10-301/601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Hidden Markov Models (Part II)

Matt Gormley
Lecture 19
Mar. 28, 2022

Reminders

- Exam 2 (Thu, Mar 3rd)
  - Thu, Mar. 31, 6:30pm – 8:30pm
- Practice for Exam 2
  - Practice problems released on course website
    - Out: Fri, Mar. 25
  - Mock Exam 2
    - Out: Fri, Mar. 25
    - Due Wed, Mar. 30 at 11:59pm

TO HMMS AND BEYOND…

Unsupervised Learning for HMMs

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

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

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

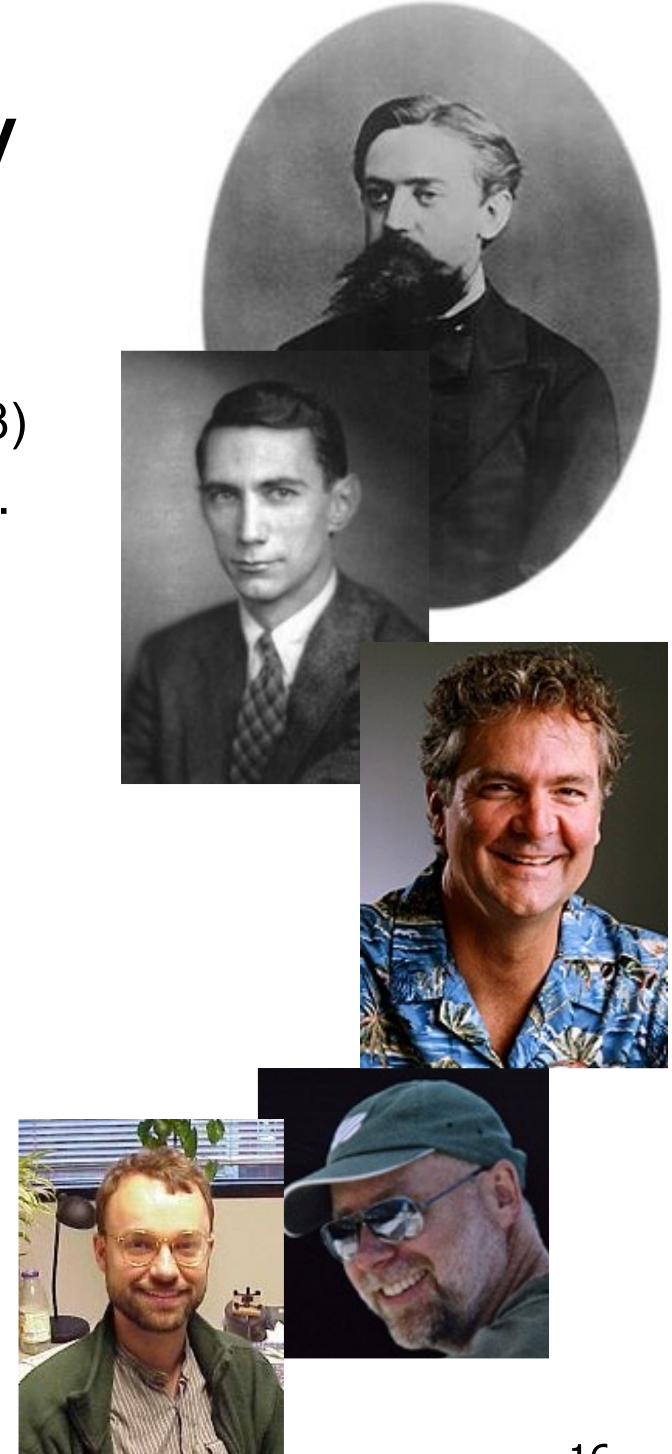
The log-likelihood of the data is thus:

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \log \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \\ &= \sum_{i=1}^{N} \log \sum_{\mathbf{y} \in \mathcal{Y}} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{y}) \end{aligned} \tag{3}$$

Beyond the scope of today's lecture!

HMMs: History

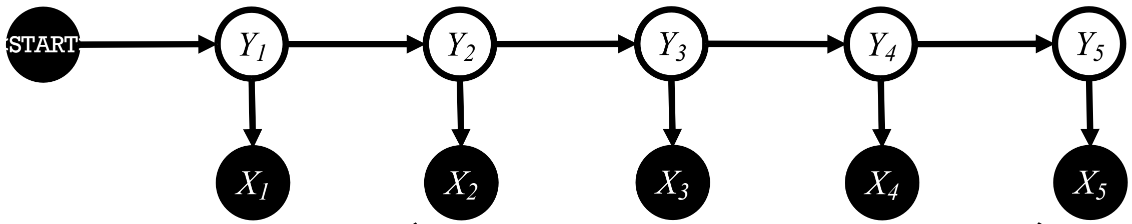
- Markov chains: Andrey Markov (1906)
  - Random walks and Brownian motion
- Used in Shannon's work on information theory (1948)
- Baum-Welsh learning algorithm: late 60's, early 70's.
  - Used mainly for speech in 60s-70s.
- Late 80's and 90's: David Haussler (major player in learning theory in 80's)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



