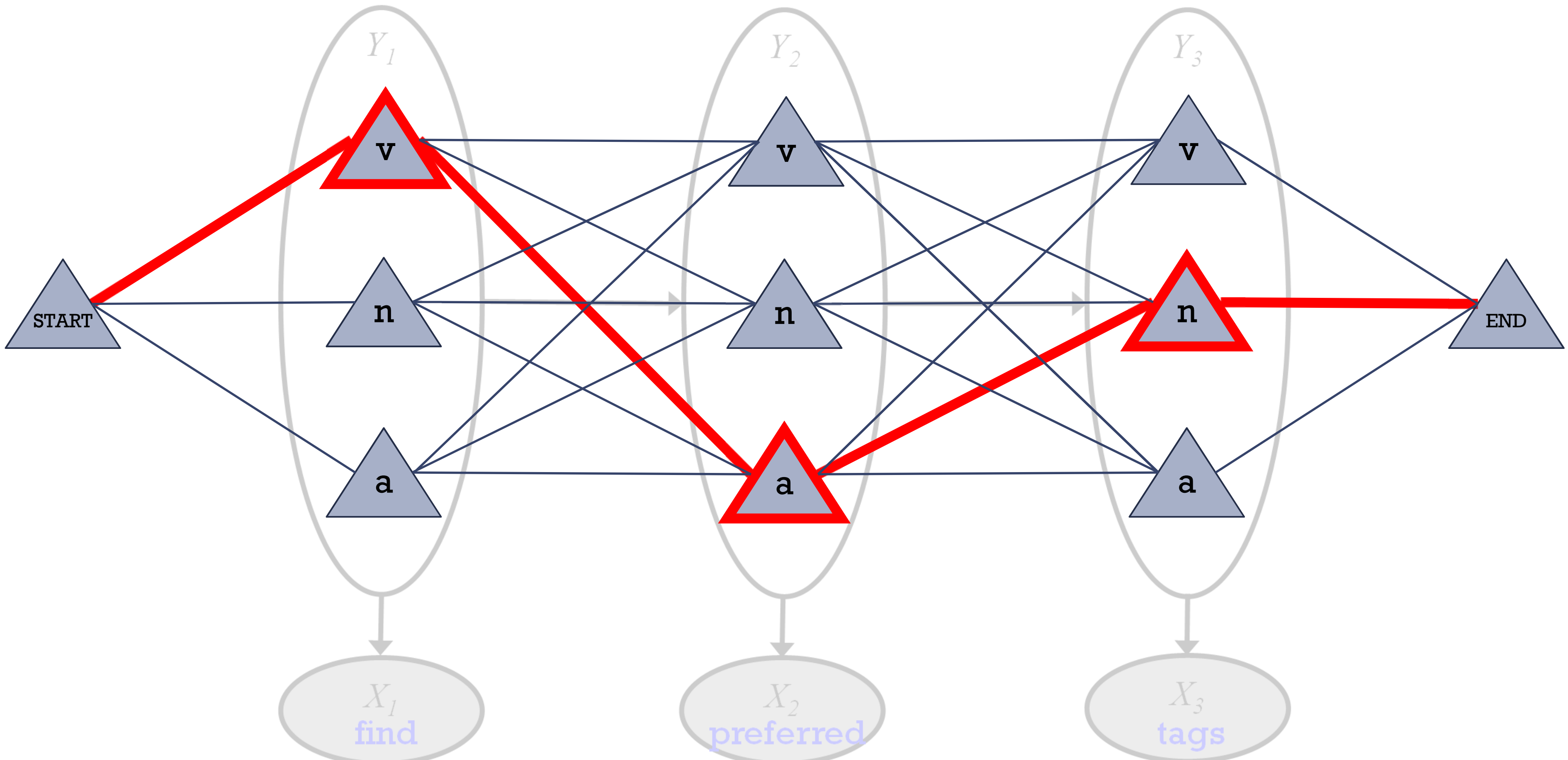
- Let's show the possible *values* for each variable

Forward-Backward Algorithm



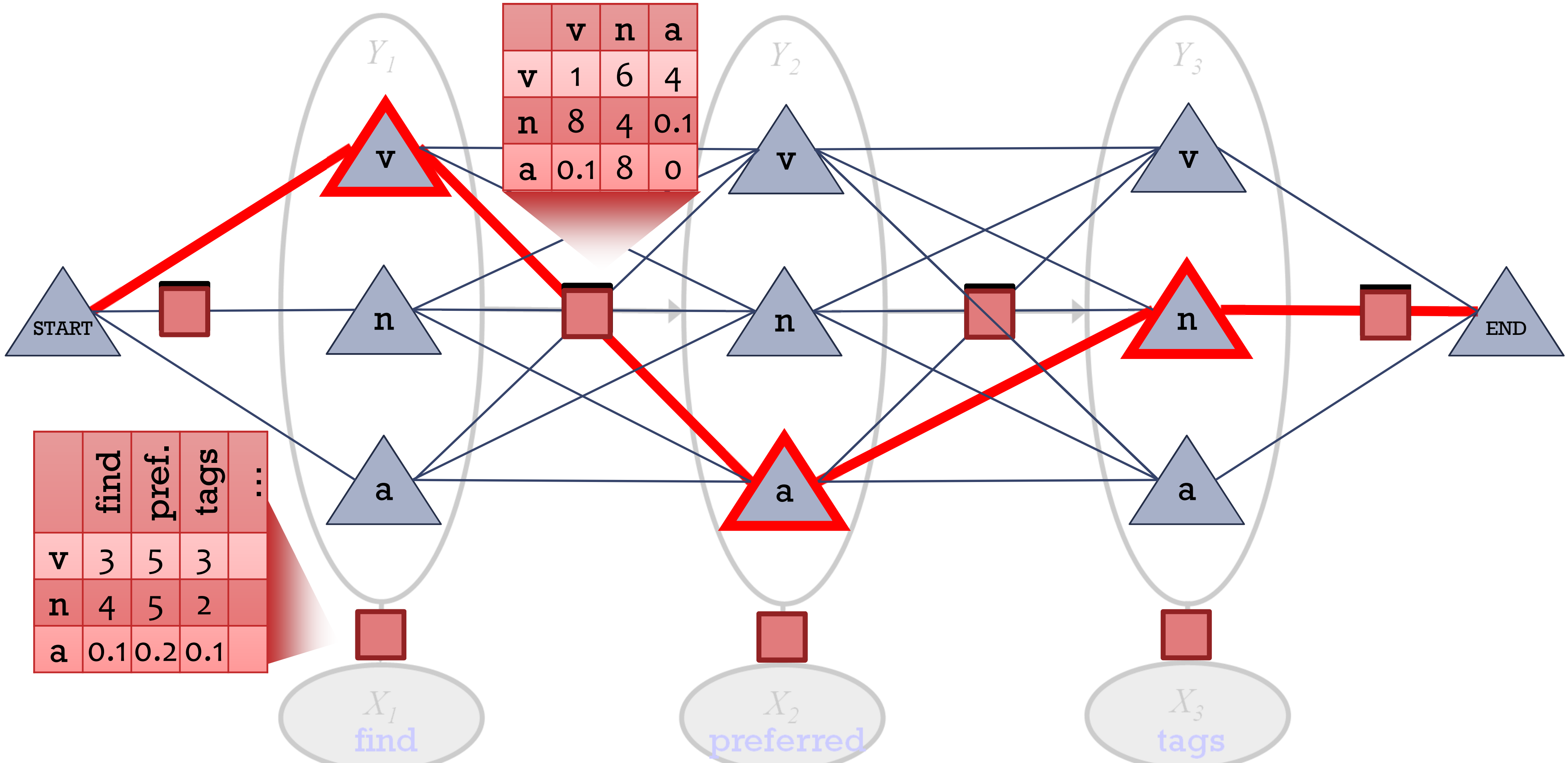
- Let's show the possible *values* for each variable
- One possible assignment

