THE OHIO STATE UNIVERSITY

CSE 5525: Foundations of Speech and Language Processing

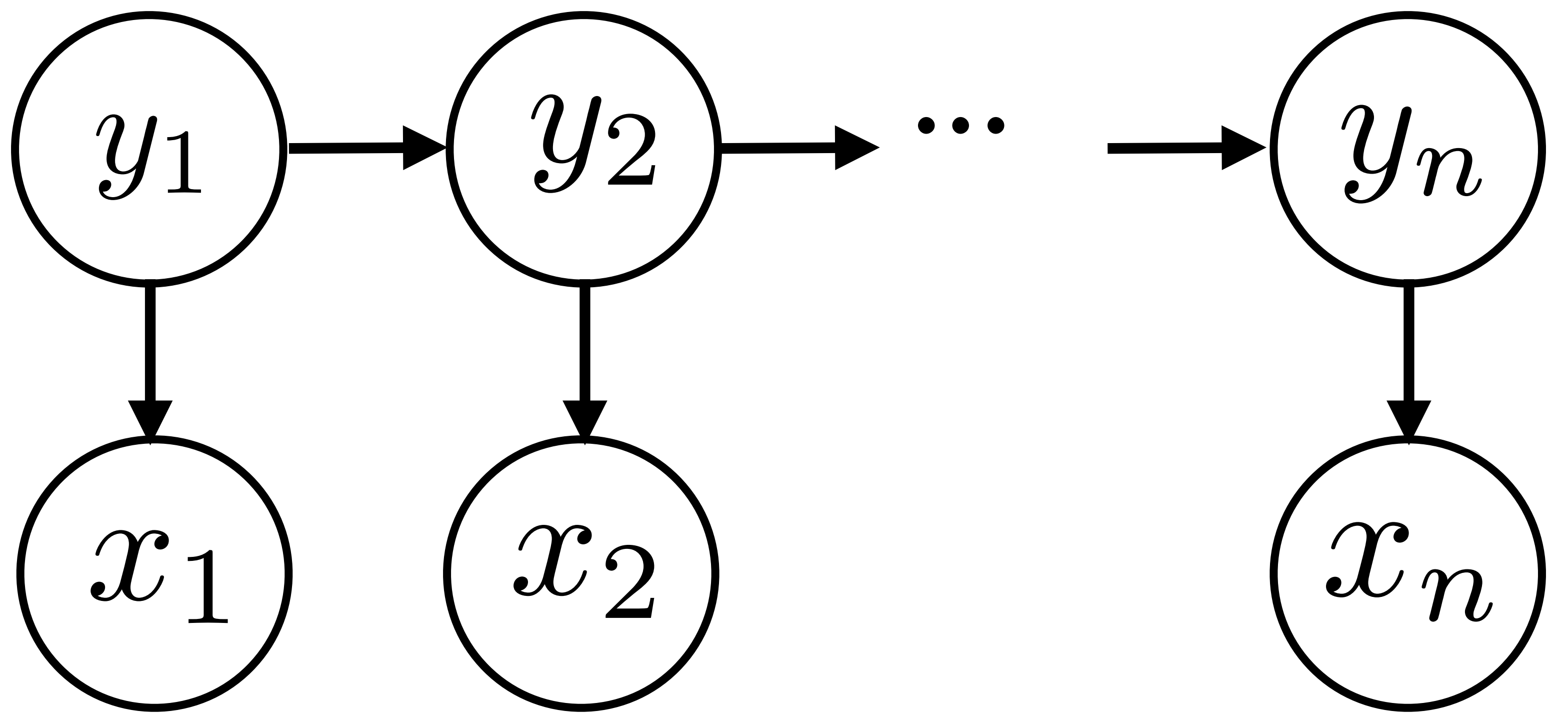
Conditional Random Fields (CRFs)

Huan Sun (CSE@OSU)

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

Recall: HMMs

- Observations O (= input x) Output Q (sequence of states) = labels 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: $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}} \frac{P(\mathbf{y}, \mathbf{x})}{\cancel{P(\mathbf{x})}}$
- Viterbi: $\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1}) P(x_i|s) \text{score}_{i-1}(y_{i-1})$

Recall: Viterbi Algorithm

- Initialization

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

- Recursion

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

- Termination

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

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*

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

slide credit: Ray Mooney

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)

Read more in lecture notes:

(1) Sutton CRFs 2.3, 2.6.1, Eisenstein 7.5, 8.3,
(2) Wallach CRFs tutorial, Illinois NER
especially, (1)

Helpful blog: https://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/

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**

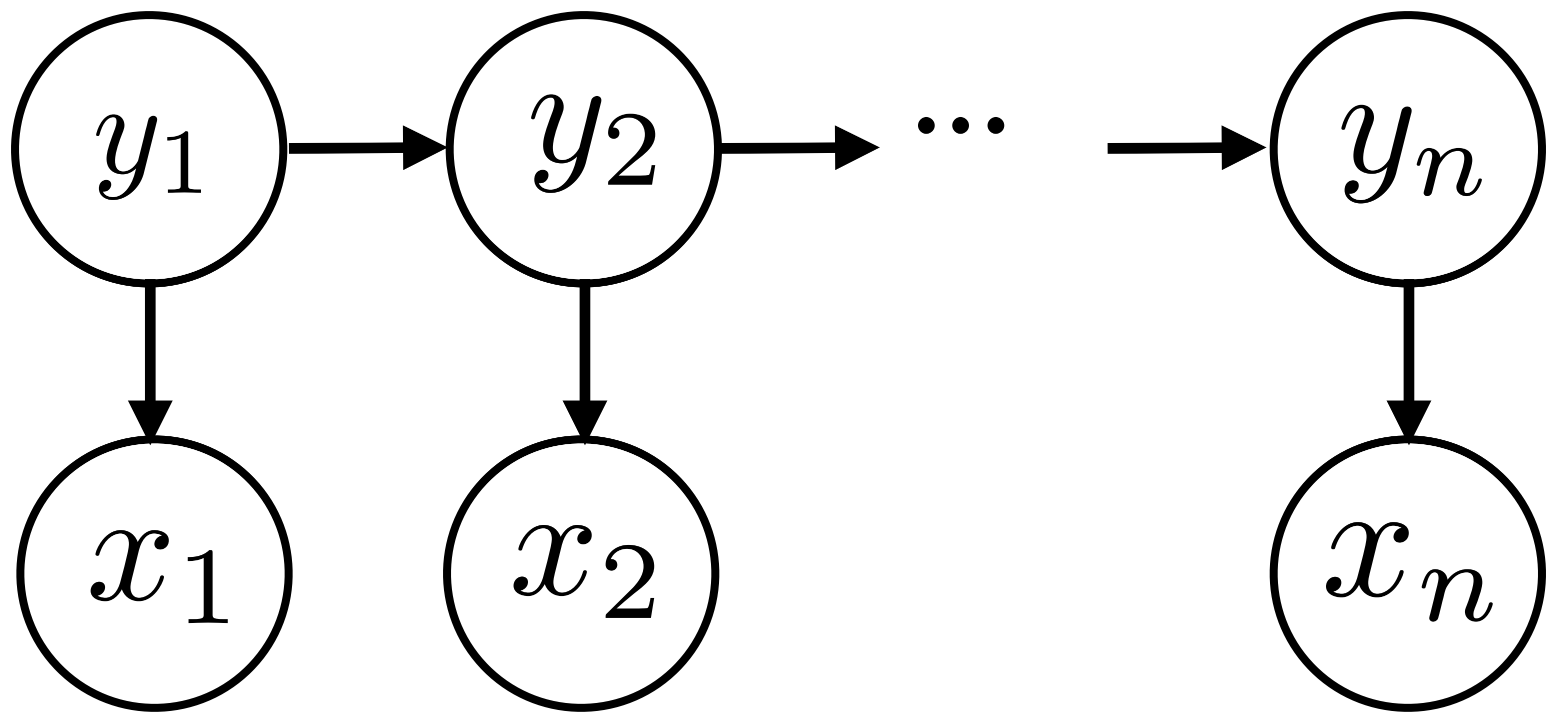
Posn_in_sentence=1st & **Label=B-PER**

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

…

HMMs, Formally

- HMMs are expressible as Bayes nets (factor graphs)



