

Sequence Models II

Wei Xu

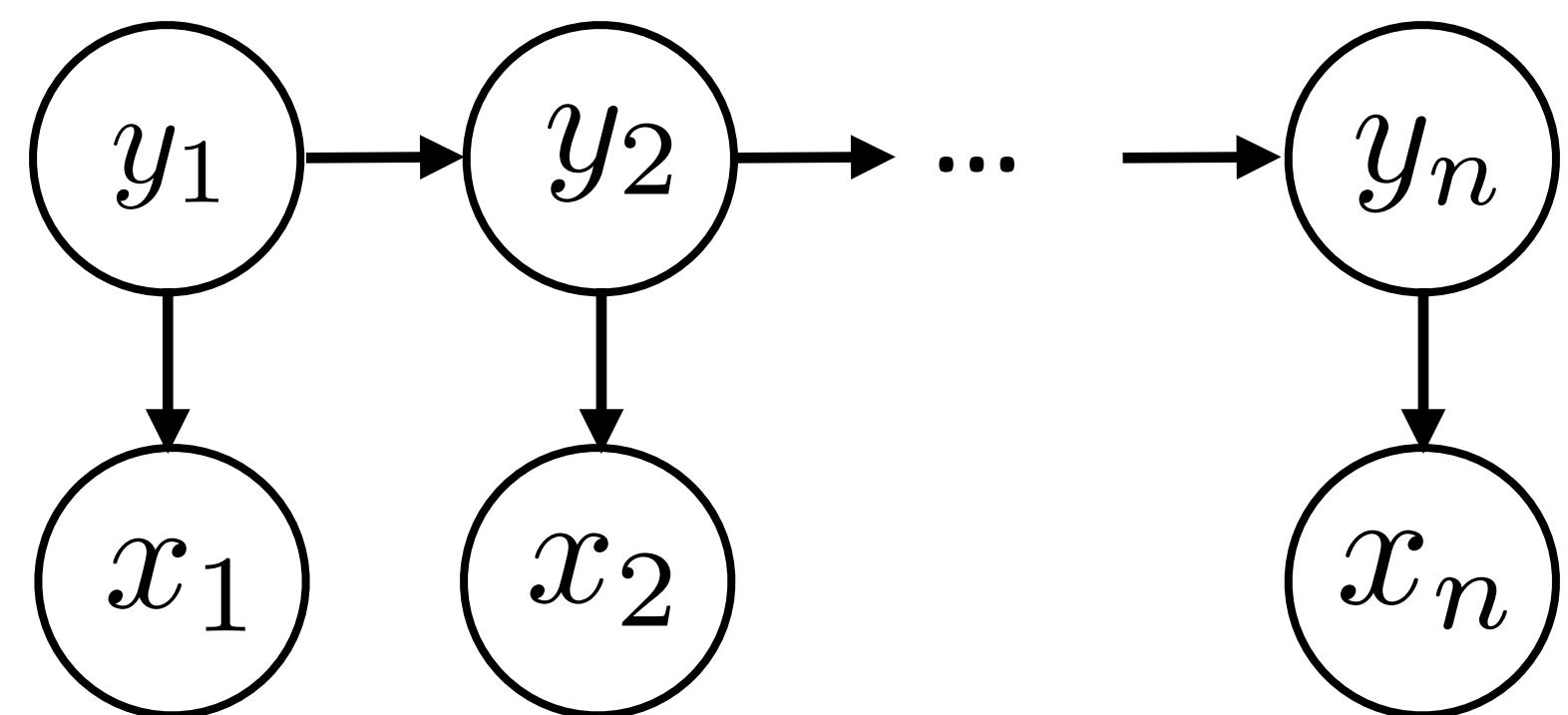
(many slides from Greg Durrett, Dan Klein, Vivek Srikumar, Chris Manning, Yoav Artzi)

Administrivia

- ▶ Homework 2 due next Tuesday
- ▶ Reading: Eisenstein Chapter 7 & 8.3

Recall: HMMs

- ▶ Input $\mathbf{x} = (x_1, \dots, x_n)$ Output $\mathbf{y} = (y_1, \dots, y_n)$



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

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



Andrew Viterbi, 1967

This Lecture

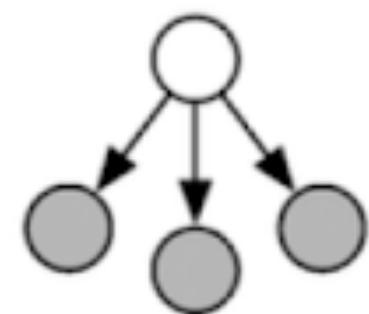
- ▶ CRFs: model (+features for NER), inference, learning
- ▶ Named entity recognition (NER)
- ▶ (if time) Beam search

Named Entity Recognition

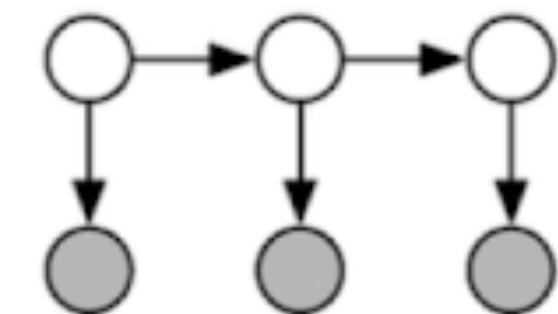
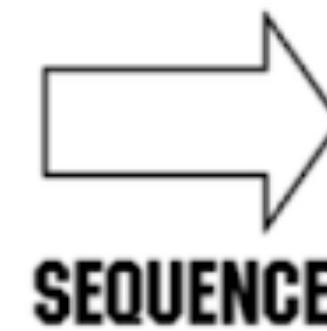
B-PER I-PER O O O B-LOC O O O B-ORG O O
Barack Obama will travel to Hangzhou today for the G20 meeting .

- ▶ 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, so tags aren't as informative about context
 - ▶ Insufficient features/capacity with multinomials (especially for unks)

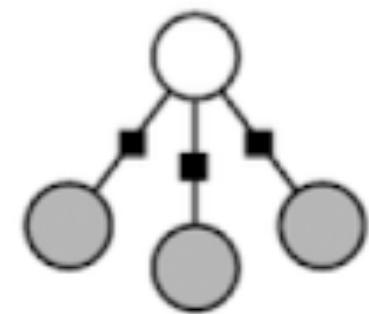
CRFs



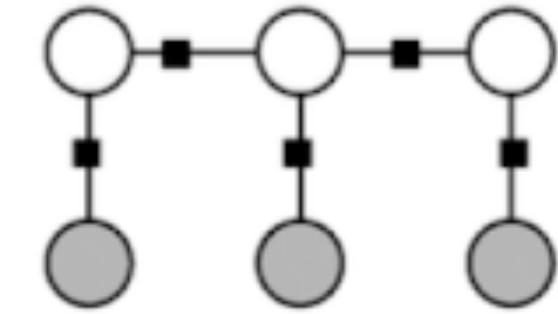
Naive Bayes



HMMs



Logistic Regression



Linear-chain CRFs

Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data

John Lafferty^{†*}

Andrew McCallum^{*,†}

Fernando Pereira^{*,‡}

LAFFERTY@CS.CMU.EDU

MCCALLUM@WHIZBANG.COM

FPEREIRA@WHIZBANG.COM

*WhizBang! Labs—Research, 4616 Henry Street, Pittsburgh, PA 15213 USA

†School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA

‡Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA

Abstract

We present *conditional random fields*, a framework for building probabilistic models to segment and label sequence data. Conditional random fields offer several advantages over hidden Markov models and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Conditional random fields also avoid a fundamental limitation of maximum entropy Markov models (MEMMs) and other discriminative Markov models based on directed graphical models which can be biased towards states

mize the joint likelihood of training examples. To define a joint probability over observation and label sequences, a generative model needs to enumerate all possible observation sequences, typically requiring a representation in which observations are task-appropriate atomic entities, such as words or nucleotides. In particular, it is not practical to represent multiple interacting features or long-range dependencies of the observations, since the inference problem for such models is intractable.

This difficulty is one of the main motivations for looking at conditional models as an alternative. A conditional model specifies the probabilities of possible label sequences given an observation sequence. Therefore it does not expend