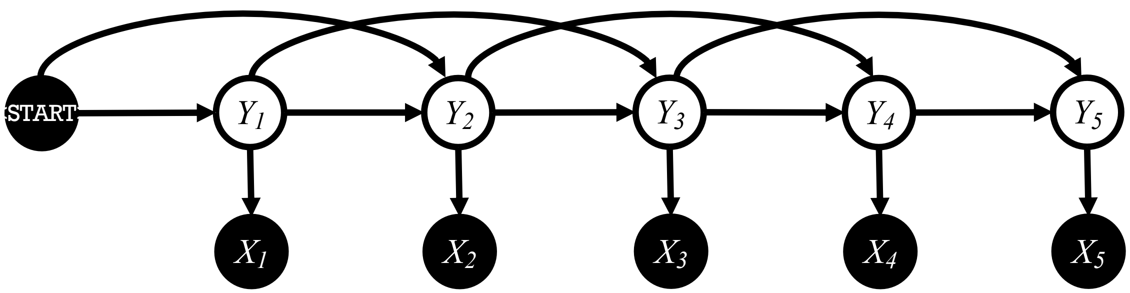
Slide from William Cohen

Higher-order HMMs

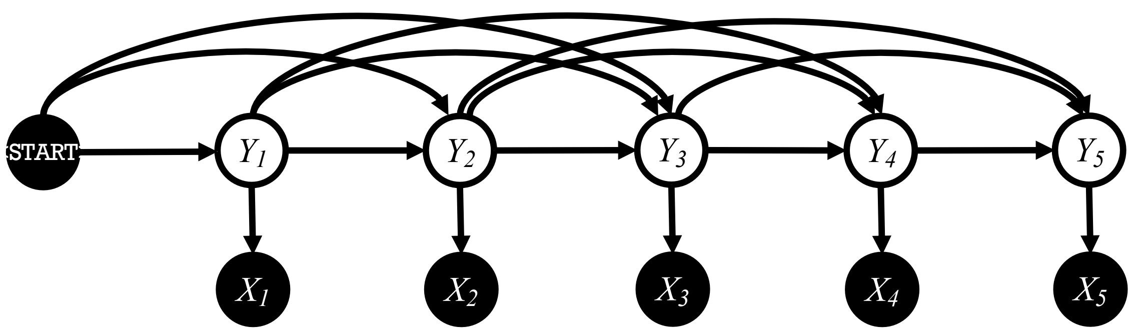
- 1^{st}-order HMM (i.e. bigram HMM)



- 2^{nd}-order HMM (i.e. trigram HMM)

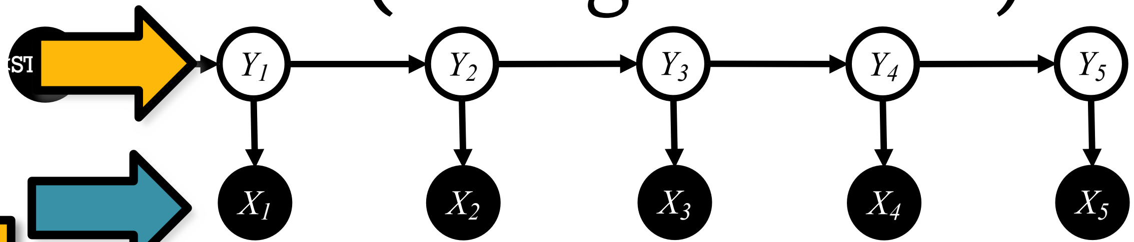


- 3^{rd}-order HMM



Higher-order HMMs

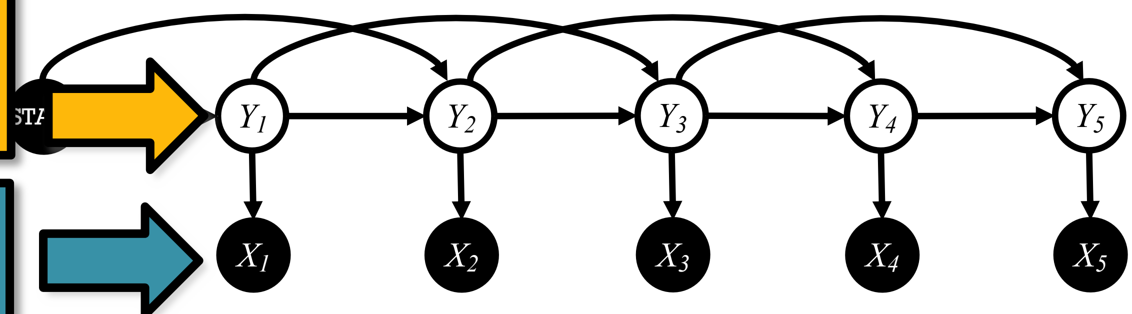
- 1^{st}-order HMM (i.e. bigram HMM)



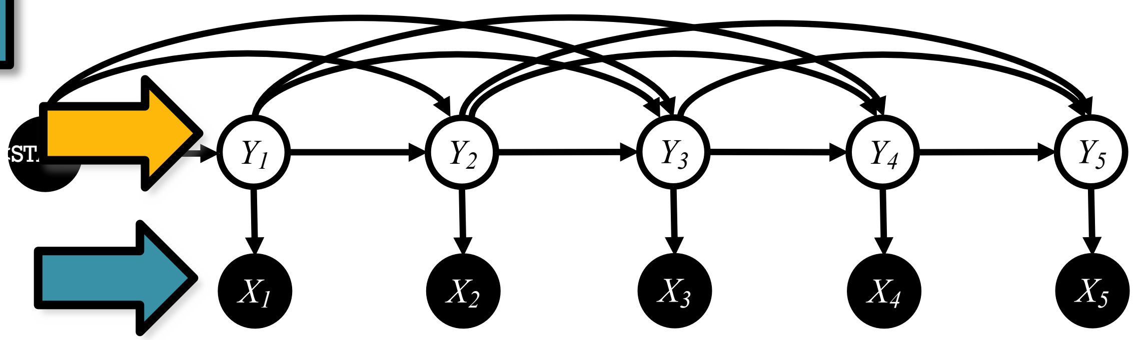
Hidden States, **y**

Observa-tions, **x**

- rder HMM (i.e. trigram HMM)