- This reflects the following decomposition:

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

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

HMMs vs. CRFs

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

any real-valued scoring function of its arguments

HMMs vs. 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: 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

any real-valued scoring function of its arguments

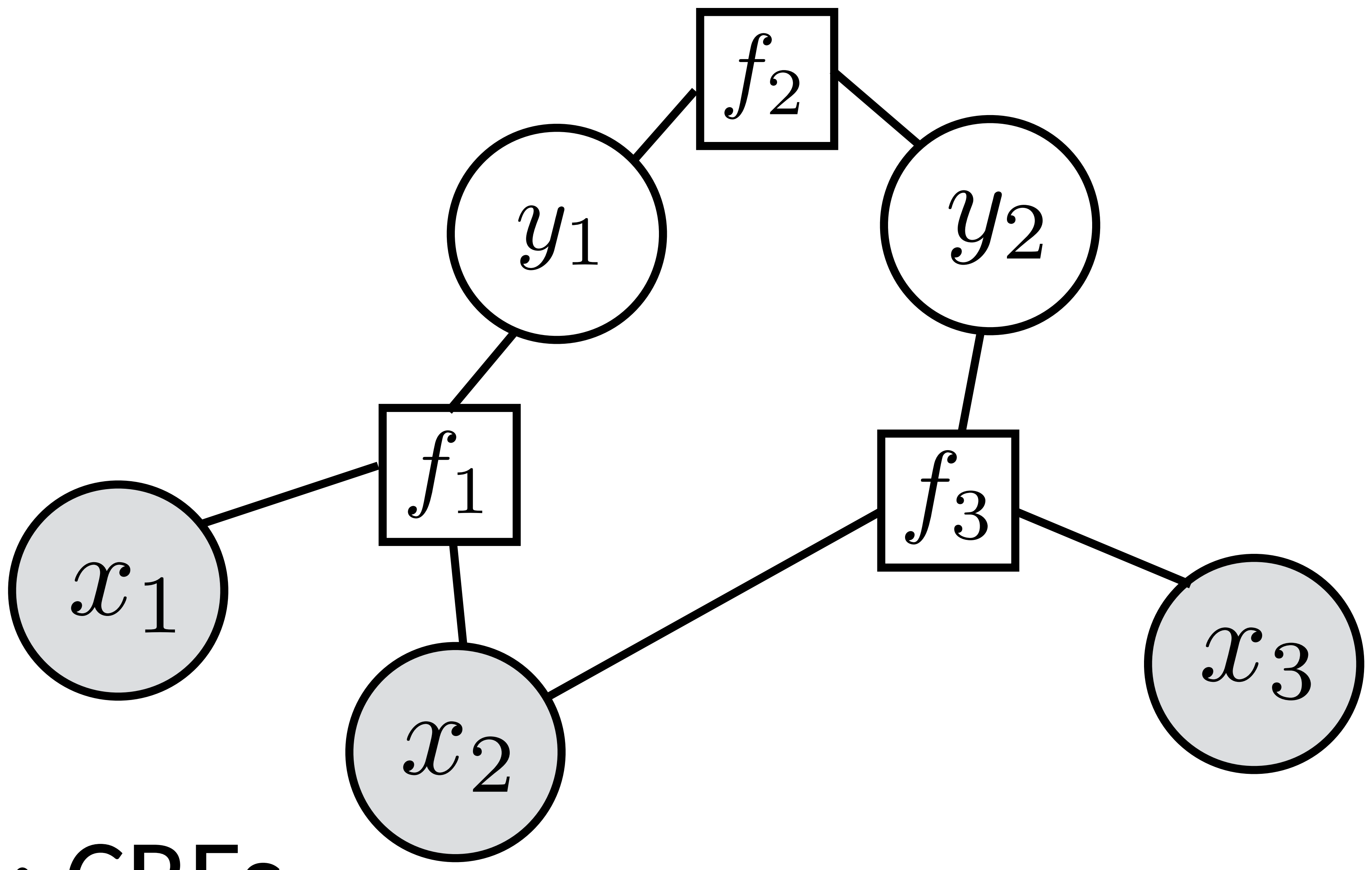
- Special case: linear feature-based potentials $\phi_k(\mathbf{x}, \mathbf{y}) = w^\top f_k(\mathbf{x}, \mathbf{y})$