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

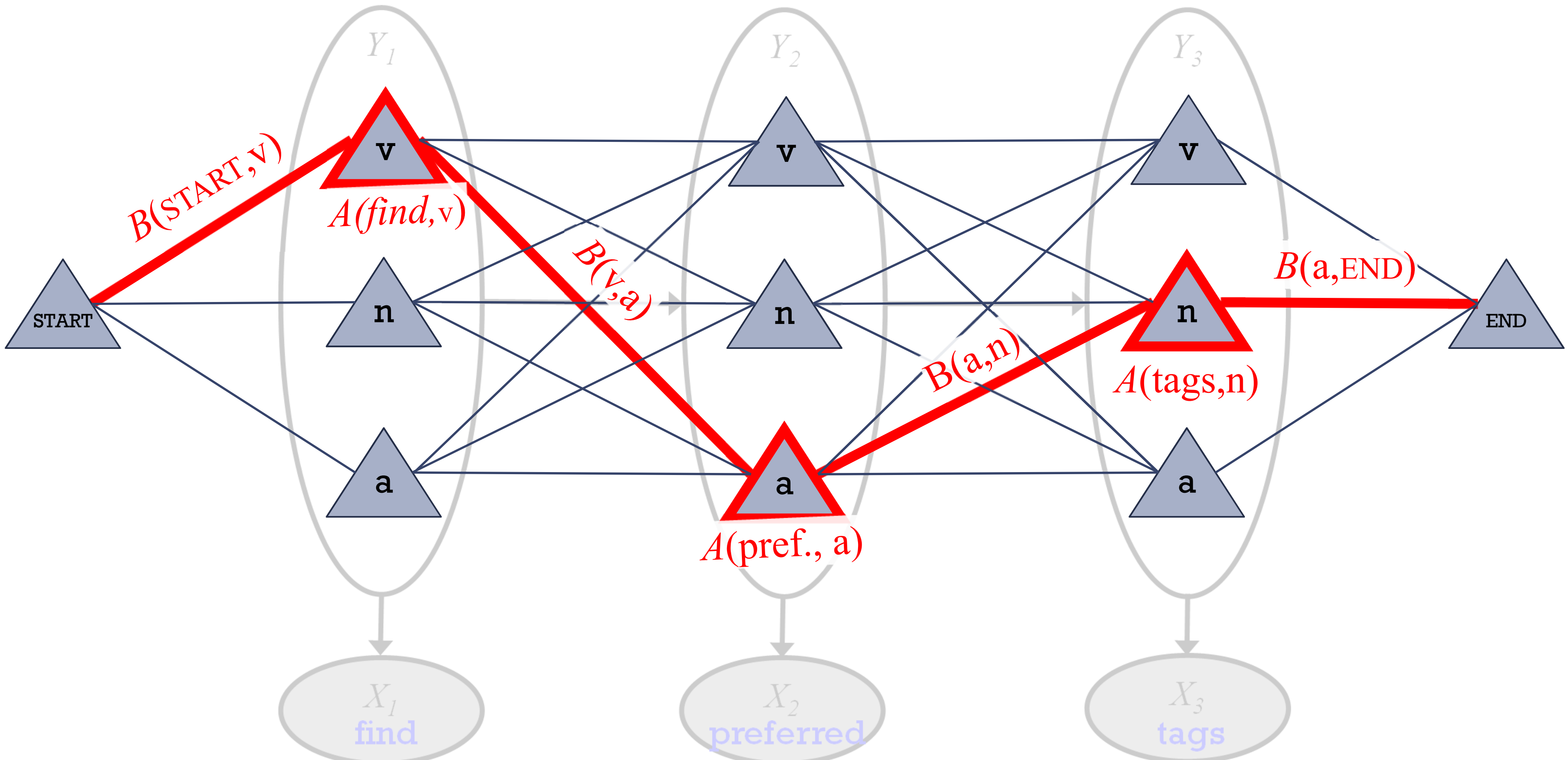


	v	n	a
v	1	6	4
n	8	4	0.1
a	0.1	8	0

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

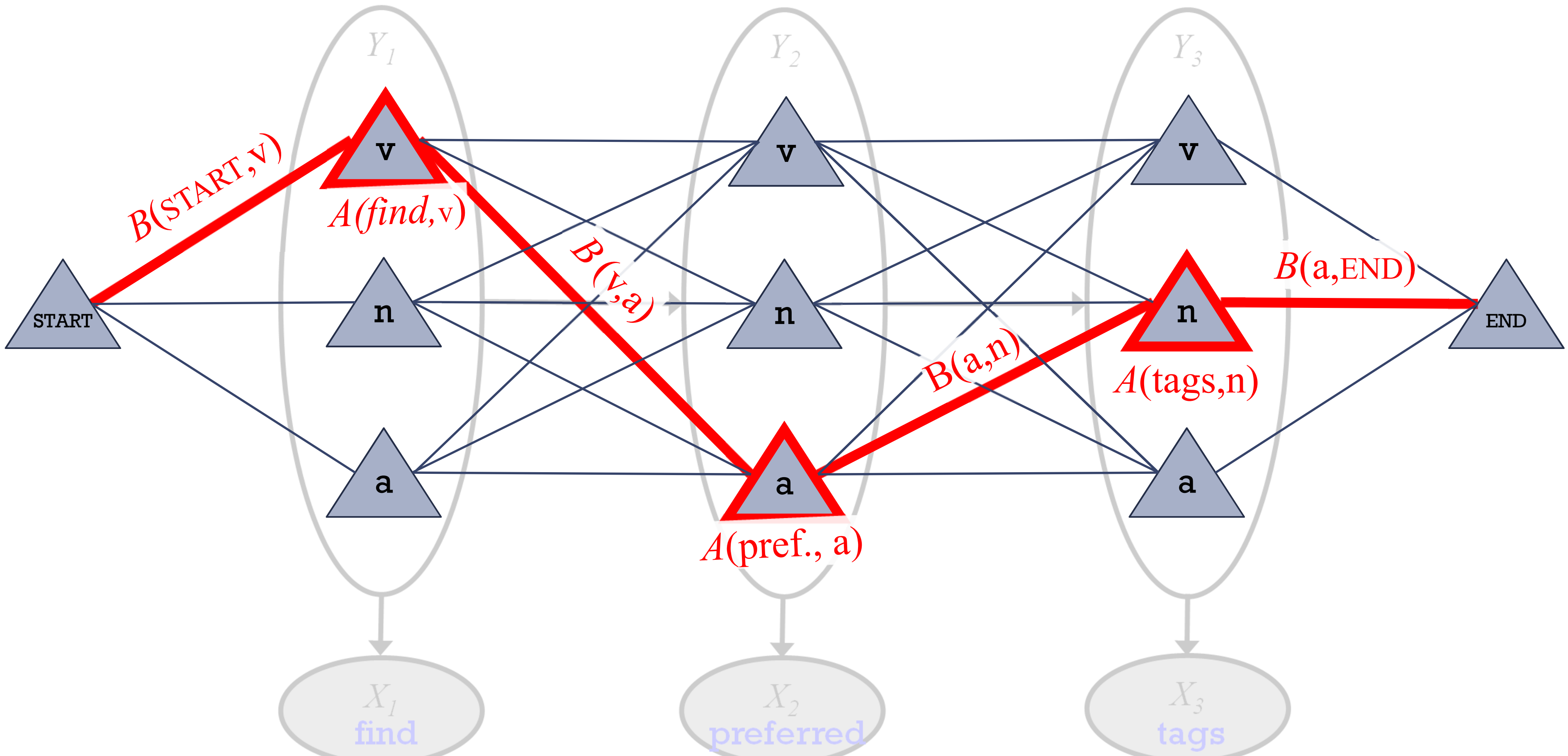
- Let’s show the possible *values* for each variable
- One possible assignment