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

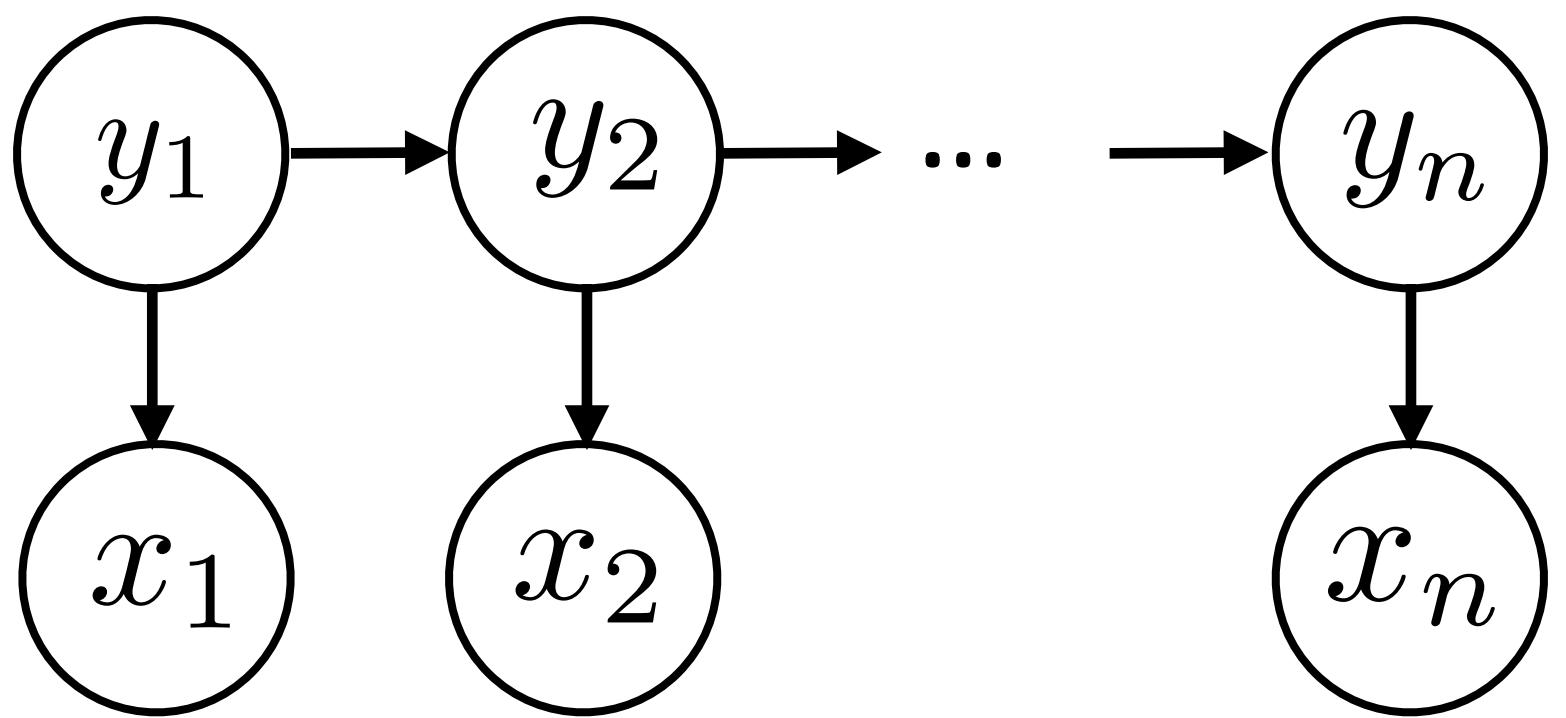
Posn_in_sentence=1st & Label=B-PER

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:

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

↑
any real-valued scoring function of its arguments

normalizer

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

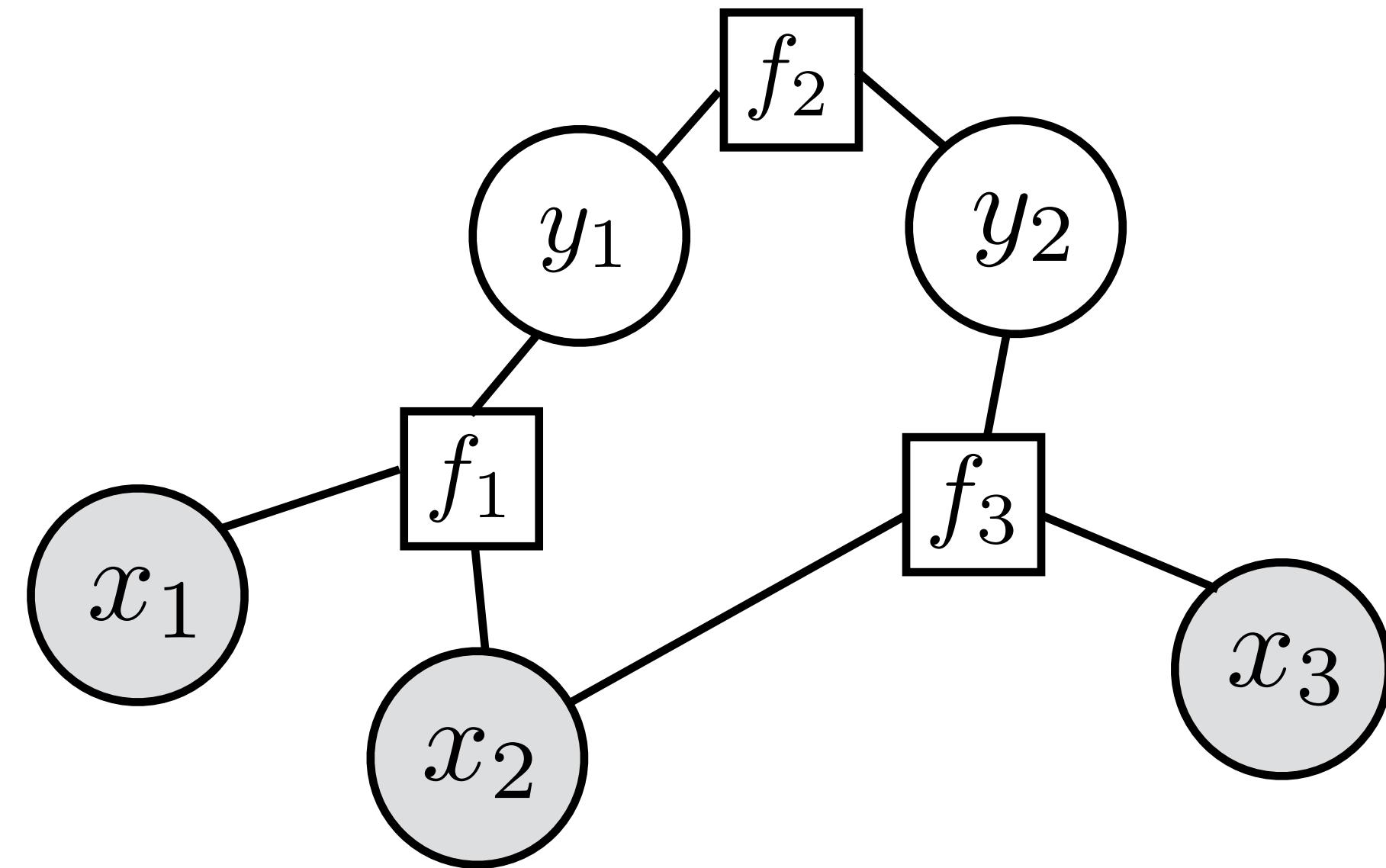
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp \left(\sum_{k=1}^n w^\top f_k(\mathbf{x}, \mathbf{y}) \right)$$

- ▶ 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: \mathbf{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

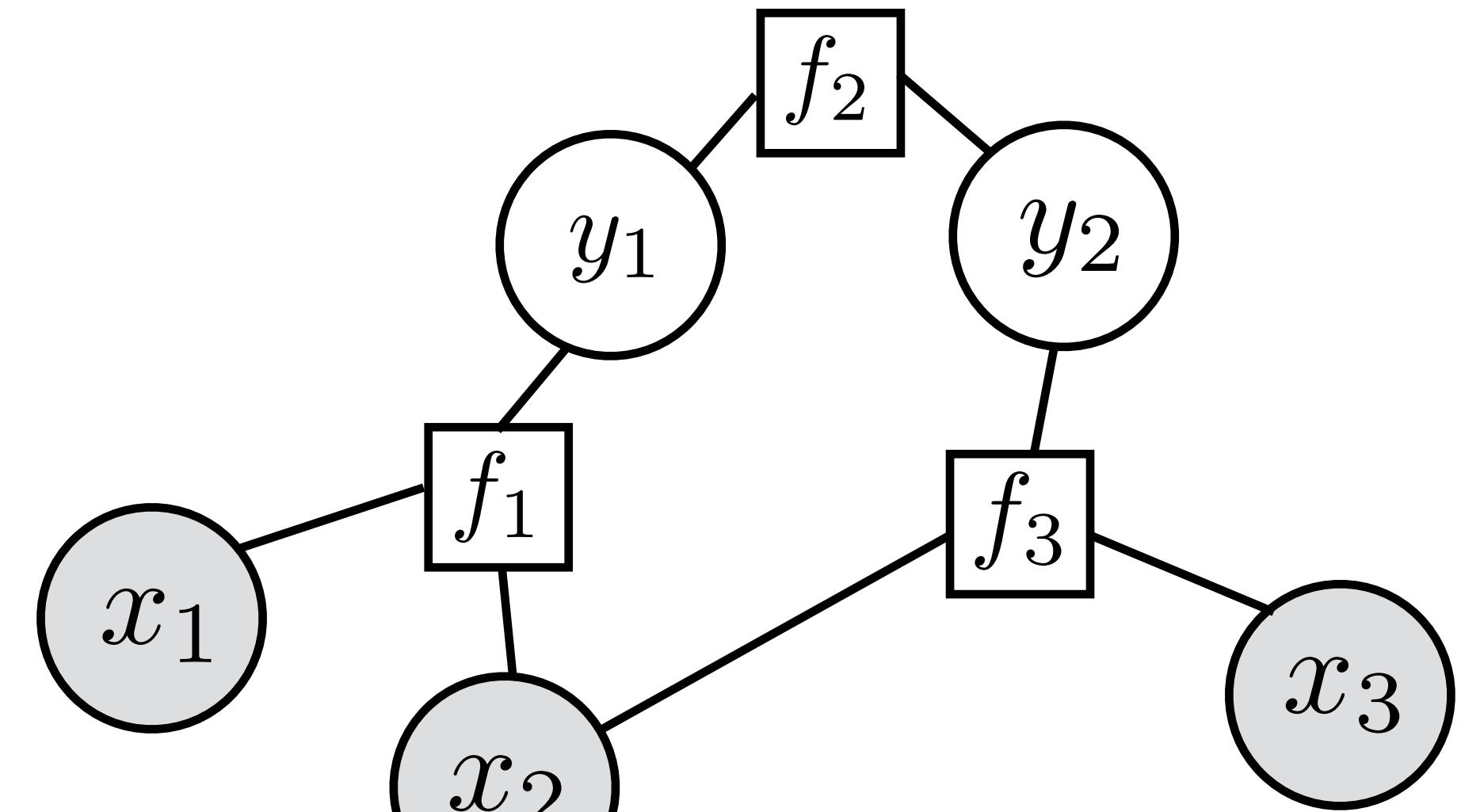
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp \left(\sum_{k=1}^n w^\top f_k(\mathbf{x}, \mathbf{y}) \right)$$