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

- Looks like our single weight vector multiclass logistic regression model

HMMs vs. CRFs

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

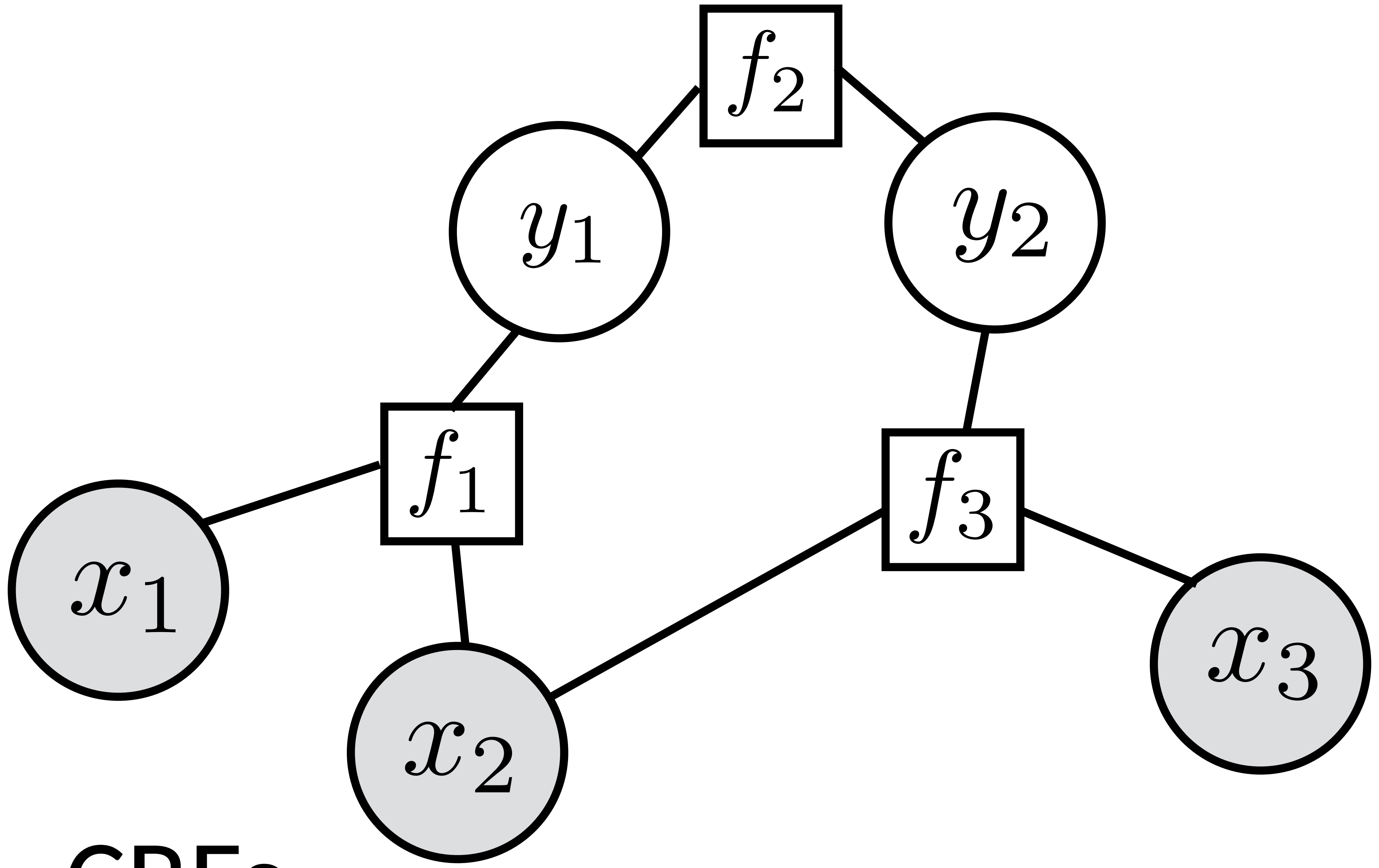


- Conditional model: x's are observed
- Naive Bayes : logistic regression :: HMMs : CRFs
  local vs. global normalization <-> generative vs. discriminative

(locally normalized discriminative models do exist (MEMMs))

HMMs vs. CRFs

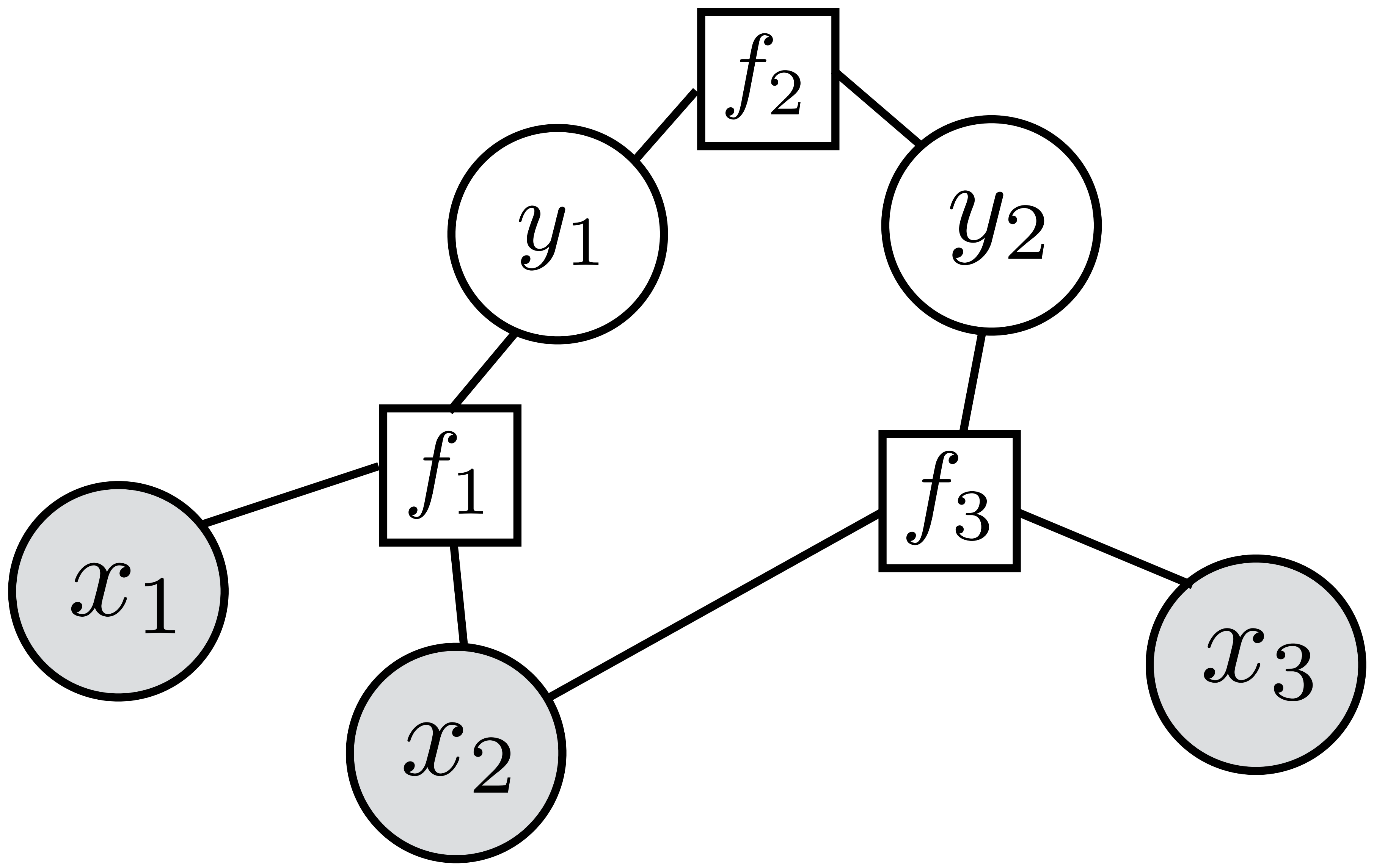
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{k=1}^{n} w^\top f_k(\mathbf{x}, \mathbf{y})\right)$$



- Conditional model: x's are observed
- Naive Bayes : logistic regression :: HMMs : CRFs
  local vs. global normalization <-> 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: $\mathbf{y}_{\text{best}} = \text{argmax}_{\mathbf{y}'}\exp\left(\sum_{k=1}^{n} w^\top f_k(\mathbf{x},\mathbf{y}')\right)$
- If 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)

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

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

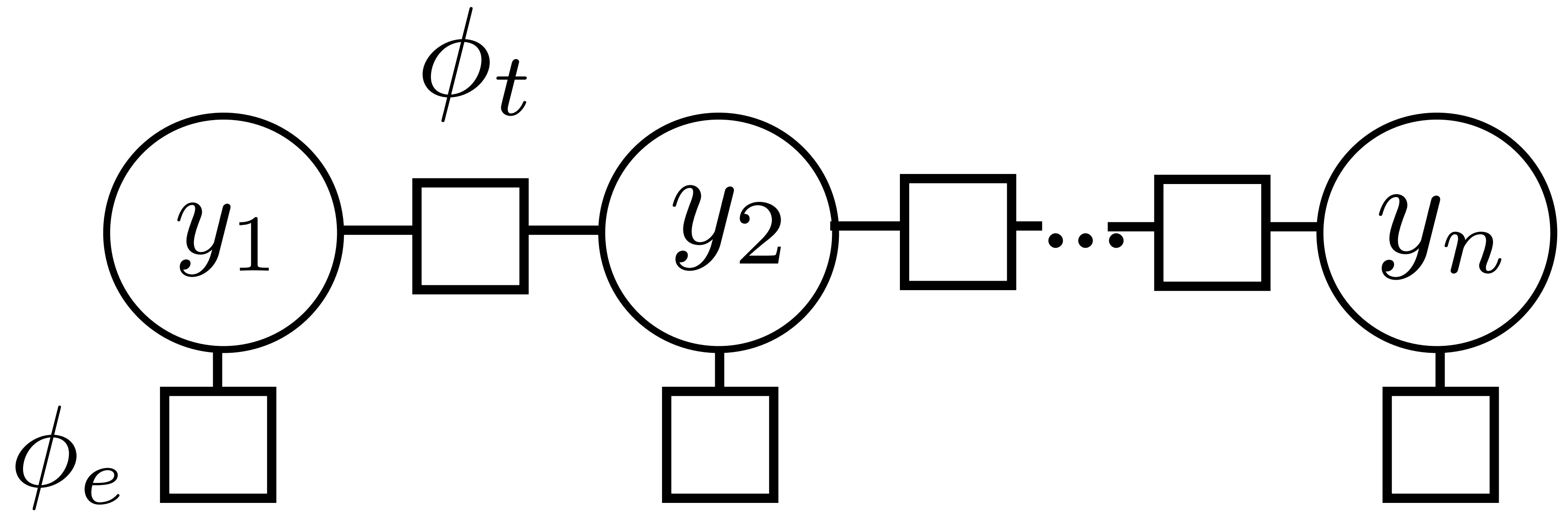
Linear-chain CRFs (Sutton, Section 2.3)

