



**THE OHIO STATE
UNIVERSITY**

CSE 5525: Foundations of Speech and Language Processing

Conditional Random Fields (CRFs)

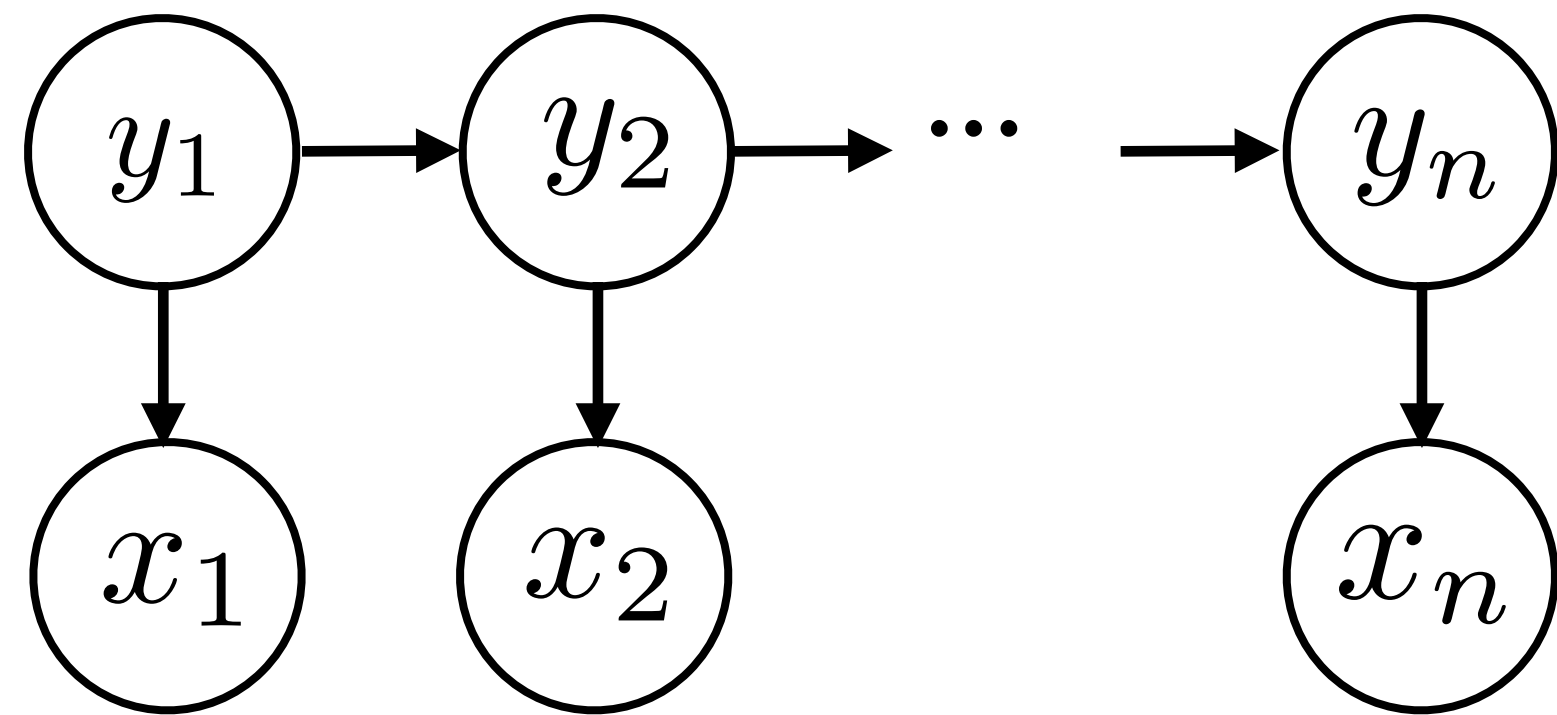
Huan Sun (CSE@OSU)

Many thanks to Prof. Greg Durrett @ UT Austin for sharing his slides.

Recall: HMMs

▶ Observations O (= input \mathbf{x})

Output Q (sequence of states) = labels \mathbf{y}



$$P(\mathbf{y}, \mathbf{x}) = P(y_1) \prod_{i=2}^n P(y_i | y_{i-1}) \prod_{i=1}^n P(x_i | y_i)$$

▶ Training: maximum likelihood estimation (with smoothing)

▶ Inference problem: $\operatorname{argmax}_{\mathbf{y}} P(\mathbf{y} | \mathbf{x}) = \operatorname{argmax}_{\mathbf{y}} \frac{P(\mathbf{y}, \mathbf{x})}{P(\mathbf{x})}$

▶ Viterbi: $\operatorname{score}_i(s) = \max_{y_{i-1}} P(s | y_{i-1}) P(x_i | s) \operatorname{score}_{i-1}(y_{i-1})$

Recall: Viterbi Algorithm

- Initialization

$$v_1(j) = a_{0j} b_j(o_1) \quad 1 \leq j \leq N$$

a_0 : Initial state distribution

a_{ij} : Probability of i - j transition

$b_j(o_t)$: Probability of emitting symbol o_t from state j

- Recursion

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \leq j \leq N, \quad 1 < t \leq T$$

- Termination

$$P^* = v_{T+1}(s_F) = \max_{i=1}^N v_T(i) a_{iF}$$

This only calculates the max. To get final answer (*argmax*),

- keep track of which state corresponds to the max at each step
- build the answer using these back pointers

Viterbi/HMMs: Other Resources

- ▶ Lecture notes from our course website (posted online)
 - ▶ <http://web.cse.ohio-state.edu/~sun.397/courses/au2020/cse5525.html>
- ▶ Eisenstein Chapter 7.3 **but** the notation covers a more general case than what's discussed for HMMs
- ▶ Jurafsky+Martin 8.4.5

This Lecture

- ▶ CRFs: model (+features for NER), inference, learning
- ▶ Named entity recognition (NER)
- ▶ (if time) Beam search

Named Entity Recognition

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG

- ▶ BIO tagset: begin, inside, outside
- ▶ Sequence of tags — should we use an HMM?
- ▶ Why might an HMM not do so well here?
 - ▶ Lots of O's
 - ▶ Insufficient features/capacity with multinomials (especially for unks)

CRFs

Where we're going

- ▶ Flexible discriminative model for tagging tasks that can use **arbitrary features of the input**. Similar to logistic regression, but structured

B-PER I-PER

Barack Obama will travel to *Hangzhou* today for the *G20* meeting .