- ▶ Normalizing constant

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

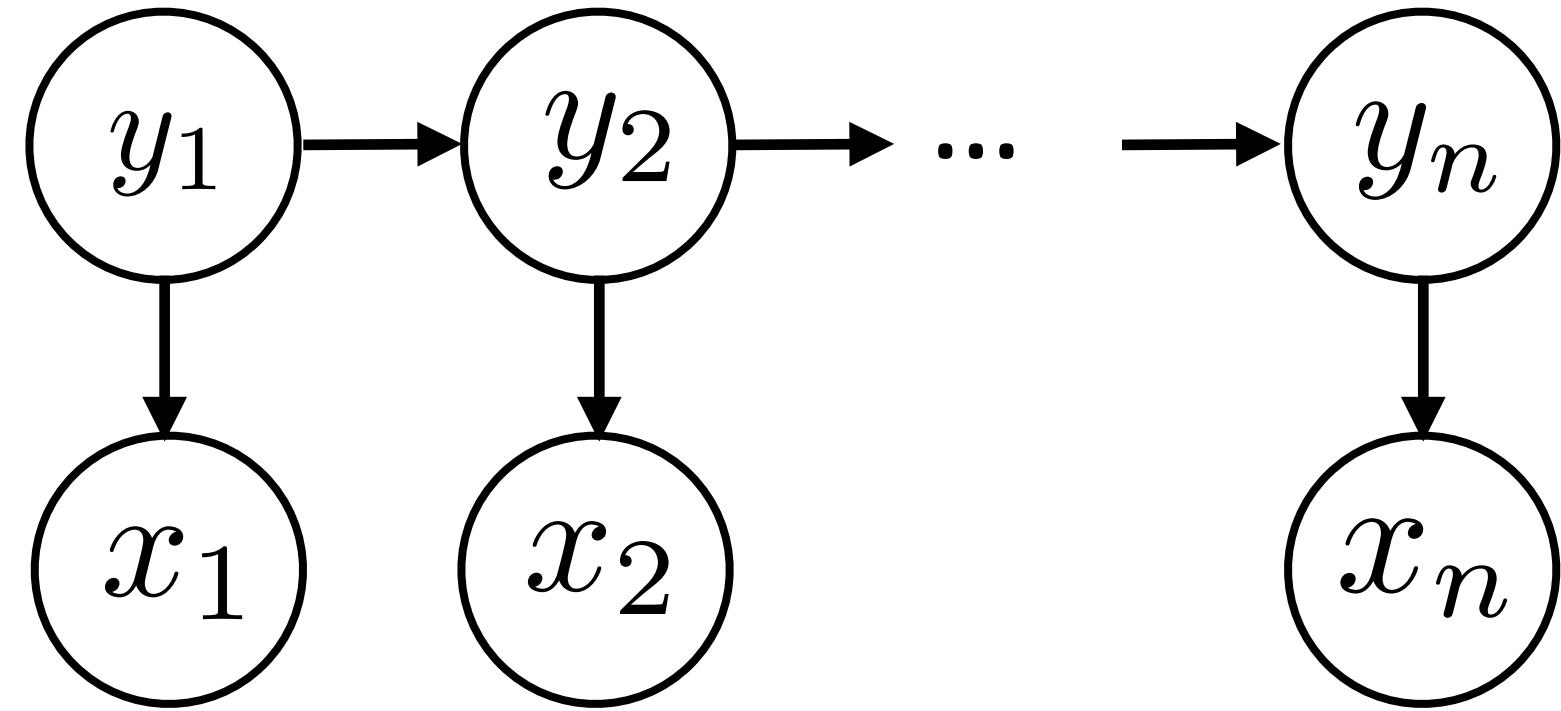
- ▶ Inference: $\mathbf{y}_{\text{best}} = \operatorname{argmax}_{\mathbf{y}'} \exp \left(\sum_{k=1}^n w^\top f_k(\mathbf{x}, \mathbf{y}') \right)$

- ▶ 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



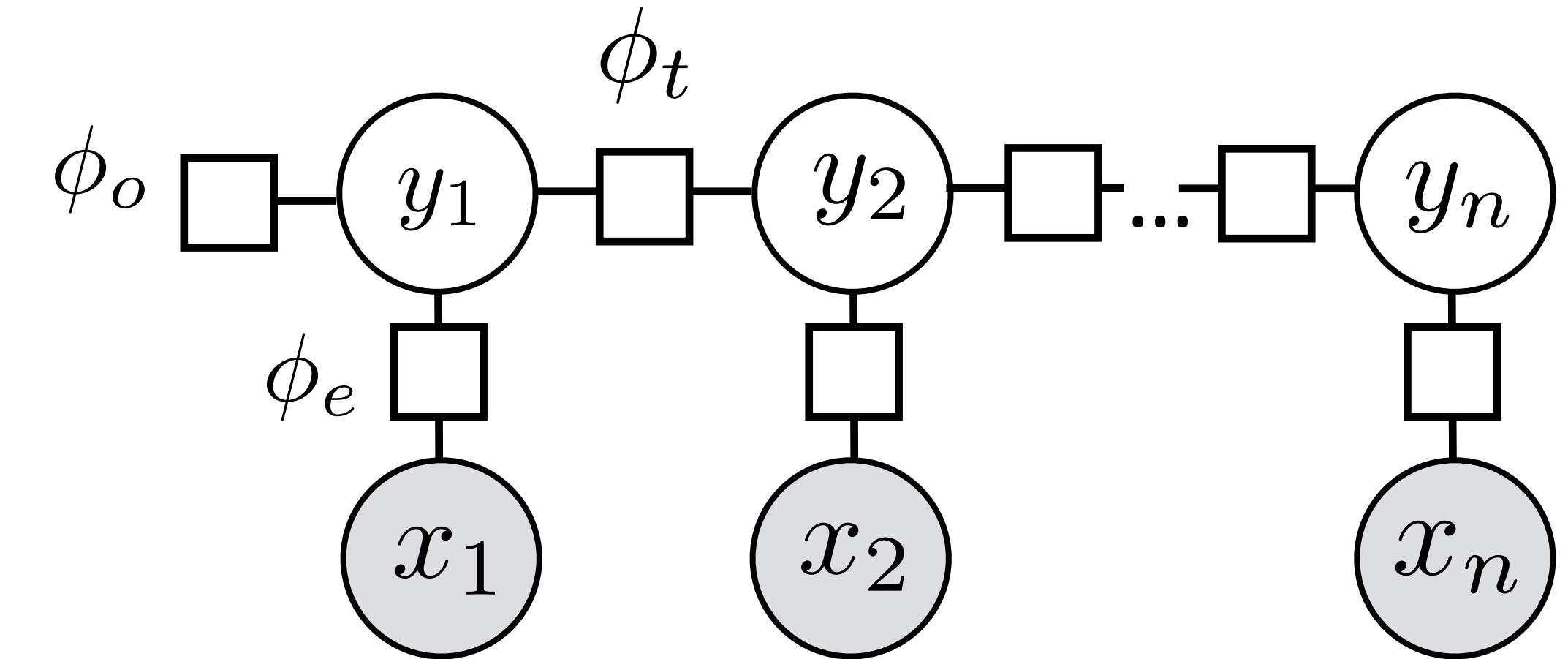
Sequential CRFs

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



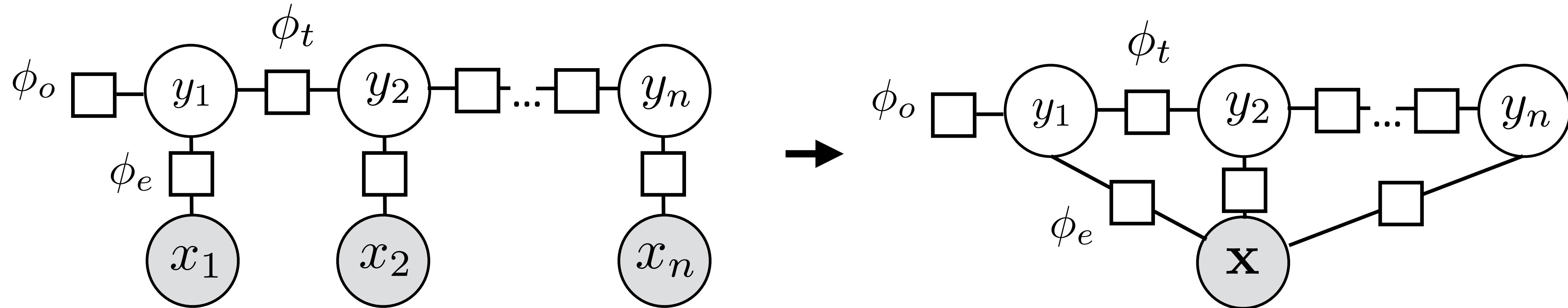
- ▶ CRFs:

$$P(\mathbf{y}|\mathbf{x}) \propto \prod_k \exp(\phi_k(\mathbf{x}, \mathbf{y}))$$



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

Sequential CRFs

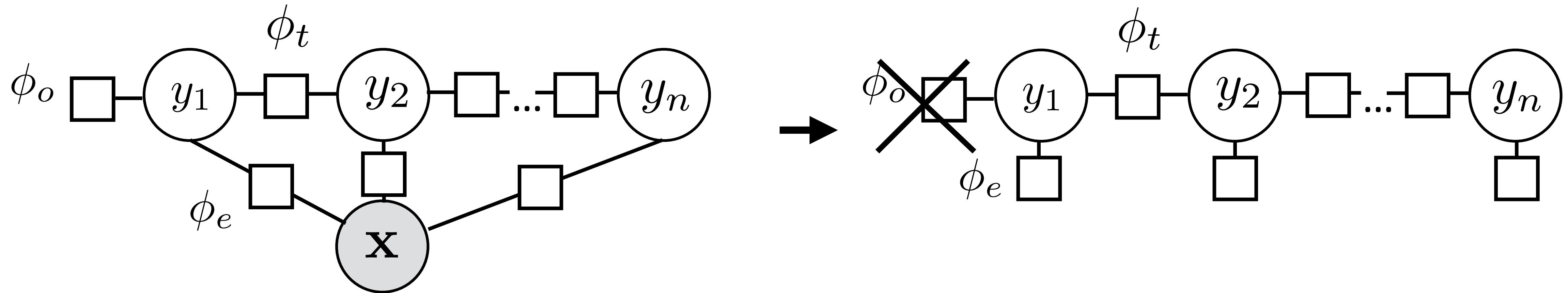


$$P(\mathbf{y}|\mathbf{x}) \propto \exp(\phi_o(y_1)) \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\cancel{\phi_e(x_i, y_i)})$$

$$\prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

- We condition on \mathbf{x} , so every factor can depend on all of \mathbf{x} (including transitions, but we won't do this)
 - \mathbf{y} can't depend arbitrarily on \mathbf{x} in a generative model
- token index — lets us look at current word

