CS378: Natural Language Processing
Lecture 8: Expectation Maximization (EM)

The University of Texas at Austin

Eunsol Choi

Parts of this lecture adapted from Greg Durrett, Yejin Choi, Yoav Artzi

Recap: MEMM

- Model:

$$p(y_1 \dots y_n | x_1 \dots x_n) = \prod_{i=1}^{n} p(y_i | y_1 \dots y_{i-1}, x_1 \dots x_n)$$ Chain rule

$$= \prod_{i=1}^{n} p(y_i | y_{i-1}, x_1 \dots x_n)$$ Independence assumption

- Scoring:

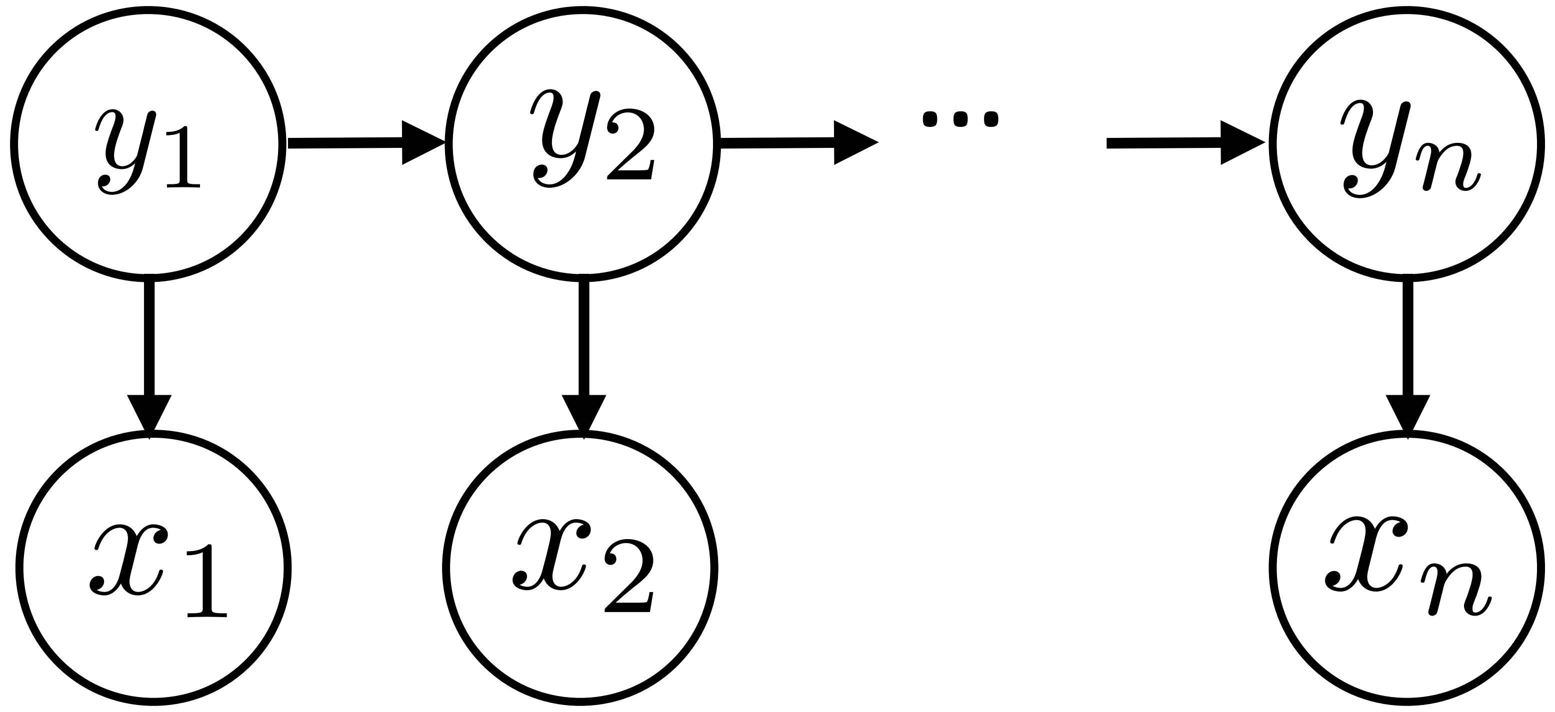
$$p(y_i | y_{i-1}, x_1 \dots x_n) = \frac{e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y')}}$$

- Learning Objective: log likelihood of training data

$$L = \sum_{i=1}^{n} \log P(y_i | y_1, \dots, y_{i-1}, x_1, x_2 \dots x_n)$$

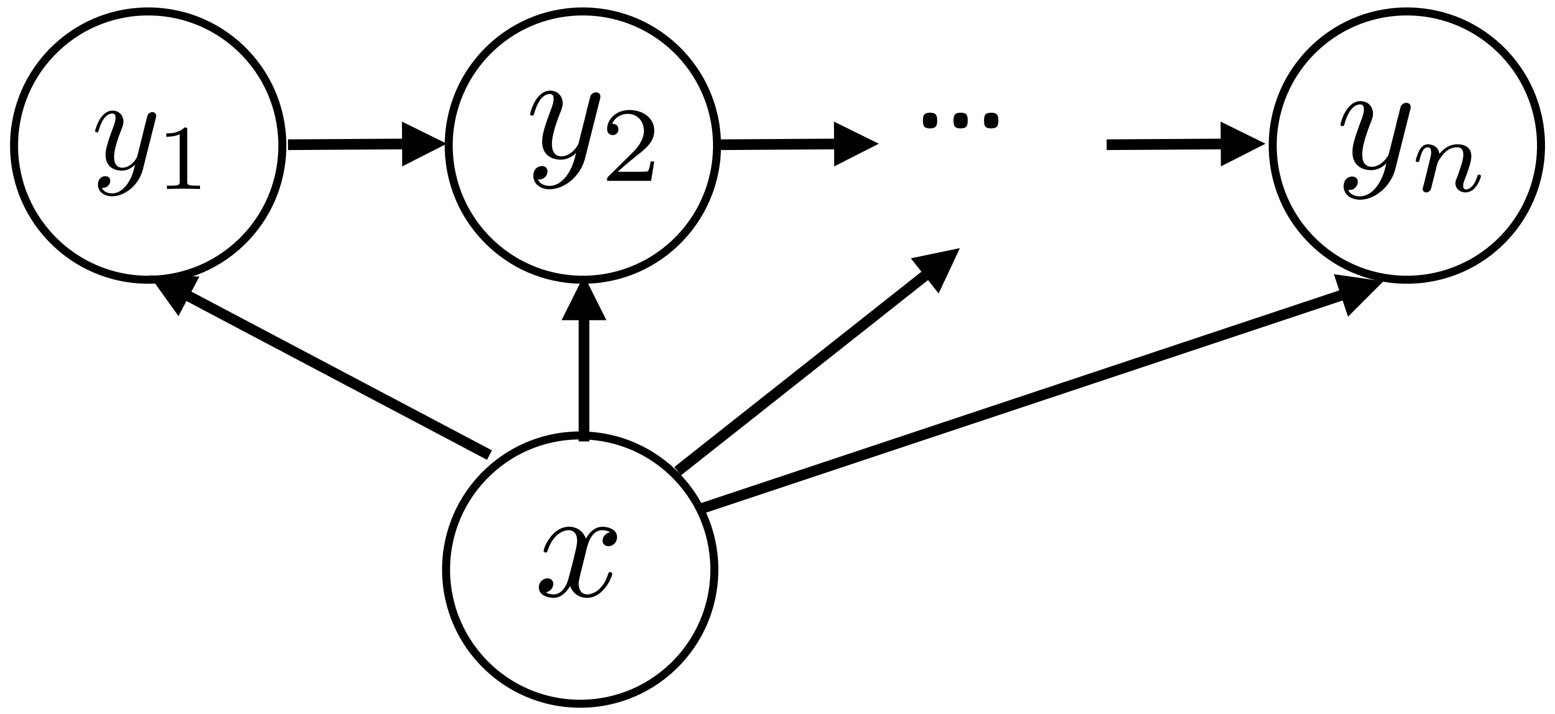
Recap: HMM vs. MEMM

- HMM models joint distribution:



$$p(x_1 \ldots x_n, y_1 \ldots y_n) = q(STOP|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)$$

- MEMM models conditional distribution:



$$p(y_1 \ldots y_n | x_1 \ldots x_n) = \prod_{i=1}^{n} p(y_i | y_{i-1}, x_1 \ldots x_n)$$

The Viterbi Algorithm

- Dynamic program for computing (for all i)

$$\pi(i, y_i) = \max_{y_1 \dots y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_i)$$

- Iterative Computation: $\pi(0, y_0) = \begin{cases} 1 \text{ if } y_0 == START \\ 0 \text{ otherwise} \end{cases}$
- For I = 1… n:
  - Store score

$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

  - Store back-pointer

$$bp(i, y_i) = \arg\max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

- We end up with a single most likely sequence.

Recap: Decoding for MEMM

- Given your model, finding the highest scoring **y**

- Not very different from decoding for HMMs - dynamic programming!