Curr_word=Barack & **Label=B-PER**

Next_word=Obama & **Label=B-PER**

Curr_word_starts_with_capital=True & **Label=B-PER**

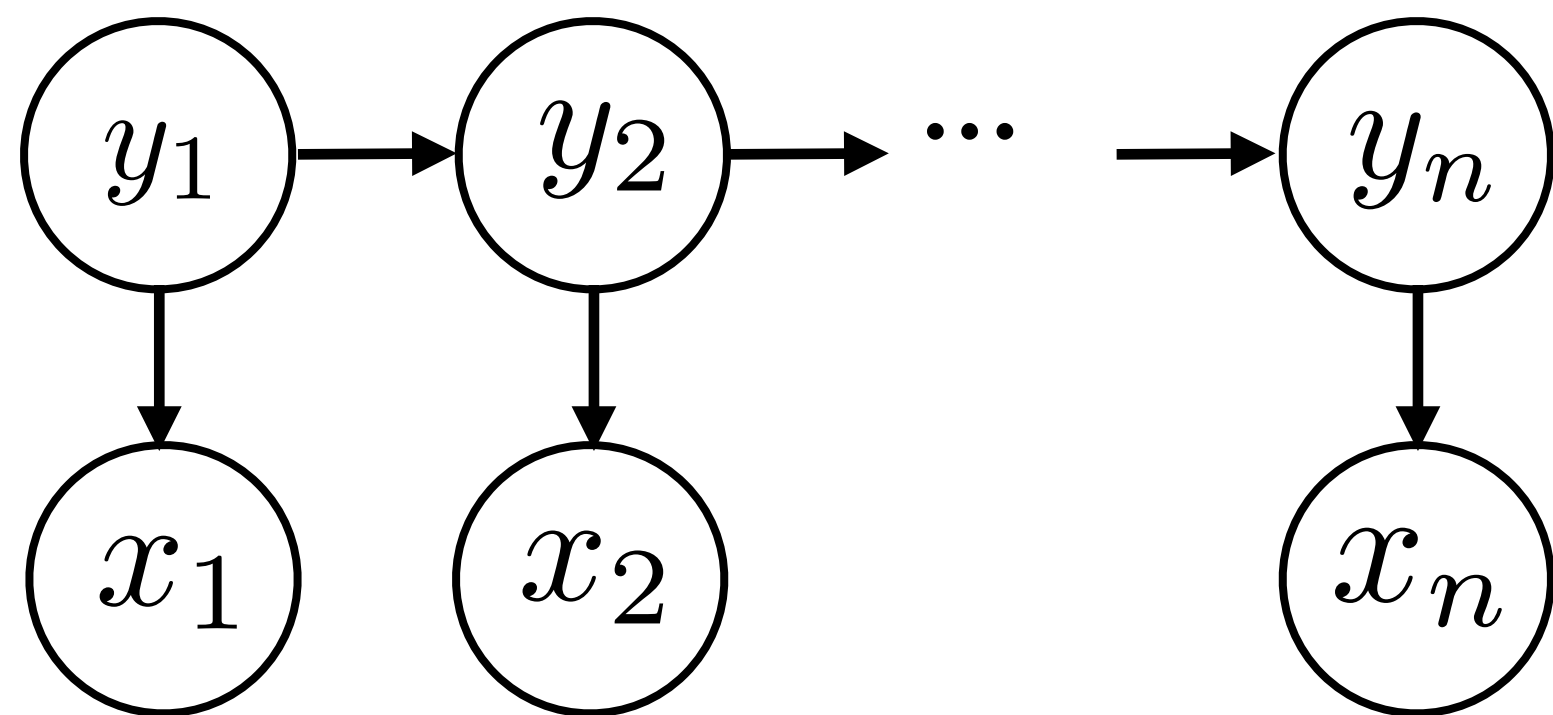
Posn_in_sentence=1st & **Label=B-PER**

Label=B-PER & Next-Label = I-PER

...

HMMs, Formally

- ▶ HMMs are expressible as Bayes nets (factor graphs)



- ▶ This reflects the following decomposition:

$$P(\mathbf{y}, \mathbf{x}) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2) \dots$$

- ▶ Locally normalized model: each factor is a probability distribution that normalizes

Conditional Random Fields

▶ **HMMs:** $P(\mathbf{y}, \mathbf{x}) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2) \dots$

▶ **CRFs:** discriminative models with the following globally-normalized form:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_k \exp(\phi_k(\mathbf{x}, \mathbf{y}))$$

normalizer \uparrow any real-valued scoring function of its arguments

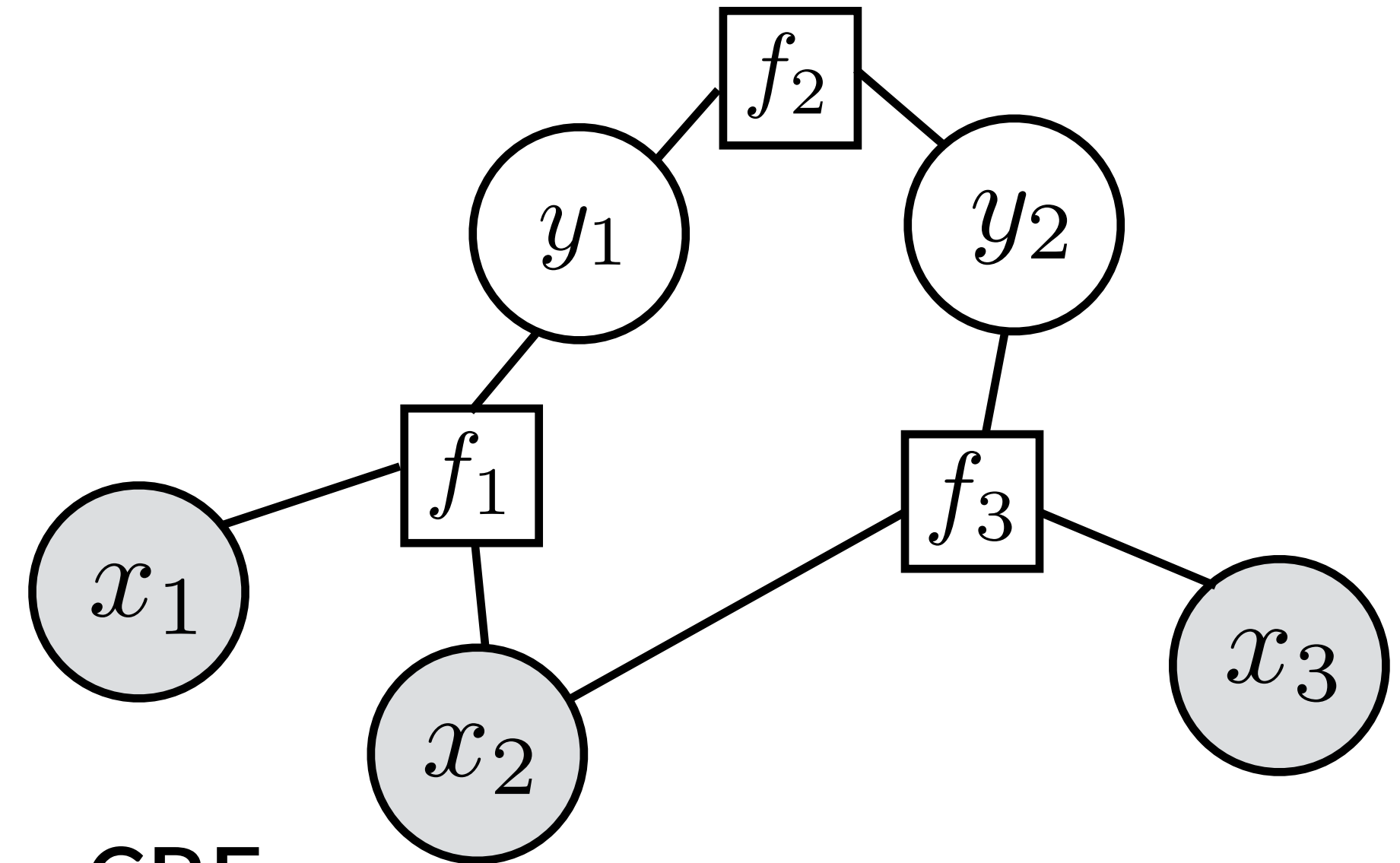
▶ **Special case: linear feature-based potentials** $\phi_k(\mathbf{x}, \mathbf{y}) = w^\top f_k(\mathbf{x}, \mathbf{y})$