Sequential CRFs



- ▶ Notation: omit \mathbf{x} from the factor graph entirely (implicit)
- ▶ Don't include initial distribution, can bake into other factors

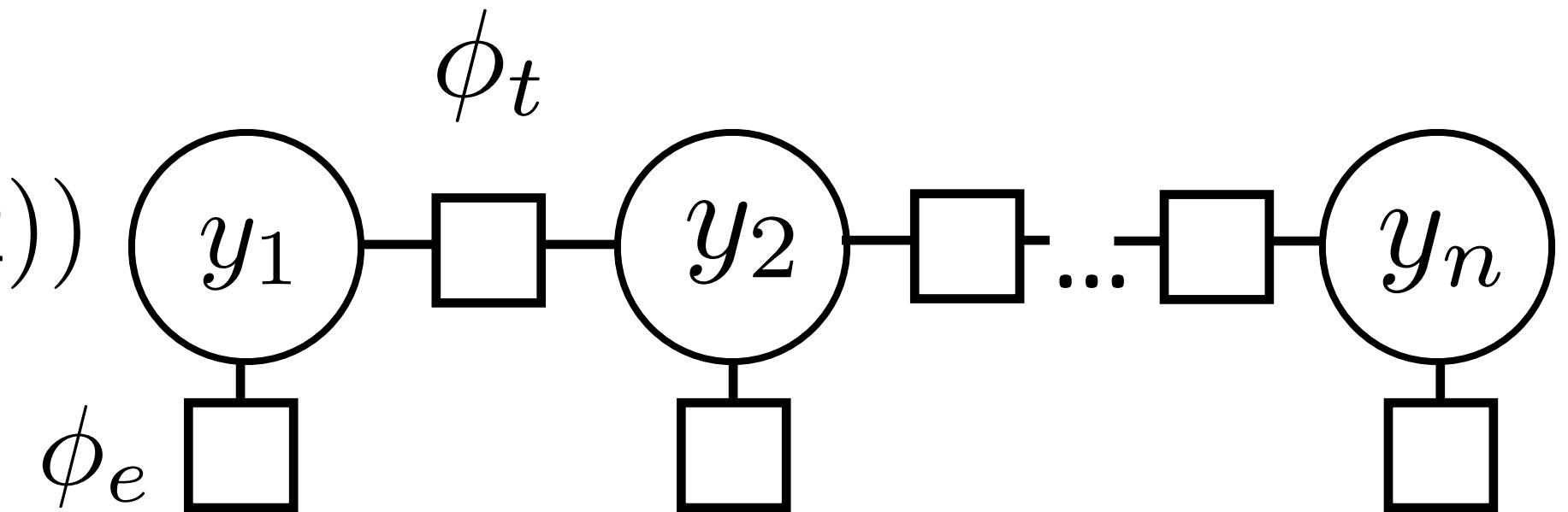
Sequential CRFs:

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

Features for NER

Feature Functions

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



- This can be almost anything! Here we use linear functions of sparse features

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

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

- Looks like our single weight vector multiclass logistic regression model

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

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

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

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

According to the New York Times...

ORG

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

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

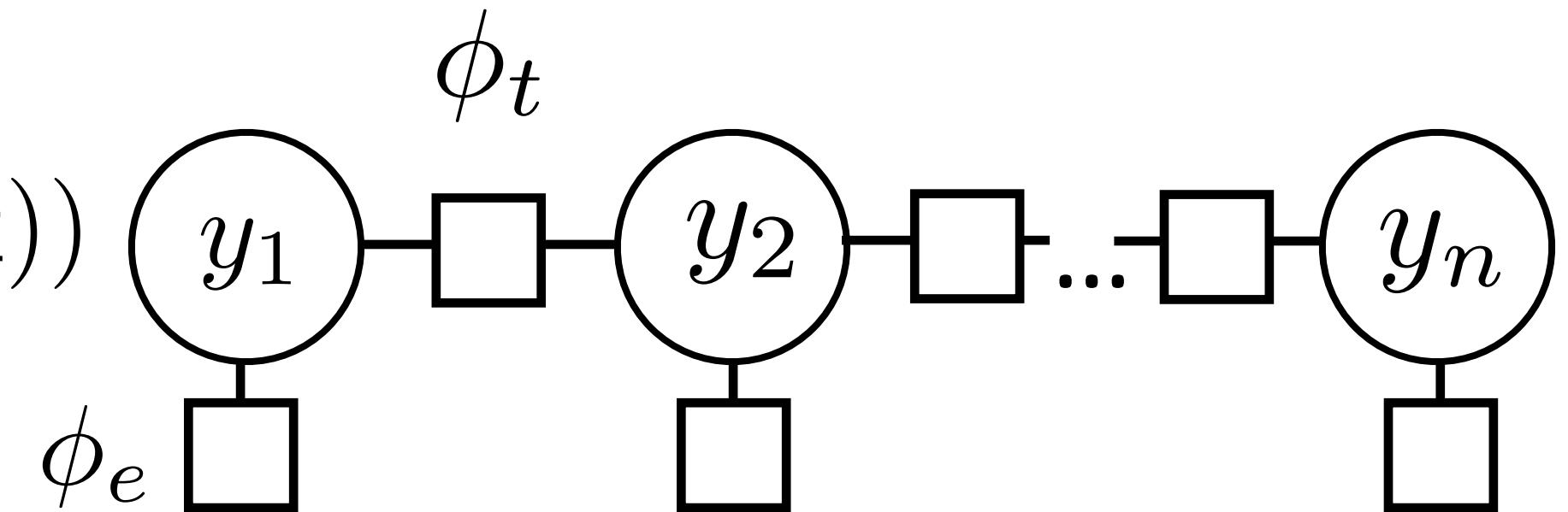
► Inference

► Learning

Inference and Learning in CRFs

Computing (arg)maxes

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



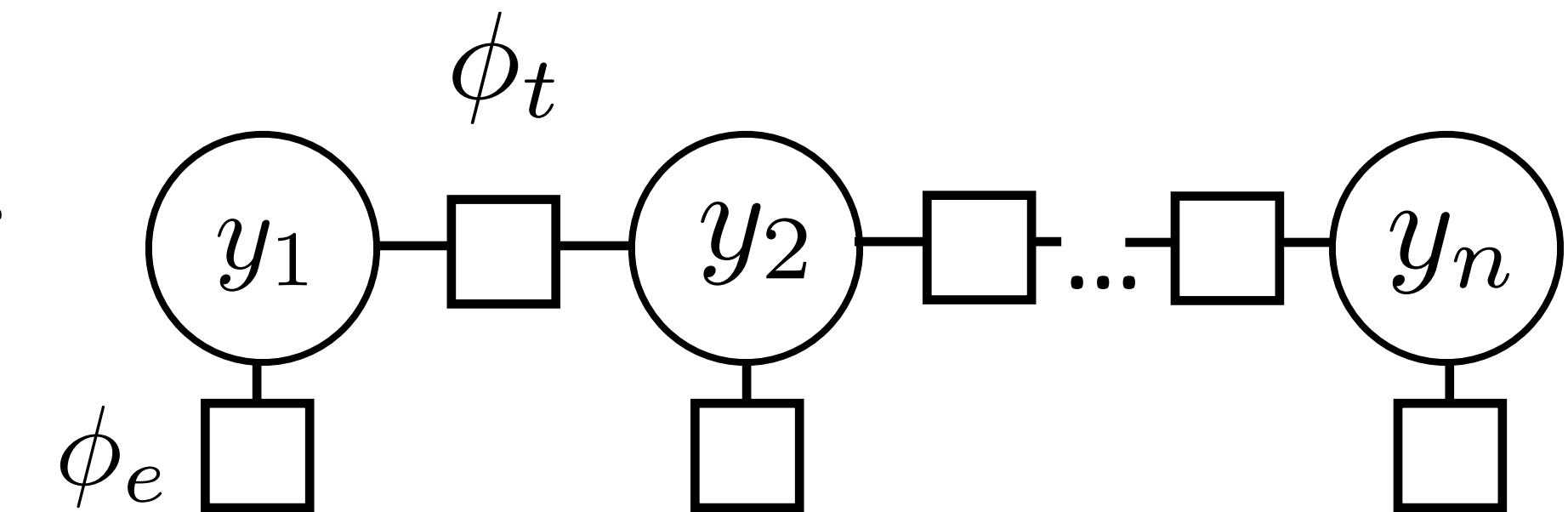
- $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$: can use Viterbi exactly as in HMM case

$$\begin{aligned} & \max_{y_1, \dots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_e(y_n, n, \mathbf{x})} \dots e^{\phi_e(y_2, 2, \mathbf{x})} e^{\phi_t(y_1, y_2)} e^{\phi_e(y_1, 1, \mathbf{x})} \\ &= \max_{y_2, \dots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_e(y_n, n, \mathbf{x})} \dots e^{\phi_e(y_2, 2, \mathbf{x})} \boxed{\max_{y_1} e^{\phi_t(y_1, y_2)} \underbrace{e^{\phi_e(y_1, 1, \mathbf{x})}}_{\text{score}_1(y_1)}} \\ &= \max_{y_3, \dots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_e(y_n, n, \mathbf{x})} \dots \max_{y_2} e^{\phi_t(y_2, y_3)} e^{\phi_e(y_2, 2, \mathbf{x})} \underbrace{\max_{y_1} e^{\phi_t(y_1, y_2)} \text{score}_1(y_1)}_{\text{score}_2(y_2)} \end{aligned}$$

