Machine Learning Department
School of Computer Science
Carnegie Mellon University

Hidden Markov Models

+

Midterm Exam 2 Review

Matt Gormley
Lecture 20
Mar. 29, 2019

Reminders

- Homework 6: Learning Theory / Generative Models
  - Out: Fri, Mar 22
  - Due: Fri, Mar 29 at 11:59pm (1 week)
- Midterm Exam 2
  - Thu, Apr 4 – evening exam, details announced on Piazza
- Homework 7: HMMs
  - Out: Fri, Mar 29
  - Due: Wed, Apr 10 at 11:59pm

- Today's In-Class Poll
  - http://p20.mlcourse.org

THE FORWARD-BACKWARD ALGORITHM

Inference for HMMs

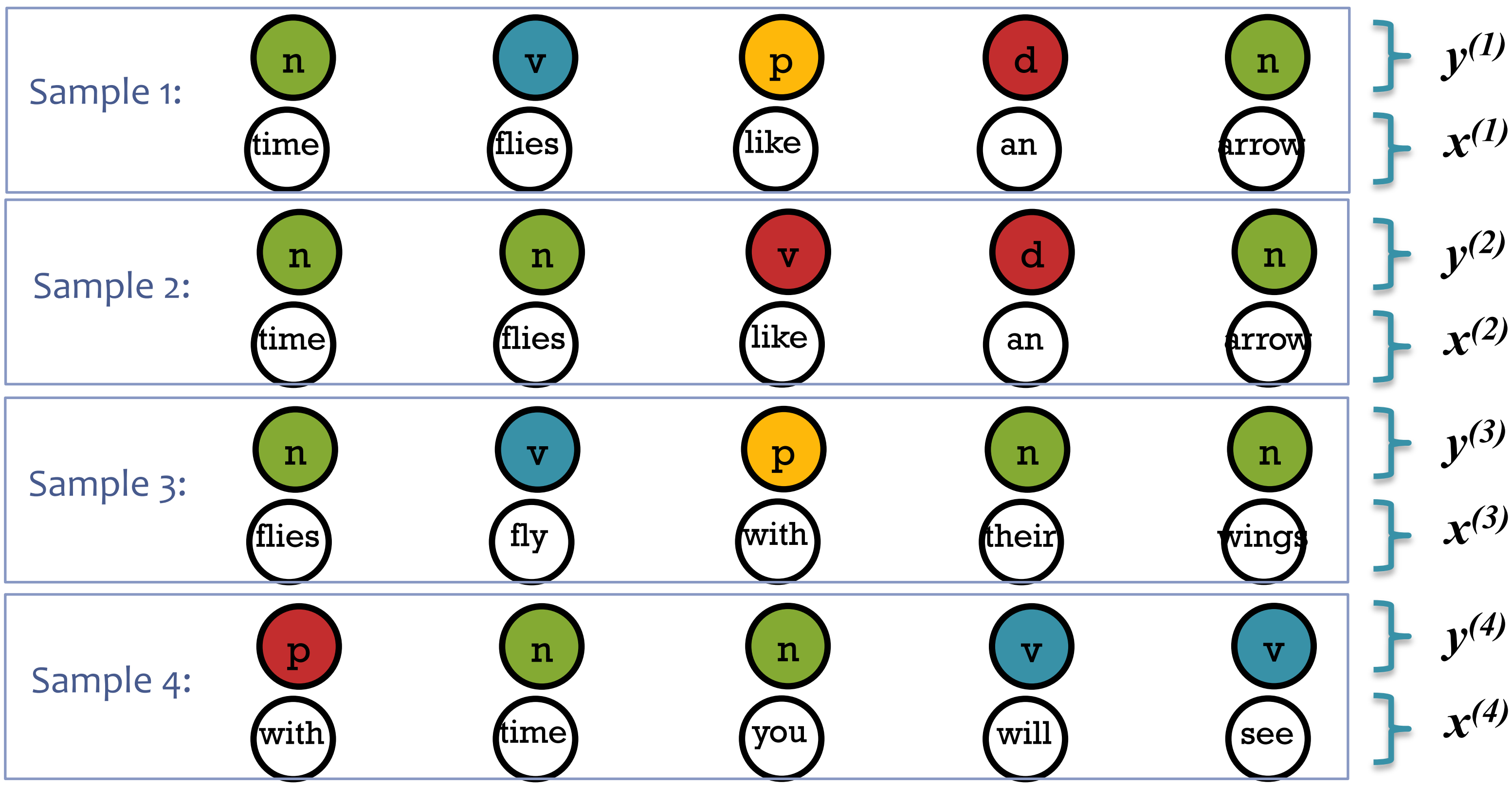
Whiteboard

– Three Inference Problems for an HMM

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

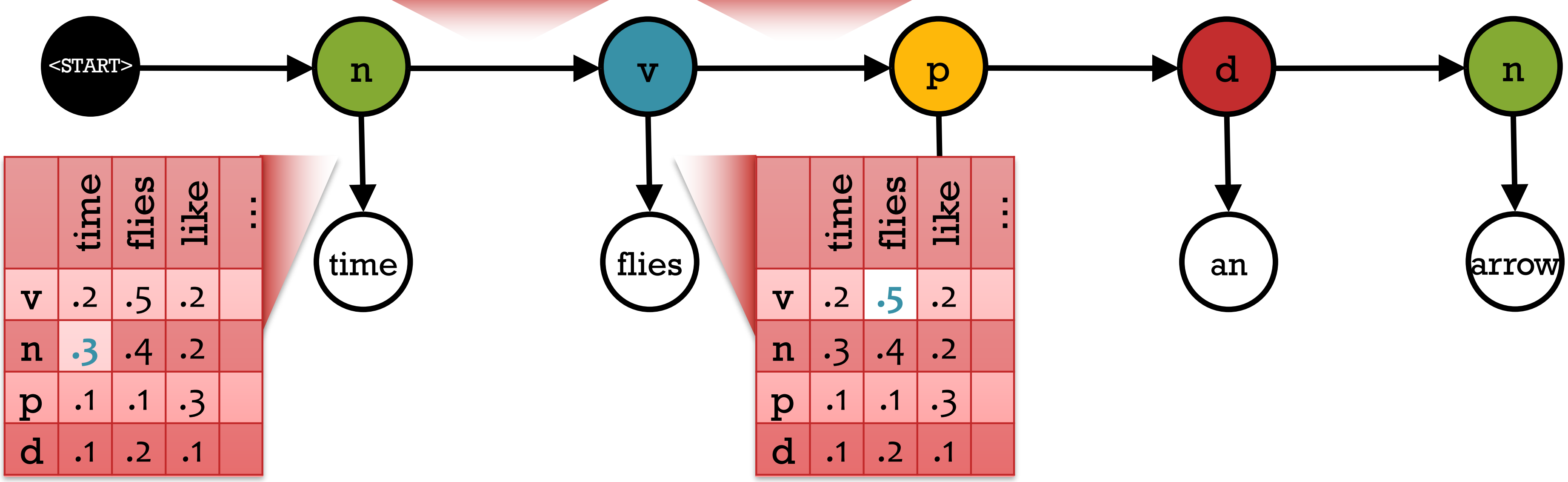
Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$



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.

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



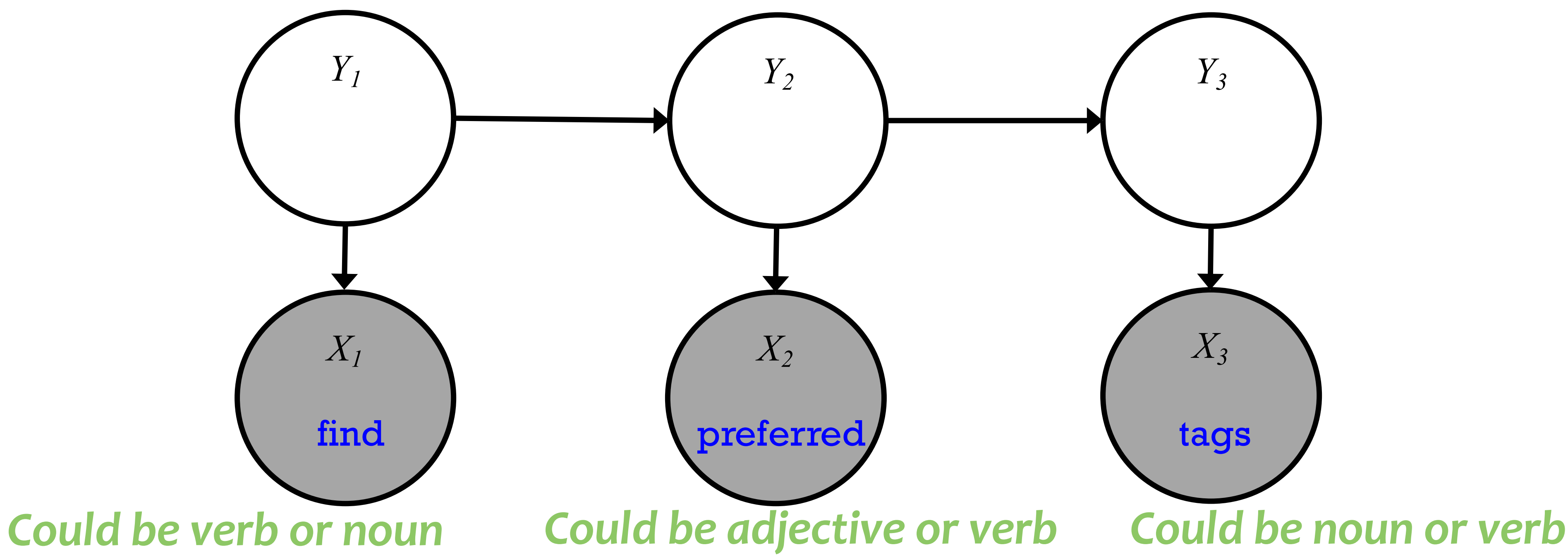
	v	n	p	d
v	.1	.4	.2	.3
n	.8	.1	.1	0
p	.2	.3	.2	.3
d	.2	.8	0	0