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp \left(\sum_{k=1}^n w^\top f_k(\mathbf{x}, \mathbf{y}) \right)$$

▶ Looks like our single weight vector multiclass logistic regression model

HMMs vs. CRFs

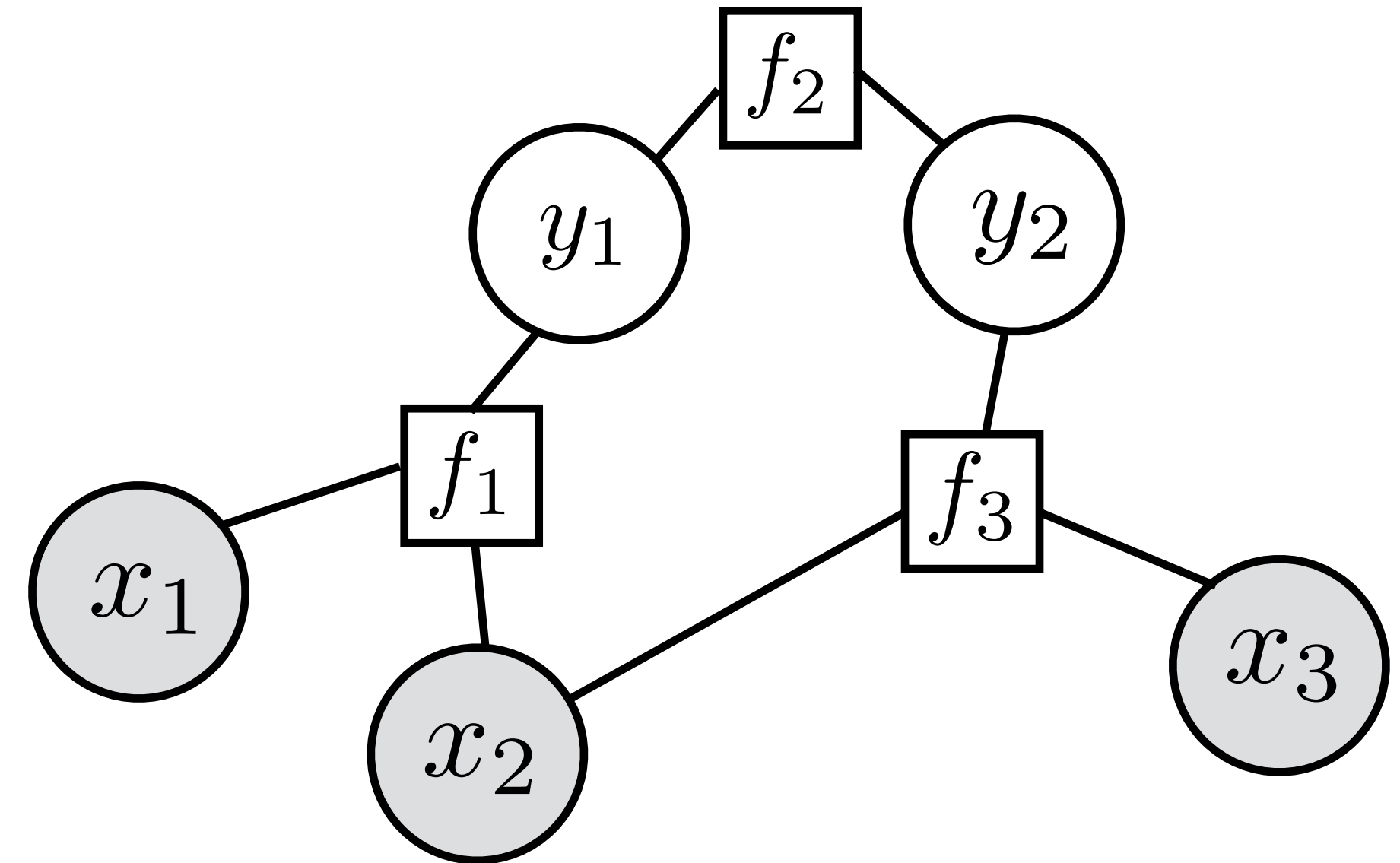
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp \left(\sum_{k=1}^n w^\top f_k(\mathbf{x}, \mathbf{y}) \right)$$



- ▶ Conditional model: x 's are observed
- ▶ Naive Bayes : logistic regression :: HMMs : CRFs
local vs. global normalization \leftrightarrow generative vs. discriminative
(locally normalized discriminative models do exist (MEMMs))
- ▶ HMMs: in the standard setup, emissions consider one word at a time
- ▶ CRFs: features over many words simultaneously, non-independent features (e.g., suffixes and prefixes), doesn't have to be a generative model

Problem with CRFs

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp \left(\sum_{k=1}^n w^\top f_k(\mathbf{x}, \mathbf{y}) \right)$$



- ▶ Normalizing constant

$$Z = \sum_{\mathbf{y}'} \exp \left(\sum_{k=1}^n w^\top f_k(\mathbf{x}, \mathbf{y}') \right)$$

- ▶ Inference: $\mathbf{y}_{\text{best}} = \operatorname{argmax}_{\mathbf{y}'} \exp \left(\sum_{k=1}^n w^\top f_k(\mathbf{x}, \mathbf{y}') \right)$