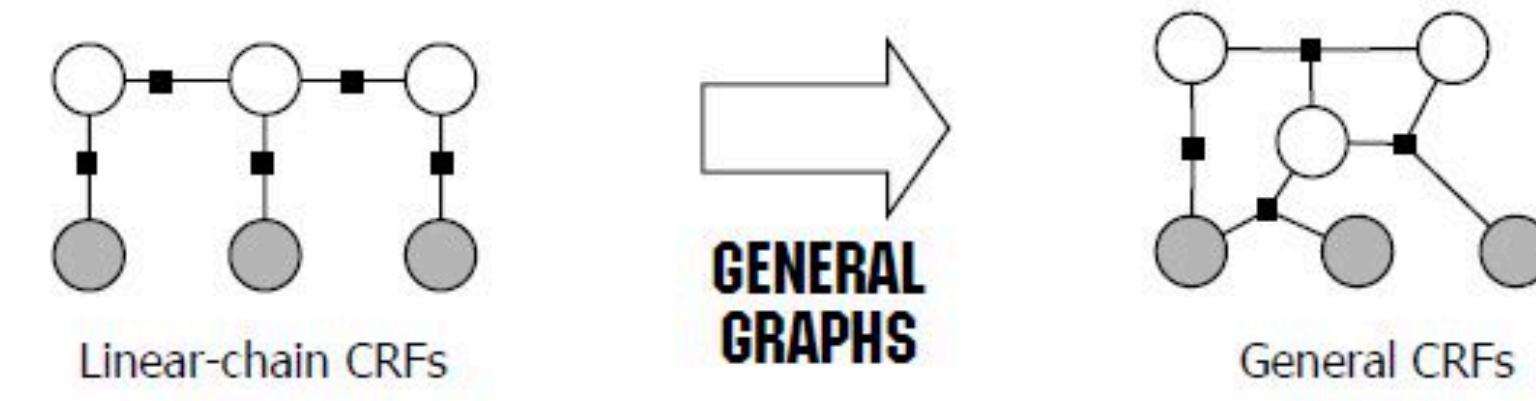
- $\exp(\phi_t(y_{i-1}, y_i))$ and $\exp(\phi_e(y_i, i, \mathbf{x}))$ play the role of the Ps now, same dynamic program

Inference in General CRFs

- ▶ Can do 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

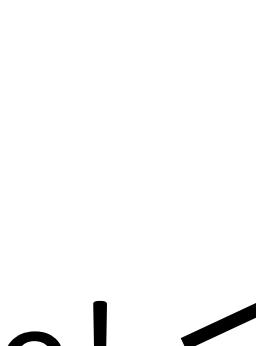
- Model: $P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$
$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$$
- Inference: argmax $P(\mathbf{y}|\mathbf{x})$ 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!  $-\mathbb{E}_{\mathbf{y}} \left[\sum_{i=2}^n f_t(y_{i-1}, y_i) + \sum_{i=1}^n f_e(y_i, i, \mathbf{x}) \right]$

Training CRFs

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

► Let's focus on emission feature expectation

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

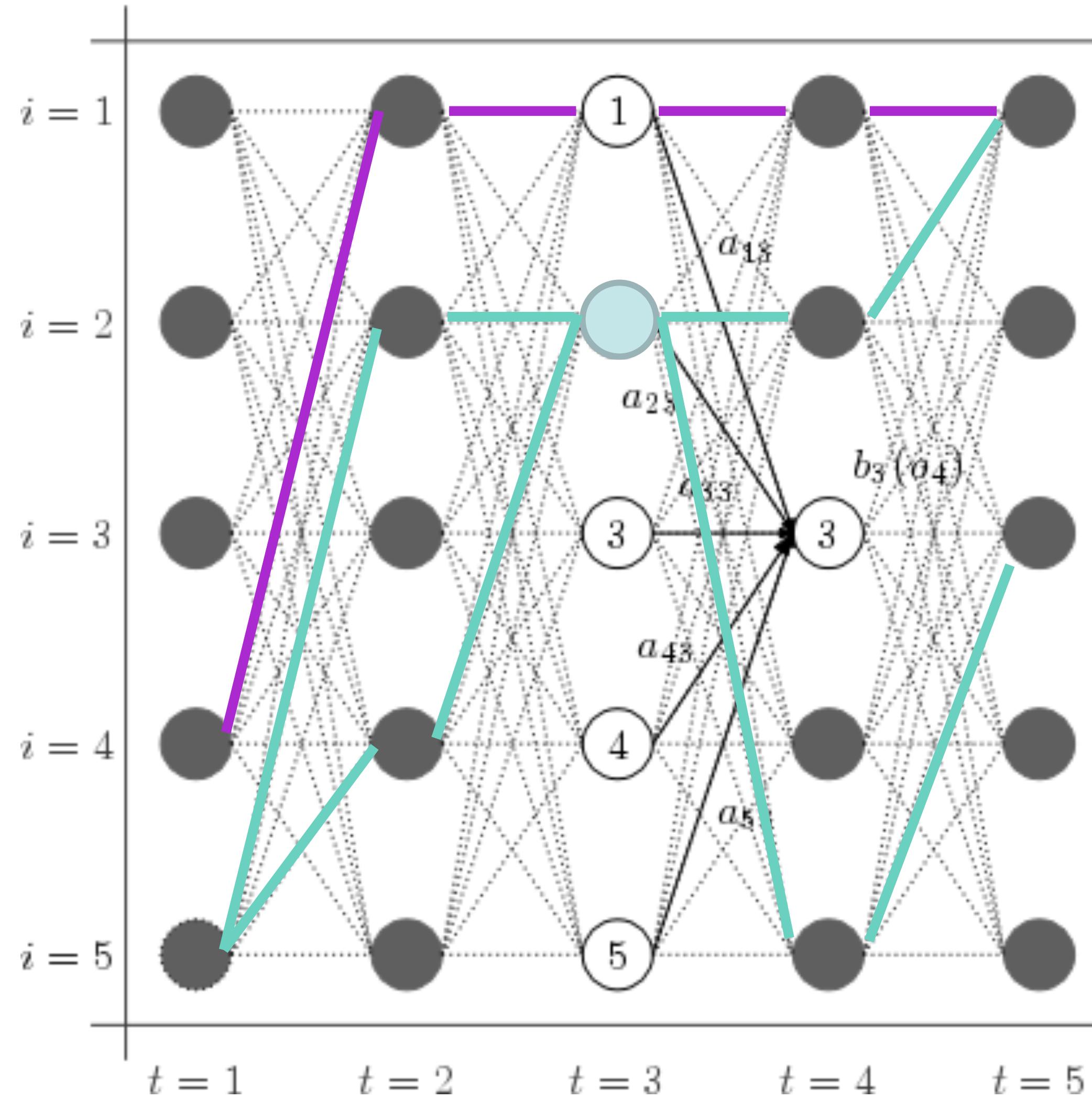
Forward-Backward Algorithm

- ▶ 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 an algorithm called forward-backward