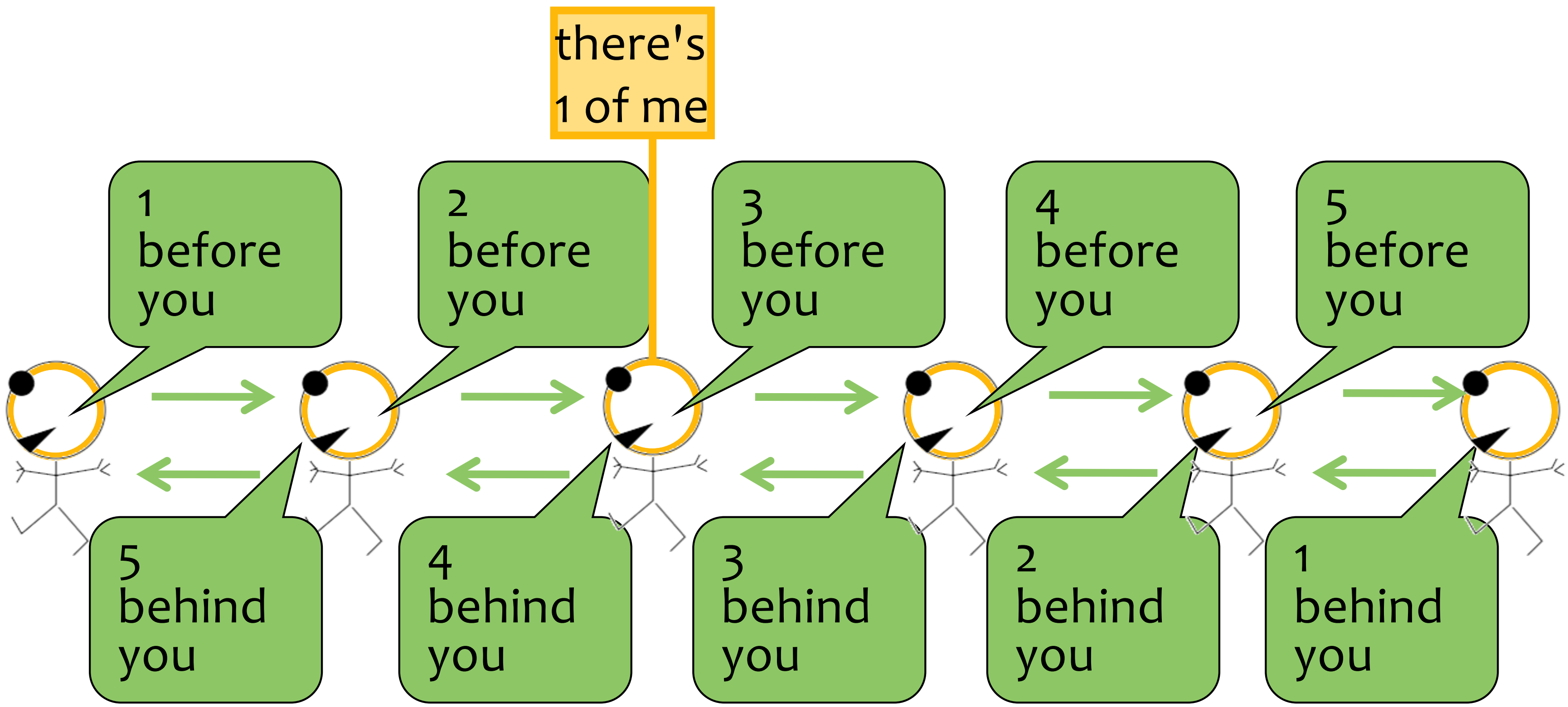
- rder HMM



BACKGROUND: MESSAGE PASSING

Great Ideas in ML: Message Passing

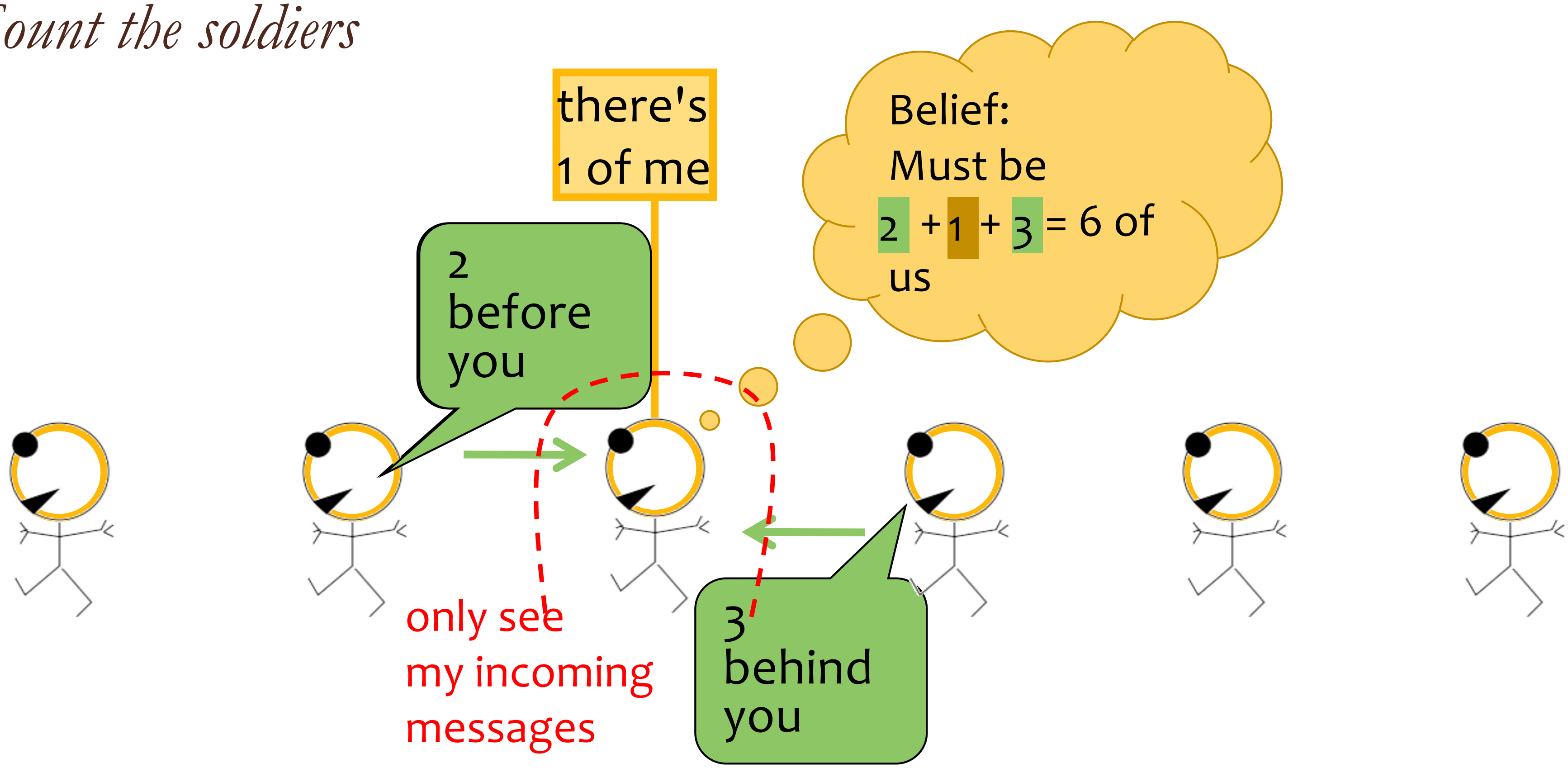
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