- Viterbi for HMMs

$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

- Viterbi for MEMM

$$\pi(i, y_i) = \max_{y_{i-1}} p(y_i|y_{i-1}, x_1 \ldots x_n)\pi(i-1, y_{i-1})$$

Recap: Perceptron vs. CRF

- Both compute a score $w \cdot \phi(\mathbf{x}, \mathbf{y})$
- Perceptron:
  - Iteratively processes the data, reacting to training errors
    - $\mathbf{y}* = \text{argmax}_{\mathbf{y} \in Y} w \cdot \phi(\mathbf{x}, \mathbf{y})$
    - $w = w + \phi(\mathbf{x_i}, \mathbf{y_i}) - \phi(\mathbf{x_i}, \mathbf{y}*)$
- Conditional Random Field:
  - Model a probability distribution over y with softmax

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} \text{exp}(\sum_j w \cdot \phi_j(\mathbf{x}, \mathbf{y}))$$

- For efficient inference, both limit to local features: $\phi(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{n} \phi(x, j, y_{j-1}, y_j)$

Another goal: Marginal Inference

- What if we are interested in the distribution of tags for each step?
- Find the marginal probability of each tag y_i: $p(y_i | x_1, \ldots x_n)$

$$p(x_1 \ldots x_n, y_i) = \sum_{y_1 \cdots y_{i-1}} \sum_{y_{i+1} \cdots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$

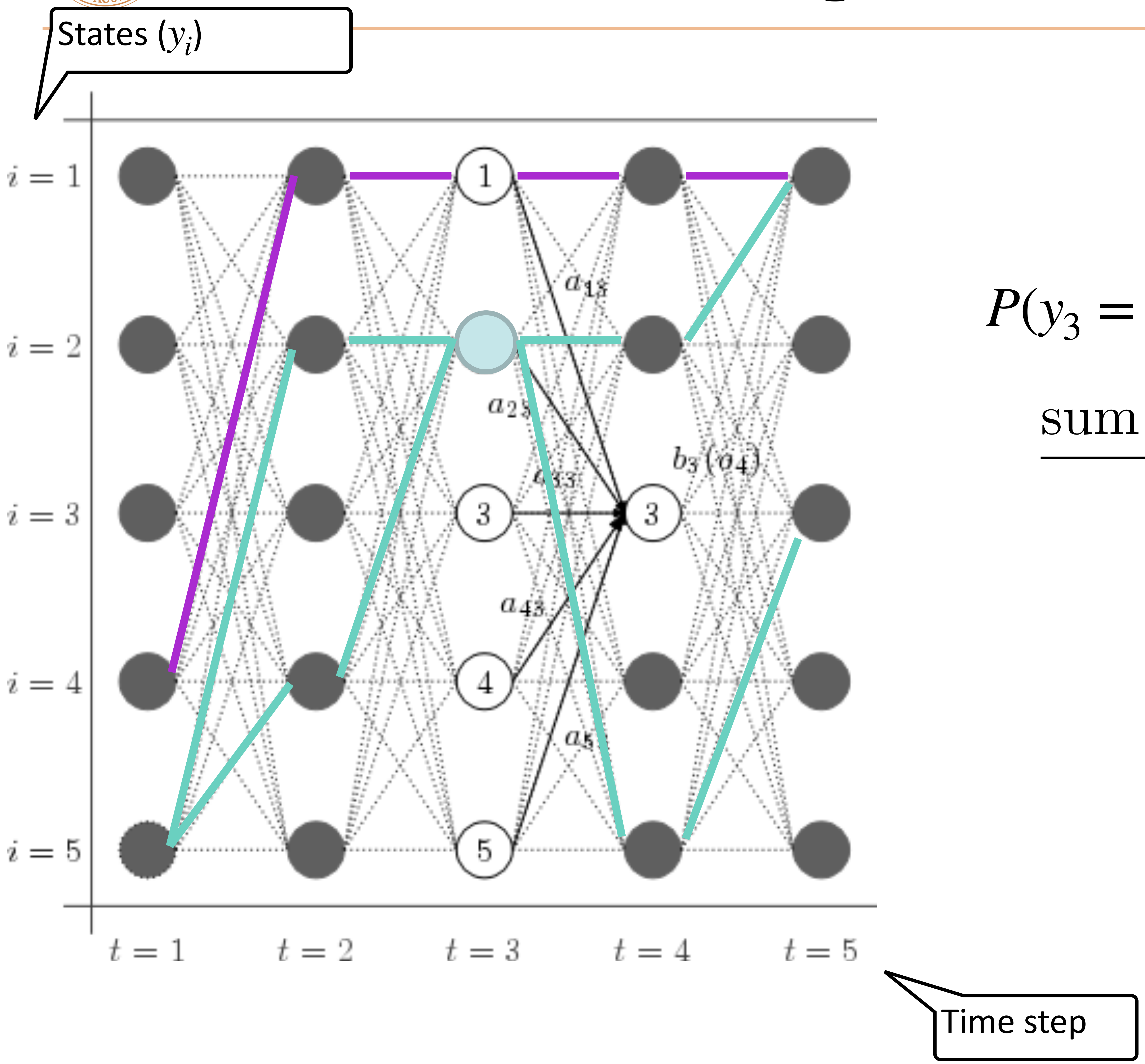
$$p(x_1 \ldots x_n) = \sum_{y_1 \cdots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$

$$= \ldots ? \ldots \quad p(x_1 \ldots x_n, y_i)$$

$$= \sum_{y_i} p(x_1 \ldots x_n, y_i)$$

- Let’s compute this together!

Marginal Inference

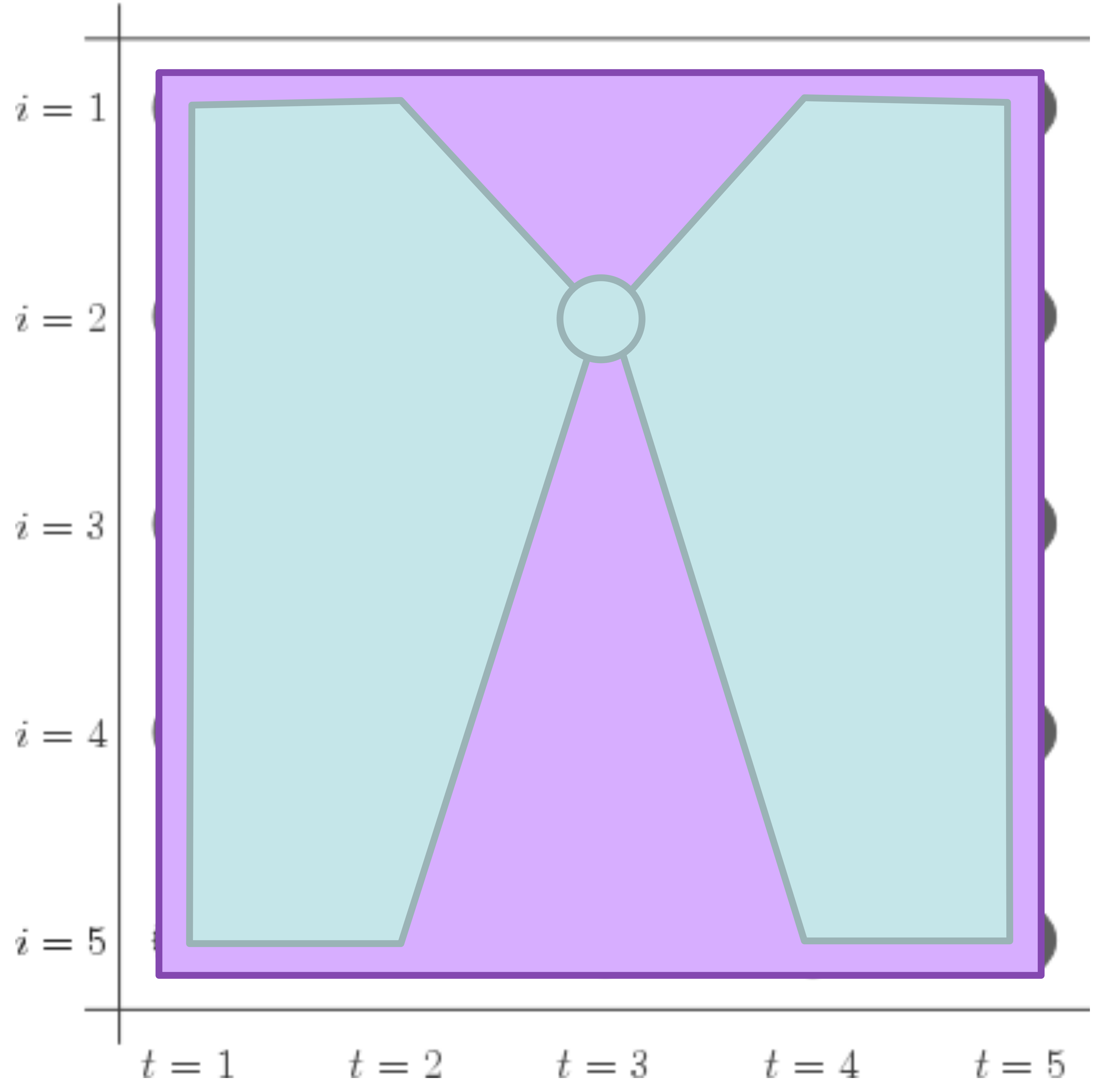


$$p(y_i \mid x_1, \ldots x_n)$$

$$P(y_3 = 2 \mid x_1, \ldots x_n) = \frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$