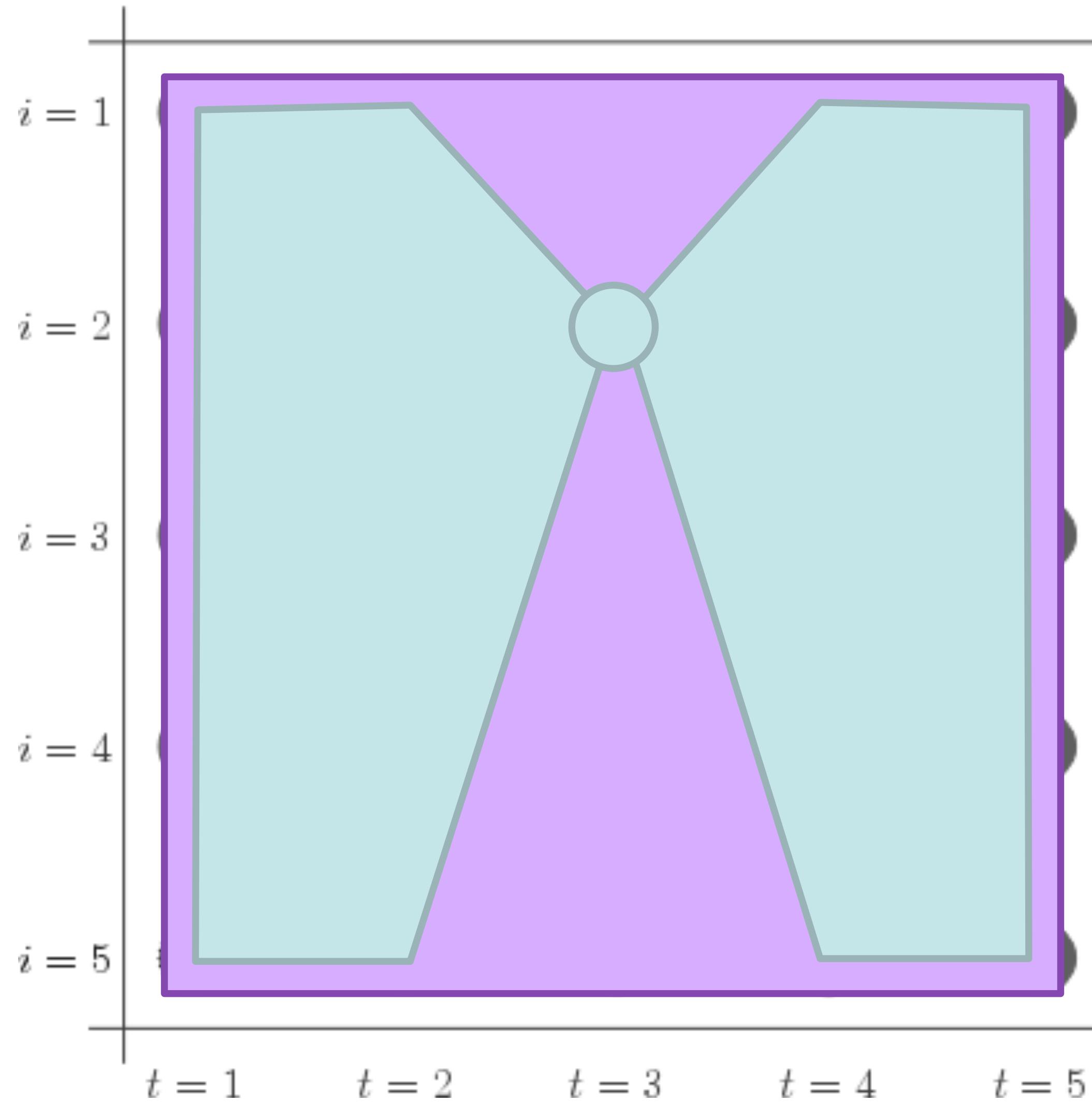
Forward-Backward Algorithm



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

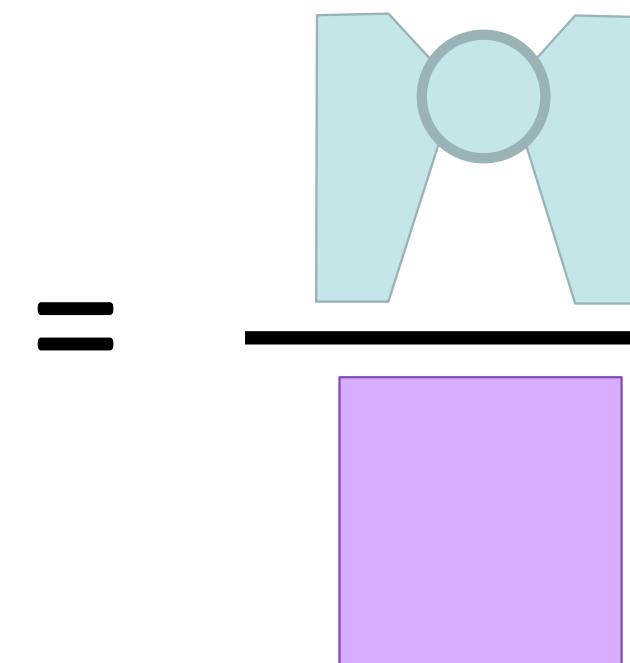
$$\frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$

Forward-Backward Algorithm



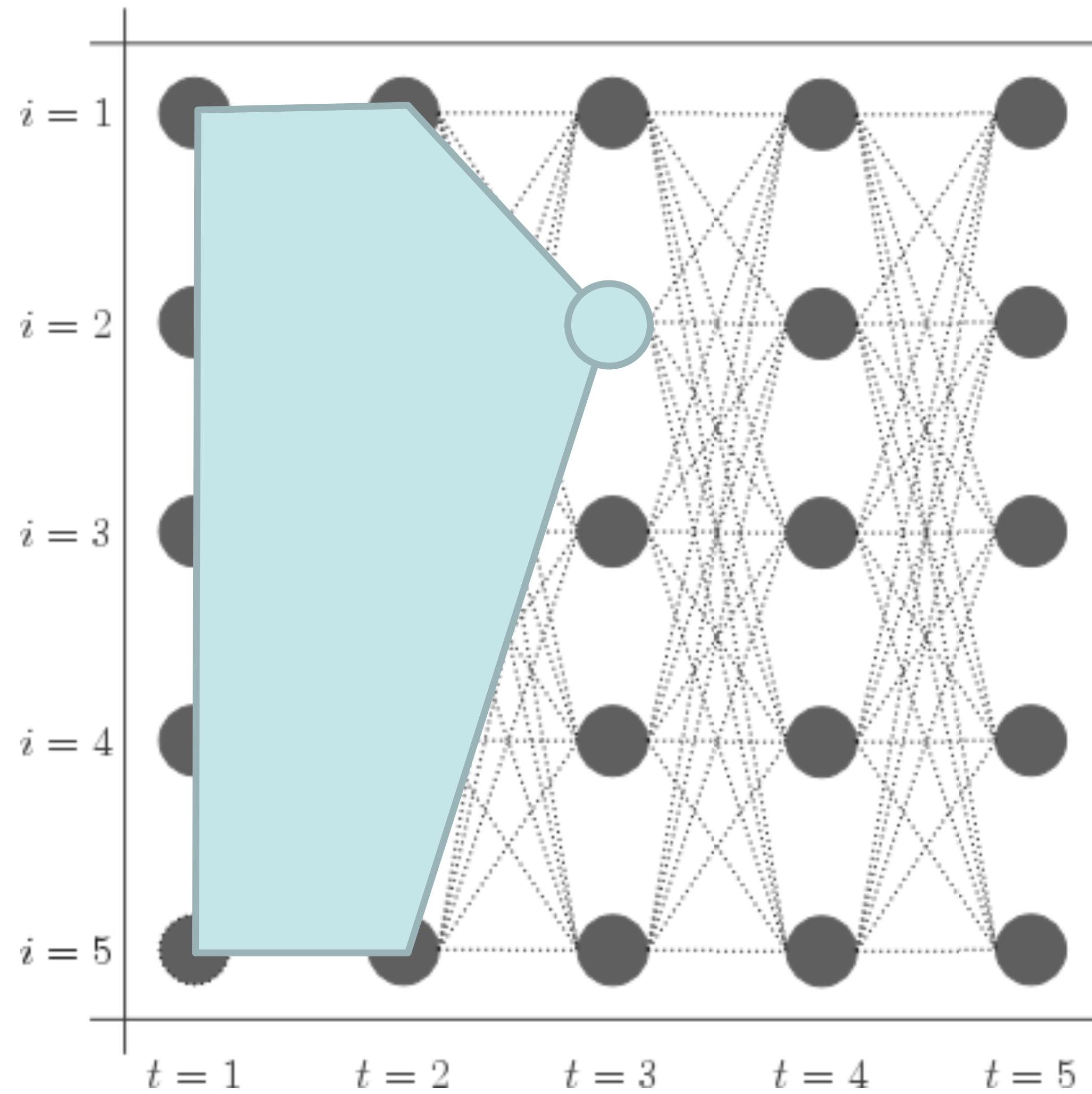
$$P(y_3 = 2 | \mathbf{x}) =$$

$$\frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$



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

Forward-Backward Algorithm



► Initial:

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

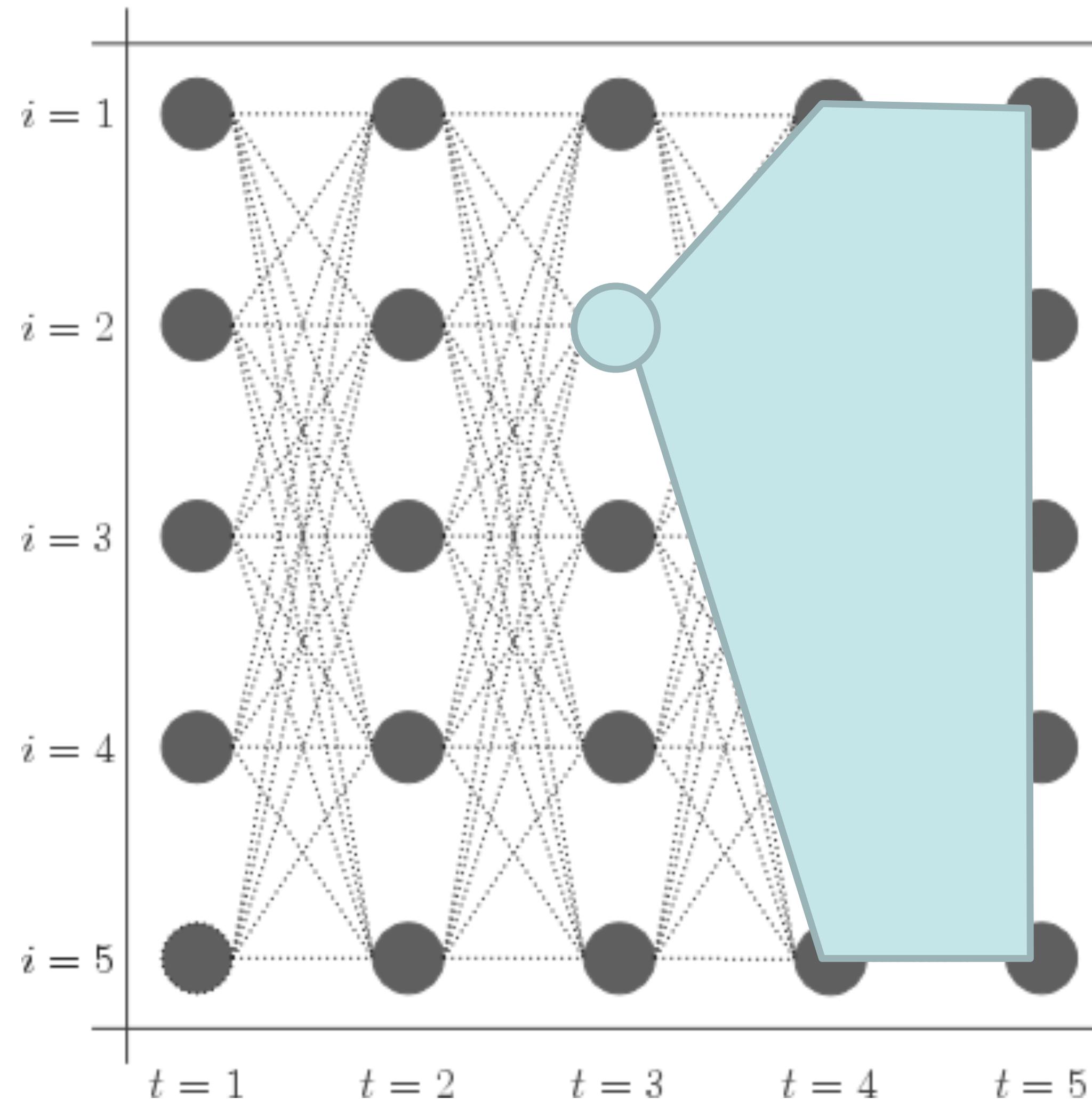
► Recurrence:

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

► Same as Viterbi but summing instead of maxing!