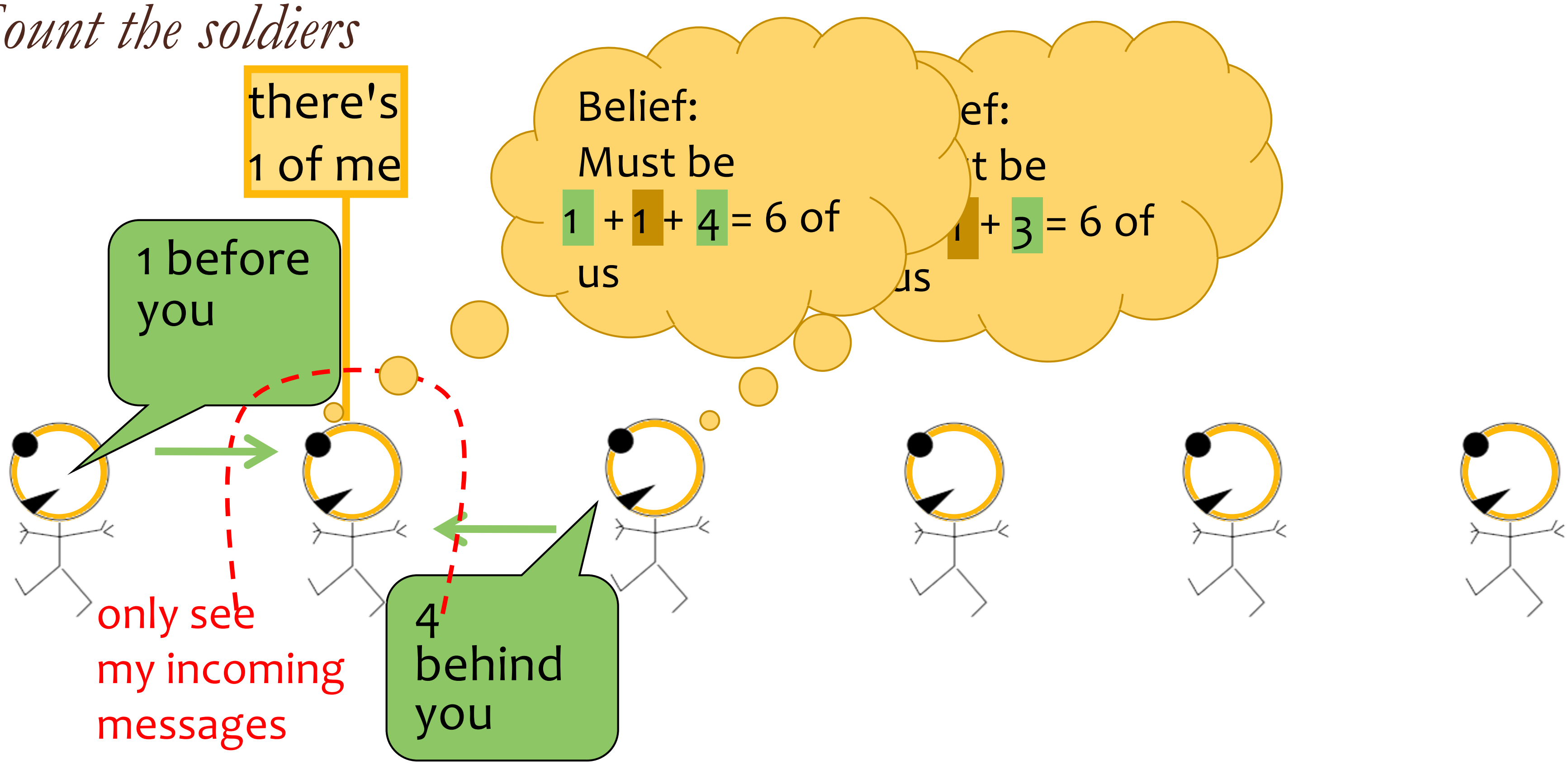
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

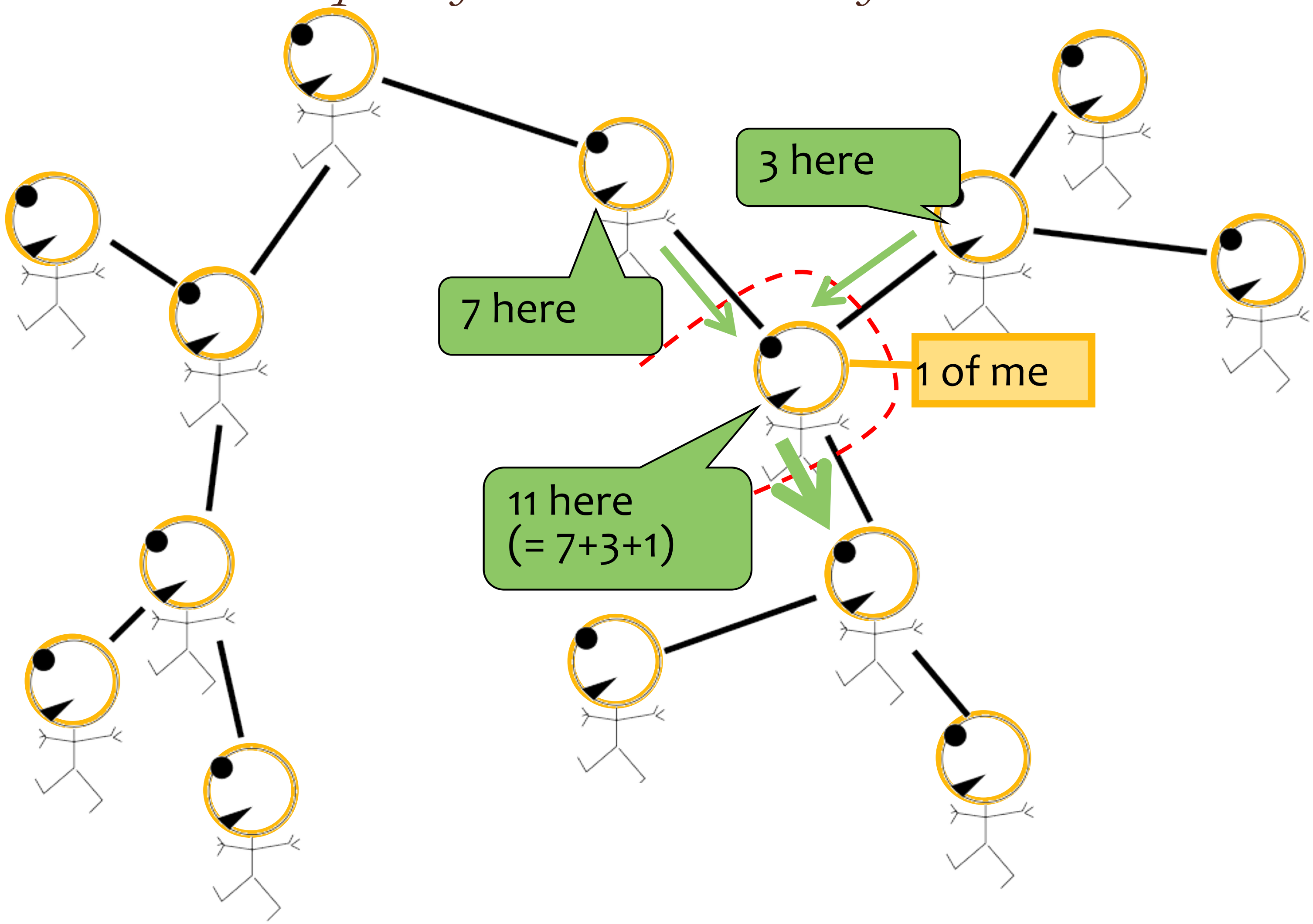
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