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 **x** from the **factor graph** entirely (implicit), but every Phi_e feature function connects to it



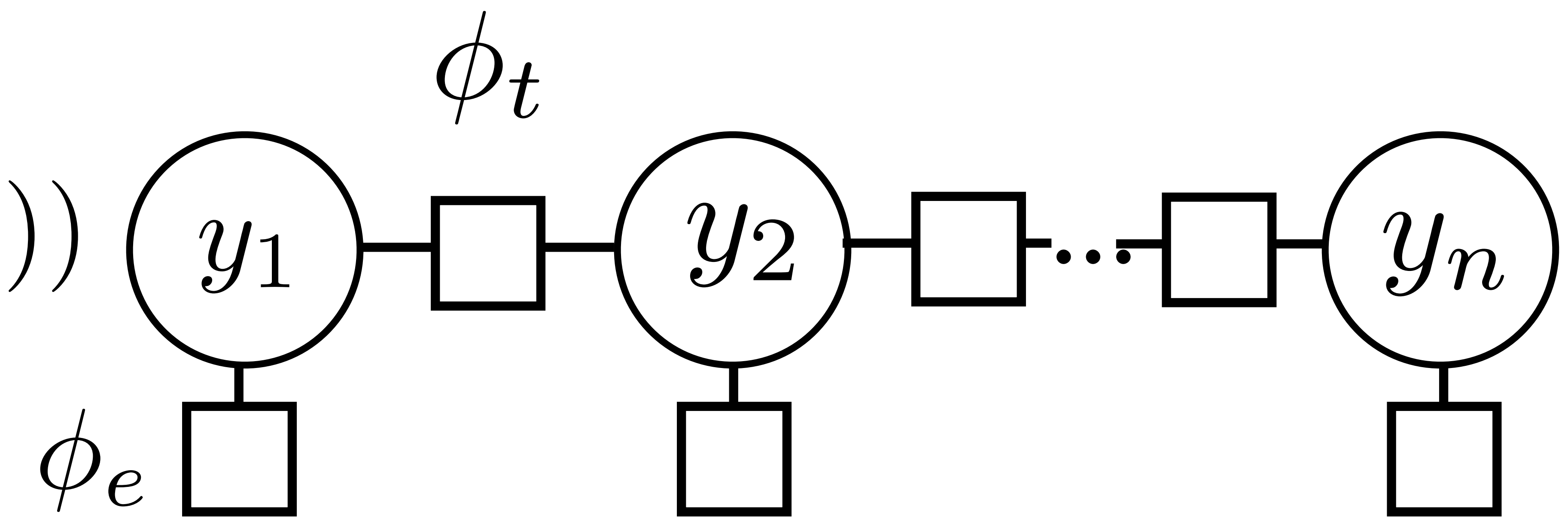
- Two types of factors: *transitions* ϕ_t (look at adjacent y's, but not x) and *emissions* ϕ_e (look at y and all of **x**)

Linear-chain CRFs (Sutton, Section 2.3)

Features for NER

Feature Functions

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



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

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

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

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

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

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

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

Ind[B-LOC & Prev word = *to*]

Emission Features for NER

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

LOC

Leicestershire is a nice place to visit...

PER

Leonardo DiCaprio won an award...

LOC

I took a vacation to Boston

ORG

Apple released a new version...

LOC PER

Texas governor Greg Abbott said

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 (i.e, a list of known entity names)