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

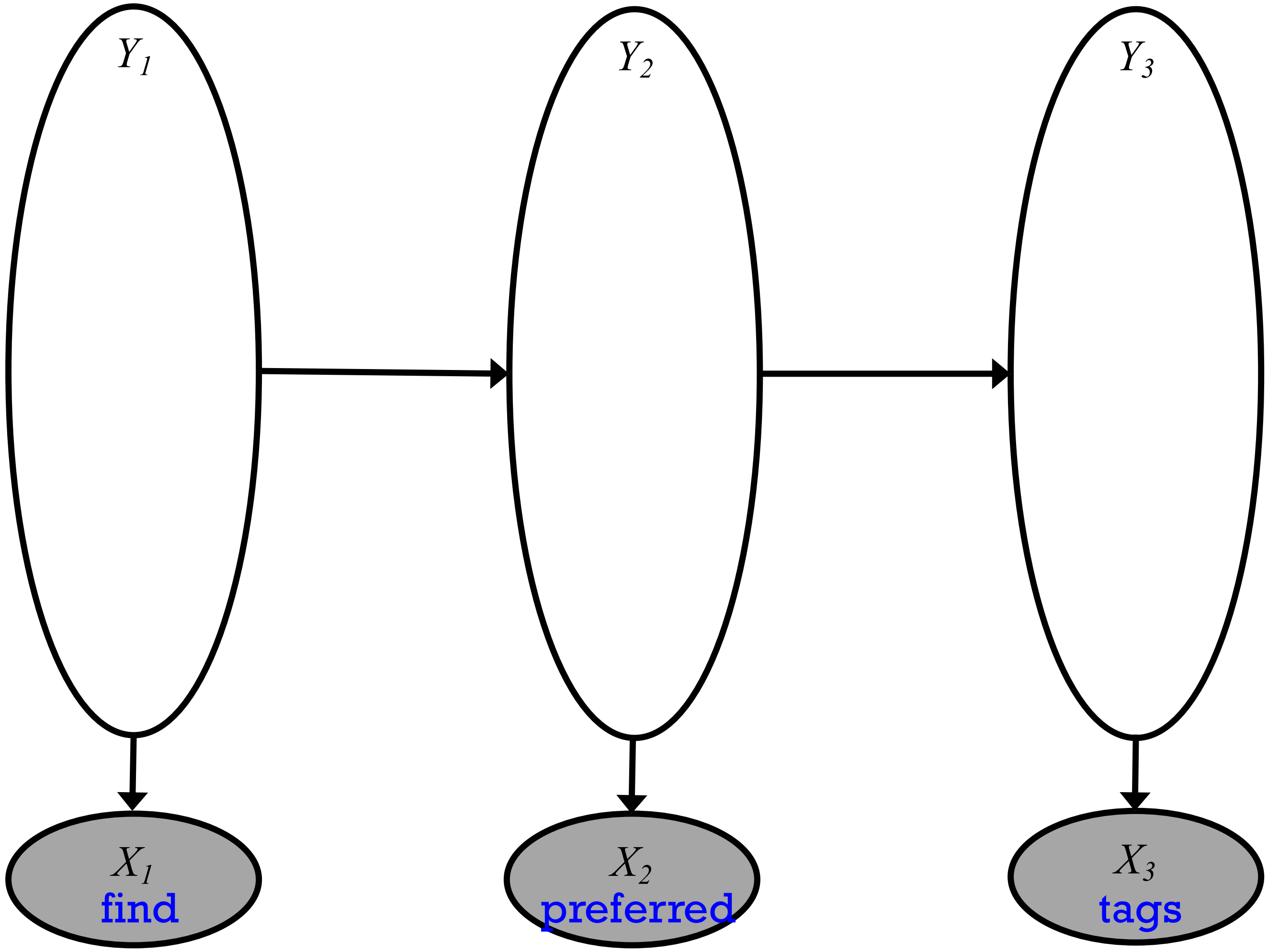
	time	flies	like	…
v	.2	.5	.2	
n	.3	.4	.2	
p	.1	.1	.3	
d	.1	.2	.1	