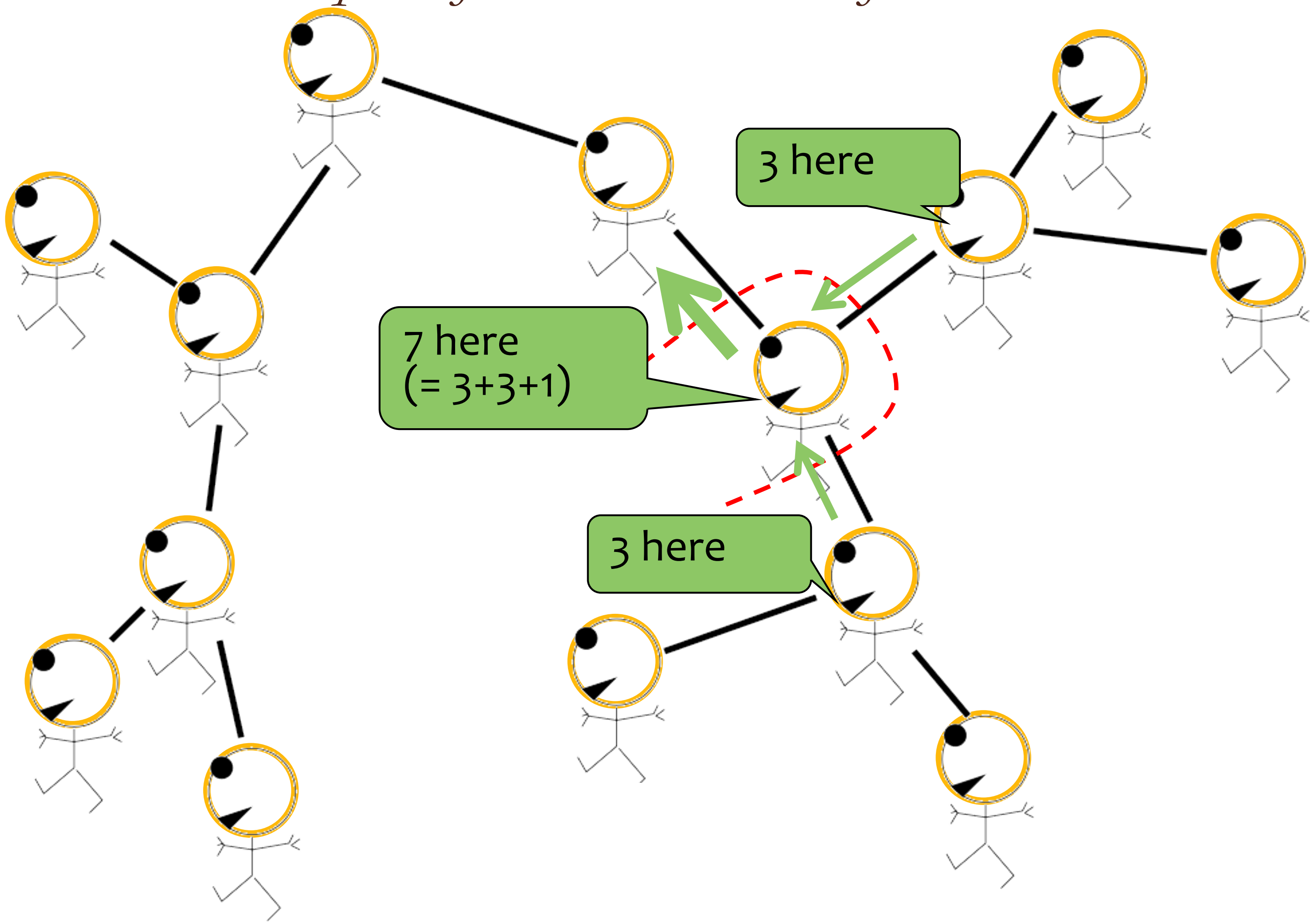
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