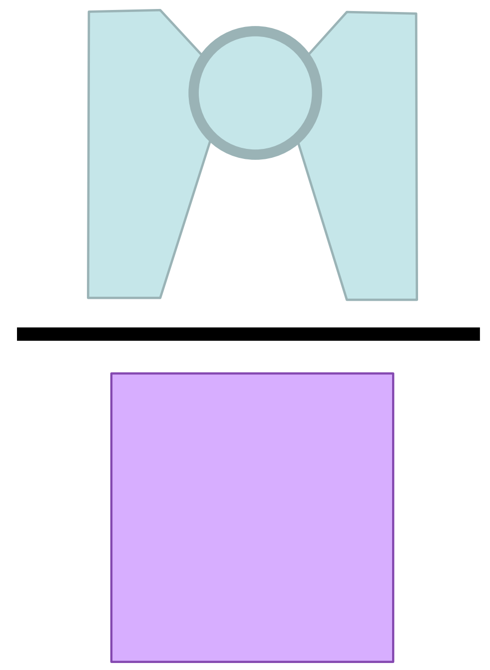
The probability of input data sequence $p(x_1 \ldots x_n)$

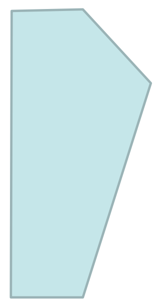
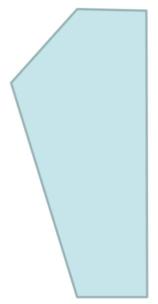
Marginal Inference



$$P(y_3 = 2 \,|\, x_1, \ldots x_n) = \frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$

=



- Easiest and most flexible to do one pass to compute  and one to compute 

slide credit: Dan Klein

Marginal Inference

- Decompose into two probabilities

$$p(x_1 \dots x_n, y_i) = p(x_1 \dots x_i, y_i) p(x_{i+1} \dots x_n | y_i)$$

- Dynamic programming on both sides

$$\alpha(i, y_i) = p(x_1 \dots x_i, y_i) = \sum_{y_1 \dots y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_i)$$

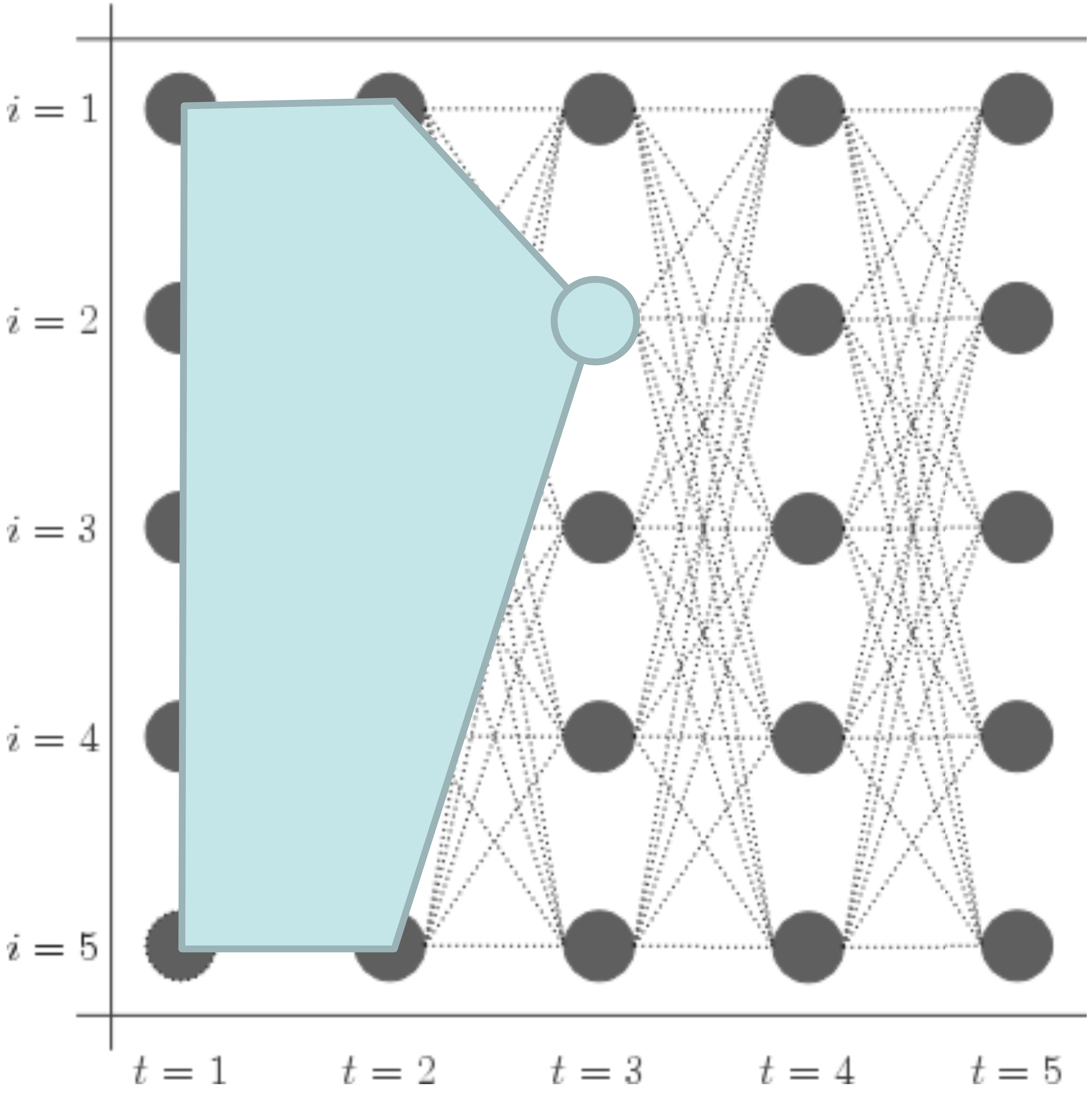
Forward pass

$$\beta(i, y_i) = p(x_{i+1} \dots x_n | y_i) = \sum_{y_{i+1} \dots y_n} p(x_{i+1} \dots x_n, y_{i+1} \dots y_n | y_i)$$

Backward pass

Forward-Backward Algorithm

$$\alpha(i, y_i) = p(x_1 \dots x_i, y_i) = \sum_{y_1 \dots y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_i)$$



- Initial: $\alpha(0, y_0) = \begin{cases} 1 \text{ if } y_0 == START \\ 0 \text{ otherwise} \end{cases}$

- Recurrence:

$$\alpha(i, y_i) = \sum_{y_{i-1}} e(x_i | y_i) q(y_i | y_{i-1}) \alpha(i-1, y_{i-1})$$

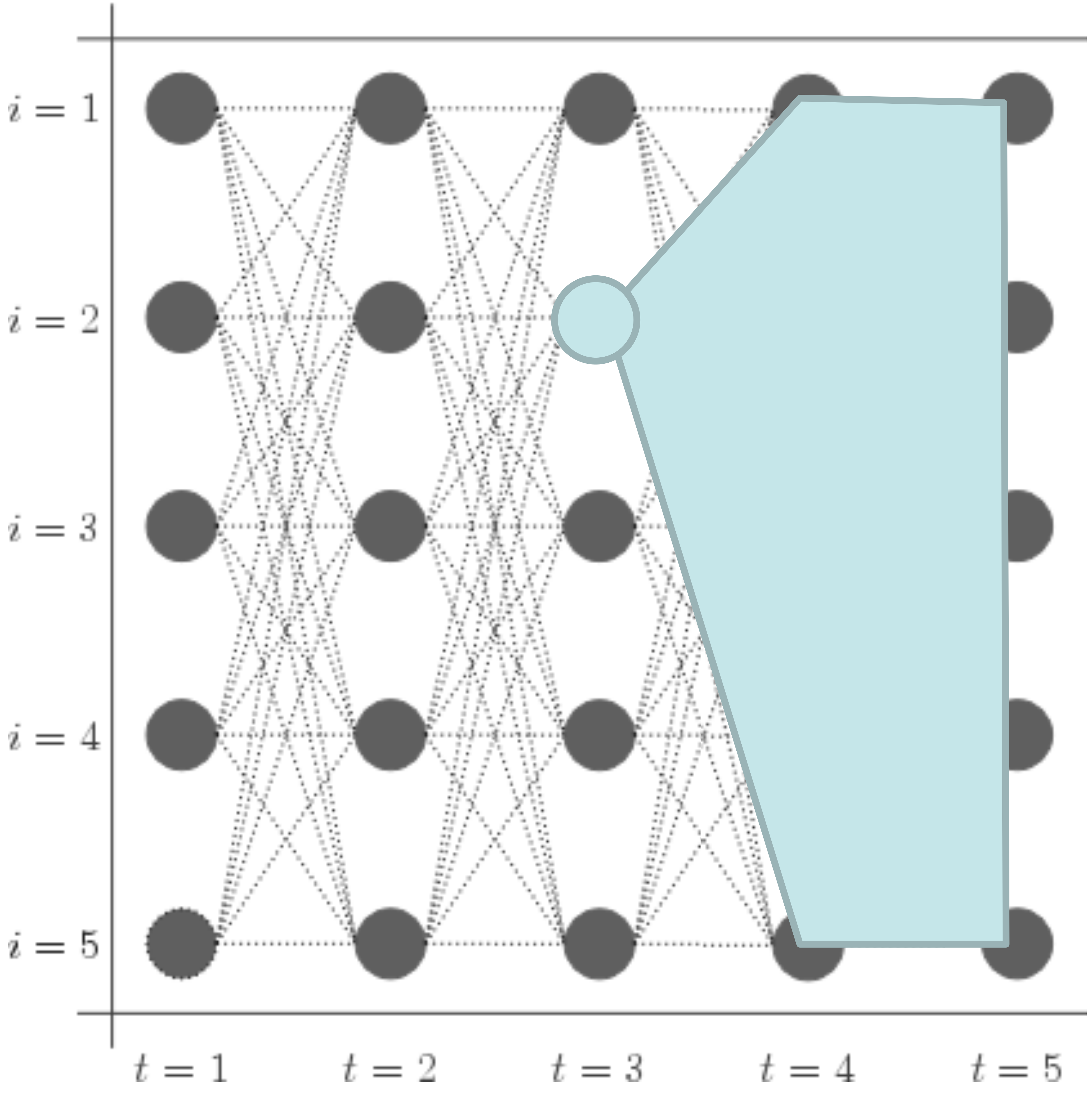
- Recap: Viterbi for HMM:

$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i | y_i) q(y_i | y_{i-1}) \pi(i-1, y_{i-1})$$

- Same as Viterbi but sum instead of max!

Forward-Backward Algorithm

$$\beta(i, y_i) = p(x_{i+1} \dots x_n | y_i) = \sum_{y_{i+1} \dots y_n} p(x_{i+1} \dots x_n, y_{i+1} \dots y_n | y_i)$$



- Initial:

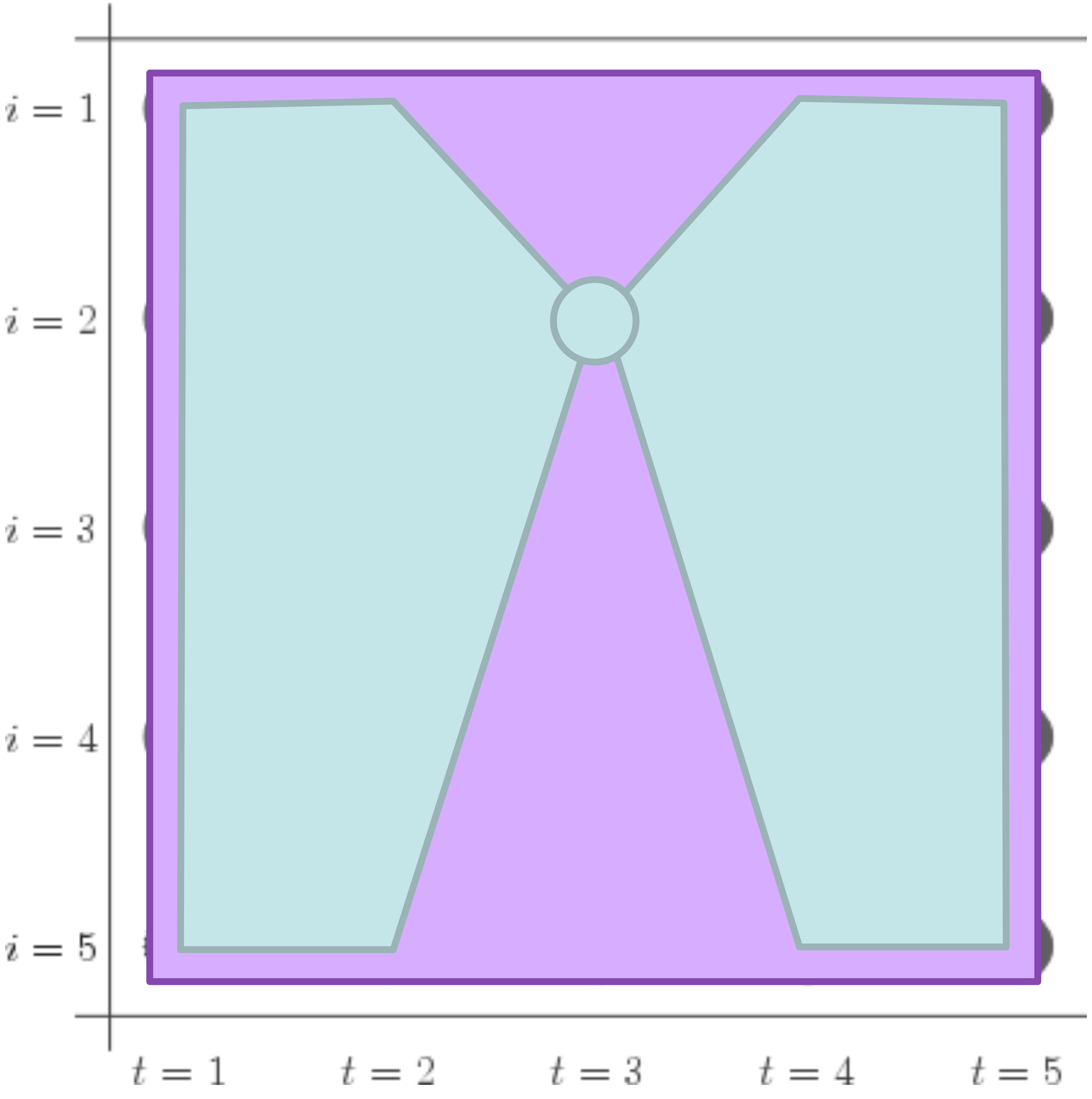
$$\beta(n, y_n) = \begin{cases} q(y_{n+1} | y_n) \text{ if } y_{n+1} = \text{STOP} \\ 0 \text{ otherwise} \end{cases}$$

- Recurrence:

$$\beta(i, y_i) = \sum_{y_{i+1}} e(x_{i+1} | y_{i+1}) q(y_{i+1} | y_i) \beta(i+1, y_{i+1})$$

- Big differences: count emission for the *next* timestep (not current one)

Forward-Backward Algorithm



$$p(y_i|x_1 \ldots x_n) = \frac{p(x_1 \ldots x_n, y_i)}{p(x_1 \ldots x_n)}$$

$$P(y_3 = 2 \,|\, x_1, \ldots x_n) = \frac{\alpha(3,2)\beta(3,2)}{\sum_i \alpha(3,i)\beta(3,i)}$$

Two Inference Methods

- Viterbi algorithm

- Forward Backward algorithm

- Computational Costs?

How are these relevant in 2021?

Published as a conference paper at ICLR 2017

STRUCTURED ATTENTION NETWORKS

Yoon Kim* **Carl Denton*** **Luong Hoang** **Alexander M. Rush**
{yoonkim@seas,carldenton@college,lhoang@g,srush@seas}.harvard.edu
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138, USA

procedure FORWARDBACKWARD(θ)
$\alpha[0, \langle t \rangle] \leftarrow 0$
$\beta[n+1, \langle t \rangle] \leftarrow 0$
for $i = 1, \ldots, n; c \in \mathcal{C}$ **do**
$\alpha[i, c] \leftarrow \bigoplus_y \alpha[i-1, y] \otimes \theta_{i-1,i}(y, c)$
for $i = n, \ldots, 1; c \in \mathcal{C}$ **do**
$\beta[i, c] \leftarrow \bigoplus_y \beta[i+1, y] \otimes \theta_{i,i+1}(c, y)$
$A \leftarrow \alpha[n+1, \langle t \rangle]$

procedure BACKPROPFORWARDBACKWARD($\theta, p, \nabla_p^{\mathcal{L}}$)
$\nabla_\alpha^{\mathcal{L}} \leftarrow \log p \otimes \log \nabla_p^{\mathcal{L}} \otimes \beta \otimes -A$
$\nabla_\beta^{\mathcal{L}} \leftarrow \log p \otimes \log \nabla_p^{\mathcal{L}} \otimes \alpha \otimes -A$
$\hat{\alpha}[0, \langle t \rangle] \leftarrow$
$\hat{\beta}[n+1, \langle t$
for $i = n, \ldots$
$\hat{\beta}[i, c]$
for $i = 1, \ldots$

Latent Predictor Networks for Code Generation

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Andrew Senior, Fumin Wang, Phil Blunsom

ACL 2016

While the number of possible paths grows exponentially, α and β can be computed efficiently using the forward-backward algorithm for Semi-Markov models (Sarawagi and Cohen, 2005), where we associate $P(r_t \mid y_1..y_{t-1}, x)$ to edges and $P(s_t \mid y_1..y_{t-1}, x, r_t)$ to nodes in the Markov chain.