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

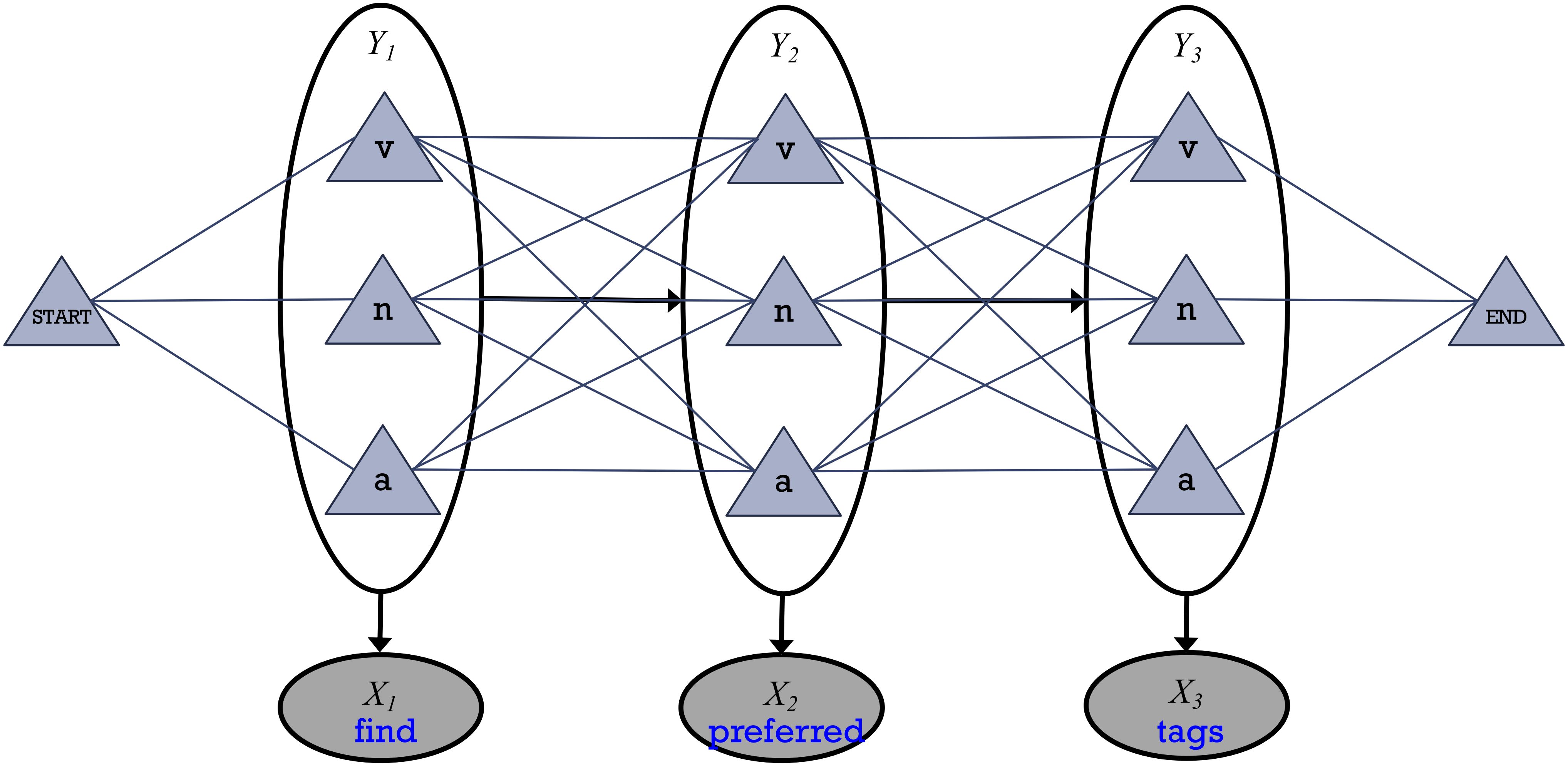
Forward-Backward Algorithm



Forward-Backward Algorithm

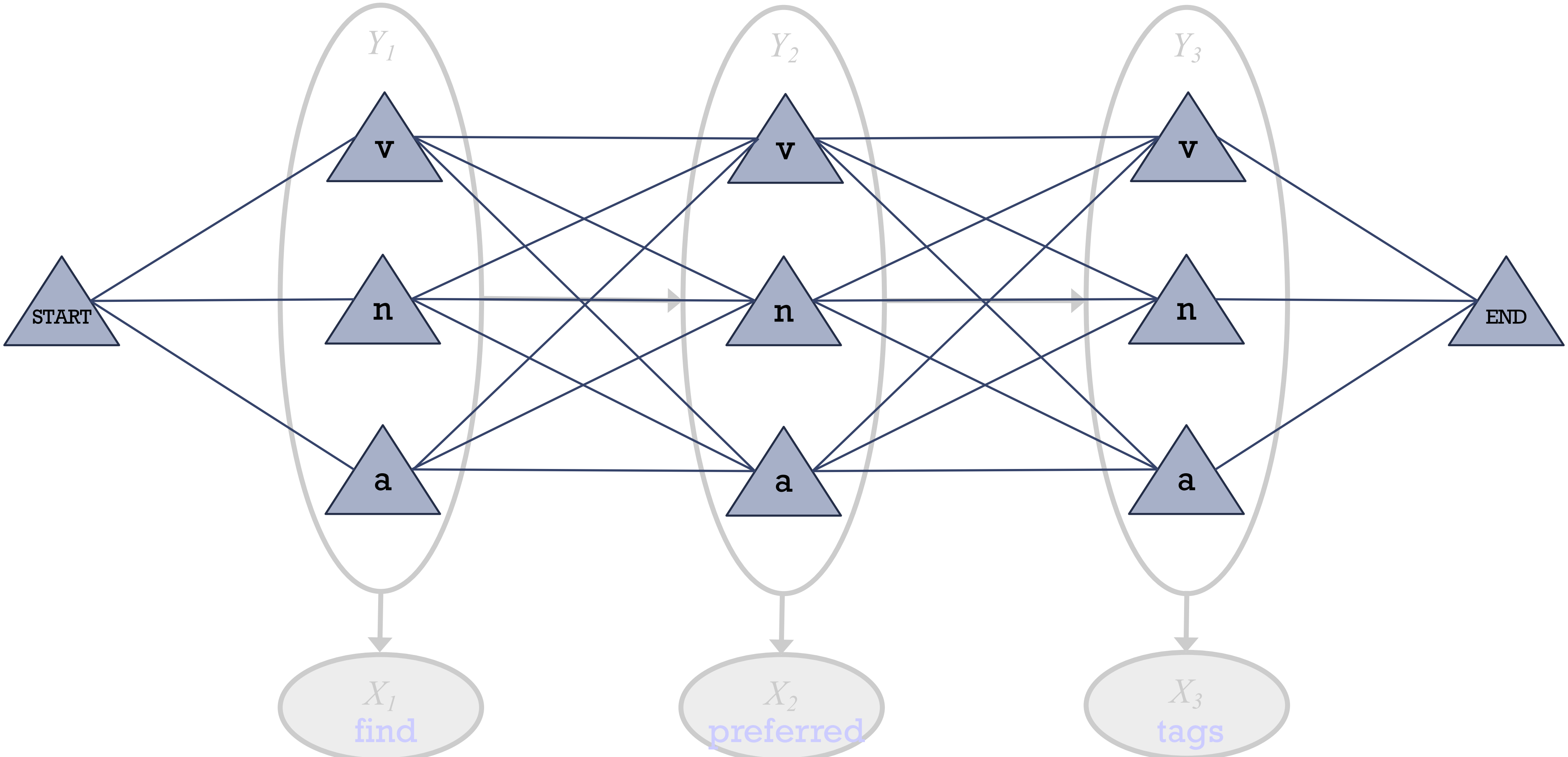


Forward-Backward Algorithm



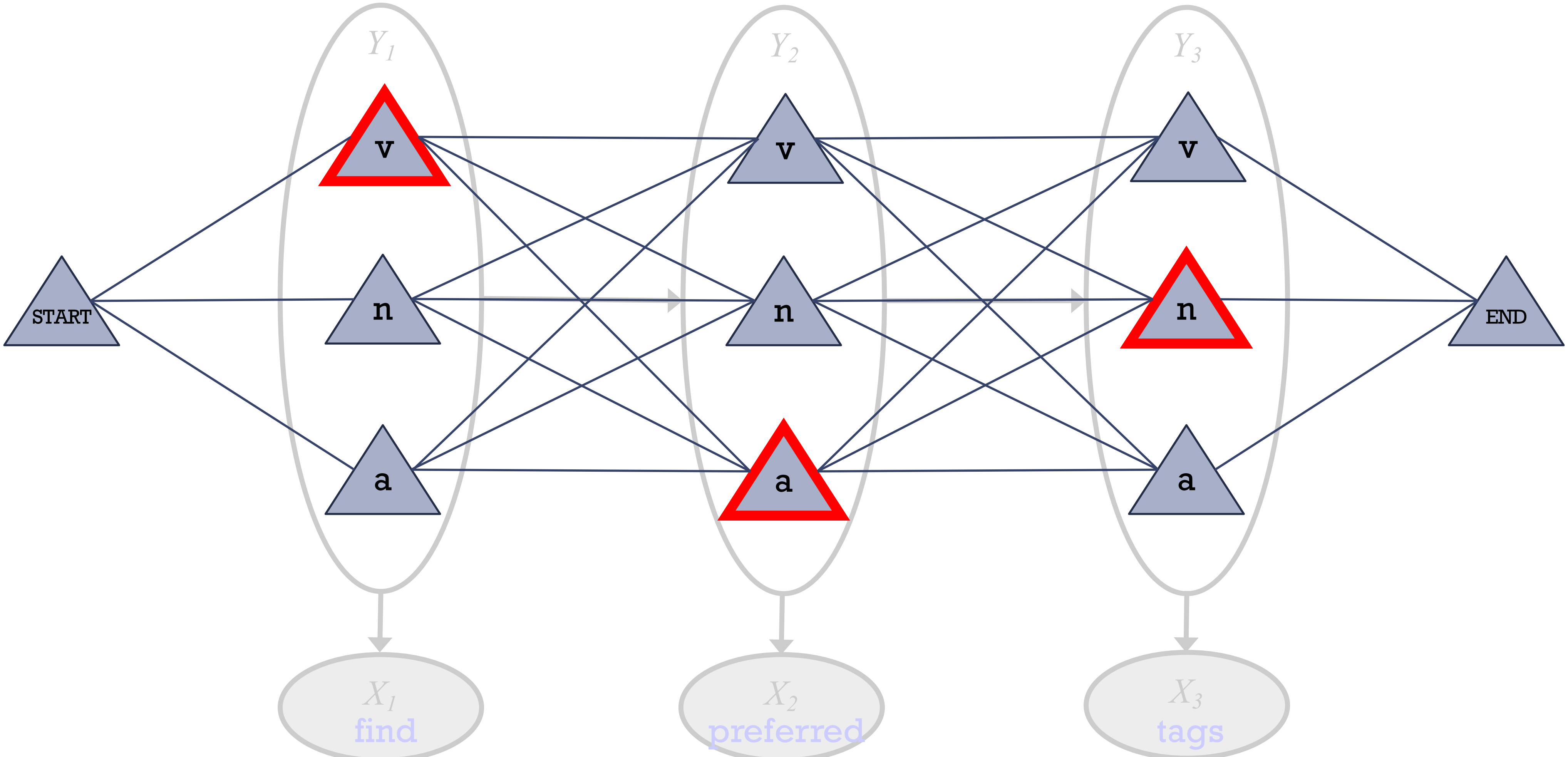
- Let's show the possible *values* for each variable

Forward-Backward Algorithm



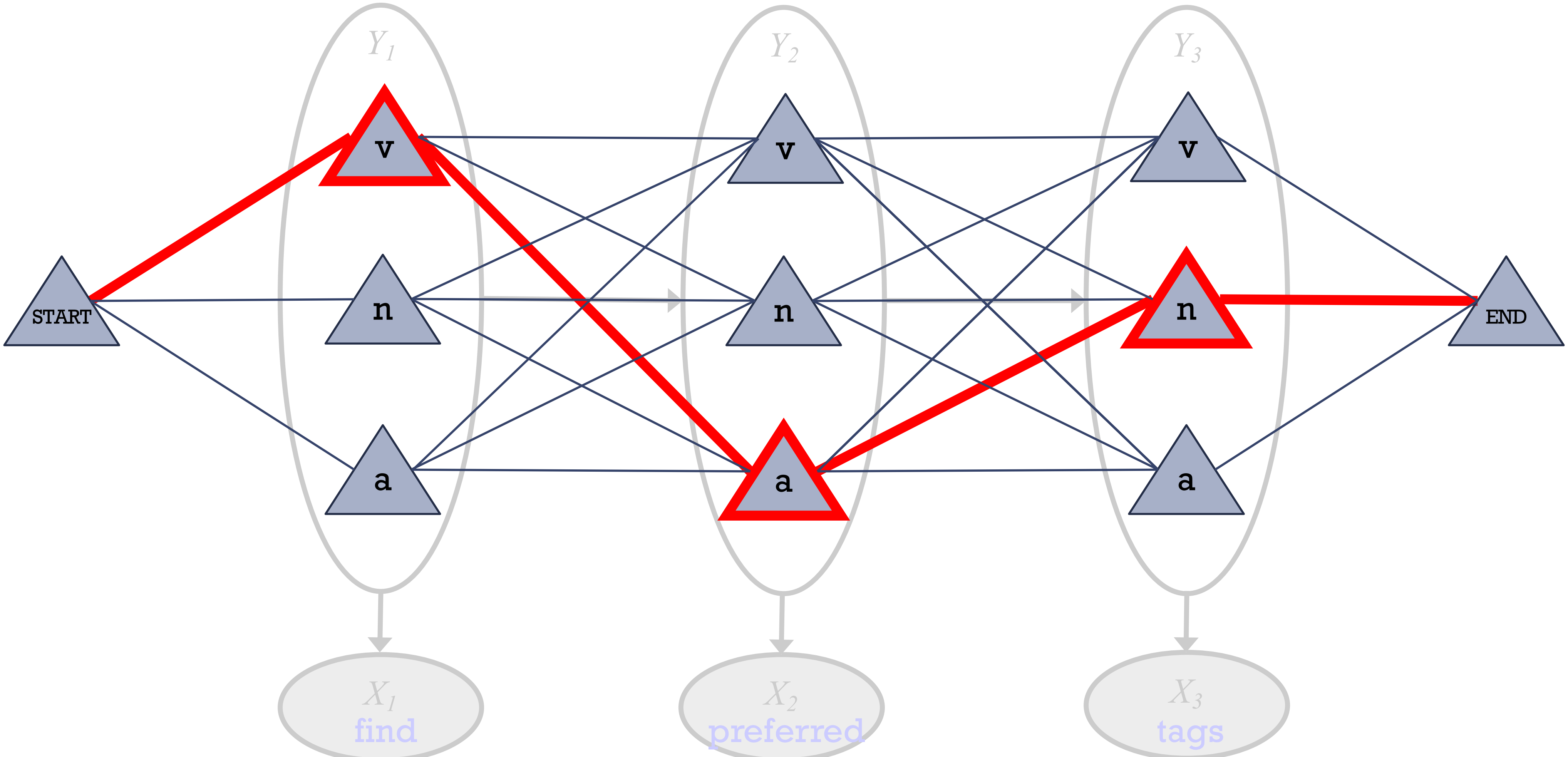
- Let's show the possible *values* for each variable

Forward-Backward Algorithm



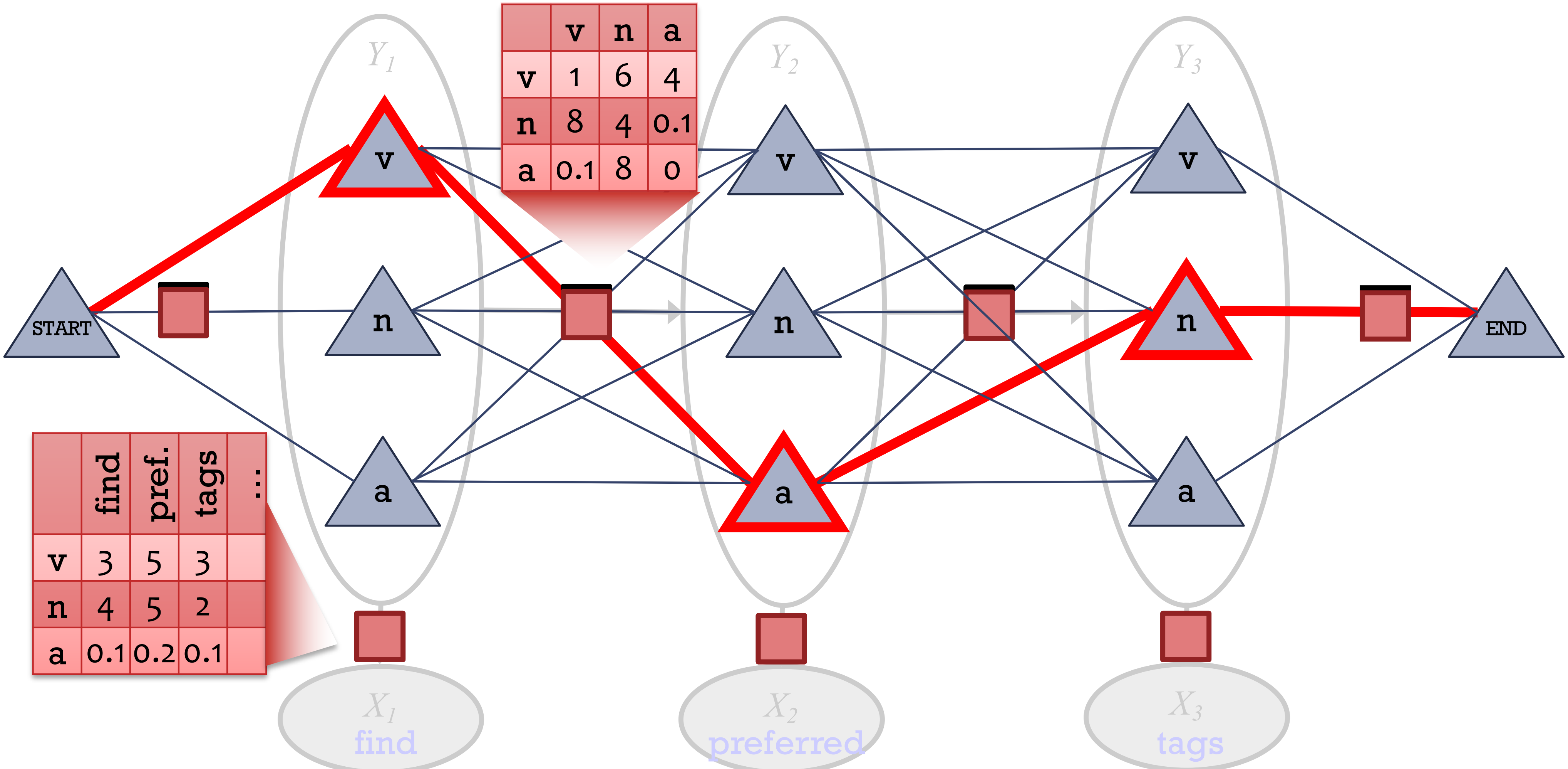
- Let's show the possible *values* for each variable
- One possible assignment