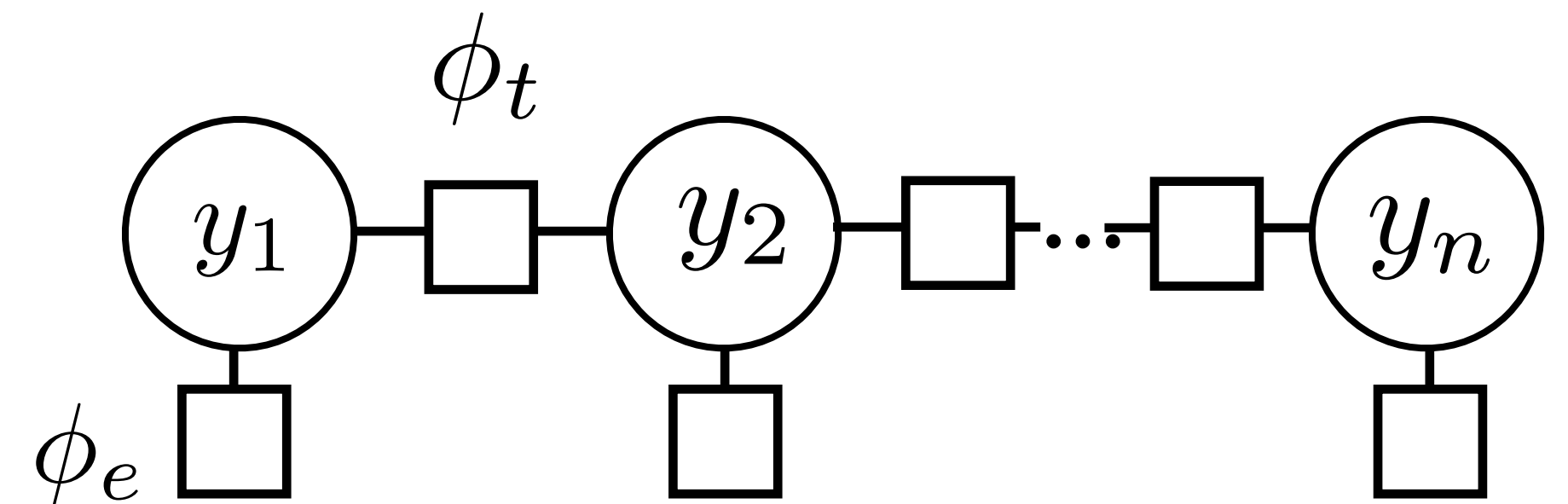
- ▶ If \mathbf{y} consists of 5 variables with 30 values each, how expensive are these?
- ▶ Need to constrain the form of our CRFs to make it tractable

Sequential CRFs

Sequential CRF: (one form)

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

- ▶ Notation: omit \mathbf{x} from the factor graph entirely (implicit), but every feature function connects to it



- ▶ Two types of factors: *transitions* ϕ_t (look at adjacent y 's, but not \mathbf{x}) and *emissions* ϕ_e (look at y and all of \mathbf{x})

Features for NER

Feature Functions

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

The diagram illustrates a Markov chain with nodes y_1, y_2, \dots, y_n represented by circles. Each node y_i is connected to its neighbors y_{i-1} and y_{i+1} by horizontal lines. Below each node y_i is a square node representing a feature function ϕ_e , connected to y_i by a vertical line. The transition function ϕ_t is shown between y_1 and y_2 by a horizontal line above the nodes.

- Phi's are flexible (can be NN with 1B+ parameters). Here: sparse linear fcns
(looks like HW 1 features)

$$\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x}) \quad \phi_t(y_{i-1}, y_i) = w^\top f_t(y_{i-1}, y_i)$$

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$$

Basic Features for NER

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$$

O B-LOC

A diagram illustrating the state transitions and emission for the word 'Hangzhou'. It shows a transition from state 'O' to state 'B-LOC' (indicated by a dashed box) and the emission of the word 'Hangzhou' (indicated by an orange box).

Barack Obama will travel to Hangzhou today for the G20 meeting .

Transitions: $f_t(y_{i-1}, y_i) = \text{Ind}[y_{i-1} \ \& \ y_i] = \text{Ind}[O \text{ — } B\text{-LOC}]$

Emissions: $f_e(y_6, 6, \mathbf{x}) = \text{Ind}[B\text{-LOC} \ \& \ \text{Current word} = \textit{Hangzhou}]$
 $\text{Ind}[B\text{-LOC} \ \& \ \text{Prev word} = \textit{to}]$

Emission Features for NER

LOC

Leicestershire is a nice place to visit...

$$\phi_e(y_i, i, \mathbf{x})$$

PER

Leonardo DiCaprio won an award...

LOC

I took a vacation to **Boston**

ORG

Apple released a new version...

LOC

Texas governor **Greg Abbott** said

PER

ORG

According to the **New York Times**...

Emission Features for NER

- ▶ Word features (can use in HMM)
 - ▶ Capitalization
 - ▶ Word shape
 - ▶ Prefixes/suffixes
 - ▶ Lexical indicators
- ▶ Context features (can't use in HMM!)
 - ▶ Words before/after
 - ▶ Tags before/after
- ▶ Word clusters
- ▶ Gazetteers

Leicestershire

Boston

Apple released a new version...

According to the *New York Times*...

CRFs Outline

► **Model:**
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$$

► Inference

► Learning

Inference and Learning in CRFs

Computing (arg)maxes

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

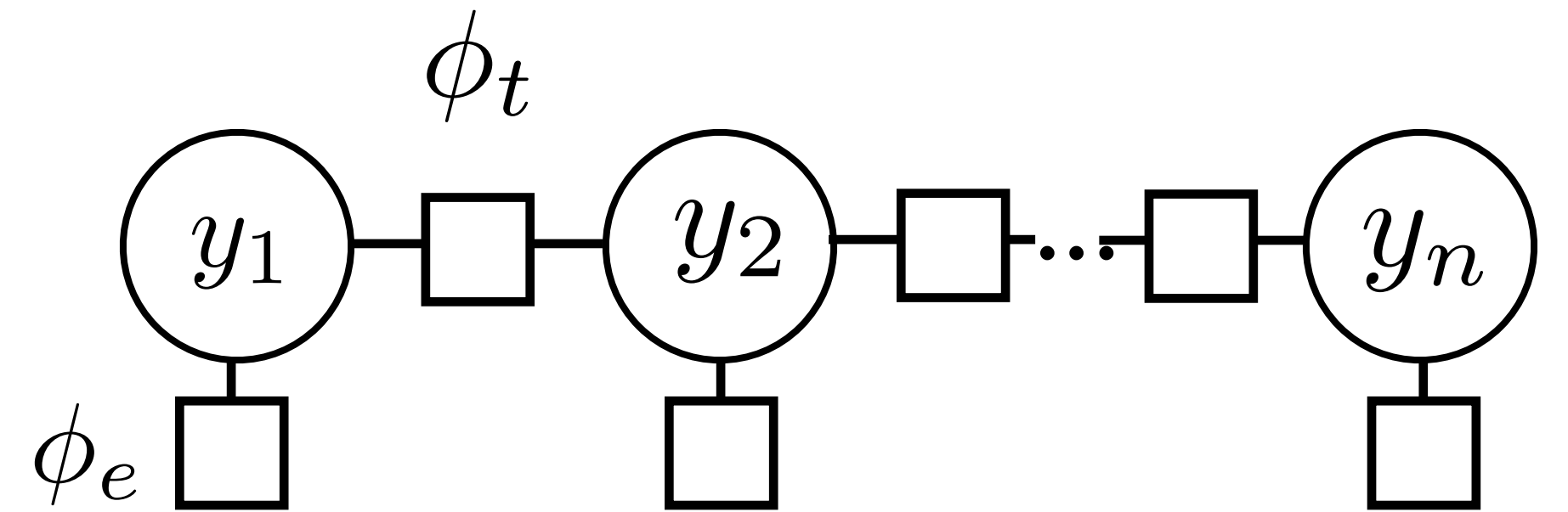
- ▶ $\operatorname{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$: can use Viterbi exactly as in HMM case