- And what the 7 transition / emission factors think of it …

Viterbi Algorithm: Most Probable Assignment



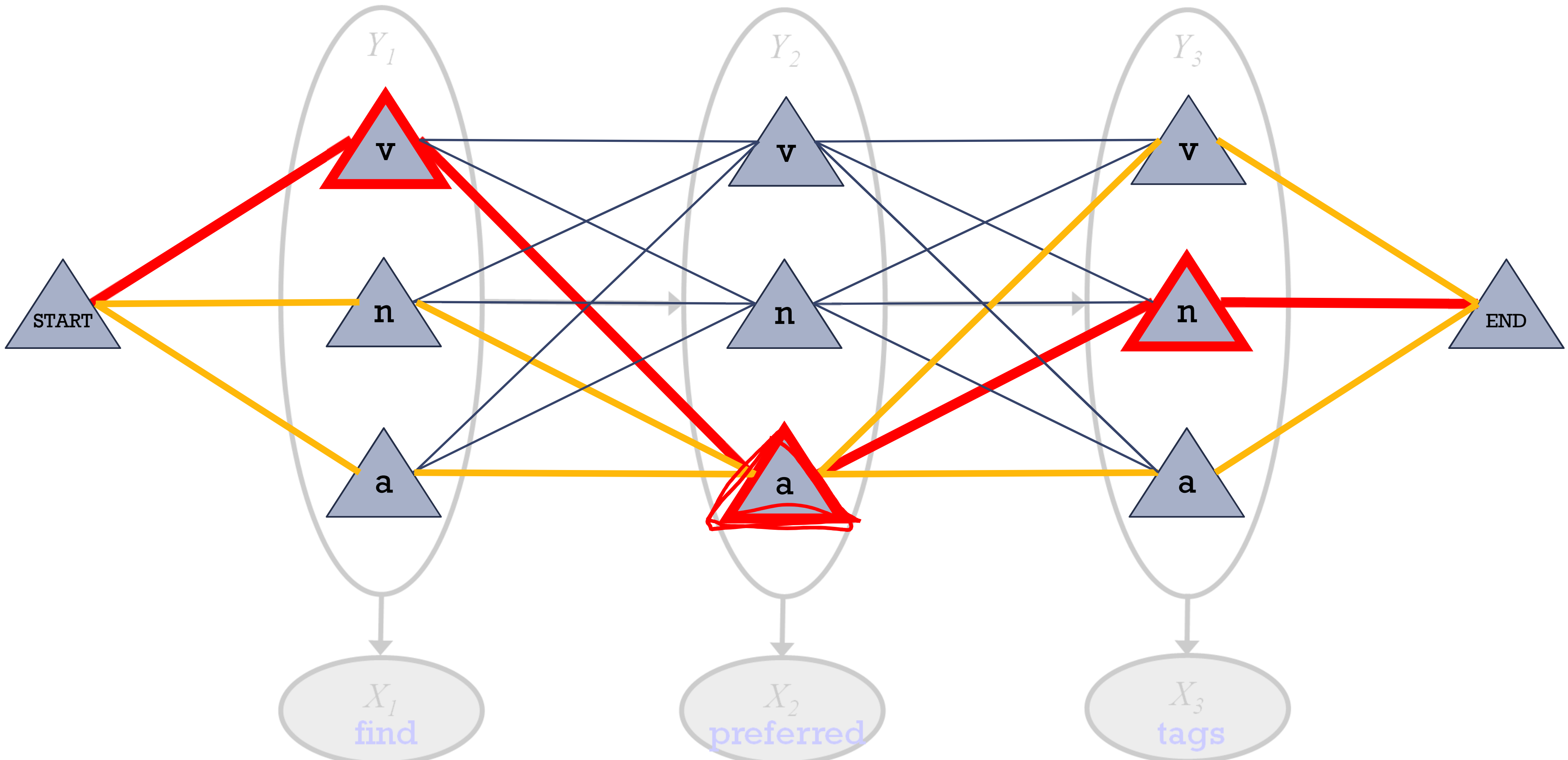
- So p(**v a n**) = (1/Z) * product of 7 numbers
- Numbers associated with edges and nodes of path
- Most probable assignment = **path with highest product**