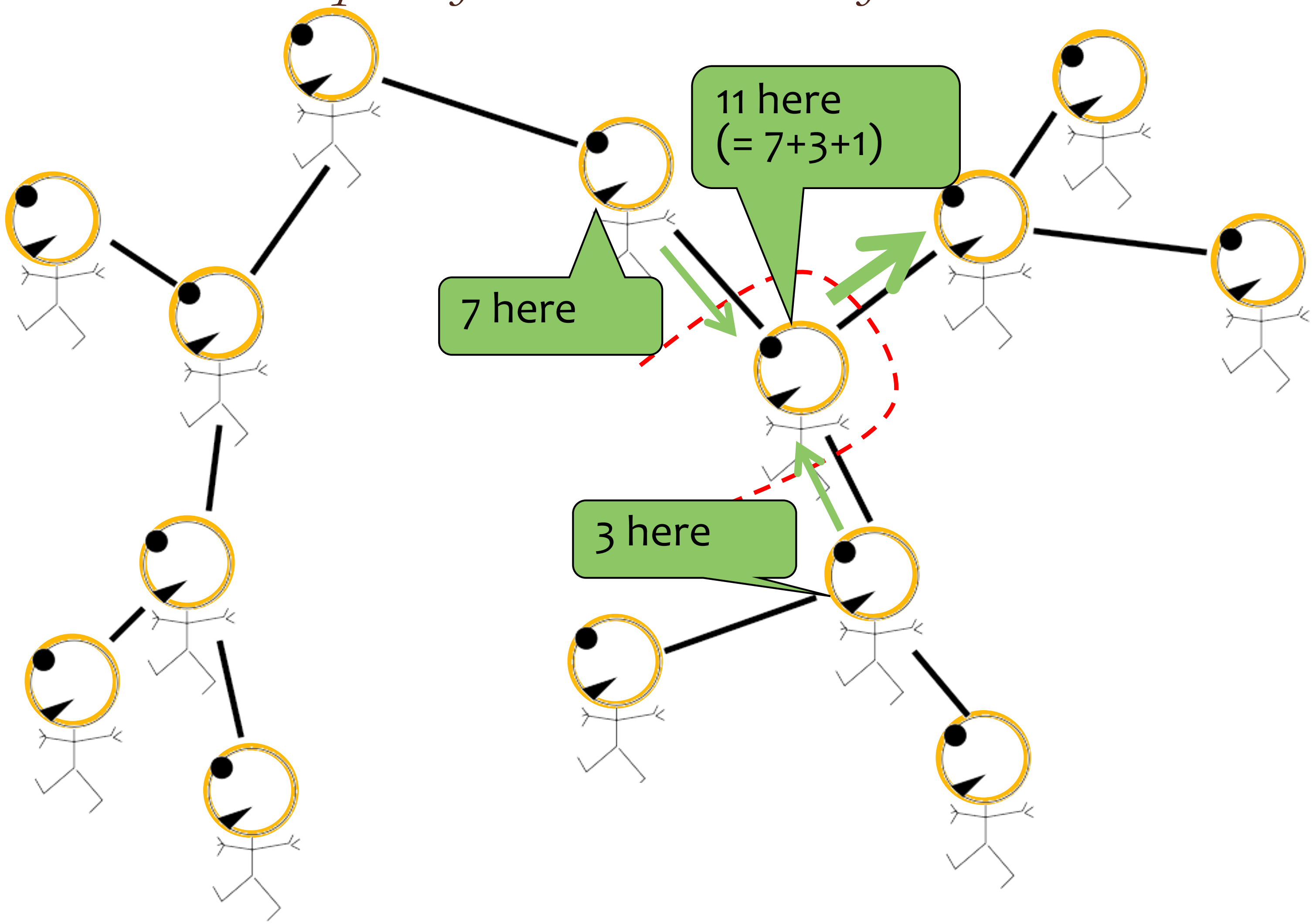
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

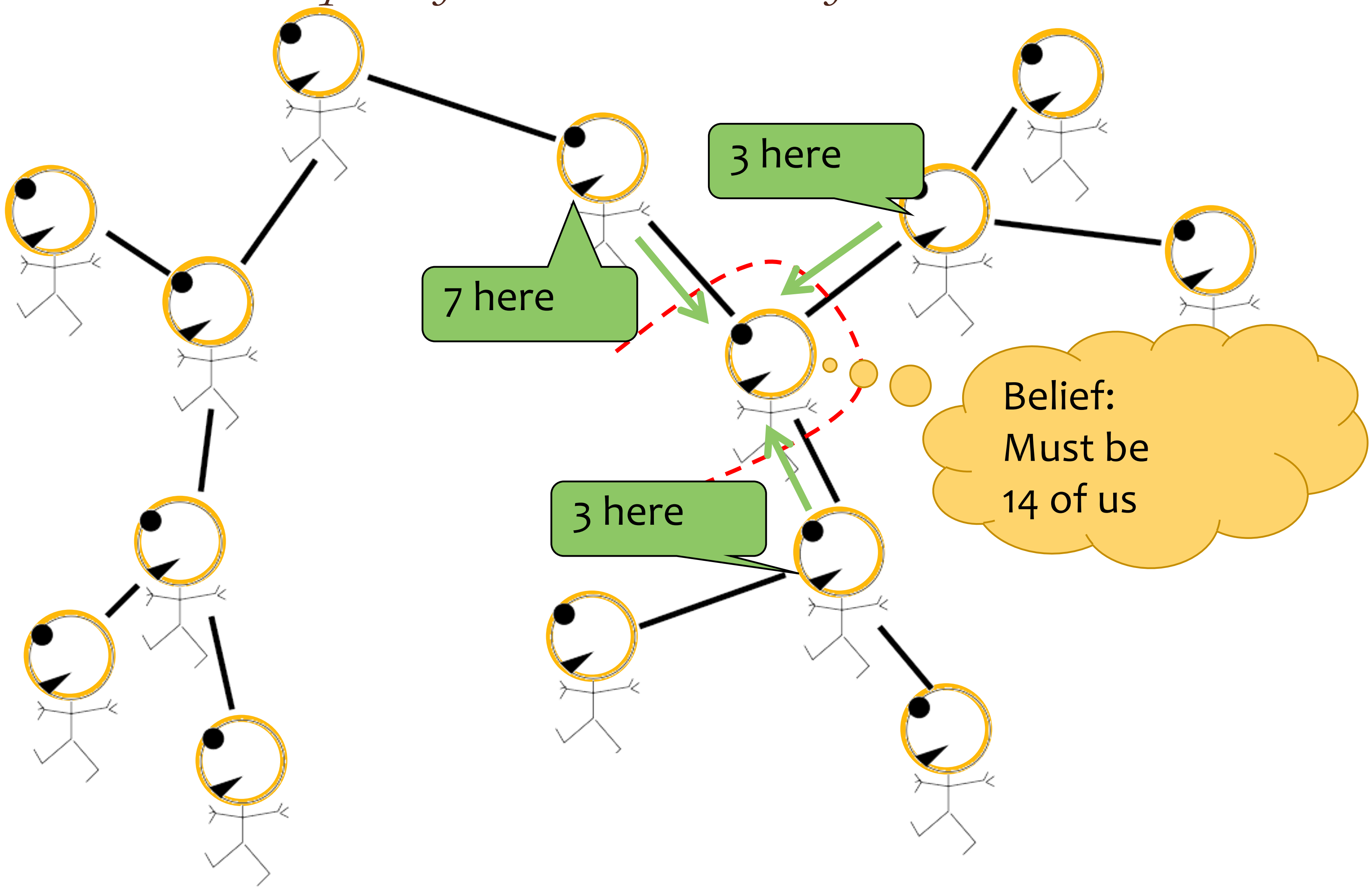
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