$$\begin{aligned} & \max_{y_1, \dots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_e(y_n, n, \mathbf{x})} \dots e^{\phi_e(y_2, 2, \mathbf{x})} e^{\phi_t(y_1, y_2)} e^{\phi_e(y_1, 1, \mathbf{x})} \\ = & \max_{y_2, \dots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_e(y_n, n, \mathbf{x})} \dots e^{\phi_e(y_2, 2, \mathbf{x})} \boxed{\max_{y_1} e^{\phi_t(y_1, y_2)} e^{\phi_e(y_1, 1, \mathbf{x})}} \\ = & \max_{y_3, \dots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_e(y_n, n, \mathbf{x})} \dots \max_{y_2} e^{\phi_t(y_2, y_3)} e^{\phi_e(y_2, 2, \mathbf{x})} \underbrace{\max_{y_1} e^{\phi_t(y_1, y_2)} e^{\phi_e(y_1, 1, \mathbf{x})}}_{\text{score}_1(y_1)} \end{aligned}$$

- ▶ $\exp(\phi_t(y_{i-1}, y_i))$ and $\exp(\phi_e(y_i, i, \mathbf{x}))$ play the role of the Ps now, same dynamic program

Inference in General CRFs

- ▶ Can do efficient inference in any tree-structured CRF



- ▶ Max-product algorithm: generalization of Viterbi to arbitrary tree-structured graphs (sum-product is generalization of forward-backward)

CRFs Outline

► **Model:**
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$$

- Inference: $\operatorname{argmax} P(\mathbf{y}|\mathbf{x})$ from Viterbi
- Learning

Training CRFs

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$$

- ▶ Logistic regression $P(y|x) \propto \exp w^\top f(x, y)$
- ▶ Maximize $\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \log P(\mathbf{y}^* | \mathbf{x})$
- ▶ Gradient is completely analogous to logistic regression:

$$\frac{\partial}{\partial w} \mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=2}^n f_t(y_{i-1}^*, y_i^*) + \sum_{i=1}^n f_e(y_i^*, i, \mathbf{x})$$

intractable! $\rightarrow -\mathbb{E}_{\mathbf{y}} \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$

Training CRFs

$$\frac{\partial}{\partial w} \mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=2}^n f_t(y_{i-1}^*, y_i^*) + \sum_{i=1}^n f_e(y_i^*, i, \mathbf{x}) - \mathbb{E}_{\mathbf{y}} \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$$

- ▶ Let's focus on emission feature expectation

$$\begin{aligned} \mathbb{E}_{\mathbf{y}} \left[\sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right] &= \sum_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y} | \mathbf{x}) \left[\sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right] = \sum_{i=1}^n \sum_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y} | \mathbf{x}) f_e(y_i, i, \mathbf{x}) \\ &= \sum_{i=1}^n \sum_s P(y_i = s | \mathbf{x}) f_e(s, i, \mathbf{x}) \end{aligned}$$

Forward-Backward Algorithm

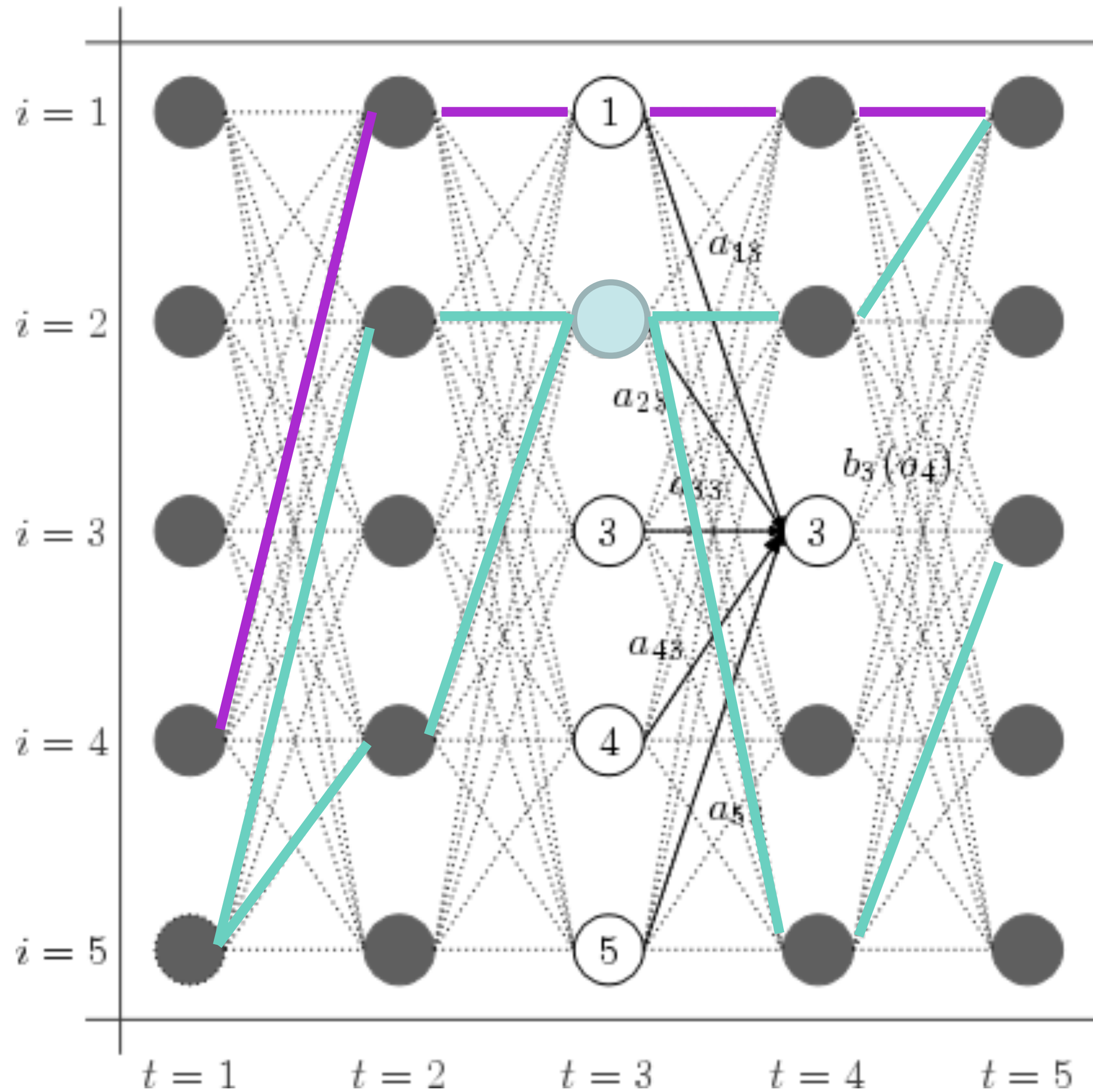
- ▶ How do we compute these marginals $P(y_i = s | \mathbf{x})$?