Viterbi Algorithm: Most Probable Assignment



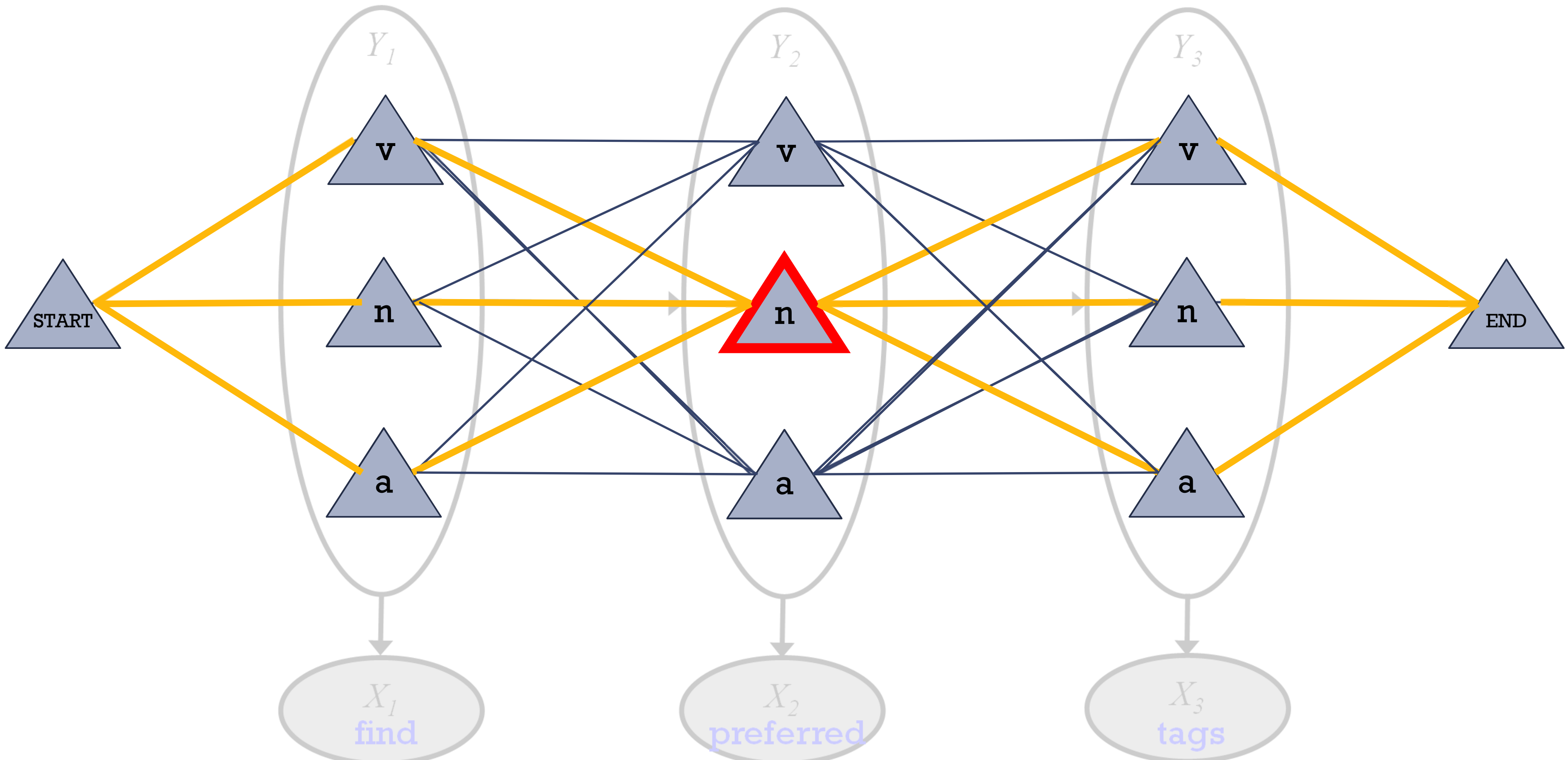
- So p(**v a n** | $\vec{x}$) = (1/Z) * product weight of one path

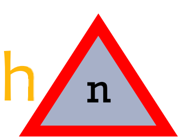
Forward-Backward Algorithm: Finds Marginals



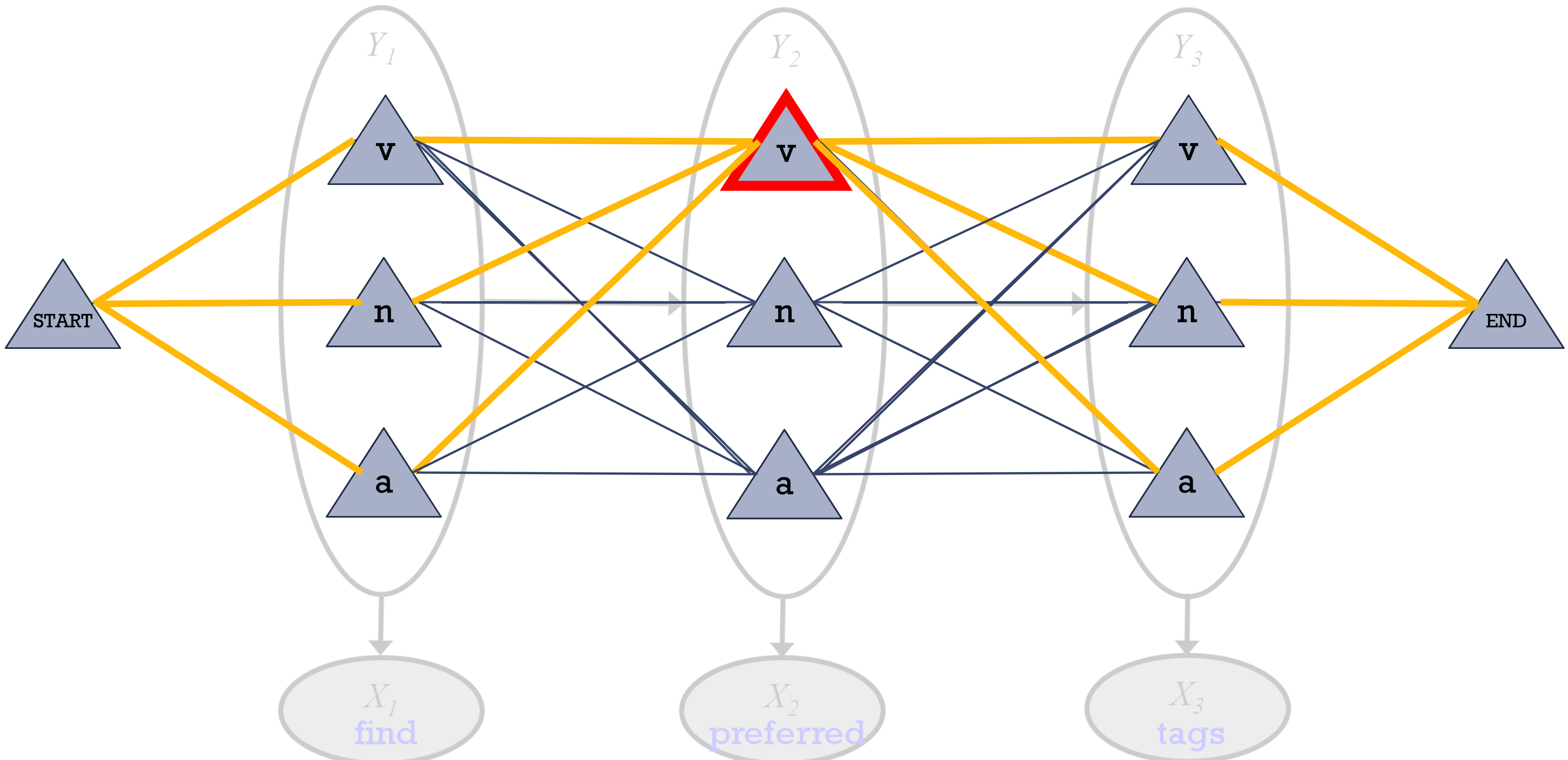
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability p(Y_2 = a | $\vec{x}$)
  = (1/Z) * total weight of *all* paths through a