The derivative $\frac{\partial \log P(y|x)}{\partial P(s_t|y_1..y_{t-1},x,r_t)}$ can be computed using the same logic:

$$\frac{\partial \alpha_{t,s_t} P(s_t \mid y_1..y_{t-1}, x, r_t) \beta_{t+|s_t|-1} + \xi_{r_t}}{P(y \mid x) \partial P(s_t \mid y_1..y_{t-1}, x, r_t)} = \frac{\alpha_{t,r_t} \beta_{t+|s_t|-1}}{\alpha_{|y|+1}}$$

Inside-Outside & Forward-Backward Algorithms are just Backprop

(tutorial paper)

Jason Eisner

JOHNS HOPKINS UNIVERSITY

CENTER FOR LANGUAGE AND SPEECH PROCESSING

"The inside-outside algorithm is the hardest algorithm I know."

– *a senior NLP researcher, in the 1990's*

β α

Can we learn the latent states without supervised training dataset?

Sequence Labeling: Partially Observed

- We have a sequence **x** and **y**, and a joint distribution $p(\mathbf{x}, \mathbf{y} \mid \theta)$

Model parameters

- So far, we had fully observable data $(\mathbf{x}_i, \mathbf{y}_i)$ pairs,

Learning Objective

$$L(\theta) = \sum_{i=1}^{n} \log P(\mathbf{x}_i, \mathbf{y}_i \mid \theta)$$

# of training examples

- What if we only have access to observations x?

$$L(\theta) = \sum_{i} \log P(\mathbf{x_i} \mid \theta)$$

Maximum Likelihood Estimate

- Data points $x_1, x_2, x_3 \ldots x_n$
- Parameter vector θ and a parameter space Ω
- Probability distribution $P(x \mid \theta)$ any θ in Ω.
- Likelihood $P(x_1, x_2, \ldots, x_n \mid \theta) = \prod_{i=1}^{n} P(x_i \mid \theta)$
- Log likelihood: $L(\theta) = \sum_{i=1}^{n} \log P(x_i \mid \theta)$
- Goal: increase the probability of training data!

Expectation Maximization

- Learning objective:

$$L(\theta) = \sum_i \log \sum_{y \in \mathcal{Y}} P(x_i, y \mid \theta)$$

- The EM (Expectation Maximization) algorithm is a method for finding

$$\theta_{MLE} = \arg\max_\theta L(\theta) = \arg\max_\theta \sum_i \log \sum_{y \in \mathcal{Y}} P(x_i, y \mid \theta)$$

- We will look into EM in HMM!

$$p(x_1 \ldots x_n, y_1 \ldots y_n) = q(STOP|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)$$

General Idea

- Initially guess the model parameters θ

- Iterate two steps:

  - Expectation step: Use the current parameters (and observations) to reconstruct hidden states

  - Maximization step: Use that hidden states (and observations) to re-estimate the parameters

EM for HMM

- Maximum Likelihood Parameters (supervised):

$$q_{ML}(y_i|y_{i-1}) = \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} \qquad e_{ML}(x|y) = \frac{c(y, x)}{c(y)}$$

- For unsupervised learning, replace actual count with **expected** counts

- Expected emission counts:

$$\text{(expected) count}(\text{NN} \rightarrow \text{apple}) = \sum_i p(y_i = \text{NN}, x_i = \text{apple}|x_1 \ldots x_n)$$

$$= \sum_{i:x_i=\text{apple}} p(y_i = \text{NN}|x_1 \ldots x_n)$$

EM Intuition

- What we want is... $$p(y_i|x_1...x_n) = \frac{p(x_1...x_n, y_i)}{p(x_1...x_n)}$$
- We can compute: $$\text{(expected) count(NN)} = \sum_i p(y_i = \text{NN}|x_1...x_n)$$
- If we have.... $$p(y_i y_{i+1}|x_1...x_n) = \frac{p(x_1...x_n, y_i, y_{i+1})}{p(x_1...x_n)}$$
- Then we can compute expected transition counts: $$\text{(expected) count(NN} \rightarrow \text{VB)} = \sum_i p(y_i = \text{NN}, y_{i+1} = \text{VB}|x_1...x_n)$$
- Above marginals can be computed from followings: $$p(x_1...x_n, y_i) = \alpha(i, y_i)\beta(i, y_i)$$ $$p(x_1...x_n, y_i, y_{i+1}) = \alpha(i, y_i)q(y_{i+1}|y_i)e(x_{i+1}|y_{i+1})\beta(i+1, y_{i+1})$$

Expectation Maximization

- Initialize transition and emission parameters:
  - random, uniform, or more informed initialization

- **Iterate** until convergence
  - E-step: computing expected counts

$$\text{(expected) count(NN)} = \sum_i p(y_i = \text{NN} | x_1 \dots x_n)$$

$$\text{(expected) count(NN} \rightarrow \text{VB)} = \sum_i p(y_i = \text{NN}, y_{i+1} = \text{VB} | x_1 \dots x_n)$$

$$\text{(expected) count(NN} \rightarrow \text{apple)} = \sum_i p(y_i = \text{NN}, x_i = \text{apple} | x_1 \dots x_n)$$

  - M-step: computing new transition and emission parameters

$$q_{ML}(y_i | y_{i-1}) = \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} \qquad e_{ML}(x|y) = \frac{c(y,x)}{c(y)}$$

- Convergence? Yes. Global Optimum? No.

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B

iterate until convergence

E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(N)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Equivalent to the procedure given in the textbook (J&M Appendix A) – slightly different notations

Toy Example: Ice Cream Climatology

- You are studying global warming. You cannot find records of weather, but you can find records of how much ice cream was consumed each day. Can you estimate the weather history from the ice cream history?
  - Observations (x): Number of ice cream purchase
    - {1, 2, 3}
  - State (y): Weather
    - {C (cold), H (hot)}

Toy Example: Ice Cream Climatology

If today is cold (C) or hot (H), how many cones did I prob. eat?

	P(..|C)	P(..|H)
P(1|..)	0.7	0.1
P(2|..)	0.2	0.2
P(3|..)	0.1	0.7

	P(..|C)	P(..|H)	P(..|start)
P(C|..)	0.8	0.1	0.5
P(H|..)	0.1	0.8	0.5
P(Stop|..)	0.1	0.1	0

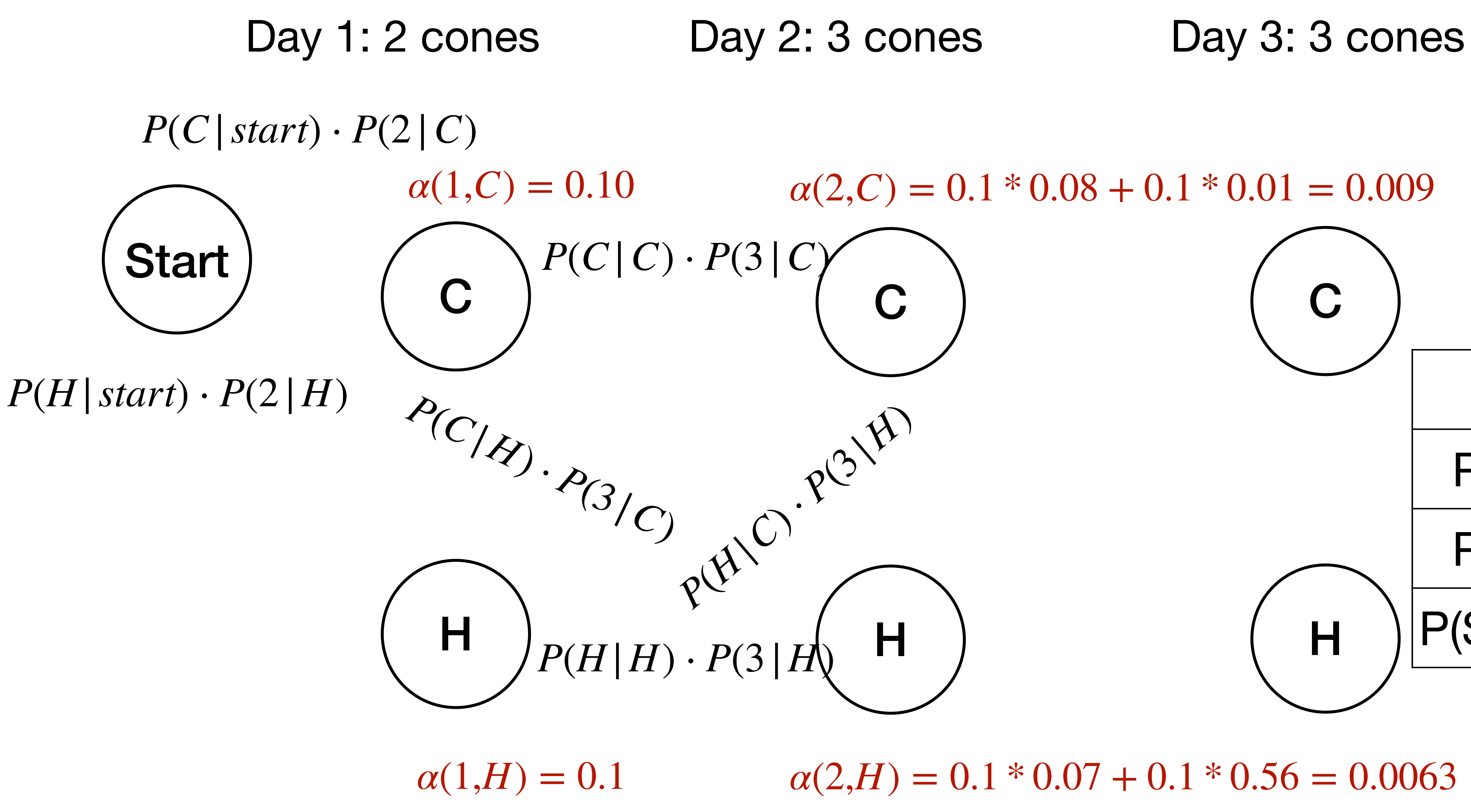
If today is cold (C) or hot (H),what will tomorrow's weather be?