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

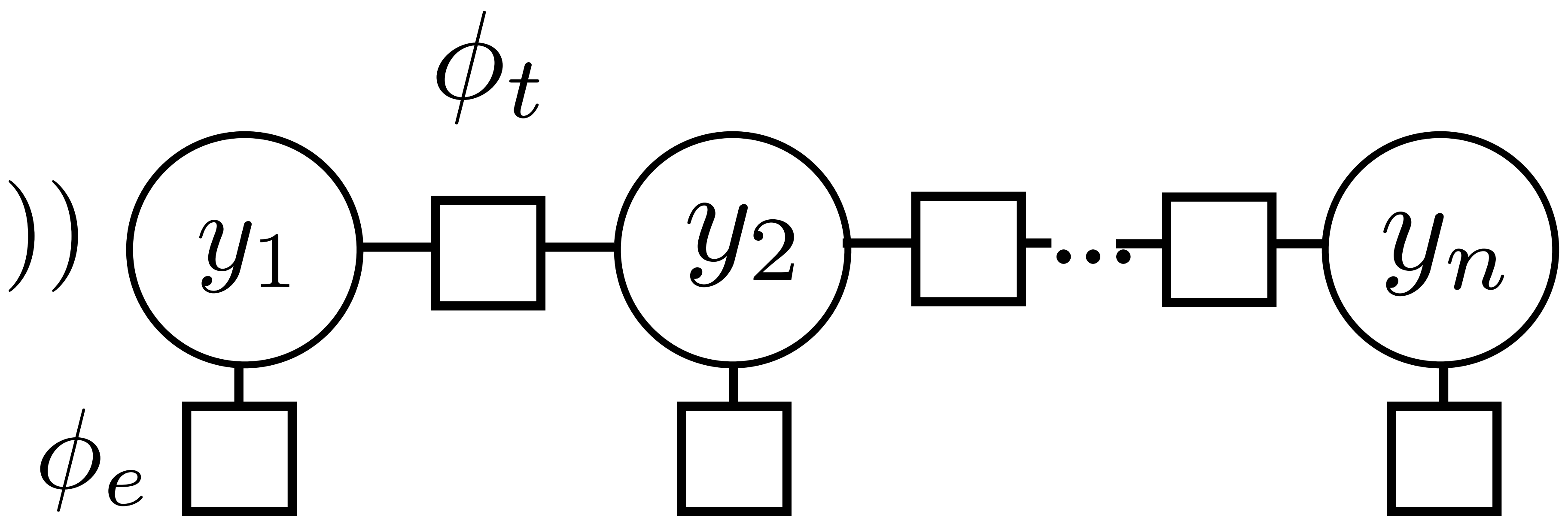
- Learning

__t:__ transition feature functions; *__e:__* emission feature functions

Inference and Learning in CRFs

Computing (arg)maxes

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

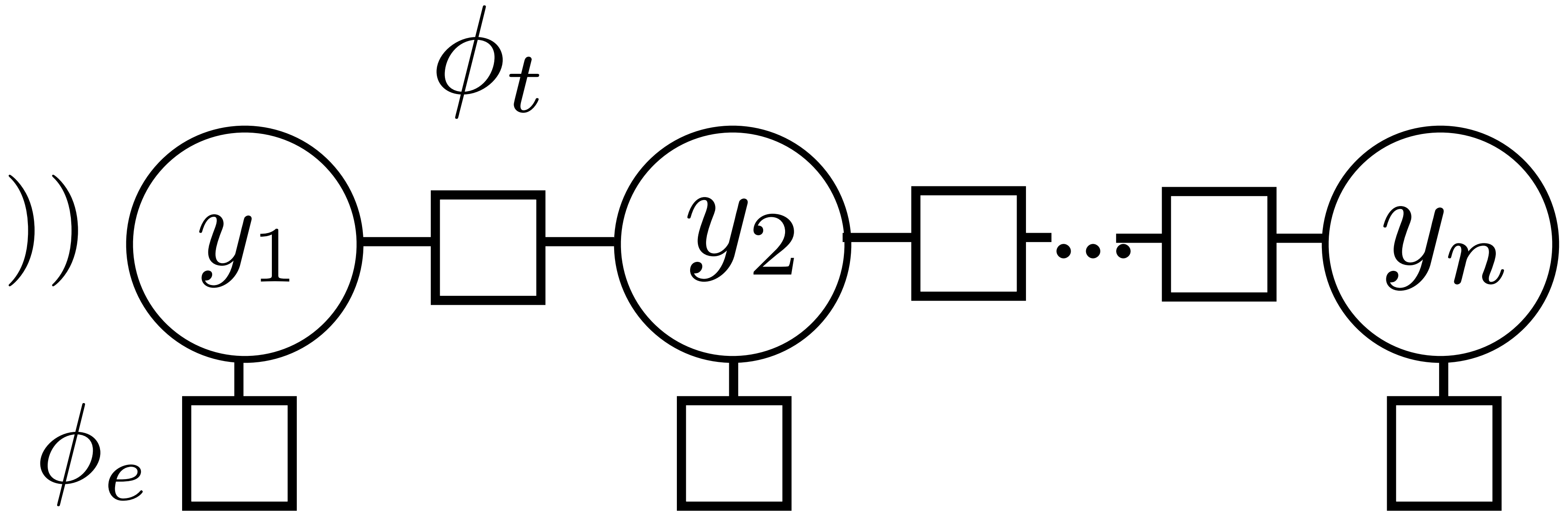


- $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$: Inference can use Viterbi exactly as in HMM case

$$\max_{y_1,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)} e^{\phi_e(y_n,n,\mathbf{x})} \ldots e^{\phi_e(y_2,2,\mathbf{x})} e^{\phi_t(y_1,y_2)} e^{\phi_e(y_1,1,\mathbf{x})}$$

Computing (arg)maxes

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



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

$$\max_{y_1,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)} e^{\phi_e(y_n,n,\mathbf{x})} \ldots e^{\phi_e(y_2,2,\mathbf{x})} e^{\phi_t(y_1,y_2)} e^{\phi_e(y_1,1,\mathbf{x})}$$

$$= \max_{y_2,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)} e^{\phi_e(y_n,n,\mathbf{x})} \ldots e^{\phi_e(y_2,2,\mathbf{x})} \boxed{\max_{y_1}} e^{\phi_t(y_1,y_2)} \underbrace{e^{\phi_e(y_1,1,\mathbf{x})}}$$

$$= \max_{y_3,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)} e^{\phi_e(y_n,n,\mathbf{x})} \ldots \max_{y_2} e^{\phi_t(y_2,y_3)} \underbrace{e^{\phi_e(y_2,2,\mathbf{x})} \max_{y_1} e^{\phi_t(y_1,y_2)} \text{score}_1(y_1)}$$

- $\exp(\phi_t(y_{i-1}, y_i))$ and $\exp(\phi_e(y_i, i, \mathbf{x}))$ play the role of the P's (in HMM decoding) now, same dynamic program (consider using *log* for computations).

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(**y**|**x**) from Viterbi (Review HMM, if needed)
- 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 (LR):

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

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

intractable!

Review how we computed gradients for LR, if needed

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

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

$$= \sum_{i=1}^{n}\sum_{s} \boxed{P(y_i = s|\mathbf{x})} f_e(s, i, \mathbf{x})$$

Forward-Backward Algorithm

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

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

Forward-Backward Algorithm

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

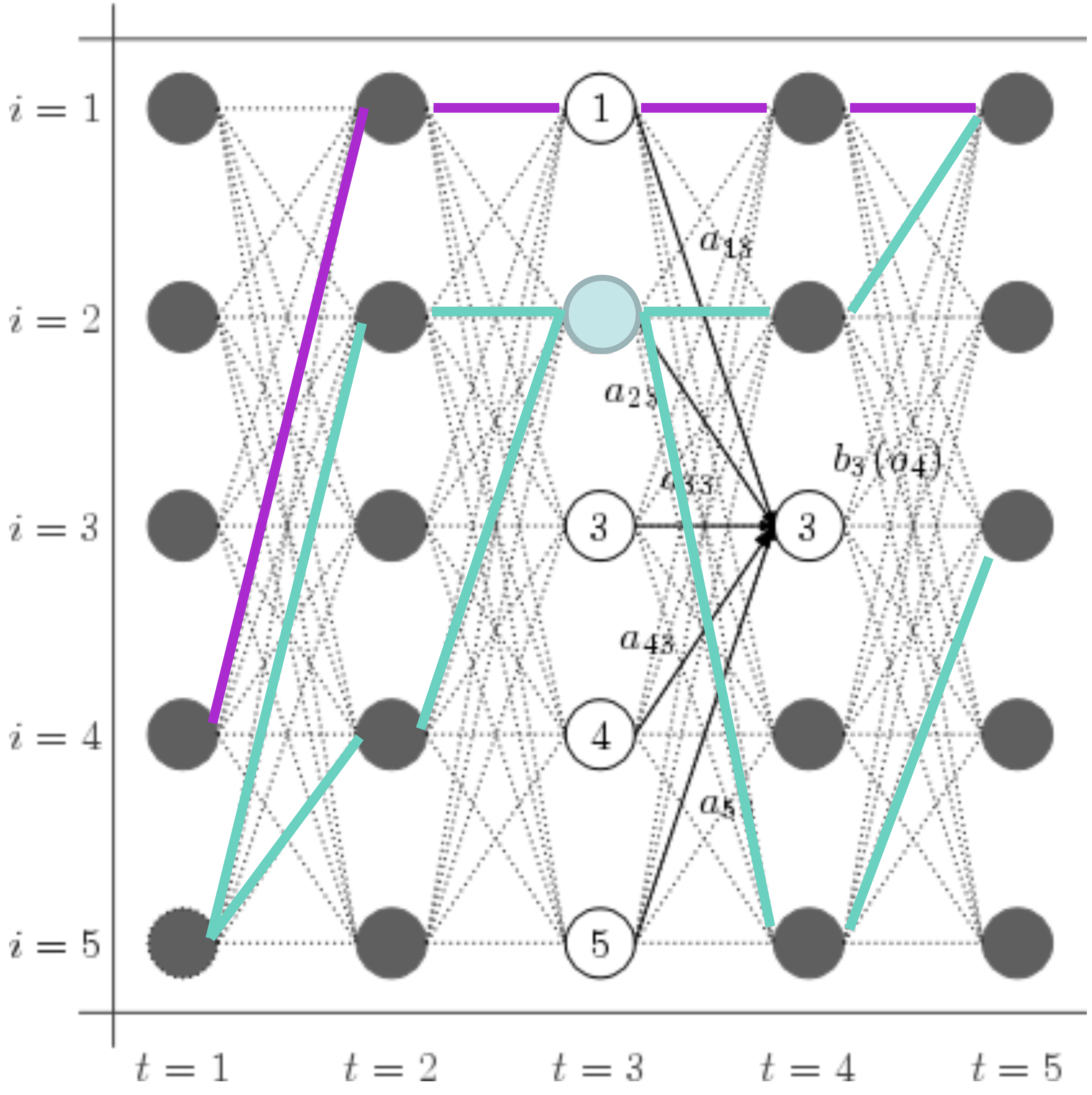
$$P(y_i = s|\mathbf{x}) = \sum_{y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_n} P(\mathbf{y}|\mathbf{x})$$