► These quantities get very small!
Store everything as log probabilities

Forward-Backward Algorithm



► Initial:

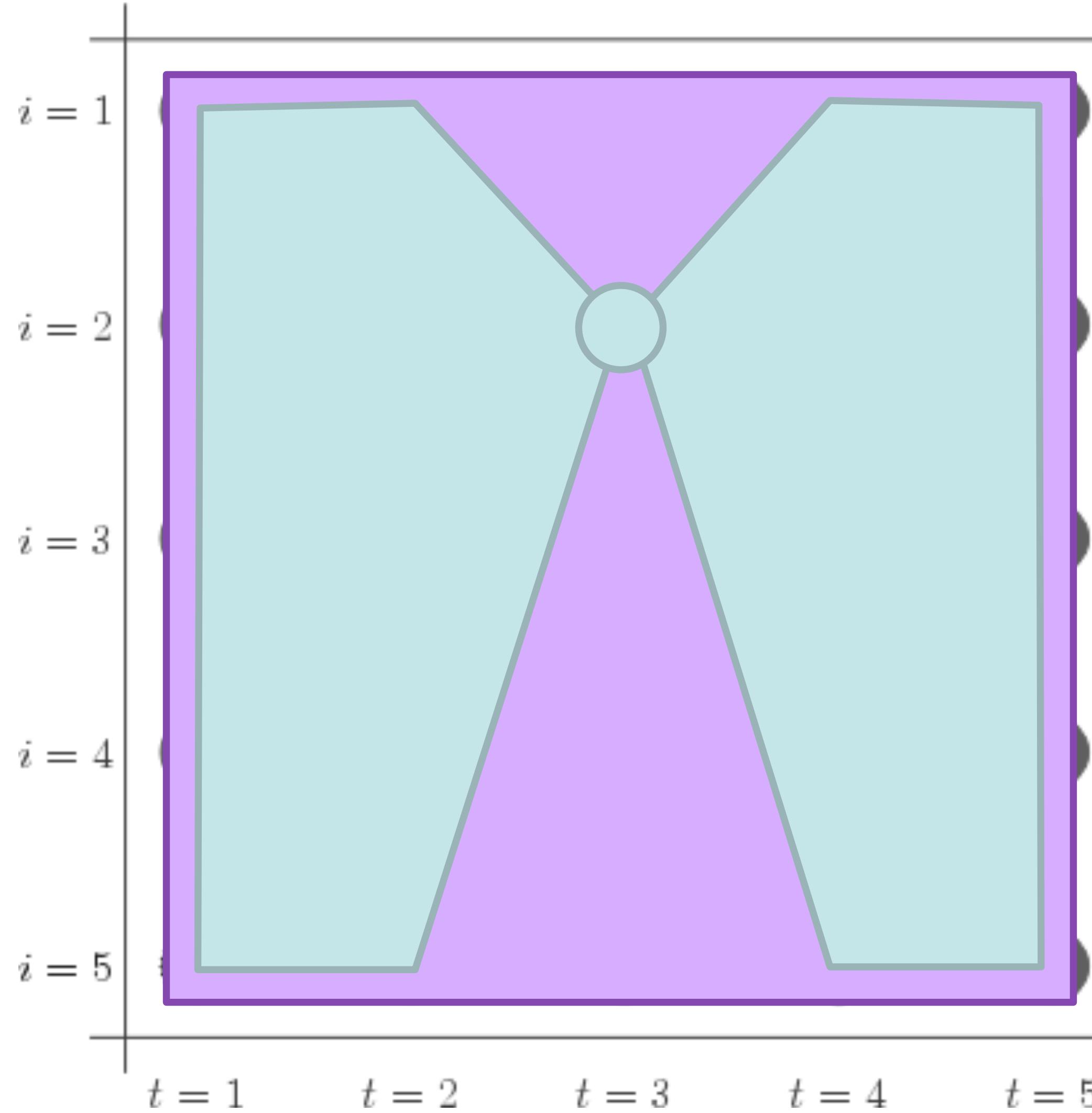
$$\beta_n(s) = 1$$

► 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



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

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

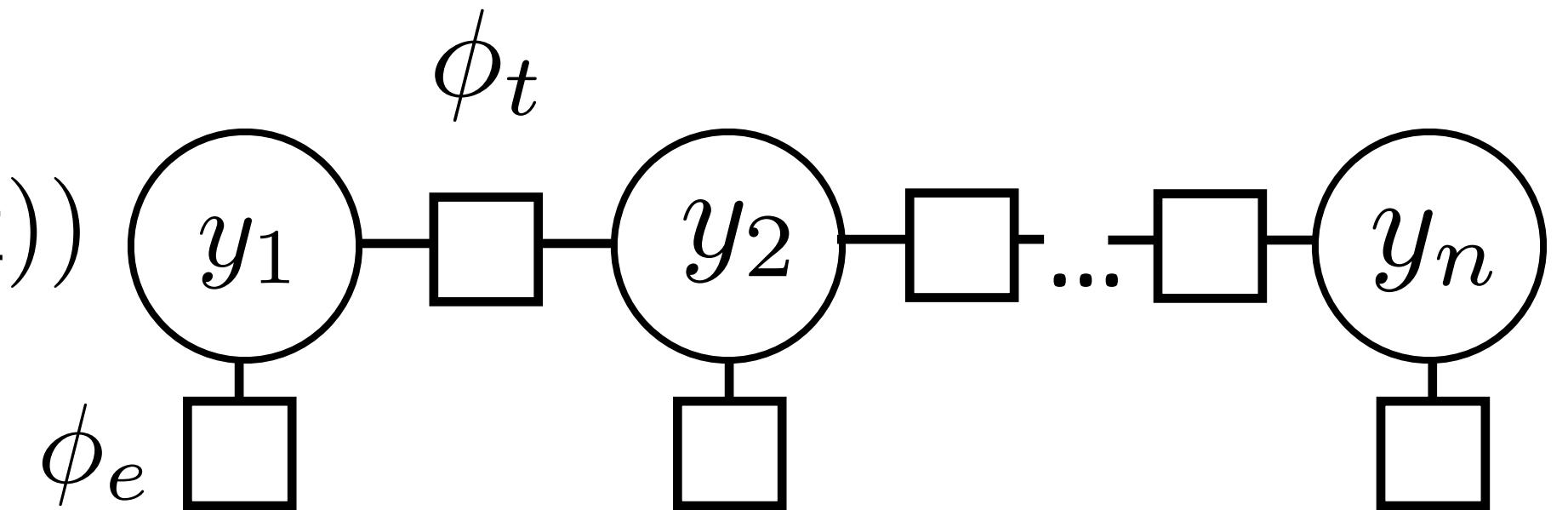
$$\beta_n(s) = 1$$

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

► 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} !)

HMM

$P(x_i | y_i), P(y_i | y_{i-1})$

Model parameter (usually multinomial distribution)

CRF

Undefined (model is by definition conditioned on \mathbf{x})

$P(y_i | \mathbf{x}), P(y_{i-1}, y_i | \mathbf{x})$

Inferred quantity from forward-backward

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 (e.g., 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: 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 $\sum_s \text{forward}_i(s) \text{backward}_i(s)$ 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? Can you fit a small training set?
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

Structured Perceptron

Structured Perceptron

- ▶ Structured Perceptron Update:

$$\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} w^\top f(x, y) \quad \xleftarrow{\text{Viterbi Algorithm}}$$

$$w = w + f(x, y^*) - f(x, \hat{y})$$

- ▶ Compare to gradient of CRF:

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

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

Replaces Expectation
With argmax

NER

NER

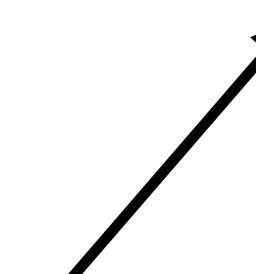
- ▶ CRF with lexical features can get around 85 F1 on this problem
- ▶ Other pieces of information that many systems capture
- ▶ World knowledge:

The delegation met the president at the airport, **Tanjug** said.

Tanjug

From Wikipedia, the free encyclopedia

Tanjug (/tʌnjʊg/) ([Serbian Cyrillic](#): Танјуг) is a Serbian state news agency based in [Belgrade](#).^[2]



Nonlocal Features

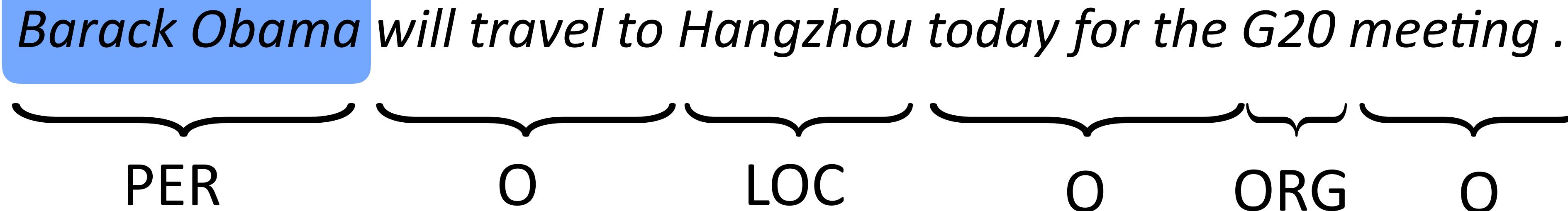
The news agency Tanjug reported on the outcome of the meeting.

ORG?
PER?

The delegation met the president at the airport, Tanjug said.

- More complex factor graph structures can let you capture this, or just decode sentences in order and use features on previous sentences

Semi-Markov Models



- ▶ Chunk-level prediction rather than token-level BIO
- ▶ y is a set of touching spans of the sentence
- ▶ Pros: features can look at whole span at once
- ▶ Cons: there's an extra factor of n in the dynamic programs

Evaluating NER

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

- ▶ Prediction of all Os still gets 66% accuracy on this example!
- ▶ What we really want to know: how many named entity *chunk* predictions did we get right?
- ▶ Precision: of the ones we predicted, how many are right?
- ▶ Recall: of the gold named entities, how many did we find?
- ▶ F-measure: harmonic mean of these two

How well do NER systems do?