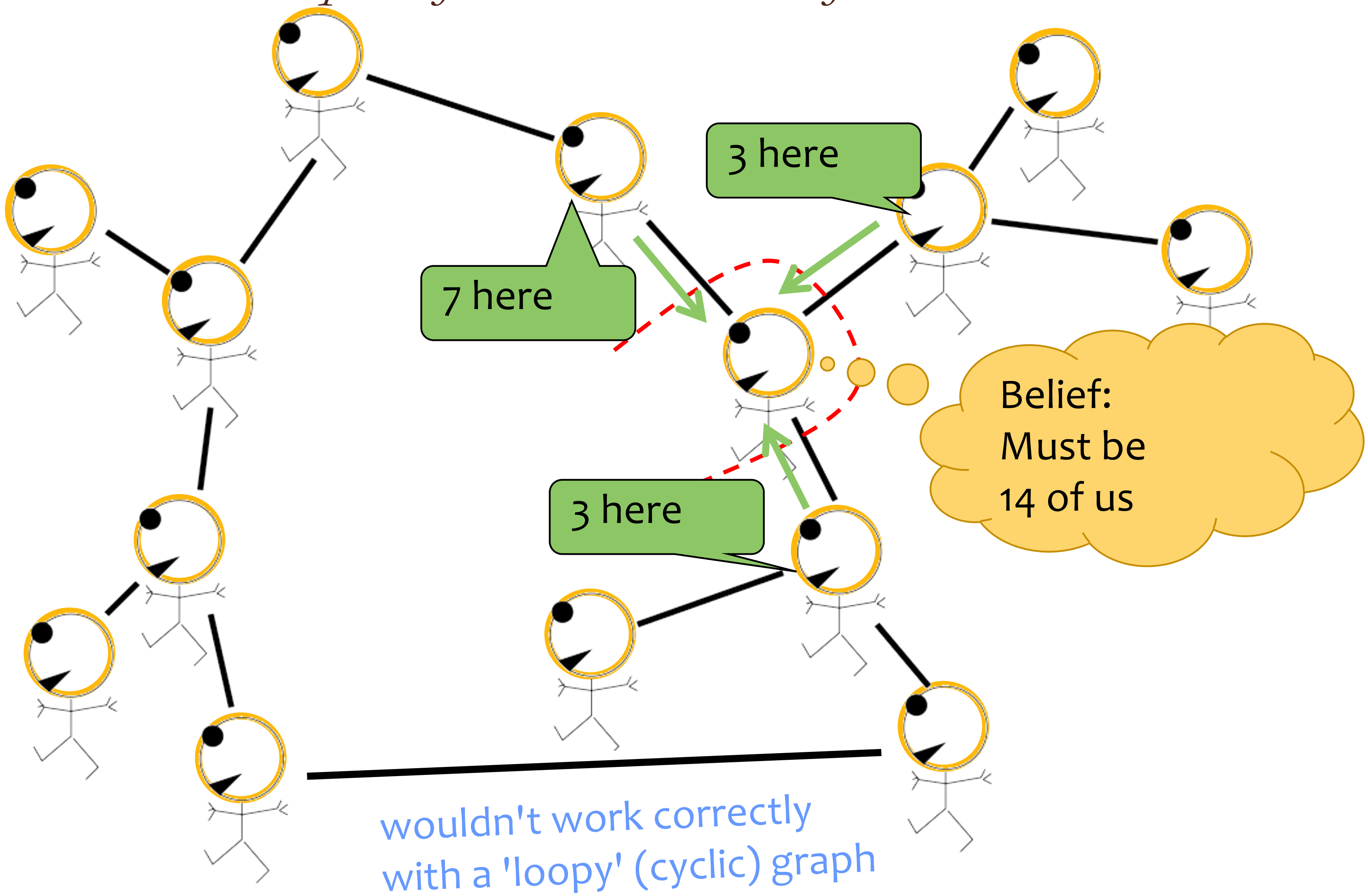
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