$$P(y_i = s | \mathbf{x}) = \sum_{y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n} P(\mathbf{y} | \mathbf{x})$$

- ▶ What did Viterbi compute? $P(\mathbf{y}_{\max} | \mathbf{x}) = \max_{y_1, \dots, y_n} P(\mathbf{y} | \mathbf{x})$

- ▶ Can compute marginals with dynamic programming as well using forward-backward

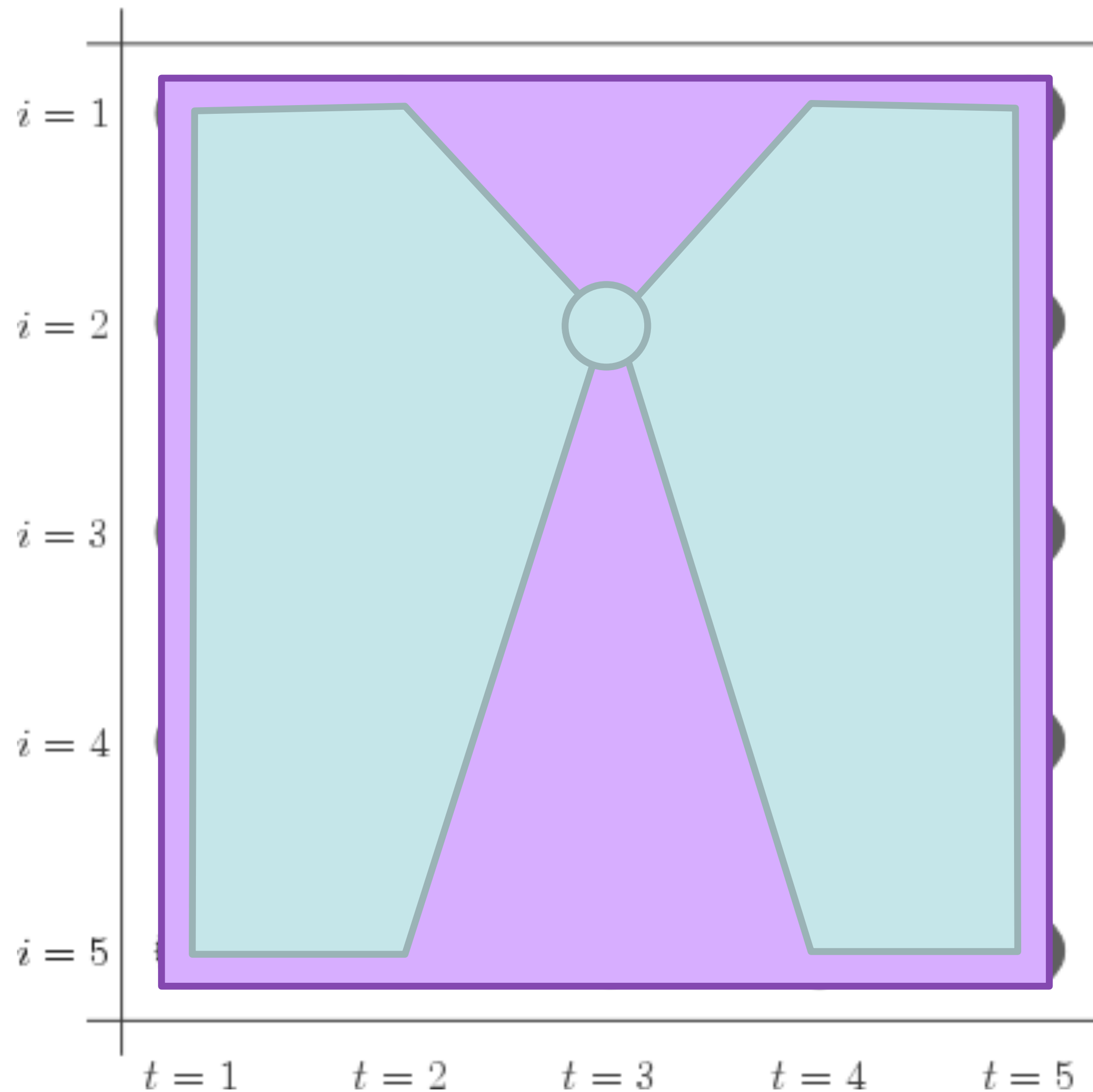
Forward-Backward Algorithm



$$P(y_3 = 2 | \mathbf{x}) =$$

sum of all paths through state 2 at time 3
 sum of all paths


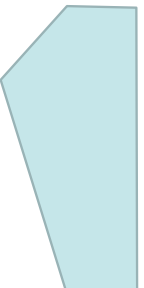
Forward-Backward Algorithm



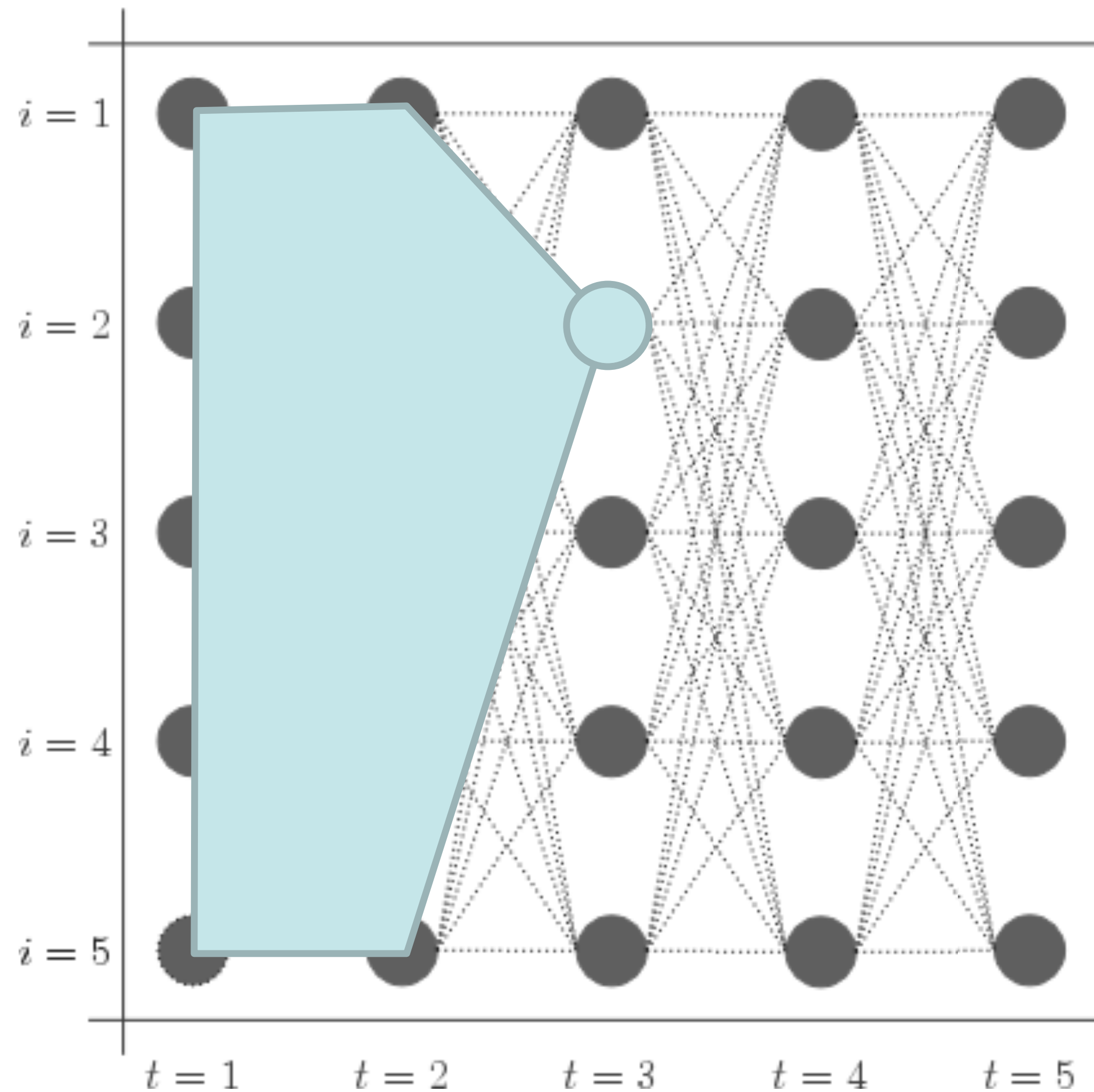
$$P(y_3 = 2 | \mathbf{x}) =$$