Forward-Backward Algorithm: Finds Marginals



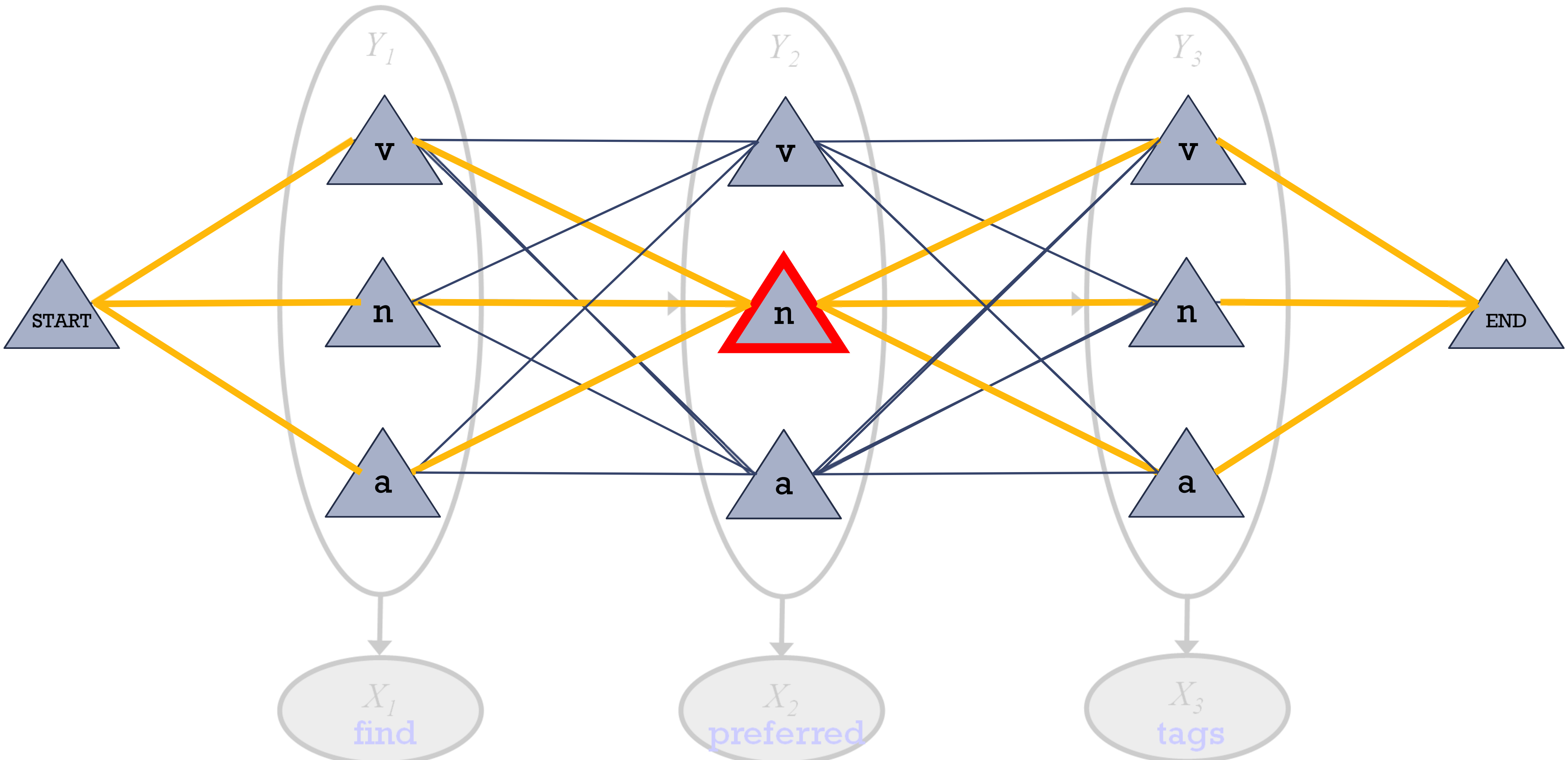
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(Y_2 = n)$
  = (1/Z) * total weight of *all* paths through 

Forward-Backward Algorithm: Finds *Marginals*



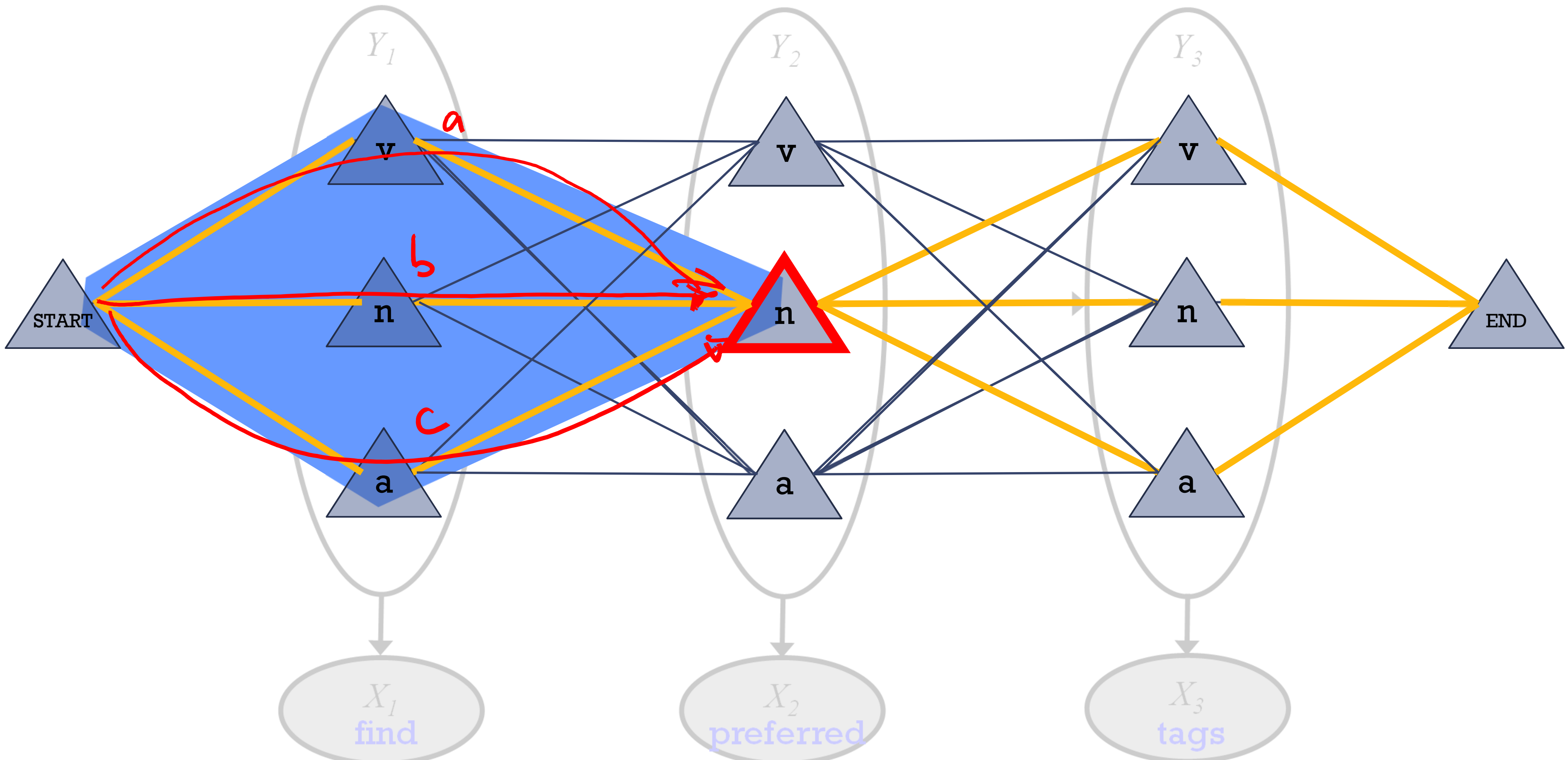
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(Y_2 = v)$
  = (1/Z) * total weight of *all* paths through **v**