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

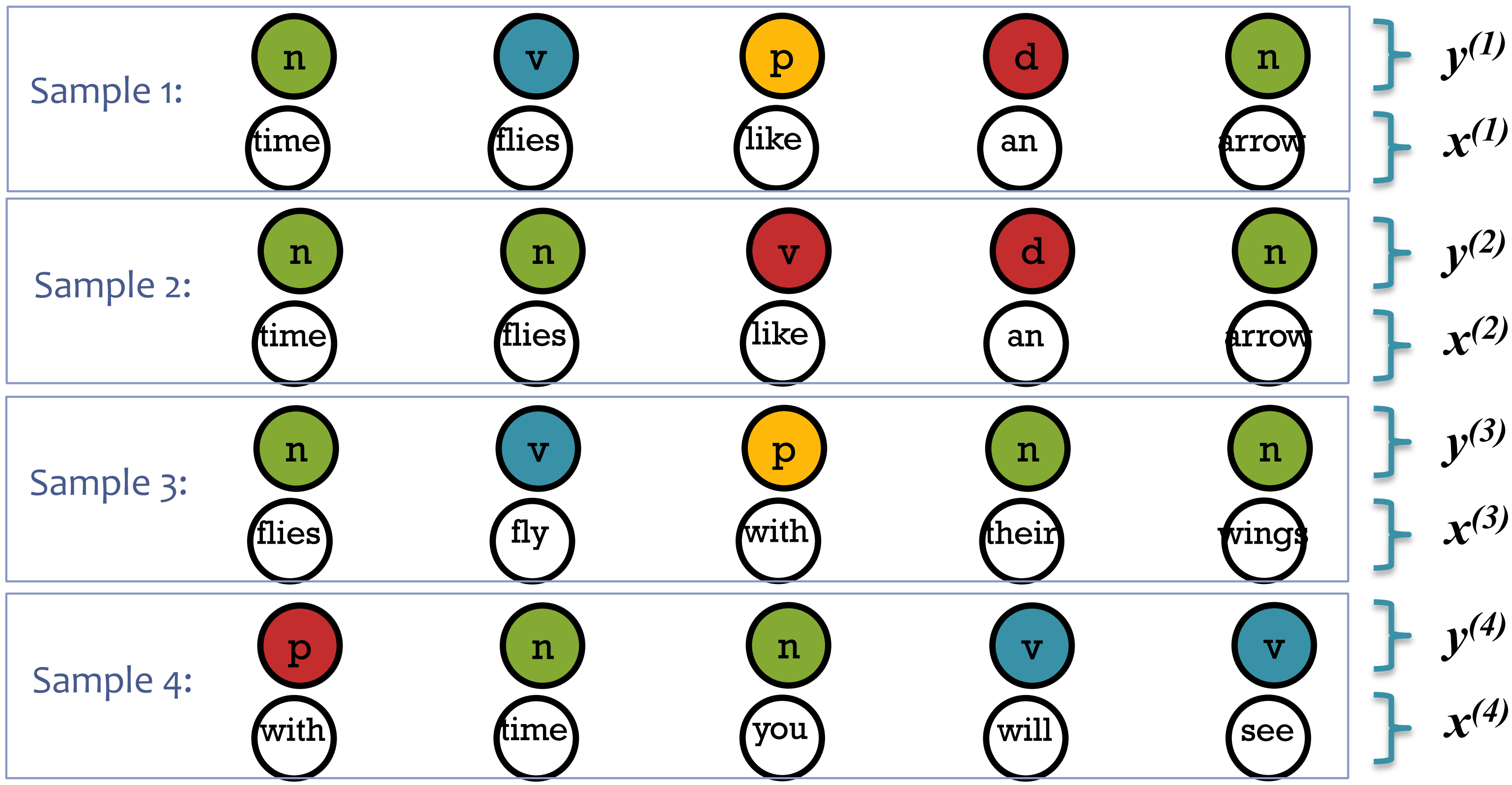
Whiteboard

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



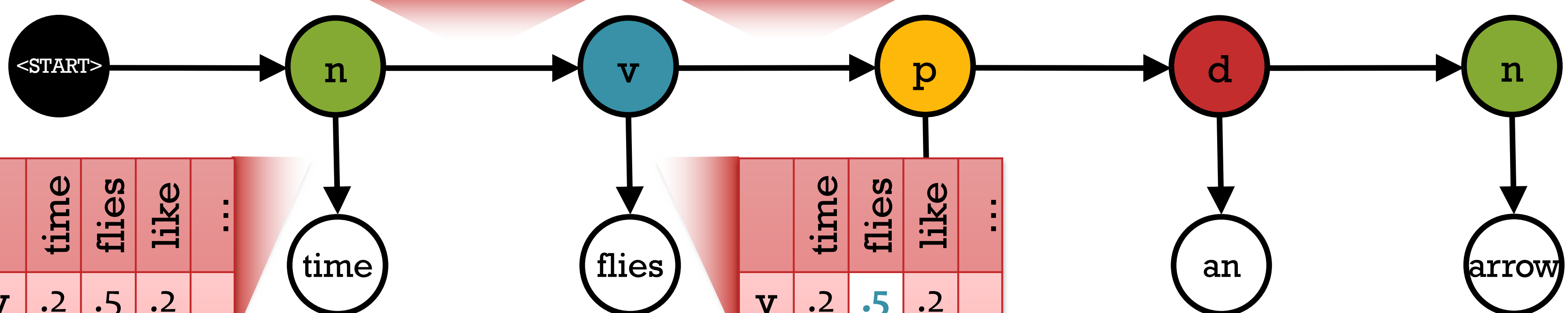
Example: HMM for POS Tagging

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

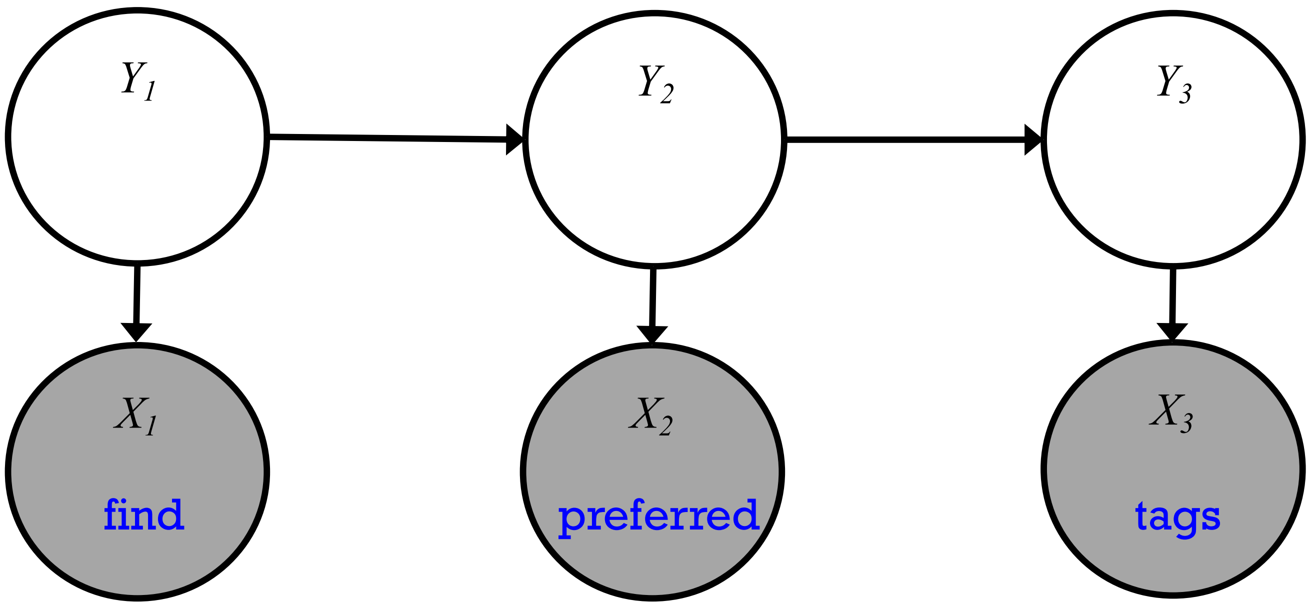
	v	n	p	d
v	.1	.4	.2	.3
n	.8	.1	.1	0
p	.2	.3	.2	.3
d	.2	.8	0	0



	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

HOW IS EFFICIENT COMPUTATION EVEN POSSIBLE?

How is efficient computation even possible?

- The short answer is **dynamic programming**!

- The key idea is this:
  - We first come up with a **recursive definition** for the quantity we want to compute
  - We then observe that many of the recursive intermediate terms are **reused** across timesteps and tags
  - We then perform **bottom-up dynamic programming** by running the recursion in reverse, **storing the intermediate quantities** along the way!

- This enables us to search the **exponentially large** space in **polynomial time**!

Derivation of Forward Algorithm

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

Derivation:

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

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

THE FORWARD-BACKWARD ALGORITHM

Inference for HMMs

Whiteboard

- Forward-backward algorithm (edge weights version)

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 $t = T, \ldots, T$:

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]

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

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

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

② For $t = 1, \ldots, T$:

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

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

③ For $t = T, \ldots, 1$:

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

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

O(K)

O(K²T)

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]

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