sum of all paths through state 2 at time 3
sum of all paths

$$= \frac{\text{[Diagram of forward pass through state 2 at time 3]}}{\text{[Diagram of full forward pass]}}$$

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

Forward-Backward Algorithm



- ▶ **Initial:**

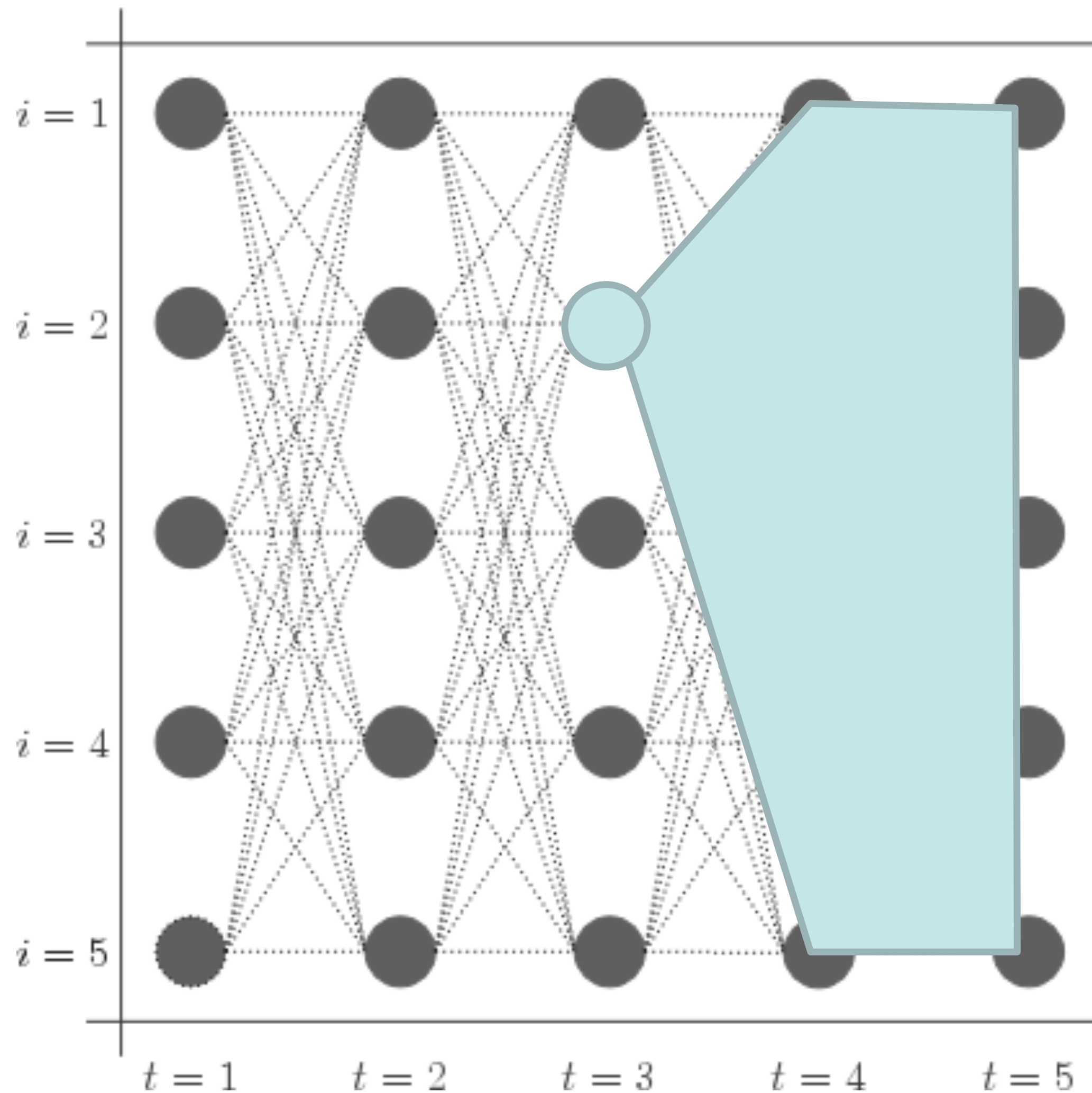
$$\alpha_1(s) = \exp(\phi_e(s, 1, \mathbf{x}))$$

- ▶ **Recurrence:**

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

- ▶ Same as Viterbi but summing instead of maxing!
- ▶ These quantities get very small! Store everything as log probabilities

Forward-Backward Algorithm



► Initial:

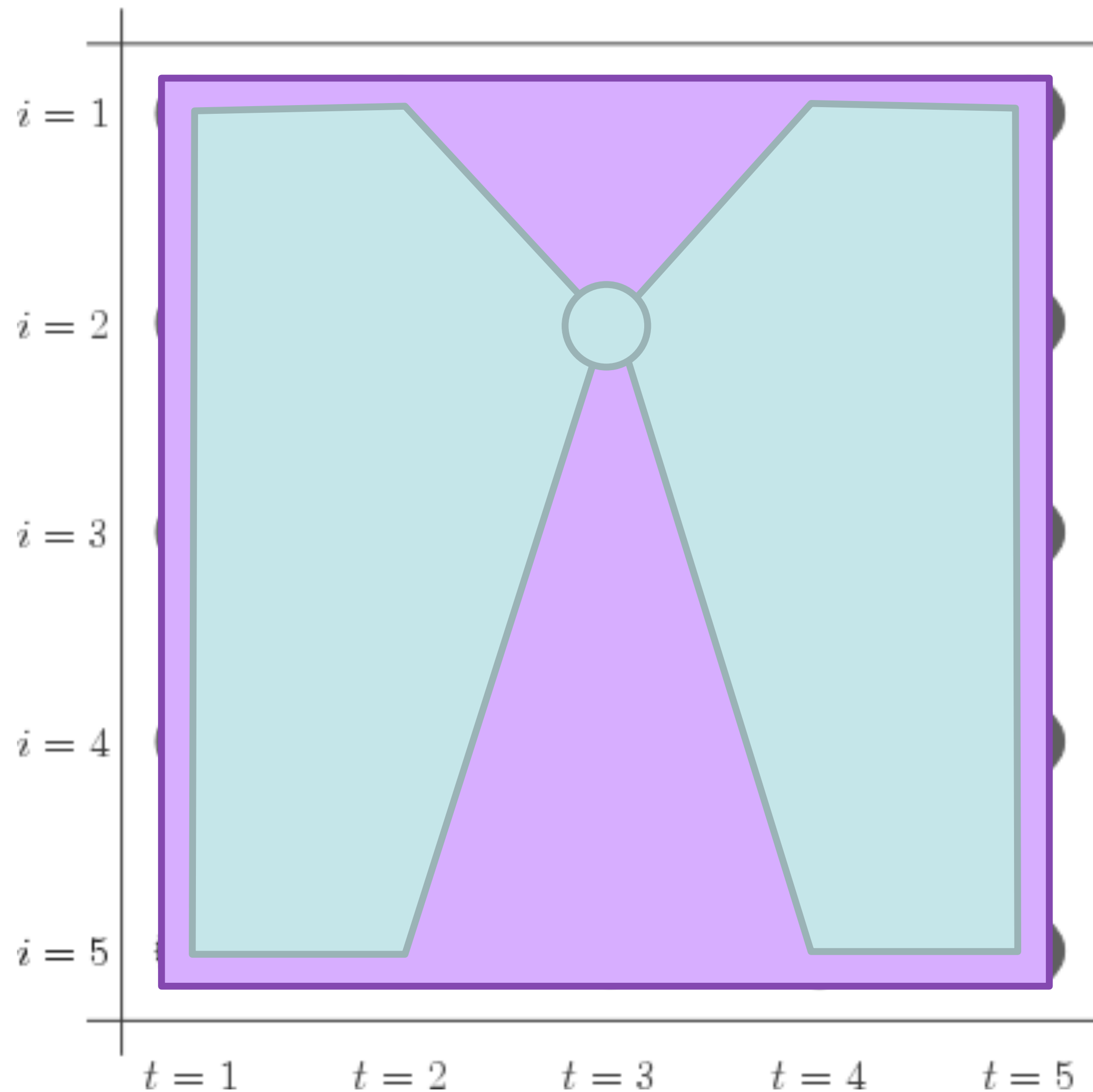
$$\beta_n(s) = 1$$

► 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



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

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

$$\beta_n(s) = 1$$

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

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

- ▶ Does this explain why beta is what it is?
- ▶ What is the denominator here? $P(\mathbf{x})$

Computing Marginals

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

▶ Normalizing constant $Z = \sum_{\mathbf{y}} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$

▶ Analogous to $P(\mathbf{x})$ for HMMs

▶ For both HMMs and CRFs:

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

Z for CRFs, $P(\mathbf{x})$
for HMMs

Posteriors vs. Probabilities

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

- ▶ Posterior is *derived* from the parameters and the data (conditioned on \mathbf{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 \mathbf{x})	Inferred quantity from forward-backward

Training CRFs

- ▶ For emission features:

$$\frac{\partial}{\partial w} \mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=1}^n f_e(y_i^*, i, \mathbf{x}) - \sum_{i=1}^n \sum_s P(y_i = s | \mathbf{x}) f_e(s, i, \mathbf{x})$$

gold features — expected features under model

- ▶ Transition features: need to compute $P(y_i = s_1, y_{i+1} = s_2 | \mathbf{x})$ using forward-backward as well
- ▶ ...but you can build a pretty good system without learned transition features (use heuristic weights, or just enforce constraints like B-PER → I-ORG is illegal)

CRFs Outline

► **Model:**
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$$

► Inference: $\operatorname{argmax} P(\mathbf{y}|\mathbf{x})$ from Viterbi

► Learning: run forward-backward to compute posterior probabilities; then

$$\frac{\partial}{\partial w} \mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=1}^n f_e(y_i^*, i, \mathbf{x}) - \sum_{i=1}^n \sum_s P(y_i = s | \mathbf{x}) f_e(s, i, \mathbf{x})$$

Pseudocode

for each epoch

 for each example

 extract features on each emission and transition (look up in cache)

 compute potentials ϕ based on features + weights

 compute marginal probabilities with forward-backward

 accumulate gradient over all emissions and transitions

Implementation Tips for CRFs

- ▶ Caching is your friend! Cache feature vectors especially
- ▶ Try to reduce redundant computation, e.g. if you compute both the gradient and the objective value, don't rerun the dynamic program
- ▶ Exploit sparsity in feature vectors where possible, especially in feature vectors and gradients
- ▶ Do all dynamic program computation in log space to avoid underflow
- ▶ If things are too slow, run a profiler and see where time is being spent. Forward-backward should take most of the time

Debugging Tips for CRFs

- ▶ Hard to know whether inference, learning, or the model is broken!
- ▶ Compute the objective — is optimization working?
 - ▶ **Inference:** check gradient computation (most likely place for bugs)
 - ▶ Is $\sum \text{forward}_i(s) \text{backward}_i(s)$ the same for all i ?
 - ▶ Do probabilities normalize correctly + look “reasonable”? (Nearly uniform when untrained, then slowly converging to the right thing)
 - ▶ **Learning:** is the objective going down? Try to fit 1 example / 10 examples. Are you applying the gradient correctly?
- ▶ If objective is going down but model performance is bad:
 - ▶ **Inference:** check performance if you decode the training set