- Maximum Likelihood Parameters (supervised):

$$e_{ML}(x|y) = \frac{c(y,x)}{c(y)} \qquad q_{ML}(y_i|y_{i-1}) = \frac{c(y_{i-1},y_i)}{c(y_{i-1})}$$

- Now, we do not have the weather record (real counts), so we start with guessed probability table, and compute **expected counts**

Toy Example: Ice Cream Climatology



	P(..|C)	P(..|H)
P(1|..)	0.7	0.1
P(2|..)	0.2	0.2
P(3|..)	0.1	0.7

	P(..|C)	P(..|H)	P(..|start)
P(C|..)	0.8	0.1	0.5
P(H|..)	0.1	0.8	0.5
P(Stop|..)	0.1	0.1	0

Day #	Ice Creams	p(□C)	p(□H)
1	2	0.129	0.871
2	3	0.023	0.977
3	3	0.011	0.989
4	2	0.027	0.973
5	3	0.013	0.987
6	2	0.032	0.968
7	3	0.022	0.978
8	2	0.069	0.931
9	2	0.089	0.911
10	3	0.082	0.918
11	1	0.248	0.752
12	3	0.144	0.856
13	3	0.221	0.779
14	1	0.887	0.113
15	1	0.98	0.02
16	1	0.991	0.009
17	2	0.977	0.023
18	1	0.994	0.006
19	1	0.994	0.006
20	1	0.977	0.023
21	3	0.857	0.143
22	1	0.962	0.038
23	2	0.961	0.039
24	1	0.989	0.011
25	1	0.985	0.015
26	1	0.926	0.074
27	2	0.507	0.493
28	3	0.087	0.913
29	3	0.032	0.968
30	2	0.053	0.947
31	3	0.045	0.955
32	2	0.146	0.854
33	2	0.225	0.775
		14.679	**18.321**

- We compute forward and backward, from which we compute P(y_i|**x**) at each day (i).

$$p(x_1 \ldots x_n, y_i) = \alpha(i, y_i)\beta(i, y_i)$$

Toy Example: Ice Cream Climatology

p(□C)	p(□H)
0.129	0.871
0.023	0.977
0.011	0.989
0.027	0.973
0.013	0.987
0.032	0.968
0.022	0.978
0.069	0.931
0.089	0.911
0.082	0.918
0.248	0.752
0.144	0.856
0.221	0.779
0.887	0.113
0.98	0.02
0.991	0.009
0.977	0.023
0.994	0.006
0.994	0.006
0.977	0.023
0.857	0.143
0.962	0.038
0.961	0.039
0.989	0.011
0.985	0.015
0.926	0.074
0.507	0.493
0.087	0.913
0.032	0.968
0.053	0.947
0.045	0.955
0.146	0.854
0.225	0.775
14.679	**18.321**

p(□C,1)	p(□C,2)	p(□C,3)	p(□H,1)	p(□H,2)	p(□H,3)	p(C□C)	p(H□C)	p(C□H)	p(H□H)
0	0.129	0	0	0.871	0	#N/A	#N/A	#N/A	#N/A
0	0	0.023	0	0	0.977	0.021	0.003	0.109	0.868
0	0	0.011	0	0	0.989	0.006	0.005	0.017	0.972
0	0.027	0	0	0.973	0	0.006	0.021	0.005	0.969
0	0	0.013	0	0	0.987	0.007	0.005	0.02	0.968
0	0.032	0	0	0.968	0	0.008	0.024	0.005	0.963
0	0	0.022	0	0	0.978	0.012	0.009	0.02	0.959
0	0.069	0	0	0.931	0	0.017	0.052	0.005	0.927
0	0.089	0	0	0.911	0	0.05	0.038	0.018	0.893
0	0	0.082	0	0	0.918	0.057	0.025	0.031	0.886
0.248	0	0	0.752	0	0	0.077	0.171	0.005	0.747
0	0	0.144	0	0	0.856	0.131	0.013	0.117	0.739
0	0	0.221	0	0	0.779	0.128	0.093	0.016	0.762
0.887	0	0	0.113	0	0	0.221	0.666	0.001	0.113
0.98	0	0	0.02	0	0	0.884	0.095	0.003	0.018
0.991	0	0	0.009	0	0	0.975	0.016	0.005	0.005
0	0.977	0	0	0.023	0	0.973	0.004	0.018	0.005
0.994	0	0	0.006	0	0	0.974	0.019	0.003	0.004
0.994	0	0	0.006	0	0	0.989	0.005	0.005	0.002
0.977	0	0	0.023	0	0	0.974	0.003	0.019	0.004
0	0	0.857	0	0	0.143	0.855	0.002	0.122	0.021
0.962	0	0	0.038	0	0	0.853	0.109	0.004	0.034
0	0.961	0	0	0.039	0	0.944	0.017	0.018	0.021
0.989	0	0	0.011	0	0	0.957	0.032	0.004	0.008
0.985	0	0	0.015	0	0	0.978	0.007	0.011	0.005
0.926	0	0	0.074	0	0	0.924	0.003	0.061	0.012
0	0.507	0	0	0.493	0	0.505	0.001	0.421	0.072
0	0	0.087	0	0	0.913	0.085	0.002	0.421	0.492
0	0	0.032	0	0	0.968	0.026	0.006	0.061	0.907
0	0.053	0	0	0.947	0	0.022	0.031	0.01	0.937
0	0	0.045	0	0	0.955	0.028	0.017	0.025	0.931
0	0.146	0	0	0.854	0	0.04	0.106	0.005	0.849
0	0.225	0	0	0.775	0	0.13	0.095	0.016	0.759
9.931	**3.212**	**1.537**	**1.069**	**7.788**	**9.463**	**12.855**	**1.695**	**1.599**	**15.85**

- With the expected counts for hot and cold days for each day, we compute the following

$$p(x_1 \ldots x_n, y_i) = \alpha(i, y_i)\beta(i, y_i)$$

$$p(x_1 \ldots x_n, y_i, y_{i+1}) = \alpha(i, y_i)q(y_{i+1}|y_i)e(x_{i+1}|y_{i+1})\beta(i+1, y_{i+1})$$

- Use these values that count to re-compute transition probability and emission probability

	P(..|C)	P(..|H)
P(1|..)	0.6765	0.0584
P(2|..)	0.2188	0.4251
P(3|..)	0.1047	0.5165

	P(..|C)	P(..|H)	P(..|start)
P(C|..)	0.8757	0.0925	0.1291
P(H|..)	0.109	0.8652	0.8709
P(Stop|..)	0.0153	0.0423	0

Quiz: p(S1) vs. p(S2)