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 **n**

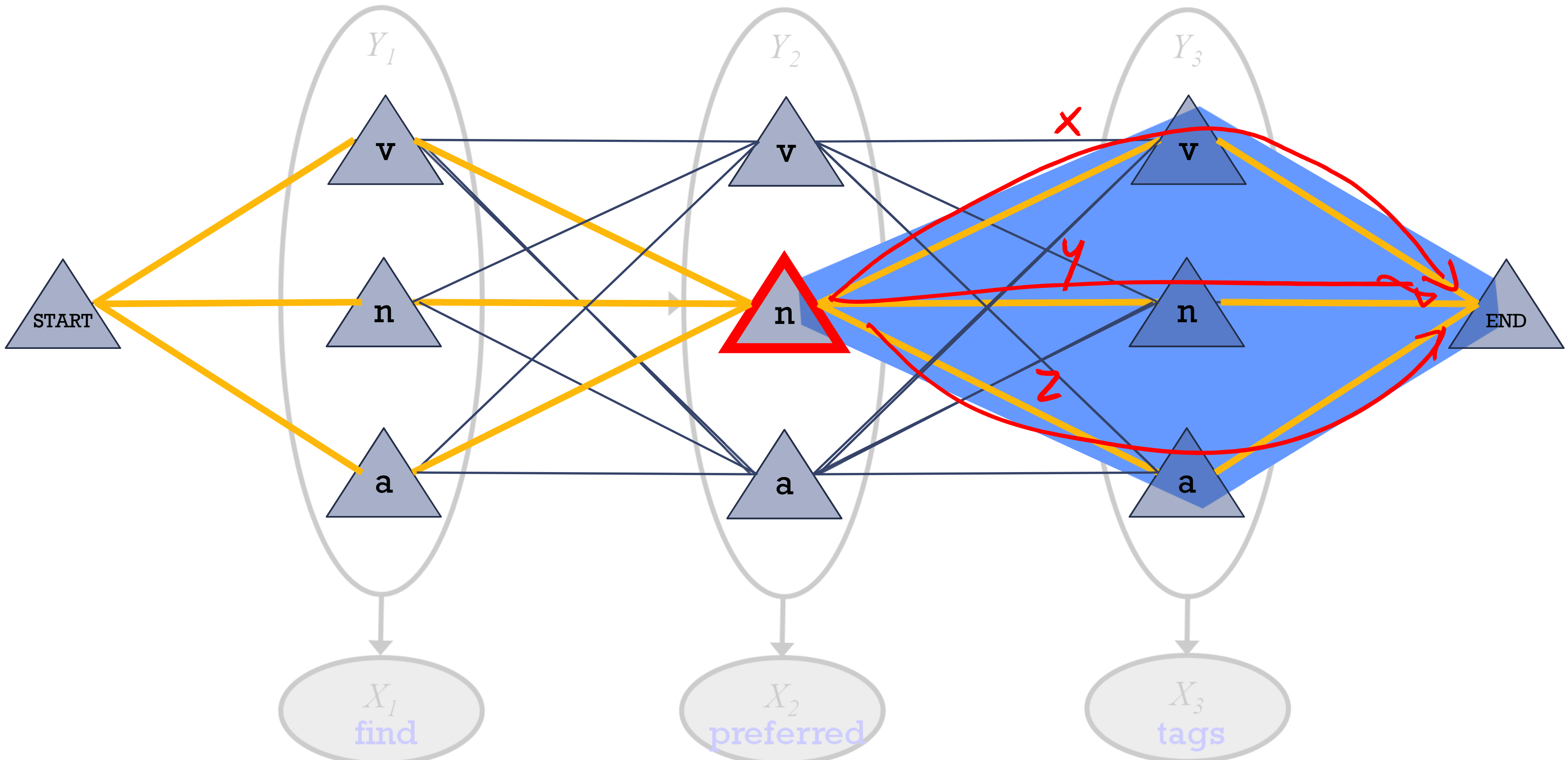
Forward-Backward Algorithm: Finds Marginals



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

(found by dynamic programming: matrix-vector products)