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

  and also review the forward algorithm in HMM

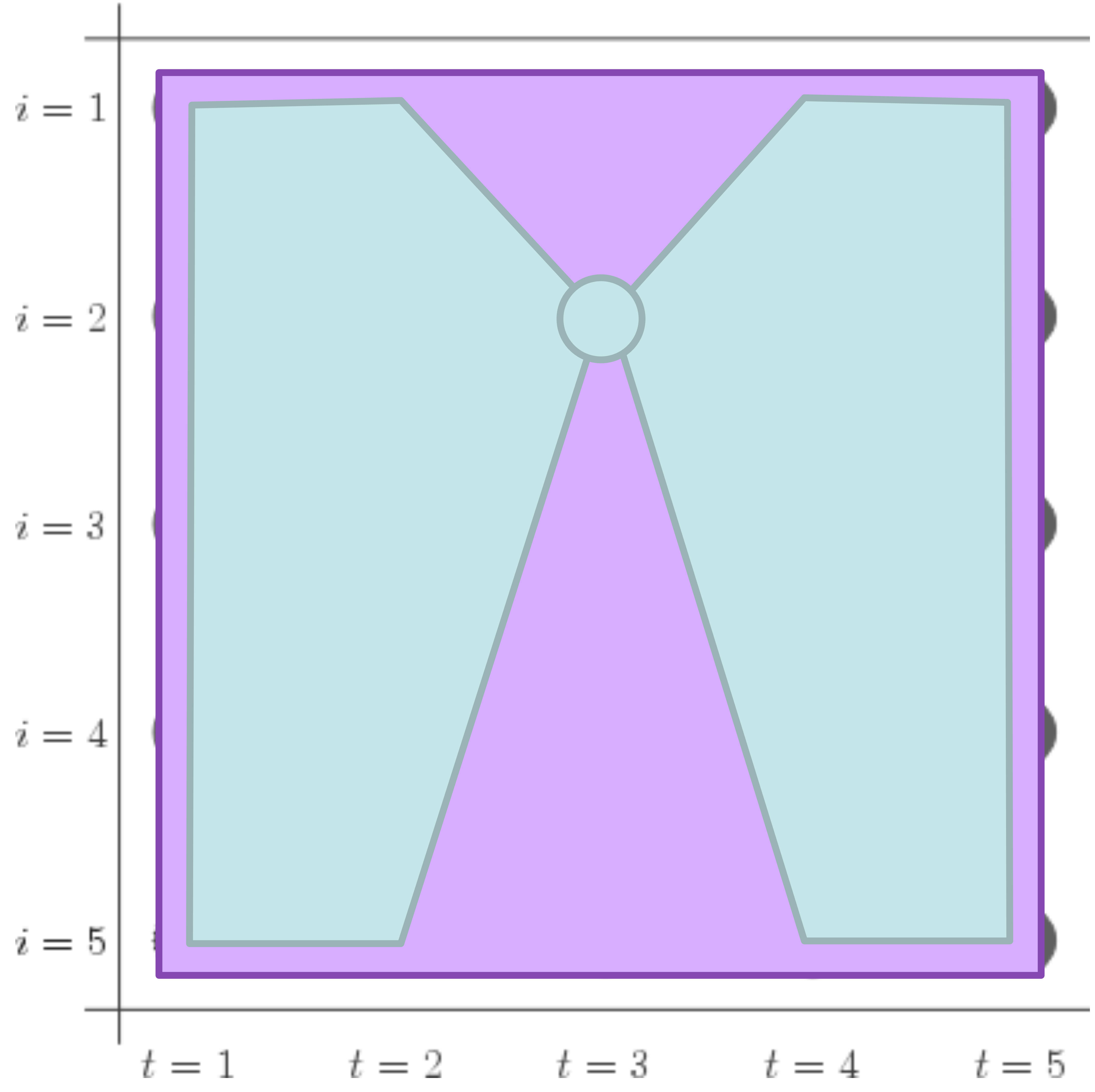
- Can compute marginals with dynamic programming as well using **forward-backward**

Forward-Backward Algorithm

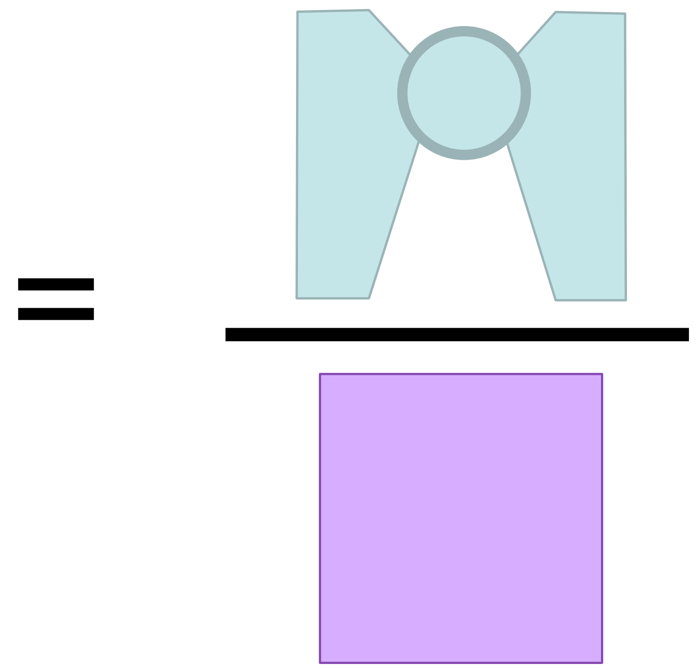


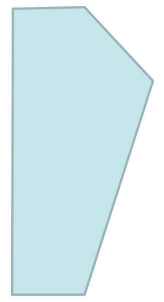
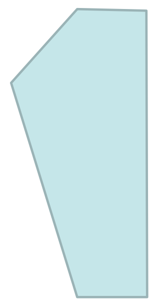
$$P(y_3 = 2|\mathbf{x}) = \frac{\text{score of all paths through state 2 at time 3}}{\text{score of all paths}}$$