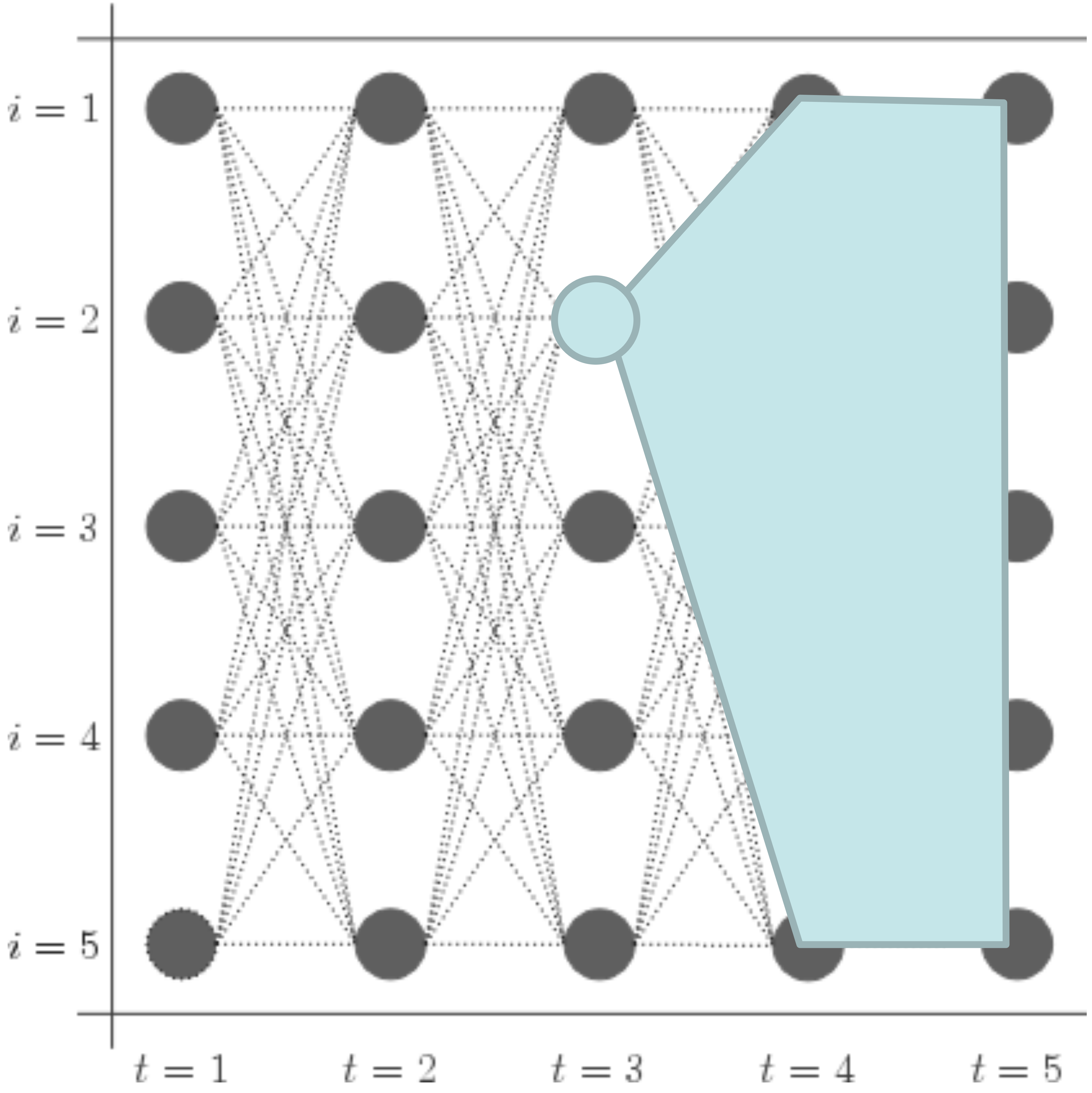
- S1 = Colorless green ideas sleep furiously.
- S2 = Furiously sleep ideas green colorless
  - “It is fair to assume that neither sentence (S1) nor (S2) had ever occurred in an English discourse. Hence, in any statistical model for grammaticalness, these sentences will be ruled out on identical grounds as equally "remote" from English” (Chomsky 1957)

- How would p(S1) and p(S2) compare based on (smoothed) bigram language models?
- How would p(S1) and p(S2) compare based on marginal probability based on POS-tagging HMMs?
  - i.e., marginalized over all possible sequences of POS tags

Summary: Sequence Models

- Sequence Modeling Problems in NLP
- Generative Model: Hidden Markov Models (HMM)
- Discriminative Model:
  Maximum Entropy Markov Models (MEMM)
  Conditional Random Fields
- Unsupervised Learning: Expectation Maximization

Forward-Backward Algorithm



- Initial:

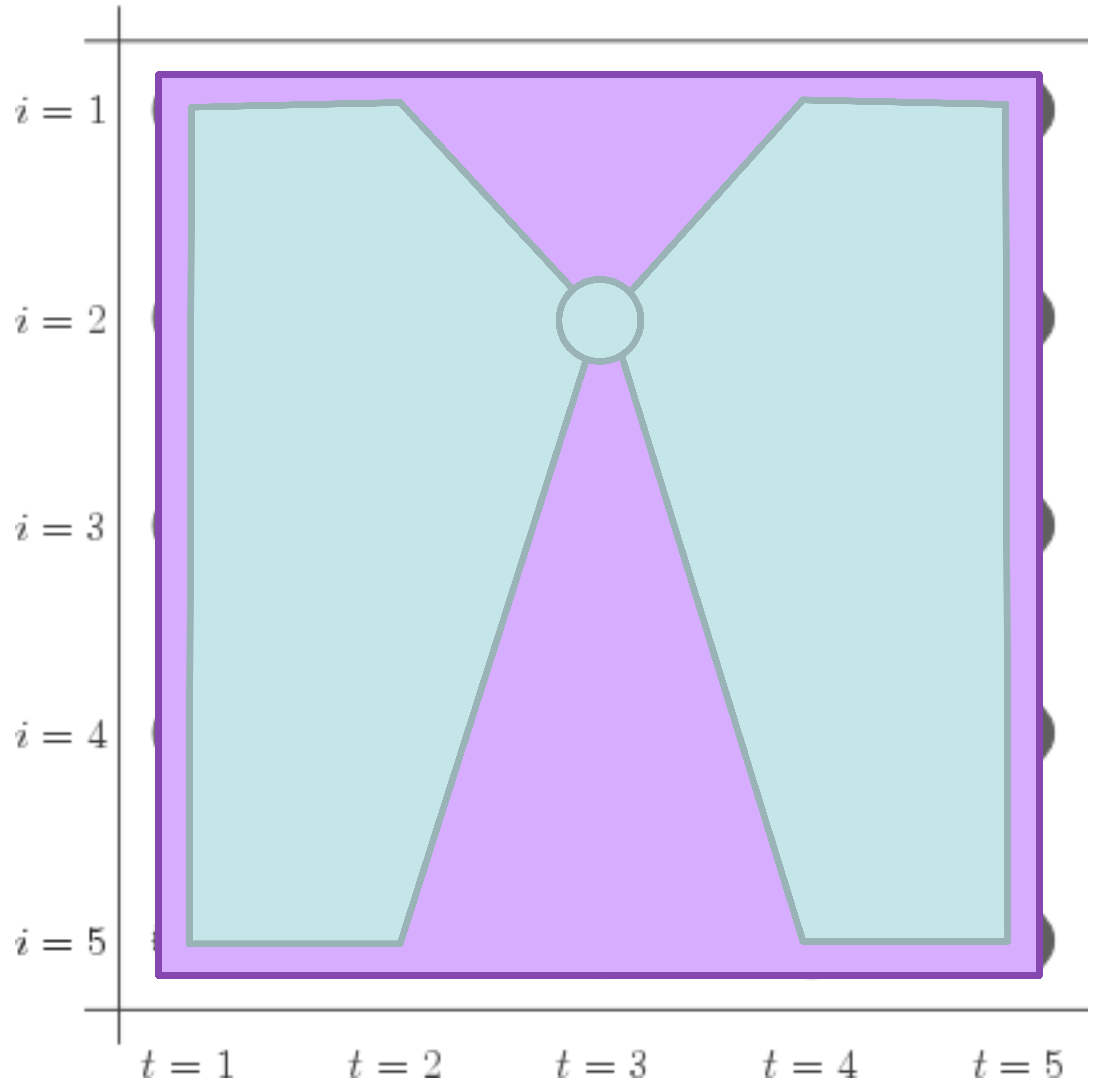
$$\beta_n(s) = 1$$

- Recurrence:

$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \exp(\phi_e(s_{t+1}, t+1, \mathbf{x})) \exp(\phi_t(s_t, s_{t+1}))$$

- Big differences: count emission for the *next* timestep (not current one)

Forward-Backward Algorithm



$$\alpha_1(s) = \exp(\phi_e(s,1,\mathbf{x}))$$

$$\alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1}) \exp(\phi_e(s_t,t,\mathbf{x})) \exp(\phi_t(s_{t-1},s_t))$$

$$\beta_n(s) = 1$$

$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \exp(\phi_e(s_{t+1},t+1,\mathbf{x})) \exp(\phi_t(s_t,s_{t+1}))$$

$$P(s_3 = 2|\mathbf{x}) = \frac{\alpha_3(2)\beta_3(2)}{\sum_i \alpha_3(i)\beta_3(i)}$$

- What is the denominator here? $P(\mathbf{x})$

Forward Backward for HMM

- Two passes: one forward, one back
  - Forward:

$$\alpha(0, y_0) = \begin{cases} 1 \text{ if } y_0 == START \\ 0 \text{ otherwise} \end{cases}$$

    - For i = 1 … n

$$\alpha(i, y_i) = \sum_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \alpha(i-1, y_{i-1})$$

  - Backward:

$$\beta(n, y_n) = \begin{cases} q(y_{n+1}|y_n) \text{ if } y_{n+1} = \text{STOP} \\ 0 \text{ otherwise} \end{cases}$$

    - For i = n-1 … 0

$$\beta(i, y_i) = \sum_{y_{i+1}} e(x_{i+1}|y_{i+1}) q(y_{i+1}|y_i) \beta(i+1, y_{i+1})$$

$$p(x_1 \ldots x_n, y_i) = \alpha(i, y_i) \beta(i, y_i)$$

Other Marginal Inference

- Can we compute this?

$$p(x_1 \ldots x_n) = \sum_{y_i} p(x_1 \ldots x_n, y_i)$$

- Relation with forward quantity?

$$\alpha(i, y_i) = p(x_1 \ldots x_i, y_i) = \sum_{y_1 \ldots y_{i-1}} p(x_1 \ldots x_i, y_1 \ldots y_i)$$

$$p(x_1 \ldots x_n) = \sum_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

$$= \quad \ldots ? \ldots \quad \alpha(n, y_n)$$

$$= \sum_{y_n} q(STOP|y_n)\alpha(n, y_n) \quad := \alpha(n+1, STOP)$$

EM Intuition

- What we want is...