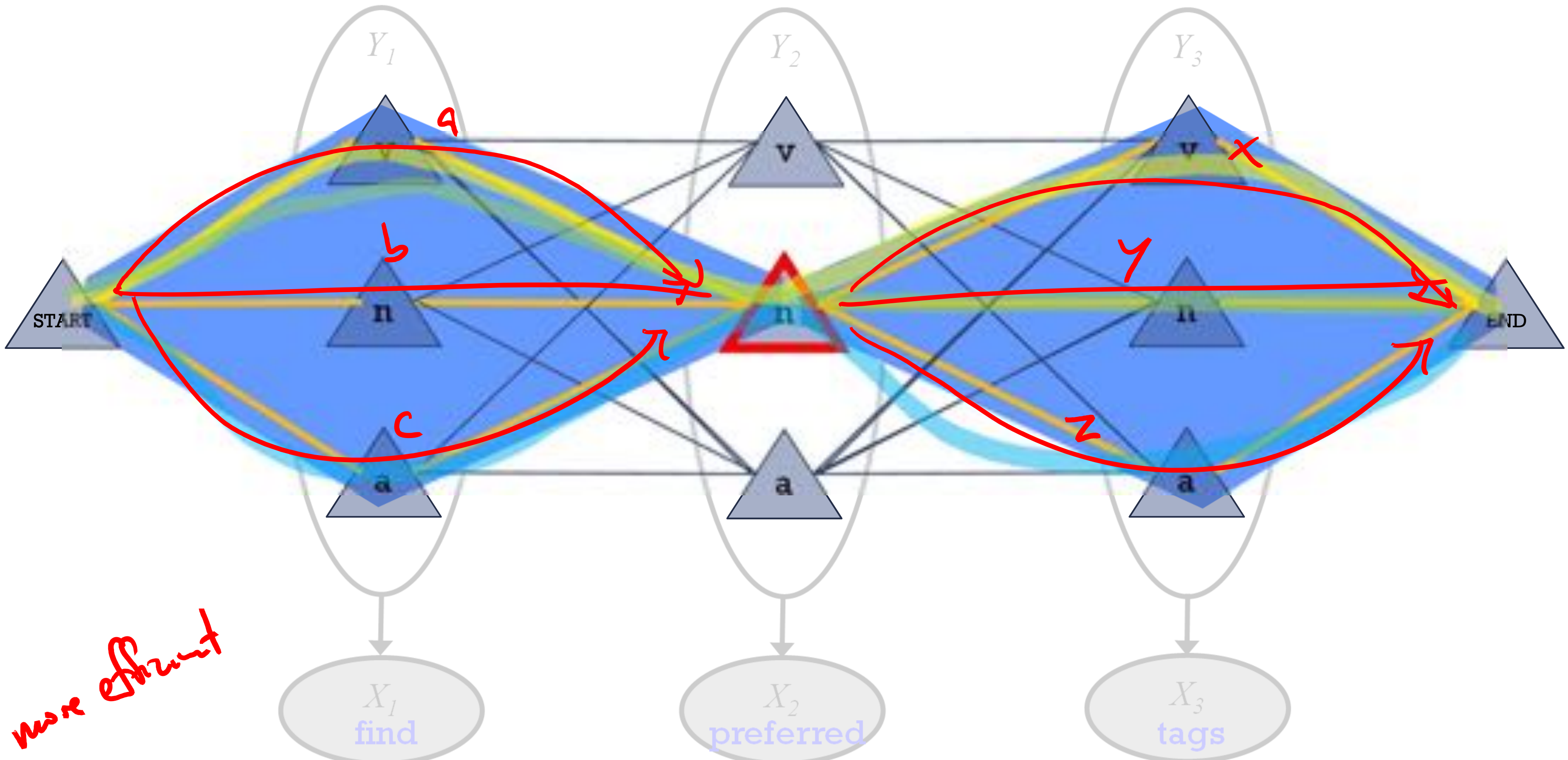
Forward-Backward Algorithm: Finds *Marginals*



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

(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)

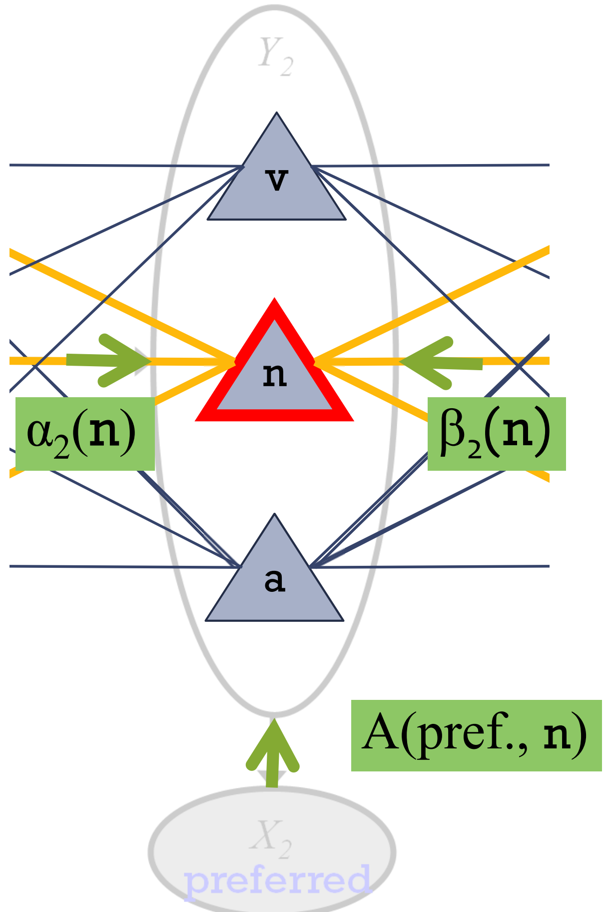
$\alpha_2(n)\beta_n(n) = (a+b+c)(x+y+z)$

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.

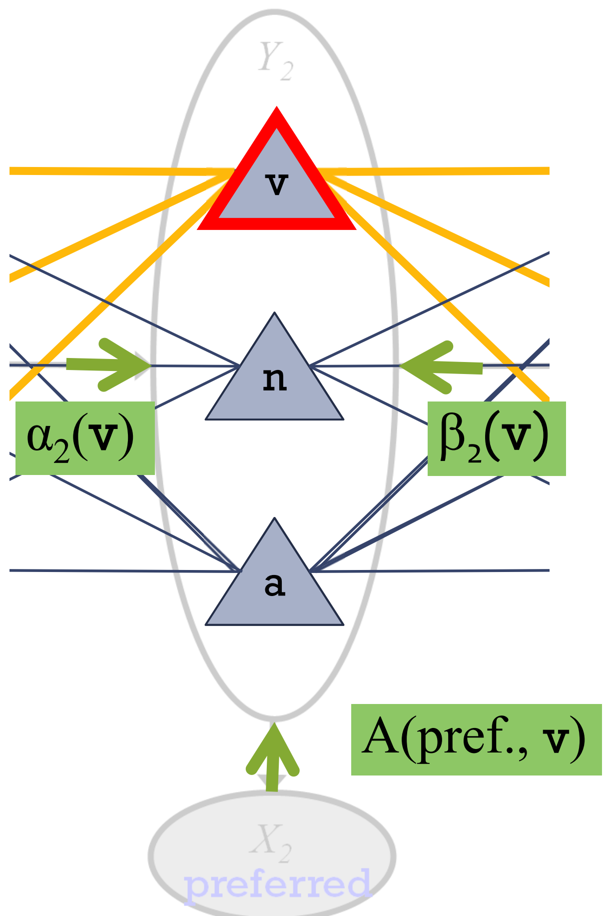


"belief that $Y_2 = \mathbf{n}$"