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

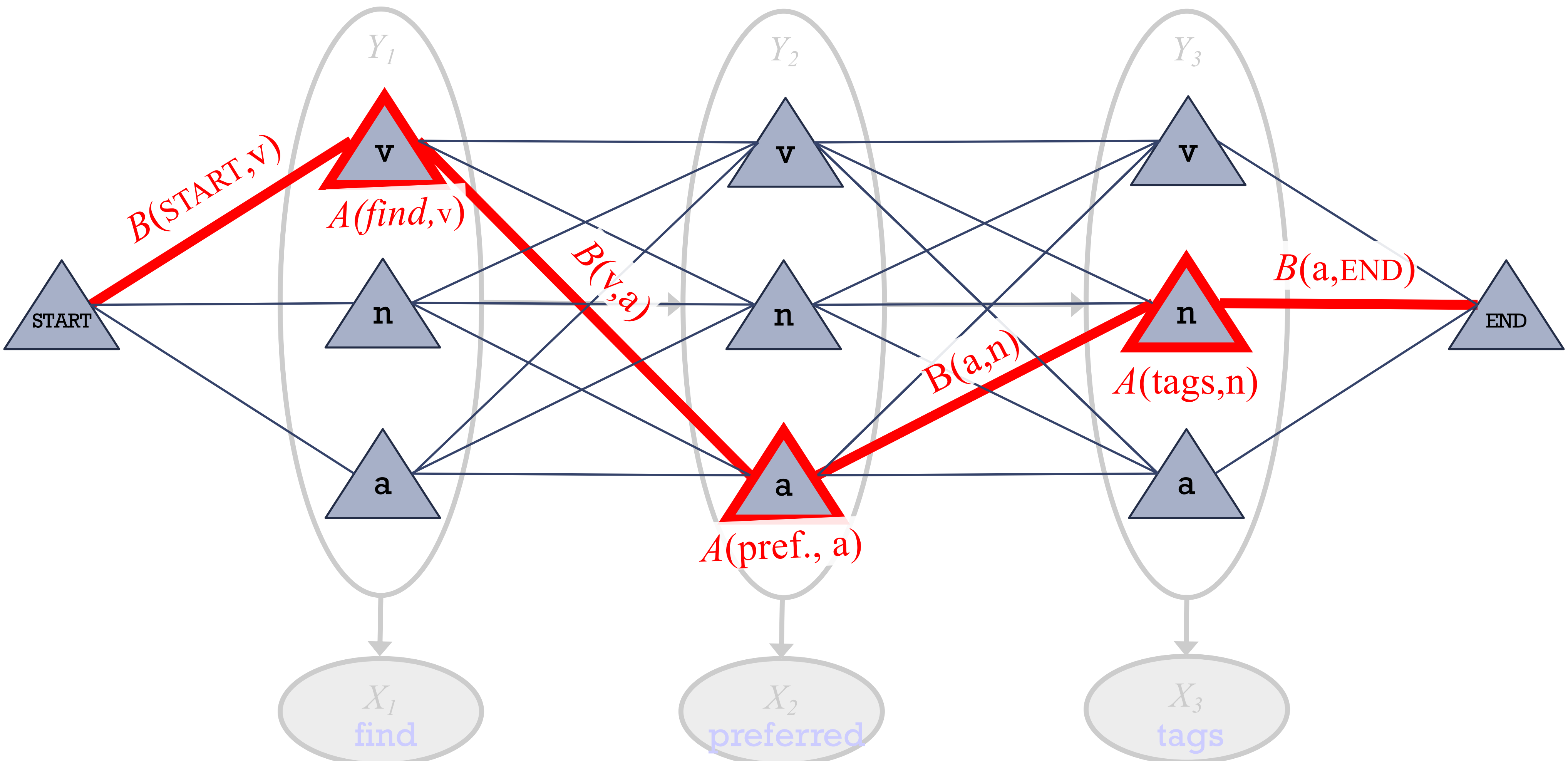


	v	n	a
v	1	6	4
n	8	4	0.1
a	0.1	8	0

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

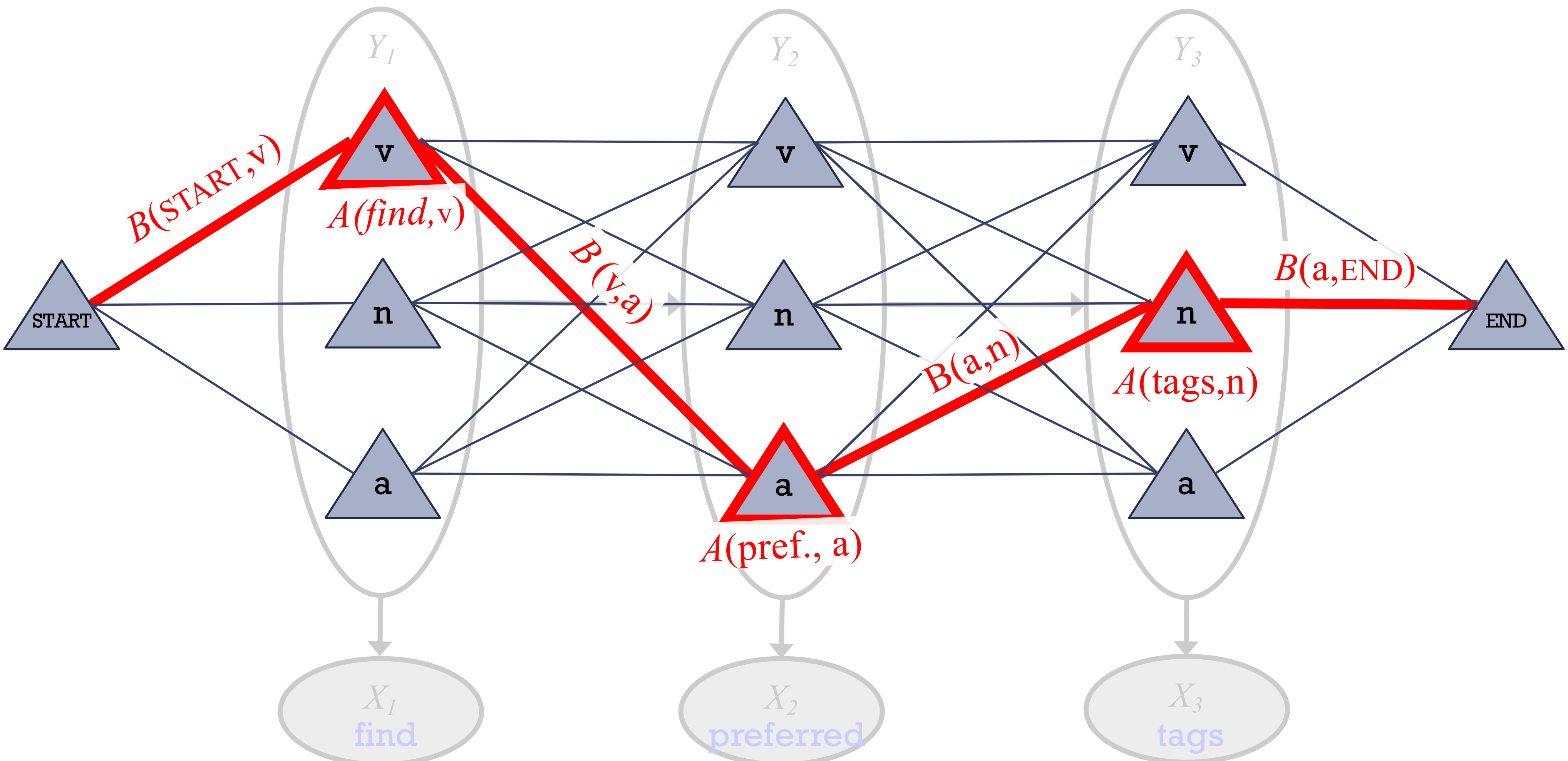
- Let's show the possible *values* for each variable
- One possible assignment
- And what the 7 transition / emission factors think of it …

Viterbi Algorithm: Most Probable Assignment



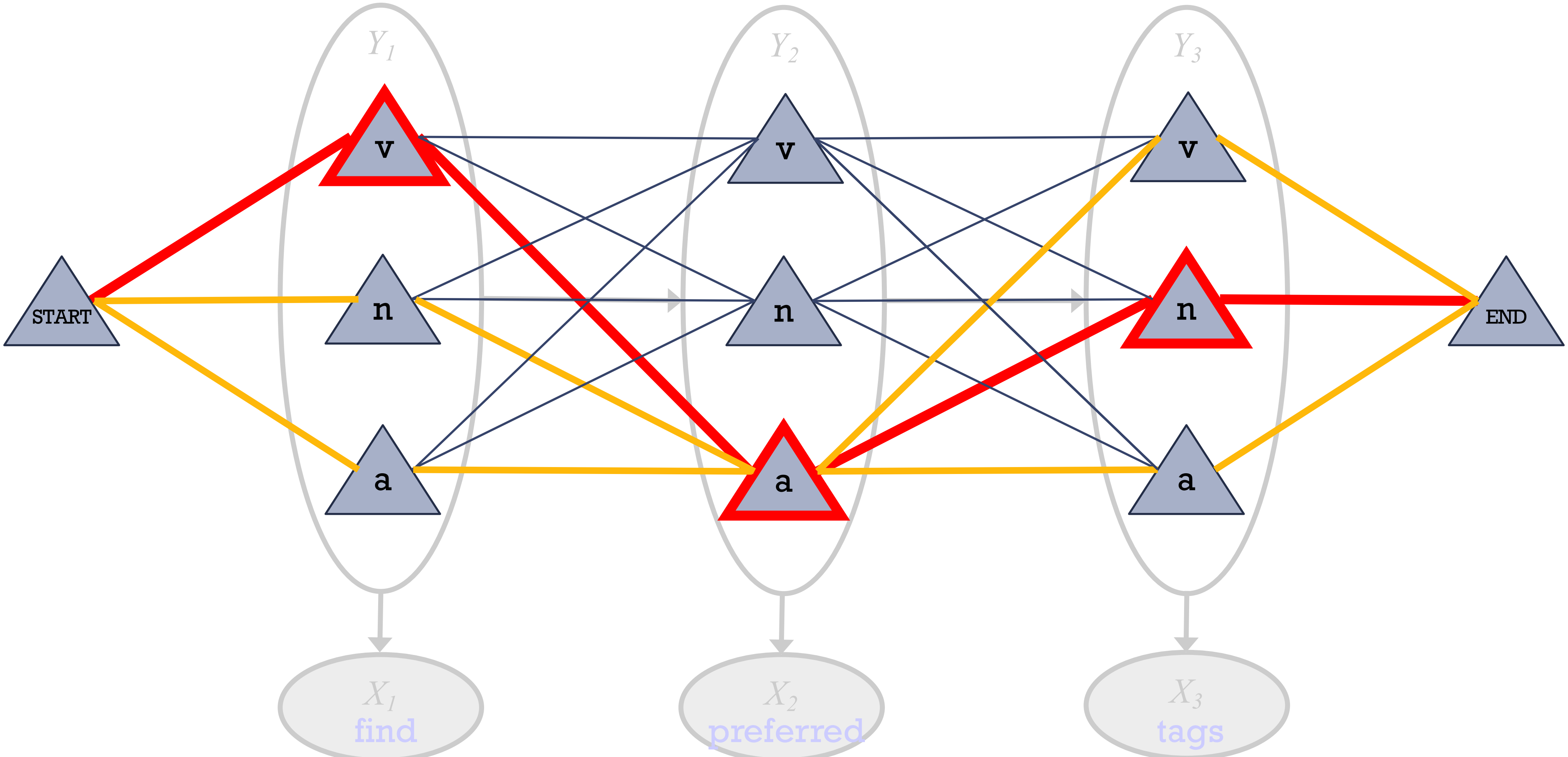
- So p(**v a n**) = (1/Z) * product of 7 numbers
- Numbers associated with edges and nodes of path
- Most probable assignment = **path with highest product**