$$p(y_i|x_1...x_n) = \frac{p(x_1...x_n, y_i)}{p(x_1...x_n)}$$

- We can compute:

$$\text{(expected) count(NN)} = \sum_i p(y_i = \text{NN}|x_1...x_n)$$

- If we have....

$$p(y_i y_{i+1}|x_1...x_n) = \frac{p(x_1...x_n, y_i, y_{i+1})}{p(x_1...x_n)}$$

- Then we can compute expected transition counts:

$$\text{(expected) count(NN} \rightarrow \text{VB)} = \sum_i p(y_i = \text{NN}, y_{i+1} = \text{VB}|x_1...x_n)$$

- Above marginals can be computed from followings:

$$p(x_1...x_n, y_i) = \alpha(i, y_i)\beta(i, y_i)$$

$$p(x_1...x_n, y_i, y_{i+1}) = \alpha(i, y_i)q(y_{i+1}|y_i)e(x_{i+1}|y_{i+1})\beta(i+1, y_{i+1})$$

Expectation Maximization

- Initialize transition and emission parameters:
  - random, uniform, or more informed initialization

- **Iterate** until convergence
  - E-step: computing expected counts

$$\text{(expected) count(NN)} = \sum_i p(y_i = \text{NN} | x_1 ... x_n)$$

$$\text{(expected) count(NN} \to \text{VB)} = \sum_i p(y_i = \text{NN}, y_{i+1} = \text{VB} | x_1 ... x_n)$$

$$\text{(expected) count(NN} \to \text{apple)} = \sum_i p(y_i = \text{NN}, x_i = \text{apple} | x_1 ... x_n)$$

  - M-step: computing new transition and emission parameters

$$q_{ML}(y_i | y_{i-1}) = \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} \qquad e_{ML}(x|y) = \frac{c(y,x)}{c(y)}$$

- Convergence? Yes. Global Optimum? No.

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B

iterate until convergence

E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} \;\forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(N)} \;\forall\, t,\, i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Equivalent to the procedure given in the textbook (J&M Appendix A) – slightly different notations

How is this even possible?

- I water the garden everyday
- Saw a weird bug in that garden …
- While I was thinking of an equation …

Noun

S: (n) **garden** (a plot of ground where plants are cultivated)

S: (n) **garden** (the flowers or vegetables or fruits or herbs that are cultivated in a garden)

S: (n) **garden** (a yard or lawn adjoining a house)

Verb

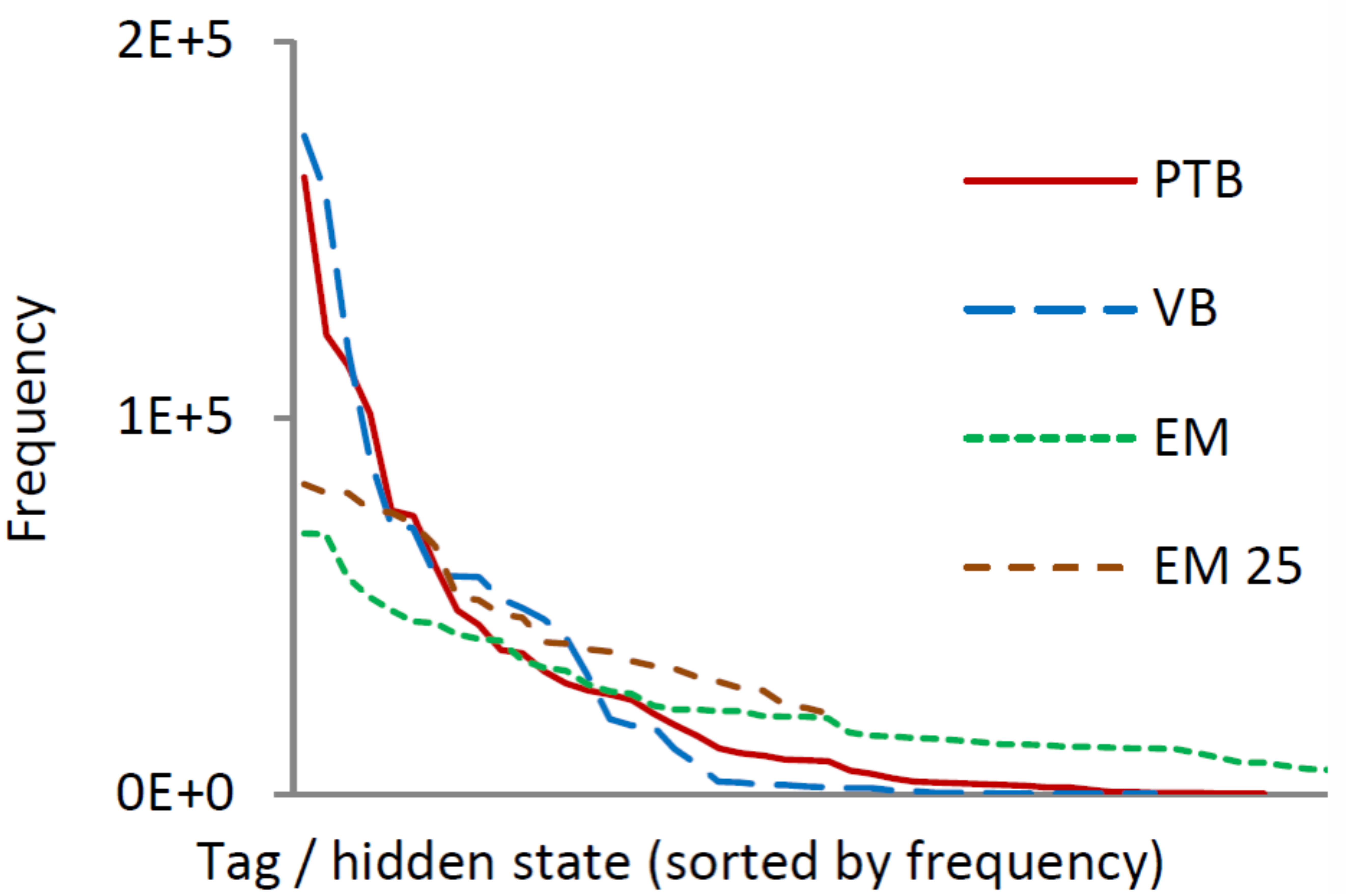
S: (v) **garden** (work in the garden) *"My hobby is gardening"*

Adjective

S: (adj) **garden** (the usual or familiar type) *"it is a common or garden sparrow"*

We can start with a tag dictionary (a list of potential POSs per word)

Does EM learn good POS tagger?



HMMs estimated by EM generally assign a roughly equal number of word tokens to each hidden state, while the empirical distribution of tokens to POS tags is highly skewed

"Why doesn't EM find good HMM POS-taggers", Johnson, EMNLP 2007

HMM vs. CRF

$$P(y_i = s|\mathbf{x}) = \frac{\text{forward}_i(s)\text{backward}_i(s)}{\sum_{s'} \text{forward}_i(s')\text{backward}_i(s')}$$

- Posterior is *derived* from the parameters and the data (conditioned on **x**!)

	$P(x_i|y_i), P(y_i|y_{i-1})$	$P(y_i|\mathbf{x}), P(y_{i-1}, y_i|\mathbf{x})$
HMM	Model parameter (usually multinomial distribution)	Inferred quantity from forward-backward
CRF	Undefined (model is by definition conditioned on **x**)	Inferred quantity from forward-backward

POS Results

- Baseline: assign each word its most frequent tag: ~90% accuracy

- Trigram HMM: ~95% accuracy / 55% on unknown words

- TnT tagger (Brants 1998, tuned HMM): 96.2% accuracy / 86.0% on unks

- MEMM [Ratnaparkhi 1996]: 96.8% accuracy / 86.9% on dunks

- EM for HMM: 74.7%

Quiz: p(S1) vs. p(S2)

- S1 = Colorless green ideas sleep furiously.
- S2 = Furiously sleep ideas green colorless
  - “It is fair to assume that neither sentence (S1) nor (S2) had ever occurred in an English discourse. Hence, in any statistical model for grammaticalness, these sentences will be ruled out on identical grounds as equally "remote" from English” (Chomsky 1957)

- How would p(S1) and p(S2) compare based on (smoothed) bigram language models?
- How would p(S1) and p(S2) compare based on marginal probability based on POS-tagging HMMs?
  - i.e., marginalized over all possible sequences of POS tags

Summary: Sequence Models

- Sequence Modeling Problems in NLP
- Generative Model: Hidden Markov Models (HMM)
- Discriminative Model:
  Maximum Entropy Markov Models (MEMM)
  Conditional Random Fields
- Unsupervised Learning: Expectation Maximization

Summary: Sequence Models

- Generative vs. Discriminative
- Structured or not
  - Independent predictions are effective, but global structure helps
  - But need to be careful about computational overhead
- Model expressivity
  - Expressive feature set can be better
  - But more expensive, overfitting
- The higher accuracy of discriminative models comes at the price of much slower training