Forward-Backward Algorithm



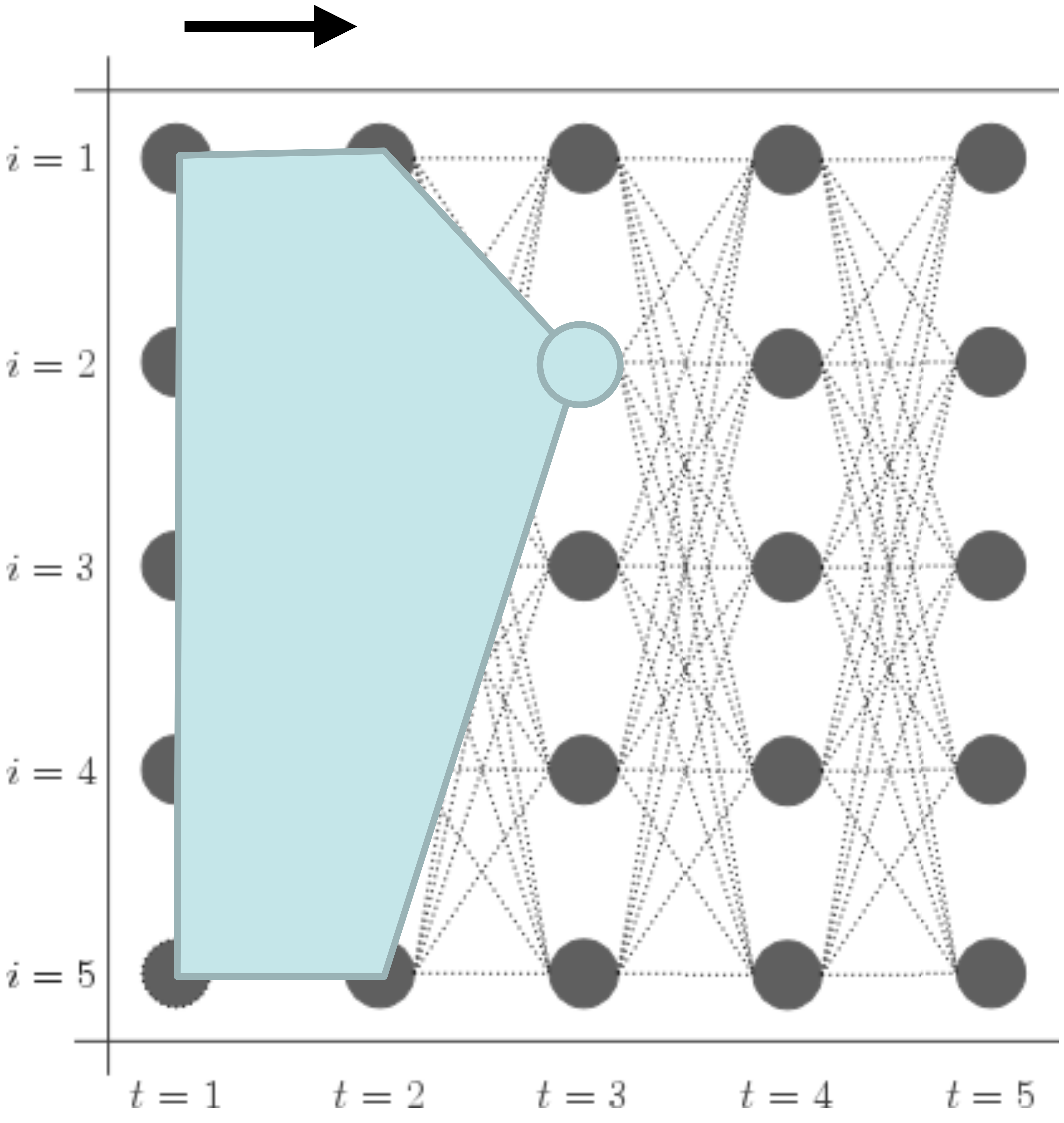
$$P(y_3 = 2|\mathbf{x}) = \frac{\text{score of all paths through state 2 at time 3}}{\text{score of all paths}}$$



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

slide credit: Dan Klein

Forward-Backward Algorithm



Forward vectors

- Initial:

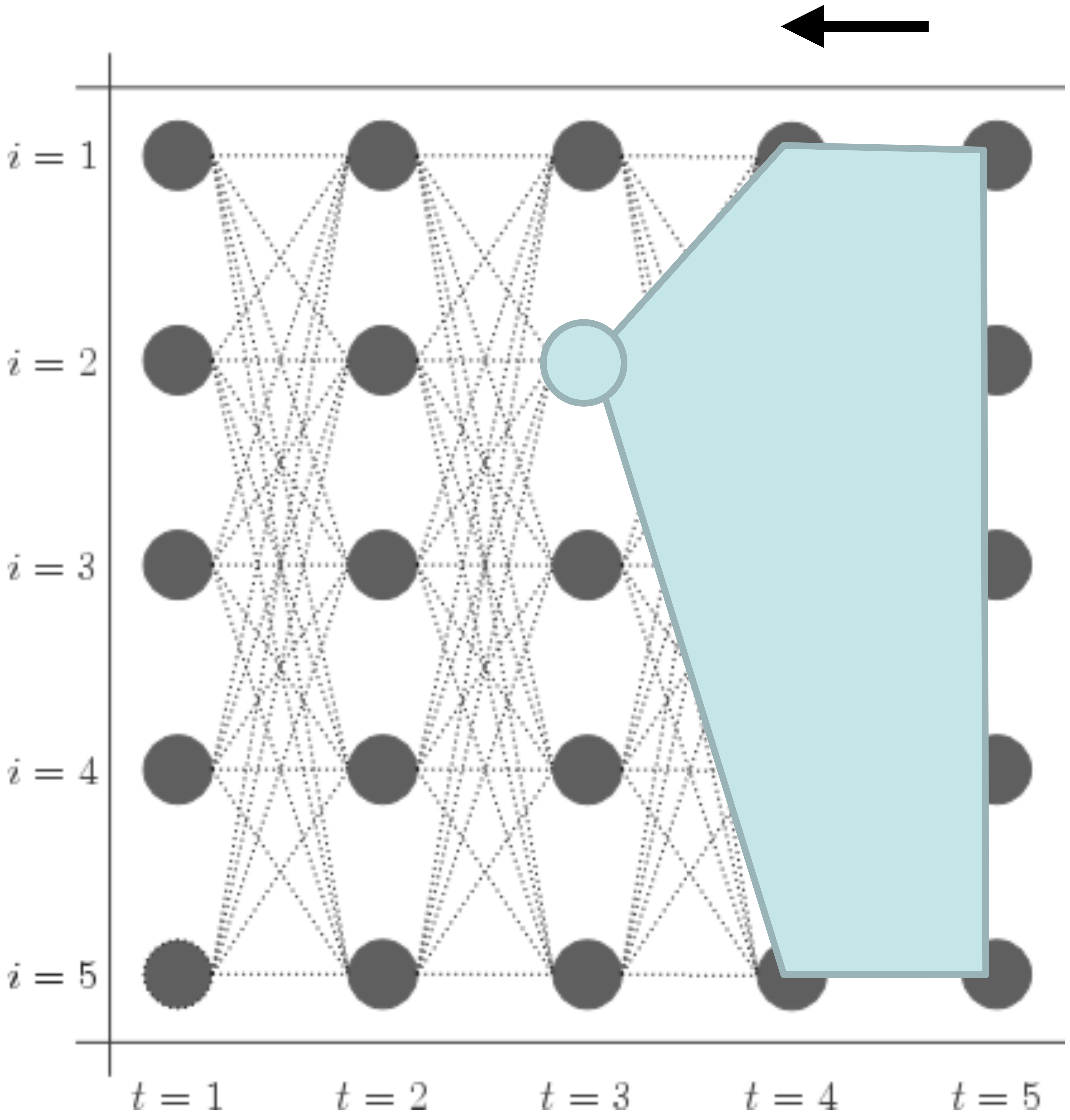
$$\alpha_1(s) = \exp(\phi_e(s, 1, \mathbf{x}))$$

- 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 in logarithm

Forward-Backward Algorithm



Backward vectors

- Initial:

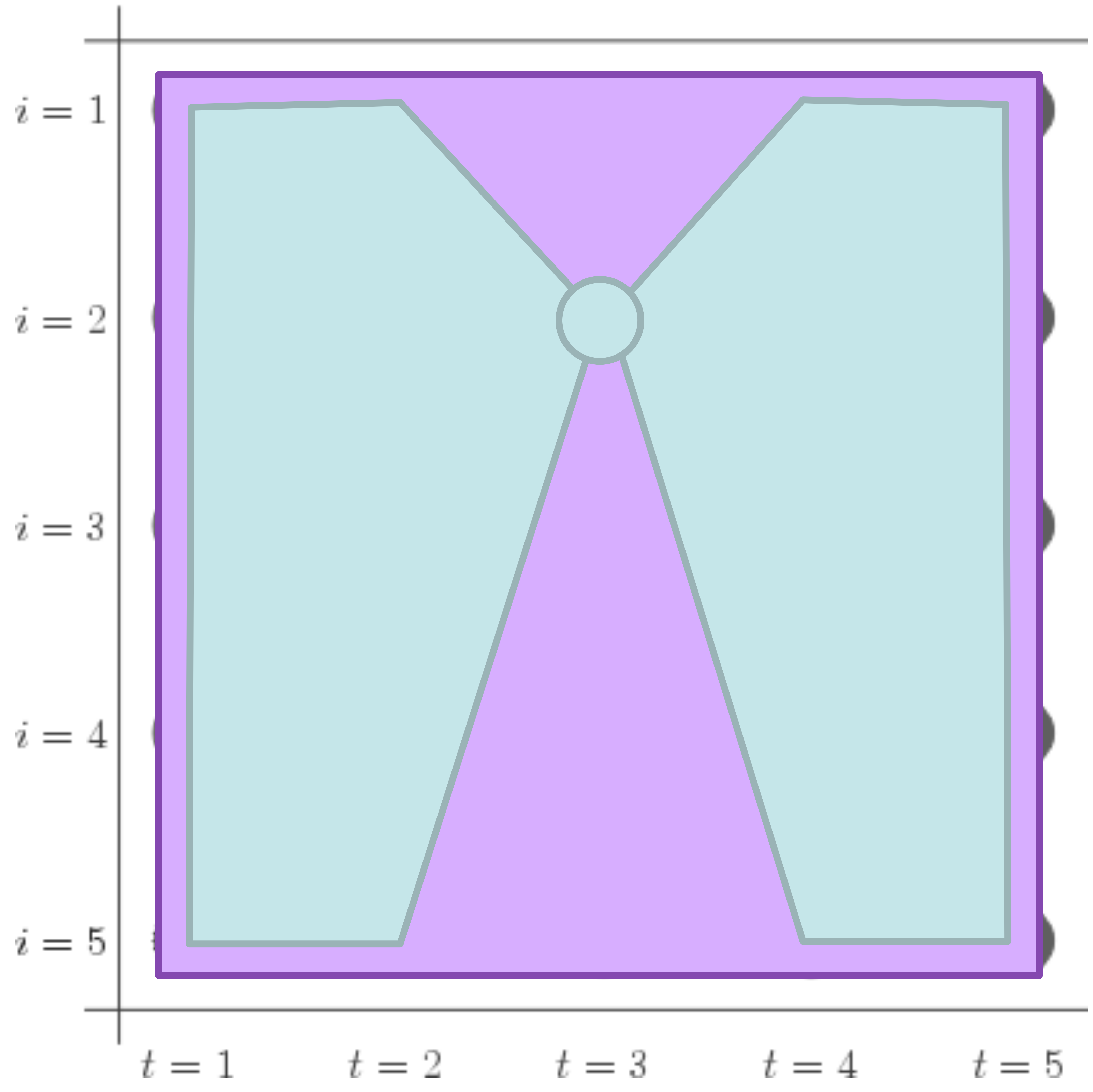
$$\beta_n(s) = 1 \qquad \text{(last time step n)}$$

- 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 with forward vectors in the previous page: Consider emission for the *next* time step (*not* current one; because the current one has been considered in forward pass)

Forward-Backward Algorithm



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

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

$$\beta_n(s) = 1$$

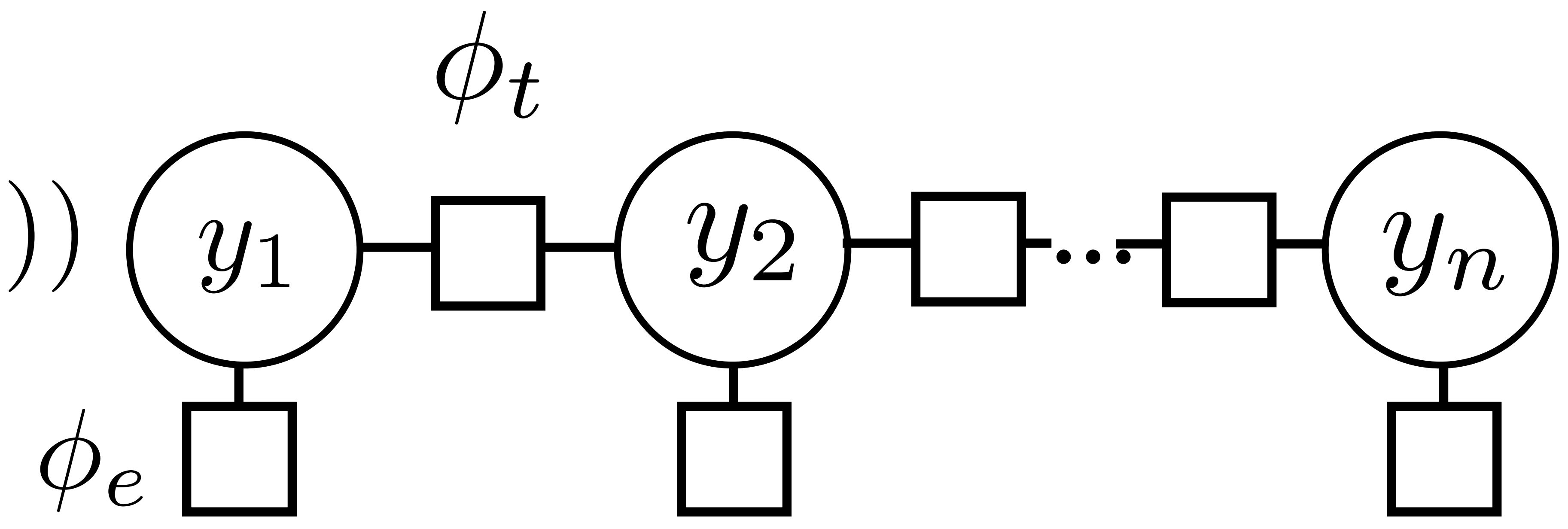
$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \exp(\phi_e(s_{t+1}, t+1, \mathbf{x})) \exp(\phi_t(s_t, s_{t+1}))$$

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

- Does this explain why beta is what it is?
- What is the denominator here?

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(**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(**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 **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

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

(similar to logistic regression)

- 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 such as 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(**y**|**x**) from Viterbi

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

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

Pseudocode

for each epoch

    for each example

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

        compute potentials phi (page 18) based on features + weights

        compute marginal probabilities with forward-backward

        accumulate gradient over all emissions and transitions

- **In your HW2, you don't need transition phi, but can just enforce constraints such as B-PER -> I-ORG being illegal (see more in HW2 instructions)**

Implementation Tips for CRFs

Offline reading, if interested

- 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

Offline reading, if interested

- 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? 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