Viterbi Algorithm: Most Probable Assignment



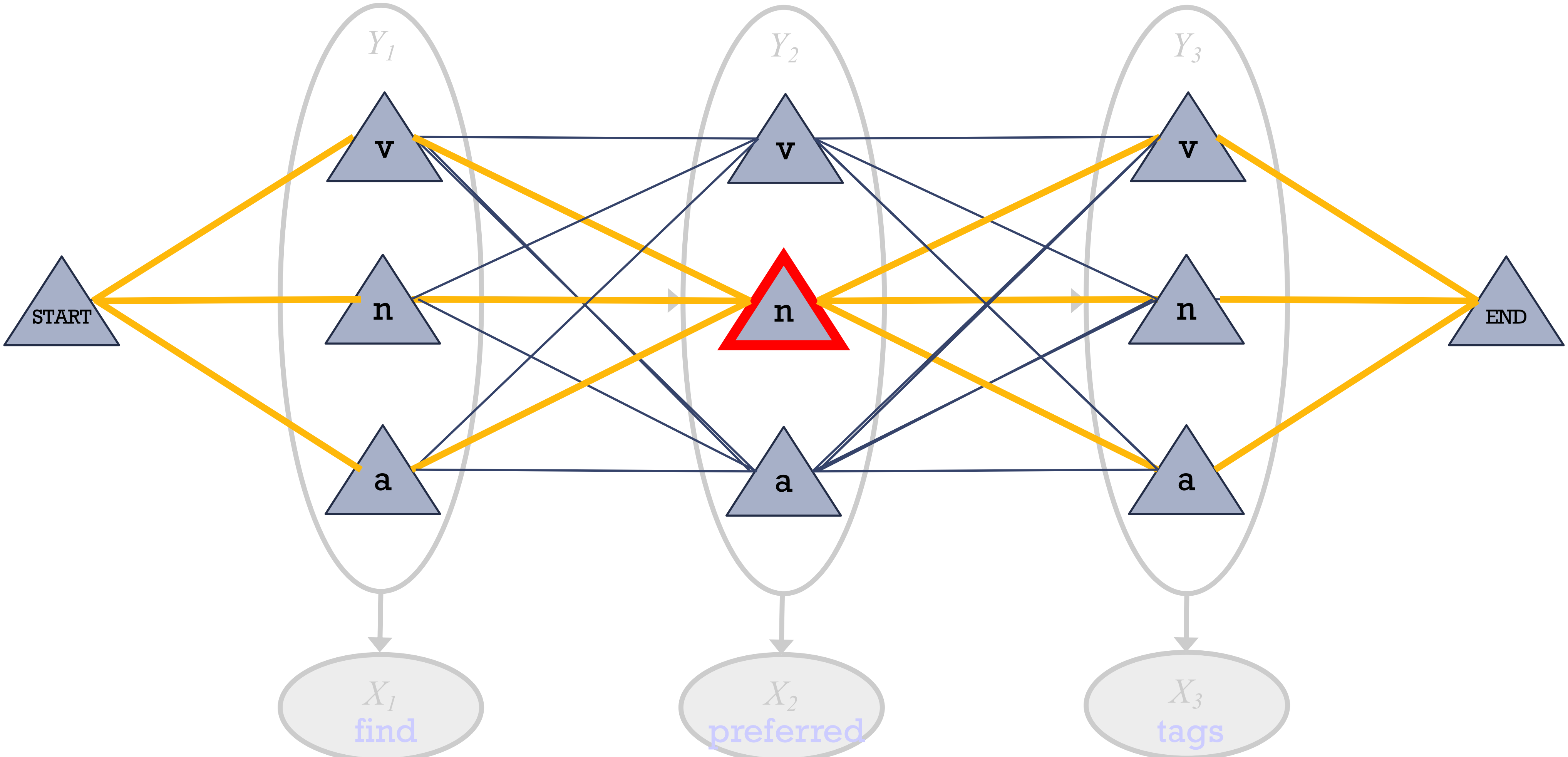
- So p(**v a n**) = (1/Z) * product weight of one path

Forward-Backward Algorithm: Finds Marginals



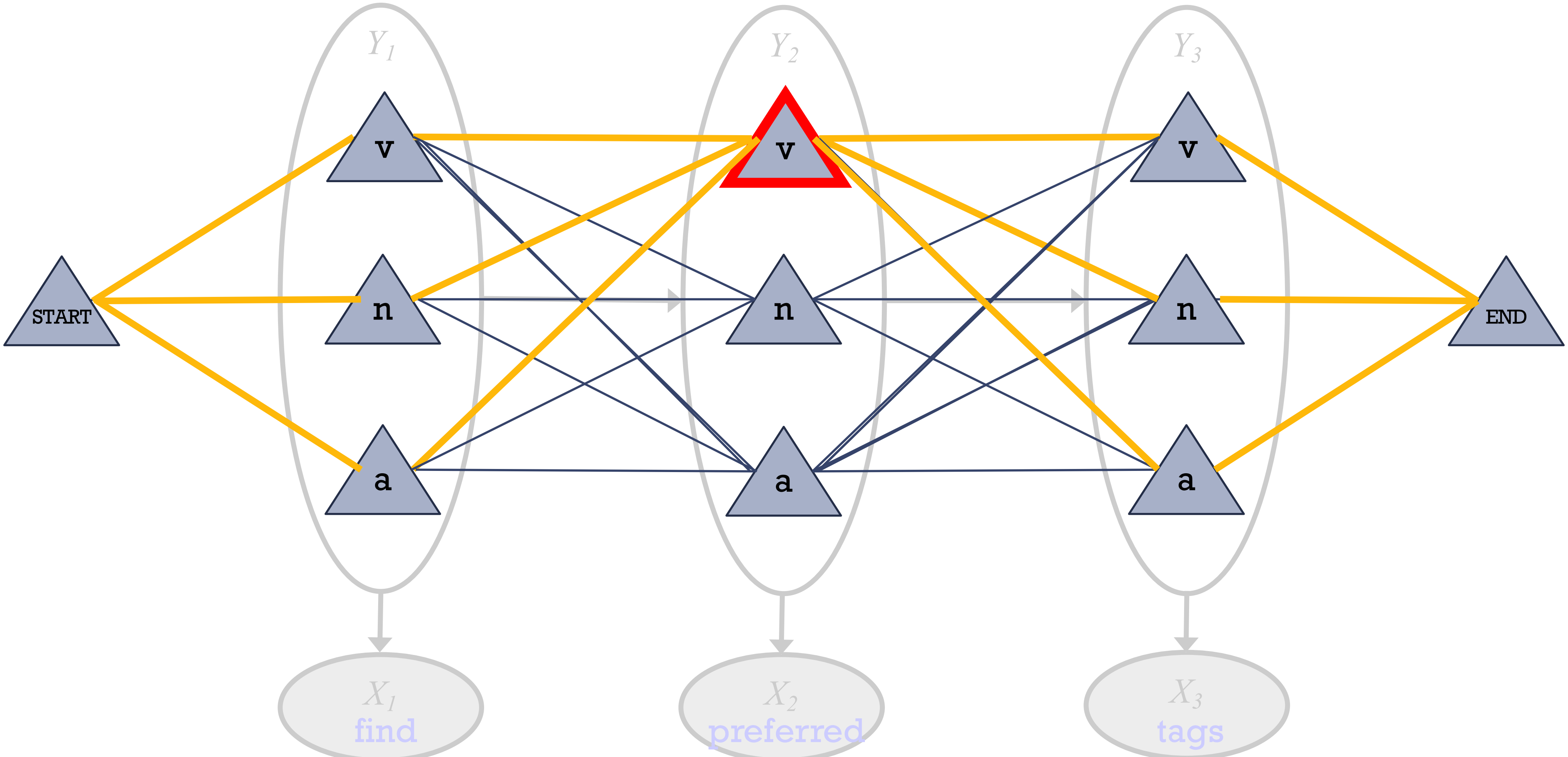
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability p(Y_2 = a)
  = (1/Z) * total weight of *all* paths through a

Forward-Backward Algorithm: Finds Marginals



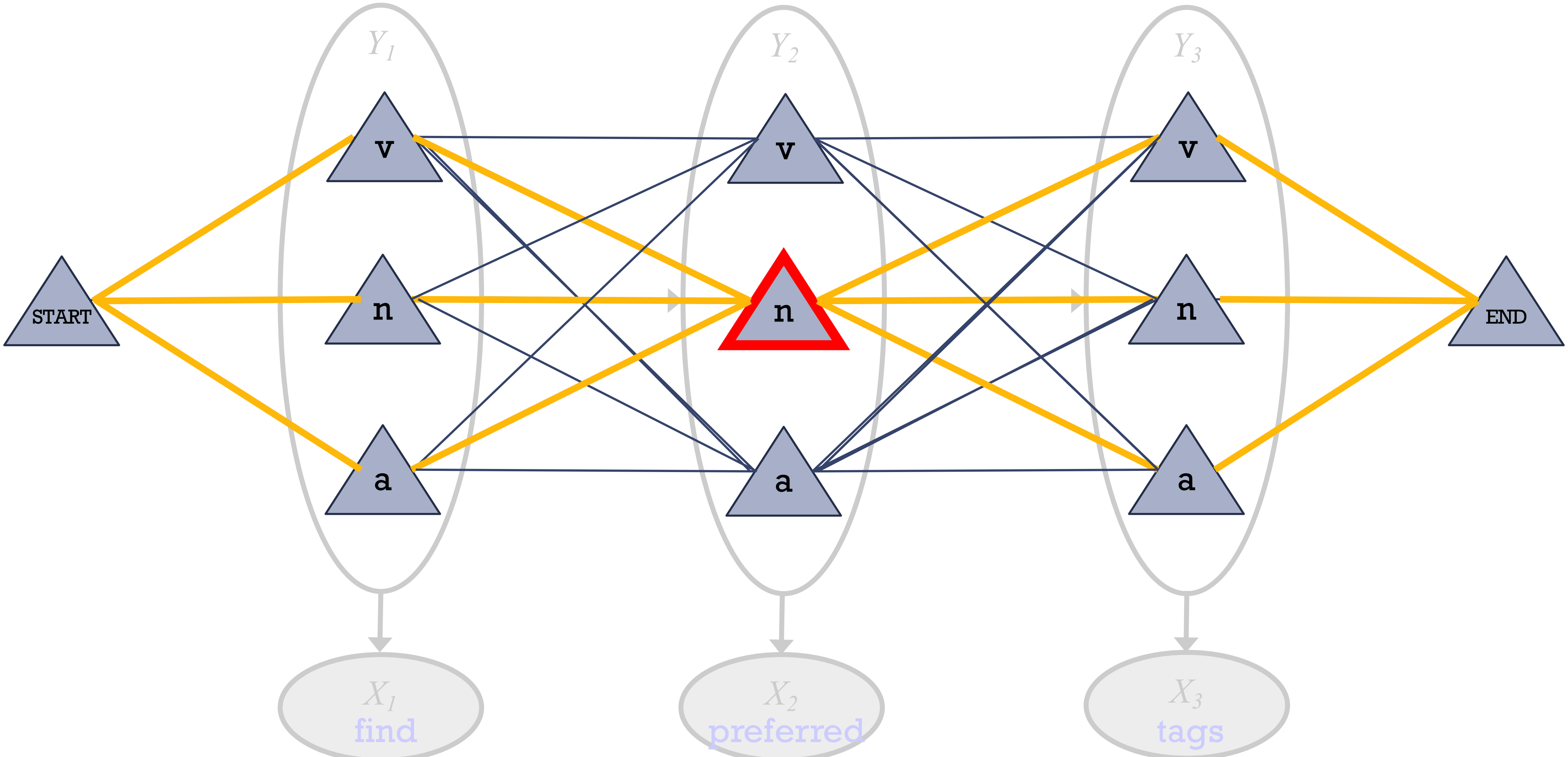
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(Y_2 = n)$
  = (1/Z) * total weight of *all* paths through n

Forward-Backward Algorithm: Finds Marginals

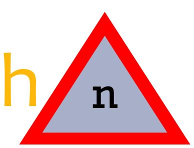


- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(Y_2 = v)$
  = (1/Z) * total weight of *all* paths through **v**