total weight of *all* paths through n

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

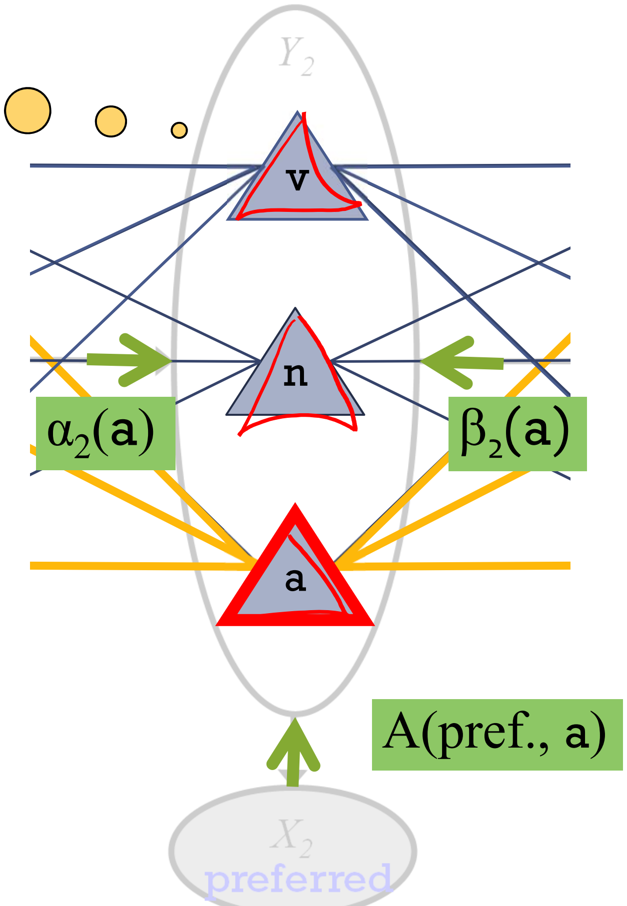
= $\alpha_2(\mathbf{v})$ A(pref., **v**) $\beta_2(\mathbf{v})$

Forward-Backward Algorithm: Finds Marginals

v	0.1
n	0
a	0.4

divide by Z=0.5 to get marginal probs

v	0.2
n	0
a	0.8



"belief that Y_2 = **v**"

"belief that Y_2 = **n**"

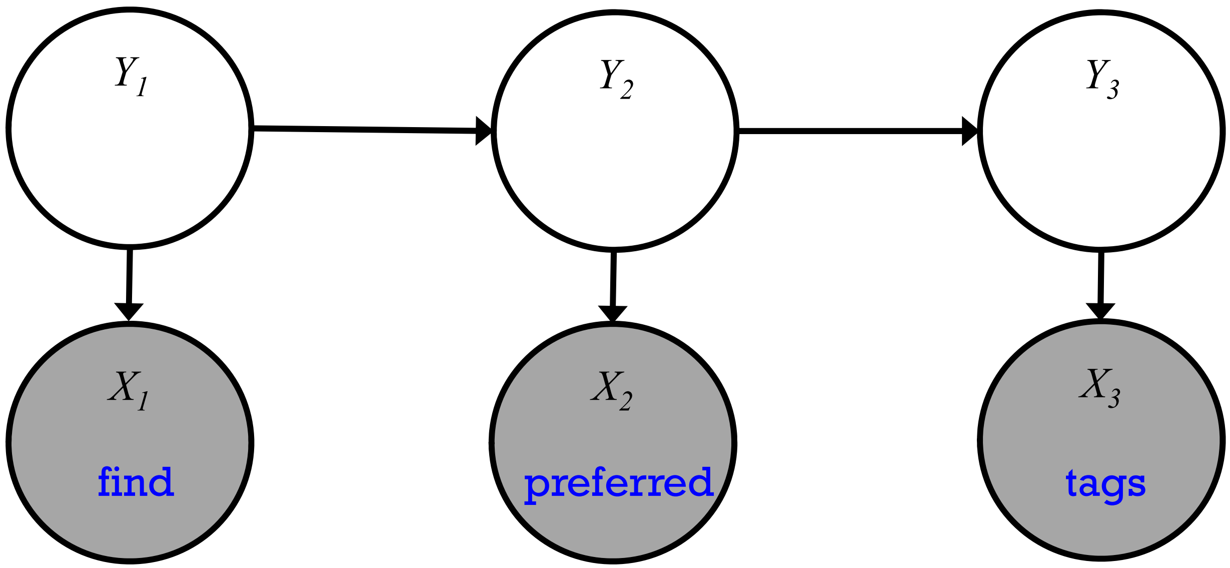
"belief that Y_2 = **a**"

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

total weight of *all* paths through a

= α_2(a) A(pref., a) β_2(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

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

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

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

forward algo

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

backward algo

③ For $t = T, \ldots, 1$:

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

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

Eval. → ④ Compute $p(\vec{x}) = \alpha_{T+1}(END)$ [Evaluation]

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

Derivation of Forward Algorithm

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

Derivation:

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

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

Viterbi Algorithm

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

"backpointers" → $b_t(k) \triangleq \underset{y_1,\ldots,y_{t-1}}{\operatorname{argmax}}\ p(x_1,\ldots,x_t, y_1,\ldots,y_{t-1}, y_t = k)$

Assume $y_0 = \text{START}$

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

② For $t = 1,\ldots,T$:

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

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

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

③ Compute Most Probable Assignment [Decoding]

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

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

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

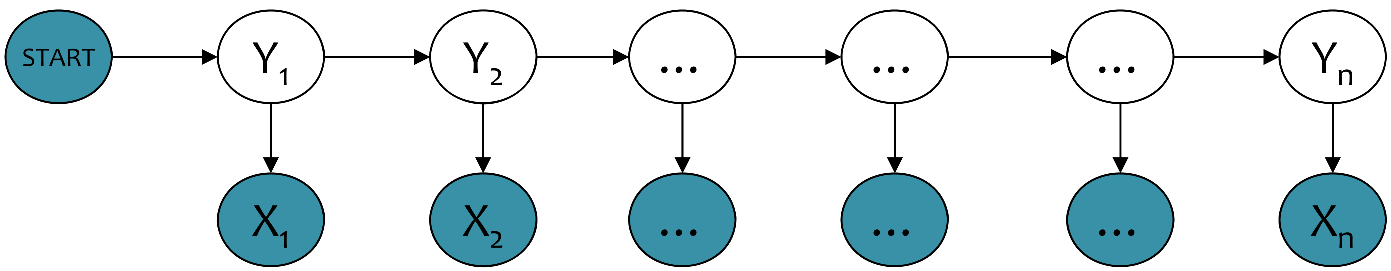
] follow the "backpointers"

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(**Y**, **X**), but in a prediction task, we need the conditional probability P(**Y**|**X**)

MBR DECODING

Inference for HMMs

– ~~Three~~ Four 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 $l(\mathbf{y}', \mathbf{y})$ and are asked for a single tagging
- How should we choose just one from our probability distribution $p(\mathbf{y}|\mathbf{x})$?
- A minimum Bayes risk (MBR) decoder $h(\mathbf{x})$ 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 *1* only if the two assignments are identical and *0* 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

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

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:

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

This decomposes across variables and requires the variable marginals.

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