	System	Resources Used	F_1	
+	LBJ-NER	Wikipedia, Nonlocal Features, Word-class Model	90.80	Lample et al. (2016)
-	(Suzuki and Isozaki, 2008)	Semi-supervised on 1G-word unlabeled data	89.92	LSTM-CRF (no char) 90.20 LSTM-CRF 90.94
-	(Ando and Zhang, 2005)	Semi-supervised on 27M-word unlabeled data	89.31	S-LSTM (no char) 87.96 S-LSTM 90.33
-	(Kazama and Torisawa, 2007a)	Wikipedia	88.02	BiLSTM-CRF + ELMo 92.2
-	(Krishnan and Manning, 2006)	Non-local Features	87.24	Peters et al. (2018)
-	(Kazama and Torisawa, 2007b)	Non-local Features	87.17	Devlin et al. (2019) Fine-tuning approach
+	(Finkel et al., 2005)	Non-local Features	86.86	$\text{BERT}_{\text{LARGE}}$ 96.6 92.8 $\text{BERT}_{\text{BASE}}$ 96.4 92.4

Structured SVM

- ▶ CRF: $\log P(\mathbf{y}|\mathbf{x}) \propto \sum_{i=2}^n w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^n w^\top f_e(x_i, y_i)$
- ▶ We can formulate an SVM using the same features

$$w^\top f(\mathbf{x}, \mathbf{y}) = \sum_{i=2}^n w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^n w^\top f_e(x_i, y_i)$$

$$\begin{aligned} & \text{Minimize } \lambda \|w\|_2^2 + \sum_{j=1}^m \xi_j \\ \text{s.t. } & \forall j \quad \xi_j \geq 0 \\ & \forall j \forall \mathbf{y} \in \mathcal{Y} \quad w^\top f(\mathbf{x}_j, \mathbf{y}_j^*) \geq w^\top f(\mathbf{x}_j, \mathbf{y}) + \ell(\mathbf{y}, \mathbf{y}_j^*) - \xi_j \end{aligned}$$

Structured SVM

$$w^\top f(\mathbf{x}, \mathbf{y}) = \sum_{i=2}^n w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^n w^\top f_e(x_i, y_i)$$

$$\begin{aligned} & \text{Minimize } \lambda \|w\|_2^2 + \sum_{j=1}^m \xi_j \\ \text{s.t. } & \forall j \quad \xi_j \geq 0 \\ & \forall j \forall \mathbf{y} \in \mathcal{Y} \quad w^\top f(\mathbf{x}_j, \mathbf{y}_j^*) \geq w^\top f(\mathbf{x}_j, \mathbf{y}) + \ell(\mathbf{y}, \mathbf{y}_j^*) - \xi_j \end{aligned}$$


- ▶ Exponentially large state space! Use Viterbi for loss-augmented decode
- ▶ Same as normal Viterbi but boost wrong labels' scores by 1 (if using Hamming loss)
- ▶ Only need Viterbi, not forward-backward...hmm...

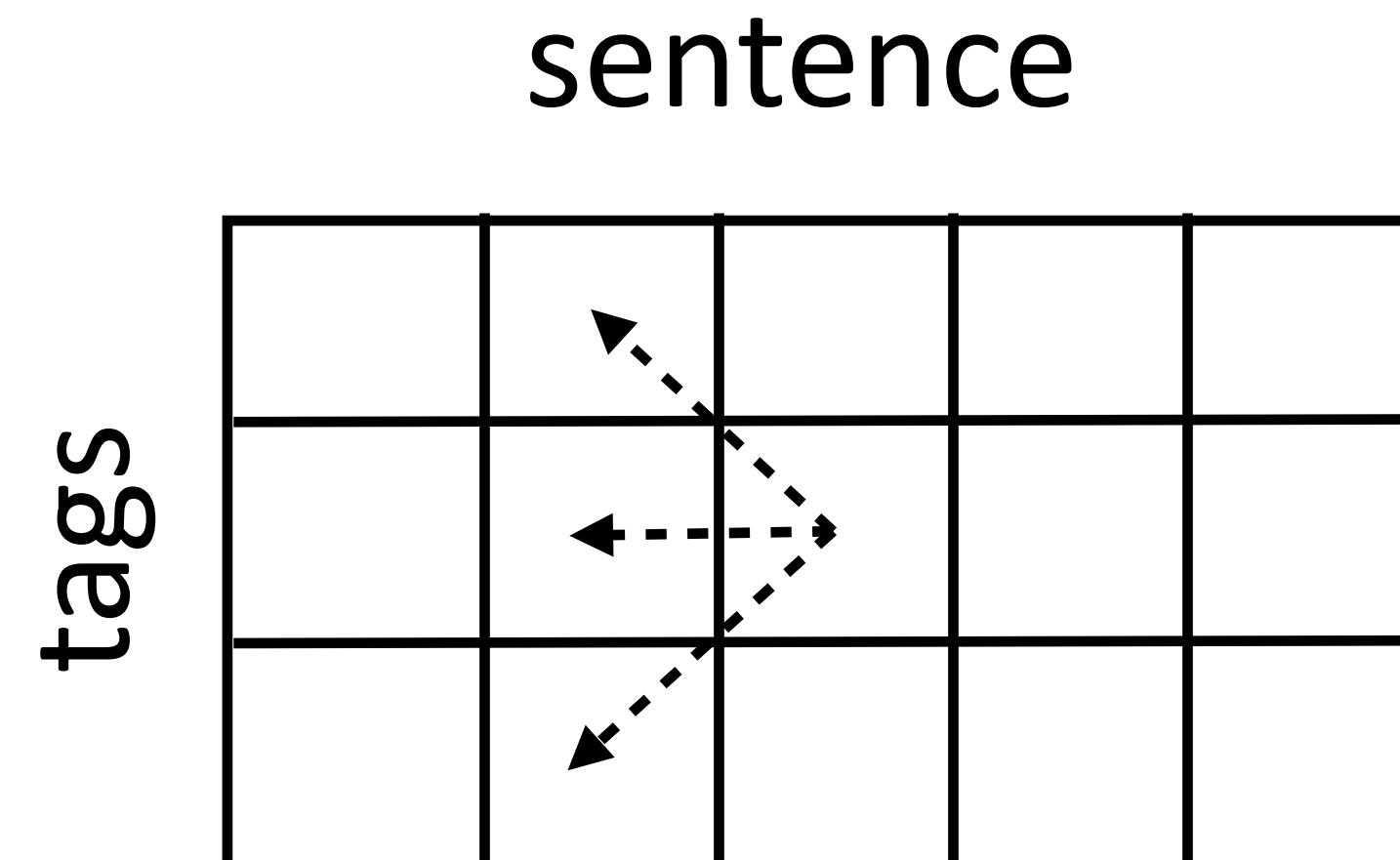
Beam Search

Viterbi Time Complexity

VBD	VB				
VBN	VBZ	VBP	VBZ		
NNP	NNS	NN	NNS	CD	NN

Fed raises interest rates 0.5 percent

- ▶ n word sentence, s tags to consider – what is the time complexity?



- ▶ $O(ns^2)$ – s is ~40 for POS, n is ~20

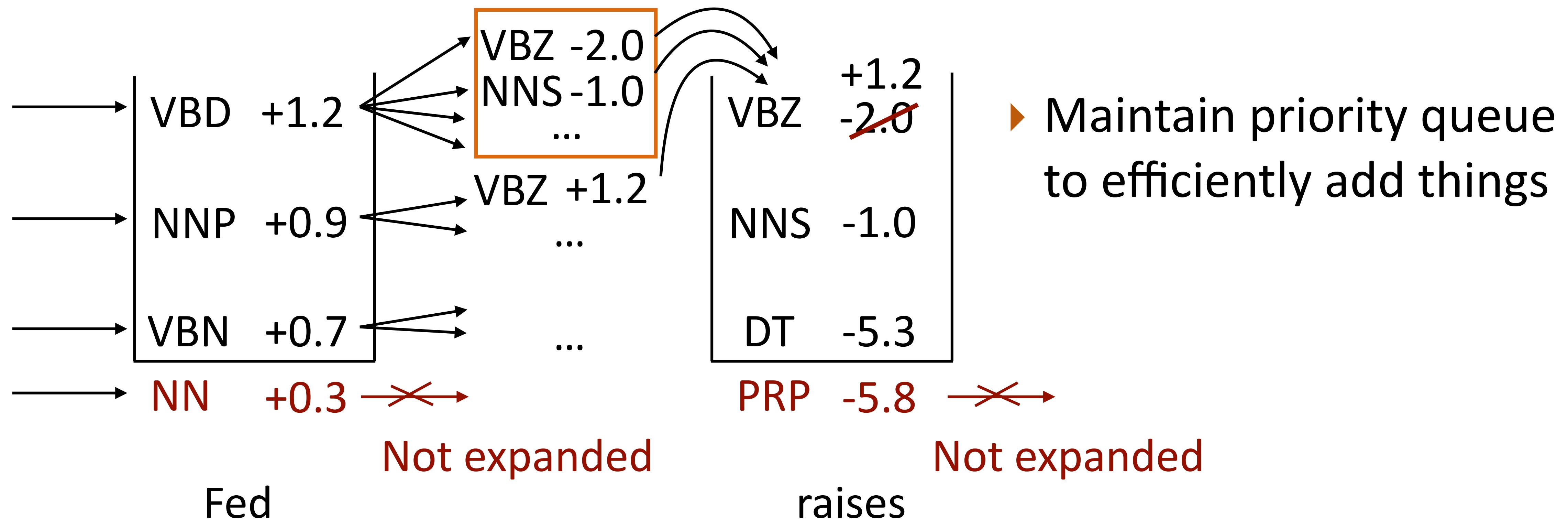
Viterbi Time Complexity

VBD	VB				
VBN	VBZ	VBP	VBZ		
NNP	NNS	NN	NNS	CD	NN
Fed raises interest rates 0.5 percent					

- ▶ Many tags are totally implausible
- ▶ Can any of these be:
 - ▶ Determiners?
 - ▶ Prepositions?
 - ▶ Adjectives?
- ▶ Features quickly eliminate many outcomes from consideration — don't need to consider these going forward

Beam Search

- ▶ Maintain a beam of k plausible states at the current timestep
- ▶ Expand all states, only keep k top hypotheses at new timestep



- ▶ Beam size of k , time complexity $O(nks \log(ks))$

How good is beam search?

- ▶ $k=1$: greedy search
- ▶ Choosing beam size:
 - ▶ 2 is usually better than 1
 - ▶ Usually don't use larger than 50
 - ▶ Depends on problem structure
- ▶ If beam search is much faster than computing full sums, can use structured perceptron instead of CRFs
- ▶ Very similar to structured SVM