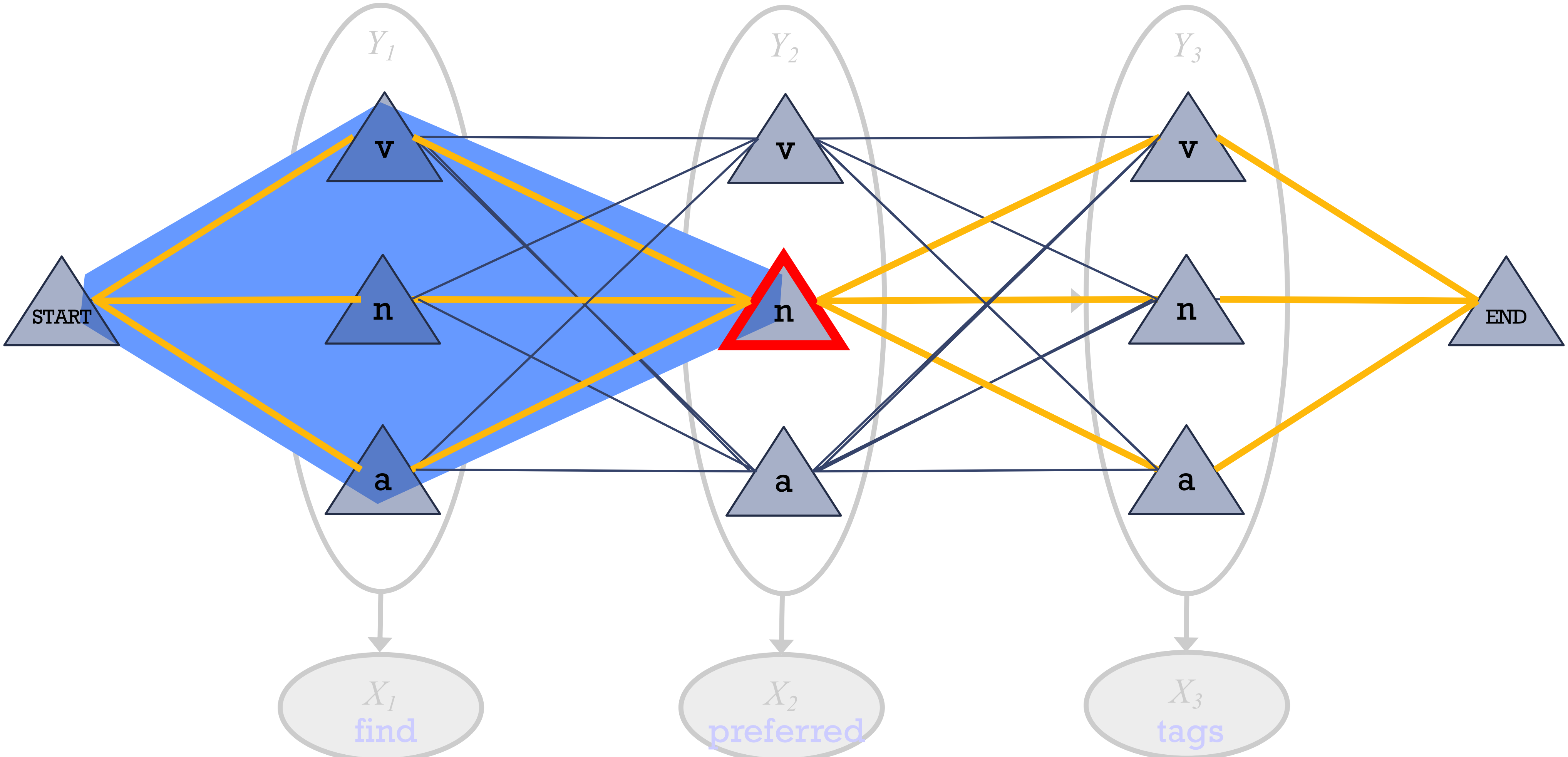
Forward-Backward Algorithm: Finds Marginals



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



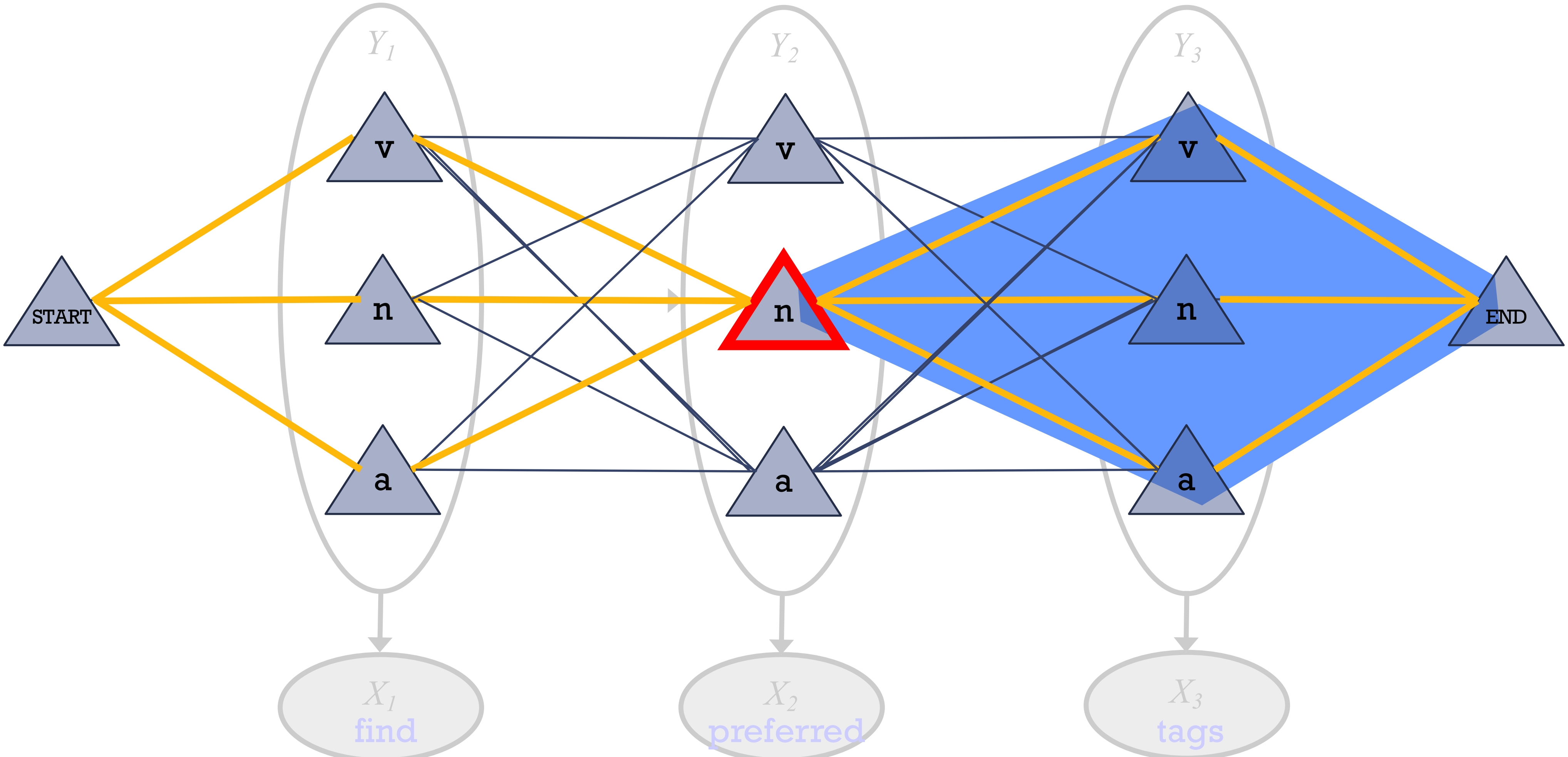
Forward-Backward Algorithm: Finds Marginals



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

(found by dynamic programming: matrix-vector products)

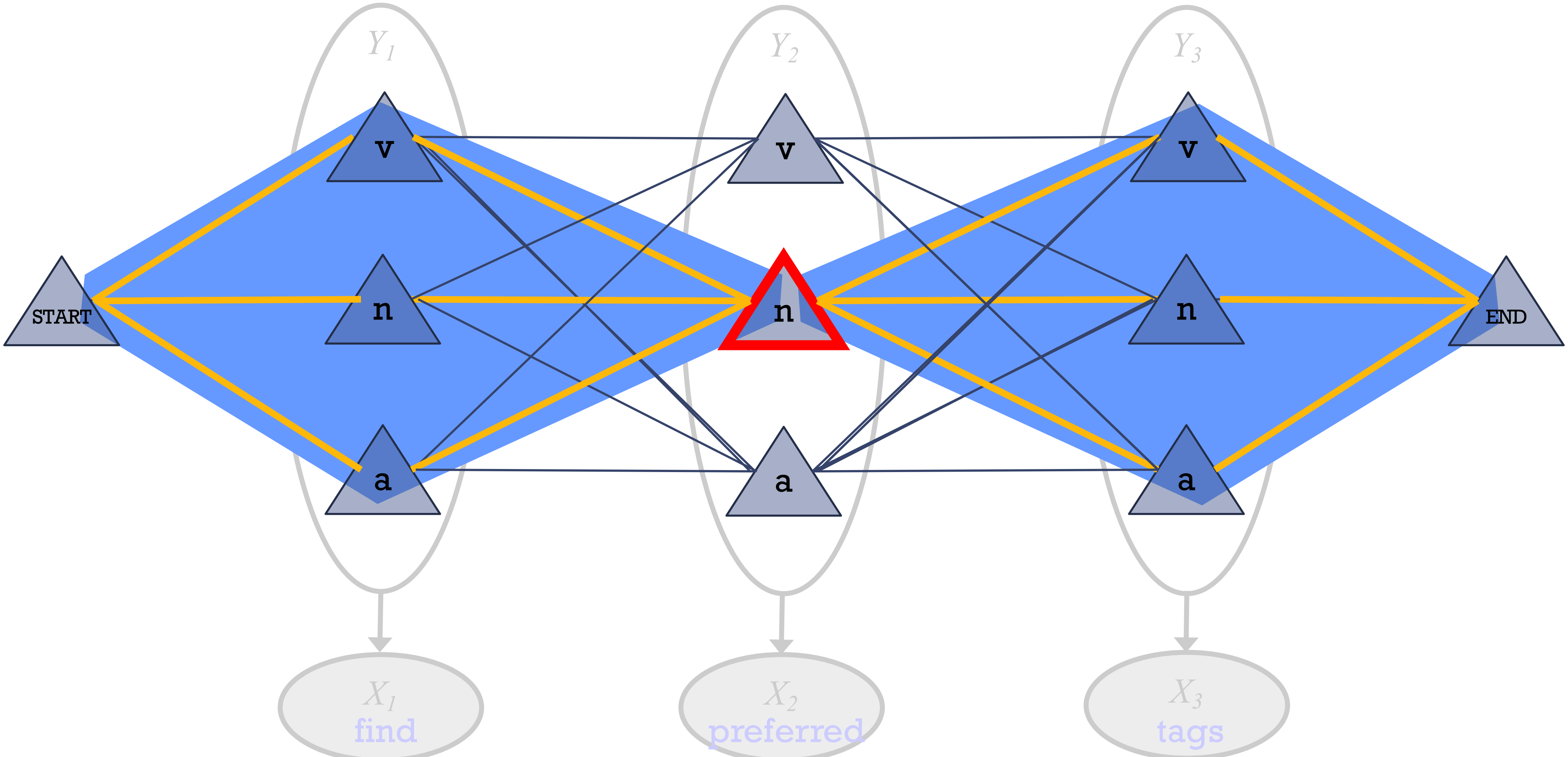
Forward-Backward Algorithm: Finds Marginals



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

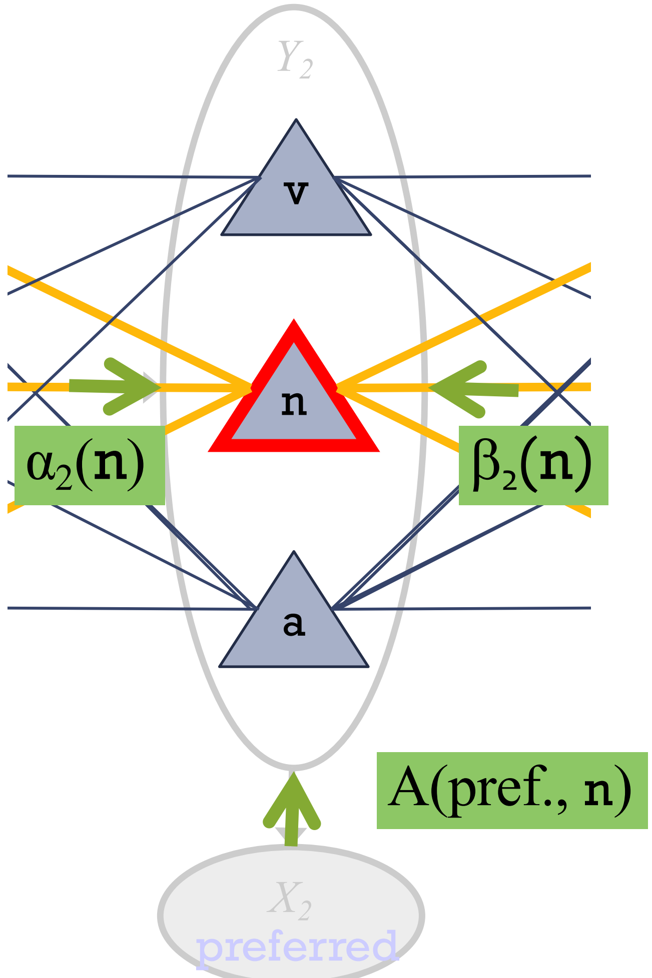
$\beta_2(\mathbf{n})$ = total weight of these path *suffixes* (x + y + z)

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.

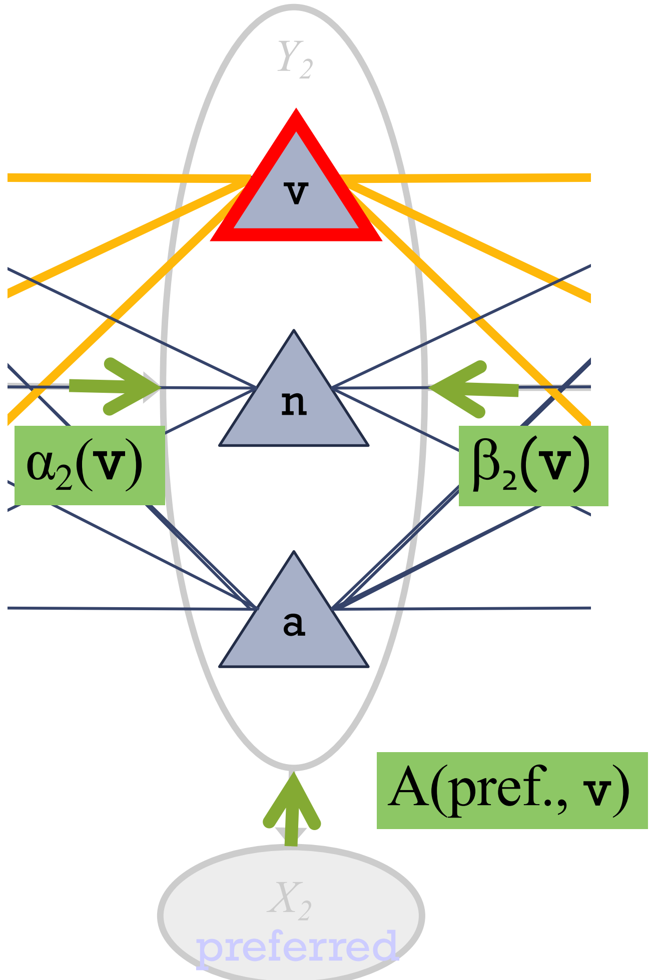


"belief that $Y_2 = \mathbf{n}$"

total weight of *all* paths through n

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

Forward-Backward Algorithm: Finds Marginals



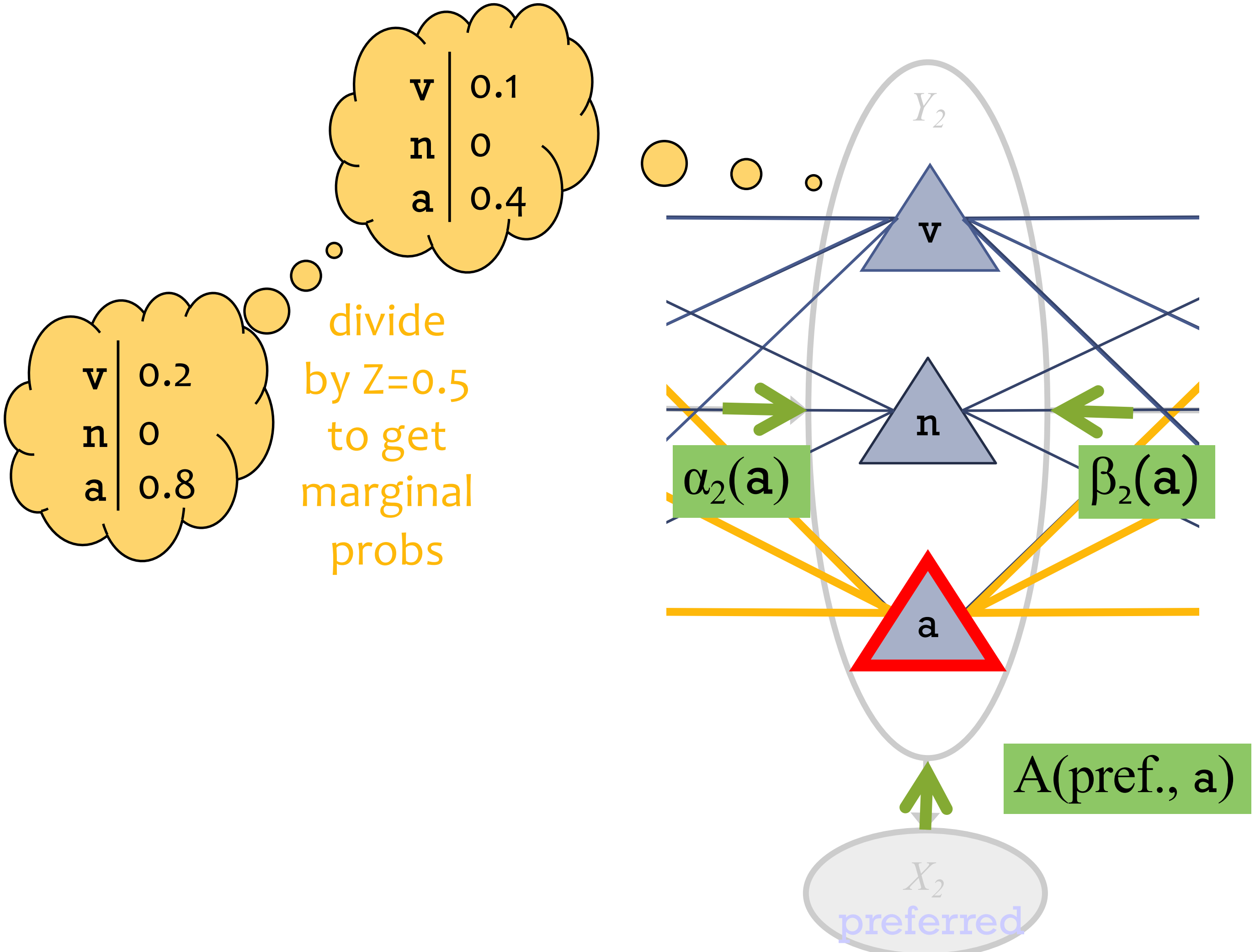
"belief that Y_2 = **v**"

"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

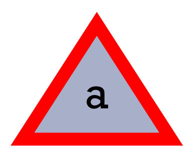


"belief that Y_2 = **v**"

"belief that Y_2 = **n**"

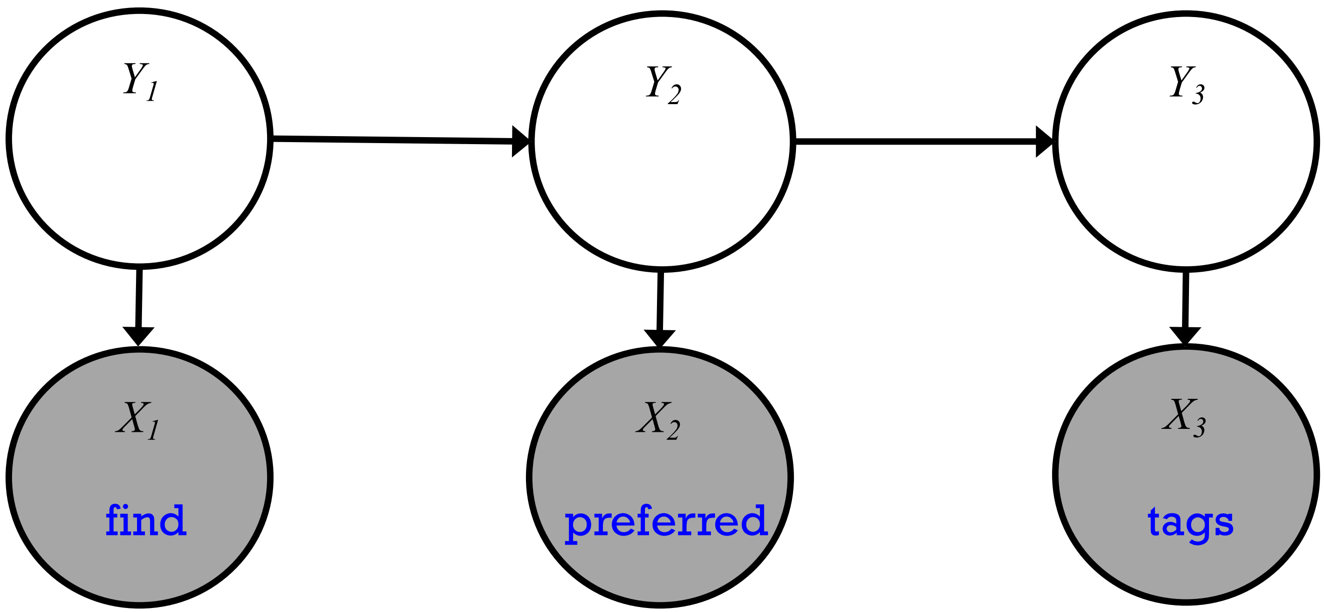
"belief that Y_2 = **a**"

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

total weight of *all* paths through 

= 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

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 "$y_T = END$"

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

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

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

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 $t = T, \ldots, 1$:

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

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

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]

Viterbi Algorithm

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

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

<u>Assume</u> $y_0 = \text{START}$

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

② For $t = 1, \ldots, T$:

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

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

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

③ Compute Most Probable Assignment [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"

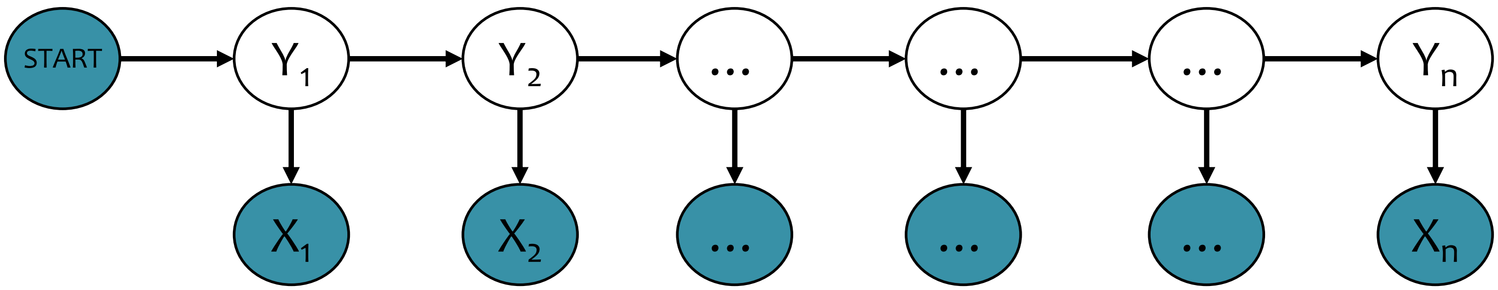
Inference in HMMs

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

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

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

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

MIDTERM EXAM LOGISTICS

Midterm Exam

- **Time / Location**
  - **Time:** Evening Exam
    Thu, Apr. 4 at 6:30pm – 8:00pm
  - **Room**: We will contact each student individually with **your room assignment**. The rooms are **not** based on section.
  - **Seats:** There will be **assigned seats**. Please arrive early.
  - Please watch Piazza carefully for announcements regarding room / seat assignments.
- **Logistics**
  - Covered material: Lecture 9 – Lecture 18 (95%), Lecture 1 – 8 (5%)
  - Format of questions:
    - Multiple choice
    - True / False (with justification)
    - Derivations
    - Short answers
    - Interpreting figures
    - Implementing algorithms on paper
  - No electronic devices
  - You are allowed to **bring** one 8½ x 11 sheet of notes (front and back)

Midterm Exam

- **How to Prepare**
  - Attend the midterm review lecture (right now!)
  - Review prior year's exam and solutions (we'll post them)
  - Review this year's homework problems
  - Consider whether you have achieved the "learning objectives" for each lecture / section

Midterm Exam

- **Advice (for during the exam)**
  - Solve the easy problems first
    (e.g. multiple choice before derivations)
    - if a problem seems extremely complicated you're likely missing something
  - Don't leave any answer blank!
  - If you make an assumption, write it down
  - If you look at a question and don't know the answer:
    - we probably haven't told you the answer
    - but we've told you enough to work it out
    - imagine arguing for some answer and see if you like it

Topics for Midterm 1

- Foundations
  - Probability, Linear Algebra, Geometry, Calculus
  - Optimization
- Important Concepts
  - Overfitting
  - Experimental Design
- Classification
  - Decision Tree
  - KNN
  - Perceptron
- Regression
  - Linear Regression

Topics for Midterm 2

- Classification
  - Binary Logistic Regression
  - Multinomial Logistic Regression
- Important Concepts
  - Regularization
  - Feature Engineering
- Feature Learning
  - Neural Networks
  - Basic NN Architectures
  - Backpropagation
- Learning Theory
  - PAC Learning
- Generative Models
  - Generative vs. Discriminative
  - MLE / MAP
  - Naïve Bayes

SAMPLE QUESTIONS

Sample Questions

3.2 Logistic regression

Given a training set $\{(x_i, y_i), i = 1, \ldots, n\}$ where $x_i \in \mathbb{R}^d$ is a feature vector and $y_i \in \{0, 1\}$ is a binary label, we want to find the parameters $\hat{w}$ that maximize the likelihood for the training set, assuming a parametric model of the form

$$p(y = 1|x; w) = \frac{1}{1 + \exp(-w^T x)}.$$

The conditional log likelihood of the training set is

$$\ell(w) = \sum_{i=1}^{n} y_i \log p(y_i, |x_i; w) + (1 - y_i) \log(1 - p(y_i, |x_i; w)),$$

and the gradient is

$$\nabla \ell(w) = \sum_{i=1}^{n} (y_i - p(y_i|x_i; w)) x_i.$$

(b) [5 pts.] What is the form of the classifier output by logistic regression?

(c) [2 pts.] **Extra Credit:** Consider the case with binary features, i.e, $x \in \{0, 1\}^d \subset \mathbb{R}^d$, where feature x_1 is rare and happens to appear in the training set with only label 1. What is $\hat{w}_1$? Is the gradient ever zero for any finite w? Why is it important to include a regularization term to control the norm of $\hat{w}$?

Samples Questions

2.1 Train and test errors

In this problem, we will see how you can debug a classifier by looking at its train and test errors. Consider a classifier trained till convergence on some training data $\mathcal{D}^{\text{train}}$, and tested on a separate test set $\mathcal{D}^{\text{test}}$. You look at the test error, and find that it is very high. You then compute the training error and find that it is close to 0.

1. **[4 pts]** Which of the following is expected to help? Select all that apply.

   (a) Increase the training data size.

   (b) Decrease the training data size.

   (c) Increase model complexity (For example, if your classifier is an SVM, use a more complex kernel. Or if it is a decision tree, increase the depth).

   (d) Decrease model complexity.

   (e) Train on a combination of $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ and test on $\mathcal{D}^{\text{test}}$

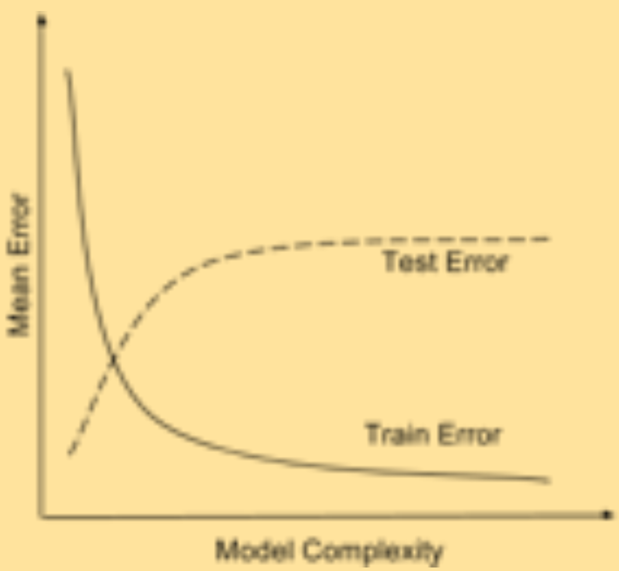
   (f) Conclude that Machine Learning does not work.

Samples Questions

2.1 Train and test errors

In this problem, we will see how you can debug a classifier by looking at its train and test errors. Consider a classifier trained till convergence on some training data $\mathcal{D}^{\text{train}}$, and tested on a separate test set $\mathcal{D}^{\text{test}}$. You look at the test error, and find that it is very high. You then compute the training error and find that it is close to 0.

4. **[1 pts]** Say you plot the train and test errors as a function of the model complexity. Which of the following two plots is your plot expected to look like?



(a)

(b)

Sample Questions

5 Learning Theory [20 pts.]

(a) [3 pts.] **T or F**: It is possible to label 4 points in $\mathbb{R}^2$ in all possible 2^4 ways via linear separators in $\mathbb{R}^2$.

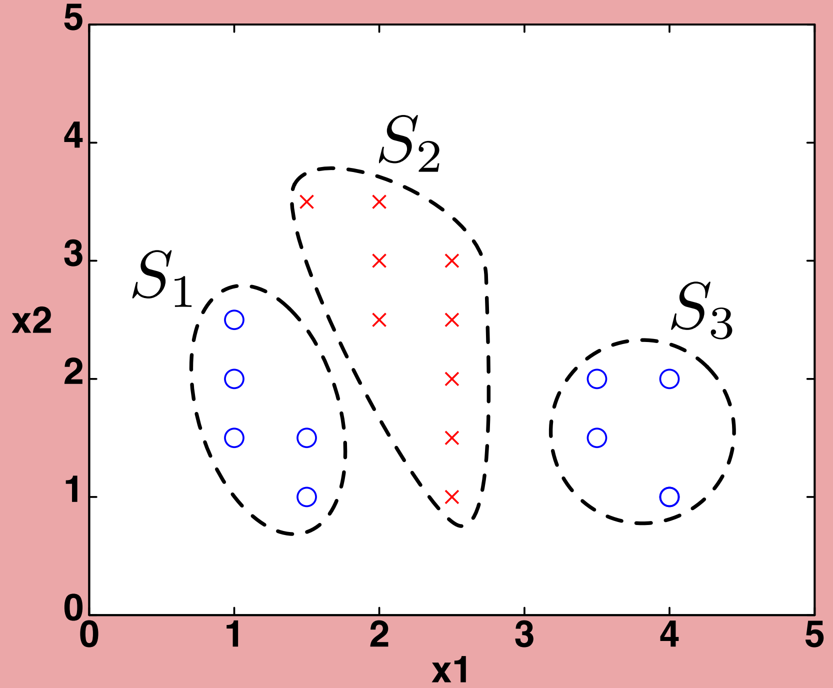
(d) [3 pts.] **T or F**: The VC dimension of a concept class with infinite size is also infinite.

(f) [3 pts.] **T or F**: Given a realizable concept class and a set of training instances, a consistent learner will output a concept that achieves 0 error on the training instances.

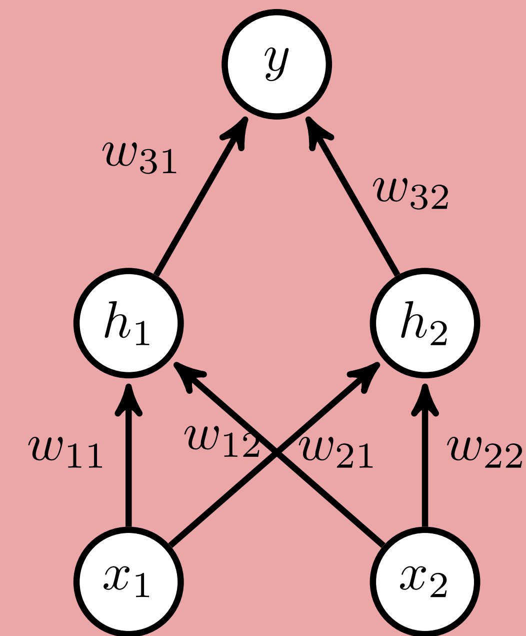
Sample Questions

Neural Networks

Can the neural network in Figure (b) correctly classify the dataset given in Figure (a)?



(a) The dataset with groups S_1, S_2, and S_3.

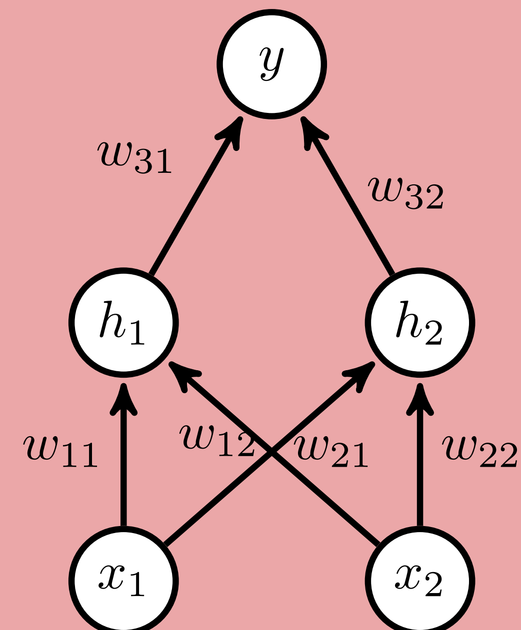


(b) The neural network architecture

Sample Questions

Neural Networks

Apply the backpropagation algorithm to obtain the partial derivative of the mean-squared error of y with the true value y^* with respect to the weight w_{22} assuming a sigmoid nonlinear activation function for the hidden layer.



(b) The neural network architecture

Sample Questions

1.2 Maximum Likelihood Estimation (MLE)

Assume we have a random sample that is Bernoulli distributed $X_1, \ldots, X_n \sim \text{Bernoulli}(\theta)$. We are going to derive the MLE for θ. Recall that a Bernoulli random variable X takes values in $\{0, 1\}$ and has probability mass function given by

$$P(X; \theta) = \theta^X (1 - \theta)^{1-X}.$$

(a) [2 pts.] Derive the likelihood, $L(\theta; X_1, \ldots, X_n)$.

(c) **Extra Credit:** [2 pts.] Derive the following formula for the MLE: $\hat{\theta} = \frac{1}{n} \left(\sum_{i=1}^n X_i\right)$.

Sample Questions

1.3 MAP vs MLE

Answer each question with **T** or **F** and **provide a one sentence explanation of your answer**:

(a) [2 pts.] **T or F:** In the limit, as n (the number of samples) increases, the MAP and MLE estimates become the same.

Sample Questions

1.1 Naive Bayes

You are given a data set of 10,000 students with their sex, height, and hair color. You are trying to build a classifier to predict the sex of a student, so you randomly split the data into a training set and a testing set. Here are the specifications of the data set:

- sex $\in$ {male,female}
- height $\in$ [0,300] centimeters
- hair $\in$ {brown, black, blond, red, green}
- 3240 men in the data set
- 6760 women in the data set

Under the assumptions necessary for Naive Bayes (not the distributional assumptions you might naturally or intuitively make about the dataset) answer each question with **T** or **F** and **provide a one sentence explanation of your answer**:

(a) [2 pts.] **T or F:** As height is a continuous valued variable, Naive Bayes is not appropriate since it cannot handle continuous valued variables.

(c) [2 pts.] **T or F:** $P(\texttt{height}|\texttt{sex}, \texttt{hair}) = P(\texttt{height}|\texttt{sex})$.